EINES DE DISSENY DIGITAL AVANÇAT.
INTRODUCCIÓ AL VHDL

Josep Lluís Rosselló

VHDL

- *Documentació de dissenys*
- *Simulació del comportament d'un circuit*
- *Síntesi lògica*

- Nivells d'abstracció:
  - Algoritmica
  - Transferència de registres
  - Nivell de portes lògiques

Composat per:

- Entitat: Defineix entrades i sortides (si son bits o arrays de bits). Cada entitat identificada amb un únic nom
- Arquitectura. Funcionalitat d'una determinada entitat. Pot ser de tres tipus:
  - Comportament (algoritme)
  - Fluxe de dades (descriu estructures i comportament)
  - Estructural (defineix components i interconnexions)

```
ENTITY nom_entitat IS
   [ports]  -- Els comentaris comencen amb --
   [declaracions (constants, tipus, senyals)]
END nom_entitat;

ARCHITECTURE nom_ arquitectura OF nom_entitat IS
    [part declarativa] -- definim tipus, constants, subprogrames, senyals etc..
BEGIN
   [descripció del disseny]
END nom_arquitectura;
```

Exemple (comparador)

```
ENTITY compare IS
PORT (a,b: in bit;
          c: out bit);
END compare;

ARCHITECTURE first OF compare IS
BEGIN
c <= NOT (a XOR b);
END first;
```

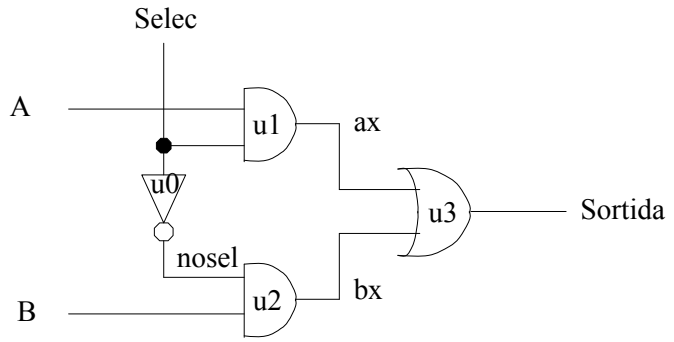
Mode comportament (porta AND)

```
ENTITY andgate IS
PORT (a,b: IN BIT;
          c: OUT BIT);
END andgate;

ARCHITECTURE arch1 OF andgate IS
BEGIN
   PROCESS (a,b);  -- El bloc PROCESS s'executa en sèrie quan varien a ò b
   BEGIN
        IF a='1' AND b='1' THEN
        c<='1';
        ELSE
        c<='0';
        ENDIF;
   END PROCESS;
END arch1;
```

Mode de transferència de registres (AND2)

```
ENTITY xor2 IS
   GENERIC (m: time:=1.0 ns);
   PORT (a,b: IN BIT;
             c: OUT BIT);
END xor2

ARCHITECTURE dataflow OF xor2 IS
BEGIN
   c<=a AND b;
END dataflow;
```

Multiplexor de dos bits



```
ARCHITECTURE transf1 OF mux IS
BEGIN
    sortida <= a WHEN selec='1' ELSE b;
END transf1;

ARCHITECTURE transf2 OF mux IS
SIGNAL nosel,ax,bx: bit;
BEGIN
    nosel<=NOT selec;
    ax<=a AND selec;
    bx<=b AND nosel;
    salida<=ax OR bx;
END transf2;

-- Descripció concurrent o de transferència
   entre registres. Les instruccions
   s'executen en paral·lel. Es a dir,
   s'executen quan canvia qualsevol
   senyal implicada
```

```
ARCHITECTURE comportament OF mux IS
BEGIN
   PROCESS (a,b,selec)
   BEGIN
      IF (selec='1') THEN
            sortida<=a;
      ELSE
            sortida<=b;
      END IF;
   END PROCESS;
END comportament;

-- Descripció comportamental. El process
   s'executa quan varien a,b ó selec.
   (anomenada llista sensible)
```

