# I. modul: Kombinációs hálózatok

## 3. lecke: Egy- és többkimenetű kombinációs hálózatok tervezése

**Cél:**

A lecke célja, hogy a hallgató képes legyen az előző leckékben tanult függvényfelírási és egyszerűsítési módszerek segítségével egy- és többkimenetű kombinációs hálózatok algebrai és kapu szintű megtervezésére.

**Követelmények:**

Ön akkor sajátította el megfelelően a tananyagot, ha

- képes megadni egy teljesen vagy nem teljesen specifikált kombinációs hálózat mintermes vagy maxtermes algebrai függvényalakját,
- a De-Morgan féle szabályok szükségszerű alkalmazásával képes a specifikáció szerinti konjunktív vagy diszjunktív függvényformulák egymás közti átkonvertálására
- meg tudja határozni egy specifikus kombinációs hálózat legegyszerűbb függvényalakját és kapu szintű realizációját

**Időszükséglet:**

A tananyag elsajátításához kb. 90 percre van szükség.

**Kulcsfogalmak:**

- minimalizálás, legegyszerűbb függvényalak
- redundáns és irredundáns lefedés
- közös implikáns
- NAND-NAND realizáció

Tevékenység: tanulmányozza a példákban található teljesen és nem teljesen specifikált Karnaugh táblákon létrehozható redundáns és irredundáns lefedési lehetőségeket, valósítsa meg kapu szinten a leckében előforduló függvényeket.

## 3.1 Egykimenetű kombinációs hálózatok tervezése

### 3.1.1 Teljesen specifikált egykimenetű kombinációs hálózatok tervezése

Az egykimenetű, teljesen specifikált kombinációs hálózatot egyetlen, teljesen határozott logikai függvénnyel specifikáljuk. Ennek meghatározása történhet igazságtábla megadásának segítségével, Karnaugh táblás ábrázolással, az ’1’-es mintermek számjegyes felsorolásával vagy algebrai alak megadásával. A tervezés szükséges lépései a következők:

1. igazságtábla és/vagy Karnaugh tábla megadása,
2. függvény kiolvasás és egyszerűsítés a Karnaugh táblán (a minimalizálás céljából),
3. a rendelkezésre álló logikai építőelemekhez igazodó függvényalak felírás (átalakítás),
4. kapuszintű realizáció a rendelkezésre álló logikai építőelemek segítségével.

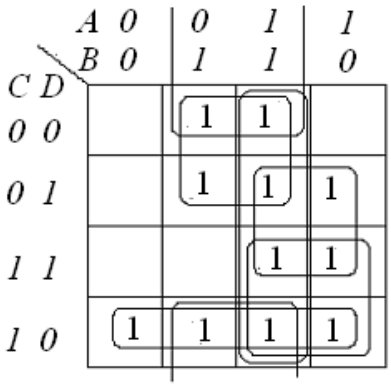
Igazságtábla vagy számjegyes minterm felsorolás esetén az ’1’-es mintermek táblázatba vitele után megindulhat a prímimplikánsok megkeresése, és a szükséges prímimplikánsok kiválasztása. Algebrai alak esetén a megadott szorzattermek Karnaugh táblán történő ábrázolása után célszerű egy egyszerűbb lefedést találni.

***Példa:***

Tervezzük meg NEM-ÉS (NAND) kapukkal a következő specifikációval megadott négybemenetű(A,B,C,D), egykimenetű(F) teljesen specifikált kombinációs hálózatot!

$$F(\text{'1'}) : (2, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15)$$

Az 3.1.1.a ábra mutatja a Karnaugh táblát az összevonásokkal, illetve a lehetséges prímimplikánsokkal:




*3.1.1.a ábra[1]*

*Az F függvény lehetséges prímimplikánsai*

Kiolvasva a prímimplikánsokat:

$$B\overline{D},\quad B\overline{C},\quad AD,\quad AC,\quad C\overline{D},\quad AB$$

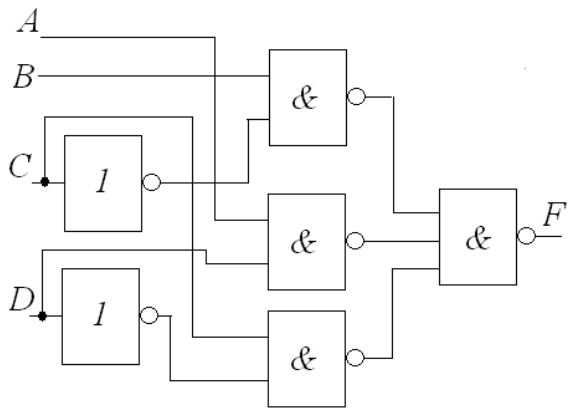
Meghatározva a minimális irredundáns lefedést :

$$B\overline{C},\quad AD,\quad C\overline{D}$$

Mivel a feladatkiírás szerint a realizációt tisztán NAND kapuk felhasználásával kell elvégezni, ezért a kapott függvényalakra a következő De-Morgan átalakítást kell alkalmaznunk:

$$F = \overline{\overline{B\overline{C} + AD + C\overline{D}}} \quad = \overline{\overline{B\overline{C}} \cdot \overline{AD} \cdot \overline{C\overline{D}}}$$

