

Specificatia CASE

Aceasta specificatie permite selectarea unui grup de specificatii secentiale in functie de valoarea unei expresii atunci cand nu sunt egaleate toate alternativele selectiilor, restul sunt preluate de clauza OTHERS.

CASE expresie IS

WHEN alternativa_1 => secenta specificati_1;
WHEN alternativa_2 => —“—;
.....
WHEN OTHERS => secenta specif-N;

END CASE;

ex) Să se implementeze în VHDL folosind specificatia CASE un decodor GRAY cu intrare pe 3 biti.

ai [2...0]	y [2...0]
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 1
0 1 1	0 1 0
1 0 0	1 1 0
1 0 1	1 1 1
1 1 0	1 0 1
1 1 1	1 0 1

```

LIBRARY IEEE;
USE IEEE. std-logic-1164.all;

ENTITY dec-gray IS
  PORT(a: IN std-logic-vector (3 DOWNTO 0);
       y: OUT std-logic-vector (2 DOWNTO 0));
END dec-gray;

ARCHITECTURE descr OF dec-gray IS
BEGIN
  PROCESS(a)
    BEGIN
      CASE a IS
        WHEN "000" => y <= "000";
        WHEN "001" => y <= "001";
        -----
        WHEN OTHERS => y <= "100";
      END CASE;
    END PROCESS;
  END descr;

```

Specificatie

RÉPETITIVĂ

(nu este pt. verificare)

Această specificație repetă mai multe instrucțiuni conditional, conditional sau iterativ.

Modelul conditional:

WHILE condiție LOOP

- secvență specif. secvențială

END LOOP;

Secv. de specif. conditională se repetă atât timp cât condiție e adevarată.

Forma iterativă:

FOR variabilă IN interval LOOP.

-secvență specif. secvențială

END LOOP;

Secvență de specif. secv. se repetă de memorul specificat în interval.

Această specif. se recomandă a fi utilizată în simulare și mai puțin în realizarea de module (sinteză). Aceasta implică multi registrii, dar și semnale de interconectare a acestora.

Rezultatul acestor iterări consumă multă structură hardware, dar totodată se obțin și tempi mari de execuție.

Specificatio-

NEXT

Permite oprirea iterării în curs de desfășurare a unei buecle.

NEXT {eticheta}

NEXT etichete {xHEN condiții}

Eticheta reprezintă buclă care se

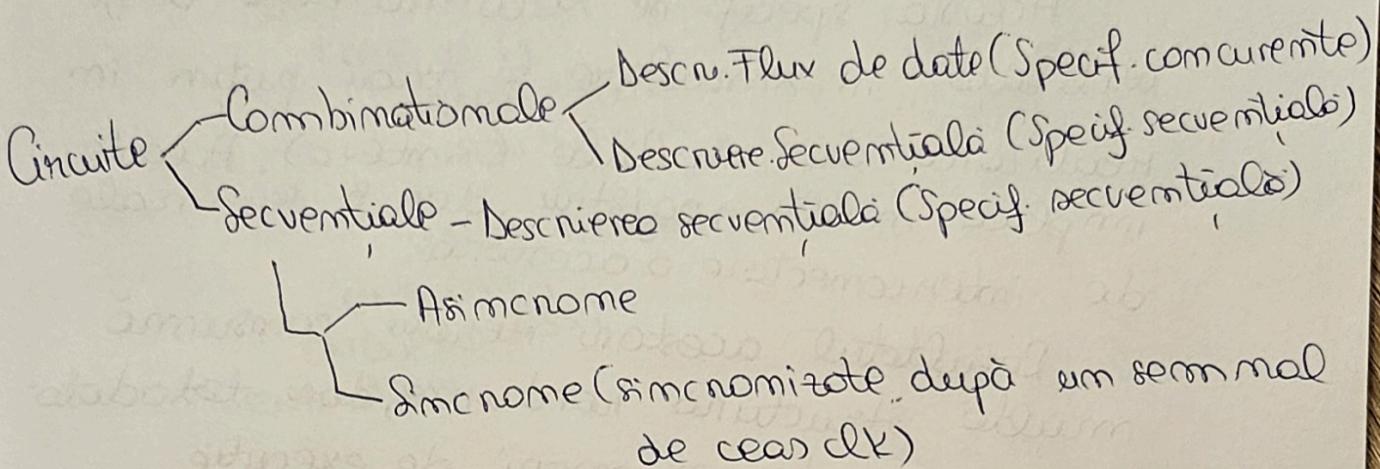
va opri.

Specificație EXIT

Permite ieșirea definitivă dintr-o buclă în curs de desfășurare.

