پیش گزارش آزسیستم دیجتال ۲

سهیل مهریزی ۹۸۱۲۱۰۱۱۰
آرش جوادی 980121010019

۱. شیفت رجیستر یونیورسال
۲. D FLIP FLOP
۳. 2_1_MULTIPLEXER

شیفت رجیستر یونیورسال :

با استفاده از شیفت رجیستر یونیورسال میتوان سه عملیات

- Parallel Load .
- Shift to Right .
- Shift to Left .

را تنها با تغییر در ورودی سلکتور انجام داد :

Selector:

00 ⟶ Parallel Load

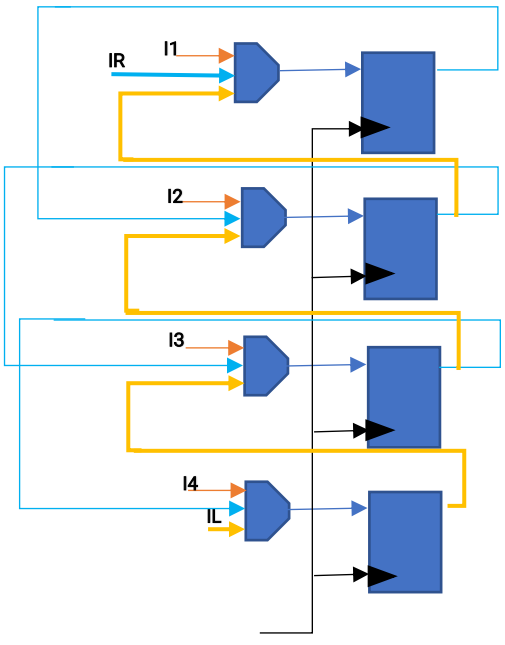
01 ⟶ Shift to Right

10 ⟶ Shift to Left

شیفت رجیستر یونیورسال را میتوان با دوعنصر اساسی فلیپ فلاپ و مالتی پلکسر طراحی کرد .

در این مدار ما یک شیفت رجیستر یونیورسال ۴ بیتی داریم که با مالتی پلکسر های ۱-۴ نوع عملیات آن تعیین میشود .

در این مدار خروجی مالتی پلکسرها به ورودی فلیپ فلاپ های نوع D متصل میشود و با سلکتور های مالتی پلکسر نوع عملیات مشخص میشود ، در خط سلکتور های 0 ورودی های بعدی فیلپ فلاپ قرار میگیرد ، در خط ۱ خروجی فلیپ فلاپ قبلی (در چپ ترین عنصر ، ورودی Instance right)(Shift to right) و در خط سلکتور ۲ خروجی فلیپ فلاپ بعدی (راست ترین عنصر : shift to left(Instance Left



D_Flip Flop Simulation

در این بخش از آزمایش ، شبیه سازی ماژول فلیپ فلاپ نوع D را داریم با استفاده ازVHDL .

شبیه سازی به دوبخش تقسیم میشود .

معماری ماژول که بصورت رفتاری طراحی شده است (BEHAVIORAL)

در بخش اول ساختار دهی رفتاری فیلپ فلاپ را طراحی میکنیم :

```
--this vhdl code represents the D FlipFlop Behavioral Architecture."
 --wer're gonna use this module for structuring the Universal Shift Register"


library ieee;
use ieee.std_logic_1164.all;
entity D_FF is
port (D, clk, rst: in std_logic;
      Q: out std_logic);
end D_FF;

Architecture behavioral of D_FF is
begin
    process(D, clk, rst)
    begin
        if (rst ='1') then
            Q <= '0';
        elsif(rising_edge(clk))then
        Q <= D;
        end if;
    end process;
end behavioral;
```

در ابتدای این کد کتابخانه های استاندارد IEEE را اضافه میکنیم .

در این ساختار برای فیلپ فلاپ پورت های D, CLK, RST بعنوان ورودی و پورت Q بعنوان خروجی .

در ساختار رفتاری فلیپ‌فلاپ :

وقتی که rst برابر ۱ بود ، خروجی Q را برابر صفر قرار میدهیم .

در غیراینصورت اگر rising_edge کلاک به وقع پیوست .

