

S3. Proiectarea arhitecturilor ierarhice

Instanțierea modulelor în limbajul Verilog

1. Obiective

O1. Însușirea tehnicii de a instanția un modul în limbajul Verilog

O2. Implementarea unui dispozitiv decodificator în Verilog HDL

O3. Construirea unui design ierarhic

2. Considerații teoretice

2.1 Design-ul ierarhic

Vom sublinia faptul că un design ierarhic:

- facilitează design-ul unor arhitecturi complexe
- încurajează refolosirea design-ului proiectat

2.1.1 Instanța

Instanța poate fi definită ca un modul folosit ca o componentă într-un modul mai cuprinzător.

O instanță are:

- Un **modul**: furnizează definiția pentru instanță.
- Un **recipient**: modulul în care instanța este creată.

Procedura prin care creăm o nouă instanță se numește **instanțiere**.

O instanță este construită prin următoarele elemente:

- modulul care trebuie instanțiat, urmat de
- numele instanței (care diferă de alte instanțe ale aceluiași modul), urmat de
- o listă de porturi

Lista de porturi specifică la care semnale din recipient sunt legate porturile din instanță.

O conexiune de port este definită cel mai simplu astfel:

`.<module_port>(<container_signal>),` în care:

- **module_port** este un port al modulului instanțiat
- **container_signal** este un semnal intern sau un port al recipientului

2.1.2 Arhitectura unui multiplexor 4-la-1

Exercițiu:

Se cere construirea unui multiplexor 4-la-1 format din multiplexoare 2-la-1.

Soluție:

Arhitectura unui multiplexor 4-la-1 folosind trei module de multiplexoare 2-la-1 este ilustrată mai jos. Intrarea de date, **d**, este pe 4 linii iar linia de selecție **s**, pe 2 biți, propagă 1 bit de la intrarea ***d*** către ieșirea ***o***.

