Pannon Egyetem
Villamosmérnöki és Információs Rendszerek Tanszék

Digitális Áramkörök

(Villamosmérnök BSc / Mechatronikai mérnök MSc)

8-9. hét – Sorrendi hálózatok alapfogalmai.
Elemi sorrendi hálózatok (tárolók)

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

Kapcsolódó jegyzet, segédanyag:

- http://www.virt.uni-pannon.hu
  → Oktatás → Tantárgyak → Digitális Áramkörök (Villamosmérnöki BSc / Mechatronikus BSc/MSc).
  - Fóliák, óravázlatok (.ppt)
  - Frissítésük folyamatosan

Eddig:

- Ideális áramkörök: a kommunikáció (két kapu közötti információátvitel) sebességét végtelenül gyorsnak tekintettük (K.H).
  - (S.H.) Valós áramkörök esetében azonban a kapuk és összeköttetések véges kapukésleltetéssel (propagálási idő) rendelkeznek, amelyet figyelembe kell venni!
- A kimeneti értékek generálása csak az aktuális bemeneti kombinációtól függött. (K.H)
  - (S.H.) Azonban a korábbi állapotok értékét is figyelembe kell vennünk!

Sorrendi (szekvenciális) hálózatok tervezése

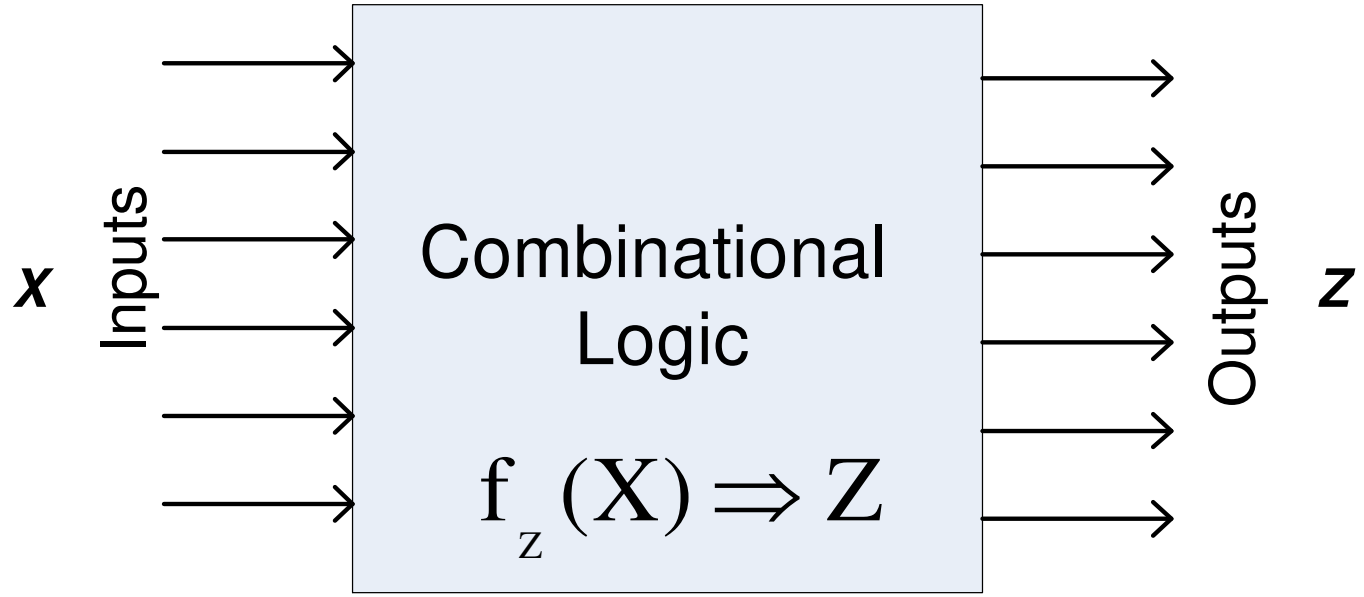
Sorrendi hálózatok (S.H.)

- S.H. modelljei: **Mealy** vs. **Moore**
- S.H. működési módjai
  - Állapottábla, vagy állapot-gráf
- Elemi sorrendi hálózatok (**tárolók**)

- **Szinkron** vs. **Aszinkron** Sorrendi Hálózatok
- Állapot összevonás és állapotkódok
- Működési hibák (hazárd) vizsgálata: késleltetés
  - Kritikus versenyhelyzet (rendszer hazárd)
  - Lényeges hazárd