ورودی D را به خروجی Q انتقال میدهد .

و در اینجا ساختار رفتاری پایان میابد .

در این مرحله باید برای ساختاری که برای Dflipflop تعریف کرده ایم تست بنچ بنویسیم تا مطمئن شویم ساختار تعریف شده به درستی کار میکند .

کد آن به شرح زیر است :

```
--this code would test the functionality of D_FlipFlop module

library ieee;
use ieee.std_logic_1164.all;

entity DFF_testbench is
end DFF_testbench;

Architecture testbench of DFF_testbench is
component D_FF
port (D, clk, rst:in std_logic;
      Q:out std_logic);
end component;

signal D, clk, rst, Q : std_logic;

begin
b1: D_FF port map(D, clk, rst, Q);

    process
        begin
            D <= '0';
            clk <= '0';
            rst <= '0';
            wait for 10 ns;
            clk <= '1';

            wait for 10 ns;
            clk <= '1';
            wait for 10 ns;
            rst <= '0';
            D <= '0';
            wait for 40 ns;
            D <= '1';
            wait for 40 ns;

        end process;
end testbench;
```

در خط کامپوننت ساختار فیلیپ فلاپ را به تست بنچ معرفی میکنیم ،

سیگنال ها را تعریف میکنیم

در پورت مپ ، در واقع سیگنال هارا به پورت های فلیپ فلاپ وصل میکنیم .

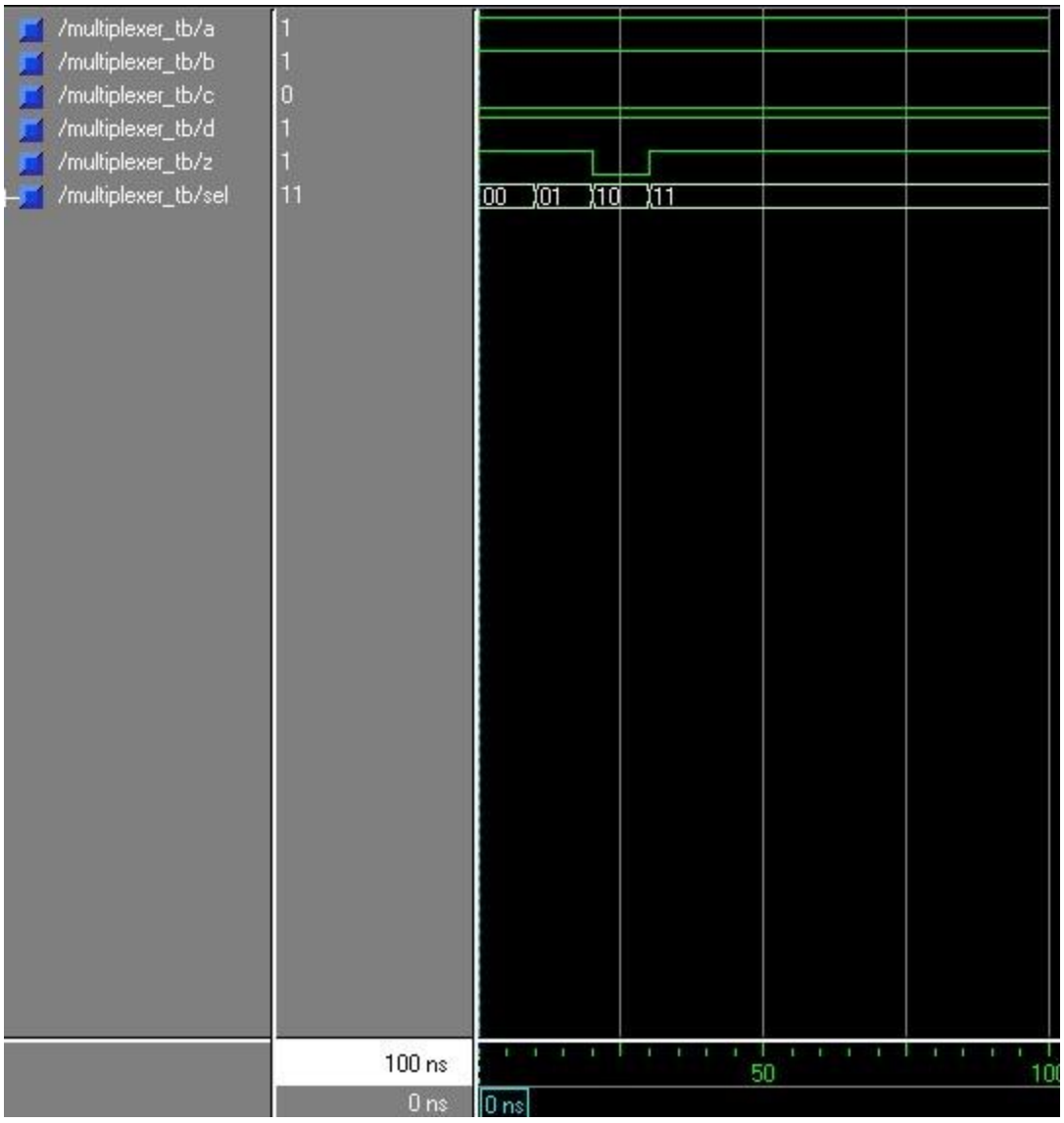
در مقدار دهی اول به تمام پورت ها صفر اختصاص میدهیم تا از حالت unknown خارج شوند .