VHDL estructural

```
ARCHITECTURE estructura OF mux IS
COMPONENT and2
   PORT(e1,e2: IN bit; y: OUT bit);
END COMPONENT;
COMPONENT or2
   PORT(e1,e2: IN bit; y: OUT bit);
END COMPONENT;
COMPONENT inv
   PORT(e: IN bit; y: OUT bit);
END COMPONENT;
SIGNAL ax,bx,nosel: bit; --Aquestes senyals equivalen a connexions físiques entre components
BEGIN
   u0: inv PORT MAP(e=>selec,y=>nosel);
   u1: and2 PORT MAP(e1=>a,e2=>selec,y=>ax);
   u2: and2 PORT MAP(e1=>b,e2=>nosel,y=>bx);
   u3: or2 PORT MAP(e1=>ax,e2=>bx,y=>sortida);
END estructura;
```

Es defineix cada component del sistema així com les distintes interconnexions entre ells

Si no es declara cap llibreria els components es cercaran a la llibreria WORK

Elements sintàctics VHDL

- Entrades/sortides
  - IN: Es poden llegir però no escriure
  - OUT: Es poden escriure però no llegir
- Constants: Valor que no es pot canviar una vegada inicialitzat
- Variables:Definides dins el PROCESS. El seu valor pot ser variat en qualsevol moment. Serveixen de forma auxiliar
- Senyals (SIGNALS): Definides dins l'arquitectura o en PACKAGE. Poden ser pensades com senyals elèctriques dins un circuit

Diferències entre Variable i senyal

- Les senyals esperen a la seva actualització al següent pas de simulació. (Al final del process en el cas comportamental)
  - S'assignen amb el símbol <=
- Les variables s'actualitzen en el moment de l'assignació. Només tenen sentit en entorns de programació en sèrie (PROCESS)
  - S'assignen amb el símbol :=

Sistema de simulació

Cada senyal actualitza el seu valor al següent pas de simulació.

sortida <= a WHEN selec='0' ELSE b;

(Quan varien a,b ò selec la senyal 'sortida' s'actualitza immediatament. A això se li diu un pas de simulació quan estam en mode concurrent)

Senyal=(Valor Actual, Valor Futur) s'assignen amb <=

Un bloc PROCESS equival a una única instrucció concurrent

```
PROCESS (a,b,selec)                     -- a,b,selec son senyals.
BEGIN                                   -- Suposa selec=(1,1), sortida=(0,0) b=(1,1)
        IF (selec='0') THEN
                sortida<=a;
        ELSE
                sortida<=b;  -- s'executa aquesta línia ja que el valor actual de selec es 1
        END IF;              --una vegada executada sortida val sortida=(0,1)
END PROCESS;                 --Al sortir del process sobreescrivim el valor actual i sortida=(1,1)
```

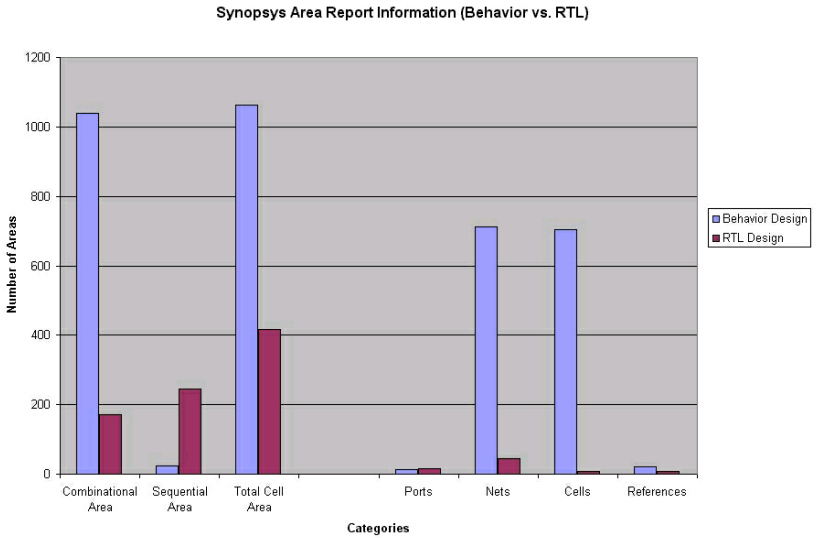
(Quan varien a,b ò selec la senyal 'sortida'actualitza al final del PROCESS. A això se li diu un pas de simulació quan estam en mode comportament)

