4. Diseinatzeko moduak

4.1. Egiturazko estiloa

Diseinatu nahi den sistema azpizirkuitutan banatzen denean, eta haien arteko konexioak adierazten direnean, *egiturazko estiloa* dugu. Azpizirkuitu bakoitza *osagai (component)* eta *osagaiaren deklarazioa*rekin deklaratzen da. Hala, azpizirkuituaren izena eta sarrera-eta irteera-portuak ezagunak izango ditugu.

```
COMPONENT osagaiaren izena
PORT (port_list);
END osagaiaren izena;
```

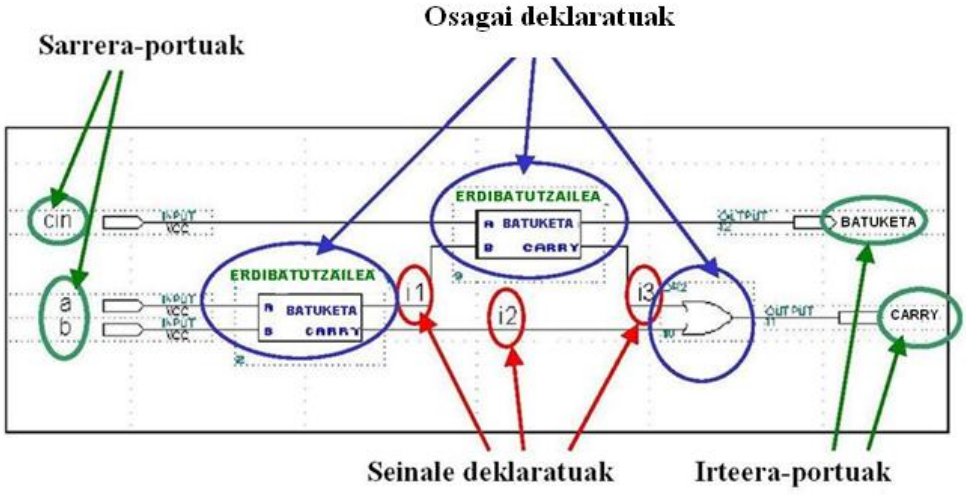
Osagaiak deklaratzeko lekua da, arkitekturaren barnean, eta BEGIN sententziaren aurretik. Deklarazio hori entitate baten deklarazioaren antzekoa da, hots, portaera azaldu ordez, kanporanzko terminalak soilik adierazten dira. Portaera hori deklarazioa egiten den arkitekturatik kanpo definitu behar da. Horretarako, osagaiaren izen berdina duen entitatea definitu behar da, eta baita osagai horri dagokion arkitektura bat ere.

Azpizirkuitua deskripzioan sartzeko esaldi hau erabiltzen da:

```
etiqueta : osagai_izena PORT MAP (ports_zerrenda);
```

Horiek horrela, programaren kodea irakurtzearekin ezin da ondorioztatu zirkuituaren operatibitatea; izan ere, kode horretan osagaiak eta haien arteko loturak besterik ez da agertzen. Hortaz, ulergarria izan dadin, bloke bakoitzaren operatibitatea ezagutu behar da.

