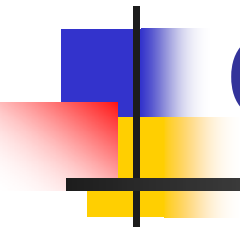
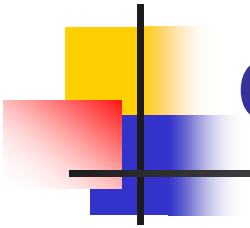


Circuite integrate digitale

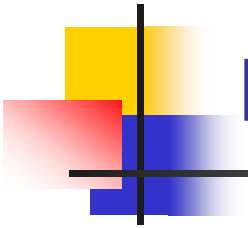
Curs 7



Cuprins

- Funcţii logice: definire prin diagrame
- corespondenţe (expresii, tabel, diagramă)
- minimizarea funcţiilor logice cu diagrame
- implementare cu MUX, DMUX, DCD

- breviar operatori Verilog



Funcţii logice

- definire: funcţii care au variabile logice şi iau valori logice
- exprimare
  - expresii logice
  - tabel de adevăr
  - diagrame VK (Veitch, Karnaugh)
- implementare

Implementarea unei funcţii logice

- **funcţie exprimată în limbaj natural**

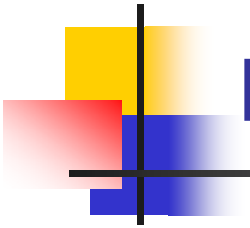
... cu ajutorul tabelului sau al raţionamentelor logice deducem

- **expresia logică a funcţiei**

... cu ajutorul algebrei logice sau al diagramelor urmărim să ajungem la

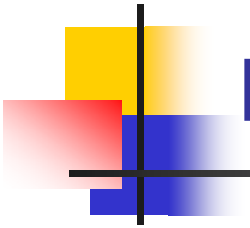
- **expresia minimală a funcţiei** (cea mai simplă)

- **circuitul logic cu porţi** "transcrie" această expresie



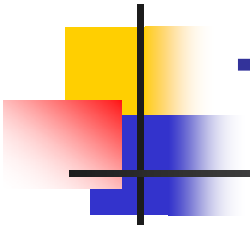
Minimizarea funcţiilor logice

- aducerea la forma cea mai simplă
  - cu algebră logică
  - cu un algoritm care foloseşte diagrame logice
- circuitele cu porţi logice traduc expresia!



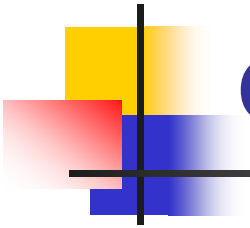
Definiţii

- configuraţie binară pe *n* biţi
- mintermen (minterm)
- termeni logici adiacenţi
- FND



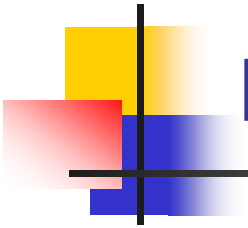
Tabele de adevăr

- definire exhaustivă
- pentru n variabile: 2^n linii
  - există exp(2^n) funcţii de n variabile
- fiecare linie corespunde unui mintermen



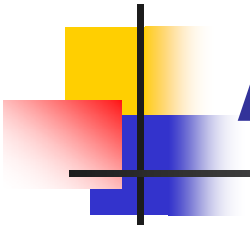
Corespondenţe

- tabel – expresie (FND)
- expresie – tabel



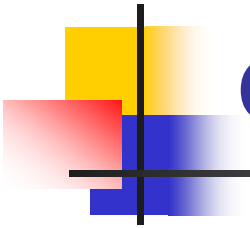
Diagrame logice (diagrame VK)

- simplificarea este generată de adiacenţe
- diagramele reprezintă o redesenare a tabelelor de adevăr pentru a evidenţia adiacenţele
- ex 2, 3, 4 variabile
- corespondenţe tabel – diagramă, diagramă – tabel