EXEMPLE UTILITZACIÓ DE VARIABLES

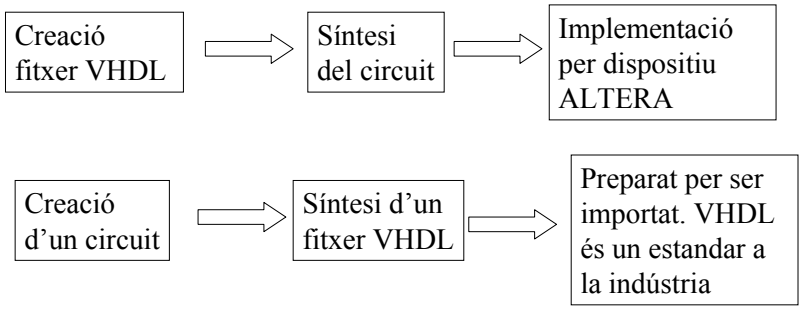
```
ENTITY counter IS    -- EXEMPLE: COMPTADOR DE "UNS"
    PORT (d: IN BIT_VECTOR (3 DOWNTO 0);
          q: OUT INTEGER RANGE 0 TO 4);
END counter;

ARCHITECTURE arch1 OF counter IS
BEGIN
    PROCESS (d)                         -- Totes les entrades declarades
    VARIABLE n: INTEGER;                -- Variable auxiliar. Només te valor actual
    BEGIN
        n:=0;
        FOR i IN d'RANGE LOOP
            IF d(i)='1' THEN
                n:=n+1;
            END IF;
        END LOOP;
        q<= n;                          --Al final, la variable s'escriu sobre un senyal
    END PROCESS;
END arch1;
```

Quin tipus d'estructura usar?

- Per simulacions els models de comportament son més eficients en termes de espai de memòria i temps d'execució.
- Si el model es vol amb la idea de montar un layout d'un circuit, aleshores el mode estructural és més apropiat
- A l'hora de fer síntesi s'ha de anar amb cuita amb el mode comportament ja que els algoritmes de síntesi poden donar lloc a problemes en cas de dissenys complexos



Possibilitats d'us. Entorn MAX+PLUS II



•MAX+PLUS II incorpora l'estándar IEEE 1076-1987 VHDL
•Fitxers .vhd poden ser incroporats dins un disseny jeràrquic anomenat projecte
•A un fitxer VHDL es poden incorporar les macrofuncions proporcionades per ALTERA
•També es poden crear funcions o llibreries adaptades a les necessitats del usuari

Implementació de circuits combinacionals

- Usarem normalment VHDL concurrent o "PROCESS" en determinats casos.
- Mode concurrent
  - Assignació simple de senyals
  - Assignació condicional de senyals
  - Assignació de senyals seleccionades
- Mode comportament
  - A la llista sensible del PROCESS han de ser totes les senyals d'entrada

MODE CONCURRENT
Assignació simple de senyals

```
ENTITY simpsig IS      -- EXEMPLE: PORTA AND
    PORT (a,b: IN BIT;
                c: OUT BIT);
END simpsig;

ARCHITECTURE arch1 OF simpsig IS
BEGIN
    c<= a AND b;
END arch1;
```

MODE CONCURRENT
Assignació condicional de senyals

```
ENTITY condsig1 IS   -- EXEMPLE: MULTIPLEXOR DE 2-a-1
    PORT (a0,a1,selec: IN BIT;
                y: OUT BIT);
END condsig1;

ARCHITECTURE arch1 OF condsig1 IS
BEGIN
    y<= a0  WHEN selec='0' ELSE a1;
END arch1;
```

MODE CONCURRENT
Assignació condicional de senyals

```
ENTITY condsig2 IS    -- EXEMPLE: ENDODIFICADOR AMB PRIORITAT
    PORT (high,mid,low: IN BIT;
                q: OUT INTEGER RANGE 0 TO 3);
END condsig2;

ARCHITECTURE arch1 OF condsig2 IS
BEGIN
    q<=  3 WHEN high='1' ELSE
         2 WHEN mid='1' ELSE
         1 WHEN low='1' ELSE
         0;
END arch1;
```