EXIT {eticheta}

EXIT {eticheta} WHEN condiție



Proiectarea de module

digitale în logică sincronă / asincronă
pe structurile configurabile

Circuitele sincrone au toate celelele de memorie sincronizate după un semnal de ceas.

Structurile hardware reconfigurabile (FPGA) contin o rețea complexă de blocuri logice în care există logică combinatorială, dar

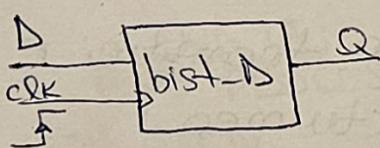
• registrii pentru memorarea stărilor.

Registrii sunt proiectați pt. a fi utilizati
împreună

Structurile FPGA dețin și sau mai multe
semnale de ceas globale, semnale
separate de rețea globală rotată a.î.
să se poată transmite frecvențe maxime
prin acestea.

În proiectarea circuitelor cu FPGA
se recomandă realizarea de circuite
bazate pe logica sincronă pentru utili-
zarea la maxim a capacitatii acesto-
ra.

ex să se implementeze un bistabil de
tip D cu trinșenarea pe front pozitiv a
semnalului de ceas.



	D	Q
F	0	0
↑	1	1
în rest		

memorare
stare anterioră

```

LIBRARY IEEE;
USE IEEE.STD.LOGIC_1164.ALL;
ENTITY bist_D IS
  PORT(D:IN STD_LOGIC;
       Q:OUT STD_LOGIC);
END bist_D;
ARCHITECTURE dscr OF bist_D IS
BEGIN
  PROCESS(CLK)
  BEGIN
    IF(CLK'EVENT AND CLK='1') THEN
      Q=D;
    END IF;
  END PROCESS;
END dscr;

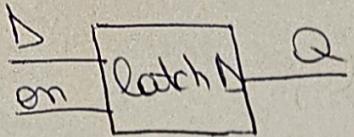
```

Pentru condiție de front creșător
 s-a testat dacă a apărut un eveniment pe semnalul de ceas și valoarea finală în urmele transiției e împreună cu aceste prime instrucțiuni:

```
IF(CLK'EVENT AND CLK='1') THEN
```

În cazul în care nu este îndeplinită condiție valoarea lui Q rămâne memoriazată.

Bistabilul de tip D poate fi transformat într-un latch de tip D.



en	D	Q
'1'	x	x
0		

me mă rezo
starea descriptivă

LIBRARY IEEE

USE IEEE.std_logic_1164.all;

ENTITY latch_d IS

PORT(D: IN std_logic;
Q: OUT std_logic);

END latch_d;

ARCHITECTURE dscr OF latch_d

BEGIN

PROCESS(D, en)

BEGIN

IF en = '1' THEN

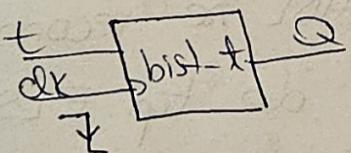
Q <= D;

END IF;

END PROCESS;

END dscr;

ex) să se descrie un bistabil de tip T



$$t=0 \quad Q_+ = Q$$

$$t=1 \quad Q_+ = \text{NOT } Q$$

```

LIBRARY IEEE;
USE IEEE.STD.LOGIC_1164.ALL;

ENTITY bist_t IS
    PORT (T:IN STD.LOGIC;
          Q:OUT STD.LOGIC);
END bist_t;

ARCHITECTURE descr OF bist_t IS
BEGIN
    PROCESS(CLK)
        VARIABLE temp: STD.LOGIC := '0';
    BEGIN
        IF (CLK'EVENT AND CLK = '0') THEN
            IF T = '1' THEN
                temp := NOT TEMP;
            ELSE temp := temp;
            END IF;
        END IF;
        Q := temp;
    END PROCESS;
END descr;

```

În bistabilul tip t avem semnalul sincron CLK și semnalul t sincronizat după CLK.

În cadrul codului VHDL se recomandă a fi tratată și nemulțime de tip ELSE a instrucțiunii IF chiar dacă rezultatul acesteia este un lanch (nu e necesar).

OBS! În cadrul circuitelor sincrone, în lista de sensibilități a procesului se introduce semnalul după care se face sincronizarea și semnalele asincrone. (dacă există)

În cadrul procesului sunt tratate prima dată condițiile semnalelor asincrone; condiția semnalelor de sincronizare pe semnalul sincron și, în final condițiile pentru semnale sincrone în cadrul condițiilor de sincronizare.

Circuit sincron:

- semnale asincrone
- semnalul după care se face sincronizarea (clk)
- semnale sincrone

PROCESS (clk, semnale - asincrone)

BEGIN

1. Testare semnale asincrone (logica a semnalelor asincrone)

2. Test sincronizare după clk (IF clk event...)

2.1. logica semnale sincrone

END PROCESS;