Az áramköri realizáció rajza a 3.1.1.b ábrán látható:




*3.1.1.b ábra[1]*

*NAND-NAND realizáció*

*Megjegyzés: NAND realizáció esetében az inverter alkalmazása megengedett, mind kvázi NEM-ÉS tag (tekinthető egy két bemenetét közösített NEM-ÉS kapu elemnek).*

## 3.1.2 Nem teljesen specifikált egykimenetű kombinációs hálózatok tervezése

Az egykimenetű, nem teljesen specifikált kombinációs hálózatot egyetlen, nem teljesen határozott logikai függvénnyel specifikáljuk. Ennek meghatározása történhet igazságtábla megadásának segítségével, Karnaugh táblás ábrázolással, az ’1’-es és a közömbös (don’t care = ’-’) mintermek számjegyes felsorolásával vagy algebrai alak megadásával. A tervezés szükséges lépései a következők:

1. igazságtábla és/vagy Karnaugh tábla megadása,
2. függvény kiolvasás és egyszerűsítés a Karnaugh táblán (a minimalizálás céljából),
3. a rendelkezésre álló logikai építőelemekhez igazodó függvényalak felírás (átalakítás),
4. kapuszintű realizáció a rendelkezésre álló logikai építőelemek segítségével.

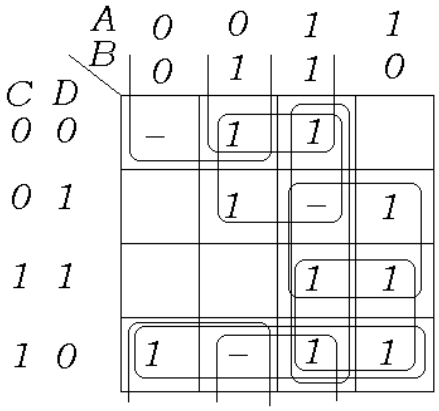
***Példa:***

Tervezzük meg NEM-ÉS (NAND) kapukkal a következő specifikációval megadott négybemenetű(A,B,C,D), egykimenetű(F), nem teljesen specifikált kombinációs hálózatot!

$$F('1') : (2, 4, 5, 9, 10, 11, 12, 14, 15)$$

$$F('-'): (0, 6, 13)$$

Az 3.1.2.a ábra mutatja a Karnaugh táblát az összevonásokkal, illetve a lehetséges prímimplikánsokkal:




*3.1.2.a ábra[1]*

*Az F('1') és F('-') függvények lehetséges prímimplikánsai*

A Karnaugh táblából kiolvasható prímimplikánsok :

$$\overline{A}\,\overline{D},\quad B\,\overline{D},\quad B\,\overline{C},\quad AD,\quad AC,\quad C\,\overline{D},\quad AB$$

A minimális irredundáns lefedés termjei:

$$B\overline{C}, \quad AD, \quad C\overline{D}$$

Így a végleges, közömbös mintermek felhasználásával felírt *F('1')* logikai függvény termek logikai összegével felírt alakja:

$$F1 = B\overline{C} + AD + C\overline{D}$$

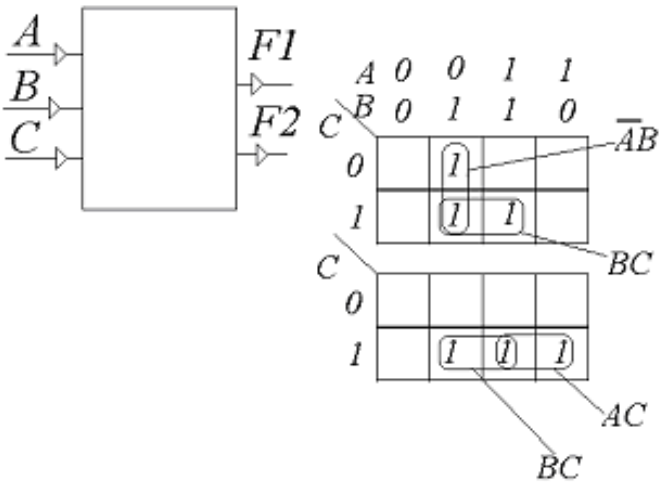
Az áramköri realizáció –az előző példa megoldásaként adódott végső függvényalakkal történő egyezőség miatt- azonos a 3.1.1.b ábrán láthatóval.

