

Pannon Egyetem

Villamosmérnöki és Információs

Rendszerek Tanszék



Digitális Rendszerek és Számítógép Architektúrák (BSc államvizsga tétel)

3. tétel: Vezérlő egységek, programozható
logikai eszközök (modell – implementáció)

Összeállította: Dr. Vörösházi Zsolt

voroshazi@vision.vein.hu

Jegyzetek, segédanyagok:

- Tétel: Informatika / Programtervező Informatikus / Gazdaságinformatikus BSc alapszakos hallgatóknak (2012. május)
- Könyvfejezetek:
 - <http://www.virt.uni-pannon.hu>
→ Oktatás → Tantárgyak → Digitális Rendszerek és Sz.gép Arc. / Számítógép Architektúrák
(chapter05.pdf)

Vezérlő egységek általánosan

- A számítógép vezérlési funkcióit ellátó *szekvenciális* egység.
- **Feladata**: az operatív tárban lévő gépi kódú utasítások értelmezése, részműveletekre bontása, és a szekvenciális (sorrendi) hálózat egyes funkcionális részeinek vezérlése (vezérlőjel- és cím-generálás)
- Vezérlő egység tervezési lépései:
 - megfelelő technológia, és rendszerkomponensek kiválasztása
 - komponensek összekapcsolása a működési sorrendnek megfelelően
 - RTL leírás alkalmazása az akciók ill. adatátvitel pontos leírására
 - **adatút (data-path)** megtervezése (*legfontosabb!*)
 - kívánt vezérlő jelek azonosítása, meghatározása

Vezérlő egységek fajtái:

- Huzalozott (klasszikus módszerrel):
 - Mealy-modell,
 - Moore-modell (Példa: FIR szűrő tervezése – FSM állapotgép segítségével).
 - Multiplexeres / késleltetési / Shift-regiszteres megvalósítások
- Mikroprogramozott (reguláris vezérlési szerkezettel):
 - Horizontális mikrokódos vezérlő,
 - Vertikális mikrokódos vezérlő.
- Programozható logikai eszközök (PLD):
 - Makrocellás típusok: PLA, PAL, PROM, CPLD,
 - FPGA: Field Programmable Gate Array: tetszőlegesen újrakonfigurálható kapu-áramkörök

Kombinációs hálózatok

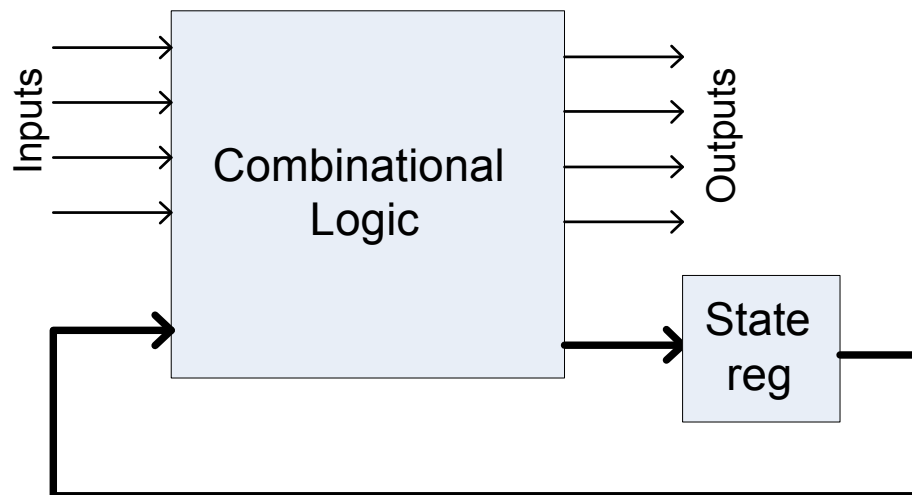
- **(K.H.) Kombinációs logikai hálózatról** beszélünk: ha a mindenkori kimeneti kombinációk értéke csupán a bemeneti kombinációk pillanatnyi értékétől függ (tároló „kapacitás”, vagy memória nélküli hálózatok).



Sorrendi hálózatok:

- **(S.H.) Sorrendi (szekvenciális) logikai hálózatról** beszélünk: ha a mindenkori kimeneti kombinációt, nemcsak a pillanatnyi bemeneti kombinációk, hanem a korábban fennállt bemeneti kombinációk és azok sorrendje is befolyásolja. (A *szekunder /másodlagos kombinációk* segítségével az ilyen hálózatok képessé válnak arra, hogy az ugyanolyan bemeneti kombinációkhoz más-más kimeneti kombinációt szolgáltatassanak, attól függően, hogy a bemeneti kombináció fellépésekor, milyen értékű a szekunder kombináció, pl. a State Register tartalma)

Vezérlő egységek
alapjául szolgáló
hálózat!




Időzítő – vezérlő egység:

- Az időzítő (ütemező) határozza meg a vezérlő jelek előállításának *sorrendjét*.
- Egy időzítő-vezérlő egység általános feladata az egyes funkciók megvalósítását végző áramkörü elemek (pl. ALU, memória elemek) összehangolt működésének biztosítása.
- Az időzítő-vezérlő áramkörök **szekvenciális rendszerek** – mivel az áramkörü egységek tevékenységének egymáshoz viszonyított *időbeli sorrendiségét* biztosítják – melyek az aktuális *kimenet* értékét a *bemenet*, és az *állapotok* függvényében határozzák meg.

Az időzítő-vezérlő lehet:

- **huzalozott:** áramkörökkel, *dedikált* összeköttetésekkel fizikailag megvalósított (Mealy, Moore, MUX-os modellek alapján, illetve programozható PLD-k)
- **mikroprogramozott:** az adatútvonal (data-path) vezérlési pontjait memóriából (ROM) kiolvasott *vertikális-* vagy *horizontális-*mikrokódú utasításokkal állítják be