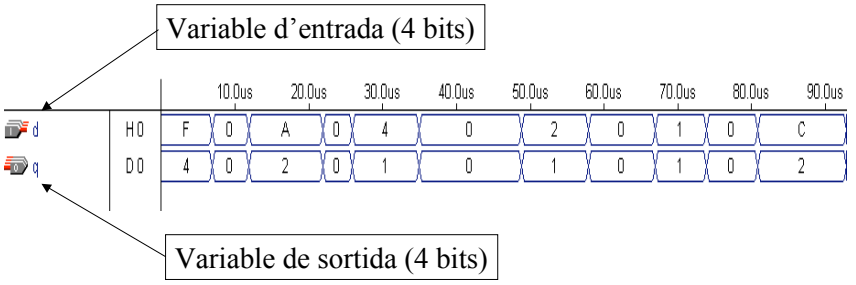
MODE CONCURRENT
Assignació de senyals seleccionades

```
ENTITY selsig IS    -- EXEMPLE: MULTIPLEXOR 4-a-1
    PORT (d0,d1,d2,d3: IN BIT;
          s: IN INTEGER RANGE 0 TO 3;
          y: OUT BIT);
END selsig;

ARCHITECTURE arch1 OF selsig IS
BEGIN
WITH s SELECT
    y<=  d0 WHEN 0,
         d1 WHEN 1,
         d2 WHEN 2,
         d3 WHEN 3;
END arch1;
```

MODE COMPORTAMENT
Utilització de l'instrucció PROCESS

```
ENTITY counter IS    -- EXEMPLE: COMPTADOR DE "UNS"
    PORT (d: IN BIT_VECTOR (3 DOWNTO 0);
          q: OUT INTEGER RANGE 0 TO 4);
END counter;

ARCHITECTURE arch1 OF counter IS
BEGIN
    PROCESS (d)                    -- Totes les entrades declarades
    VARIABLE n: INTEGER;           -- Variable auxiliar
    BEGIN
        n:=0;
        FOR i IN d'RANGE LOOP
            IF d(i)='1' THEN
                n:=n+1;
            END IF;
        END LOOP;
        q<= n;
    END PROCESS;
END arch1;
```

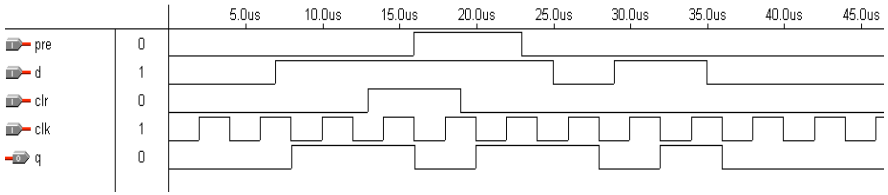
MODE COMPORTAMENT
Utilització de la instrucció PROCESS
Simulació del comptador d'uns.



Implementació de circuits sequencials

- La lògica sequencial està implementada amb instruccions PROCESS
- Síntesi de registres, comptadors i màquines d'estats

Simulació Registre



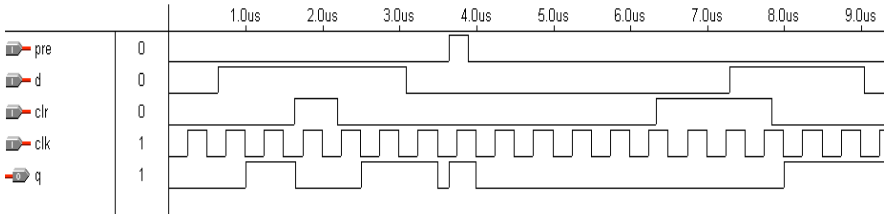
Registre actiu per nivell baix amb clear i preset síncron

```
ENTITY registre IS
    PORT(d,clk,clr,pre: IN BIT;
                q: OUT BIT);
END registre;
ARCHITECTURE arch1 OF registre IS
BEGIN
    PROCESS (clk)
    BEGIN
        IF clk'EVENT AND clk='0' THEN
                IF clr='1' THEN
                        q<='0';
                ELSIF pre='1' THEN
                        q<='1';
                ELSE
                        q<=d;
                END IF;
        END IF;
    END PROCESS;
END arch1;
```

