

***Descripció de sistemes amb el llenguatge VHDL***

1. Descripció de primitives bàsiques
2. Models de comportament de sistemes
3. El paquet estàndard de síntesi IEEE Std 1076.3-1997
4. Estàndard de síntesi a nivell RTL (IEEE 1076.6)

## ***1. Descripció de primitives bàsiques***

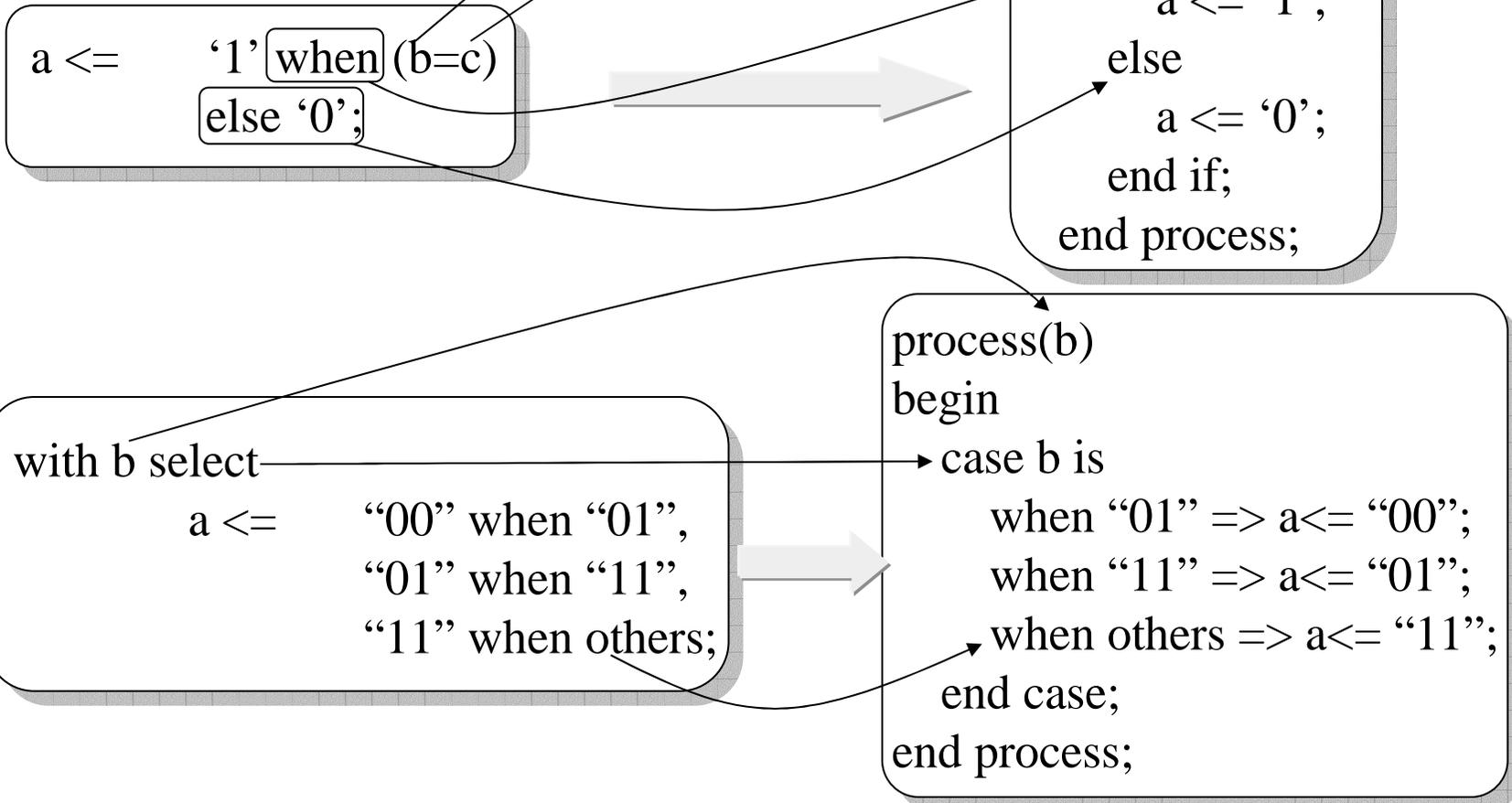
- **Assignacions concurrents i processos:**

➤ **Assignacions concurrents:** Es projecta un canvi sobre el senyal situat a l'esquerra de l'assignació quan es produeixi un canvi a qualsevol dels senyals situats a la dreta

$a \leq b \text{ xor } c;$

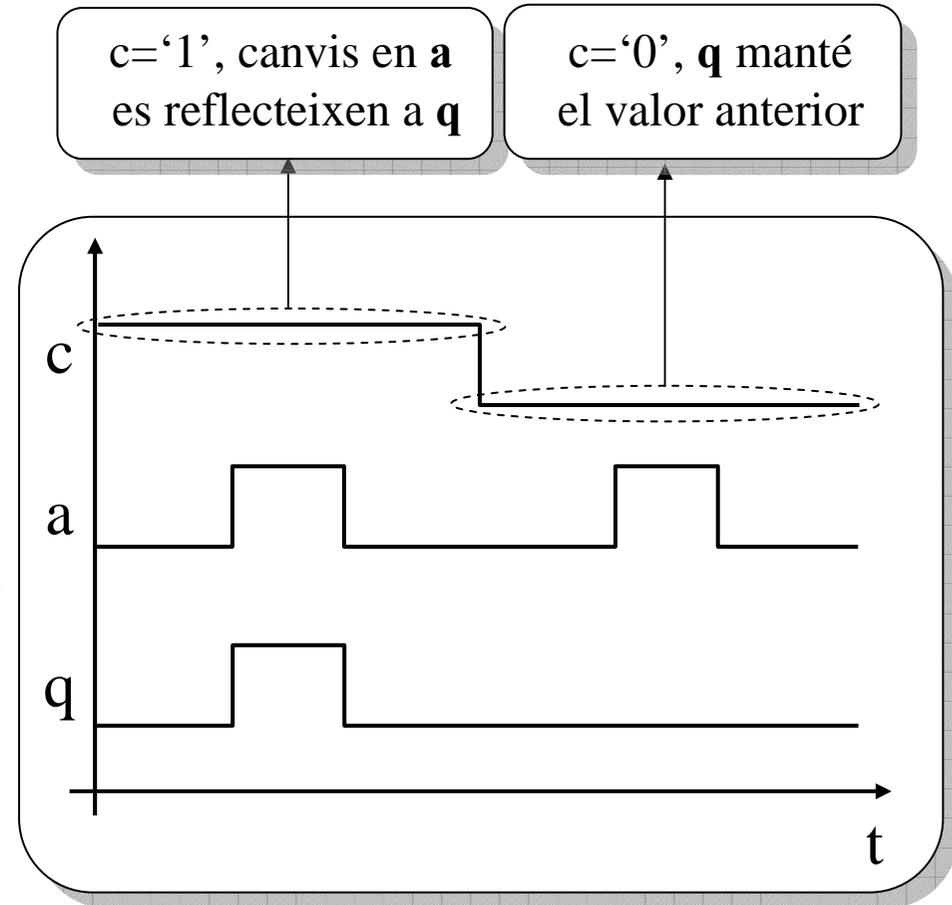
➤ **Processos:** Els canvis als senyals tenen lloc quan el procés finalitza i abans de la seva propera activació. L'activació del procés és conseqüència de canvis a senyals

• **Equivalència:**



• ***Memòria en processos:***

```
process(a,c)
begin
    if (c='1') then
        q <= a;
    end if;
end process;
```



- ***Regles generals de descripció:***

- **Lògica combinacional:**

- Assignacions concurrents (recomanat)
- Processos (llista de sensibilitat, cas per defecte)