پس از ۱۰ ثانیه به سیگنال کلاک 1 میدهیم (به ترتیب در بازهای مشخص به کلاک مقادیر مختلف میدهیم .)

در بازه مختلف کلاک به ریست و D مقادیر مختلف میدهیم .

باید رفتاری مشابه یک فلیپ فلاپ نشان دهد ،

سیگنال ها بصورت زیر خواهد بود :



Multiplexer 2_1:

در مالتی پلکسر ۲ به ۱ باید با تغییر خطوط سلکتور ، مقادیر خط انتخاب شده به خروجی مالتی پلکسر انتقال بیابد .

```vhdl
---This is a VHDL Code to Structure a Multiplexer


library ieee;
use ieee.std_logic_1164.all;
entity Multiplexer is
port (a, b, c, d: in std_logic;
      sel: in bit_vector(1 downto 0);
      z: out std_logic);
end Multiplexer;

Architecture structure of Multiplexer is
begin
--- this architecture is going to determine the behavior of 2_1 Multiplexer
  with sel select
    z <= a when "00",
     b when "01",
     c when "10",
     d  when others;
end structure;
```

در معماری ساختاری مالتی پلکسر ،

پورت ها a, b, c, d از نوع std logic ورودی

پورت Z std_logic خروجی .

و پورت سلکتور sel آرایه ۲ عنصری .

از ساختار with when این پیاده سازی را انجام میدهیم .

چهار خط انتخاب خواهیم داشت :

- در صورتی سلکتور برابر 00 شد محتوای متغیر a را به z انتقال میدهد
- در صورتی که سلکتور برابر 01 شد محتوای متغیر b را به z انتقال میدهد
- در صورتی که سلکتور برابر 10 شد محتوای متغیر c را به z انتقال میدهد
- در صورتی که سلکتور حالت دیگری گرفت ، d را به z انتقال میدهد .

و در اینجا طراحی ساختار پایان میابد .

در پایان test bench مالتی پلکسر را طراحی میکنیم .

با استفاده از Test bench مطمئن میشویم که شبیه سازی مطابق آنچه از یک مالتی پلکسر انتظار داریم رفتار خواهد کرد .

```
library ieee;
use ieee.std_logic_1164.all;

entity Multiplexer_tb is
end Multiplexer_tb;

Architecture tb of Multiplexer_tb is
component Multiplexer
port(a, b, c, d: in std_logic;
     sel: in bit_vector(1 downto 0);
     z: out std_logic);
end component;

signal a, b, c, d, z: std_logic; signal sel: bit_vector(1 downto 0);

begin
b1 : Multiplexer port map(a, b, c, d, sel, z);
    --- this test bench tests the multiplexer module functionality
    sel <= "00", "01" after 10 ns, "10" after 20 ns, "11" after 30 ns
    a <= '1';
    b <= '1';
    c <= '0';
    d <= '1';
end tb;
```