Algoritmul de minimizare

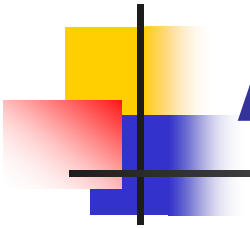
- varianta clasică
- funcţii incomplet definite
- funcţii cu variabile incluse



Corespondenţe

- diagramă – expresie
- expresie – diagramă

- obs. atunci când trebuie să trecem de la o expresie la un tabel de adevăr, este mai avantajos să completăm mai întâi diagrama



Aplicaţii

- sumatorul complet pe 1 bit – bitul de carry
- diferite diagrame 3 variabile
- multiplicator pentru numere pe 2 biţi
- diferite diagrame 4 variabile

Exemplu: sumator complet pe 1 bit

sum = a′b′c + a′bc′ + ab′c′ + abc=

= a′(b′c + bc′) + a(b′c′ + bc)=

= a′(b ⊕ c) + a(b ⊕ c)′=

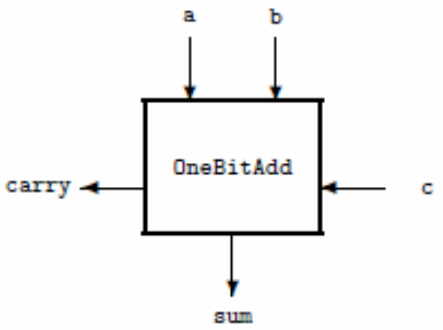
= a ⊕ (b ⊕ c)=

= a ⊕ b ⊕ c

carry = a′bc + ab′c + abc′ + abc=

= (a′b+ab′)c+ab(c′+c)=

= (a ⊕ b)c + ab



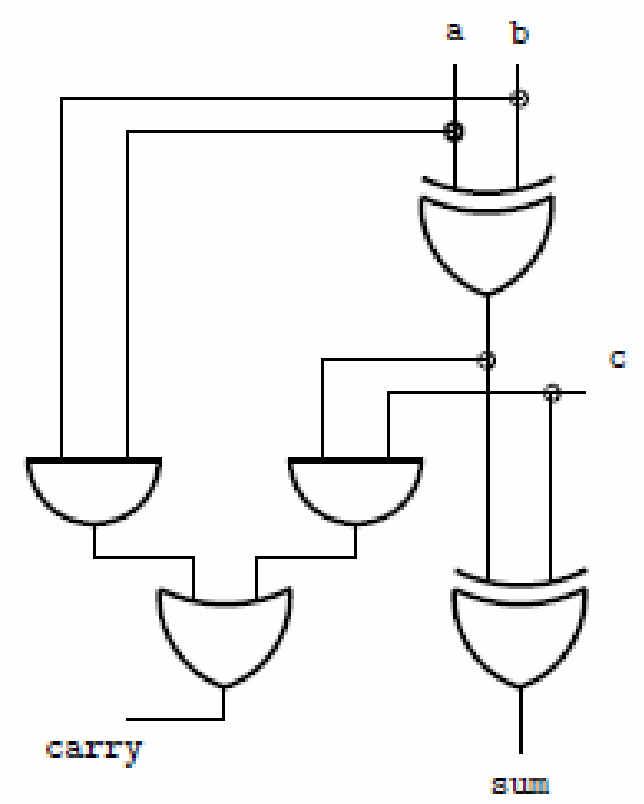
a	b	c	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

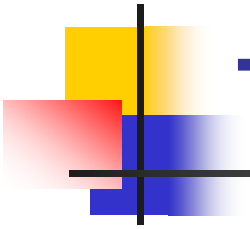
Sumator complet pe 1 bit: schemă logică

Expresii logice:

sum = a ⊕ b ⊕ c = (a ⊕ b) ⊕ c

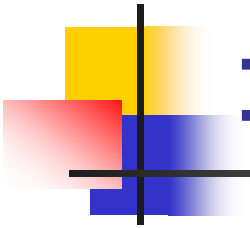
carry = a b + c (a ⊕ b)





