S5. Proiectarea modulelor reutilizabile

Construirea modulelor parametrizate în Verilog

1. Obiective

O1. Însușirea modului de separare a instanțelor *datapath* și *control path*

O2. Introducerea conceptului de *modul reutilizabil*

O3. Deprinderea modului de a redacta *module parametrizate*

2. Considerații teoretice

Datapath

Instanța **Datapath** propriu-zis se constituie din elemente care procesează datele: nu sunt luate decizii. Componentele tipice aici sunt:

- multiplexoare (MUX)
- registre (REG)
- unitățile aritmetico-logice (ALU)

În plus instanța Datapath construiește magistrale comune folosind porți logice cu trei stări.

Control path

Instanța **Control path** presupune luarea de decizii și este descrisă în termeni de mașini cu stări.

Notă: Componente cu stări de validare (precum registrele, numărătoarele) pot face parte de asemenea din datapath.

Module reutilizabile

Modulele reutilizabile sunt definite având **parametrii**, care pot fi redefiniți. În standardul Verilog 2001 parametrii modulului sunt specificați într-o secțiune dedicată, marcată de simbolurile #(s, i). Codul de mai jos descrie un registru paralel, având parametrizabile lățimea (nr. de biți) și vectorul de inițializare (conținutul registrului după reset).

```
module rgst #(
   parameter w = 8, //parametrul de lățime a registrului, valoarea implicită 8

   parameter iv  = {w{1'b0}} //inițializarea valorii parametrului
)(
input clk , //semnalul de tact
input rst_b , //reset asincron; activ pe low
input [w-1:0] d , //datele de intrare pe w bti
input ld , // load sincron; activ pe high
input clr , //clear sincron; activ pe high
output reg [w-1:0] q //continutul registrului, pe w biti
);
always  @  ( posedge clk, negedge rst_b )
begin
             if (! rst_b)
                q <= iv; //setarea continutului la valoarea de initializare
           else if (clr)
                q <= iv; //setarea continutului la valoarea de initializare
           else if ( ld )
            q <= d;
end
endmodule
```

3. Aplicații

Exercițiu: Construiți un register file 4×8.

Soluție: Un **register file** $M \times N$ este un element de stocare organizat ca un vector de M registre, fiecare registru având N biți. Permite simultan citirea unui registru intern și scrierea unui registru intern (posibil același).

Interfața unui register file include următoarele conexiuni:

- O intrare de date pe N biți, pentru scrierea regiștrilor interni (*wr_data*)
- O ieșire de date pe N biți, pentru citirea regiștrilor interni (*rd_data*)
- O adresă de intrare, pentru selectarea registrului care urmează a fi scris (*wr_addr*)
- O adresă de ieșire, pentru selectarea registrului care urmează a fi citit (*rd_addr*)
- Semnal de validare scriere (*wr_e*)
- Semnal de validare citire (*rd_e*)

Liniile de validare pentru portul de scriere/citire sunt opționale, M în general, ia forma lui 2^k: adresele de intrare/ieșire folosesc ***k*** biți.

Interfața unei file registru de forma $4 \times n$ este prezentată mai jos:

Fig. 1 – Structura unui register file

Pentru scenariul prezentat, interfața se constituie din:

- Portul de scriere (*wr_data*, *wr_addr*, *wr_e*)
- Portul de citire (*rd_data*, *rd_addr*, *rd_e*)
- Semnalul de ceas (*clk*)
- Semnalul de resetare (*rst_b*)

Un register file 4×8 fără linie de validare pe ieșire:

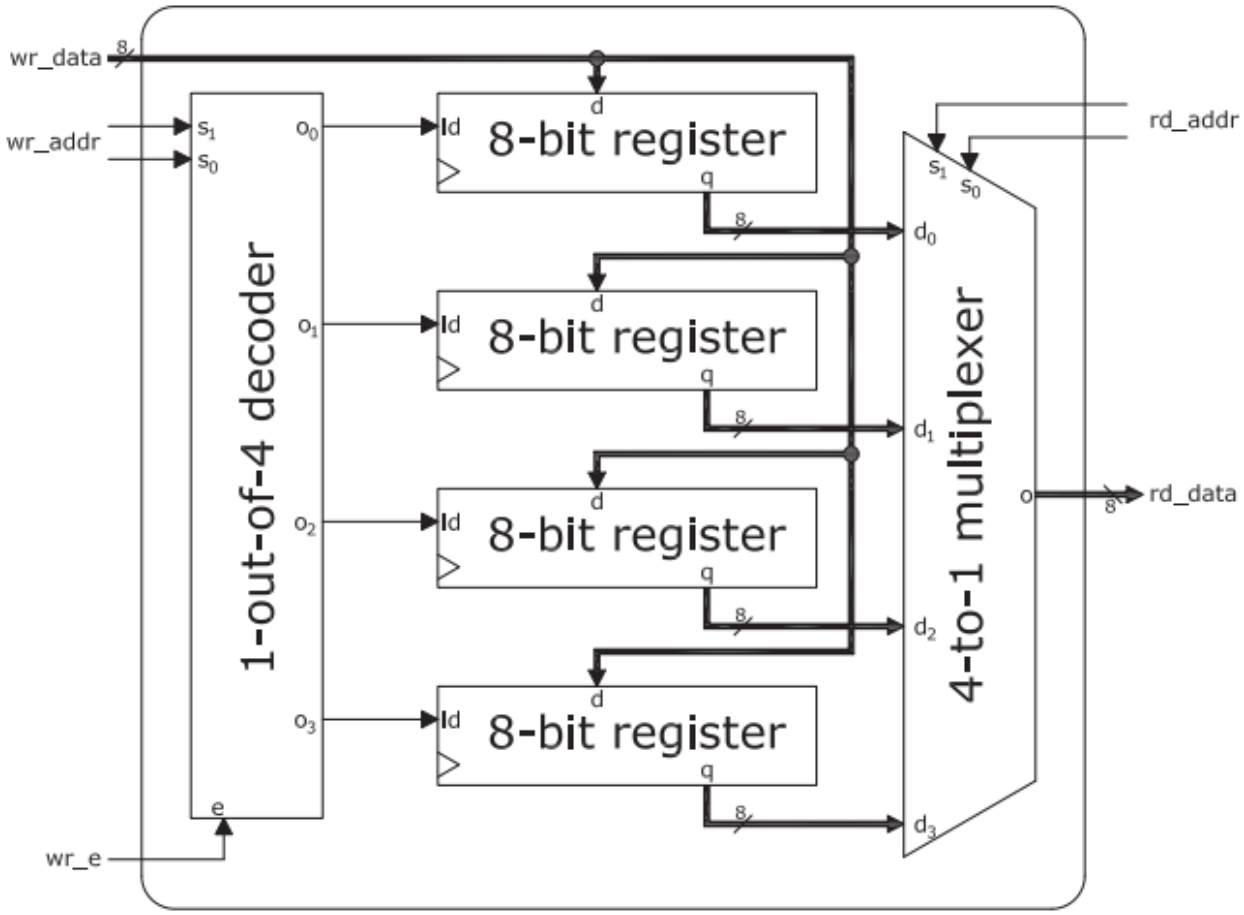
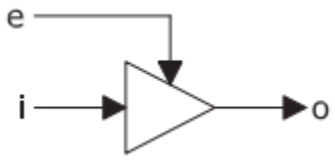


Fig. 2 – Arhitectura unui register file

Notă: Liniile de *clock* și *reset* au fost omise pentru concizie.

Poarta logică cu trei stări



Se folosește pentru conectarea mai multor componente pe o linie sau magistrală comună.

Ieșirea ***o*** este setată pe i atunci când linia de validare ***e***, este activă, și pe **impedanță ridicată** în caz contrar. O ieșire setată pe impedanță ridicată (simbolizat z, în Verilog) permite altei componente conectate la aceeași linie (sau magistrală) să seteze valoarea liniei.

Fragmentul de cod de mai jos exemplifică comandarea unui semnal spre impedanță ridicată, folosind o linie de control e:

```
wire [15:0] data, data_hiz ;

assign data_hiz = (e) ? data : 16'bz
```

Deoarece simbolul z, de impedanță ridicată, reprezintă cel mai semnificativ bit al constantei pe linia 2, se va extinde pe 16 biți z.

Construirea unui multiplexor folosind porți cu trei stări

Un multiplexor 4-la-1 pe n biți este implementat cu ajutorul porților logice cu trei stări:

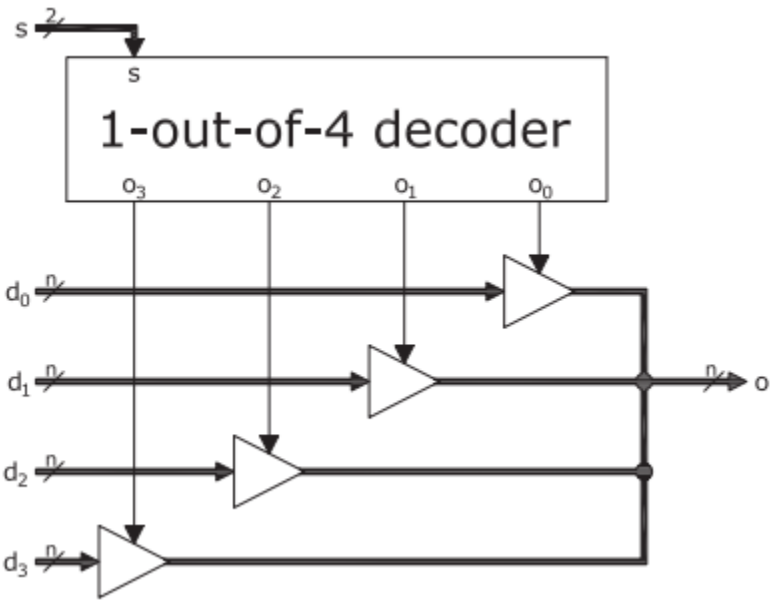


Fig. 3 – Arhitectura unui decodificator implementat cu porți trivalente

---

Observație:
În general redefinirea explicită a parametrilor unui modul în Verilog 2001 folosește următorul format:

```
module-name #( .parameter-name(value) , ... )
  instance_name ( .port-name(signal) ,  ... )
```

Codul de pe pagina următoare instanțiază un registru pe 16 biți, cu o valoare de inițializare setată pe 0.

```
rgst #(
    .w(16)
) registru1 (
    .clk(clk), ...
);
```

Alt cod mai jos instanțiază un registru pe 4 biți, cu o valoare de inițializare setată pe 15.

```
rgst #(
    .w(4)
    .iv(4'd15)
) registru2 (
    .clk(clk), ...
);
```