Spre exemplu, dacă s = (1,0), linia d[2] este propagată către *o.*

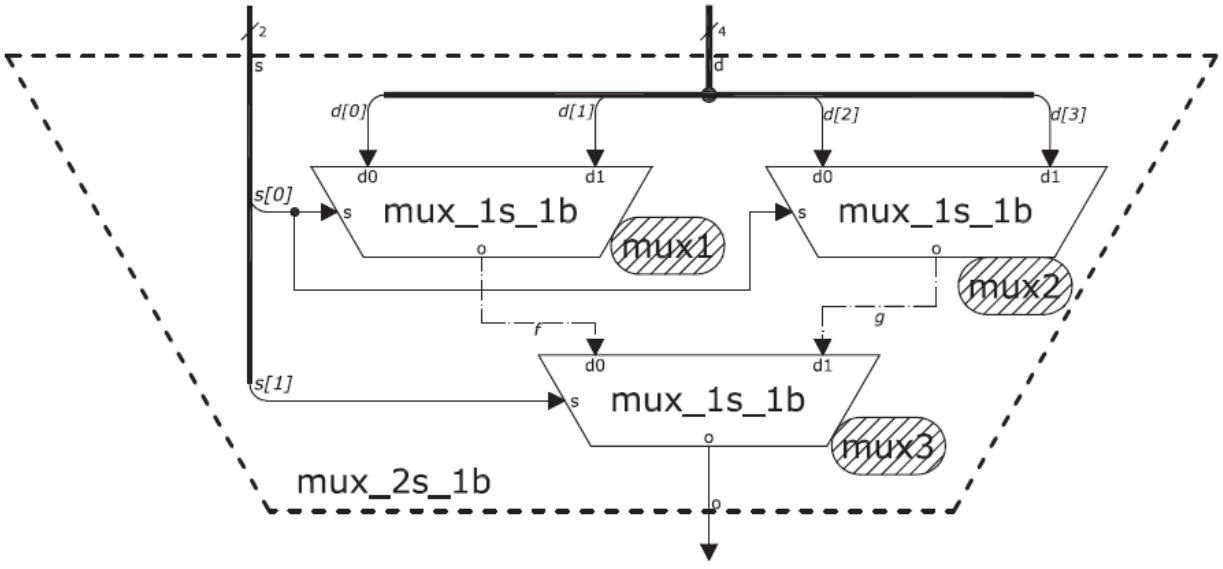

Fig. 1 - Design ierarhic multiplexor 4-la-1

Codul Verilog care implementează arhitectura unui multiplexor 4-la-1 este prezentat mai jos:

```
module mux_1s_1b (
input [3:0] d,
input [1:0] s,
output o
);

wire f;
wire g;

mux_1s_1b mux1 (
.d0(d[0]) ,
.d1(d[1]) ,
.s(s[0]) ,
.o(f)
);

mux_1s_1b mux2 (
.d0(d[2]) ,
.d1(d[3]) ,
.s(s[0]) ,
.o(g)
);

mux_1s_1b mux3 (
 .d0(f) ,
.d1(g) ,
.s(s[1]) ,
.o(o)
);
endmodule
```

Analizând cu atenție prima instanță creată în codul multiplexorului 4-la-1, putem observa componentele unei instanțieri:

- modulul care urmează a fi instanțiat este **mux_1s_1b** (obligatoriu trebuie să fie numele unui modul Verilog existent)
- numele instanței este **mux1** (trebuie să fie un nume Verilog valid)
- lista porturilor de conexiune, poziționate între **paranteze rotunde** (mai multe detalii mai jos pe pagina curentă)

```
mux_1s_1b mux1 (
.d0(d[0]),
.d1(d[1]),
.s(s[0]),
.o(f)
);
```

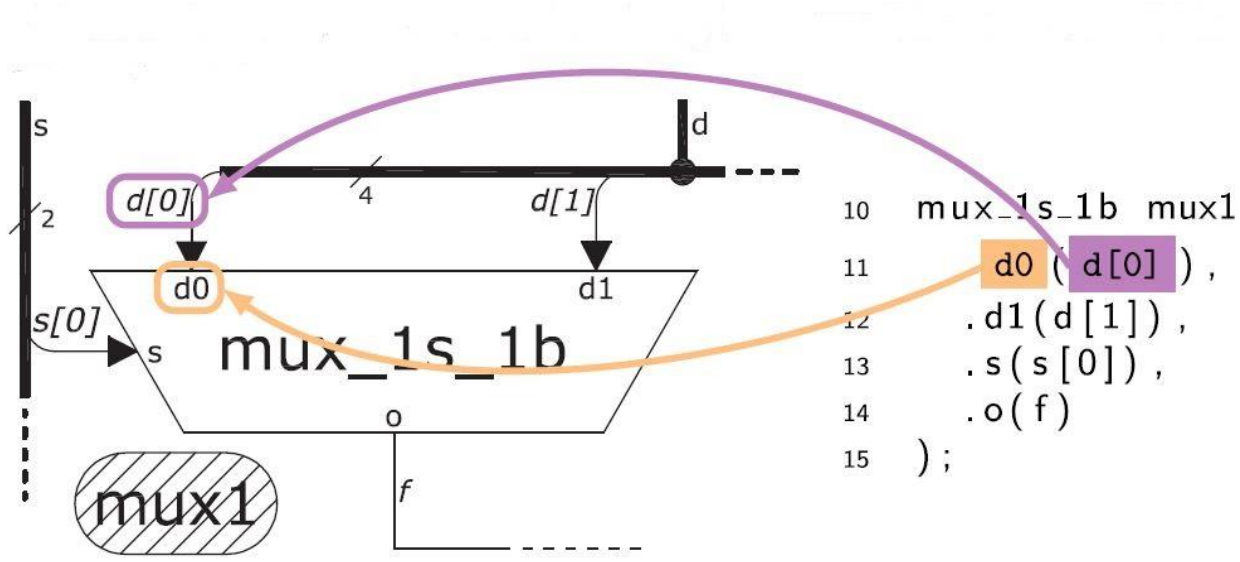

Fig. 2 - Instanțierea modulului mux1

Elementele primului port de conexiune sunt:

- **d0** – port a modulului mux_1s_1b care urmează a fi asociat cu
- **d[0]** – semnal în interiorul recipientului mux_2s_1 (d[0] este un port a unui recipient, astfel un semnal în recipient) la care este conectat **d0**

Legăturile interne de tip **wire** conectează instanțele interne:

- anexează o ieșire a instanței către o intrare a altei instanțe.
- trebuie să fie declarate în recipientul modulului ca și elemente de tip ***wire***

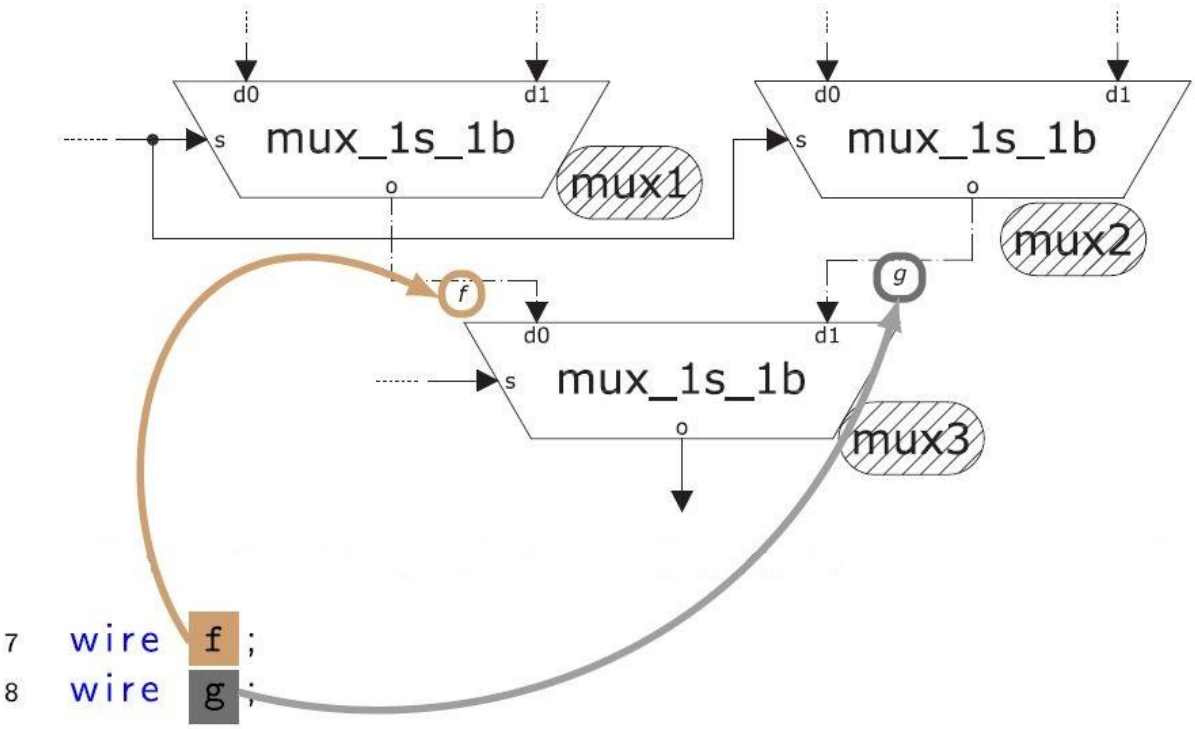

Fig. 3 – Declararea firelor fizice interne ale arhitecturii mux_2s_1b

Important: Atunci când conectăm un wire la un port, lățimile (dimensiunile) lor trebuie să fie identice !

Dacă cele două porturi care trebuiesc conectate au dimensiuni diferite, alegeți pentru dimensiunea firului fizic care urmează să fie conectat o lățime a uneia dintre ele. Pentru a conecta firul la portul dorit:

- dacă firul fizic are mai puține linii iar celălalt port este o ieșire, unele dintre liniile de ieșire vor rămâne neconectate
- dacă firul fizic are mai puține linii iar portul la care se conectează este o intrare, folosiți operația de concatenare pentru a furniza liniile de intrare necesare
  ex: **`.x({4'd0, data}),`**
- dacă firul dispune de mai multe linii iar portul antagonic reprezintă o intrare, folosiți part-select pentru a furniza doar liniile de intrare necesare
  ex: **`.s(addr[15:14]),`**

- dacă firul are mai multe linii iar portul la care se conectează este o ieșire, folosiți part-select pentru a furniza liniile de ieșire necesare
  ex: **`.z(next_addr[7:0]),`**