Adibidea: bit bakarreko batutzailea



79. Irudia

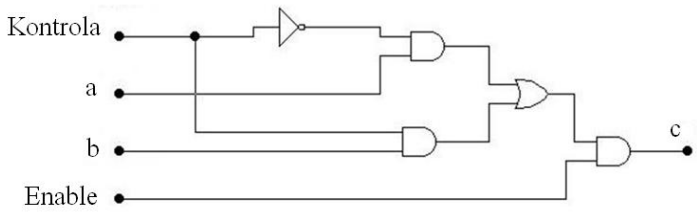
```
ENTITY batutzaile IS
PORT( a, b, cin : IN BIT;
batuketa, carry : OUT BIT);
END batutzaile;
ARCHITECTURE batuketa OF batutzaile IS
SIGNAL i1, i2, i3 : BIT;
-- Erdibatutzaile osagaiaren deklarazioa
COMPONENT erdibatutzailea
PORT (a, b : IN BIT; batuketa, carry : OUT BIT);
END COMPONENT;
-- OR atea osagai moduan
COMPONENT or_atea
PORT (a, b : IN BIT; z : OUT BIT);
END COMPONENT;
BEGIN
u1 : erdibatutzailea PORT MAP (a, b, i1, i2);
u2 : erdibatutzailea PORT MAP (i1, cin, batuketa, i3);
u3 or_atea PORT MAP (i2, i3, carry);
END batuketa;
-- Erdibatutzaile osagaiaren definizioa
ENTITY erdibatutzailea IS
PORT (a, b : IN BIT;
batuketa, carry : OUT BIT);
END erdibatutzailea;
ARCHITECTURE erdi OF erdibatutzailea IS
BEGIN
batuketa <= a XOR b;
carry <= a AND b;
END erdi;
--or_atea osagaiaren definizioa
ENTITY or_atea IS
PORT (a, b : IN BIT;
z : OUT BIT);
END or_atea;
```

```
ARCHITECTURE or_atea OF or_atea IS
BEGIN
z <= a OR b;
END or_atea;
```

4.2. Datu-fluxuzko estiloa

Gailuaren funtzionalitatea adierazteko konkurrente moduan exekutatu behar direnean ekuazioak, datu-fluxu estilo deritzogu. Ekuazio horiek eragile aritmetiko eta logikoak erabiliz osatzen dira, eta lortutako adierazpena seinale bati esleitzen zaio.

Gomendatzen da diseinuak sinpleak direnean soilik erabiltzea. Adibide moduan, 80. Irudian agertzen den zirkuituaren deskripzioa egiten da:



80. Irudia

```
entity multi is port(
a,b     : in  bit;
kontrola : in  bit;
enable  : in  bit;
c       : out bit
);
end multi;
architecture archmul of multi is
signal aux1,aux2,aux3: bit;
begin
aux1 <= b and (not(kontrola));
aux2 <= a and kontrola;
aux3 <= aux1 or aux2;
c    <= enable and aux3;
end archmul;
```

4.3. Estilo sekuentziala

Sententzia sekuentzialak eta ez-konkurrenteak erabiltzen direnean, estilo sekuentziala dugu. Sententzia sekuentzialak sententzia konkurrenteetatik bereizita jarri behar dira Prozesuak (*Process*) erabiliz, eta Arkitekturaren gorputzean agertu behar dute.

Nahitaezkoa da IF, CASE eta LOOP sententziak Prozesu baten barruan jartzea.

Prozesu baten itxura hau izango da:

Aukerako_etiketa : PROCESS (aldagai-zerrenda edo ebentoa) deklarazioak
BEGIN
Sententzia sekuentzialak
END PROCESS Aukerako_etiketa;

Adibidez:

```
OR_atea: PROCESS (a, b)
BEGIN
IF a='1' or b='1' THEN
z <= '1';
ELSE
z<= '0';
END IF;
END PROCESS;
```

Prozesua exekutatzeko beharrezkoak diren seinaleak *aldagai-zerrendan* adierazten dira, hau da, zeintzuk aldagai aldatu behar diren prozesua exekutatu dadin.

Prozesuan erabiltzen diren aldagaiak, prozesuaren barnean bakarrik existitzen dira eta haiei dagokien balioa mementoan esleitzen zaie. Seinaleek, berriz, prozesua bukatzen denean aldatzen dute balioa.

Prozesu batean *aldagai-zerrendarik* agertzen ez bada, bukaerarik gabeko begizta bat lortzen da; hortaz, prozesua etengabe errepikatuko litzateke. Orduan, prozesua geldiarazteko modua *Wait* sententzia da.

```
PROCESS
BEGIN
IF (alarma_ordua = oraingo_ordua) THEN
alarma_soinua <= '1';
ELSE
alarma_soinua <= '0';
```

```
END IF;
WAIT ON alarma_ordua, oraingo_ordua;
END PROCESS;

PROCESS (alarma_ordua, oraingo_ordua)
BEGIN
IF (alarma_ordua = oraingo_ordua) THEN
alarma_soinua <= '1';
ELSE
alarma_soinua <= '0';
END IF;
END PROCESS;
```