

# Construirea ierarhică a arhitecturilor în Verilog

Oprîtoiu Flavius  
flavius.opritoiu@cs.upt.ro

16 octombrie 2024

# Ierarhii în Verilog

Obiective:

- ▶ Deprinderea modului de instanțiere în Verilog
- ▶ Construirea ierarhiei unui design

Proiectarea ierarhică

- Facilitează proiectarea arhitecturilor complexe
- Promovează reutilizarea componentelor

*Instanță*: copie a unui modul utilizată într-un design mai larg.

O instanță are:

- Un *modul*: definește instanța respectivă.
- Un *container*: designul în care instanța este creată.

Crearea unei instanțe noi este numită *instanțiere*.

# Instanțiere

Pentru a crea o instanță sunt necesare următoarele elemente:

- numele *modulului* care se va instanția
- *numele instanței* (o diferențiază de alte instanțe ale aceluiași modul)
- *lista de conexiuni*

Lista de conexiuni specifică semnalele din container care sunt legate la porturile instanței.

O conexiune este specificată prin:

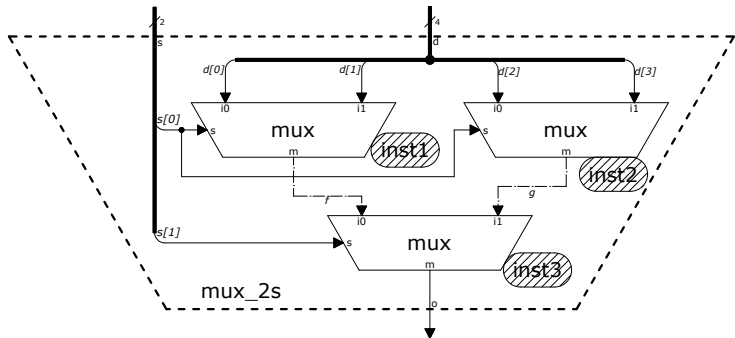
.<module\_port>( <container\_signal> ), în care

- *module\_port* este un port al instanței
- *container\_signal* este un semnal din container (poate fi un port al acestuia)

## Multiplexor 4-la-1

*Exercițiu:* Construiți un multiplexor 4-la-1 din multiplexoare 2-la-1.

*Soluție:* Multiplexorul 4-la-1, *mux\_2s*, utilizează multiplexoare 2-la-1, *mux*, cu intrările și ieșirile descrise în diagrama de mai jos:



## Multiplexor 4-la-1 (contin.)

Codul Verilog care implementează arhitectura precedentă:

```
1  module mux (
2      input s, i1, i0,
3      output m
4  );
5
6      always @ (*)
7          if (s) m = i1;
8          else m = i0;
9  endmodule
10
11 module mux_2s (
12     input [3:0] d,
13     input [1:0] s,
14     output o
15 );
16
17     wire f;
18     wire g;
19
20     mux inst1 (
21         .i0(d[0]),
22         .i1(d[1]),
23         .s(s[0]),
24         .m(f)
25     );
26     mux inst2 (
27         .i0(d[2]),
28         .i1(d[3]),
29         .s(s[0]),
30         .m(g)
31     );
32     mux inst3 (
33         .i0(f),
34         .i1(g),
35         .s(s[1]),
36         .m(o)
37     );
38 endmodule
```

## Multiplexor 4-la-1 (contin.)

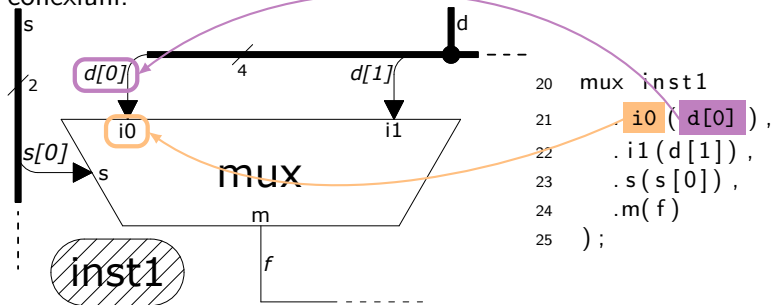
Instanța din liniile 20-25 ale slideului anterior are următoarele componente:

- *modulul* de instanțiat este `mux` (trebuie să fie numele unui modul existent)
- *numele instanței* este `inst1`
- *lista de conexiuni*, între paranteze rotunde (mai multe detalii în slide-ul următor)

```
20 mux inst1 (  
21     .i0(d[0]),  
22     .i1(d[1]),  
23     .s(s[0]),  
24     .m(f)  
25 );
```

