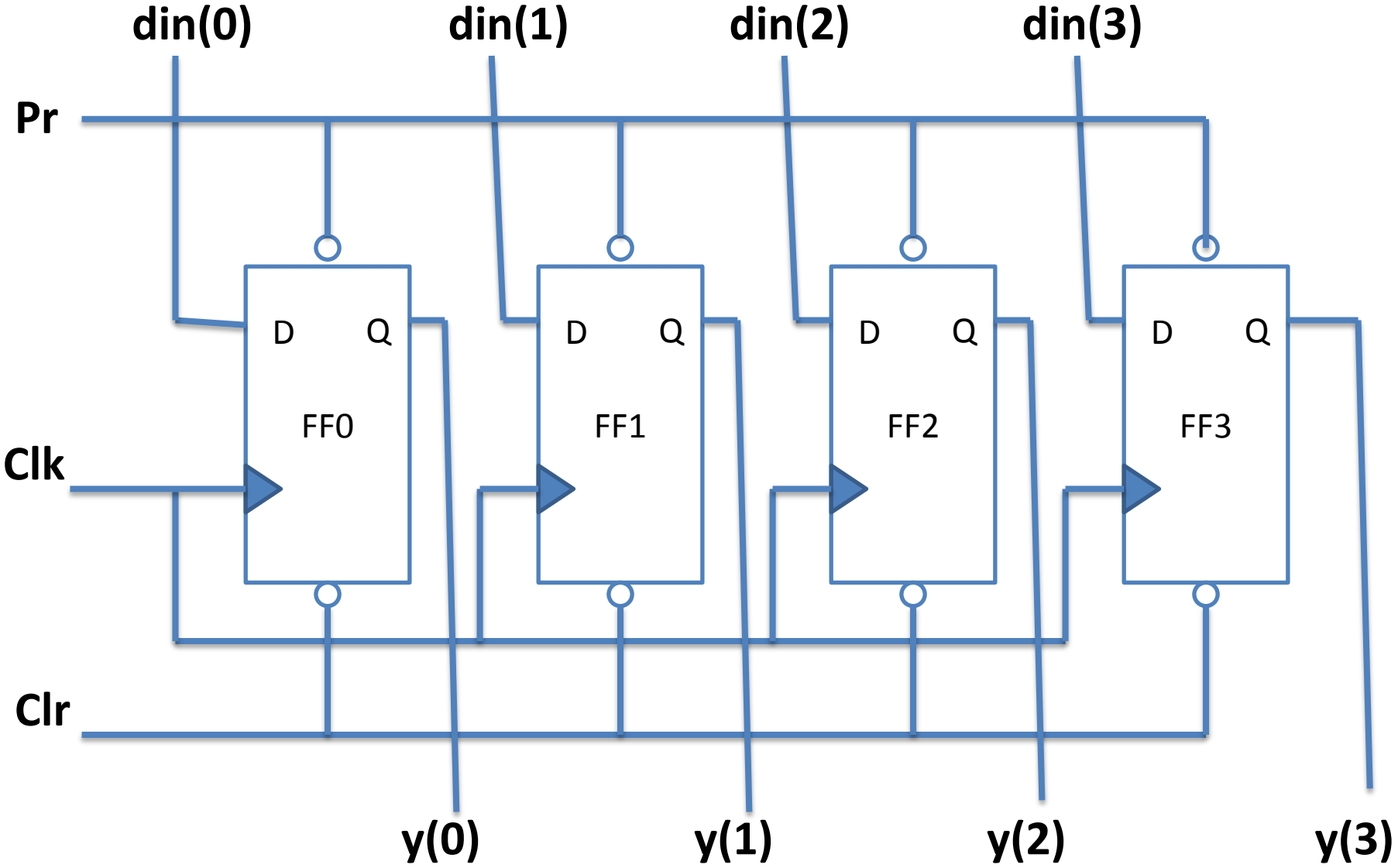
Parallel in parallel out shift register



Data Flow style

```
Entity ppf_reg is
Port ( din: in bit_vector (0 to 3);
        clk , clr, pr : in bit;
         y: out bit_vector (0 to 3));
End ppf_reg;
Architecture  d_f  of ppf_reg is
Y<= din 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';
End d_f;
```

Behavioral style

```
Entity ppf_reg is
Port ( din: in bit_vector (0 to 3);
        clk , clr, pr : in bit;
         y: out bit_vector (0 to 3));
End ppf_reg;
Architecture  beh_f  of ppf_reg is
begin
Process (clk, clr, pr)
begin
  If (clr='1' and pr= '1' and clk ='0' and clk'event) then        Y<= din;
  Elsif (clr='1' and pr='0' ) then
              Y<= "1111";
   Elsif (clr='0' and pr='1' ) then
               Y<= "0000";
  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(0), clk, clr, pr,  y(0));
ff1: dff port map (din(1), clk, clr, pr,  y(1));
ff2: dff port map (din(2), clk, clr, pr,  y(2));
ff3: dff port map (din(3), clk, clr, pr,  y(3));
End str_f;
```