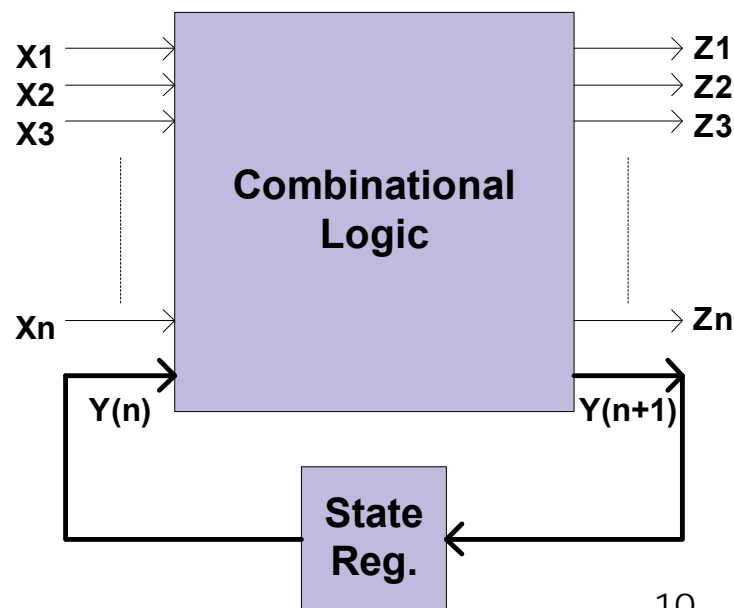


Huzalozott vezérlő egységek

- klasszikus vezérlési
struktúrák

1.) Mealy-modell

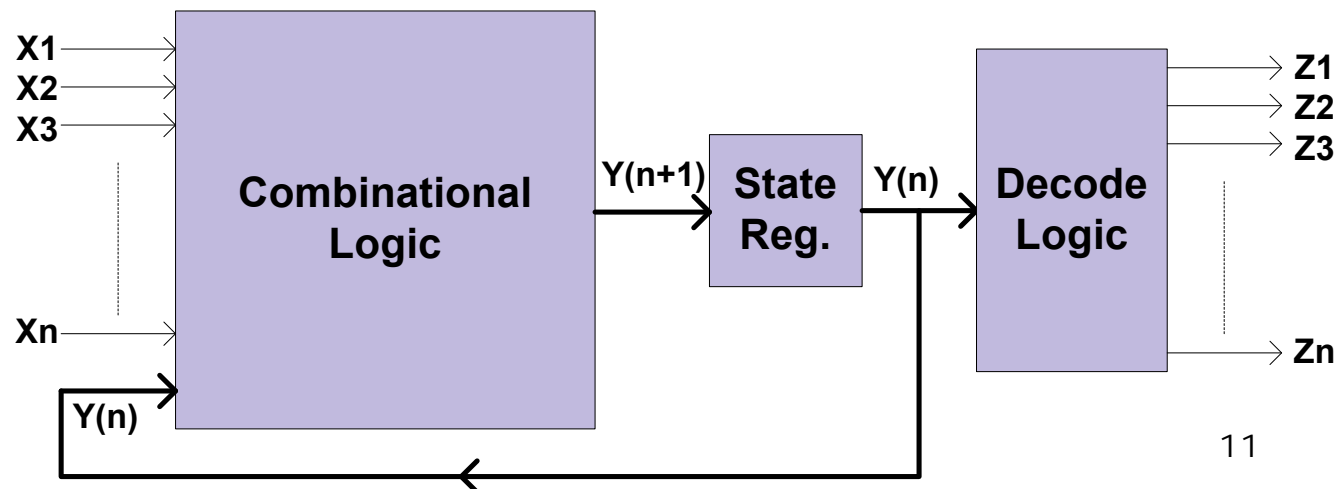
- A *sorrendi hálózatok* egyik alapmodellje. Késleltetés: a kimeneten az eredmény véges időn belül jelenik meg! Korábbi értékek visszacsatolódnak a bemenetre: kimenetek nemcsak a bemenetek pillanatnyi, hanem a korábbi állapotoktól is függenek. Problémák merülhetnek fel az állapotok és bemenetek közötti *szinkronizáció hiánya* miatt (változó hosszúságú kimenetet - dekódolás). Ezért alkalmazzuk legtöbbször a második, Moore-féle automata modellt.
- Három halmaza van: (Visszacsatolni az állapotregisztert a késleltetés miatt kell)
 - X – a bemenetek,
 - Z – a kimenetek,
 - Y – az állapotok halmaza.
- Két leképezési szabály a halmazok között:
 - $\delta(X_n, Y_n) \rightarrow Y_{n+1}$: következő állapot fgv.
 - $\mu(X_n, Y_n) \rightarrow Z_n$: kimeneti fgv.



2.) Moore-modell

- A kimenetek közvetlenül csak a pillanatnyi állapottól függenek (bemenettől függetlenek v. közvetve függenek). Tehát a kimenetet nem a bemenetekhez, hanem mindig az aktuális állapotoknak megfelelően szinkronizáljuk.
- Három halmaza van:
 - X – a bemenetek,
 - Z – a kimenetek,
 - Y – az állapotok halmaza.
- Két leképezési szabályok
 - $\delta(X_n, Y_n) \rightarrow Y_{n+1}$: köv. állapot fgv.
 - $\mu(Y_n) \rightarrow Z_n$: kimeneti fgv.

Input \Rightarrow Next-State \Rightarrow Present-State \Rightarrow Output





Mikrokódos vezérlők – reguláris vezérlési struktúrák

Ismétlés: Vezérlő egységek

- Általánosságban: a vezérlő egység feladata a memóriában lévő gépi kódú program utasításainak
 - értelmezése (decode),
 - részműveletekre bontása,
 - és ezek alapján az egyes funkcionális egységek vezérlése **(a vezérlőjelek megfelelő sorrendben történő előállítása).**

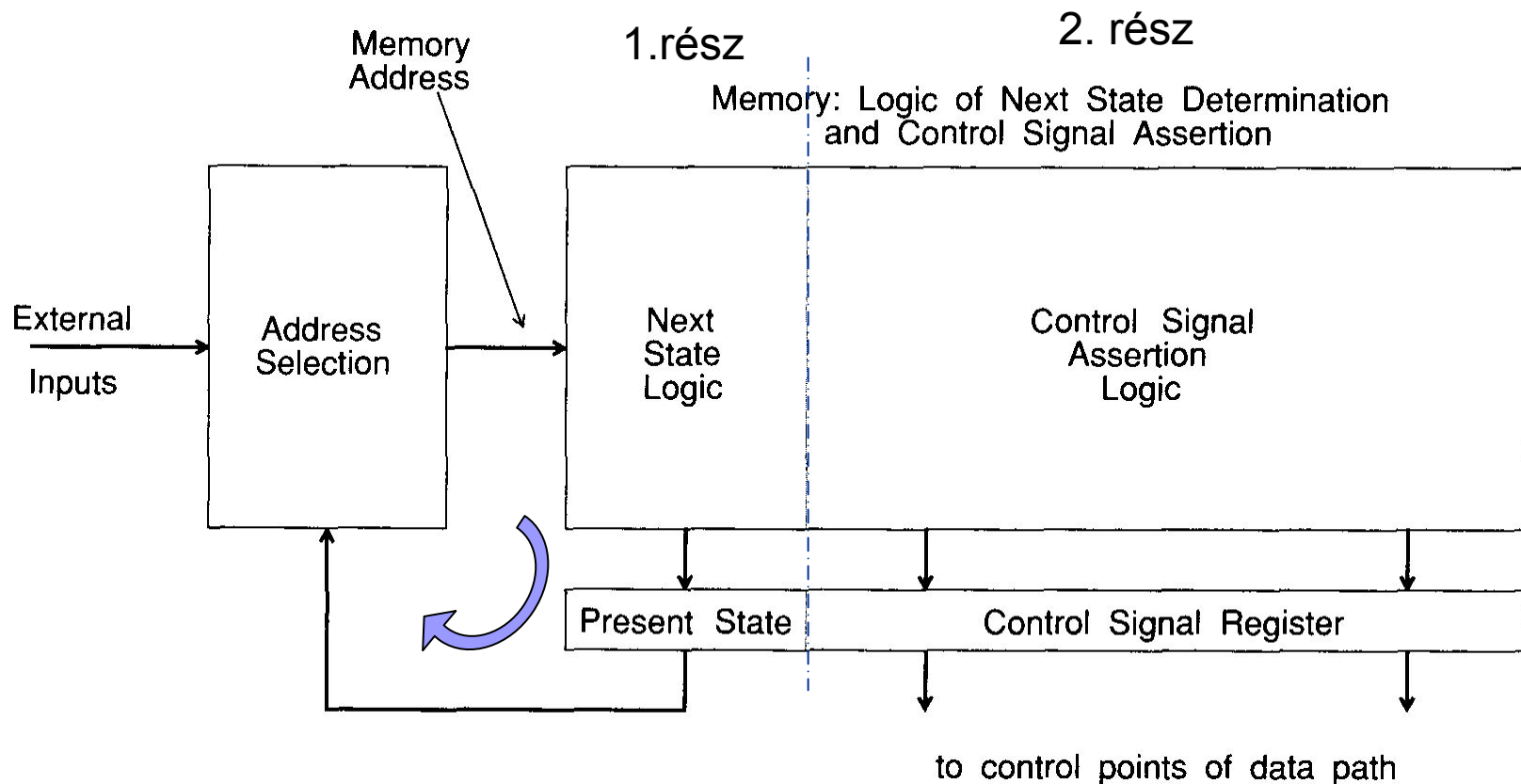
Klasszikus vagy reguláris módszer

- Eddig a „klasszikus”, késleltetéses módszereket tárgyaltuk (*huzalozott és shift-regiszteres* példákkal). A rendszer tervezésekor, miután a feladat elvégzéséhez szükséges vezérlőjeleket definiáltuk, meg kell határozni a kiválasztásuk sorrendjét, és egyéb specifikus információkat (rendszer ismeret, tervezési technikák, viselkedési leírások – pl.VHDL)
- Wilkes (1951): A komplex vezérlési folyamatokat „**reguláris módszerrel**” lehet egyszerűsíteni: nevezetesen **gyors memória elemeket** kell használni az utasítássorozatok tárolásánál. Állapotgépekkel (FSM) modellezik a vezérlő egység működését, és ezt a modellt transzformálják át **mikrokódot** használva. Az adatútvonal vezérlési pontjait memóriából (ROM) kiolvasott *vertikális- vagy horizontális-mikrokódú utasításokkal állítják be!*

Reguláris módszer: mikrokódos vezérlés tulajdonságai

- Mikrokód: *gépi kódú utasításokat (IR) → alacsonyabb szintű áramköri utasítások sorozatára leképező köztes kód*
- Szerepe: **értelmezés** (interpreter / translator) a fenti két szint között:
 - A gépi kódú utasítások változtatásának lehetősége (RISC, CISC), anélkül hogy a HW változna
 - Mai rendszerek olvasható mikrokódját gyors memóriában (általában ROM), vagy PLD-ben tárolják (írható esetben RAM, vagy Flash is lehet)