- **Lògica seqüencial:**

- Processos amb primitiva **wait** ⇒ Flipflops
- Processos amb llista de sensibilitat ⇒ Latches

• ***Descripció de lògica combinacional:***

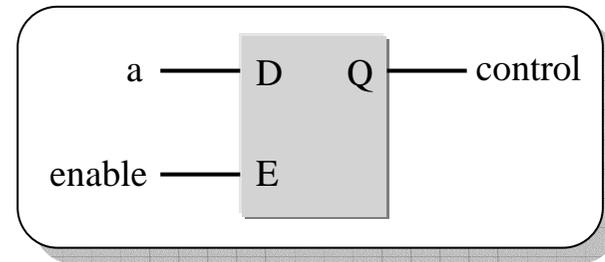
<i>n m ... b a</i>	<i>f</i>
0 0 ... 0 0	$f_0$
0 0 ... 0 1	$f_1$
0 0 ... 1 0	$f_2$
0 0 ... 1 1	$f_3$
...	
1 1 ... 1 1	$f_n$

$f <=$   $f_0$  when ( $n \& m \& \dots \& b \& a = "00\dots 00"$ )  
else  $f_1$  when ( $n \& m \& \dots \& b \& a = "00\dots 01"$ )  
else  $f_2$  when ( $n \& m \& \dots \& b \& a = "00\dots 10"$ )  
else  $f_3$  when ( $n \& m \& \dots \& b \& a = "00\dots 11"$ )  
...  
**else  $f_n$ ;**

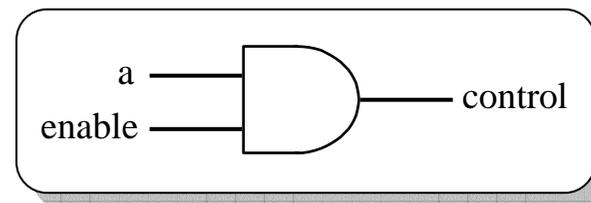
with ( $n \& m \& \dots \& b \& a$ ) select  
 $f <=$   $f_0$  when "00...00",  
 $f_1$  when "00...01",  
 $f_2$  when "00...10",  
 $f_3$  when "00...11",  
...  
 **$f_n$  when others;**

• ***Realimentació en assignacions concurrents:***

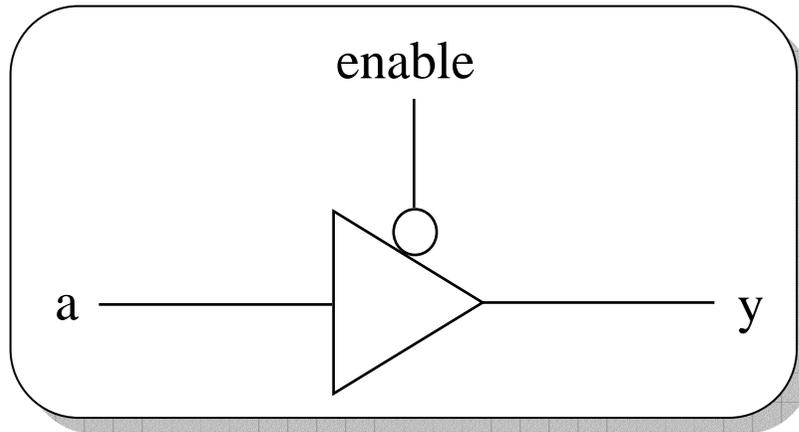
```
control <= a when (enable='1')  
else control;
```



```
control <= a when (enable='1')  
else '0';
```



• ***Buffers tri-estat:***



```
y <= a when (enable='0')  
     else 'Z';
```

Tipus `std_logic`  
Llibreria `ieee`

- ***Descripció de lògica seqüencial:***

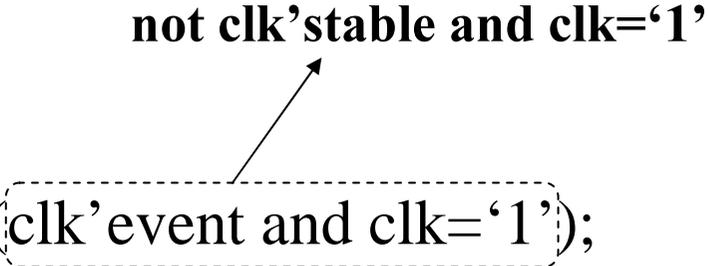
- **Definició de senyals de rellotge:**

- **Primitiva wait:**

```
process
begin
    wait until (clk'event and clk='1');
...

```

not clk'stable and clk='1'



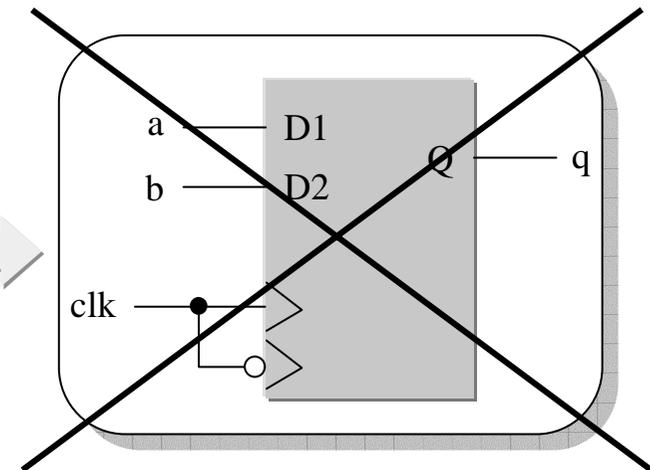
- **Llista de sensibilitat (senyals de control asíncrons):**

```
process(clk)
begin
    if (clk'event and clk='1')
...

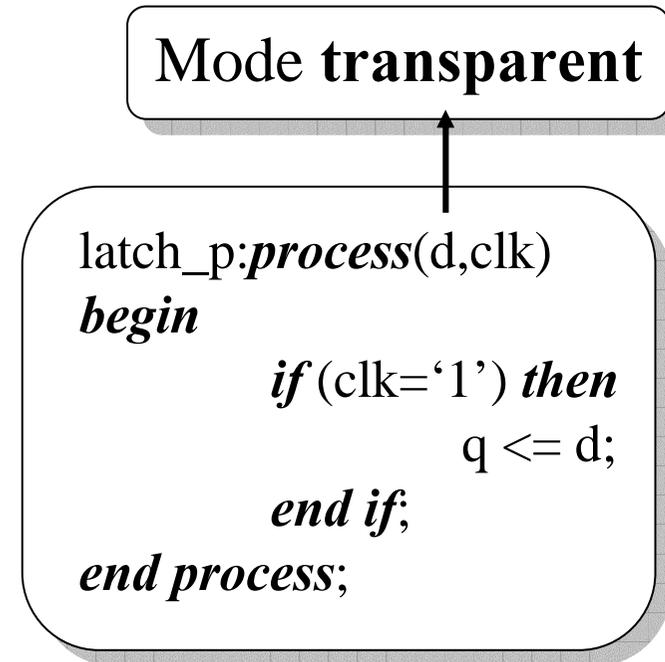
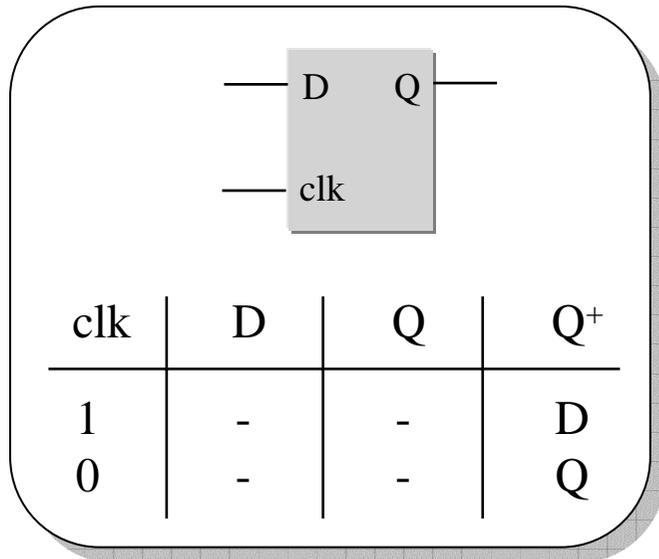
```

