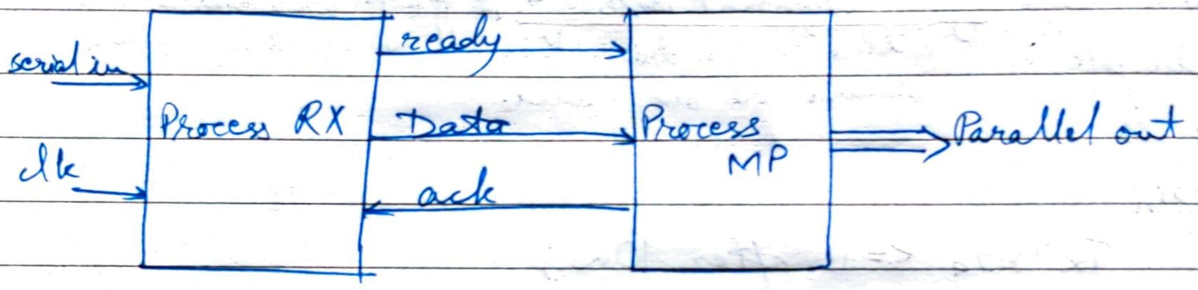
✯ Multiple Processes (Pg. 118)

* "Process" statement is concurrent
* Multiple "Process" statements possible inside a single architecture
* There processes can be interactive with each other.

* Example: Two interacting processes: RX (Receiver) & MP (uP)

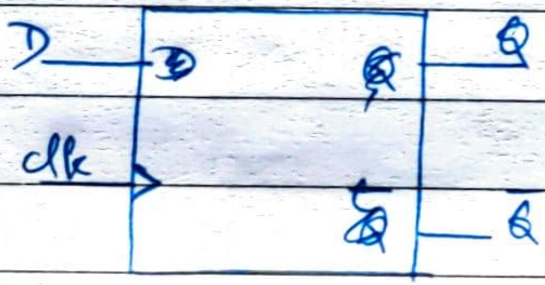


```
library ieee;
use ieee.std_logic_1164.all;
entity interacting_process is
   port(serial_in, clk: in bit;
        parallel_out : out bit_vector (0 to 7));
end interacting_process;
architecture two_pro of interacting_process is
signal ready, ack: bit;
signal data: bit_vector (0 to 7);
begin
    RX: process {
    begin
        read_word(serial_in, clk); ⟹ it a procedure
        ready <= '1';                that reads serial data on
        wait until ack = '1';        every clock pulse & converts
        ready <= '0';                it to a parallel data
        wait for 40 ns;              in signal DATA
    end process RX;                  It takes 10 ns to do this
```

```
MP: process
Begin
    wait for 25ns;
     parallel_out <= data;
     ack <= '1', '0' after 25ns;
     wait until ready = '1';
   end process MP;
end twopro;
```

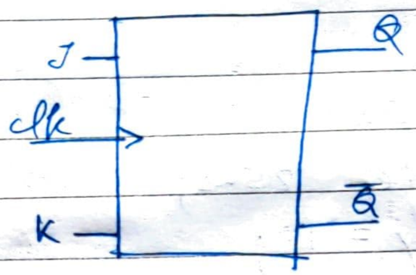
Examples

① D flip-flop



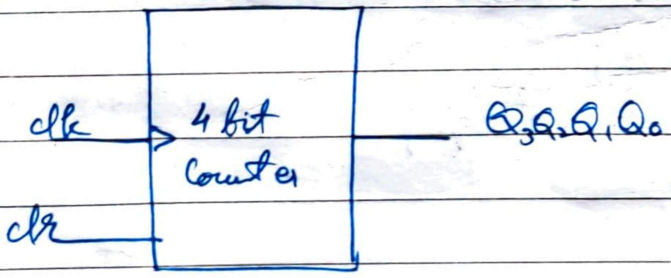
```
logic part
process (clk)          -> wait & sensi should
begin                     not appear simultaneously.
    if (clk = '1') then
        q <= d;
        wait for 0ns;
        qbar <= not (q);
    end if;
end process;
```

② JK Flip flop:



```
logic part:
  process (clk)
  if (clk'event and clk='1') then
        if (J='0' and K='0')
          ,
          ,
          ,
          ,
          ,
          ,
          ,
        end if;
    end if;
```

③ Counter



```
Hint, use ieee.std_logic_unsigned.all;
  → variable temp; bit_vector (3 downto 0);
      if : clr=1
          temp:="0000";
      elsif(                )
            temp := temp+1;
   end end if
       process
        q <= temp;
```