Registre actiu per nivell baix amb clear i preset asíncron

```
ENTITY registre IS
    PORT(d,clk,clr,pre: IN BIT;
                q: OUT BIT);
END registre;
ARCHITECTURE arch1 OF registre IS
BEGIN
    PROCESS (clk,clr,pre)
    BEGIN
        IF clr='1' THEN
                q<='0';
        ELSIF pre='1' THEN
                q<='1';
        ELSIF clk'EVENT AND clk='0' THEN
                q<=d;
        END IF;
    END PROCESS;
END arch1;
```

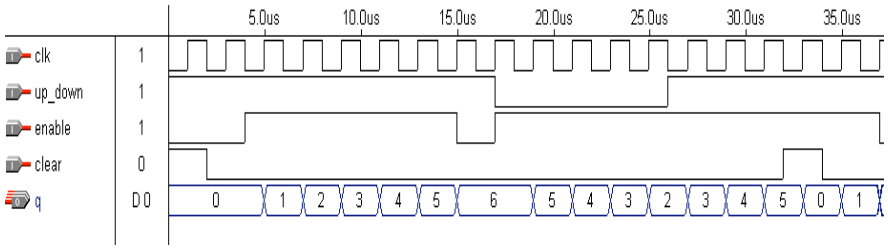
Simulació Registre



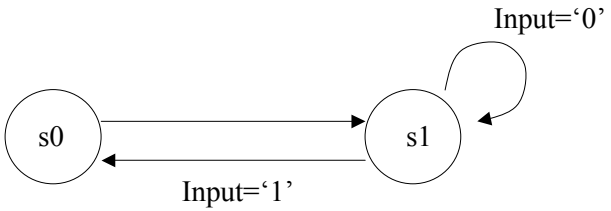
Comptador Up/Down. Amb clear síncron

```
ENTITY counter IS
        PORT (clk,clear,enable,up_down: IN BIT;
                q: OUT INTEGER RANGE 0 TO 255);
END counter;

ARCHITECTURE arch OF counter IS
BEGIN
        PROCESS (clk)
        VARIABLE cnt: INTEGER RANGE 0 TO 255;
        VARIABLE direccio: INTEGER RANGE -1 TO 1;
        BEGIN
                IF up_down ='1' THEN
                                direccio:=1;
                ELSE
                                direccio:=-1;
                END IF;
                IF (clk'EVENT AND clk='1') THEN
                                IF clear='1' THEN
                                                cnt:=0;
                                ELSIF enable='1' THEN
                                                cnt:=cnt+direccio;
                                END IF;
                END IF;
                q<=cnt;
        END PROCESS;
END arch;
```

