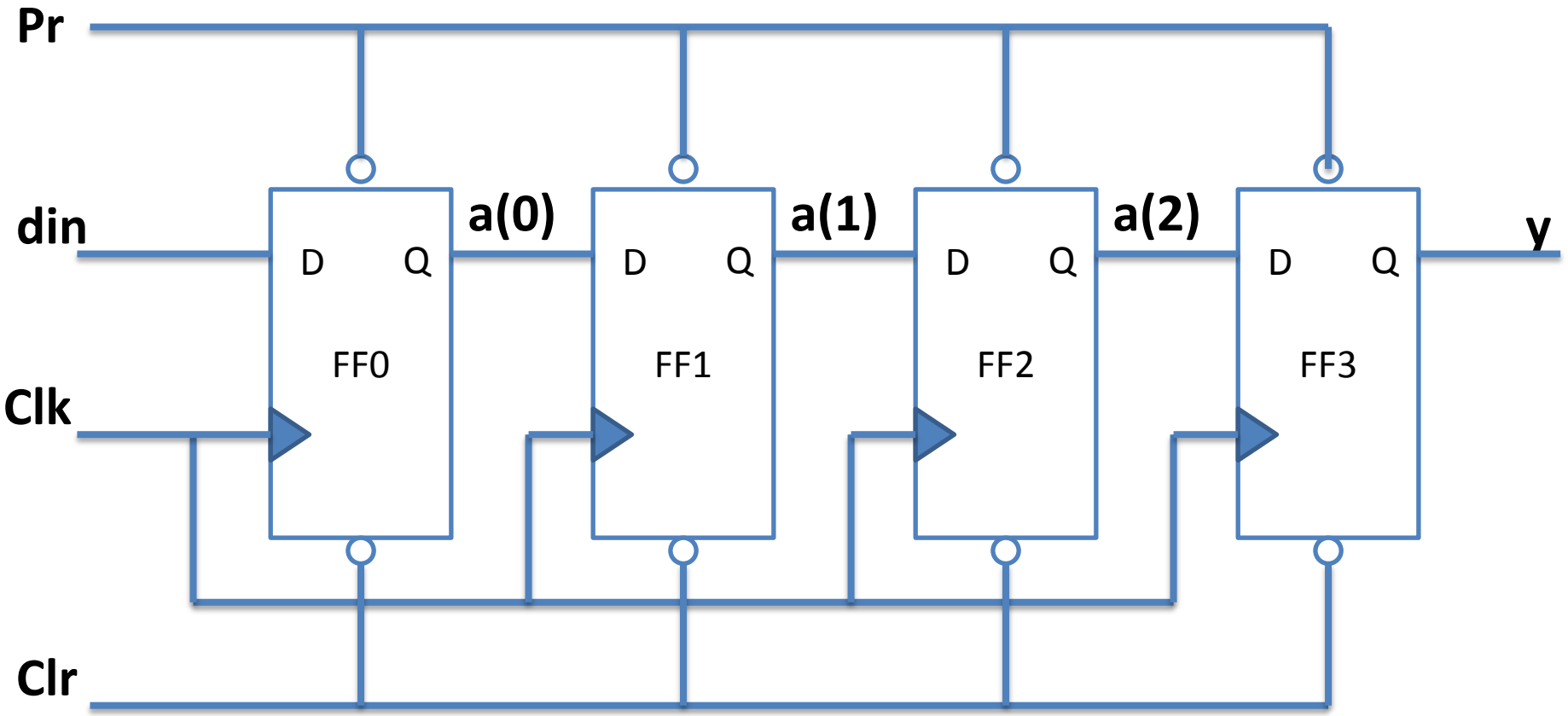
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity dff is
            port(
                        d : in STD_LOGIC;
                        pr : in STD_LOGIC;
                        cr : in STD_LOGIC;
                        clk: in std_logic;
                        q : out STD_LOGIC;
                        qbar : out STD_LOGIC
                );
end dff;
architecture dff of dff is
begin
            process(clk,pr,cr)
            begin
                        if (pr='0' and cr='1')then
                                    q <='1';

                        elsif(pr='1'and cr='0')then
                                    q<='0';

                        elsif(pr='1' and cr='1' and clk='0' and clk'event)then
                                    q<=d;
                        qbar<= not d;
                        end if;
                        end process;

end dff;
```

Serial in serial out shift register



Data Flow style

```
Entity ssft_reg is
Port ( din, clk, clr, pr:  in bit;
        y: out bit );
End ssft_reg;
Architecture  d_f  of ssft_reg is
Signal a: bit_vector (0 to 3);
Begin
a<= din & a(0) & a(1) & a(2)
When clr='1' and pr= '1' and clk ='0' and clk'event
Else
"1111" when clr='1' and pr='0' else
"0000" when clr='0' and pr='1' else
Y<= a(3);
End d_f;
```

Behavioral style

```
Entity ssft_reg is
Port ( din, clk, clr, pr:  in bit;  y: out bit );
End ssft_reg;
Architecture  beh_f  of ssft_reg is
Signal a(0), a(1), a(2) : bit;
Begin
  Process (clk, clr, pr)
   begin
   If (clr='1' and pr= '1' and clk ='0' and clk'event) then
                a(0) <= din;
                a(1)<= a(0);
                a((2)<= a(1);
                Y<= a(2);
   Elsif (clr='1' and pr='0' ) then
                a(0) <= '1';
                a(1)<= '1';
                a((2)<= '1';
                Y<= '1';
    Elsif (clr='0' and pr='1' ) then
                a(0) <= '0';
                a(1)<= '0';
                a((2)<= '0';
                Y<= '0';
  end if;   end process;
End beh  f;
```

Structural style

```
Entity ssft_reg is
Port (din, clk, clr, pr:  in bit;
        y: out bit );
End ssft_reg;
Architecture  str_f  of ssft_reg is
Component dff
Port ( a, b, c, d: in bit; o1: out bit);
End component;
Signal a: bit_vector (0 to 2);
Begin
ff0: dff port map (din, clk, clr, pr,  a(0));
ff1: dff port map (a(0), clk, clr, a(1));
ff2: dff port map (a(1), clk, clr, a(2));
ff3: dff port map (a(2), clk, clr, y);
End str_f;
```