FSM megvalósítása Memóriával

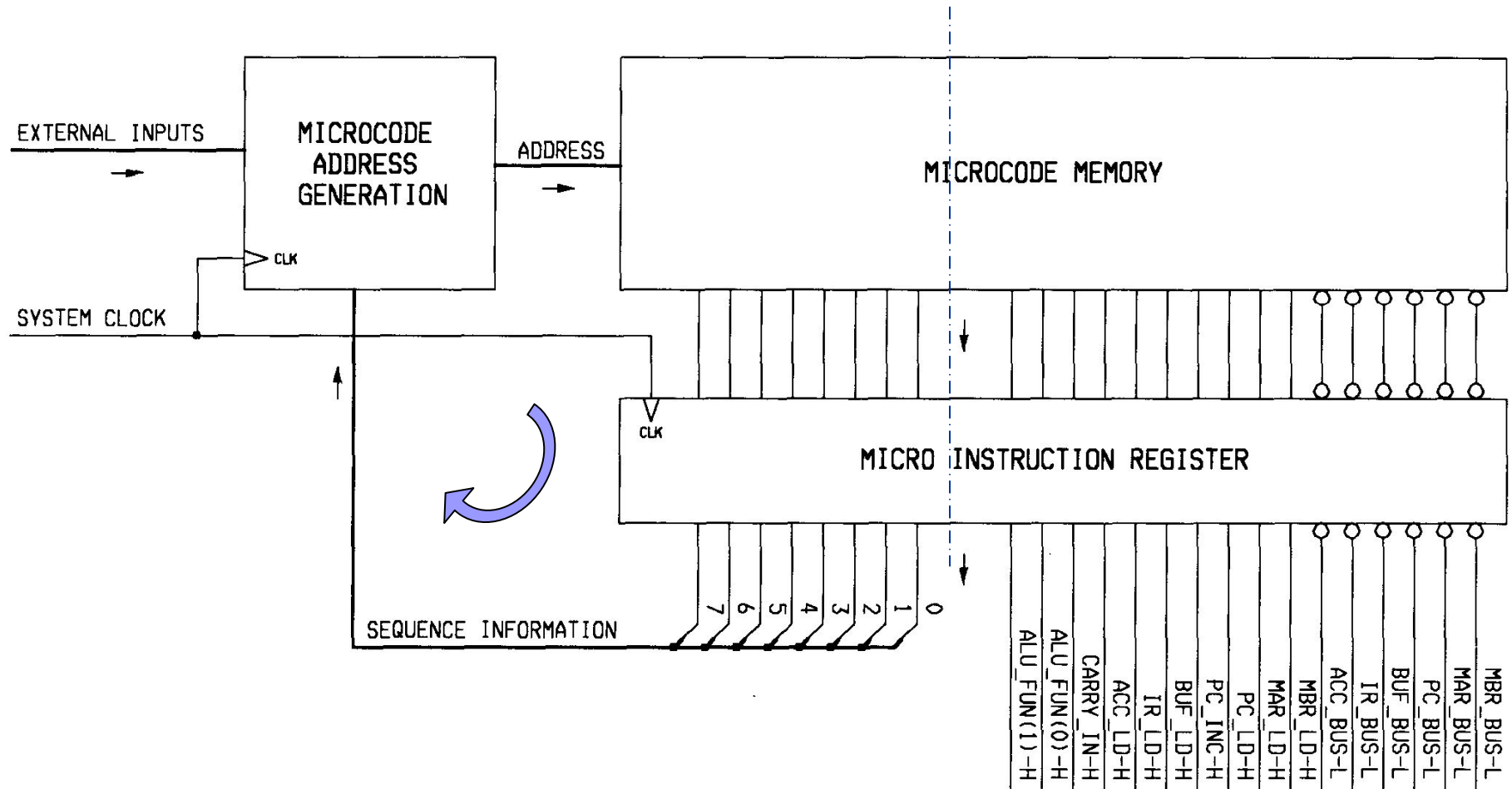


- 1.rész: szabályozza az eszköz működését a megfelelő állapotok sorrendjében
- 2.rész: szabályozza az adatfolyamot a megfelelő *vezérlőjelek beállításával* (assertion) az adatúton (vezérlési pontokon)

FSM megvalósítása Memóriával (folyt)

- **Address Selection:** (mint új elem) a következő utasítás (*Next State*), és beállítani kívánt vezérlőjel (*control signal assertion*) címére mutat a memóriában.
- A memória címet (**memory address**-t) külső bemenő jelek és a present state határozzák meg együttesen. E cím segítségével megkapjuk az adott vezérlő információ pontos helyét a memóriában, ill. ez az információ, mint új állapot betöltődik a vezérlőjel regiszterbe (*Control Signal Register*).
- **Next-State** kiválasztásához szükséges logikai memória méretét az aktuális állapotok száma, az állapotdiagram komplexitása, és a bemenetek száma határozza meg.
- **Control Signal** generálásához szükséges logikai memória méretét a bemenetek száma, a függvény (vezérlő jel) komplexitása, és a vezérlőjelek száma határozza meg.

Általános Mikrokódos vezérlő



Általános Mikrokódos vezérlő felépítése

- **Micro Instruction Register:** a „Present State” (aktuális állapot) regisztert + a Control Signal regisztert egybeolvasztja (az adatút vezérlővonalainak beállítása / kiválasztása). Mikroutasítások sorrendjében generálódik a vezérlőjel!
- **Microcode Memory:** a Control Signal Assertion Logic vezérlőjel generálás/beállítás + „Next-State” kiválasztása (mikroprogram eltárolása) összevonása
- **Microcode Address Generator:** a vezérlő jelet az aktuális mikroutasítások lépéseiként sorban generálja, de címkiválasztási folyamat komplex. **Sebesség a komplexitás rovására változhat!** (komplexebb vezérlési funkciót alacsonyabb sebességgel képes csak generálni). A következő cím kiválasztása még az aktuálisan futó mikroutasítás végrehajtása alatt végbemegy! Számlálóként működik: egyik címről a másik címre inkrementálódik (mivel a mikroutasításokat tekintve szekvenciális rendszerről van szó). Kezdetben resetelni kell.

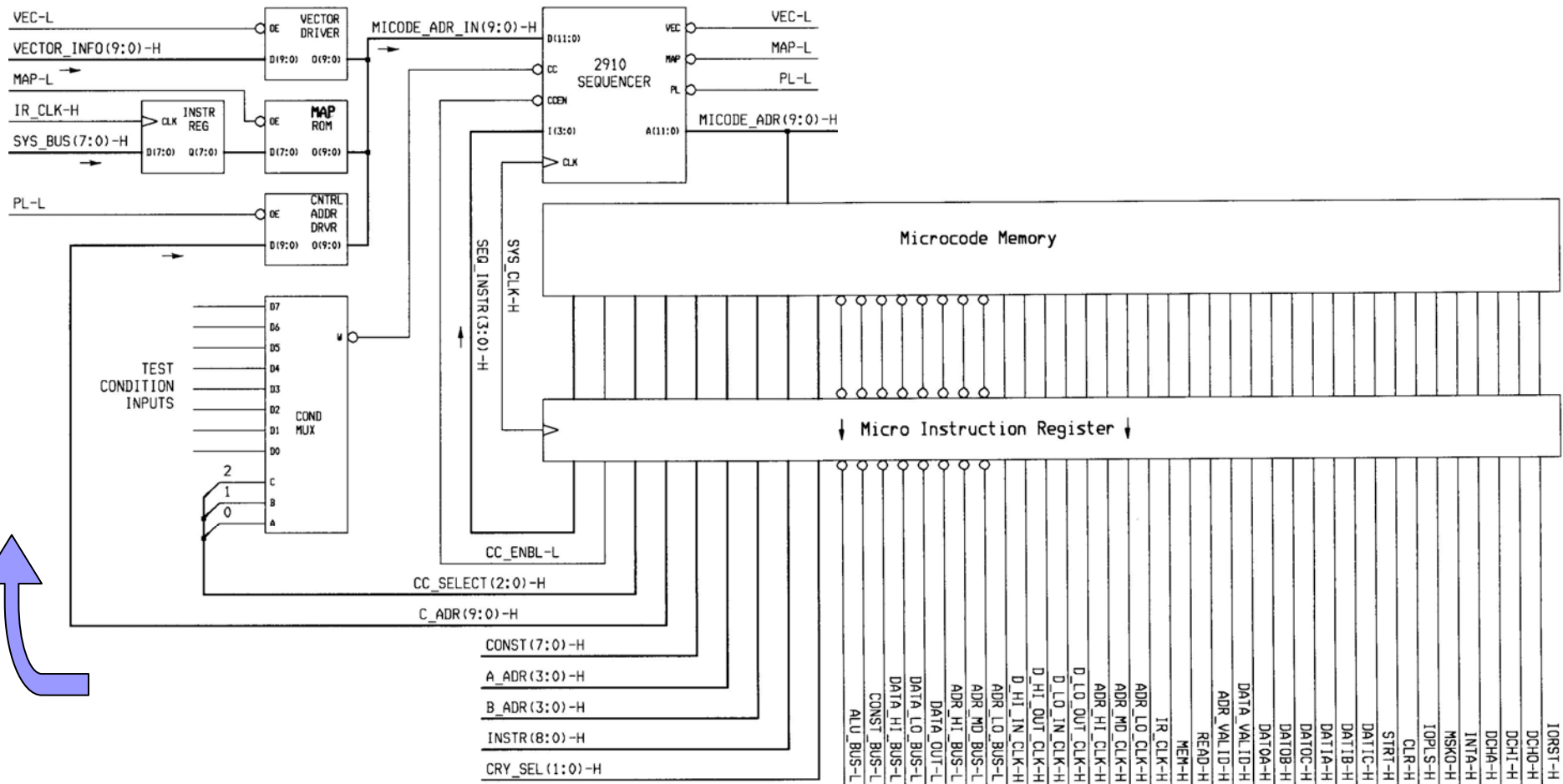
Általános Mikrokódos vezérlők tulajdonságai

