Crearea modulelor parametrizate

Opriţoiu Flavius
flavius.opritoiu@cs.upt.ro

25 octombrie 2024

Introducere

Obiective:

- Separarea căii de date și a căii de control
- Înțelegerea utilizării modulelor parametrizate

Calea de date

- Conține elemente de prelucrare a datelor: nu sunt luate decizii
- Componente tipice: multiplexoare, registre, Unități Aritmetice și Logice (ALUs), numărătoare
- Utilizarea magistralelor contruite prin drivere tri-state

Calea de control

- preocupată de luarea deciziilor
- Descrisă în termenii unei mașini cu stări finite

Notă: Componente cu stare (registre, contoare) pot face parte din calea de date.

Module reutilizabile

Modulele reutilizabile sunt definite având *parametri*, care pot fi redefiniți. În standardul Verilog 2001 parametrii modulului sunt specificați într-o secțiune dedicată, marcată de simbolurile #(și).

Codul de mai jos descrie un registru cu încărcare paralelă, având parametrizabile lățimea (nr. de biți) și vector de inițializare (conținutul registrului după reset).

```
1   module rgst #(
2           parameter w = 8,          //register's width parameter; default of 8
3           parameter iv = {w{1'b0}}//initialization value parameter
4   )(
5           input clk,
6           input rst_b,              //asynchronous reset; active low
7           input [w-1:0] d,          //input data, on w bits
8           input ld,                 //synchronous load; active high
9           input clr,                //synchronous clear; active high
10          output reg [w-1:0] q      //register's content, on w bits
11  );

13          always @ (posedge clk, negedge rst_b)
14                  if (!rst_b)
15                          q <= iv;//set content to initialization value
16                  else if (clr)
17                          q <= iv;//set content to initialization value
18                  else if (ld)
19                          q <= d;
20  endmodule
```

Module reutilizabile (contin.)

Redefinirea explicită a parametrului unui modul se face în standardul Verilog 2001 astfel:

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

Codul de mai jos instanțiază un registru pe 16 biți cu vector de inițializare 0

```
1 rgst #(
2     .w(16)
3 ) registru1 (
4     .clk(clk), ...
5 );
```

Codul următor instanțiază un registru pe 4 biți, inițializat la 15

```
1 rgst #(
2     .w(4),
3     .iv(4'd15)
4 ) registru2 (
5     .clk(clk), ...
6 );
```

Module reutilizabile (contin.)

Exercițiu rezolvat

Exercițiu: Să se construiască un register file 4x8.

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

Interfaa unui register fileinclude:

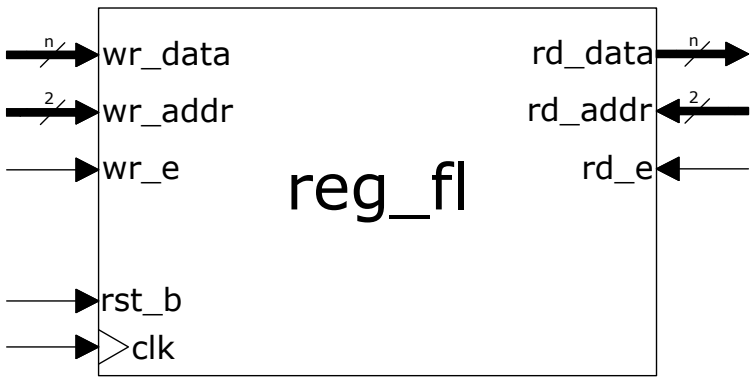
- intrare de date pe *N* biți, scriere registre interne (*wr_data*)
- ieșire de date pe *N* biți, citire registre interne (*rd_data*)
- adresă de scriere, selecție registru de scris (*wr_addr*)
- adresă de citire, selecție registru de citit (*rd_addr*)
- semnal activare scriere (*wr_e*)
- semnal activare citire (*rd_e*)

Intrările de activare sunt opționale. **M** este, tipic, de forma 2^k, caz în care adresele de citire/scriere sunt pe *k* biți.

Module reutilizabile (contin.)

Exercițiu rezolvat (contin.)