## Multiplexor 4-la-1 (contin.)

Diagramă bloc care evidențiază instanța *inst1* și lista sa de conexiuni:



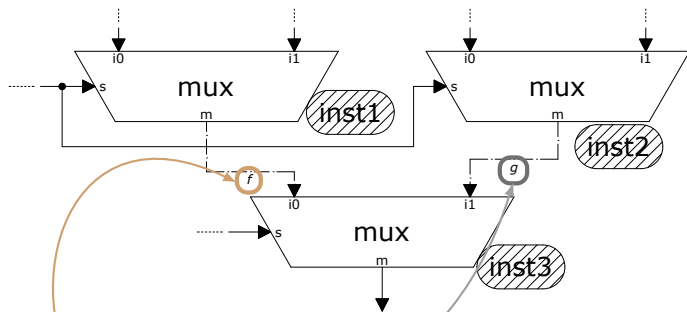
Elementele primei conexiuni:

- **i0** - port al modulului instanțiat
- **d[0]** - semnal din container (**d[0]** este port al containerului) la care portul **i0** este conectat

## Multiplexor 4-la-1 (contin.)

*Firele interne* leagă instanțele interne

- conectează ieșirea unei instanțe la intrarea altei instanțe
- declarate în interiorul containerului ca semnale de tip *wire*



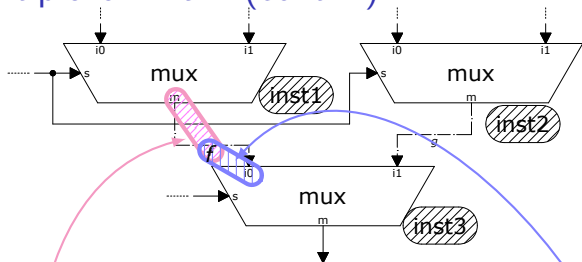
Declarația firelor interne din arhitectura mux\_2s:

```
17 wire f ;
```

```
18 wire g ;
```



## Multiplexor 4-la-1 (contin.)



Codul Verilog de mai jos:

- conectează ieșirea *m* a instanței lui *inst1* la firul *f* (stânga)
- conectează intrarea *i0* a instanței *inst3* la firul *f* (dreapta)

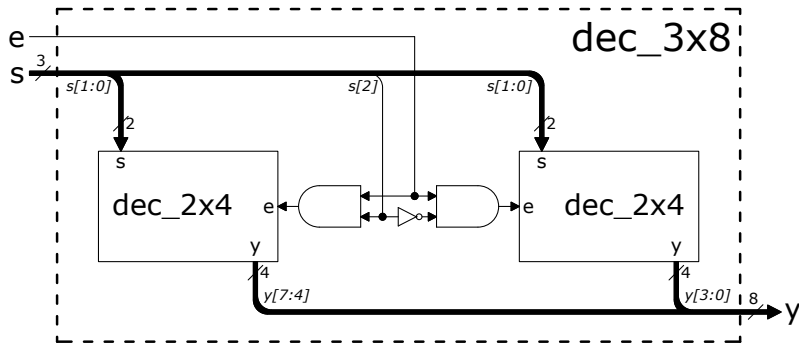
**Notă:** Lățimile portului și firului de conexiune trebuie să fie egale!

```
20 mux mux1 (  
21     .i0(d[0]),  
22     .i1(d[1]),  
23     .s(s[0]),  
24     .m(f)  
25 );  
32 mux inst3 (  
33     .i0(f),  
34     .i1(g),  
35     .s(s[1]),  
36     .m(o)  
37 );
```

## Decodor cu 3 linii de selecție și intrare de enable

*Exercițiu:* Construiți un decodificator cu 3 linii de selecție, intrare de enable și ieșiri active la 0, folosind modulul `dec_2x4` de [aici](#) (slide 12).

*Soluție:* Arhitectura este ilustrată mai jos:



## Decoder cu 3 linii de selecție și intrare de enable (contin.)

Codul Verilog care implementează arhitectura anterioară:

```
1  module dec_3x8 (
2      input e,
3      input [2:0] s,
4      output [7:0] y
5  );

6
7      dec_2x4 i1 (
8          .s(s[1:0]),
9          .e(e & s[2]),
10         .y(y[7:4])
11     );

12
13     dec_2x4 i2 (
14         .s(s[1:0]),
15         .e(e & (~s[2])),
16         .y(y[3:0])
17     );
18 endmodule
```