

Pannon Egyetem

Villamosmérnöki és Információs Tanszék



Digitális Áramkörök

(Villamosmérnök BSc /
Mechatronikai mérnök MSc)

12. hét: Kombinációs hálózatok megvalósítása
memóriával.

Előadó: Dr. Vörösházi Zsolt

voroshazi.zsolt@virt.uni-pannon.hu

Kapcsolódó jegyzet, segédanyag:

- <http://www.virt.uni-pannon.hu>
 - Oktatás → Tantárgyak → Digitális Áramkörök (Villamosmérnök BSc / Mechatronikai mérnök BSc/MSc).
- Fóliák, óravázlatok (.ppt)
- Feltöltésük folyamatosan

Kombinációs hálózatok / szekvenciális hálózatok (LSI/MSI):

- **1.) Memóriák: RAM, ROM – nagy memóriatömbök alkalmazása K.H.-ban**
- **2.) Programozható logikai áramkörök alkalmazása K.H., illetve S.H. felépítésére:**
 - PAL (Programmable AND Logic)
 - PLA (Programmable Logic Array)
 - PROM (Programmable OR – Read Only memory)
 - GAL (Generic Array Logic)
 - ↓
 - PLD / CPLD (Complex Programmable Gate Array)
 - ↓
 - FPGA (Field Programmable Gate Array)



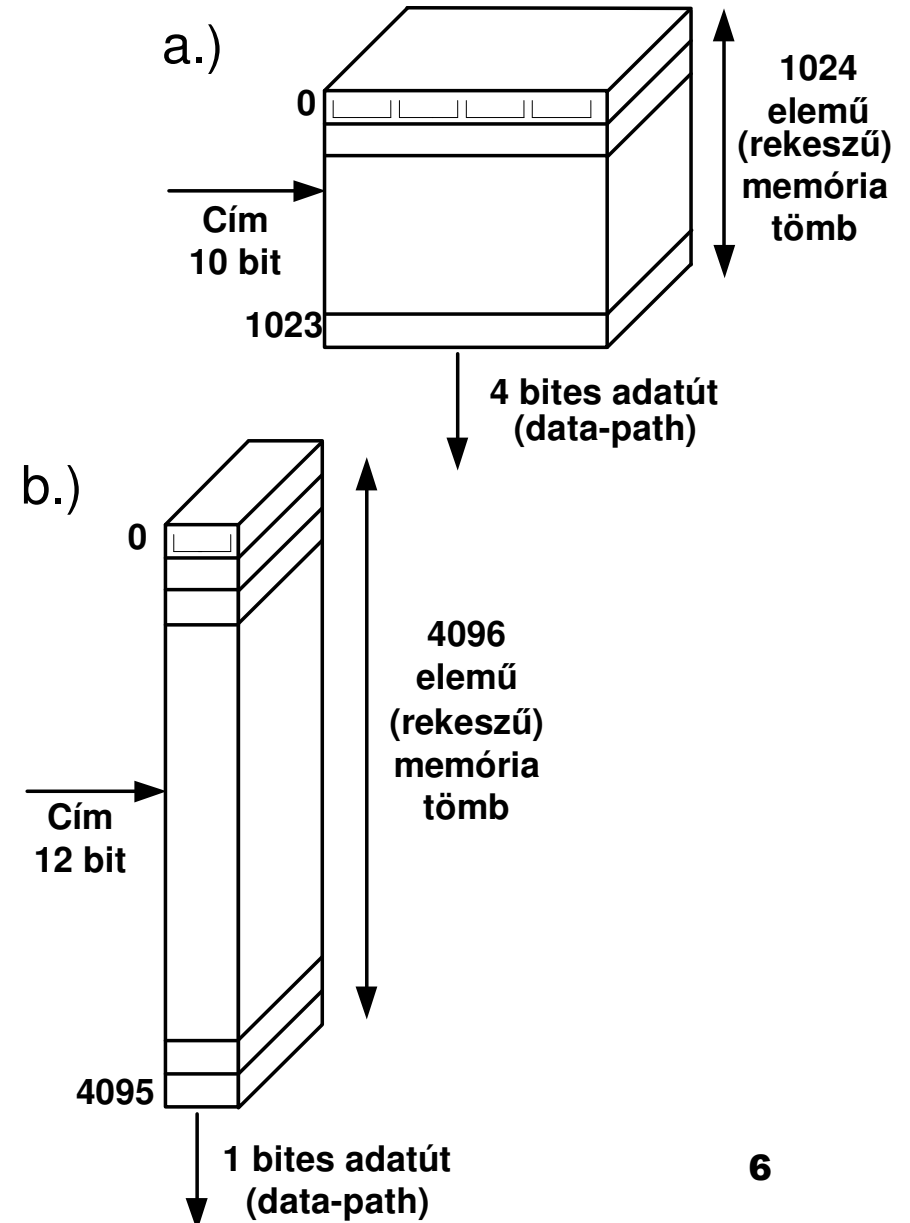
Kombinációs hálózatok megvalósítása memóriákkal