• ***Senyals de rellotge múltiples:***

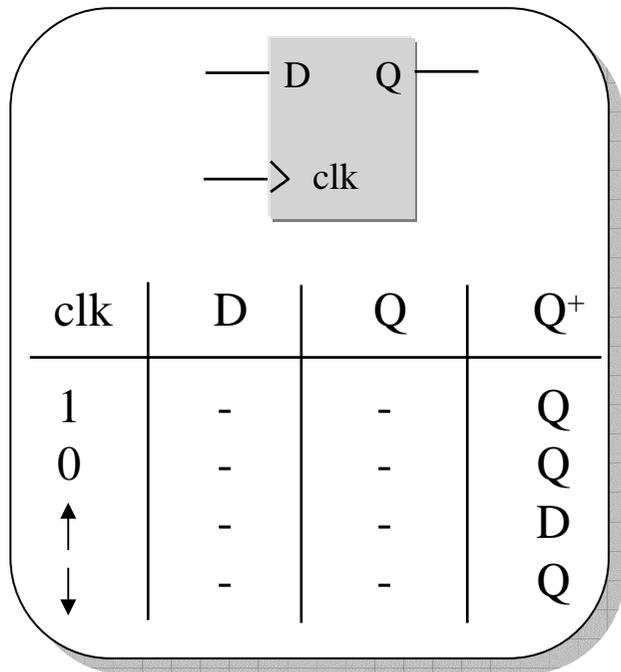
```
m_clock: process(clk)
begin
  if (clk'event and clk='0') then
    q <= a;
  elsif (clk'event and clk='1') then
    q <= b;
  end if;
end process;
```



- **Primitives seqüencials actives per nivell (latches):**



• **Primitives seqüencials actives per flanc (flipflops):**



```
flip_p1: process(clk)
begin
    if (clk'event and clk='1') then
        q <= d;
    end if;
end process;
```

```
flip_p2: process
begin
    wait until (clk'event and clk='1');
    q <= d;
end process;
```

***not (clk'stable) and clk='1'***

- ***Senyals de control:***

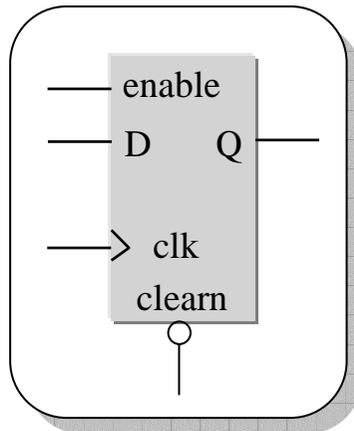
### ***Reset asíncron***

```
asinc_flip: process(clk,clearn)
begin
  if (clearn='0') then
    q <= '0';
  elsif (clk'event and clk='1') then
    q <= d;
  end if;
end process;
```

### ***Reset síncron***

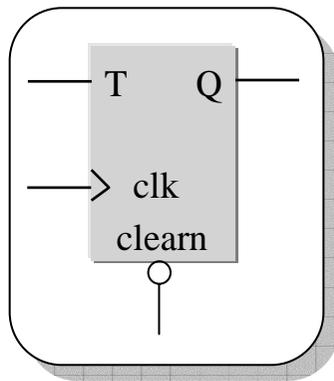
```
sinc_flip: process
begin
  wait until (clk'event and clk='0');
  if (clearn='0') then
    q <= '0';
  else
    q <= d;
  end if;
end process;
```

• ***Registre E (amb habilitació):***



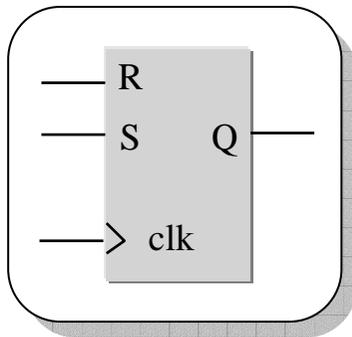
```
E_flipflop: process
begin
    wait until (clk'event and clk='1');
    if (clear='0') then
        Q <= '0';
    elsif (enable='1') then
        Q <= D;
    end if;
end process;
```

• ***Registre T (toggle):***



```
T_flipflop: process
begin
  wait until (clk'event and clk='1');
  if (clean='0') then
    Q <= '0';
  elsif (T='1') then
    Q <= not Q;
  end if;
end process;
```

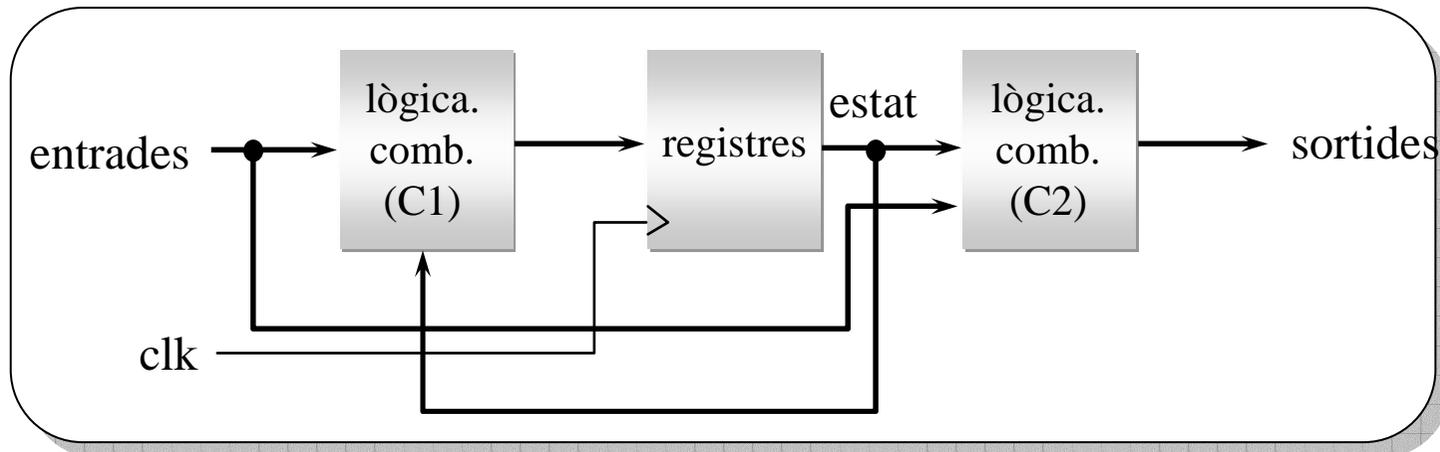
• ***Registre RS (Reset/Set):***



```
RS_flipflop: process
begin
  wait until (clk'event and clk='1');
  if (R='1') then
    Q <= '0';
  elsif (S='1') then
    Q <= '1';
  end if;
end process;
```

• ***Màquines d'estats finits:***

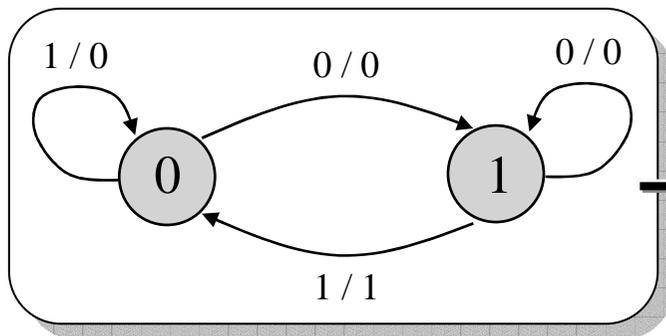
- Sistemes amb memòria (número d'estats) finita
- L'estat del sistema y les seves sortides depenen del seu estat anterior y de les entrades