Interfața unui register file *4xn* este ilustrată mai jos:



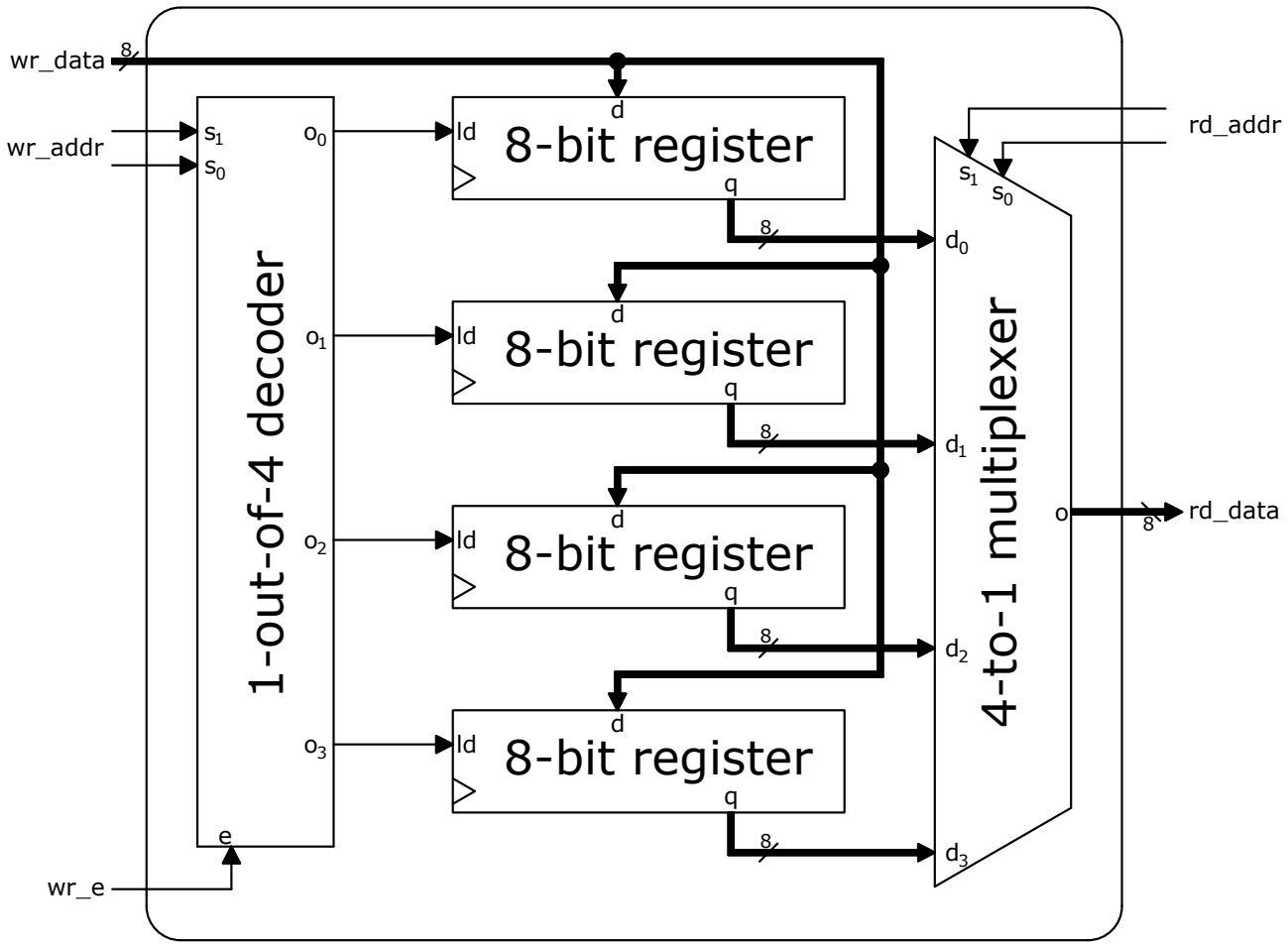
În acest caz, interfața conține:

- portul de scriere format din (*wr_data*, *wr_addr* și *wr_e*)
- portul de citire format din (*rd_data*, *rd_addr* și *rd_e*)
- semnalul de ceas (*clk*)
- semnalul de reset (*rst_b*)

Module reutilizabile (contin.)

