PANNON EGYETEM, Veszprém
Villamosmérnöki és Információs Rendszerek Tanszék

Digitális Rendszerek és Számítógép Architektúrák
(Levelező BSc)

4. előadás: Szekvenciális Hálózatok:
Vezérlő egységek, Programozható alkatrészek

Előadó: Dr. Vörösházi Zsolt
voroshazi.zsolt@mik.uni-pannon.hu

Jegyzetek, segédanyagok:

- Könyvfejezetek:
  - http://www.virt.uni-pannon.hu → Oktatás → Tantárgyak → Digitális Rendszerek és Számítógép Architektúrák (BSc Levelező)
  - (chapter05.pdf)
- Fóliák, óravázlatok .ppt (.pdf)
- Feltöltésük folyamatosan

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 tervezésének 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.

Adatút (Data-path) tervezés szempontjai

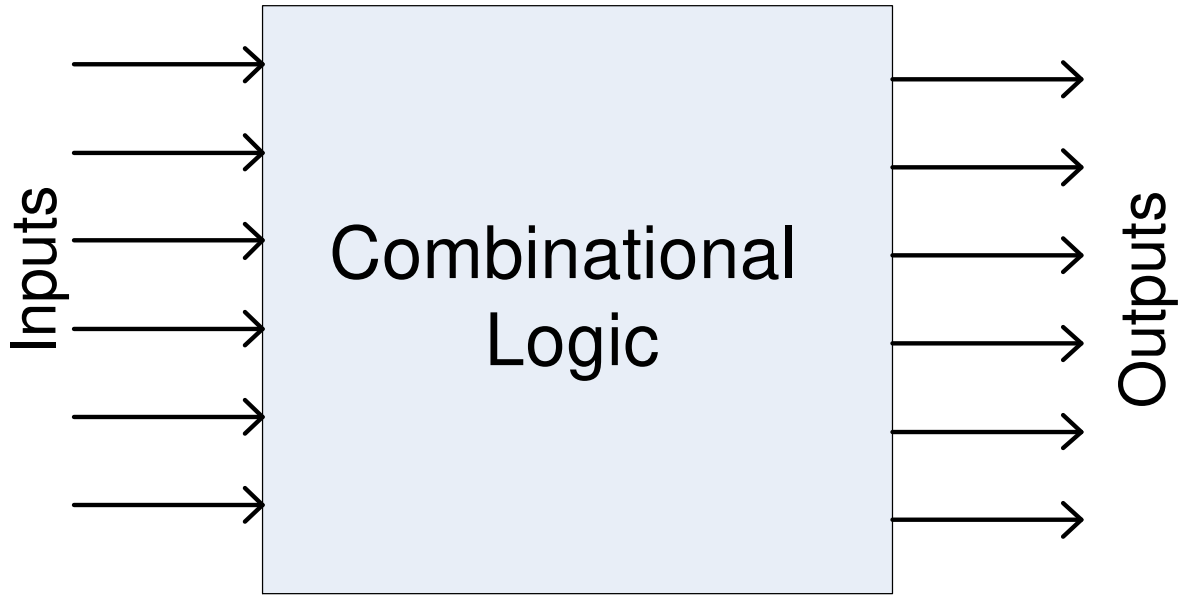
- Gazdaságosság (költség)
- Interfész szükséglet (protokollok)
- Sebesség (S)
- Felület (A)
- Energia (disszipált teljesítmény) (P, D)
- Dinamikai tartomány (számrendszerek)
- Rugalmasság (többcélúság)
- Kezelhetőség (probléma, hiba során)
- Környezet (pl. ipari v. irodai használat?)

Vezérlő egységek fajtái:

- I. **Huzalozott („klasszikus”)** módszerek (pl. korai RISC architektúrák):
  - Mealy-modell,
  - Moore-modell (Példa: FIR szűrő tervezése – FSM állapotgép segítségével).
    - Multiplexeres / késleltetéses / Shift-regiszteres megvalósítások
- II. **Mikroprogramozott** („**reguláris**” vezérlési szerkezettel – pl. CISC, ill. mai RISC architektúrák):
  - Horizontális mikrokódos vezérlő,
  - Vertikális mikrokódos vezérlő.
- III. **Programozható logikai eszközök (PLD):**
  - Maszk-programozható: PLA, PAL, PROM, CPLD.
  - Újrakonfigurálható (szoftveresen): FPGA (nem tárgyaljuk)

Ismétlés:Kombinációs hálózatok

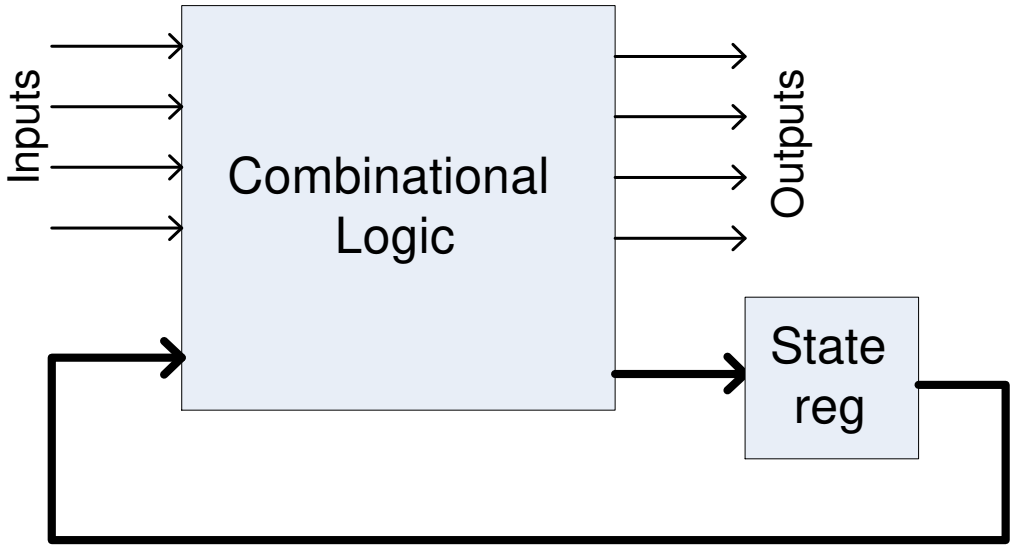
- **(K.H.) Kombinációs logikai hálózat**ról beszélünk: ha a mindenkori kimeneti kombinációk értéke csupán a bemeneti kombinációk pillanatnyi értékétől függ (tároló „kapacitás”, vagy memória nélküli hálózatok).



Sorrendi hálózatok:

- **(S.H.) Sorrendi (szekvenciális) logikai hálózat**ról beszélünk**:** ha a mindenkori kimeneti kombinációt, nemcsak a pillanatnyi bemeneti kombinációk, hanem a korábban fennállt bementi 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 szolgáltassanak, attól függően, hogy a bemeneti kombináció fellépésekor, milyen értékű a szekunder kombináció, pl. a State Register tartalma)

Vezérlő egységek alapjául szolgáló hálózat!



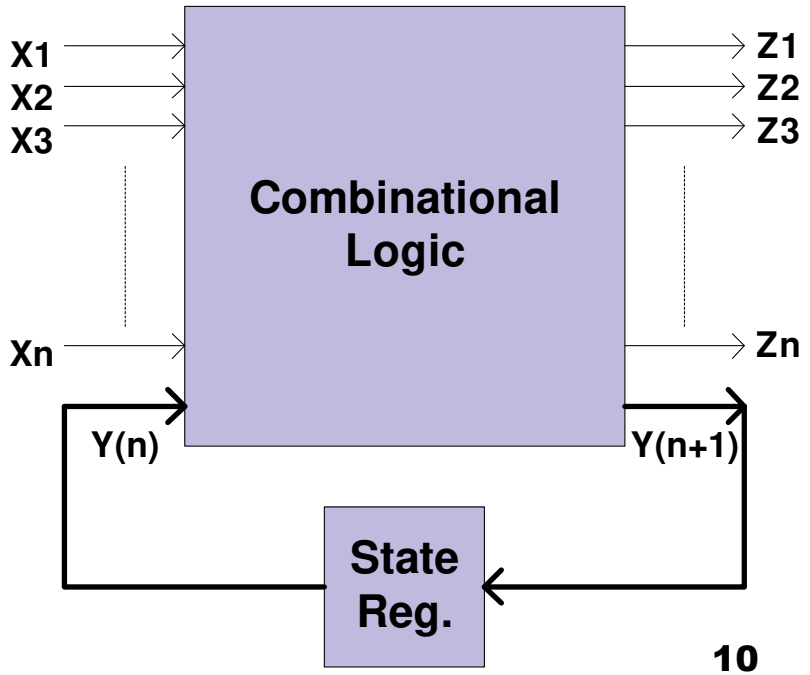
Időzítő – vezérlő egység:

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

I. Klasszikus vezérlési módszerek: Huzalozott vezérlő egységek

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 értékétől, hanem a korábbi állapotoktól is függenek. Problémák merülhetnek fel az állapotok és bemenetek közötti szinkronizáció hiánya miatt (változó hosszúságú kimenetet - dekódolás). Ezért alkalmazzuk legtöbb esetben 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:
  - $\delta(Xn,Yn) \rightarrow Yn+1$: következő állapot fgv.
  - $\mu(Xn,Yn) \rightarrow Zn$: kimeneti fgv.