1.) ROM: Read Only Memory

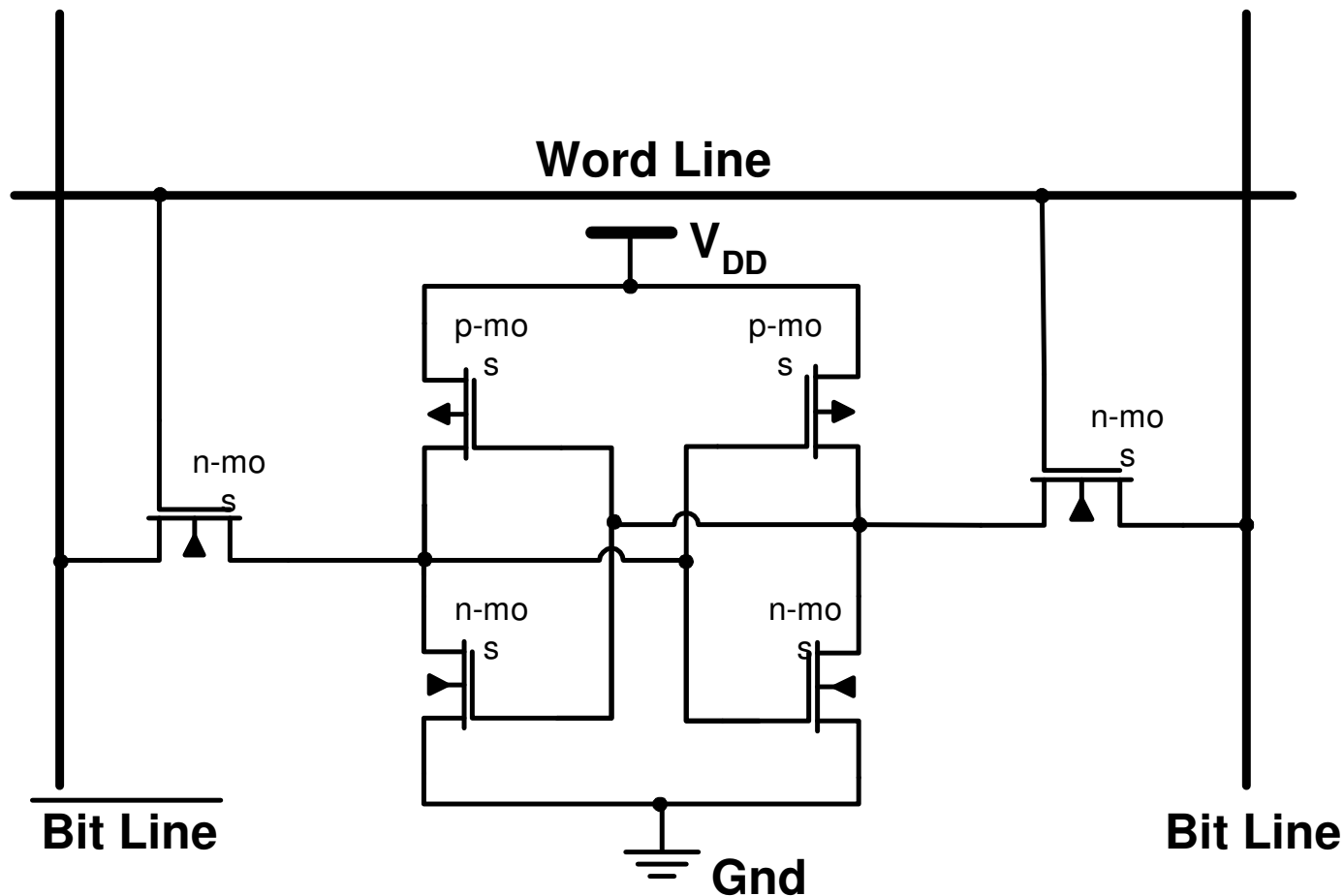
- Csak egyszer írható (általában a gyártó által), utána már csak olvasható memória.
- Kikapcsoláskor megőrzik tartalmukat!
- Szervezése a RAM-hoz hasonló (de itt nem kell Write/Read vonal)
- Fontos alkalmazásai:
 - *Firmware*: elektronikai eszközök (pl. számítógép ROM-Bios funkcióinak tárolására)
 - *Kód konverter*: pl. BCD generáló 7 szegmenses kijelzőre
 - *Logikai függvény generátor*: tetszőleges logikai fgv. szintézisénel (előállításánál) ROM-ban tároljuk az igazságtáblázatot
- Programozható ROM fajtái:
 - PROM*: egyszer programozható
 - EPROM*: programozható, és UV-fénnyel törölhető
 - EEPROM*: elektromosan programozható/törölhető (Flash)

2.) RAM: Random Access Memory

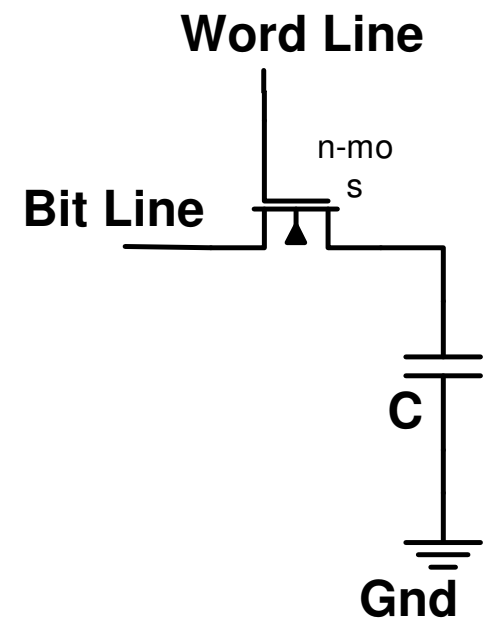
- Véletlen hozzáférésű, írható és olvasható memória.
- Címzésnél: dekódoló és multiplexer áramköröket használunk
 - Pl. 10 lábbal -> 1024 (1K) cellát tudunk megcímezni
- Jelölés: RAM celláinak száma (4096) / tároló kapacitás 4K. Példák:
 - a.) 4K = 1K x 4 RAM tartalmaz: 1024 számú 4 bites szót (word)
 - b.) 4K = 4K x 1 RAM tartalmaz: 4096 számú 1-bites szót (word) (de itt az 1 bit/szó szervezéssel lábszámot spórolunk!) 12+1



RAM: **SRAM (Statikus)** és **DRAM** (dinamikus) memória cellák felépítése:



SRAM: n-mos és p-mos (n és p csatornás tranzisztorokból épül fel) 2-2 db, + 2 db áteresztő tranzisztor (össz. CMOS tranzisztor)



DRAM: Tárolás: egy kis méretű C kondenzátor töltése és kisütése.

a.) SRAM tulajdonságai

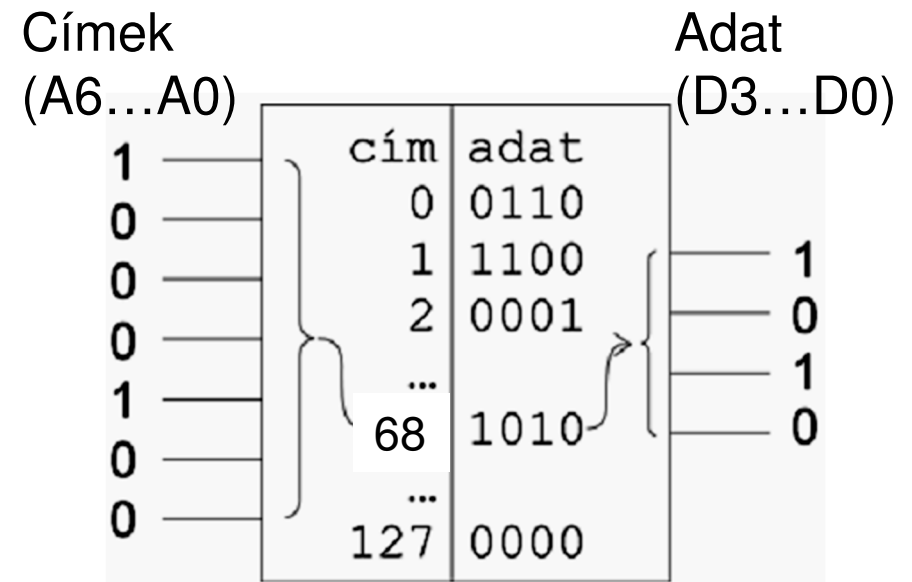
- Az információ a tápfesz. alatt is megmarad, nem kell frissíteni.
- Megvalósítható bipoláris (0,1) SRAM cellával, nMOS tranzisztorokkal, vagy CMOS tranzisztorokból (6 tranzisztor).
- Kisebb kapacitású, de gyorsabb a DRAM-nál, mivel tápfeszültség alatt sem kell frissíteni.
 - SRAM esetén a ciklusidők $T(\text{R/W Cycle}) \approx T(\text{Access Time})$ közel azonosak.
- Nagy a fogyasztása. Integritási sűrűsége 4x rosszabb. Tápfeszültség kikapcsolásával elveszti a tartalmát.
- Felhasználása:
 - Cache memóriákban, digitális oszcillátorokban, logikai analizátorokban, operatív tárukban (memória), merevlemezek gyorsító pufferében, nyomtatók memóriájában.