Exercițiu rezolvat (contin.)

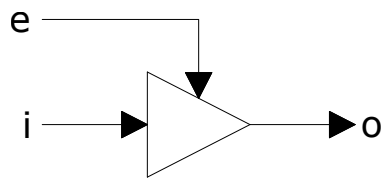
Arhitectura unui register file *4x8* fără linie de activare a citirii:



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

Driver tri-state

Sunt utilizate pentru conectarea ieșirilor mai multor componente la o linie sau magistrală comună.



Ieșirea *o* ia valoarea lui *i* când linia de activare, *e*, este 1, și este în *impedanță ridicată* altfel. O ieșire în impedanță ridicată (simbolizat în Verilog prin z) permite altei componente conectate la aceeași linie (sau magistrală) să seteze valoarea liniei.

Fragmentul de cod următor exemplifică comanda unui semnal în impedanță ridicată prin linia de control *e*:

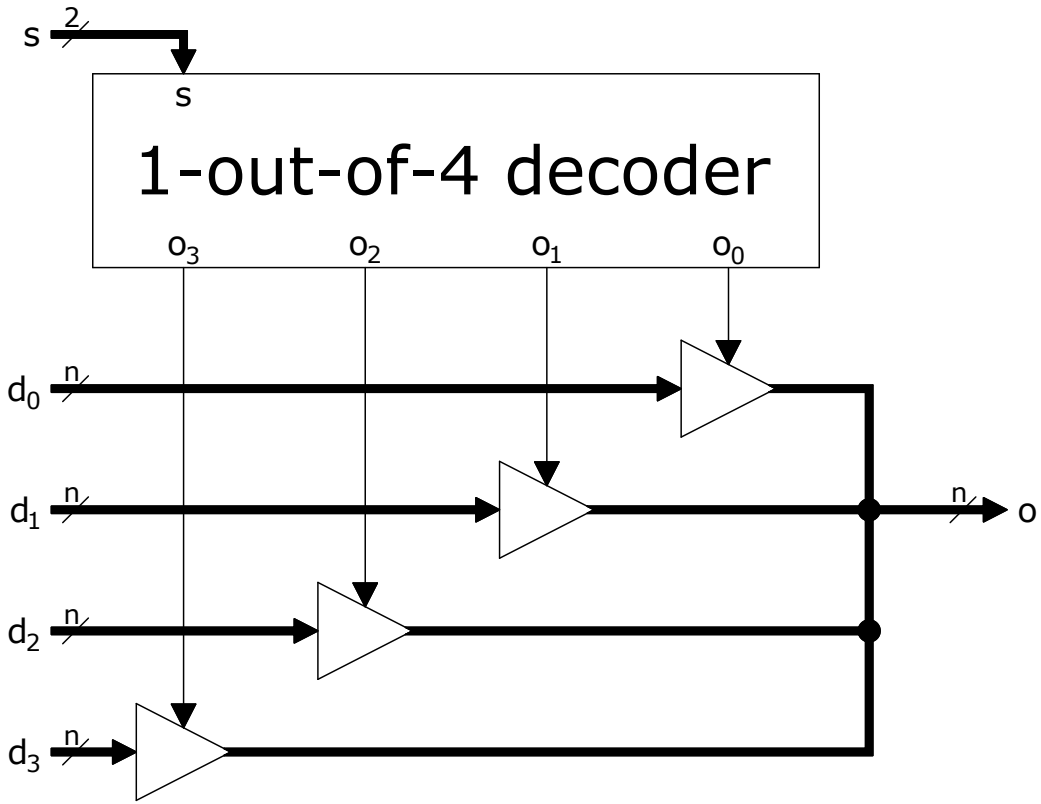
```
1    wire [15:0] data, data_hiz;
2    assign data_hiz = (e) ? data : 16'bz;
```

Pentru că simbolul z, de impedanță ridicată, este cel mai semnificativ bit al constantei din linia 2 este extins la 16 biți z.

Driver tri-state

Construirea unui multiplexor folosind drivere tri-state

Un multiplexor 4-la-1 pe n biți implementat cu drivere tri-state:



Intrarea de selecție, s, comandă decodificatorul 2-la-4. Etajul final conectează ieșirile tuturor driverelor tri-state împreună.