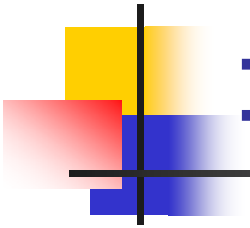
Tema 8

- Proiectaţi un transcodor BCD – 7 segmente cu porţi logice, folosind algoritmul de minimizare pentru funcţii incomplet definite. Toate funcţiile vor fi implementate pe aceeaşi schemă.
- *indicaţie: urmăriţi o minimizare pe ansamblu a funcţiilor (evidenţiaţi termenii care se pot folosi în comun).*



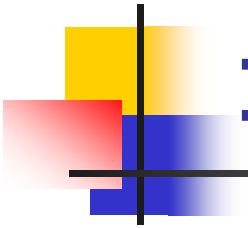
Implementarea CLD

- cu MUX
- cu DCD sau DMUX



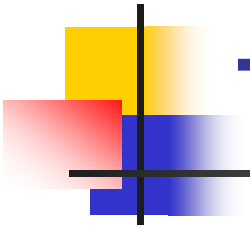
Implementarea cu MUX

- MUX : funcţia de selecţie
- porneşte de la tabelul de adevăr
- generalizare: implementarea cu ROM sau RAM
- varianta optimă: tabel cu n-2 variabile



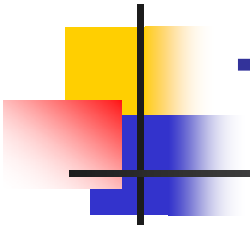
Implementarea cu DCD sau DMUX

- DCD calculează mintermenii
- porneşte de la FND
- mai multe funcţii folosesc acelaşi DCD
- cu DMUX: enable activ



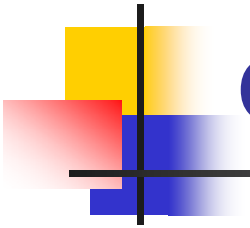
Tema 9

- Reluaţi tema 8, implementând funcţiile logice cu DCD.
- *indicaţie: se va folosi un singur decodor pentru toate funcţiile*



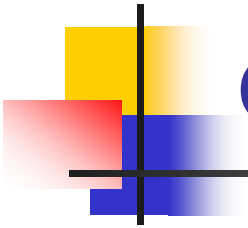
Tipuri de operatori în Verilog

1. aritmetici
2. bit cu bit (bitwise)
3. reducere
4. logici
5. relaţionali
6. deplasare
7. condiţionali



Operatori aritmetici

- adunare +
- scădere –
- înmulţire *
- împărţire /
- modulo %



C2 – complement faţă de 2

- variabilele de tip reg, numere negative, sunt reprezentate automat în C2, dar sunt interpretate ca întregi fără semn

- ex: A=5, B=2 (pe 4 biti, rezultatul pe 5 biti)

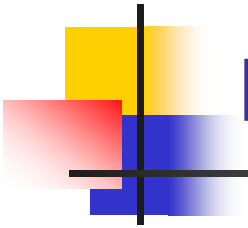
  A-B=3

  B-A=29

  - A=27

Operatori bit cu bit

Simbol	Operator
~	negare
&	si
|	sau
^	xor
~^, ^~	nxor



Exemple

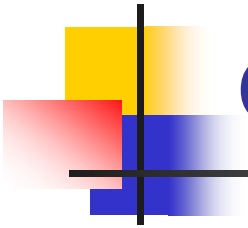
y1 = 8'b1011_1001
y2 = 8'b1101_0100

~y1 = 0100_0110

y1 & y2 = 1001_0000

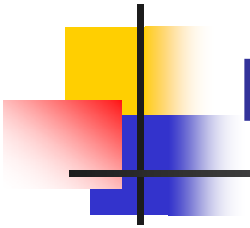
y1 | y2 = 1111_1101

y1 ^ y2 = 0110_1101



Operatori de reducere

Simbol	Operator
&	reducere, si
~&	reducere, nand
|	reducere, or
~|	reducere, nor
^	reducere, xor
~^, ^~	reducere, nxor



Exemple

- rezultatul este pe un singur bit!!!