b.) DRAM tulajdonságai

- Általában CMOS technológiával készülnek.
- A tápfeszültség alatt is frissíteni kell, mivel idővel elvesztik tartalmukat.
 - kb. *2-14 ns*-ként (mai memóriák esetén),
- Kicsi a fogyasztása
- Nagyobb kapacitású, mint az SRAM, de lassabb (frissítés!)
 - A hozzáférési idő kétszer nagyobb a memória R/W ciklusidejénél: $2 \cdot T(\text{R/W Cycle}) = T(\text{Access Time})$.
- Egyszerűbb felépítésű (1 tranzisztor + kondenzátor), szemben az SRAM-al. Integritási sűrűsége **4x** nagyobb. (IRAM: az időzítő elektronika a DRAM-ra van integrálva).
- Itt lényegében a CS=Chip Select (korábban CE: Chip Enable!) jelet két részre osztották fel: **RAS=sorkijelölő**, és **CAS=oszlopkijelölő** komponensekre.
- Felhasználása: operatív memória (DDR-, DDR-II, DDR3- DDR4 SDRAM)

Memória működése röviden

- Egy adott rekesz tartalmát úgy tudjuk kiolvasni, hogy a **címbemenetekre** (Address pins) adjuk annak sorszámát (bináris formában), amelyre az **adatkimeneteken** (DATA pins) a véges ún. **hozzáférési idő** (access time) elteltével megjelenik a rekeszben tárolt szám.
 - *Arató könyvben *ciklus időnek* van definiálva, amelyet a szakirodalomban inkább tekintenek két egymást követő olvasási, vagy írási tranzakció között eltelt időnek
- Beírás hasonló módon történhet.



Memória tulajdonságai

- A memória kapacitása a tárolható szavak számát jelöli. Ha 'n' darab *címvezetékünk* van, a memória 2^n db rekeszt tartalmaz. A tárolás jellege alapján kétféle típust különböztetünk meg (később részletesen):
 - RAM
 - ROM
- Az egy rekeszben tárolható bitek száma adja a memória **szóhosszúságát (word length)**.

Kombinációs hálózatok és a memóriák kapcsolata

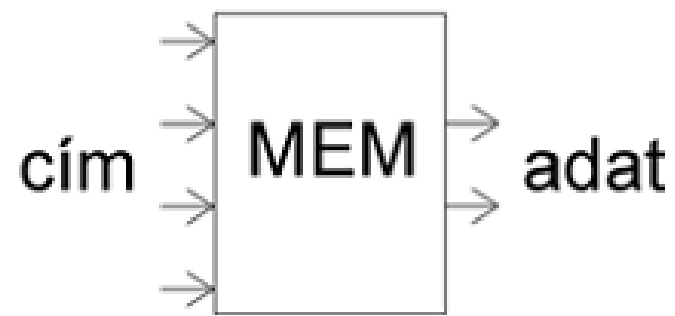
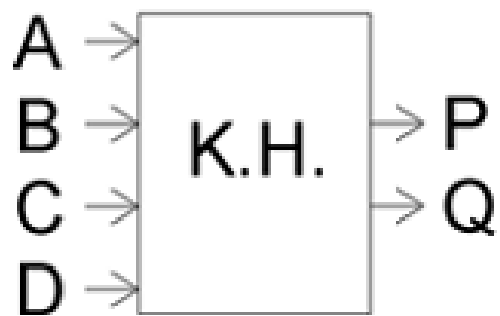
- A K.H-ban egy adott *bemeneti kombinációra* az igazságtábla ugyanazon sorában feltüntetett *kimeneti bináris kombináció a válasz*.
 - (Tehát beadunk egy bináris számot, amire válaszul egy másik bináris számot várunk a kimeneteken).
- A memóriák feladata is teljesen ugyanez: kiolvasáskor minden egyes *cím megadásakor* egy előzőleg betöltött *adat* jelenik meg. Vagyis ha egy memóriát egy vele azonos számú be-, és kimenettel rendelkező K.H. igazságtáblája szerint töltünk fel, akkor ez a **memória helyettesítheti a kombinációs hálózatot**.