Ism: Kombinációs hálózatok

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

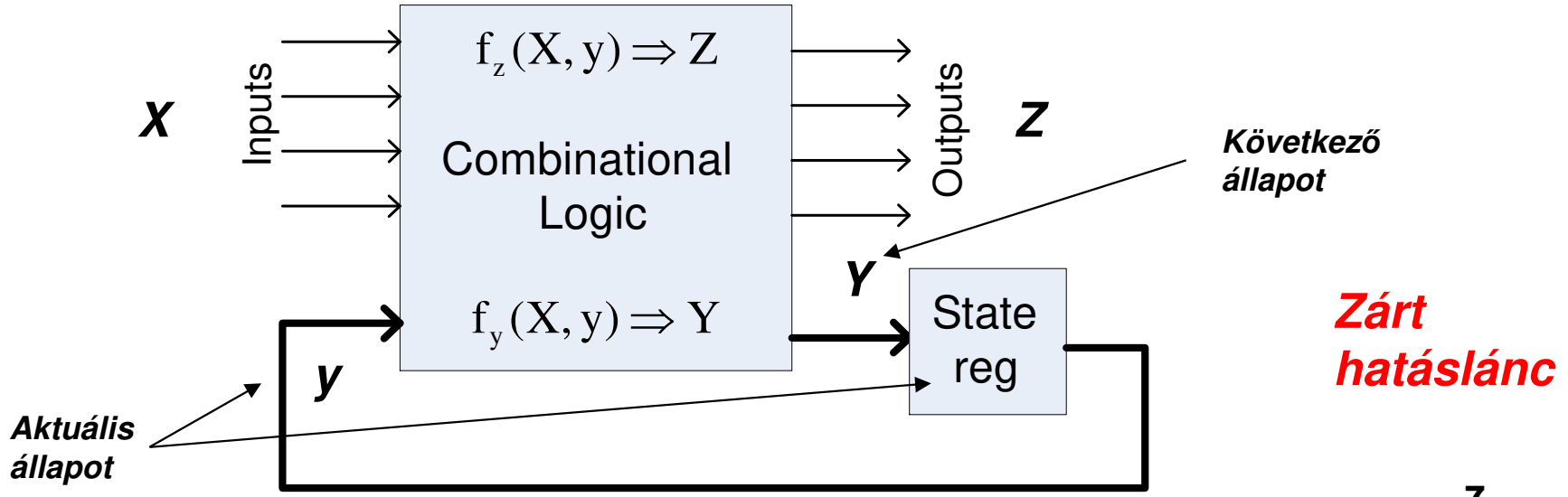


Nyílt hatáslánc

Egyértelmű hozzárendelés (de nem kölcsönösen egyértelmű)

Sorrendi (szekvenciális) 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)



Zárt hatáslánc

Sorrendi hálózatok leképezési szabályai, halmazai

$$f_z(X, y) \Rightarrow Z$$ ***vagy*** $$f'_z(y) \Rightarrow Z$$

Mealy modell ***Moore modell***

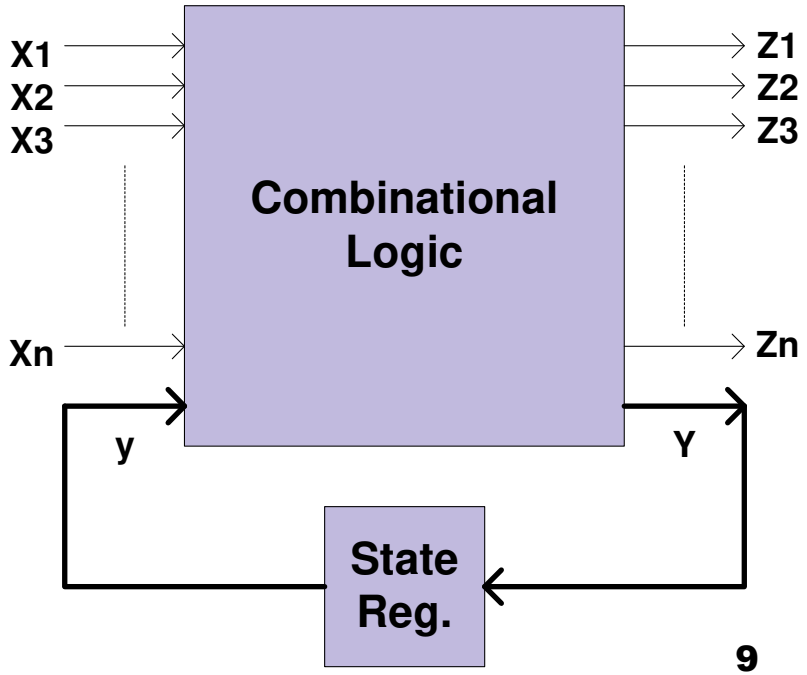
$$f_y(X, y) \Rightarrow Y$$

ahol

- X: bemeneti kombinációk halmaza
- Z: kimeneti kombinációk halmaza
- y: a bemenetre pillanatnyilag visszacsatolt szekunder kombinációk (*aktuális állapotok*) halmaza
- Y: az X és y által létrehozott *soron következő* szekunder kombinációk (*köv. állapotok*) halmaza

