Logikai függvények, kombinációs hálózatok

F1. Igazolja az alábbi azonosságokat a Boole algebra axiómái és tételei alapján!

a) A + /A*B = A+B

A + /A*B = A*1+/A*B = A*(B+/B) + /A*B =

A*B +A*/B + /A*B = A*B + A*B + A*/B + /A*B =

A*B + A*/B + A*B + /A*B = A*(B+/B) + (A+/A)*B

= A*(1) + (1)*B = A + B

Egyszerűbben: A + /A*B = A*(1) +/A*B = A*(1+B) +/A*B = A +A*B+/A*B = A+B*(A+/A)=A+B

b) A*B + /A*C + B*C = A*B + /A*C

A*B + /A*C + B*C = A*B + /A*C + B*C*(1) =
A*B + /A*C + B*C*(A + /A) = A*B + /A*C + B*C*A + BC*/A =
A*B + A*B*C + /A*C + /A*C*B = A*B*(1 + C) + /A*C*(1 + B) =
A*B*(1) + /A*C*(1) =
A*B + /A*C

c) A*B + /A*C = (A+C)*(/A+B)

(A+C)*(/A+B) = A*/A + /A*C + A*B + B*C = /A*C + A*B +B*C, ez éppen az előző feladat

F2. Hozza egyszerűbb alakra az alábbi kifejezéseket, felhasználva a Boole algebra közismert tulajdonságait:

a) Y1 = /A + A*B*/C + (/A + A*B*/C)*(A + /A*/B*C);

Az (/A + A*B*/C) tag kiemelhető. Így Y1 = (/A + A*B*/C)*(1+ (A + /A*/B*C)) = /A + A*B*/C, ami az F1.a alapján = /A + B*/C.

b) Y2 = /((A*B + C*D)*(A*/C + B*/D));

Y2 = /(A*B*/C + A*B*/D) = /((A*B)*(/C+/D)) = /((A*B)*/(C*D)) = /(A*B) + C*D = /A + /B + CD

c) Y3 = B*(A*C + /A*/C) + A*/C + /A*C;

Y3 = B*(A XNOR C) + A XOR C = B*/(A XOR C) + (A XOR C) = (A XOR C) + /(A XOR C)*B, ebben a sorrendben már teljesen azonos az F1.a feladattal, tehát = (A XOR C) + B.

d) Y4 = A*C*D + A*B*C + D*(/A + /B) + /A*C*D;

Y4 = CD*(A+/A) + A*B*C + D*/A + D*/B = C*D + A*B*C + D*/A + D*/B
= CD*(1) + A*B*C + D*/A + D*/B , ahol az 1 helyére az A*B + /(A*B) kifejezést behelyettesítve
= CD (A*B + /(A*B)) + A*B*C + D*/A + D*/B, De Morgan tétel szerint átírva
= CD(A*B + /A + /B) + A*B*C + D*/A + D*/B és elvégezve a beszorzást
= A*B*C*D + /A*C*D + /B*C*D + A*B*C + D*/A + D*/B. Eben az első 3 tag kiesik, mert
= A*B*C*(1 + D) + /A*D*(C + 1) + /B*D*(C + 1) = A*B*C + /A*D + /B*D a megoldás

F3. Tervezze meg az 1 bites teljes összeadó modul logikai áramköreit és építsen láncba kapcsolható, (kaszkádosítható) egységet belőle. Vázolja fel egy 4 bites egység felépítését, elemezze a kaszkádosítás szabályait!

A feladat a digitális technikában (és már területeken is) alkalmazott módszer, az alulról felfelé építkezés példáját mutatja be. A bináris összeadás tetszőleges operandusméretekre megvalósítható az elemi, egybites összeadó modulok egymás után kapcsolásával, egy lineáris szerkezet kialakításával, ahol az esetlegesen keletkező átvitelek mindig a következő helyiérték bitjeihez adódnak.