K.H. – Memória ekvivalencia

A	B	C	D	P	Q
0	0	0	0	1	1
0	0	0	1	1	0
0	0	1	0	0	1
	...				
1	1	1	0	1	0
1	1	1	1	0	0



cím	adat
0000	11
0001	10
0010	01
...	
1110	10
1111	00



n=4

m=2

A memóriaelemmel történő megvalósítás **előnyei:**

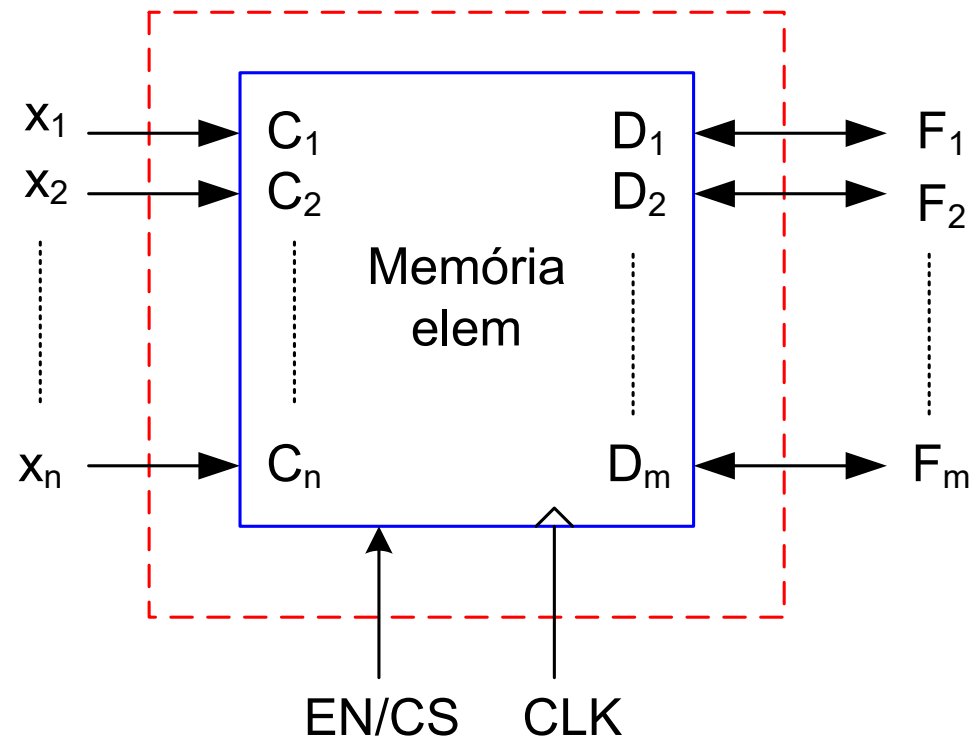
- könnyen átprogramozható, így a fejlesztési szakaszban nem kell újraépítenünk kapukból az egész áramkört, még apró változtatásnál sem,
- nem igényel függvény-egyszerűsítést (minimalizálást),
- **nem fordulhat** elő benne *statikus és dinamikus* hazard
 - mivel nem kapukból és huzalozással építjük fel, hanem az igazságtábla alapján direkt módon határozzuk meg (betöltjük) a beírandó adatokat!
- !DE itt is lehet *funkcionális hazard*, amit pl. szinkronizációval szüntethetünk meg (nem szomszédos bemeneti címváltozásokra → kimeneti adatváltozások)
 - Erre megoldás az EN engedélyező/tiltó bemenet.

A memóriaelemmel történő megvalósítás hátrányai:

- Egy memória-áramkör sokkal lassabb lehet, mint a logikai kapukból összeállított huzalozott / dedikált kombinációs hálózat (főleg ha több hierarchia szinten összekötött memória áramkörök késleltetését tekintjük),
- speciális időzítési feltételekkel fogadhat csak jeleket (pl. a címnek bizonyos ideig stabilnak kell maradnia, hogy előálljon a kimenet),
- míg a függvény-egyszerűsítéssel kapott megoldás esetleg csak néhány kapuból állna, addig a memóriába a teljes igazságtáblázatot be kell programozni (tárolni):
 - n darab bemenethez mindenképp egy 2^n kapacitású memóriát kell választani (katalógus),
- a memóriaelem a legtöbb esetben drágább is.

Memória elem alkalmazása több kimenetű K.H. megvalósítására

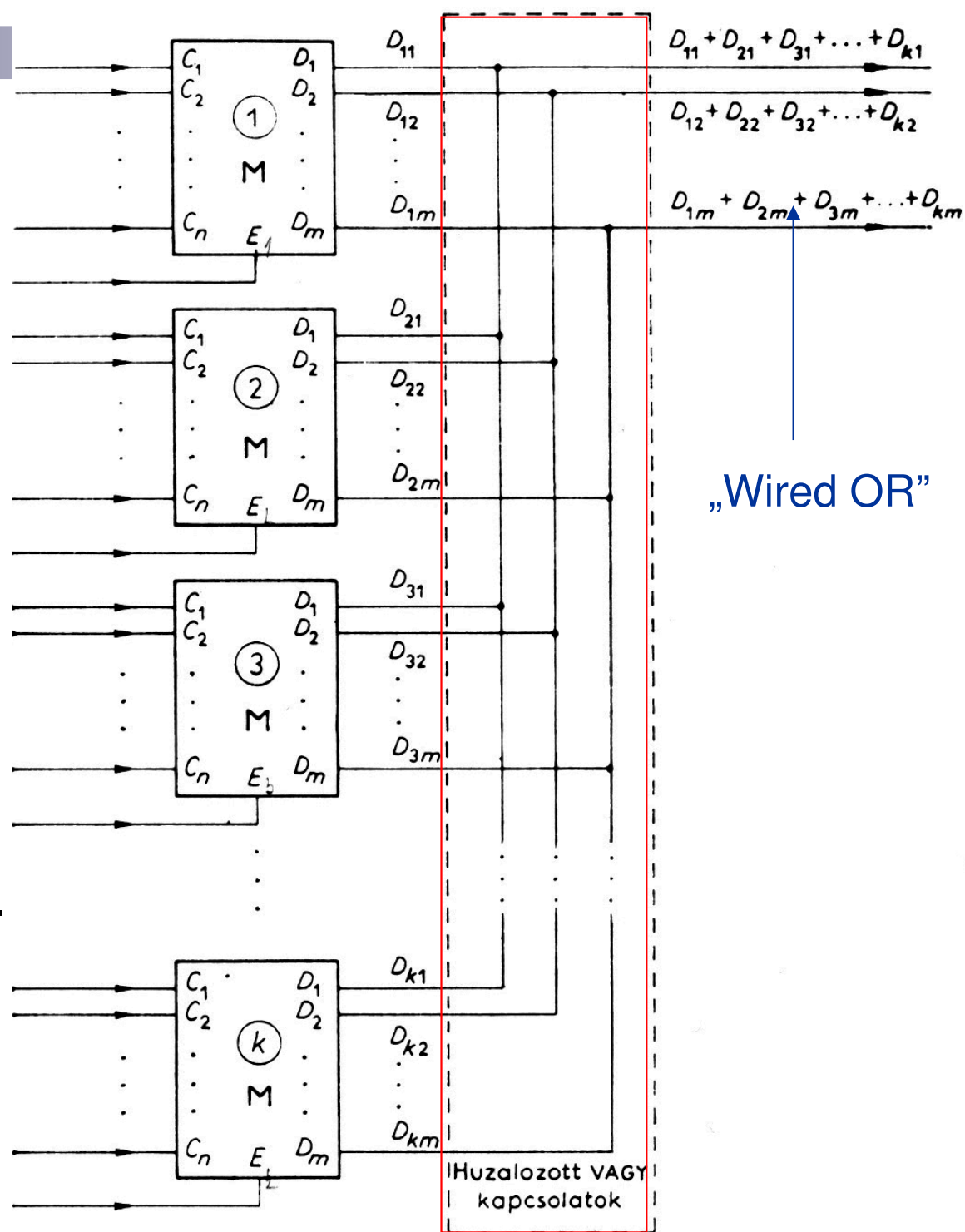
- K.H. ↔ Memória
 - Bemeneti kombináció ↔ *címek*
 - Kimeneti kombináció ↔ *adat* kimenetek
- EN/CS (Enable, más néven Chip Select) – memória elem engedélyezése, vagy tiltása 1
 - **Huzalozott VAGY kapcsolat:** akár több memória elem kimenete is összekapcsolható „huzalozott” módon.