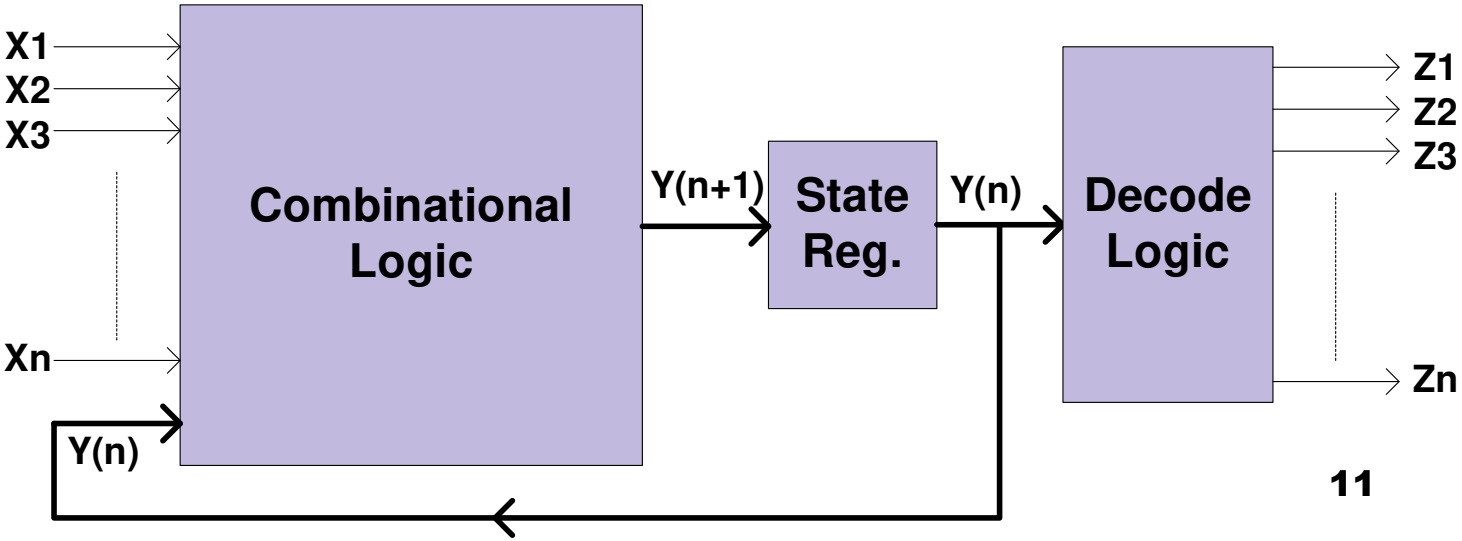


2.) Moore-modell

- A kimenetek közvetlenül csak a pillanatnyi állapottól függenek (bemenettől függetlenek v. közvetve függenek). Tehát a kimenetet nem a bemenetekhez, hanem az állapotoknak megfelelően szinkronizáljuk.
- Három halmaza van:
  - X – a bemenetek,
  - Z – a kimenetek,
  - Y – az állapotok halmaza.

Input ⇒ Next-State ⇒ Present-State ⇒ Output

- Két leképezési szabályok
  - $\delta(Xn,Yn) \rightarrow Yn+1$: köv. állapot fgv.
  - $\mu(Yn) \rightarrow Zn$: kimeneti fgv.



II. 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 vs. **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). Vezérlő egység:
  - Kombinációs hálózat (hard-wired = huzalozott), vagy
  - FSM: véges állapotú automata alapú.
- Wilkes (1951): A komplex, többcímű (operandusú), illetve vezérlési szerkezeteket **„reguláris módszerrel”** lehet gyorsítani, egyszerűsíteni: nevezetesen **gyors memória elemeket** kell használni az utasítássorozatok tárolásánál. Ugyan a klasszikus módszernél használt állapotgépekkel (FSM) modellezik a reguláris vezérlő egység működését, majd ezt a modellt transzformálják át mikrokódos memóriát (ami nem azonos az operatív memóriával!) használva. Az adatútvonal vezérlési pontjait memóriából (ROM) kiolvasott *vertikális-* vagy *horizontális-mikrokódú utasításokkal állítják be*!

Reguláris módszer: mikrokódos vezérlés tulajdonságai

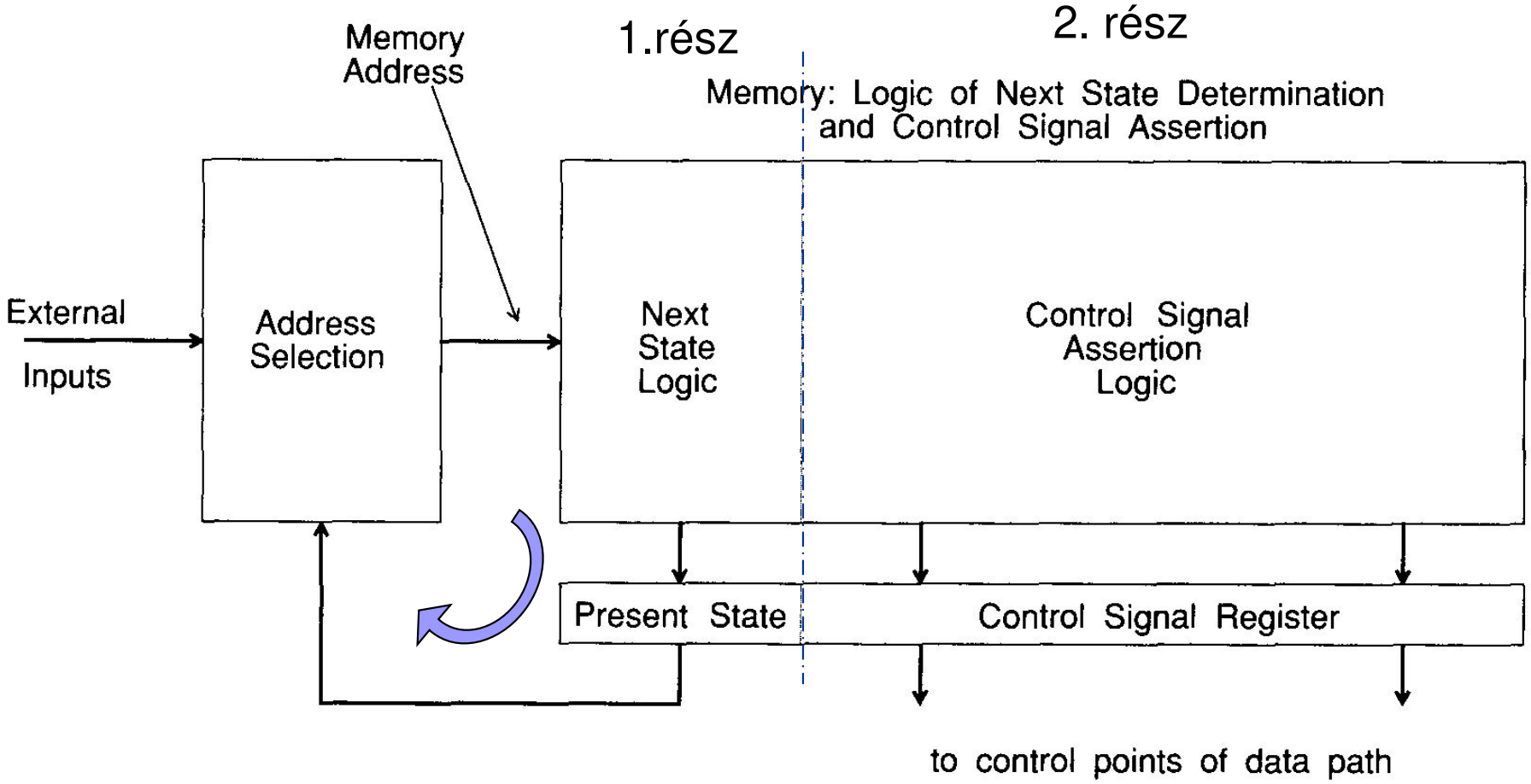
- Mikrokód: *gépi kódú utasításokat* (IR) $\xrightarrow{\text{leképezés}}$ *legalacsonyabb szintű áramköri (HW) utasítások* sorozatára leképező <u>köztes kód</u>
- Szerepe: **értelmezés** (interpreter / translator) a fenti két szint között:
  - A gépi kódú utasítások változtatásának lehetősége (RISC, 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)
- Alkalmazás: CPU, GPU, lemezvezérlők, NPU (network processor unit)