- Egy gépi ciklus alatt egy **mikroprogram** fut le (amely mikroutasítások sorozatából áll). A műveleti kód (utasítás opcode része) a végrehajtandó mikroprogramot jelöli ki. A mikrokódú memória általában csak statikus módon olvasható gyárilag konfigurált ROM, ha írható is, akkor dinamikus mikroprogramozásról beszélünk.
- Ha a mikroprogram utasításai szigorúan *szekvenciálisan* futnak le, akkor a címüket egy egyszerű számláló inkrementálásával megkaphatjuk. Memóriából érkező bitek egyik része a következő *cím kiválasztását* (Sequence Information), míg a fennmaradó bitek az *adatáramlást* biztosítják.
- Mai gyors félvezető alapú memóriáknak köszönhetően kis mértékben lassabb, mint a huzalozott vezérlő egységek, mivel ekkor a memória elérési idejével ($\sim ns$) is számolni kell (nem csak a visszacsatolt aktuális állapot késleltetésével.)²⁰

1.) Horizontális mikrokódos vezérlő

- Mindenegyres vezérlőjelhez saját vonalat rendelünk, ezáltal horizontálisan megnő a mikro-utasításregiszter kimeneteinek száma, (**horizontálisan** megnő a mikrokód). Minél több funkciót valósítunk meg a vezérlőjelekkel, annál **szélesebb** lesz a mikrokód.
- Ennek köszönhetően ez a **leggyorsabb** mikrokódos technika, mivel minden bit független egymástól ill. egy mikrokóddal többszörös (konkurens) utasítás is megadható. Pl: a megfelelő regisztereket (memória, ACC) egyszerre, egyidőben tudjuk az órajellel aktiválni, ezáltal egy órajelciklus alatt az információ mindkét irányba átvihető. Növekszik a sebesség, mivel nincs szükség a vezérlőjelek dekódolását végző dekódoló logikára. Így minimálisra csökken a műveletek ciklusideje.
- Azonban nagyobb az **erőforrás** szükséglete, **fogyasztása**.

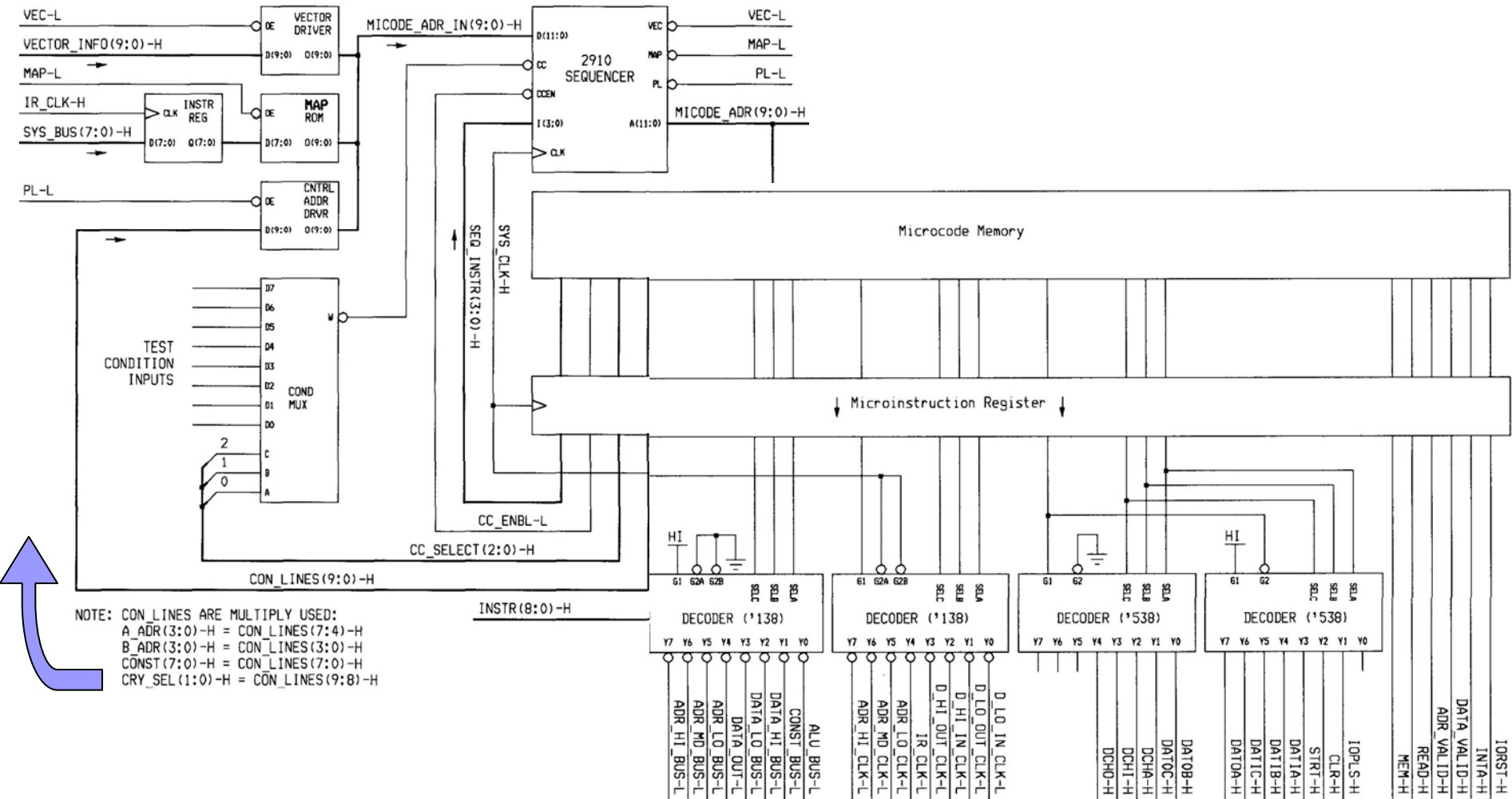
Horizontális mikrokódos vezérlő




2.) Vertikális mikrokódos vezérlő

- Nem a sebességen van a hangsúly, hanem hogy **takarékoskodjon** az erőforrásokkal (fogyasztás, mikrokódban a bitek számával), ezért is **lassabb**.
- Egyszerre csak a szükséges (korlátozott számú) biteket kezeljük, egymástól nem teljesen függetlenül, mivel közülük egyszerre csak az egyiket állítjuk be (dekódoljuk). A jeleket ezután **dekódolni** kell (több időt vesz igénybe). A kiválasztott biteket megpróbáljuk minimális számú vonalon keresztül továbbítani.
- A műveletek párhuzamos (konkurens) végrehajtása korlátozott. Dekódolás: $\log_2(N)$ számú dekódolandó bit \rightarrow N bites kimeneti busz. Több mikroutasítás szükségeltetik \Rightarrow így a mikrokódú memóriát „**vertikálisan**” meg kell növelni.