Memória:

„Huzalozott VAGY” kapcsolatok

- Mindig csak egy memóriaelem bemenete engedélyezett egyszerre, a többié tiltott!
- $EN_i = '1'$ engedélyezés.
- $EN_{j \neq i} = '0'$ tiltás

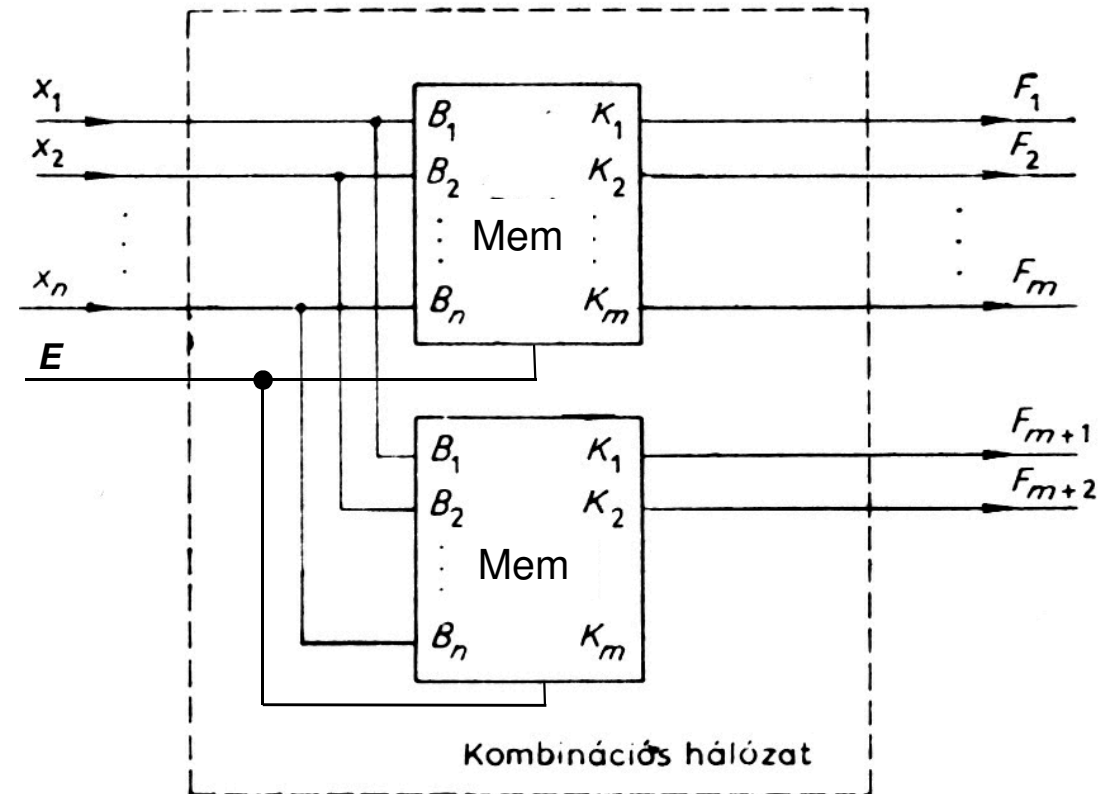


K.H. megvalósítása memóriával – lehetséges esetek

- 1.) megvalósítandó K.H. *bemenete* kevesebb, mint a memória címbemenete (nem használjuk a memória plusz cím bemeneteit, GND lekötés)
- 2.) megvalósítandó K.H. *kimenete* kevesebb, mint a memória adatkimenete (nem használjuk a memória plusz adat kimeneteit, GND lekötés)
- 3.) megvalósítandó K.H. *kimenete* több, mint a memória adatkimenete (több memória elem kell)
- 4.) megvalósítandó K.H. *bemenete* több, mint a memória címbemenete (több memória elem, és dekóder kell)
- 5.) megvalósítandó K.H. *bemenete* és *kimenete* is több, mint a memória címbemenete, ill. adatkimenete (több memória elem, és dekóder kell)

3.) K.H. megvalósítása memóriával

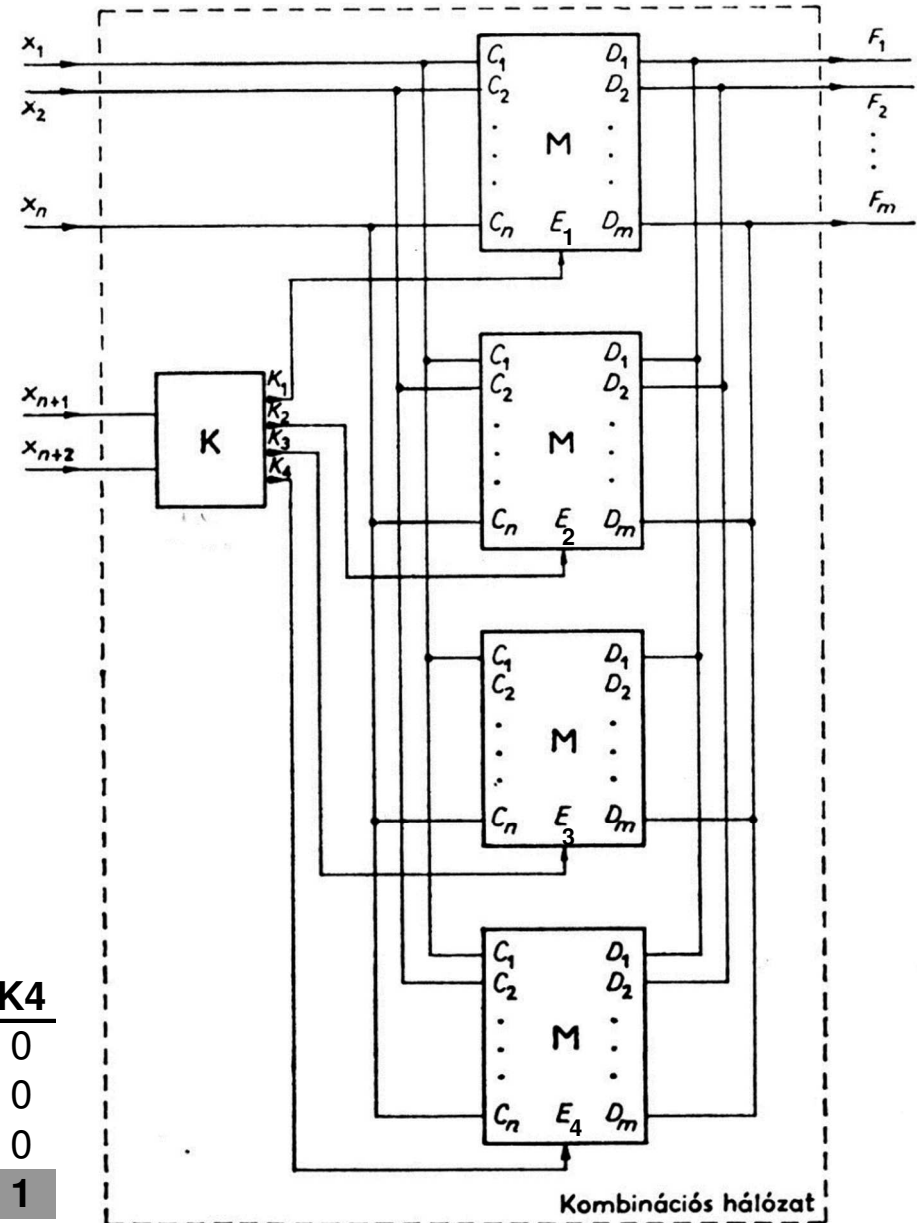
- Ha a megvalósítandó K.H. kimeneteinek száma több (pl. $m+2$), mint a memória adatkimeneteinek száma (pl. m):
 - Egyetlen memória elem nem elegendő! Bővíteni kell.