Mikroprogram = mikroutasítások sorozata

- Példa: Tipikus mikroprogram (~RTL-szintű utasítás szekvenciák sorozata):

1. Load Reg[1] to the "A" side of the ALU
2. Load Reg[7] to the "B" side of the ALU
3. Select the ALU to perform 2's-comp addition
4. Select the ALU's carry input to ’1’
5. Store the result value in Reg[8]
6. Update the "condition codes" with the ALU status flags ("Negative", "Zero", "Overflow", and "Carry")
7. Microjump (MicroPC) for the next microinstruction

FSM megvalósítása Memóriával

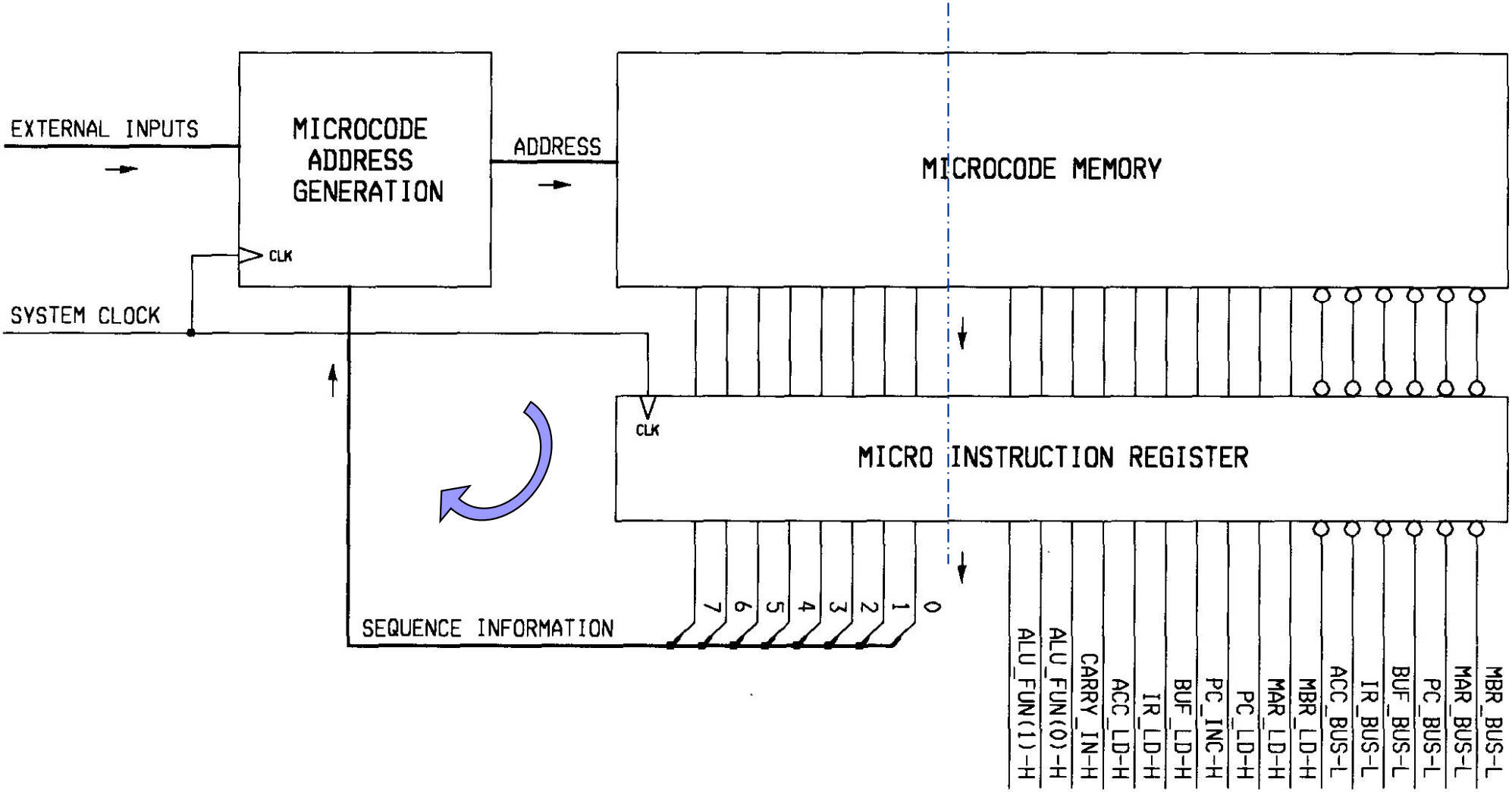


- 1.rész: szabályozza az eszköz működését a megfelelő állapotok *sorrendjében*
- 2.rész: szabályozza az adatfolyamot a megfelelő *vezérlőjelek beállításával* (assertion) az adatúton (vezérlési pontokon)

FSM megvalósítása Memóriával (folyt)

- ***Address Selection***: (mint új elem) a következő utasítás (*Next State*), és beállítani kívánt vezérlőjel (*control signal assertion*) <u>címére</u> mutat a memóriában (~ ld. MAR).
- A memória címet (***memory address***-t) külső bemenő jelek és a *present state* együttesen határozzák meg. E cím segítségével megkapjuk az adott vezérlő információ pontos helyét a memóriában, ill. ez az információ, mint új állapot betöltődik a generált vezérlőjeleket tároló (*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



Vezérlő jelek

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.

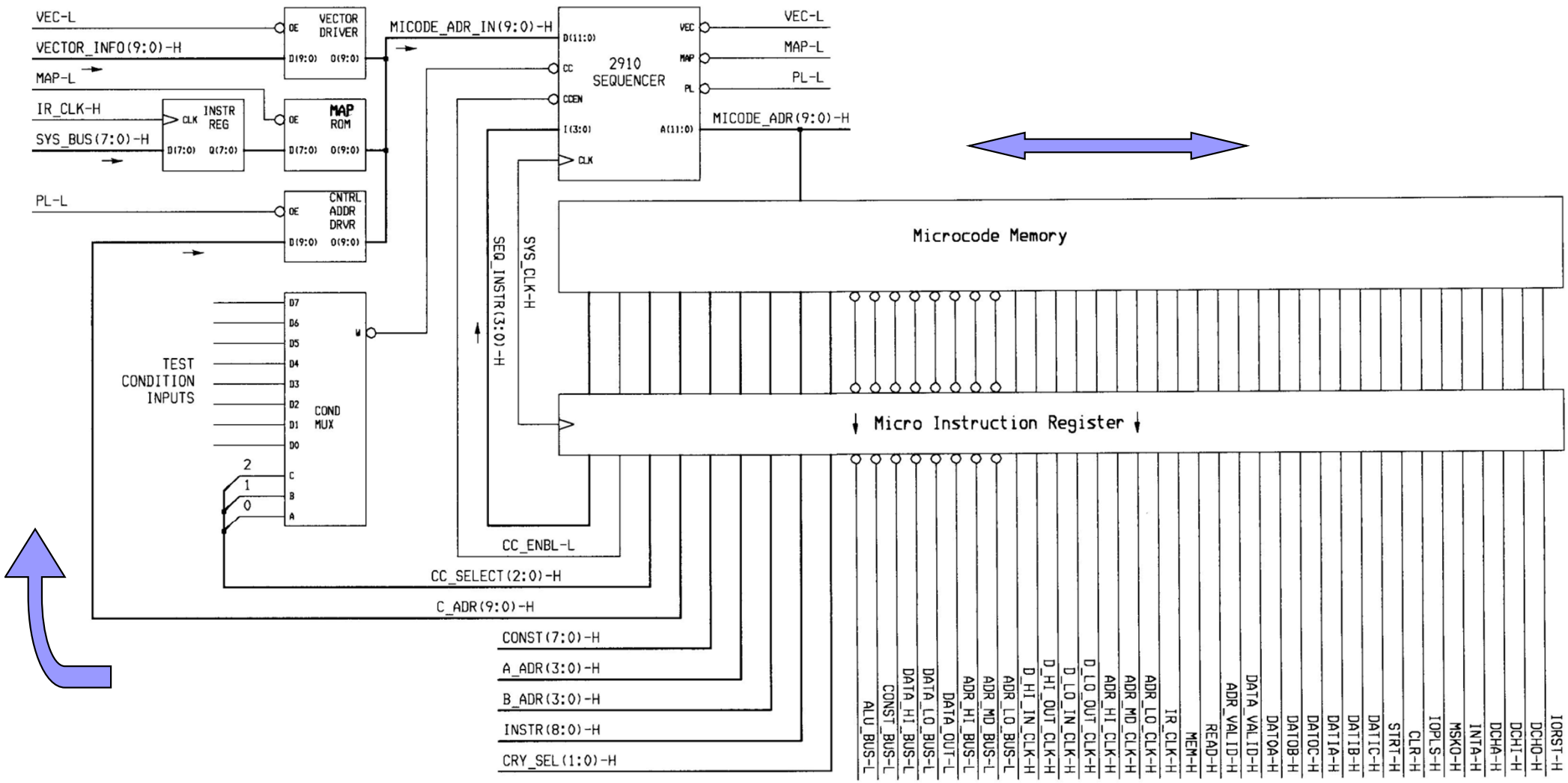
Általános Mikrokódos vezérlők tulajdonságai