Vertikális mikrokódos vezérlő

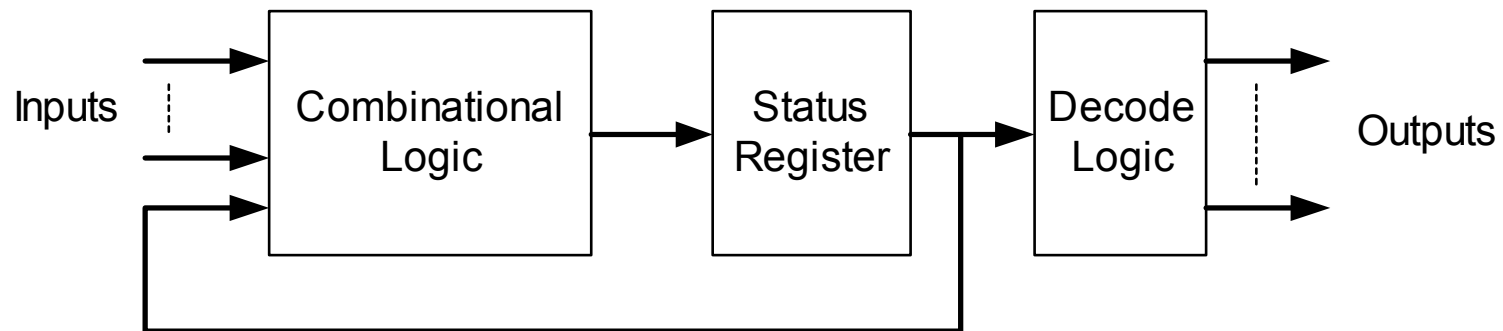




Programozható logikai eszközök (PLD)

Állapotgép FSM tervezés tulajdonságai

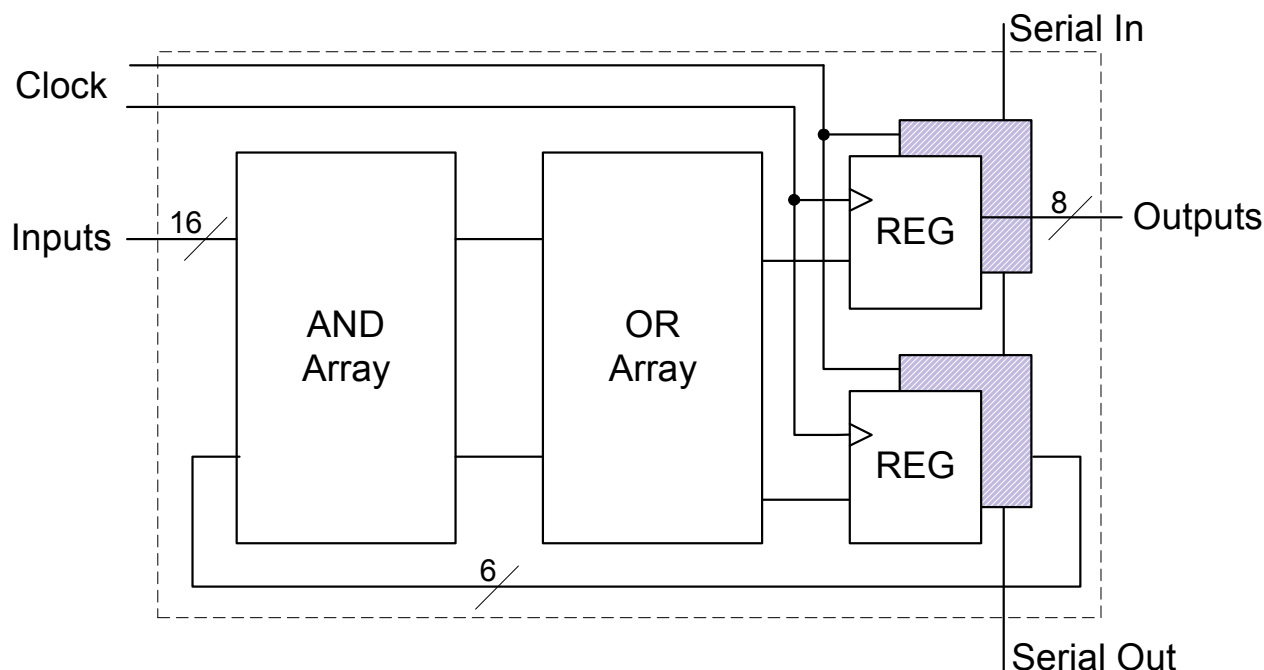
- Két kombinációs logikai hálózattól és egy regiszterből áll
- Tervezés során az állapot-átmeneteket vesszük figyelembe, **DE**
- Hibavalószínűség nagy,
- Szimulációs eszközök (Tools) hiánya,
- Hibák lehetségesek a prototípus fejlesztése során is,
- Könnyen konfigurálható / flexibilis eszközök szükségesek
→ mindezek használunk programozható alkatrészeket



Field Programmable Logic Sequencer (FPLS) – Logikai sorrendvezérlő

- Egy vezérlő egység Next-State logikai blokkjának megvalósítása a tervező, gyártó feladata. Általában változó logikát használnak a függvények megvalósításánál.
- A felhasználó által programozható logikai sorrendvezérlő (**Field Programmable Logic Sequencer**) programozható alkatrészekből építhető fel, amelyek a következőkben részletesen ismertetésre kerülnek.

Field Programmable Logic Sequencer (FPLS) – Logikai sorrendvezérlő

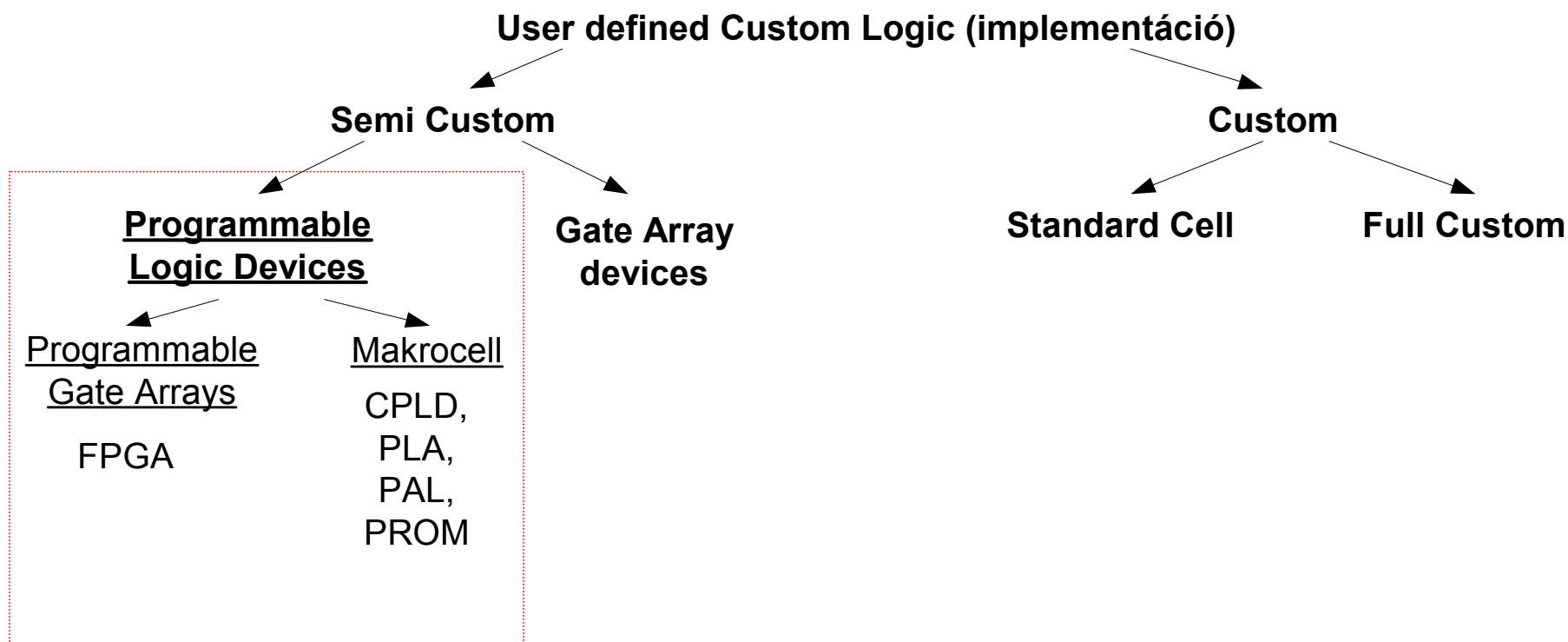


Működése:



- Normál (D-FF),
- Debug (Shift Reg.)

A logikai sorrendvezérlőnek 16 külső bemeneti vonala van, 1 RESET és 1 kimenet-engedélyező vonala, ill. 8 kimeneti vonala. A regiszterek egy-egy állapotot tárolnak, amelyek az órajel hatására a kimenetre íródnak, vagy a 6 belső, *visszacsatoló* vonalon keresztül visszacsatolódnak. A Next-State ill. a kimeneti szintek meghatározásánál programozható AND/OR tömböket használnak.

