**به نام خدا**

**گزارش آزمایش هفتم**

**علی نوروزبیگی – فرهاد امان**

هدف آزمایش: ایجاد یک Universal Shift با قابلیت Parallel Load و Serial Load این Shift همچنین دارای ریست آسنکرون می‌باشد.

entity Shift is

port( reset, clk : in std\_logic;

sr, sl : in std\_logic;

sel : in std\_logic\_vector(1 downto 0);

pi : in std\_logic\_vector(3 downto 0);

output : out std\_logic\_vector(3 downto 0)

);

end Shift;

این شیفت یک خروجی برای نمایش وضعیت فلیپ فلاپ ها و همچنین یک ورودی Select برای انتخاب عملکرد مدار دارد 01 برای شیفت به راست 10 برای شیفت به چپ 00 برای Hold کردن وضعیت فعلی و در نهایت 11 برای عمل Parallel Load استفاده می‌شود.

همچنین در این مدار از مدار D-Flip-Flop با ریست آسنکرون و همچنین MUX4x1 به صورت سلسله مراتبی استفاده شده است. که در ابتدای کد Shift به صورت Component تعریف شده اند.

component mux4x1 is

port( data : in std\_logic\_vector(3 downto 0);

sel : in std\_logic\_vector(1 downto 0);

output : out std\_logic

);

end component mux4x1;

component DFF is

port(

d: in std\_logic;

clk: in std\_logic;

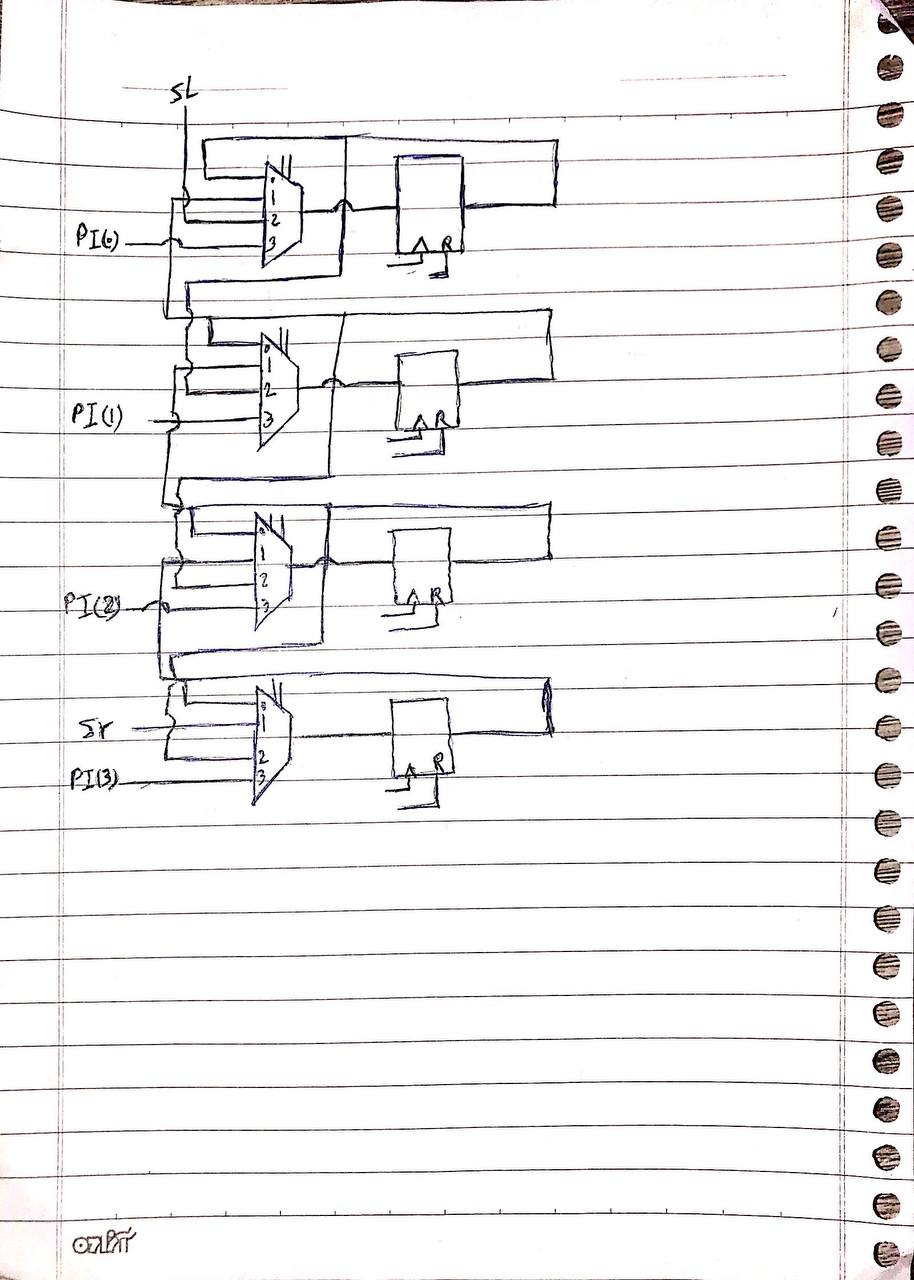
reset: in std\_logic;

q: out std\_logic

);

end component DFF;

در پایین می توانید شماتیک مدار را مشاهده کنید.



پس از تعریف Instance های مرود نیاز از هر کامپوننت کافیست تا به صورت Behavioral ورودی و خروجی

هر Component را مشخص کنیم.

data0 <= pi(0) & sl & q(1) & q(0);

data1 <= pi(1) & q(0) & q(2) & q(1);

data2 <= pi(2) & q(1) & q(3) & q(2);

data3 <= pi(3) & q(2) & sr & q(3);

mux0 : mux4x1 port map (data => data0 , sel => sel, output => d(0));

mux1 : mux4x1 port map (data => data1 , sel => sel, output => d(1));

mux2 : mux4x1 port map (data => data2 , sel => sel, output => d(2));

mux3 : mux4x1 port map (data => data3 , sel => sel, output => d(3));

dff0 : DFF port map (d => d(0), clk => clk, reset => reset, q => q(0));

dff1 : DFF port map (d => d(1), clk => clk, reset => reset, q => q(1));

dff2 : DFF port map (d => d(2), clk => clk, reset => reset, q => q(2));

dff3 : DFF port map (d => d(3), clk => clk, reset => reset, q => q(3));