- 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 (~ns) is számolni kell (nem csak a visszacsatolt aktuális állapot késleltetésével.)

1.) Horizontális mikrokódos vezérlő

- Mindenegyes 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 funkcionális egységeket (memória, ALU, regiszterek stb.) egyszerre 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**.

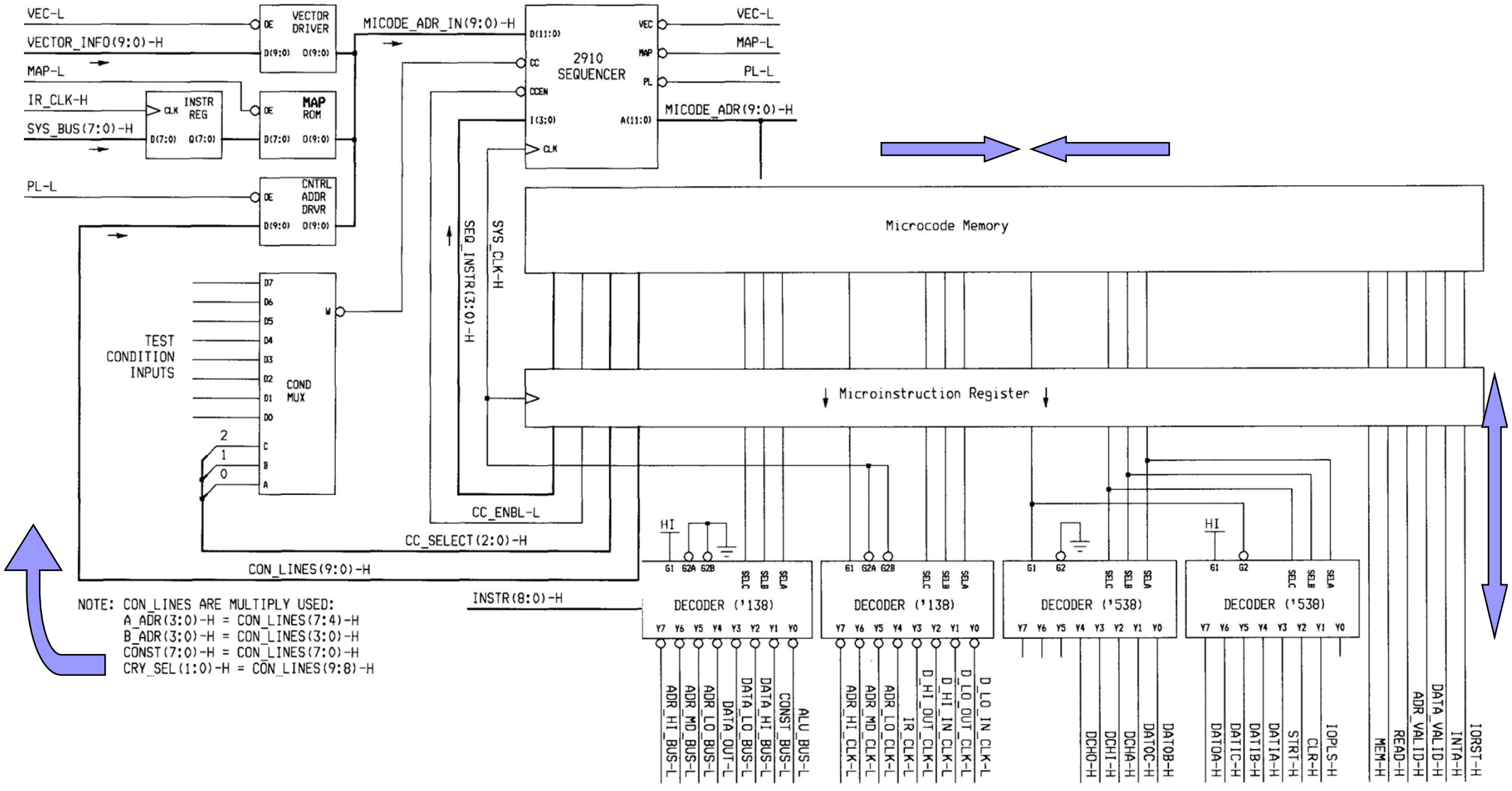
Horizontális mikrokódos vezérlő



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 -> N bites kimeneti busz. Több mikroutasítás szükségeltetik $\Rightarrow$ így a mikrokódú memóriát „**vertikálisan**” meg kell növeli.