Felhasználó által definiált logikai implementációk:

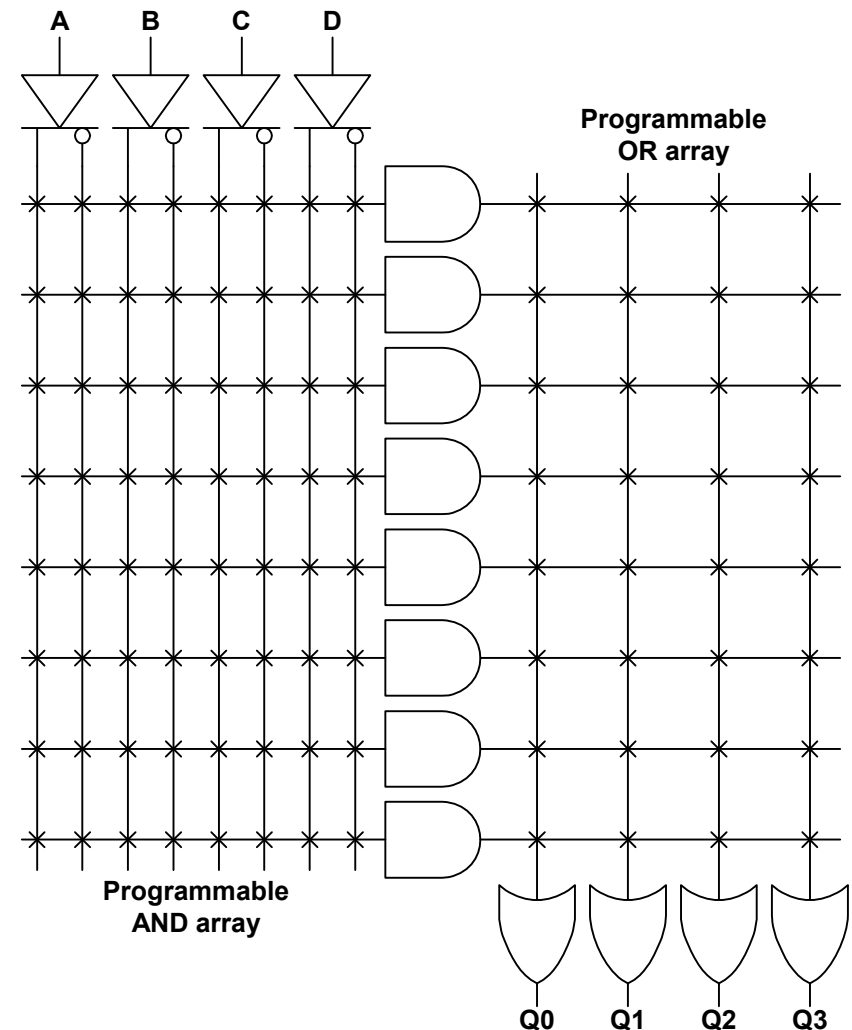


Programozható logikai eszközök (PLD-k) két fő típusa:

- 1.) Makrocellás PLD-k: (Programmable Logic Devices):
 - PLA
 - PAL
 - PROM
 - EPLD, CPLD
- 2.) FPGA (Field Programmable Gate Array):
Programozható Gate Array áramkörök
 - **XILINX** (Spartan, Virtex)  **XILINX®** > 50 %
 - **Altera,**  **ALTERA®** ~ 30 %
 - Actel (főleg úrkutatásban alkalmazott)
 - Lattice, Quicklogic, SiliconBlue, Achronix stb. sorozatok

Programmable Logic Array (PLA)

- Mindkét része (AND, OR) programozható
- Bármely kombinációja az AND / OR-nak előállítható
- Mintermek OR kapcsolata (DNF)
- Programozható kapcsolók a horizontális/ vertikális vonalak metszésében
- Q_n Kimenetek D tárolók! (visszacsat. a bemenetekre)



Programozásuk (pl. Fuse – biztosítékok) segítségével

- Az összeköttetés mátrix metszéspontjaiban akár kis biztosítékok (fuse) helyezkednek el. Gyárilag logikai '1'-est definiál, tehát vezetőképes. Ha valamilyen spec. programozó eszközzel, a küszöbnél nagyobb feszültséget kapcsolunk rá, átégethető, tehát szigetelővé (nem-vezető) válik, és logikai '0'-át fog reprezentálni.
- A biztosíték átégetése, csak egyszer lehetséges, utána már csak a programozott állapotot fogja tárolni (**OTP** – One time programmable IC).

Példa: PLA tervezése

- Realizálja a következő függvényeket:

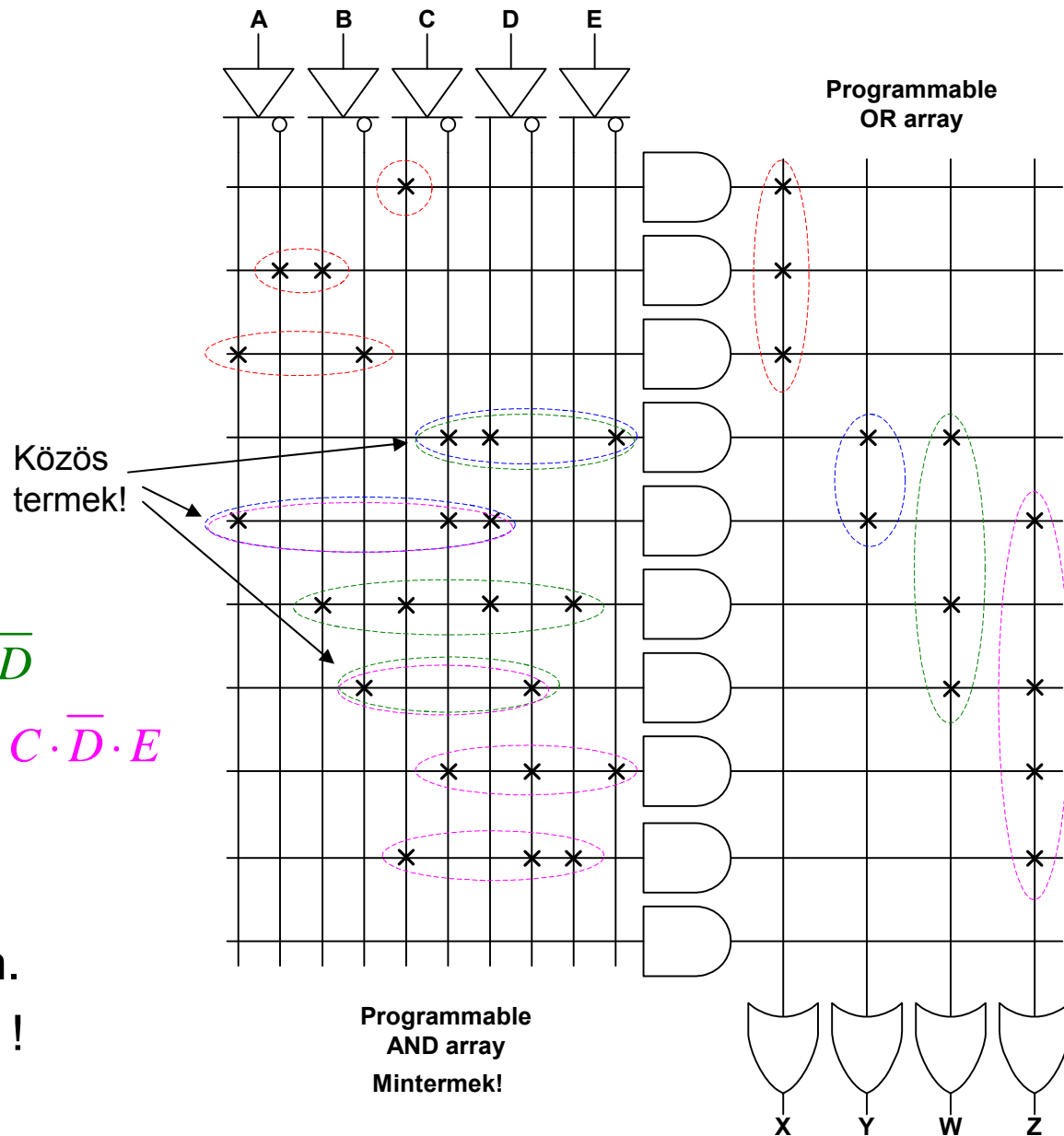
$$X = C + \bar{A} \cdot B + A \cdot \bar{B}$$

$$Y = \bar{C} \cdot D \cdot \bar{E} + A \cdot \bar{C} \cdot D$$

$$W = \bar{C} \cdot D \cdot \bar{E} + B \cdot C \cdot D \cdot E + \bar{B} \cdot \bar{D}$$