4.) K.H. megvalósítása memóriával

- Ha a megvalósítandó K.H. bemeneteinek száma $(n+2)$ több, mint a memória cím-bemeneteinek száma (n) :
 - Egyetlen memória elem nem elegendő! Bővíteni.
 - Mivel 2^n számú bináris kombináció helyett 2^{n+2} -t kell tárolni (azaz $4 \cdot 2^n$)
 - Bővíteni: 'K' átkódoló K.H. beépítése vezérli a memóriák E engedélyezését
 - $K_{1...4}$ kimeneteken „*n-ből 1 kód*” (one-hot-code)

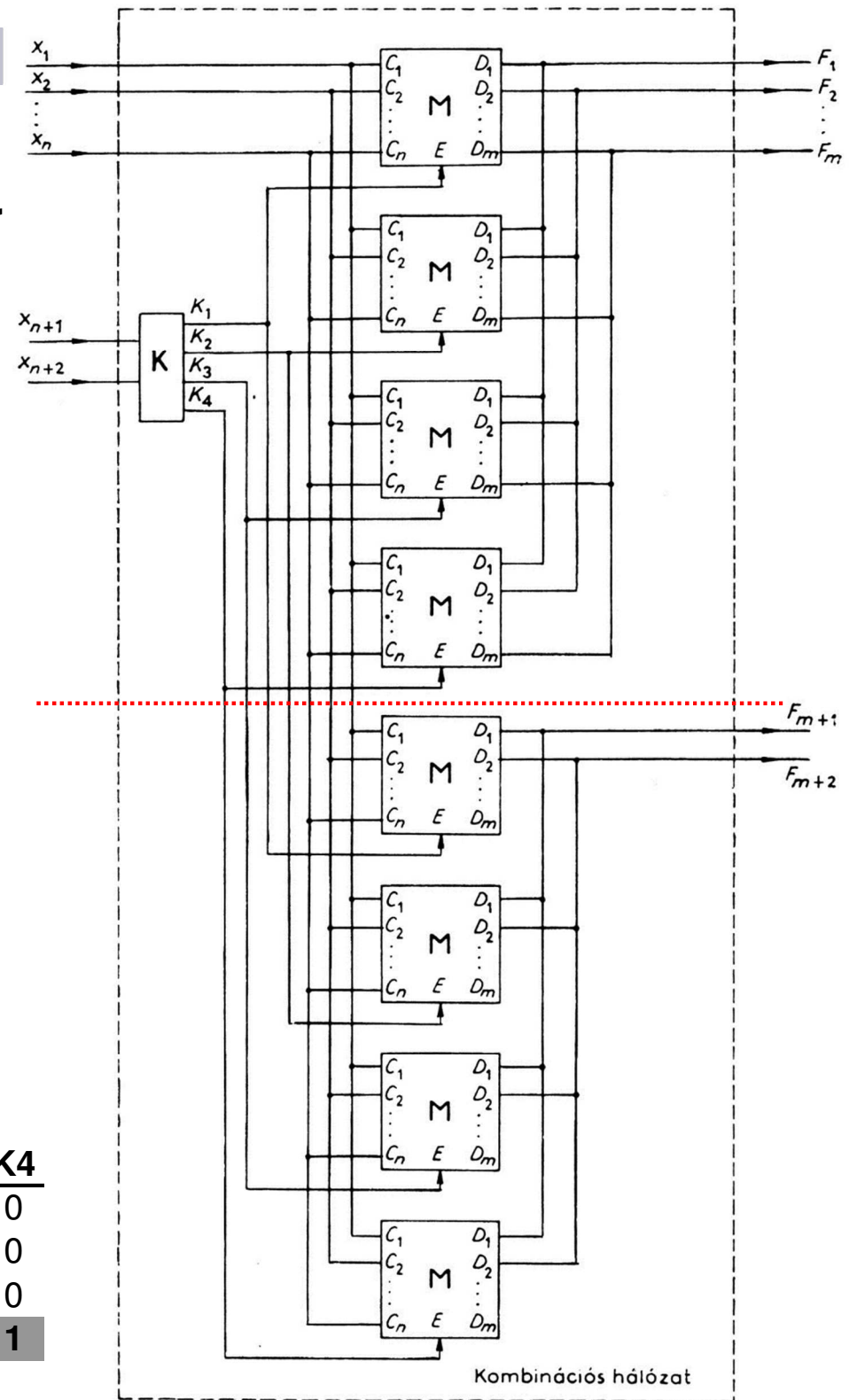
$x(n+1)$	$x(n+2)$	K1	K2	K3	K4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



5.) K.H. megvalósítása memóriával

- Ha a megvalósítandó K.H. bemeneteinek ($n+2$) és kimeneteinek száma ($m+2$) is több (tfh. 2-vel), mint a memória cím-bemeneteinek (n), és adatkimeneteinek száma (m):
 - Egyetlen memória elem nem elegendő! Bővíteni.
 - Előző 3.) és 4.) módszereket együttesen kell alkalmazni a bővítéshez.