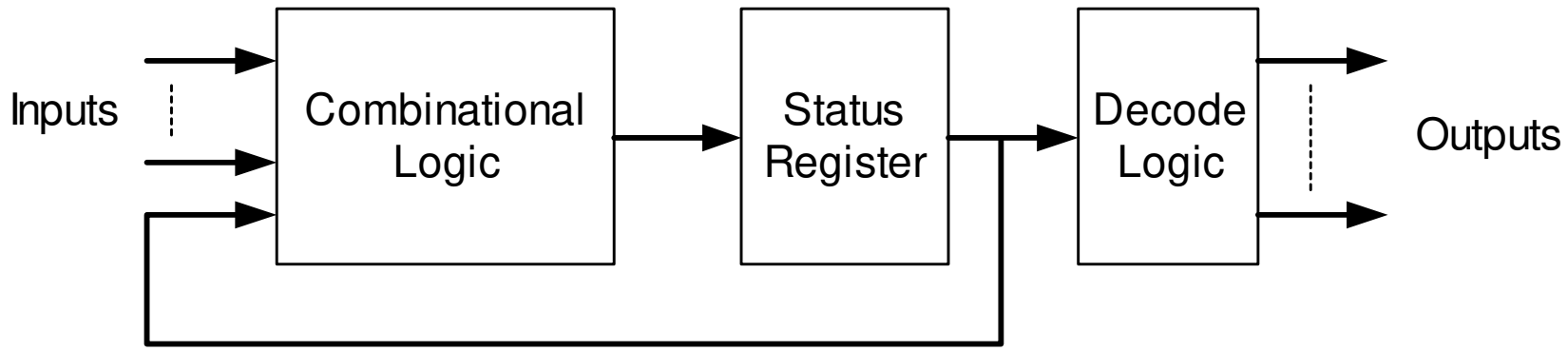
Vertikális mikrokódos vezérlő



III. Programozható logikai eszközök (PLD)

Állapotgép FSM tervezés tulajdonságai

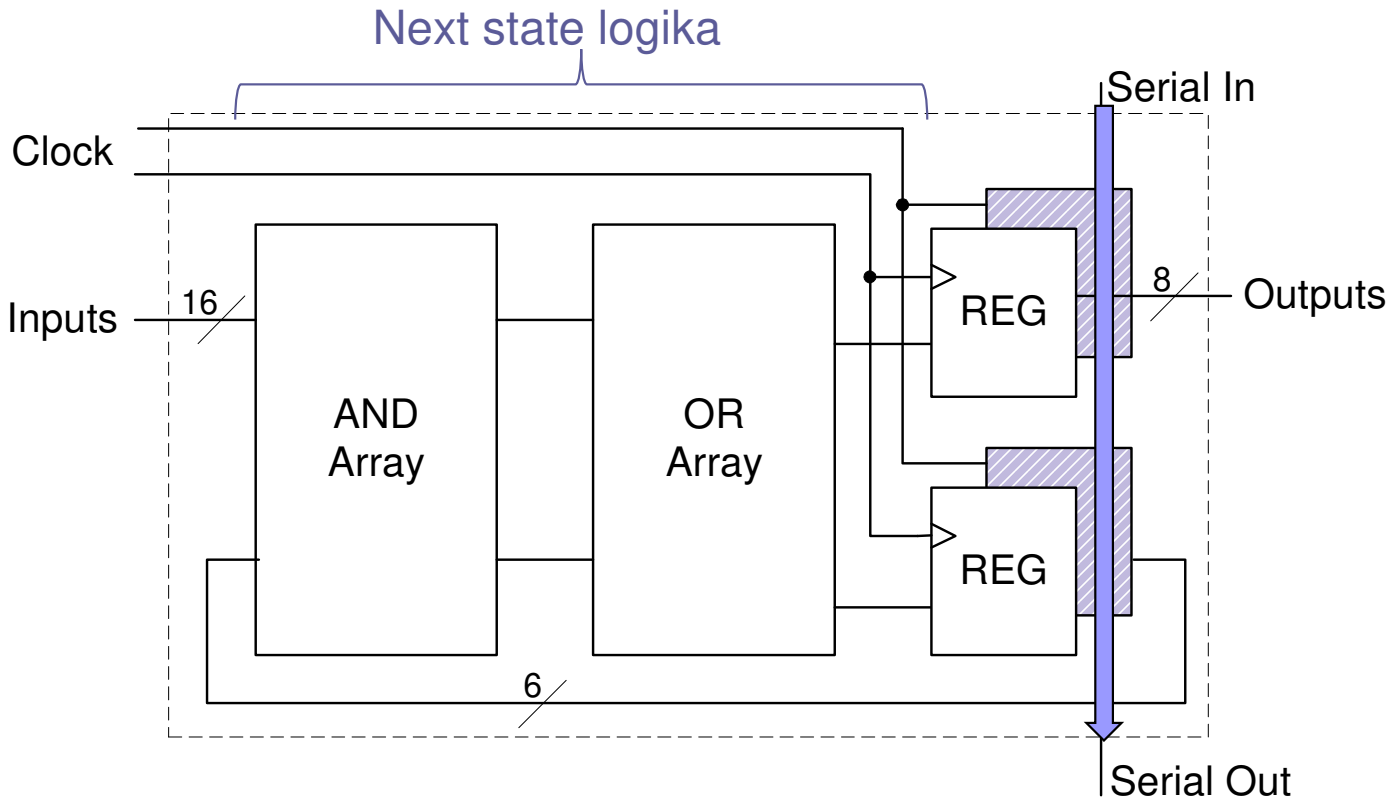
- Két kombinációs logikai hálózatból és egy regiszterből áll
- Tervezés során az állapot-átmeneteket vesszük figyelembe, **DE**
- Hibavalószínűség nagy,
- Szimulációs eszközök (CAD Tools) hiánya,
- Hibák lehetségesek a prototípus fejlesztése során is,
- Könnyen konfigurálható / flexibilis eszközök szükségesek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

Tervezési módszerek

Digitális l□gikai áramkörök tervezési módszerei

- **Részben egyedi**
  - **Pr□gram□zható l□gikai eszközök**
    - Pr□gram□zható kap□-áramkörök
      - FPGA
    - Makr□cellás
      - PLA,
      - PAL,
      - GAL,
      - CPLD,
  - **Kapu áramkörök (Gate Arrays)**
- **Egyedi**
  - **Szabvány□s cellák (Standard Cell)**
  - **Teljesen egyedi (Full Cust□m)**

ASIC

1.) Makrocellás típusok (kis/közepes integráltsági fok): a programozható logikai tömbök hálózatát egyrészt programozható logikai részekre (ÉS / VAGY tömbök), illetve kimeneti logikai cellákra, úgynevezett „makrocellákra" osztották. A *makrocellák* általában néhány logikai áramkört (inverter, multiplexert) tartalmaznak, valamint egy elemi D-tárolót. Ezáltal ha szükséges, regiszterelni lehet a kimeneti függvényt, majd pedig vissza lehet csatolni annak értékét a bemenetre.

a.) PLA: Programozható logikai tömb, OTP - általában biztosítékot használ

b.) PAL: Programozható ÉS/tömb logika, OTP – általában biztosítékot használ

c.) GAL: Generikus tömb logika: PAL továbbfejlesztése, de már többször törölhető és programozható (nagy integráltsági fok):

d.) CPLD: Komplex programozható logikai eszköz

2.) Programozható kapu-áramkörök (nagyon-nagy integráltsági fok):

FPGA: Felhasználó által tetszőlegesen programozható (újrakonfigurálható) kapu áramkör

* Fontos: Ezek a részek nem találhatóak meg a könyvben!

PLD – Programozható logikai eszközök

- A Programozható logikai áramköröket (PLD: Programmable Logic Devices) általánosan a kombinációs logikai hálózatok és sorrendi hálózatok tervezésére használhatjuk. Napjainkban ezek *VLSI* (Very-Large Scaling Integrated IC) típusú alkatrészek:
  - Több millió logikai kaput, vagy tranzisztort jelenthet
- Azonban míg a hagyományos kombinációs logikai hálózatok dedikált összeköttetésekkel, illetve kötött funkcióval (kimeneti függvény) rendelkeznek, addig a programozható logikai eszközökben pontosan ezek változtathatók, az alábbi lehetséges módokon:
  - A felhasználó által *egyszer programozható* / konfigurálható logikai eszközök (**OTP**: One Time Programmable), amelynél a gyártás során nem definiált funkció egyszer még megváltoztatható (ilyenek pl. a korai PAL, PLA eszközök)
  - *Többször*, akár *tetszőleges módon* programozható logikai eszközök = *újrakonfigurálható* (ilyenek pl. a korábbi GAL, vagy a mai modern CPLD, **FPGA** eszközök)

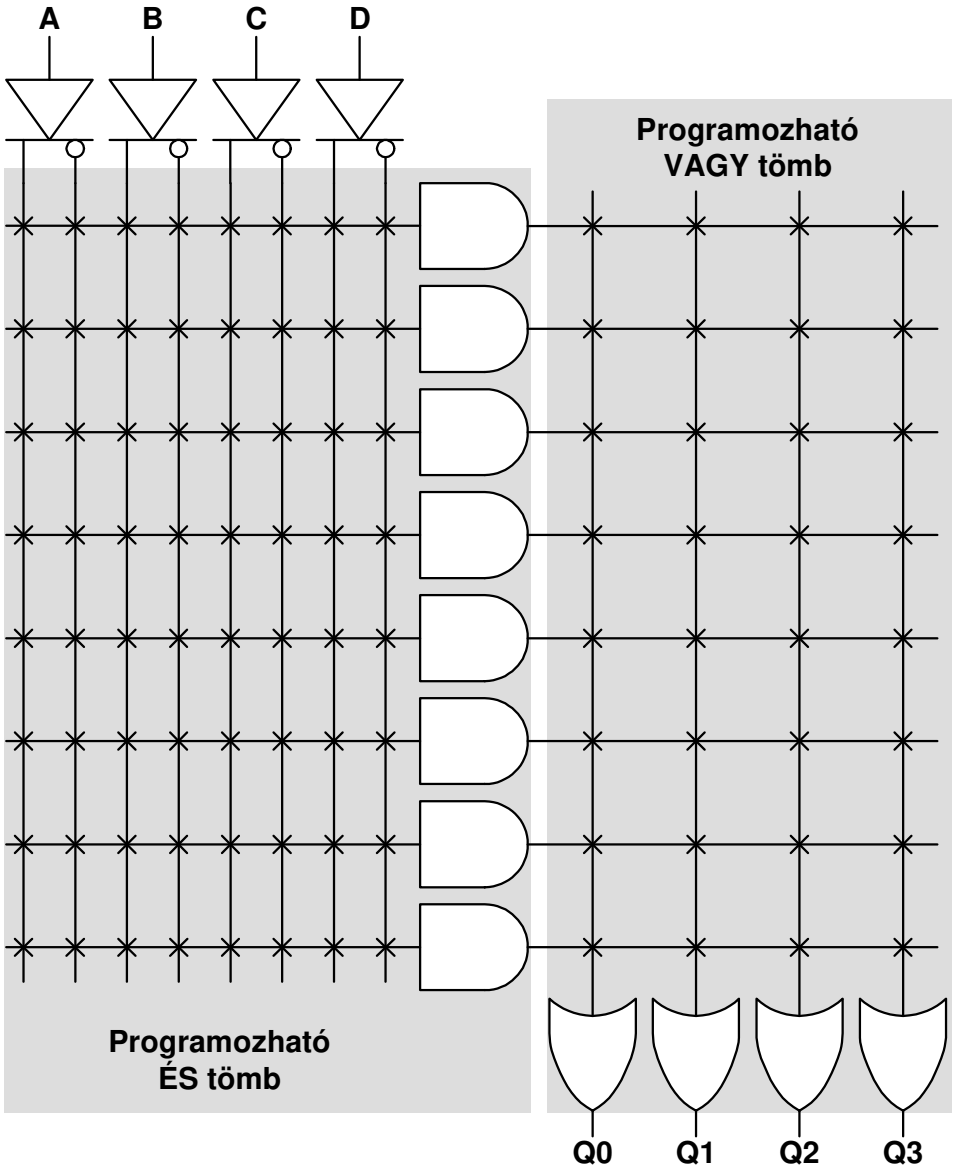
Programozható logikai eszközök (PLD-k) két fő típusa:

- 1.) Makrocellás PLD-k (Programmable Logic Devices):
  - PLA
  - PAL
  - GAL
  - CPLD
- 2.) FPGA (Field Programmable Gate Array): Programozható Gate Array áramkörök
  - **AMD-XILINX** (Spartan, Virtex, Kintex, Artix) ~ 52 % !
  - **Intel-Altra FPGA** (Stratix, Arria, Cyclone), ~ 33 %
  - Lattice, 8%
  - Microchip, ~5%
  - Reneseas, < 1%
  - További kisebb gyártók termékei <3%

📖 http://www.eetimes.com/author.asp?section_id=36&doc_id=1331443&_mc=RSS_EET_EDT

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 Kimeneteken D tárolók! (visszacsat. a bemenetekre)



PLA

- 1970-ben, a TI (Texas Instruments) által kifejlesztett eszköz mindkét részhálózata (ÉS, illetve VAGY tömb) programozható összeköttetéseket tartalmazott, amelyek segítségével tetszőleges mintermek tetszőleges VAGY kapcsolata előállítható (DNF alakot), ezáltal bármilyen kombinációs logikai hálózat realizálható volt (természetesen adott bemenet, ill. kimenet szám mellett).
- A programozható ÉS / VAGY tömbökben úgynevezett „programozható kapcsolók” vannak elhelyezve a horizontális/ vertikális vonalak metszéspontjában.
- Amennyiben a Qn kimenet(ek)re tárolókat kötünk (pl. egyszerű D tárolót), majd pedig visszacsatoljuk a programozható logikai hálózat bemenete(i)re akár egy sorrendi hálózati viselkedést is meghatározhatunk.

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:

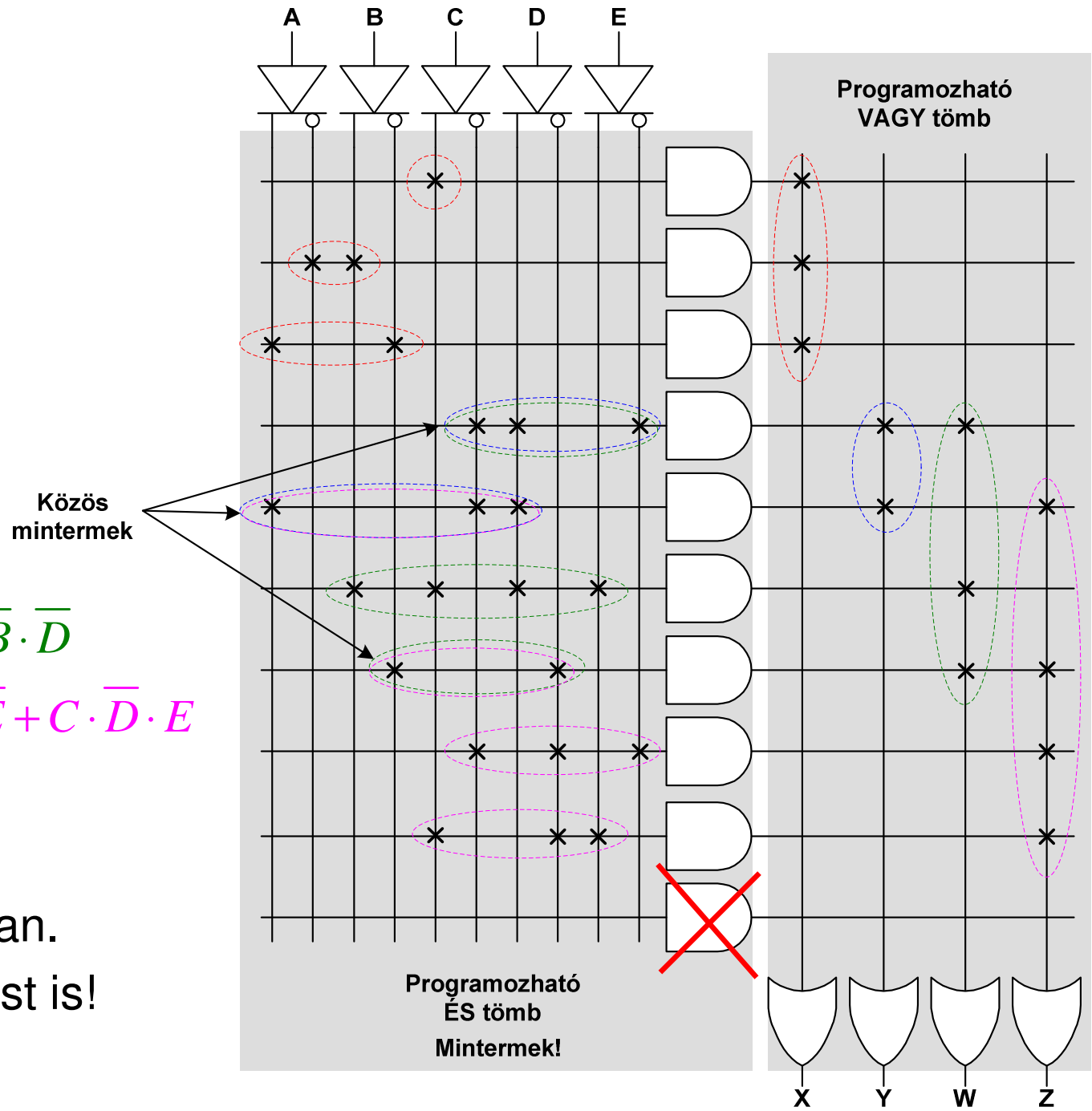
$$X = C + \overline{A} \cdot B + A \cdot \overline{B}$$