• ***Estructura d'una màquina d'estats finits:***

```
architecture descripcio of fsm is
type estat is (s0, s1, ..., sn);
signal estat_actual, estat_futur: estat;
begin
C1 | estat_futur <=      s0 when (entrades="..." and estat_actual="...")
   |                   else s1 when ...
   |                   else ...;
   |
   | registres: process
   | begin
   |     wait until (clk'event and clk='1');
   |     if (clearn='0') then
   |         estat_actual <= s0;
   |     else
   |         estat_actual <= estat_futur;
   |     end if;
   | end process;
C2 | sortides <=      "..." when (entrades="..." and estat_actual="...")
   |                   else "..." when ...
   |                   else ...;
```

• **Exemple:** Detector de la seqüència “01”



```
architecture fsm of detector is
type estat is (s0, s1);
signal estat_actual, estat_futur: stat;
begin
    estat_futur <= s0 when (data_in='1')
    else s1;

    registres: process
    begin
        wait until (clk'event and clk='1');
        if (clear='0') then
            estat_actual <= s0;
        else
            estat_actual <= estat_futur;
        end if;
    end process;
    detecta <= '1' when ((estat_actual=s1) and
        (data_in='1'))
        else '0';
end fsm;
```

## ***2. Models de comportament de sistemes***

- **Variables y senyals:**

### ***Senyals***

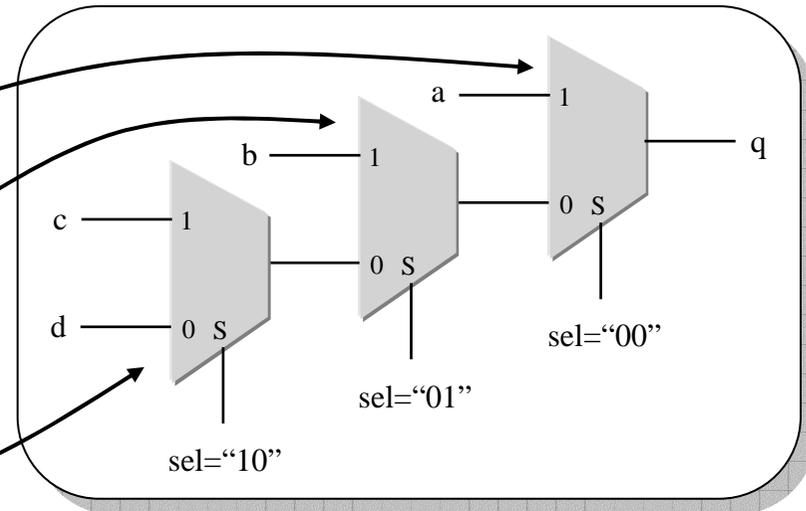
- Declarades en **architecture**
- Assignació **amb retard**
- Actualitzades **al final d'un procés**
- Simulació **lenta**

### ***Variables***

- Declarades en **process, block, function, procedure**
- Assignació **sense retard**
- Actualitzades **immediatament**
- Simulació **ràpida**

• **Primitiva if com a codificador amb prioritat (I):**

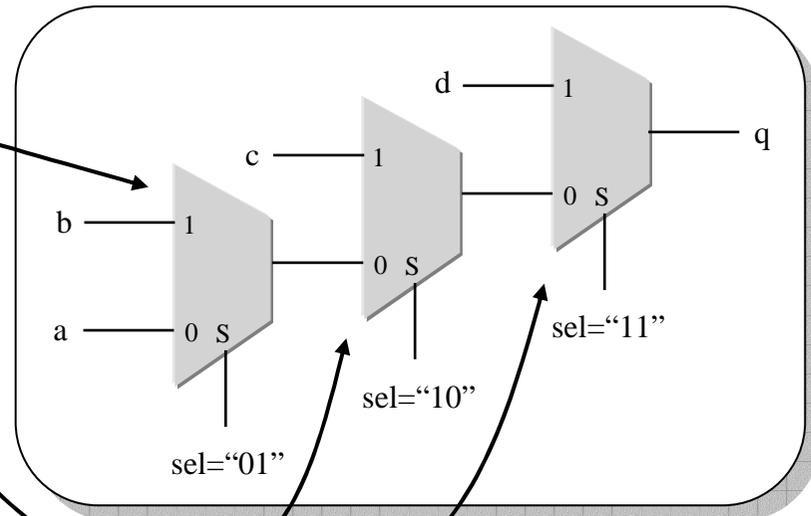
```
process (a,b,c,d,sel)
begin
  if (sel="00") then
    q <= a;
  elsif (sel="01") then
    q <= b;
  elsif (sel="10") then
    q <= c;
  else q <= d;
  end if;
end process;
```



```
q <=  a when (sel="00")
      else b when (sel="01")
      else c when (sel="10")
      else d;
```

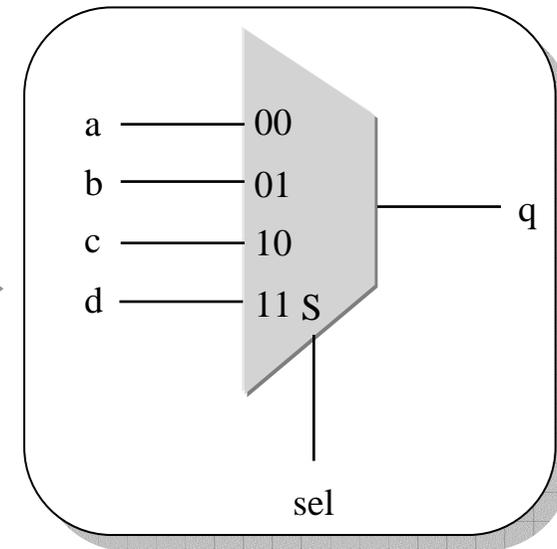
• **Primitiva if com a codificador amb prioritat (II):**

```
process (a,b,c,d,sel)
begin
    q <= a;
    if (sel="01") then
        q <= b;
    end if;
    if (sel="10") then
        q <= c;
    end if;
    if (sel="11") then
        q <= d;
    end if;
end process;
```



• **Primitiva case com a multiplexor:**

```
process (a,b,c,d,sel)
begin
  case sel is
    when "00" => q <= a;
    when "01" => q <= b;
    when "10" => q <= c;
    when others => q <= d;
  end case;
end process;
```

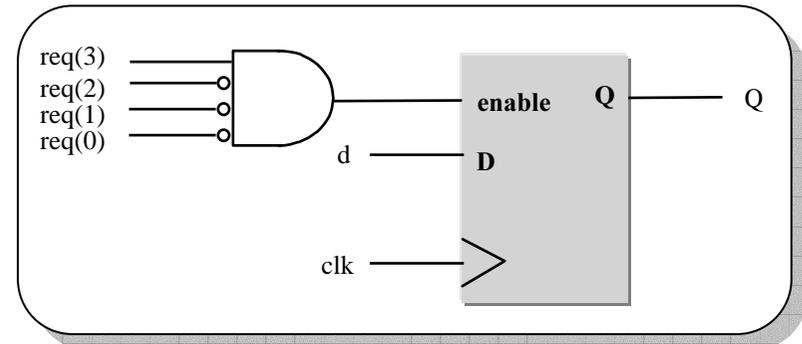


- **Case:** Descodificació complexa
- **If:** Selecció amb retard crític

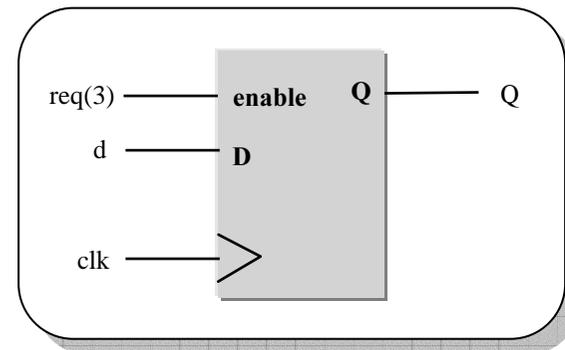
```
with sel select
  q <= a when "00",
      b when "01",
      c when "10",
      d when others;
```