Az előadáson 2 függvényt elemeztünk. Ezek szöveges specifikációi a következők voltak:

- F1: A függvény a bemeneti változók paritását jelzi. Ha a bemeneten páratlan számú aktív jel van, a kimenet jel értéke 1, egyébként 0.
- **F1 alternatív: A függvény egy bináris 1 bites teljes összeadó (A, B, C) összeg (S) kimenete. Az összeadás művelet szabályai szerint, az összeg kimenet értéke 0, 1, 2 és 3 aktív bemenet esetén: 0+0+0=0, 0+0+1=1, 0+1+1=0, és van átvitel, végül 1+1+1=1 és van átvitel.**
- F2: A függvény egy többségi szavazást jelző áramkör. Ha bemenetei között több az aktív jel, mint az inaktív, akkor a kimenet 1, különben 0.
- **F2 alternatív: A függvény egy bináris 1 bites teljes összeadó (A, B, C) átvitel (Co) kimenete. Az összeadás művelet szerint akkor van átvitel, ha az (A, B, C) bemenetek közül legalább 2 bemenet aktív, azaz a 0+1+1 vagy 1+1+1 feltételeknek felel meg.**

A kiemelt verzió az egybites összeadó specifikációja az összeg és átvitel bitekre. S == F1 és Co==F2

AZ ÖSSZEG LOGIKAI FÜGGVÉNY

TÁBLÁZATOSAN

BEMENETEK				KIM
INDX	A	B	C	F1
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

KARNAUGH TÁBLÁBAN

A \ B C	00	01	11	10
0	0	1	0	1
1	1	0	1	0

AZ ÁTVITEL LOGIKAI FÜGGVÉNY

TÁBLÁZATOSAN

BEMENETEK				KIM
INDX	A	B	C	F2
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

KARNAUGH TÁBLÁBAN

A \ B C	00	01	11	10
0	0	0	1	0
1	0	1	1	1

A Karnaugh táblák csak tájékoztatásként láthatók, nem tananyag!!!
A függvények különböző algebrai alakjai:

- S=F1 = /A*/B*C+/A*B*/C+A*/B*/C+A*B*C — DNF, SOP
- S=F1'=A XOR B XOR C — XOR forma
- S=F1 = (A+B+C)*(/A+B+/C)*(A+/B+/C)*(/A+/B+C) — CNF, POS
- Co=F2 = /A*B*C +A*/B*C +A*B*/C +A*B*C — DNF, SOP
- Co=F2' = A*B + B*C + A*C — SOP, de nem DNF
- Co=F2 = (A+B)*(B+C)*(A+C) — POS, de nem CNF

ahol a rövidítések jelentései

DNF = Diszjunktív Normál Alak (Mintermek összege)
SOP = Sum of Products (ÉS kifejezések összege)
CNF = Konjunktív Normál Alak (Maxtermek szorzata)
POS = Product of Sum (VAGY kifejezések szorzata)

és az F1' ill. F2' megfelelnek a legegyszerűbb, minimalizált verzióknak (ha a XOR is megengedett). A minimalizált függvények a következő algebrai átalakítással érhetők el:
F1 = /A*/B*C+/A*B*/C+A*/B*/C+A*B*C
C és /C kiemelésével F1 = C * (/A*/B + A*B) + /C * (/A*B + A */B)
A második zárójeles kifejezésben már látszik a XOR függvény mintája: (/A*B + A*/B) = (A XOR B)
Az első zárójeles kifejezés kicsit nehezebb, de azért, hogy a fenti mintát a C változóval való XOR kapcsolathoz elérjük, néhány lépést kell tennünk. Szeretnénk egy olyan kifejezést, ami az A XOR B invertáltja, azaz /(A XOR B), kiindulva az /A*/B + A*B kifejezésből. Alkalmazva az involúciót (kétszeres invertálás), majd a De Morgan szabályt
//(/A*/B + A*B) = /(/(/A*/B)+(A*B)) = /(/(/A*/B)*/(A*B)) = /((A+B)*(/A+/B)) =

Elvégezve a zárójelen belüli műveleteket /(A*/A + A*/B+B*/A+B*/B), az önmagukkal képzett szorzatok kiesnek, marad /(A*/B + /A*B), ami pontosan az /(A XOR B) keresett kifejezés.