Simulació comptador Up/Down



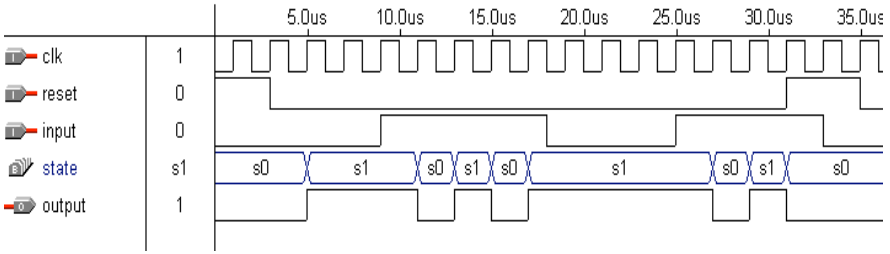
Síntesi de màquines d'estats



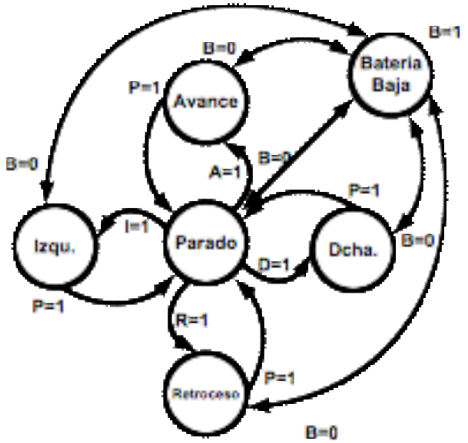
Síntesi de màquines d'estats

```
ENTITY state_machine IS
    PORT (clk,reset,input: IN BIT;
          output: OUT bit);
END state_machine;

ARCHITECTURE arch OF state_machine IS
TYPE STATE_TYP IS (s0,s1);
SIGNAL state: STATE_TYP;
BEGIN
    PROCESS (clk,reset)
    BEGIN
        IF reset ='1' THEN
            state<=s0;
        ELSIF (clk'EVENT AND clk='1') THEN
            CASE state IS
                WHEN s0 =>
                    state <= s1;
                WHEN s1=>
                    IF input='1' THEN
                        state<=s0;
                    ELSE
                        state<=s1;
                    END IF;
            END CASE;
        END IF;
    END PROCESS;
    output <='1' WHEN state=s1 ELSE '0';
END arch;
```

Simulació de màquines d'estats



Síntesi de màquines d'estats




```
ENTITY controlRobot IS
PORT(
   clk,D,I,A,R,P,B: IN bit;
   Salida: OUT INTEGER RANGE 0 TO 5);
END controlRobot;
ARCHITECTURE flujo OF controlRobot IS
TYPE estado IS (parado, retroceder, avanzar,
   izquierda, derecha, BateriaBaja);
SIGNAL presente: estado:=parado;
```

```
BEGIN
proceso:PROCESS (clk)
	BEGIN
		IF (clk'EVENT AND clk='1') THEN
			CASE presente IS
				WHEN parado =>
					IF B='1' THEN
						presente<=BateriaBaja;
					ELSIF P='1' THEN
						presente<=parado;
					ELSIF A='1' THEN
						presente<=avanzar;
					ELSIF R='1' THEN
						presente<=retroceder;
					ELSIF D='1' THEN
						presente<=derecha;
					ELSIF I='1' THEN
						presente<=izquierda;
					ELSE
						presente<=parado;
					END IF;
				WHEN retroceder =>
					IF B='1' THEN
						presente<=BateriaBaja;
					ELSIF P='1' THEN
						presente<=parado;
					ELSIF R='1' THEN
						presente<=retroceder;
					ELSIF A='1' THEN
						presente<=avanzar;
					ELSIF D='1' THEN
						presente<=derecha;
					ELSIF I='1' THEN
						presente<=izquierda;
					ELSE
						presente<=retroceder;
					END IF;
```

```
				WHEN izquierda =>
					IF B='1' THEN
						presente<=BateriaBaja;
					ELSIF P='1' THEN
						presente<=parado;
					ELSIF I='1' THEN
						presente<=izquierda;
					ELSIF A='1' THEN
						presente<=avanzar;
					ELSIF R='1' THEN
						presente<=retroceder;
					ELSIF D='1' THEN
						presente<=derecha;
					ELSE
						presente<=izquierda;
					END IF;
				WHEN BateriaBaja =>
					IF B='0' THEN
						presente<=parado;
					ELSE
						presente<=BateriaBaja;
					END IF;
			END CASE;
		END IF;
END PROCESS proceso;
Salida <=	0 WHEN presente=parado	ELSE
		1 WHEN presente=BateriaBaja ELSE
		2 WHEN presente=avanzar	ELSE
		3 WHEN presente=retroceder ELSE
		4 WHEN presente=izquierda ELSE
		5 ;
END flujo;
```