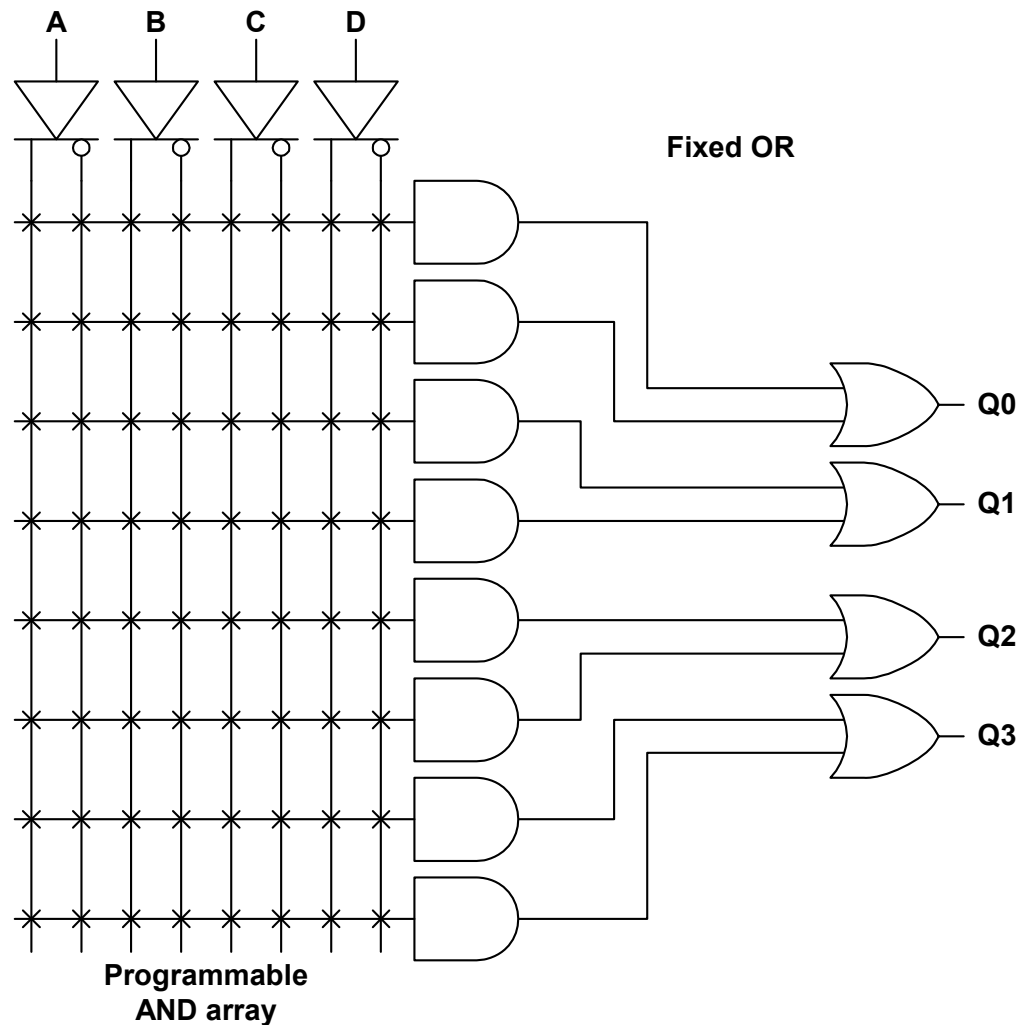
$$Z = A \cdot \bar{C} \cdot D + \bar{B} \cdot \bar{D} + \bar{C} \cdot \bar{D} \cdot \bar{E} + C \cdot \bar{D} \cdot E$$

- Tehát 5 bemenete (A,B,C,D,E) és
- 4 kimenete (X,Y,W,Z) van.
- Rajzoljuk fel a kapcsolást !



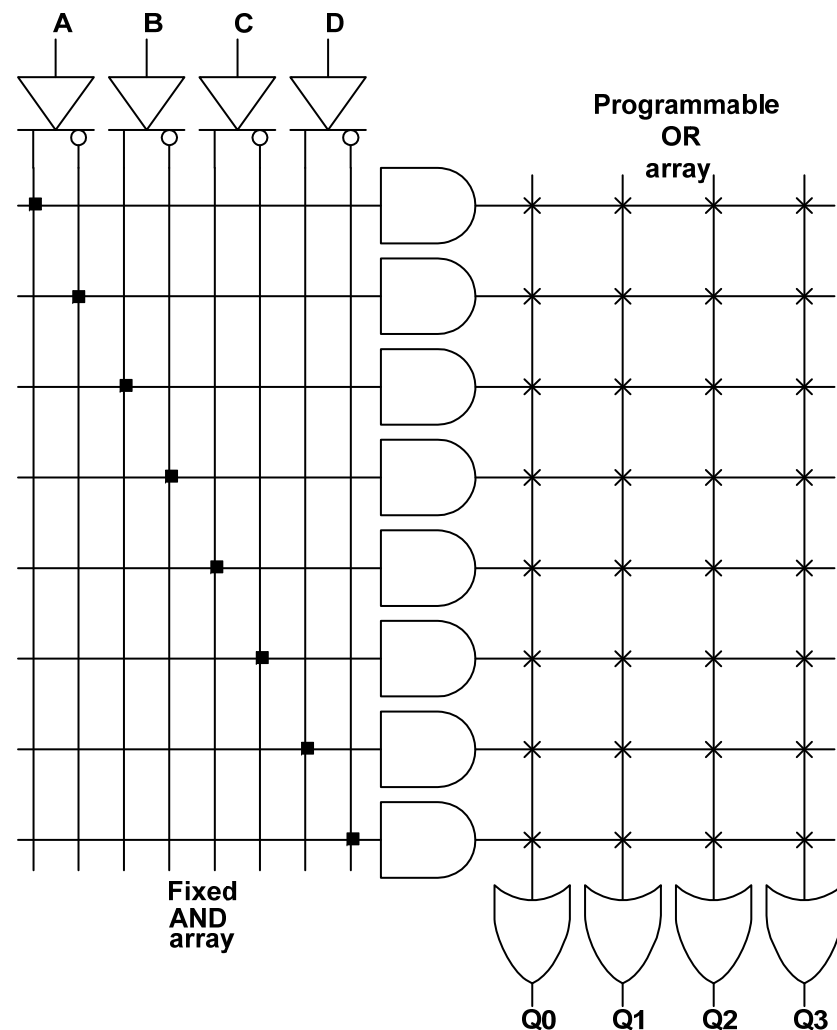
Programmable AND Logic (PAL)

- Egy programozható rész - AND / míg az OR fix
- Véges kombinációja áll elő az AND / OR kapcsolatoknak
- Metszéspontokban kevesebb kapcsoló szükséges
- Gyorsabb, mint a PLA
- Q_n kimeneteken D tárolók (visszacsatolódhatnak a bemenetekre)



Programmable Read Only Memory (PROM)

- Egy programozható rész - OR / míg az AND fix
- Véges kombinációja áll elő az AND / OR kapcsolatoknak
- Metszéspontokban kevesebb kapcsoló szükséges
- Gyorsabb, mint a PLA
- Q_n kimeneteken D tárolók! (visszacsatolódhatnak a bemenetekre)

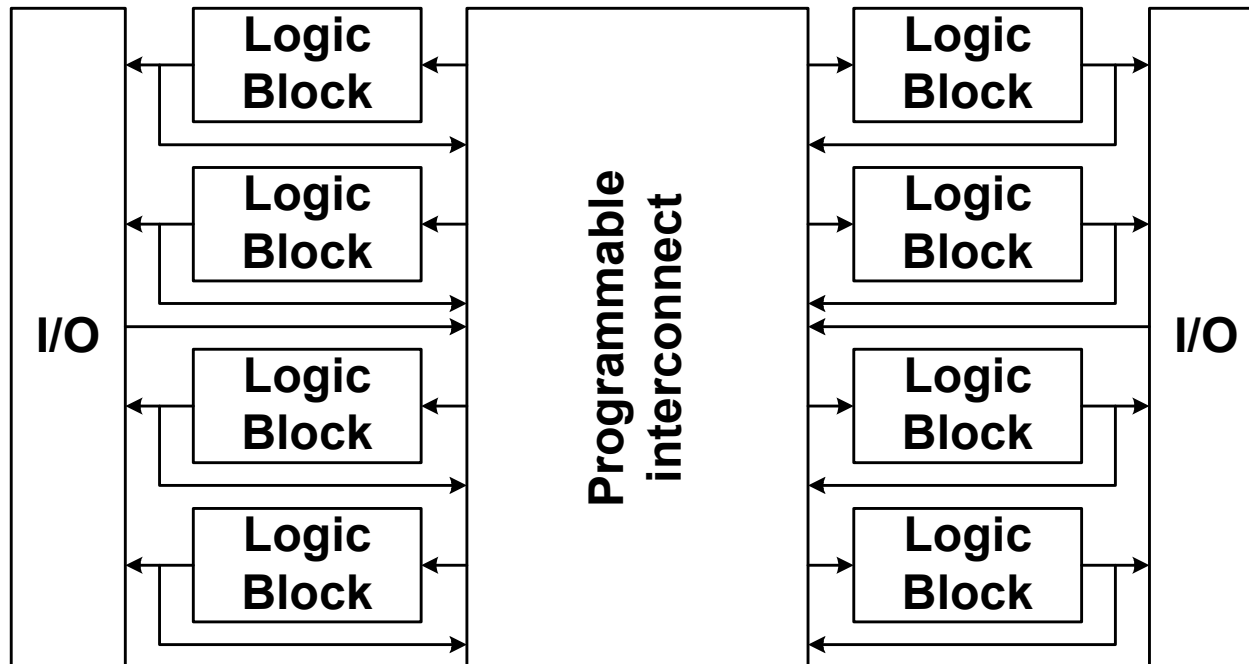


x: programozható

■: fix

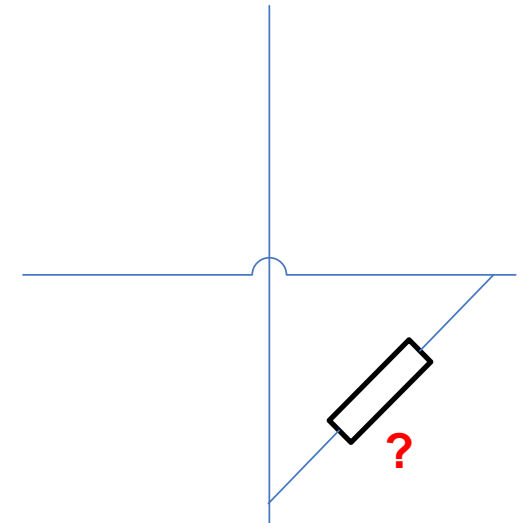
Complex Programmable Logic Device (CPLD)