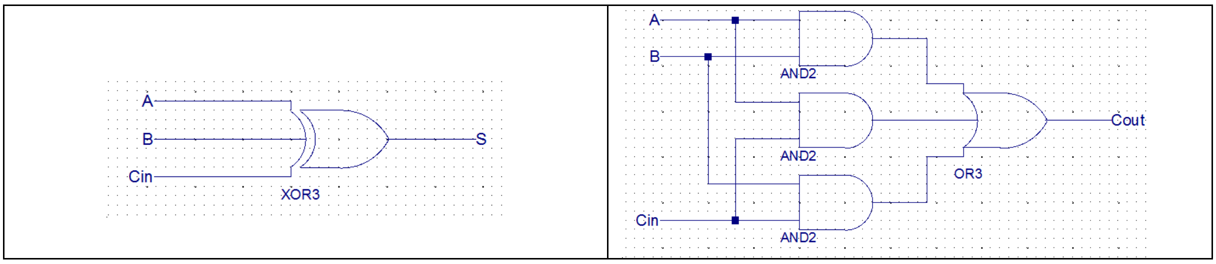
Tehát F1 = C* /(A XOR B) + /C * (A XOR B) = A XOR B XOR C.

Az F2 minimális alakja lényegesen egyszerűbb, mindenki önállóan levezetheti..

Ezeket a funkciókat Verilog HDL szintaktikával kifejezve, a C bemenetet Ci átvitel bemenetként jelölve:

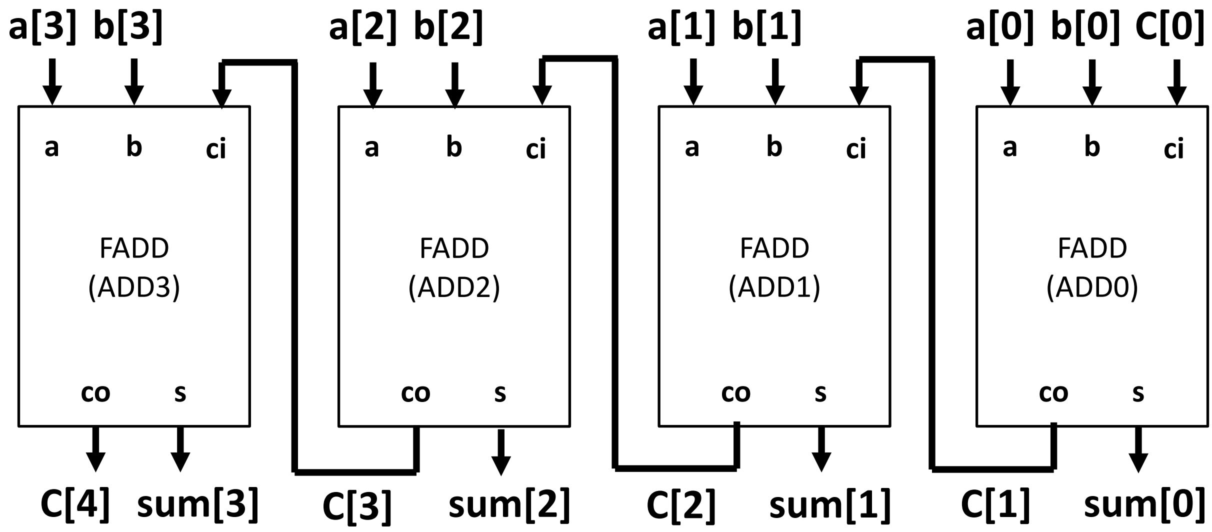
```
assign s  = a ^ b ^ ci;
assign co = a & b | a & ci | b & ci;
```

Kapcsolási rajz formában:
Az S összeg (bal oldali rajz), és a Co átvitel logika (jobb oldali rajz)



Az alábbi rajz pedig mutatja az egybites teljes összeadó egységek felhasználásával felépíthető 4 bites ADDER modult, az átviteli lánc kialakításával.

Ez a 4 bites összeadó a tárgya a Lab2_2 feladatnak, ha valaki érdeklődésből előzetesen foglalkozni kíván a feladattal.


a[3] b[3]
a[2] b[2]
a[1] b[1]
a[0] b[0] C[0]
a b ci
FADD
(ADD3)
(ADD2)
(ADD1)
(ADD0)
co s
C[4] sum[3]
C[3] sum[2]
C[2] sum[1]
C[1] sum[0]