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

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

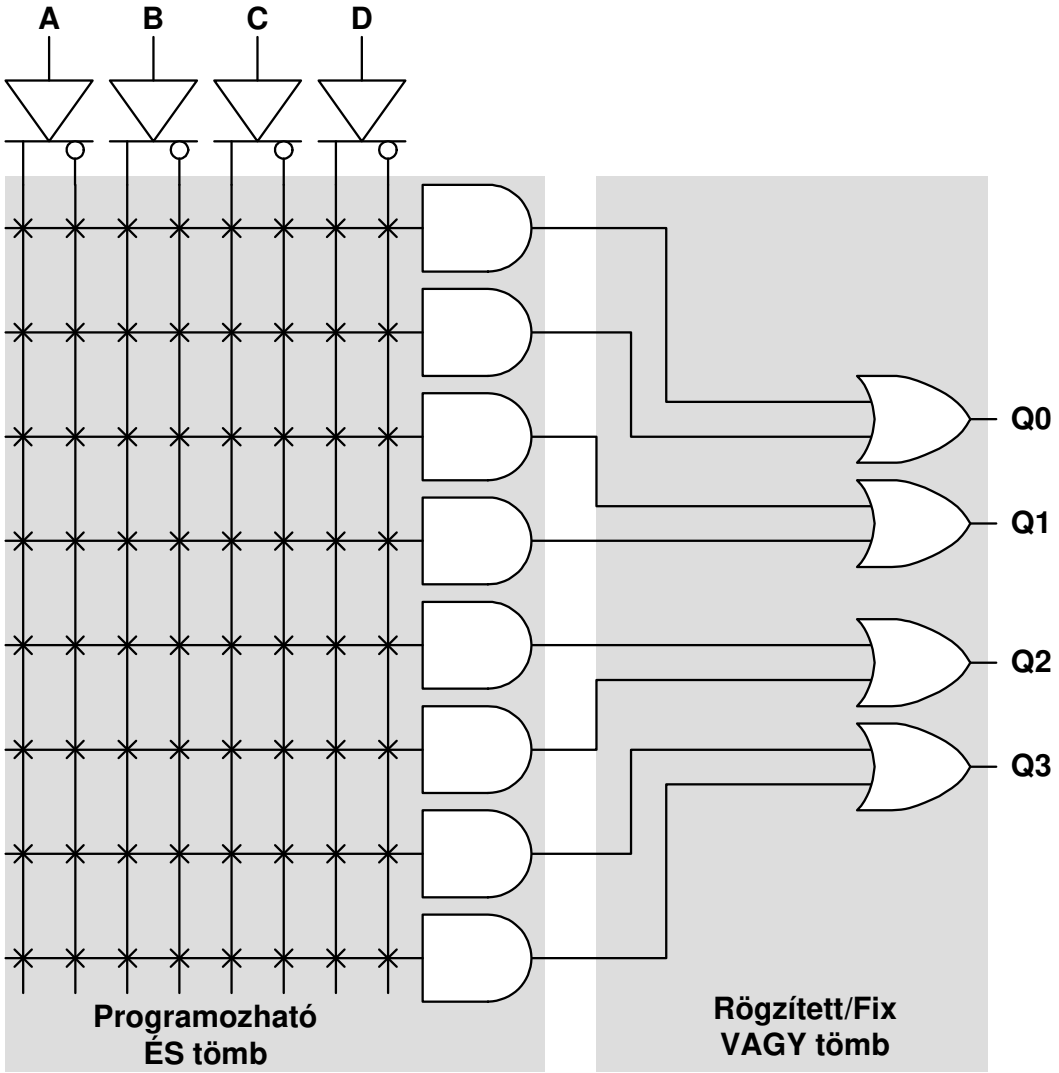
$$Z = A \cdot \overline{C} \cdot D + \overline{B} \cdot \overline{D} + \overline{C} \cdot \overline{D} \cdot \overline{E} + C \cdot \overline{D} \cdot E$$

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



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)

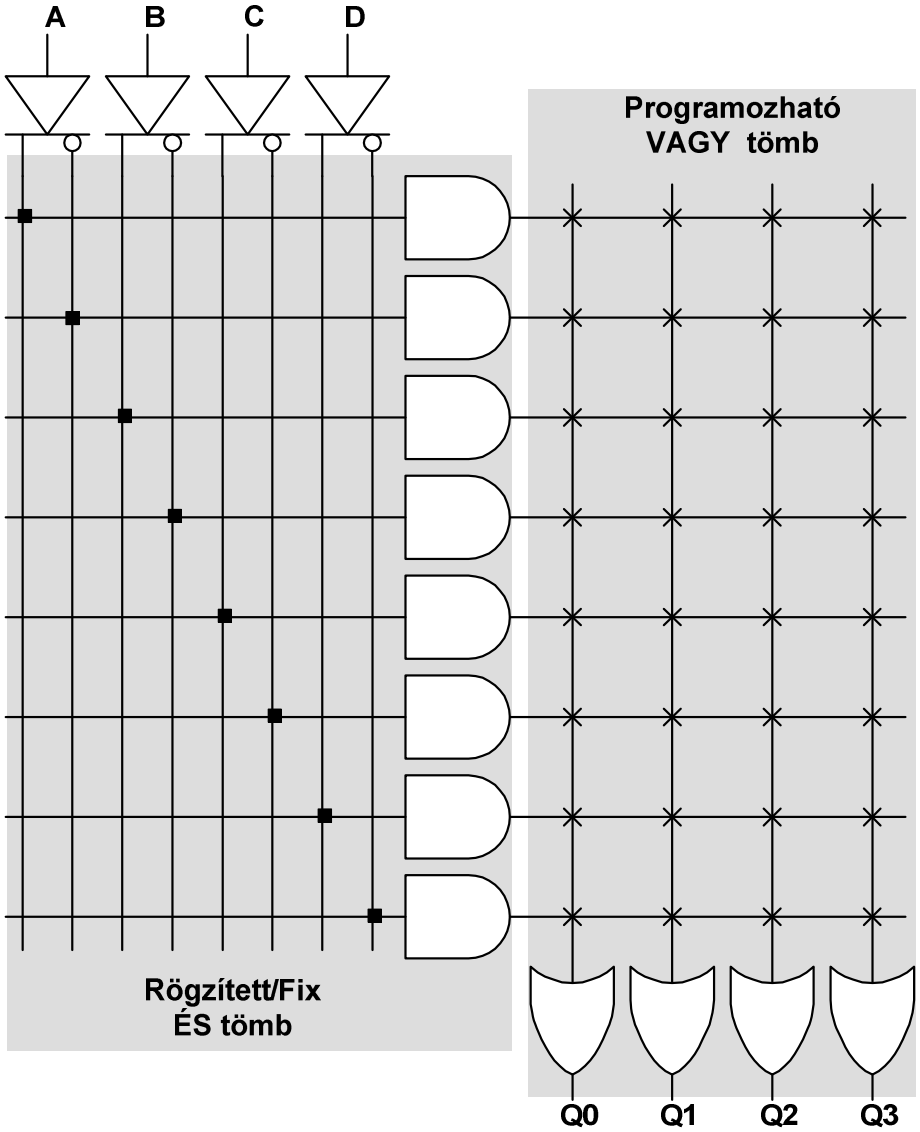


PAL

- Elsőként, 1978-ban az MMI (Monolithic Memories Inc.) jelent meg ilyen programozható eszközökkel, majd pedig későbbi jogutódja a Lattice Semiconductor, illetve az AMD a 80'-as évek végén.
- A PAL hálózatban a programozható részt az ÉS tömb jelenti, míg az VAGY tömb fix/rögzített. Így a tetszőleges mintermeknek csak egy véges kombinációja (VAGY) állítható elő: a lehetséges kimeneti függvények variálhatóságból veszítünk, cserébe viszont a VAGY részek dedikált útvonalainak jelterjedési sebessége nagyobb, míg az eszköz mérete kisebb és ezáltal olcsóbb is lesz.
- Ezáltal a metszéspontokban kevesebb kapcsoló szükséges („gyorsabb", mint a PLA). Hasonlóan a PLA-khoz, amennyiben a Qn kimenet(ek)re tárolókat kötünk (pl. egyszerű D tárolót), majd pedig visszacsatoljuk a programozható PAL logikai hálózat bemenete(i)re akár sorrendi hálózati viselkedést is könnyen valósíthatunk.

Programmable Read Only Memory (PROM)

- Egy programozható rész - OR / míg az AND fix (vmilyen struktúra szerint)
- 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/rögzített

GAL (Generic Array Logic): Általános tömb logika

- 1985-ben a Lattice Semiconductor fejlesztette ki elsőként,
  - amely a *PAL*-nak egy továbbfejlesztett változatát képviseli.
  - Ugyanolyan belső struktúrával rendelkezik, mint egy *PAL* áramkör,
  - azonban többször programozható: tehát törölhető és újraprogramozható eszköz.
  - EEPROM technológiát (lásd. lebegő-gate) alkalmaz. Később a National Semiconductor, és AMD is megjelent saját *GAL* sorozataival a piacon.

