Arhitectura calculatoarelor — 07.10.2024

(lab 2 - S2)

! De citit https://wiki.eecs.yorku.ca/course_archive/2013-14/F/3201/_media/verilog-tutorial_harvard.pdf

Verilog Hardware Description Language (HDL):

- Descriere textuală a unei realizări hardware
- Permite proiectarea la diverse nivele de abstractizare: algoritmi versus tranzistori
- Facilitează verificarea înaintea integrării fizice
- Uneltele de sinteză translatează modelele Verilog în realizări hardware

Module Verilog

Descrierea Verilog a unei arhitecturi este organizată în unul sau mai multe *module*.

Un modul Verilog este definit astfel:

- numle modulului, precedat de cuvântul rezervat `module`
- o listă a intrărilor și ieșirilor modulului, între paranteze
- implementarea modulului
- cuvântul rezervat, final, `endmodule`

Intrările și ieșirile unui modul sunt colectiv referite ca *porturi*.

Pe parcursul acestui laborator, implementarea unui modul poate cuprinde următoarele tipuri de declarații:

- *instanțe* de module Verilog
- atribuiri continue, introduse prin cuvântul rezervat `assign`
- blocuri `always`

Atribuiri continue → assign <signal> = <expression>;

- *signal* este fie un semnal fie concatenarea mai multor semnale
- *expression* se referă la o expresie Verilog validă

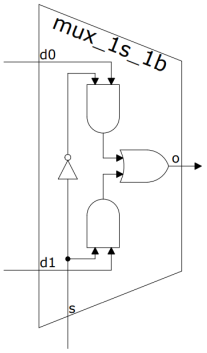
Instrucțiunea actualizează partea stângă a atribuirii ori de cîte ori un semnal din partea dreaptă se modifică.

Exercițiu: Să se implementeze un multiplexor 2-la-1 pe 1 bit utilizând Verilog.

Soluție: Implementarea Verilog și simbolul grafic a multiplexorului snt descrise mai jos

```
1  module  mux_1s_1b  (
2     input d0 ,
3     input d1 ,
4     input s ,
5     output o
6  );

8     assign o = ((~s) & d0) | (s & d1);
9  endmodule
```



→ dacă [s = 0 => iese d0
       [s = 1 => iese d1

Magistrale

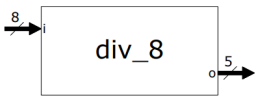
În Verilog o *magistrală*, sau un *vector* este un semnal constând dintr-o colecție de fire. Este definit specificând rangul superior și inferior, între paranteze drepte, urmate de numele magistralei.

Exercițiu: Construiți un dispozitiv pentru determinarea câtului împărțirii la 8 a unui număr întreg, fără semn, pe 8 biți.

Soluție:

```
1  module div_8 (
2    input [7:0] i,
3    output [4:0] o
4  );

6    assign o = i[7:3];
7  endmodule
```



Notă: Câtul împărțirii la $8 = 2^3$ a unui întreg fără semn se obține eliminând cei mai puțin semnificativi 3 biți.

ex: 9:8 → cât = 1

1001

0001 0001 → 17:8 → 2

→ 10 → câtul = 2

Operatorul part-select

Operatorul Verilog *part-select* permite selectarea unui set contiguu de fire dintr-o magistrală. Operatorul specifică marginile superioară și inferioară dintre biții magistralei, între paranteze drepte, și va returna toate liniile aflate între margini, inclusiv.

Exercițiu: Construiți un modul cu o linie de selecție *s* și o intrare *d*, pe 64 biți. Cand *s* este activă, ieșirea ia valoarea celor mai semnificativi 32 de biți ai intrării *d*, altfel, ieșirea va fi egală cu biții aflați între rangurile 47 și 16, inclusiv, ai intrării *d*.

Soluție:

```
1  module bus_select (
2    input [63:0] d,
3    input s,
4    output [31:0] o
5  );

7    assign o = s ? d[63:32] : d[47:16];
8  endmodule
```

Operatorul de concatenare

Operatorul Verilog de *concatenare* construiește magistrale. Concatenarea reprezintă o listă de semnale, separate prin virgulă, marginite de acolade. Semnalul cel mai din stanga va ocupa pozițiile binare cele mai semnificative în noua magistrală iar semnalul din dreapta pe cele mai puțin semnificative.

Exercițiu: Să se construiască un modul pentru inversarea ordinii biților unei valori pe 4 biți, primită la intrare.

Soluție:

```
1  module reverse_4b (
2    input [3:0] i,
3    output [3:0] o
4  );

6    assign o = {i[0], i[1], i[2], i[3]};
7  endmodule
```

Operatori aritmetici

Verilog pune la dispoziție următorii operatori aritmetici binari: +, −, *, /, %, pentru operatorul modulo și ** pentru exponențiere. Operatorii aritmetici unari, + și −, sunt folosiți pentru controlul semnului operanzilor.

Exercițiu: Construiți un sumator pe 8 biți, fără intrare de transport.