$x(n+1)$	$x(n+2)$	K1	K2	K3	K4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



K.H. megvalósítása memóriával

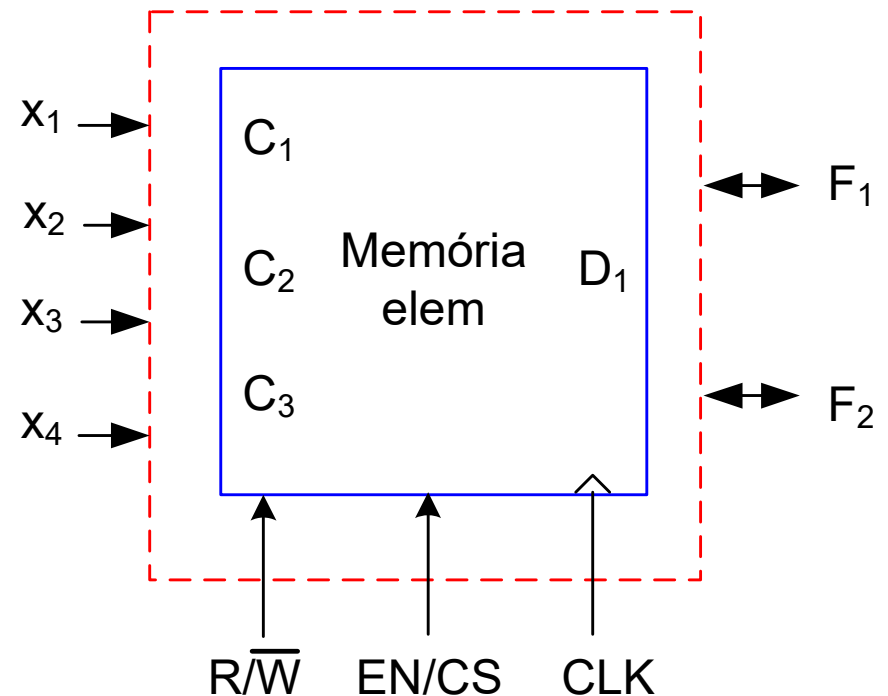
- **Megjegyzés:** nincs akadálya annak, hogy a 'K' átkódoló vagy elő-feldolgozó logikai hálózatot memóriával (tipikusan ROM-al) valósítsanak meg.
- **De** ekkor a több-szintű, sorba kapcsolt memória hálózat miatt *csökkenhet* a megvalósított hálózat működési sebessége (cím -> érvényes adat megjelenése a kimeneten stb.).

Példa 1.):

- A megadott Memória elem segítségével realizáljon egy K.H.-ot, ha adottak az alábbi függvényei:

$$F_1^{n=4}(x_1, x_2, x_3, x_4) = \sum_{i=0}^{2^n-1} (0, 1, 4, 15)$$

$$F_2^{n=4}(x_1, x_2, x_3, x_4) = \sum_{i=0}^{2^n-1} (0, 1, 8, 9, 11)$$



- a.) Ha szükséges, adja meg a dekódoló logika ('K') igazságtáblázatát is !
- b.) Adja meg pontosan a memória feltöltését is!

Példa 1. (folyt.):

- Mivel a Memória elem cím-bemeneteinek száma ($n=3$) 1-el kevesebb, illetve az adat-kimenetének száma ($m=1$) is 1-el kevesebb, mint a realizálandó K.H. bemeneteinek ($n=4$), ill. kimeneteinek ($m=2$) száma, ezért bővíteni kell:

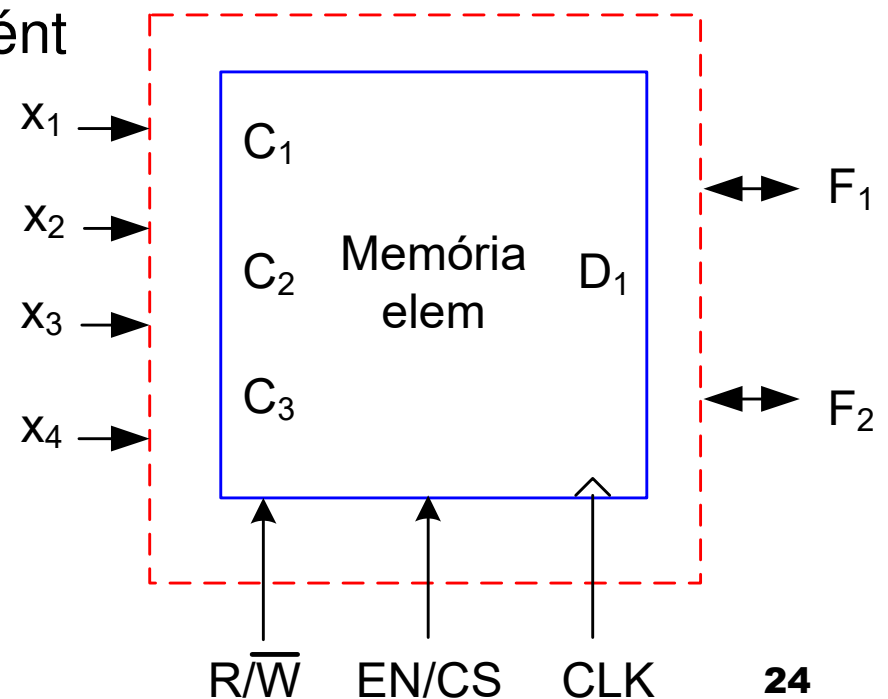
- Több memória elem

- 1-1 memória elem kimenetenként

- Dekódoló logika 'K'

- $2^{n+1} = 2 \cdot 2^n$ (1 → 2 dekóder),
azaz K1, K2 engedélyezők

Összesen 4 db memória elem
kell!