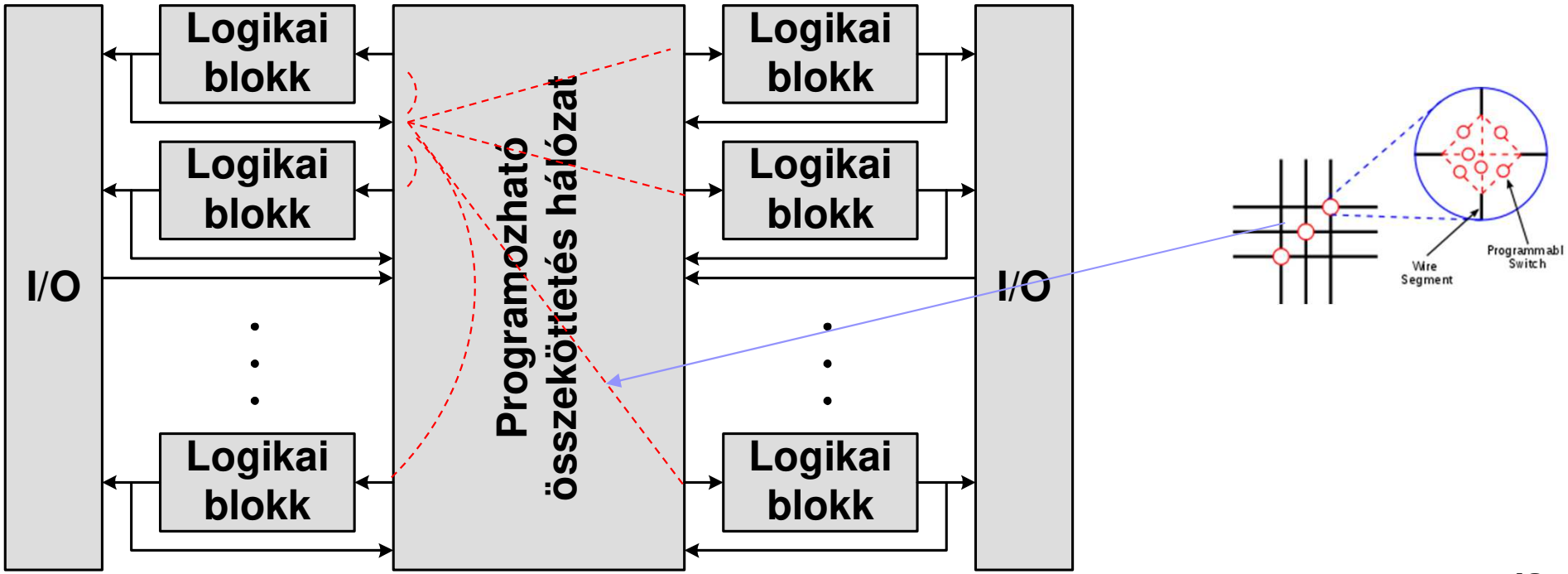
CPLD

CPLD (Comlex Programmable Logic Devices): Komplex-programozható Logikai eszközök

- Valójában *átmenetet* képeznek a kis/közepes integráltsági fokú makrocellás PLD-k GAL/PAL áramkörei, illetve a nagy integráltsági fokú FPGA kapu-áramkörök között.
- A GAL/PAL áramköröktől architekturálisan annyiban különbözik, hogy ki lett bővítve: nem egy-, hanem több logikai cellamátrixot tartalmaz, amelyek konfigurálható blokkok reguláris struktúrájában vannak elrendezve. A mai modern FPGA áramköröktől viszont az különbözteti meg felépítésben, hogy *nem tartalmaz dedikált* erőforrásokat (pl. szorzók, memória blokkok).
- a legnagyobb gyártók, amelyek jelenleg is aktív szereplői a CPLD-k piacának a következők: Xilinx, Altera, Lattice Semiconductor, Atmel stb.

Complex Programmable Logic Device (CPLD)

- 1 Logikai Blokkon belül:
  - ~ PAL / PLA (K.H.)
  - Regiszterek (D-FF)
- I/O Blokkok
- Programozható összeköttetések (PI: Programmable Interconnection)
  - Teljes (Full-crossbar), vagy
  - Részleges összeköttetés hálózat



CPLD (folyt)

A CPLD-kben található Logikai Blokk-ok (=**makrocellák**):

- egyrészt *logikai kapuk* tömbjeit tartalmazzák (hasonlóan a PAL/GAL áramkörök felépítéséhez – DNF alak),
- másrészt *regisztereket* (D-tárolókból) tartalmaznak a logikai tömbök által előállított kimenetek átmeneti tárolásához, valamint
- *multiplexereket*, mellyel a programozható összeköttetés hálózatra, vagy I/O blokkok celláihoz lehet továbbítani a belső Logikai Blokkok által előállított kimeneti értékeket. Ezáltal nemcsak logikai kombinációs hálózatokat, hanem akár sorrendi hálózatokat is egyszerűen megvalósíthatunk CPLD-k segítségével

- A CPLD-kben található *Programozható összeköttetés hálózat*
  - teljes összeköttetést (mindenki-mindekivel), vagy
  - részleges összeköttetést (valamilyen struktúra szerint, pl. bemenetet – kimenettel, főként régi CPLD típusok esetén) biztosít az egyes blokkok között.
- Kikapcsoláskor a CPLD konfigurációs memóriája megtartja értékét (non-volatile típus), ezért nem kell egy külső pl. ROM memóriát használni az inicializációs minták tárolásához, bekapcsoláskor ezek automatikusan betöltésre kerülnek. A CPLD-ket közkedvelten alkalmazzák különböző interfészek jeleinek összekapcsolásához *(glue-logic*), amennyiben a jeleken átalakításra is szükség van, továbbá áraik az FPGA-k árainál jóval kedvezőbbek

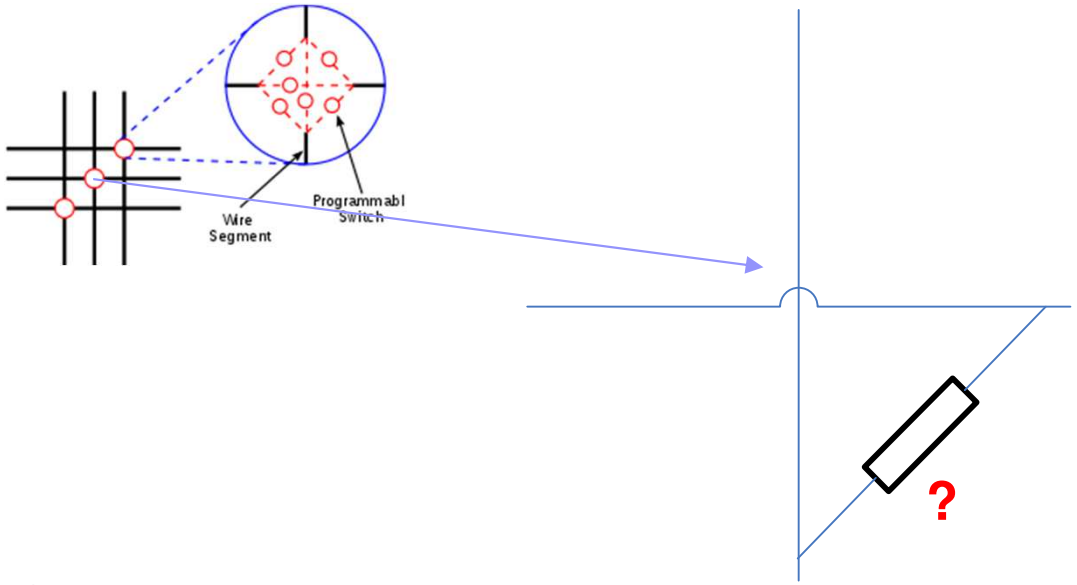
Hogyan programozhatók a VLSI alkatrészek?

Programozási technikák **összeköttetésekre**

Programozási technikák

Mi van a programozható összeköttetések csomópontjaiban, illetve milyen módszerrel programozhatóak?

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



$$\tau = R \cdot C$$

$$[ns / ps]$$

Programozás = konfigurálás

Konfigurálás (PLD/FPGA esetén) – mielőtt az eszközt használni szeretnénk egy speciális (manapság általában JTAG szabványú) programozó segítségével „fel kell programozni”: le kell tölteni a konfigurációs állományt (bitfájl, vagy object fájl). A programozás a legtöbb PLD esetében a belső programozható összeköttetésének fizikai típusától függően azok beállításával történik. A programozható összeköttetésekben a következő lehetséges alkatrészek találhatóak:

- ***Biztosíték (Fuse):*** átégetésük után nem visszafordítható a programozási folyamat (OTP). Korábban a PAL eszközök népszerű kapcsoló elemeként használták.
- ***Antifuse technológia***: (OTP), az antifuse-os kristályszerkezetű kapcsoló elem ’átolvasztása’ után egy nagyon stabilan működő összeköttetést kapunk, amely sajnos szintén nem visszafordítható folyamatot jelent. A technológia drága, az előállításához szükséges maszk-rétegek nagy száma miatt, nagyon jó zavarvédettség elérése érdekében használják (pl. űrkutatás).
- ***SRAM cella + tranzisztor***: tetszőlegesen programozható (FPGA-k esetén legelterjedtebb kapcsolás-technológia), az SRAM-ban tárolt inicializáló értéktől függően vezéreli a tranzisztor gate-elektródáját
- ***SRAM cella + multiplexer***: tetszőlegesen programozható az SRAM cellában tárolt értéktől függően (kiválasztó jel) vezérelhető a multiplexer
- ***Lebegő kapus tranzisztor (Floating Gate) technológia***: elektromosan tetszőlegesen programozható, a mai EEPROM/Flash technológia alapja.