- x = 4'b 1101

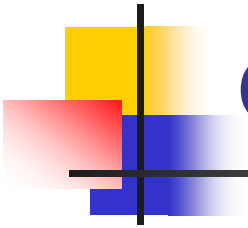
  & x = 0

  ~& x = 1

  | x = 1

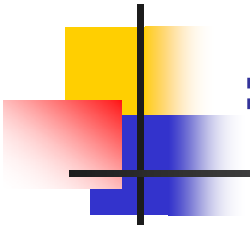
  ^ x = 1

  ~^ x = 0



Operatori logici

simbol	operator
!	negare logică
&&	şi logic
||	sau logic
==	egalitate logică
!=	inegalitate logică
===	egalitate bit cu bit
!==	inegalitate bit cu bit

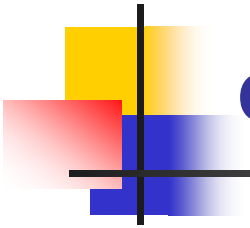


== si ===

- == egalitate logică strictă, x pt ambiguităţi
- === egalitate în logică cu 4 stări (egalitate după caz)

- ex: 110xz 110xz (===)

  110xz == 110xz are rezultatul x



operatori logici

- rezultatul este o valoare logică
- Atenţie!!! operatorii logici pot fi folosiţi inadecvat deoarece tipurile de variabile nu sunt foarte restrictive

- A && B, pentru variabile scalare, este identic cu A&B
- pentru vectori: adevărat (1 logic) ⇔ întreg pozitiv
- ex A = 3'b110, B = 3'b11x

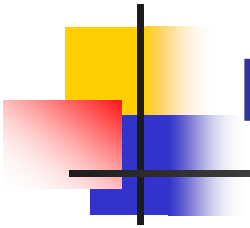
  A&&B = 0 (fals)

  A & B = 110 (logic adevarat)

  (intreg fără semn)

Operatori relaţionali

simbol	operator
<	mai mic
<=	mai mic sau egal
>	mai mare
>=	mai mare sau egal

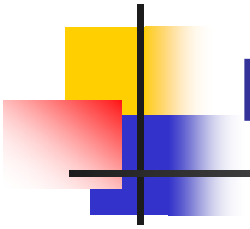


Precizări

- rezultatul este o valoare logică (adevărat sau fals)
- pentru net-uri sau reg-uri, se compară ca numere fără semn
- orice bit z sau x => rezultat x

Operatori deplasare

simbol	operator
<<	deplasare stanga
>>	deplasare dreapta
<<<	se păstrează bitul de semn
>>>	id.



Exemple

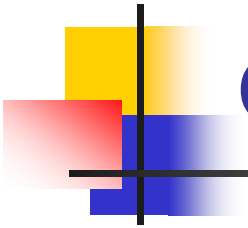
A = 1101_1001

A << 1 1011_0010

A << 3 1100_1000

A >> 2 0011_0110

A >>> 2 1001_0110



Operator condiţional

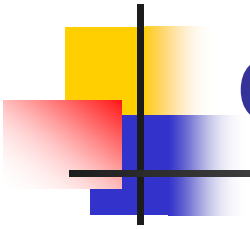
y = (a==b) ? c:d

ex. driver bus date

wire [15:0] bus_a = enable_a ? data : 16'bz

obs. definire compactata **wire** si **assign**

enable_a = x => bus_a = 16'bx



Operator concatenare

- utili pentru definirea bus-urilor
- reunesc (concateneaza) mai multi biti intr-un bus
- includ duplicarea (replicarea multipla)

{a, b, c} //.... Concatenate a, b and c into a bus
{3{a}} // ..... Replicate a, 3 times
{{5{a}}, b} //.. Replicate a, 5 times and concatenate to b

Precedenţa (prioritatea) operatorilor

simbol operator	funcţie	precendenţă
+ - ! ~(unar)	semn, invers	maximă
* / %	aritmetici	
+ - (binari)	aritmetici	
<< >>	deplasari	
< <= > >= == != === !==	relationali	
& ~& ^ ~^ | ~|	reducere	
&& ||	logici	
? :	conditional	minimă