Soluție:

```
1  module add_8b (
2    input [7:0] x,
3    input [7:0] y,
4    output [7:0] z,
5    output co
6  );

8    assign {co, z} = x + y;
9  endmodule
```

co = carry-out
z = rezultat

Un sumator modulo-2^8 are aceeași implementare eliminând, însă, ieșirea *co*.

Operatorul condițional

Operatorul condițional are următorul format în Verilog:

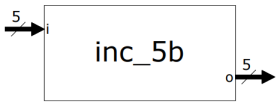
```
expression ? expression_true : expression_false
```

Exercițiu: Implementați un dispozitiv pentru incrementarea numerelor întregi, fără semn, pe 5 biți. Dacă numărul de la intrare are valoarea 31, dispozvitivul îl va furniza la ieșire nemodificat

Soluție:

```
1  module increment_5b (
2    input [4:0] i,
3    output [4:0] o
4  );

6    assign o = (i == 31) ? i : i + 1;
7  endmodule
```



Notă: Operatorul condițional oferă o modalitate de implementare a instrucțiunii condiționale (*if* <condiție> *then* ...), la fel ca și multiplexorul.

Operatorul de replicare

Operatorul de *replicare* replică o expresie de un număr de ori, concatenând copiile într-o magistrală. Formatul operatorului este {r{expr}}, replicând expresia *expr* de *r* ori.

Exercițiu: Considerând un întreg pe 8 biți, reprezentat în cod C2, construiți un modul pentru extinderea sa la 32 de biți, fără modificarea valorii.

Solution:
În codul C2, extinderea unei valori pe un număr mai mare de biți se face replicând bitul de semn, aflat pe cea mai semnificativă poziție.

```
1  module sign_extend (
2    input [7:0] i,
3    output [31:0] o
4  );

6    assign o = {{24{i[7]}}, i};
7  endmodule
```

Operatori binari și de reducere

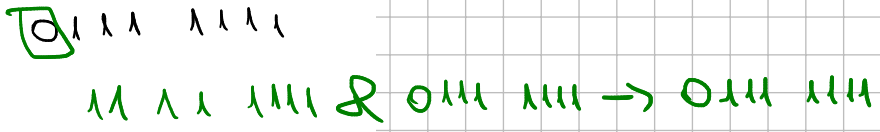
Operatorii binari operează asupra magistralelor (vectorilor). Tabelul următor prezintă simbolul, funcția și aritatea acestora.

Simbol	Funcție	Aritate
~	complementare	unar
&	ȘI	binar
|	SAU	binar
^	EXOR	binar
^~	XNOR	binar

Operatorul de *reducere* are un singur operand, de tip magistrală. Va genera o ieșire pe 1bit, obținută prin aplicarea respectivului operatoru asupra tuturor biților magistralei. Operatorii de reducere sunt &, |, ~& pentru reducere NAND, ~| pentru reducere NOR, ^ și ~^ sau ^~ pentru reducere XNOR.

Exercițiu: Construiți un modul pentru implementarea funcției $o(x) = \max(0, x)$ pentru numere cu semn, pe 8 biți.

Soluție:



```
module max (
  input [7:0] x,
  output [7:0] o
);
  assign o = {8{~x[7]}} & x;
endmodule
```

Exercițiu: Generați bitul de paritate pară pentru un semnal pe 7 biți (paritatea pară este suma modulo-2 a tuturor biților).

Soluție:

```
module parity (
  input [6:0] i,
  output p
);
  assign o = ^i;
endmodule
```

Alți operatori verilog

Operatori relaționali: <, >, <=, >=, == (egalitate), != (inegalitate), === (egalitate case) and !== (inegalitate case). Egalitatea și inegalitatea case tratează semnale Verilog cu 4 valori. În Verilog, pe lângă valorile 0 și 1, un semnal poate fi nedefinit, marcat prin simbolul x sau în impedanță ridicată, marcat prin simbolul z. Pentru semnalele $A = 1x01$ și $B = 1x01$, expresia $A === B$ este evaluată ca adevărată, iar $A == B$ ca falsă.

Operatori logici: &&, || și ! (negare logică).

Operatorii de deplasare: << (deplasare la stânga), <<< (deplasare aritmetică la stânga a valorilor cu semn), >> (deplasare la dreapta) and >>> (deplasare aritmetică la dreapta a valorilor cu semn). Operatorii relaționali și logici returnează o valoare pe 1 bit: fie 1, pentru rezultat adevărat, fie 0, pentru rezultat fals.

Constante în Verilog

Formatul constantelor în Verilog:
`<bit_width>'<radix_specifier><value>`
unde

- `bit_width`: întreg zecimal, pozitiv indicând numărul de biți alocați reprezentării constantei; opțional
- `radix_specifier`: poate fi `b`, pentru baza 2, `o` pentru octal, `d` pentru zecimal (baza implicită) și `h` pentru hexazecimal; opțional
- `value` valoarea constantei exprimată în baza specificatăthe constant's value expressed in the specified radix

Tabelul de mai jos prezintă câteva exemple de constante în Verilog:

Constantă Verilog	Stocată ca
3'b110	110
8'b0010_1101	00101101
5'd6	00110
10'h9e	0010011110