- Logikai Blokk-ban
 - ~ PAL / PLA
 - Regiszterek (D-FF)
- I/O Blokkok
- Programozható összeköttetések (Interconnect)
 - Teljes (Full-crossbar), vagy
 - Részleges összeköttetés hálózat



Programozási technikák (összeköttetésekben)

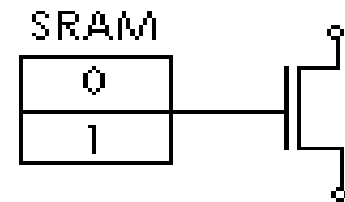
- a.) SRAM
- b.) MUX
- c.) Antifuse
- d.) Floating Gate
 - e.) EPROM/EEPROM/Flash



$$\tau = R \cdot C$$
$$[ns / ps]$$

a.) SRAM cellás

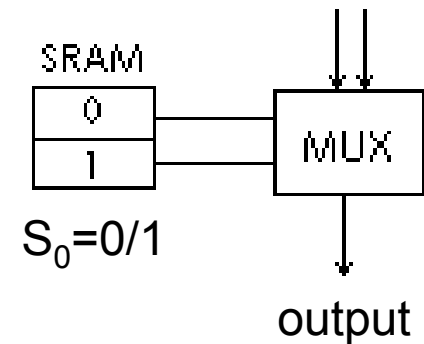
- Tulajdonságai: (pl Xilinx, Altera, Lattice etc.)
 - végtelen sokszor újraprogramozható (statikus RAM)
 - táp kikapcsolása után az SRAM elveszti tartalmát
 - bekapcsoláskor (inicializáláskor) a programot be kell tölteni, fel kell programozni
 - az SRAM cellára egy áteresztő tranzisztor van csatolva. A tranzisztor vagy kinyit (vezet), vagy lezár. Az SRAM értéke, ami egy bitet tárol ('0' vagy '1') letölthető. Összeköttetéseket, vagy MUX-ok állását is eltárolja.
 - 1 bit tárolása az SRAM-ban (min. 6 tranzisztorból áll)
 - sok tranzisztor (standard CMOS), nagy méret, nagy disszipáció
 - nem kell frissíteni az SRAM-ot
 - nagy 0.5-2 k Ω átmeneti ellenállás
 - nagy 10-20 femtoF parazita kapacitás



b.) MUX - multiplexeres

- Tulajdonságai:

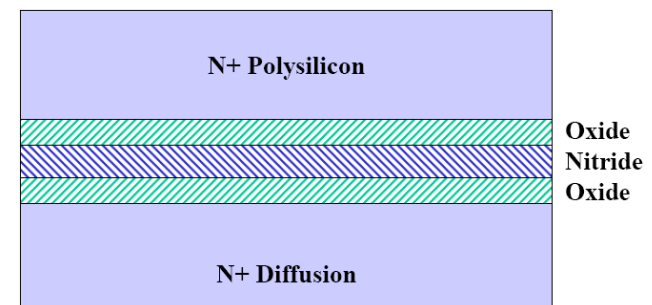
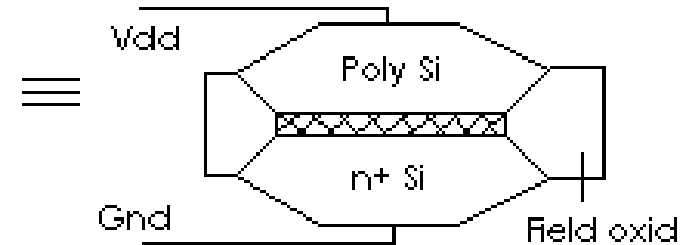
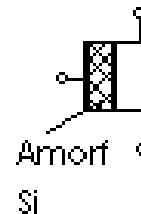
- az SRAM-ban tárolt '0' vagy '1' értéket használunk a Multiplexer bemeneti vonalának kiválasztásához. (Működése hasonló az SRAM celláéhoz.) /Bemenetek közül választ a szelektáló SRAM-beli érték segítségével és a kimenettel köti össze./



c.) Antifuse

A tranzisztor Gate-jét amorf kristályos Si alkotja, amelyet relatíve nagy feszültség (kb 20-30V) hatására átkristályosítunk (átolvasztás), így vezetővé válik véglegesen. Pl. Texas Instruments, Actel, QuickLogic alkalmazza ezt a technológiát. Tulajdonságai:

- A dielektrikum „átégetése” irreverzibilis folyamat, nem lehet újraprogramozni
- csak egyetlen egyszer programozható (OTP)
- kis méreten megvalósítható, kis disszipáció
- kis átmeneti ellenállás 300 Ω
- kis parazita kapacitás 1.1-1.3 femtoF
- előállításához sok maszkréteg szükséges, drága technológiát igényel
- Típusai
 - ☐ ONO (Oxid-Nitrid-Oxid)
 - ☐ Amorf Si

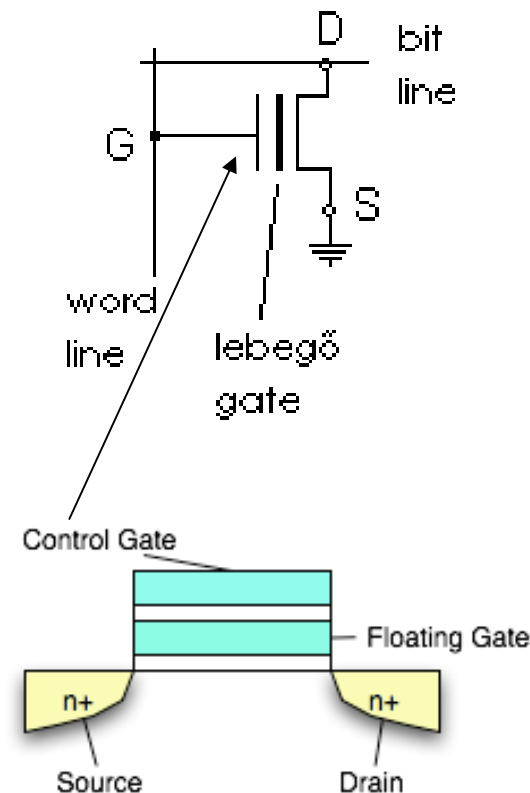


d.) Floating gate

Két-gates tranzisztor, melynek középső gate-je a lebegő gate, tárolja az információt. A másik gate fix (control vagy érzékelő gate), biztosítja az írást, olvasást. Programozható összeköttetéseknel, csomópontokban használatos.

■ Tulajdonságai:

- Programozása/írás: control gate segítségével töltéseket viszünk fel a lebegő Gate-re, kinyit a tranzisztor
- többször törölhető (kis ablakon keresztül UV fénnel)
- kikapcsoláskor is megőrzi tartalmát (non-volatile, akár 99 évig), töltések nem sülnek ki
- nagy 2-4 k Ω átmeneti ellenállás
- nagy 10-20 femtoF parazita kapacitás
- PL. Intel, Actel, Lattice



e.) EPROM / EEPROM / Flash

■ Tulajdonságai:

- ☐ „**Floating-gate**” technológiát alkalmazza!
- ☐ 10000x szer programozható
- ☐ Megőrzi tartalmát (Non-volatile)
- ☐ UV-fénnyel törölhető (EPROM)
- ☐ Elektromosan törölhető (EEPROM / Flash)
- ☐ Nagy felület
- ☐ Nagy átmeneti ellenállás, nagy parazita kapacitás
- ☐ További CMOS gyártási lépések (sok maszk réteg) szükségesek – drága
- ☐ PI: Altera (3000, 5000 - első sorozatai)