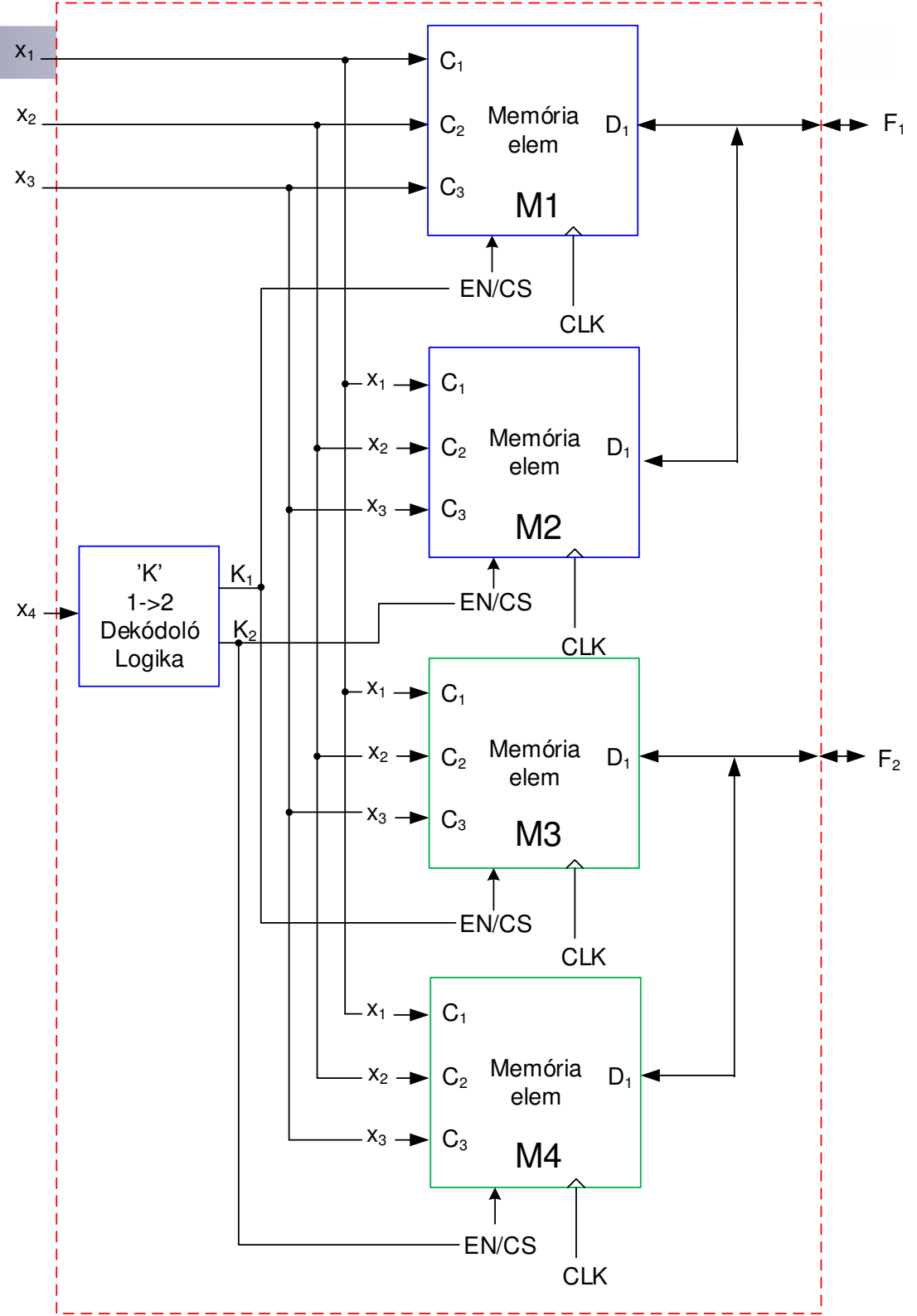
Példa 1. Megoldás

■ a.) K: dekódoló logika

$x(4)$	K1	K2
0	1	0
1	0	1

□ K1: M1, M3

□ K2: M2, M4



b.) A memória elemek feltöltése

$$F^{n=4}_1(x_1, x_2, x_3, x_4) = \sum_{i=0}^{2^n-1} (0, 1, 4, 15)$$

	Cím	F1
MEM#1	0	1
	1	1
	2	0
	3	0
	4	1
	5	0
	6	0
	7	0

$$F^{n=4}_2(x_1, x_2, x_3, x_4) = \sum_{i=0}^{2^n-1} (0, 1, 8, 9, 11)$$

	Cím	F2
MEM#3	0	1
	1	1
	2	0
	3	0
	4	0
	5	0
	6	0
	7	0

MEM#2	8	0
	9	0
	10	0
	11	0
	12	0
	13	0
	14	0
	15	1

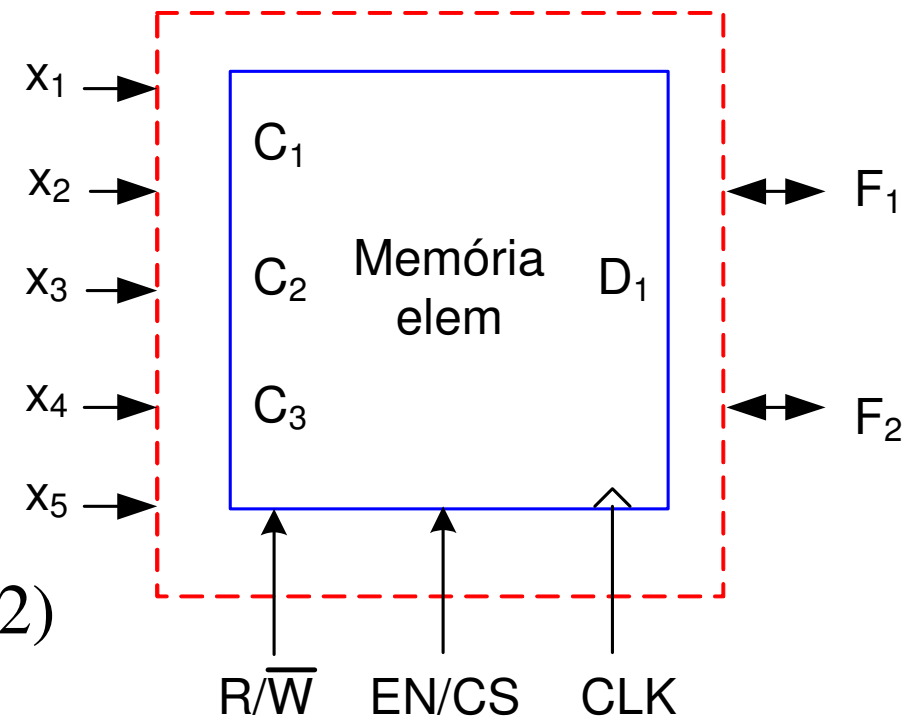
MEM#4	8	1
	9	1
	10	0
	11	1
	12	0
	13	0
	14	0
	15	0

Példa 2.):

- A megadott Memória elem segítségével realizáljon egy K.H.-ot, ha adottak az alábbi függvényei:

$$F_1^5(x_1, \dots, x_5) = \sum_{i=0}^{2^n-1} (0, 1, 4, 31)$$

$$F_2^5(x_1, \dots, x_5) = \sum_{i=0}^{2^n-1} (0, 1, 4, 15, 16, 17, 21, 22)$$



- a.) Ha szükséges, adja meg a dekódoló logika ('K') igazságtáblázatát is !
- b.) Adja meg pontosan a memória feltöltését is!