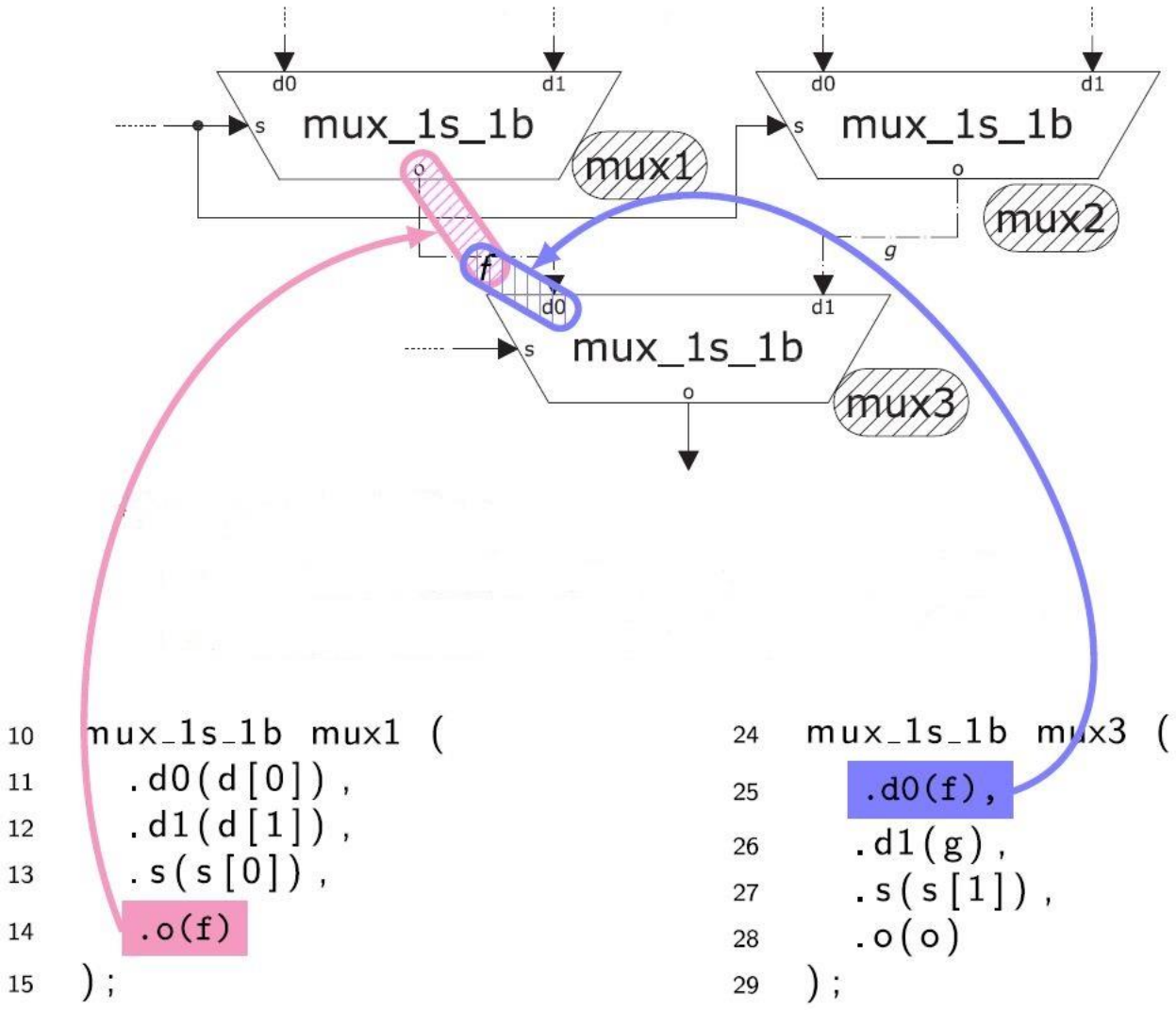

Fig. 4 – Conectarea intrărilor și ieșirilor la firele fizice declarate

Codul Verilog ilustrat mai sus:

- conectează ieșirea **o** a lui **mux1** la ***wire f*** (vezi partea stângă)
- conectează intrarea **d0** a lui **mux3** la ***wire f*** (vezi partea dreaptă)

3. Lucrare de laborator

Decodificatorul 3-la-8 linii cu validare pe intrare

Exercițiu:

Construiți un dispozitiv decodificator cu 3 linii de selecție și cu validare pe intrare folosind tehnica insanțierii dobândită în acest laborator.