## 3.2 Többkimenetű kombinációs hálózatok tervezése

A többkimenetű, például 'm' kimenetű kombinációs hálózatot 'm' számú logikai függvénnyel adunk meg. Lehetséges volna, ha ezeket egymástól teljesen függetlenül egyszerűsítenénk és realizálnánk. Ennél azonban sokszor vannak egyszerűbb alakra vezető megoldások is. Ennek illusztrálására tekintsük a következő tervezési feladatotokat.

***Példa1:***

adott egy három(*A,B,C*) bemenetű és két(*F1, F2*) kimenetű kombinációs hálózat 3.2.a ábrán látható Karnaugh táblák szerinti specifikációja:




*3.2.a ábra[1]*

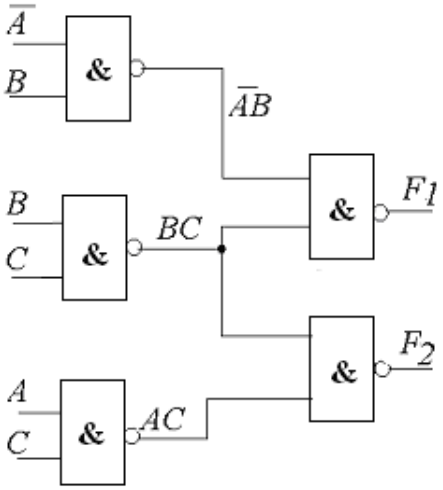
*Egy három bemenetű és két kimenetű kombinációs hálózat Karnaugh táblás specifikációja*

Ha a két kimenet függvényének prímimplikánsait egymástól függetlenül keressük meg, a következő két kifejezést kapjuk:

$$F_1 = \overline{A}B\overline{C} + \overline{A}BC + ABC = \overline{A}B + BC$$
$$F_2 = \overline{A}BC + A\overline{B}C + ABC = AC + BC$$

A két függvény prímimplikánsai között találunk egy közös, mindkét lefedésben prímimplikáns szerepet játszó termet, a *'BC'*-t. Ezt nyilvánvalóan csak egyszer, azaz egy ÉS vagy egy NEM-ÉS kapuval realizáljuk, hiszen mindkét második szinten levő VAGY, illetve NEM-ÉS kapuba bevezethető az ezt reprezentáló logikai jel (3.2.b ábra). A közös prímimplikánsokat tehát célszerű megkeresni. Arra is gondolnunk kell azonban, hogy nemcsak a közös prímimplikánsok egyszeri megvalósítása egyszerűsítheti a realizációt, hanem a közös implikánsok is. Ezek közül a legnagyobbakat érdemes megkeresni.

Fontos igazság, hogy a legnagyobb közös implikánsokat a függvények szorzatának lefedésével találjuk meg. A szorzatfüggvény prímimplikánsai között ott vannak a kért függvény legnagyobb közös implikánsai, amelyek között természetesen a közös prímimplikánsok is ott vannak. A szorzatfüggvény lefedése nagyon egyszerű, a Karnaugh táblába bejegyezzük azokat a mindkét függvényben '1'-es értékkel szereplő mintermeket. Így minden egyes függvény lefedéséhez nemcsak a saját prímimplikánsokat, hanem a legnagyobb közös implikánsokat is számításba vesszük.

