

Recapitulare

flux date

```

    " < = "
    WHEN / ELSE *
    WITH / SELECT **
```

descriere secvențială

```

    " := "
    IF / THEN * } PROCESS (listă de sens)
    CASE **     }
    LOOP
```

PROCESS (clk, semmale_asincrone)

VARIABLE
BEGIN

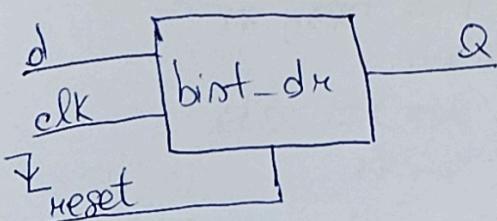
- se tratează semmale asincrone

Testare sincronizare după clk

→ - se testează semmale sincrone

END PROCESS;

Apt Sa se implementeze un bistabil de tip D activ pe frontul descrescător al semnalului de ceas care preia un semnal de reset asincron. Când semnalul reset e activ, iesirea bistabilului va fi menținută în '0' logic.



```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
ENTITY bist_dk IS
PORT (d,clk,reset : IN STD_LOGIC;
      q : OUT STD_LOGIC);
```

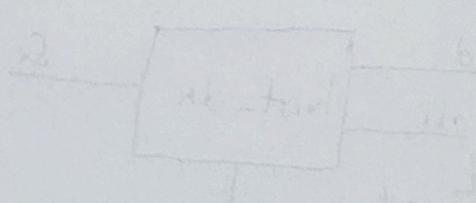
END bist_dk;

ARCHITECTURE dscr OF bist_dk IS

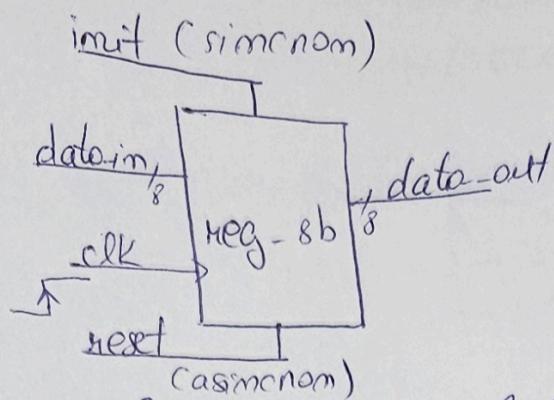
```
BEGIN
PROCESS (clk,reset)
BEGIN
IF reset = '1' THEN
  q <= 0;
ELSE IF (clk'event AND clk='0') THEN
  q <= d;
END IF;
END PROCESS;
END dscr;
```

Dacă semnalul reset este sincron, atunci procesul din programul anterior va modifica cu urmă:

```
PROCESS(clk)
BEGIN
IF (clk'event AND clk='1') THEN
  IF reset = '1' THEN
    q <= '0';
  ELSE
    q <= d;
  END IF;
END IF;
END PROCESS;
```



-) Sa se implementeze un registru pe 8 biti care are toate semnalurile inafara de "reset" sincrone.



Cind semnalul "init" e activ (semnal sincron), "data_out" e egal cu "FFH".

Cind semnal. "reset" e activ, "data_out" e egal cu "00H".

LIBRARY IEEE

USE IEEE.std_logic_1164.all;

ENTITY Reg_8b IS

PORT (init, clk, reset : IN std_logic;

data_in : IN std_logic_vector(7 DOWNTO 0);

data_out : OUT std_logic_vector(7 DOWNTO 0);

END Reg_8b;

ARCHITECTURE desc OF Reg_8b IS

BEGIN

PROCESS (clk, reset)

BEGIN

IF reset = '1' THEN

data_out <= (OTHERS => '0')

ELSE IF (clk'event AND clk = '1') THEN

IF init = '1' THEN

data_out <= x"FF";

```
ELSE  
    data_out <= data_in;  
END IF;  
END IF,  
END PROCESS;  
END descr;
```