Soluție:

Arhitectura unui astfel de decodificator 3-la-8 linii, denumit **dec_3x8** este prezentat mai jos:

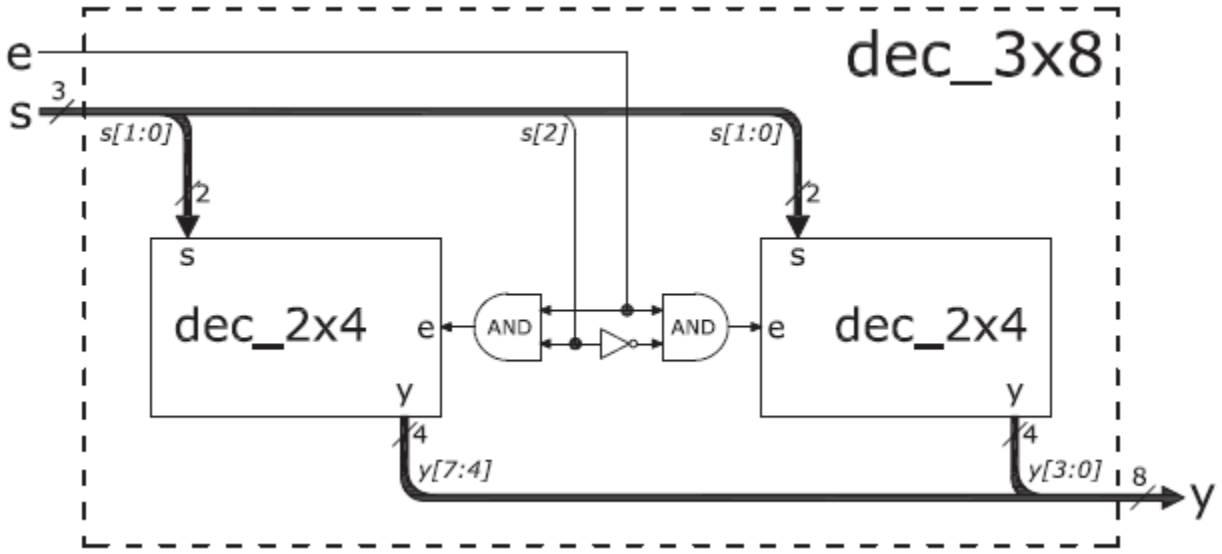

Fig. 7 – Design-ul decodificatorului cu 3 linii de selecție

Justificare: Mergând pe considerentul că decodificatorul 3-la-8 linii are 8 linii de ieșire urmează modelul componentelor decodificatorului de tip 2-la-4 linii, anume **dec_2x4** sunt necesare pentru a furniza numărul suficient de ieșiri.

Liniile de intrare **s[1:0]** sunt folosite ca linii de selecție pentru cele două componente **dec_2x4** în așa fel încât cele 4 linii de ieșire a componentei să fie adresate de configurații consecutive. Linia de selecție **s[2]** activează doar unul dintre cele două module **dec_2x4** asigurând în acest fel secvențierea corespunzătoare a celor două grupuri de 4 linii de ieșire generate de modulele **dec_2x4** la ieșirea acestora. Linia dedicată de validare activează selecția a unuia dintre cele două decodificatoare **dec_2x4.**

Arhitectura decodificatorului 3-la-8 linii face uz de:

- 2 dispozitive decodificator 2-la-4 linii, **dec_2x4**
- 2 porți AND și
- un inversor

Codul Verilog corespunzător arhitecturii propuse este descris mai jos:

```
module dec_3x8 (

input e ,

input [2:0] s ,

output [7:0] y

);

 dec_2x4 i1 (

    .s(s[1:0]) ,

    .e(e & s[2]) ,

    .y(y[7:4])

);

dec_2x4 i2 (

    .s(s[1:0]) ,

    .e(e & (~s[2])) ,

    .y(y[3:0])

);

endmodule
```