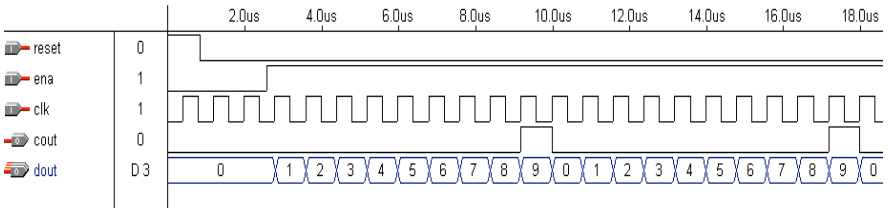
```
WHEN avanzar =>
		IF B='1' THEN
			presente<=BateriaBaja;
		ELSIF P='1' THEN
			presente<=parado;
		ELSIF A='1' THEN
			presente<=avanzar;
		ELSIF R='1' THEN
			presente<=retroceder;
		ELSIF D='1' THEN
			presente<=derecha;
		ELSIF I='1' THEN
			presente<=izquierda;
		ELSE
			presente<=avanzar;
		END IF;
WHEN derecha =>
		IF B='1' THEN
			presente<=BateriaBaja;
		ELSIF P='1' THEN
			presente<=parado;
		ELSIF D='1' THEN
			presente<=derecha;
		ELSIF A='1' THEN
			presente<=avanzar;
		ELSIF R='1' THEN
			presente<=retroceder;
		ELSIF I='1' THEN
			presente<=izquierda;
		ELSE
			presente<=derecha;
		END IF;
```

COMPTADOR DE MÒDUL 10

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY bcdcount IS
PORT(
	clk,reset,ena		:IN STD_LOGIC;
	dout				:OUT INTEGER RANGE 0 TO 9;
	cout				:OUT STD_LOGIC
);
END bcdcount;
```

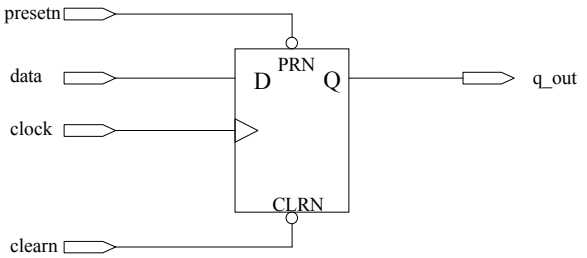
```
ARCHITECTURE bcdcounter OF bcdcount IS
BEGIN
bcd0:       PROCESS (clk)
            VARIABLE n: INTEGER RANGE 0 TO 9;
            BEGIN
                        IF (clk'EVENT AND clk='1') THEN
                                    IF reset='1' THEN
                                                n:=0;
                                    ELSE
                                                IF ena='1' AND n<9 THEN
                                                            IF n=8 THEN
                                                            n:=n+1;
                                                            cout<='1';
                                                            ELSE
                                                            n:=n+1;
                                                            cout<='0';
                                                            END IF;
                                                ELSIF ena='1' AND n=9 THEN
                                                            n:=0;
                                                            cout<='0';
                                                END IF;
                                    END IF;
                        END IF;
                        dout <= n;
      END PROCESS bcd0;
END bcdcounter;
```

SIMULACIÓ COMPTADOR MÒDUL 10



IMPLEMENTACIÓ DE MÒDULS ESTRUCTURALS

```
LIBRARY altera;       -- INCLOU LA LLIBRERIA ALTERA
USE altera.maxplus2.ALL;

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY compinst IS
      PORT
      (
            data, clock, clearn, presetn    : IN STD_LOGIC;
            q_out                           : OUT STD_LOGIC;
      );
END compinst;

ARCHITECTURE a OF compinst IS

BEGIN

      dff1 : DFF PORT MAP (d =>data, q => q_out, clk => clock, clrn => clearn, prn => presetn);

END a;
```



Conversor de set segments

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY converter IS
PORT (
    enable    :IN STD_LOGIC;
    entrada   :IN INTEGER RANGE 0 TO 9;
    sevseg    :OUT BIT_VECTOR(6 DOWNTO 0)
);
END converter;
```

```
ARCHITECTURE sevseg OF converter IS
BEGIN
    PROCESS(enable,entrada)
    BEGIN
    IF enable='0' THEN
            sevseg <="0000000" ;
    ELSE
    CASE entrada IS
            WHEN 0 => sevseg <="1111110";
            WHEN 1 => sevseg <="0110000";
            WHEN 2 => sevseg <="1101101";
            WHEN 3 => sevseg <="1111001";
            WHEN 4 => sevseg <="0110011";
            WHEN 5 => sevseg <="1011011";
            WHEN 6 => sevseg <="0011111";
            WHEN 7 => sevseg <="1110000";
            WHEN 8 => sevseg <="1111111";
            WHEN 9 => sevseg <="1110011";
    END CASE;
    END IF;
    END PROCESS;
END sevseg;
```