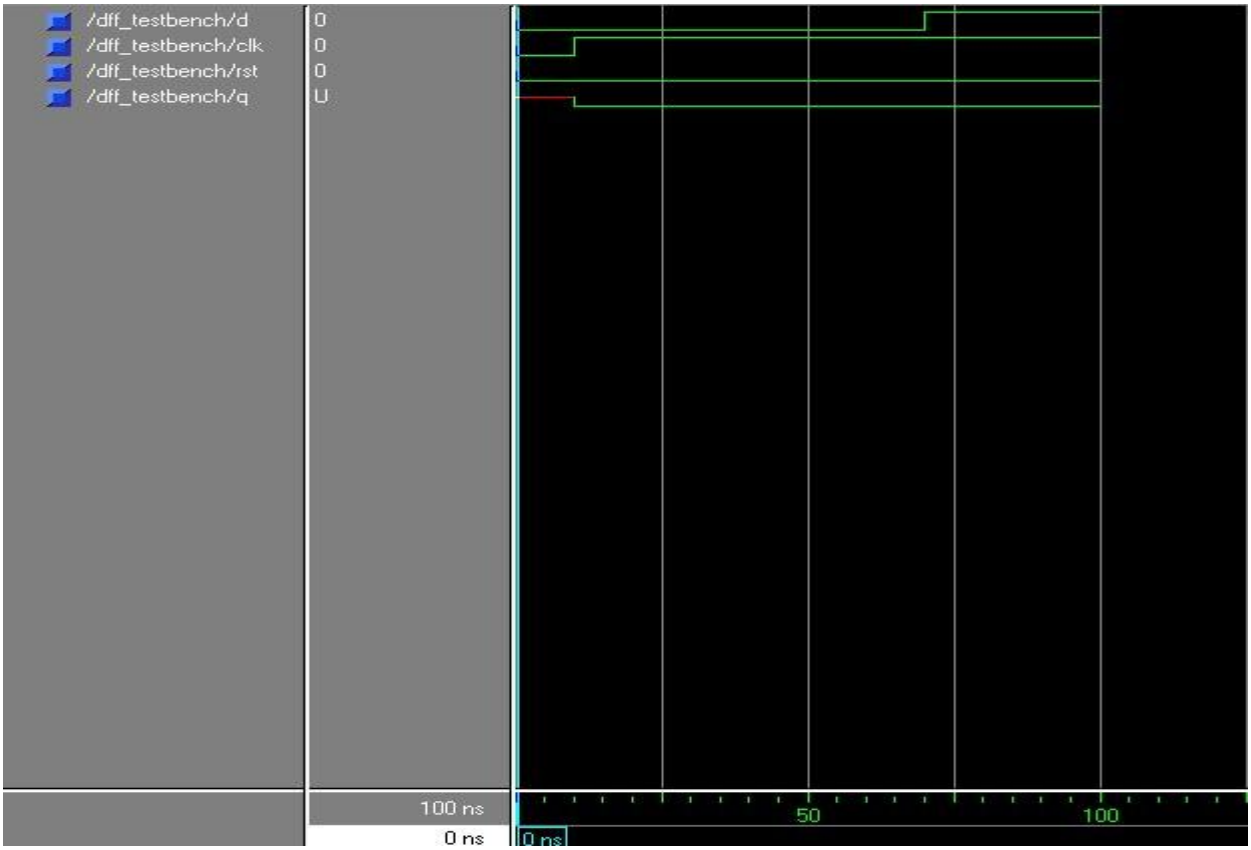
در این تست بنچ بعد از تعریف سیگنال ها و اضافه کردن کامپوننت multiplexer و مشخص کردن پورت مپ ها که سیگنال ها را به پورت های کامپوننت وصل میکند .

به پورت های a, b, c, d مقادیر دلخواه میدهیم .

و به ازای هر یک از حالت های Selector

سیگنال های خروجی را مشاهده میکنیم و باید طبق حالت سلکتور یکی از چهار ورودی a, b, c, d بر روی z قرار بگیرد .

سیگنال ها به صورت زیر خواهد بود .


/dff_testbench/d
/dff_testbench/clk
/dff_testbench/rst
/dff_testbench/q
0
0
0
U
100 ns
0 ns
50
100
0 ns