• ***Descodificació incompleta:***

```
if (clk'event and clk='1')  
  if (req="1000") then  
    q <= d;  
  end if;  
end if;
```

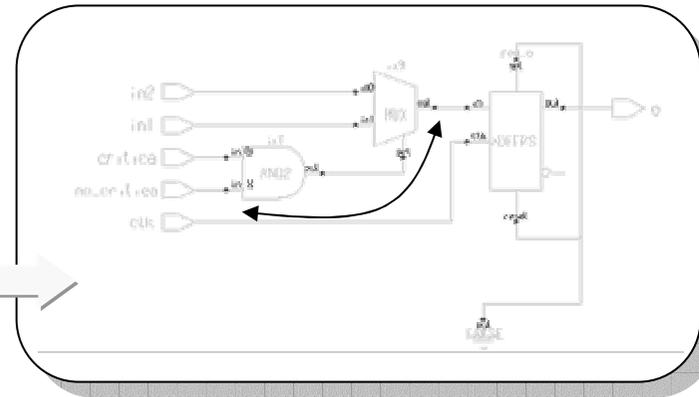


```
if (clk'event and clk='1')  
  if (req(3)='1') then  
    q <= d;  
  end if;  
end if;
```

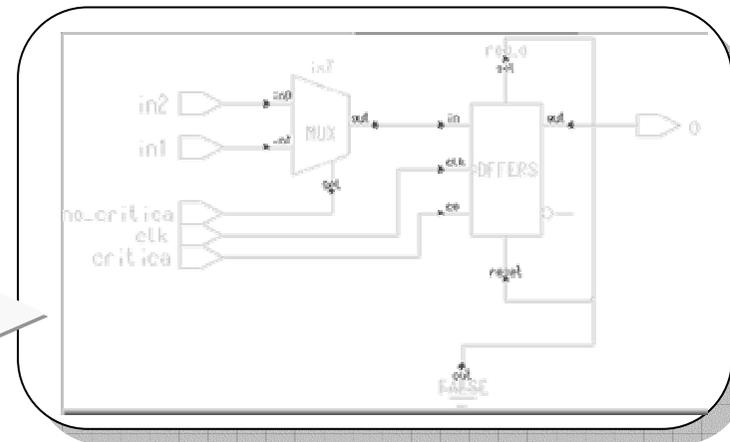


• ***Senyals amb retard crític:***

```
if (clk'event and clk='1') then
  if (critica='1' and no_critica='1') then
    o <= in1;
  else
    o <= in2;
  end if;
end if;
```

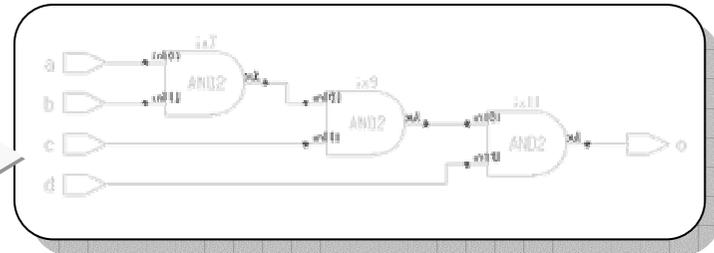


```
if (clk'event and clk='1') then
  if (critica='1') then
    if (no_critica='1') then
      o <= in1;
    else
      o <= in2;
    end if;
  end if;
end if;
```

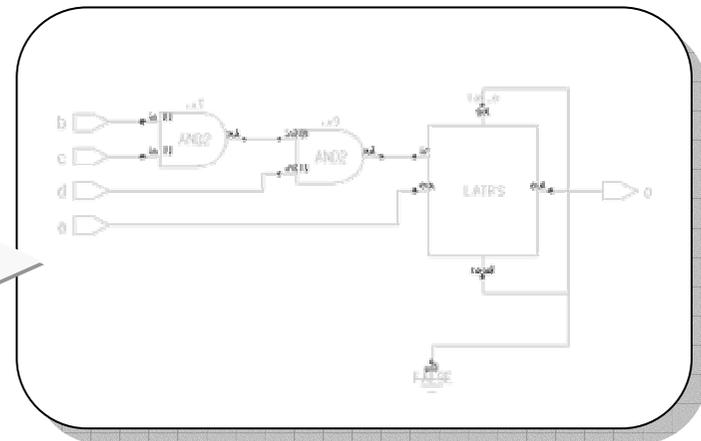


**• Inferència de latches (I):**

```
process(a,b,c,d)
begin
  if (a='1' and b='1' and c='1' and d='1') then
    o <= '1';
  else
    o <= '0';
  end if;
end process;
```

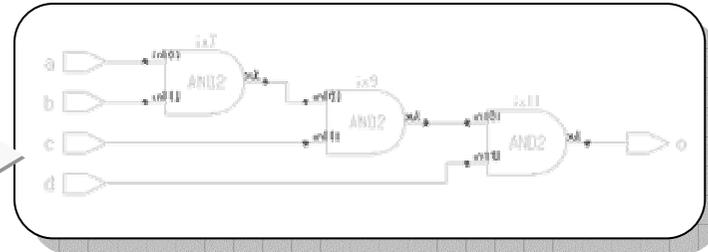


```
process(a,b,c,d)
begin
  if (a='1') then
    if (b='1' and c='1' and d='1') then
      o <= '1';
    else
      o <= '0';
    end if;
  end if;
end process;
```

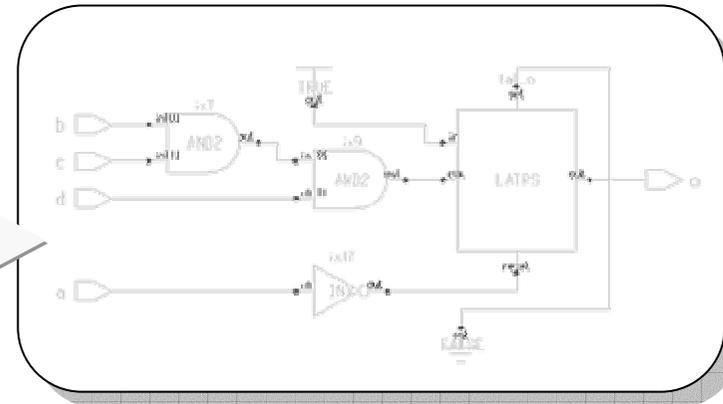


**• Inferència de latches (II):**

```
process(a,b,c,d)
begin
  if (a='1' and b='1' and c='1' and d='1') then
    o <= '1';
  else
    o <= '0';
  end if;
end process;
```

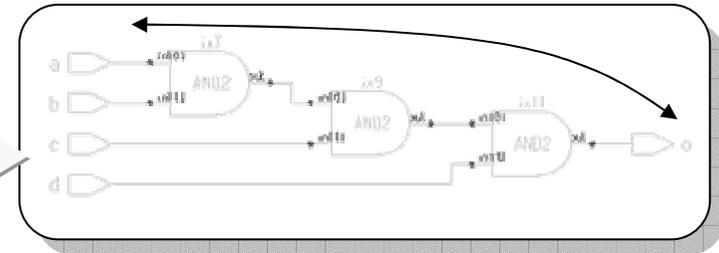


```
process(a,b,c,d)
begin
  if (a='1') then
    if (b='1' and c='1' and d='1') then
      o <= '1';
    end if;
  else
    o <= '0';
  end if;
end process;
```