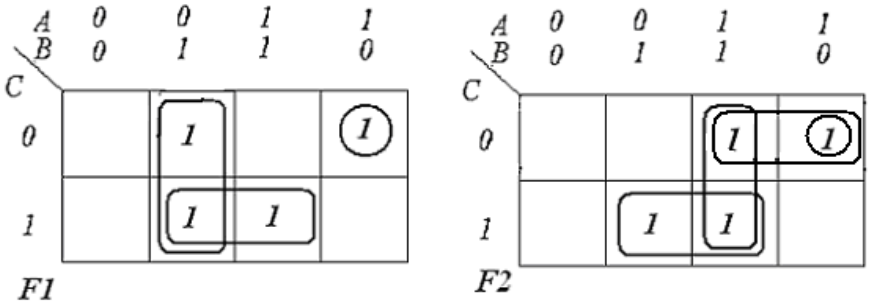



*3.2.b ábra[1]*

*NAND realizáció a közös prímimplikáns(BC) egyszeri megvalósításával*

***Példa2:***

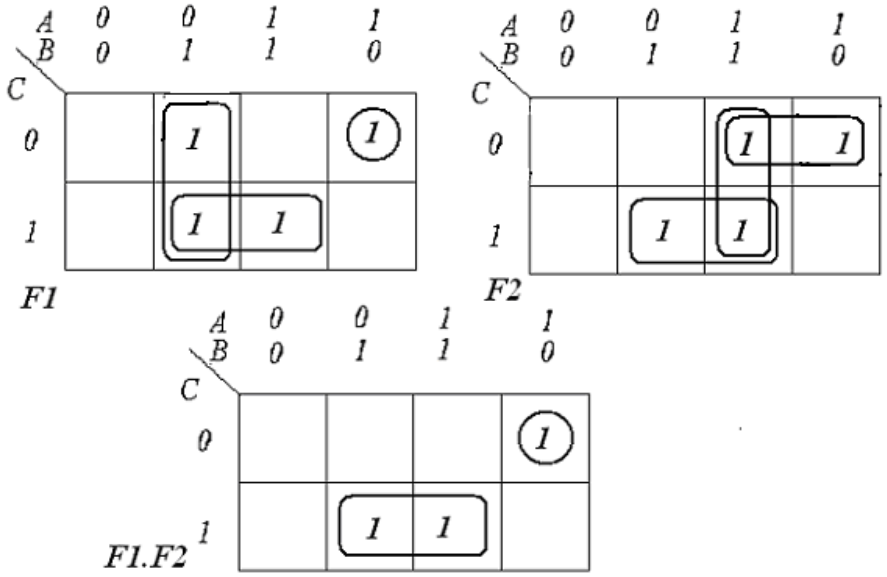
adott egy három(*A,B,C*) bemenetű és két(*F1, F2*) kimenetű kombinációs hálózat 3.2.c ábrán látható Karnaugh-táblák szerinti specifikációja:




*3.2.c ábra[1]*

*'A,B,C' bemenetű és 'F1','F2' kimenetű kombinációs hálózat Karnaugh táblás specifikációja*

Az előző példánkban találtunk egy közös, mindkét függvényben szereplő prímimplikánst, és ezt csak egyszer kellett realizálnunk. Ebből következik, hogy a két kimenetet nem célszerű egymástól függetlenül egyszerűsíteni. Az előző pontban leírtak alapján a két függvény legnagyobb közös implikánsait megkapjuk, ha előállítjuk a szorzat függvény prímimplikánsait. Ezek között a közös prímimplikánsok is megjelennek (3.2.d ábra).




*3.2.d ábra[1]*

*Legnagyobb közös implikánsok keresése a szorzat-függvény prímimplikánsainak kijelölésével*

A prímimplikáns készleteket a következő halmazok alkotják:

$$F_1: \quad \overline{A}B, \quad BC, \quad A\overline{B}\,\overline{C}$$

$$F_2: \quad BC, \quad AB, \quad A\overline{C}$$

$$F_1 \cdot F_2: \quad BC, \quad A\overline{B}\,\overline{C}$$

A ’*BC*’ közös prímimplikáns, az $A\bar{B}\bar{C}$ pedig két nem közös prímimplikáns közös része, azaz egy legnagyobb közös implikáns. A két-kimenetű hálózat függvényeinek lehetséges lefedő termjeit tehát az 3.2.c ábrán látható csoportokból állítjuk össze.

Az 3.2.c ábra alapján elvégzett lefedés-vizsgálat és a felesleges implikánsok eltávolítása során előnyben részesítjük a legnagyobb közös implikánsokat. Így az *F2* teljes lefedéséhez és felépítéséhez az

$$A\,\overline{C}$$

helyett a közös

$$A\,\overline{B}\;\overline{C}$$

termet használjuk. Természetes, hogy a ’*BC*’-t csak egyszer kell realizálni.

***<u>A fentiek több kimenetre való általánosítása alapján megfogalmazható a több-kimenetű hálózatok egyszerűsítésének lépéssorozata:</u>***

1. *felírandó valamennyi kimenethez rendelt függvény prímimplikánsa,*
2. *az összes lehetséges függvény-szorzat prímimplikánsainak megadása,*
3. *az egyes kimeneti függvények mintermjeinek lefedése a saját, más kimenetekhez nem tartozó prímimplikánsokkal, illetve azokkal a maximális közös implikánsokkal, amelyek az adott függvénynek implikánsai*

Felhasznált irodalom:

[1] Dr. Keresztes Péter: Digitális hálózatok (HEFOP elektronikus Jegyzet)