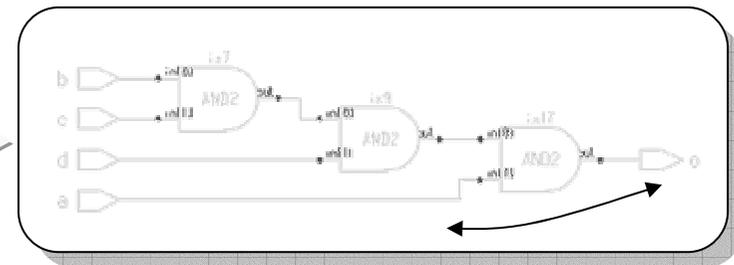


**• Priorització de senyals crítics (I):**

```
process(a,b,c,d)
begin
  if (a='1' and b='1' and c='1' and d='1') then
    o <= '1';
  else
    o <= '0';
  end if;
end process;
```

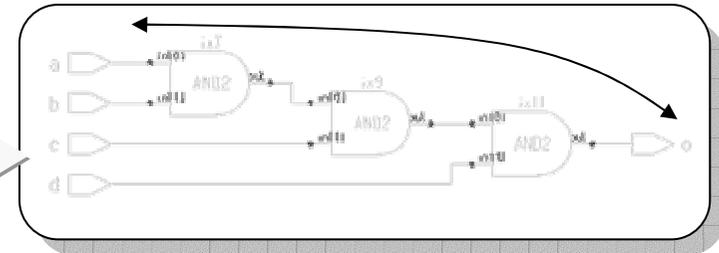


```
process(a,b,c,d)
begin
  if (a='1') then
    if (b='1' and c='1' and d='1') then
      o <= '1';
    else o <= '0';
    end if;
  else
    o <= '0';
  end if;
end process;
```

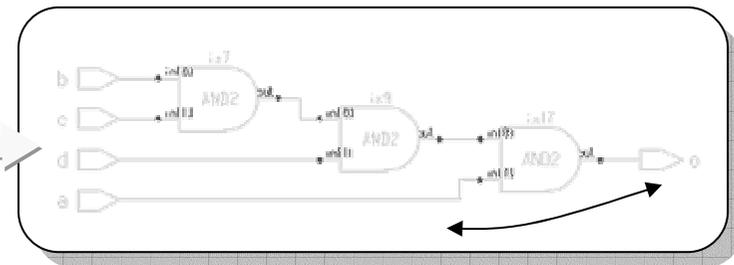


• ***Priorització de senyals crítics (II):***

```
process(a,b,c,d)
begin
  if (a='1' and b='1' and c='1' and d='1') then
    o <= '1';
  else
    o <= '0';
  end if;
end process;
```

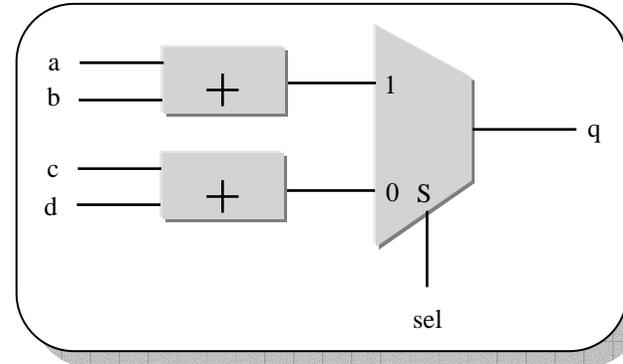


```
process(a,b,c,d)
begin
  if (a='1' and (b='1' and c='1' and d='1')) then
    o <= '1';
  else
    o <= '0';
  end if;
end process;
```

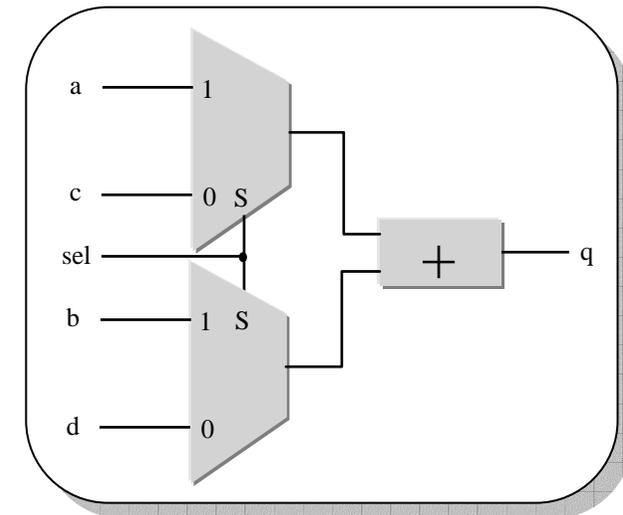


• ***Compartició de recursos:***

```
if (sel)
    q <= a+b;
else
    q <= c+d;
end if;
```



```
if (sel)
    op1 <= a;
    op2 <= b;
else
    op1 <= c;
    op2 <= d;
end if;
q <= op1+op2;
```



• ***Planificació d'operacions:***

➤ **sel1, sel2 mutuament exclusives**

```
if (sel1) then
    q <= a+b;
elsif (sel2) then
    q <= a+c;
end if;
```

```
if (sel1) then
    q <= a+b;
else
    if (sel2) then
        q <= a+c;
    end if;
end if;
```

```
if (sel1) then
    q <= a+b;
end if;
if (sel2) then
    q <= a+c;
end if;
```

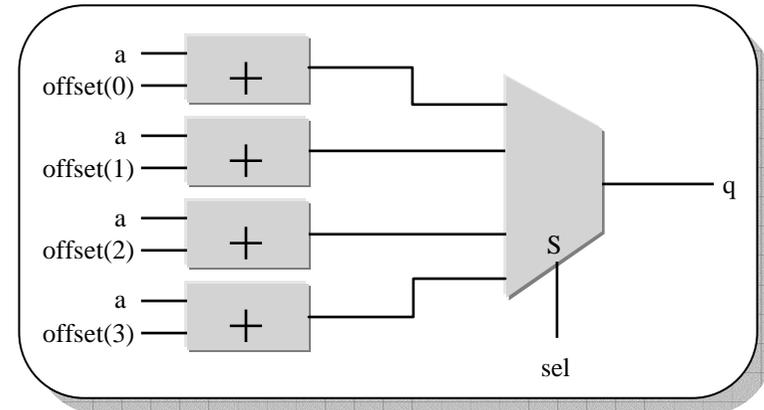
- 1 nivell de control
- 1 sumador

- 1-2 nivells de control
- 1 sumador

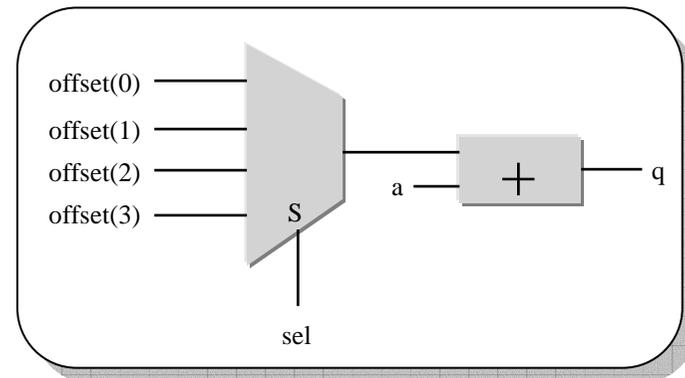
- 3-4 nivells de control
- 1-2 sumador(s)

• ***Operacions en bucles:***

```
for i in 0 to 3 loop  
  if (sel(i)='1') then  
    q <= a + offset(i);  
  end if;  
end loop;
```

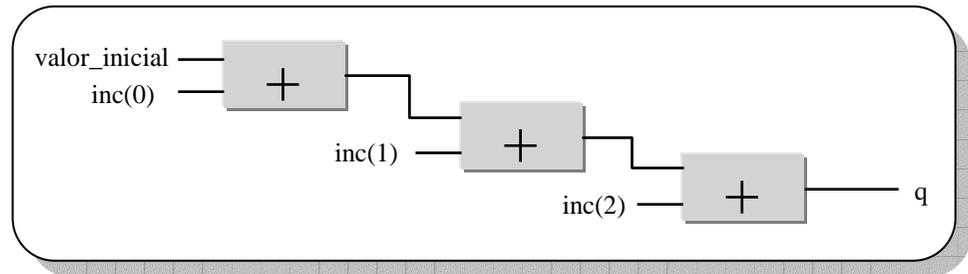


```
for i in 0 to 3 loop  
  if (sel(i)='1') then  
    add_offset <= offset(i);  
  end if;  
end loop;  
q <= a+add_offset;
```

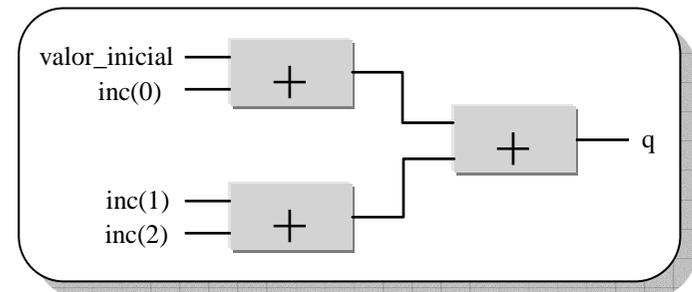


• ***Expansió de bucles:***

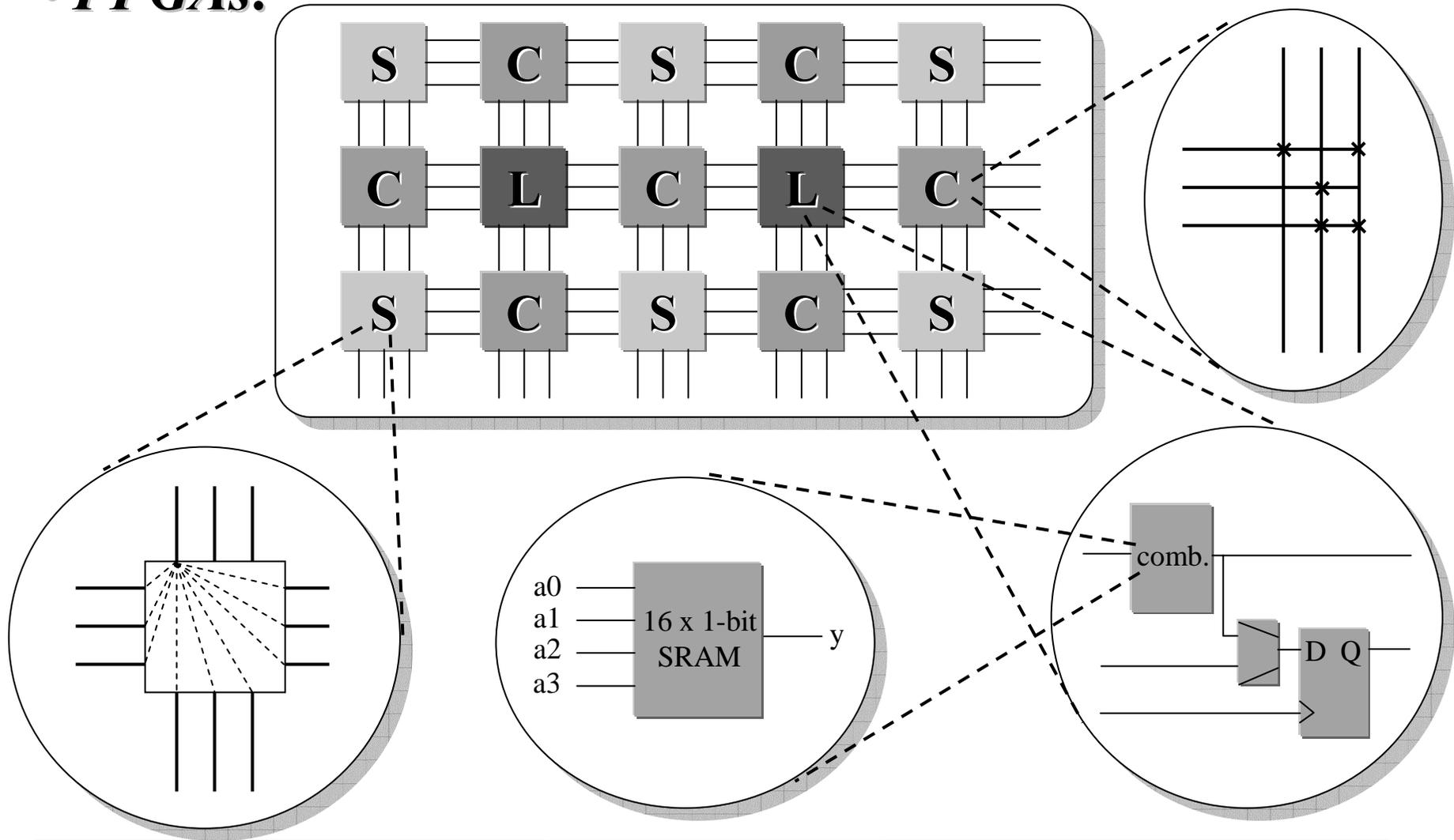
```
q <= valor_inicial;  
for i in 0 to 2 loop  
  q <= q+inc(i);  
end loop;
```



```
q <= ((valor_inicial+inc(0)) +  
      (inc(1)+inc(2)));
```



• **FPGAs:**



• ***Principis de síntesi per a FPGAs:***

➤ **Recursos limitats:**

- elements combinacionals
- elements seqüencials
- connectivitat

➤ **Retard d'execució:**

$$t_{total} = t_{component} + t_{routing} + t_{switch}$$

$t_{routing}, t_{switch} \gg t_{component}$ , i  
desconeguts abans del disseny físic (P & R)

Valors a  
minimitzar

• ***Minimització del retard de connexió:***

➤ **Utilitzar macros hardware específiques:**

- Sumadors / restadors
- Desplaçadors / rotadors
- Comptadors
- Registres de desplaçament
- Memòries

➤ **Codificació one-hot per a màquines d'estats finits**

### 3. El paquet estàndard de síntesi IEEE 1076.3-1997

- Interpretació per a la síntesi dels tipus *BIT*, *BOOLEAN* i *STD\_ULOGIC*
- Definició de la funció *STD\_MATCH*
- Definició de funcions per tractar flancs de senyals
- Definició de funcions per representar tipus amb i sense signe, així com funcions de conversió entre tipus

➤ **type** *STD\_ULOGIC* **is** ('1', '0', 'H', 'L', 'U', 'X', 'W', '-', 'Z');

resolució  
↓  
*STD\_LOGIC*

valors lògics      valors metalògics

- **Interpretació de valors lògics:**

- **Interpretació com a valor 0:**

- El valor '0' del tipus BIT
    - El valor FALSE del tipus BOOLEAN
    - Els valors '0' y 'L' del tipus STD\_ULOGIC

- **Interpretació com a valor 1:**

- El valor '1' del tipus BIT
    - El valor TRUE del tipus BOOLEAN
    - Els valors '1' y 'H' del tipus STD\_ULOGIC

- **Interpretació de valors metalògics:**

- **Expresions de relació:** En una relació de igualtat, si un operand és un valor metalògic i l'altre no és un valor estàtic, el resultat hauria de ser el valor FALSE

- **Interpretació de valors metalògics:**

- **Primitives *Case*:** Si en una selecció intervé un valor, l'eina de síntesi interpretarà que l'esmentada selecció no es produirà mai
- **Operacions aritmètiques, lògiques i de desplaçament:** L'eina de síntesi tractarà l'operació com a un error

- **Interpretació del valor alta impedància ('Z'):**

- Si el valor 'Z' apareix a una assignació, l'eina de síntesi la convertirà en un buffer tri-estat
- Si el valor 'Z' apareix a una construcció diferent a una assignació, l'eina de síntesi el tractarà com un valor metalògic

- **La funció STD-MATCH:**

- Si aquesta funció s'aplica a dos arguments de tipus STD\_ULOGIC, tornarà el valor TRUE si:
  - Ambdós valors són iguals, o
  - Un valor és '0' i l'altre és 'L', o
  - Un valor és '1' i l'altre és 'H', o
  - Com a mínim un dels valors és '-'
- Si aquesta funció s'aplica a dos arguments que són vectors de tipus STD\_ULOGIC, tornarà el valor TRUE si:
  - Ambdós arguments tenen la mateixa longitud i
  - STD\_MATCH aplicada a cada parella d'elements a la mateixa posició torna el valor TRUE

- **Detecció de flancs: *rising\_edge(senyal)*, *falling\_edge(senyal)***

- **Paquets aritmètics estàndard:**

- **NUMERIC\_BIT:** Basat en el tipus BIT
- **NUMERIC\_STD:** Basat en el tipus STD\_LOGIC

- **Nous tipus:**

- **UNSIGNED:** Sencer sense signe, amb el bit més significatiu a l'esquerra
- **SIGNED:** Sencer en complement a 2, amb el bit més significatiu a l'esquerra

#### ***4. Estàndard de síntesi a nivell RTL (IEEE 1076.6)***

- **Aprovat el Juliol de 1998 (IEEE Std 1076.6-1999)**
- **Motivació de VHDL: Modelat de sistemes electrònics**

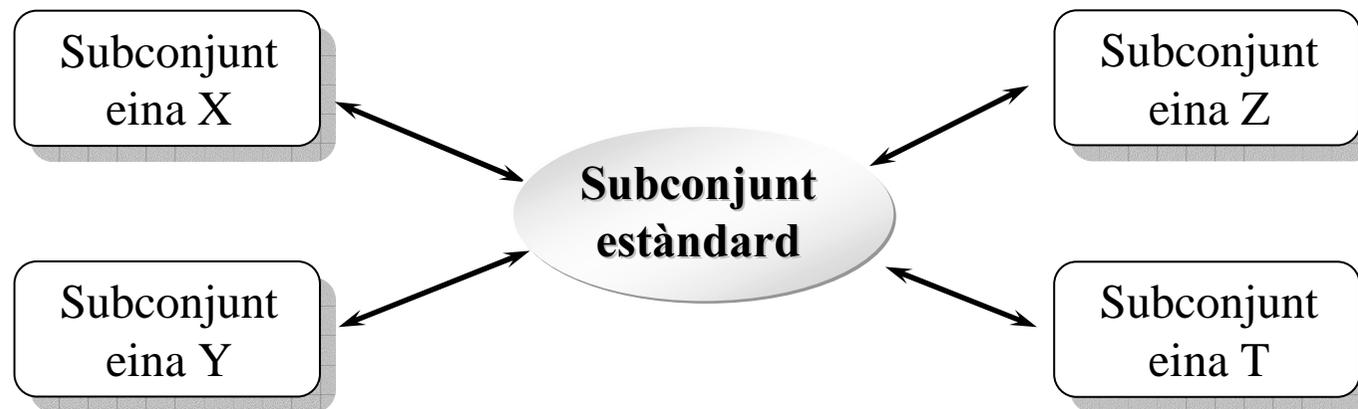
***Flexibilitat sintàctica i semàntica***

- **Impossible cobrir tot el llenguatge**
- **Síntesis depenent de les eines**

**Síntesi  
*no portable***

- **Motivació per a un subconjunt estàndard de síntesi:**

- **Portabilitat entre eines**



- **Portabilitat del disseny (llenguatge d'especificació):**



- ***Estàndard de síntesi a nivell RTL (IEEE 1076.6)***

- **Descripció RTL:**

- Transferències de dades entre elements amb memòria (registres)
- Lògica combinacional

- **Tipus de dades suportats:**

- BIT, BOOLEAN, BIT\_VECTOR, CHARACTER, STRING, INTEGER (1076-2000)
- STD\_ULOGIC, STD\_ULOGIC\_VECTOR, STD\_LOGIC, STD\_LOGIC\_VECTOR (1164-1993)
- SIGNED, UNSIGNED (1076.3-97)

- ***Primitives seqüencials actives per flanc (flipflops):***

- **Senyal de rellotge:** Tipus BIT, STD\_ULOGIC, STD\_LOGIC

- **Especificació d'un flanc (pujada) de rellotge:**

- **Amb la primitiva *if*:**

- **rising\_edge**(rellotge)

- rellotge'**event** and rellotge='1'

- rellotge='1' and rellotge'**event**

- **not** rellotge'**stable** and rellotge='1'

- rellotge='1' and **not** rellotge'**stable**

➤ **Especificació d'un flanc (pujada) de rellotge:**

• **Amb la primitiva *wait until*:**

- **rising\_edge**(rellotge)
- rellotge='1'
- rellotge'**event** and rellotge='1'
- rellotge='1' and rellotge'**event**
- **not** rellotge'**stable** and rellotge='1'
- rellotge='1' and **not** rellotge'**stable**

➤ Només es permet **un flanc de rellotge per procés**

➤ La primitiva **wait until** no està permesa dins de procedures

• **Modelat:**

```
process(rellotge)
begin
  if (rellotge'event and rellotge = '1') then
    q <= d;
  end if;
end process;
```

```
process
  variable var: unsigned(3 downto 0);
begin
  wait until (rellotge = '1');
  var := var + 1;
  count <= var;
end process;
```

```
process
begin
  wait until (rellotge = '1');
  q <= d;
end process;
```

- **Una** per procés
- **Primera** sentència

Variable llegida abans  
de ser assignada

- **Senyals de control asíncrons:**

```
process(clk, set, reset)
begin
    if (reset='1') then
        Q <= '0';
    elsif (set='1') then
        Q <= '1';
    elsif (clk'event and clk='1') then
        Q <= D;
    end if;
end process;
```

• ***Primitives seqüencials actives per nivell (latches):***

➤ **Inferència necessària:**

- Un senyal o variable s'assigna a un procés sense definició de senyals de rellotge i
- Hi han execucions del procés que no impliquen una assignació explícita sobre el senyal o variable

➤ **Inferència probable:**

- Un senyal o variable s'assigna a un procés sense definició de senyals de rellotge i
- Hi han execucions del procés a les quals el valor del senyal o variable és llegit abans de la seva assignació

- **Buffers tri-estat:** Assignació condicional del valor 'Z' sobre un senyal o variable

- **Modelat de lògica combinacional:**

- Assignacions concurrents
- Assignacions dins d'un procés que tenen lloc **sempre** que el procés s'executa

- **Meta-comentaris:**

```
-- RTL_SYNTHESIS ON  
-- RTL_SYNTHESIS OFF
```

- **Atribut ENUM\_ENCODING:** Utilitzat per especificar a l'eina de síntesi la codificació a utilitzar per als elements que componen un tipus enumerat

```
attribute ENUM_ENCODING: string;
```

```
type estat is (s0, s1, s2, s3);
```

```
attribute ENUM_ENCODING of estat: type is  
"0001 0010 0100 1000";
```