1.) Mealy-modell

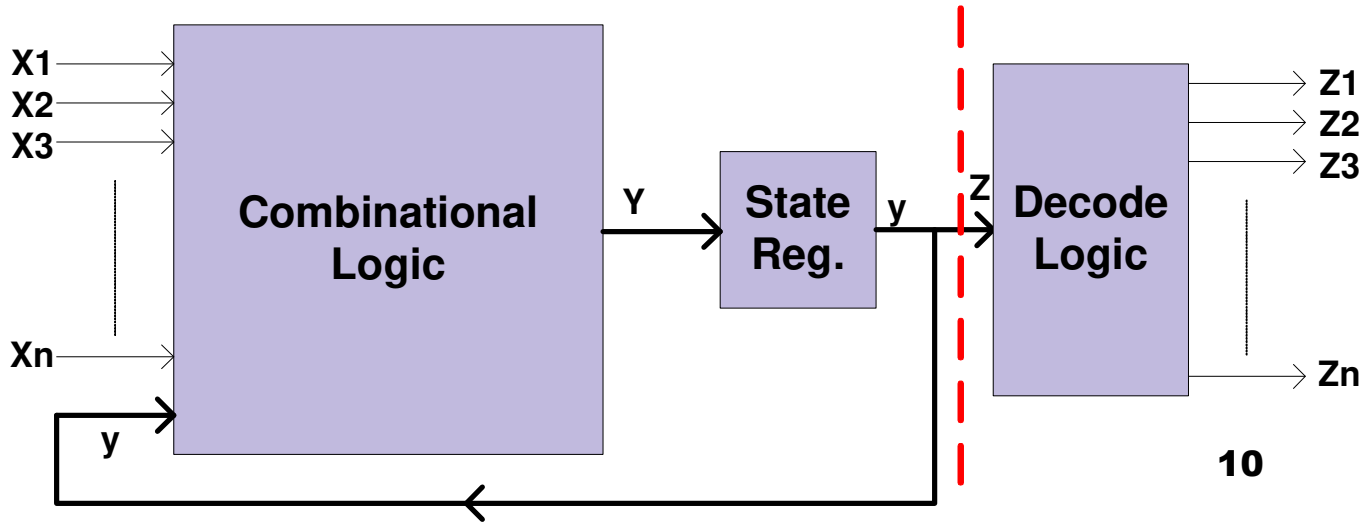
- A sorrendi hálózatok egyik alapmodellje. Késleltetés: a kimeneten az eredmény véges időn belül jelenik meg! Korábbi értékek visszacsatolódnak a bemenetre: kimenetek nemcsak a bemenetek pillanatnyi, hanem a korábbi állapotoktól is függenek. Problémák merülhetnek fel az állapotok és bemenetek közötti szinkronizáció hiánya miatt (változó hosszúságú kimenetet - dekódolás). Ezért alkalmazzuk 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:
  - $f_y(X,y) \rightarrow Y$: következő állapot fgv.
  - $f_z(X,y) \rightarrow Z$: kimeneti fgv.



2.) Moore-modell

- A kimenetek közvetlenül csak a pillanatnyi állapottól függenek (bemenettől függetlenek v. közvetett módon 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.
- Két leképezési szabályok
  - $f_y(X,y) \rightarrow Y$: következő állapot fgv.
  - $f_{z'}(y) \rightarrow Z$: kimeneti fgv.

Input ⇒ Next-State ⇒ Present-State ⇒ Output



DEF: Nyugalmi állapot (aszinkron hálózatokban)

- **Nyugalmi állapot** egy adott *X* bemeneti kombináció mellett csak akkor jöhet létre, ha egy kialakult ***Y*** szekunder kombináció a bemenetre ***y***-ként visszajutva/visszahatva (azaz y←Y) az $f_y(X,y)$ leképezés alapján ***változatlan Y*** kombinációt hoz létre.
  - Nem változik a **nyugalmi állapot** (***stabil*** állapotnak nevezzük ezt a továbbiakban)
  - Természetesen az **y≠Y** szekunder kombináció csak átmenetileg állhat fenn, ezt nevezzük ***instabil*** állapotoknak)

Instabil állapotok és oszcilláció (**aszinkron** hálózatokban)

- Instabil állapotok fennállási idejét
  - a visszacsatoló ágak „jelterjedési” (propagációs) késleltetése, valamint
  - Az f_y leképezést megvalósító logikai K.H. „megszólalási” ideje

együttesen határozzák meg

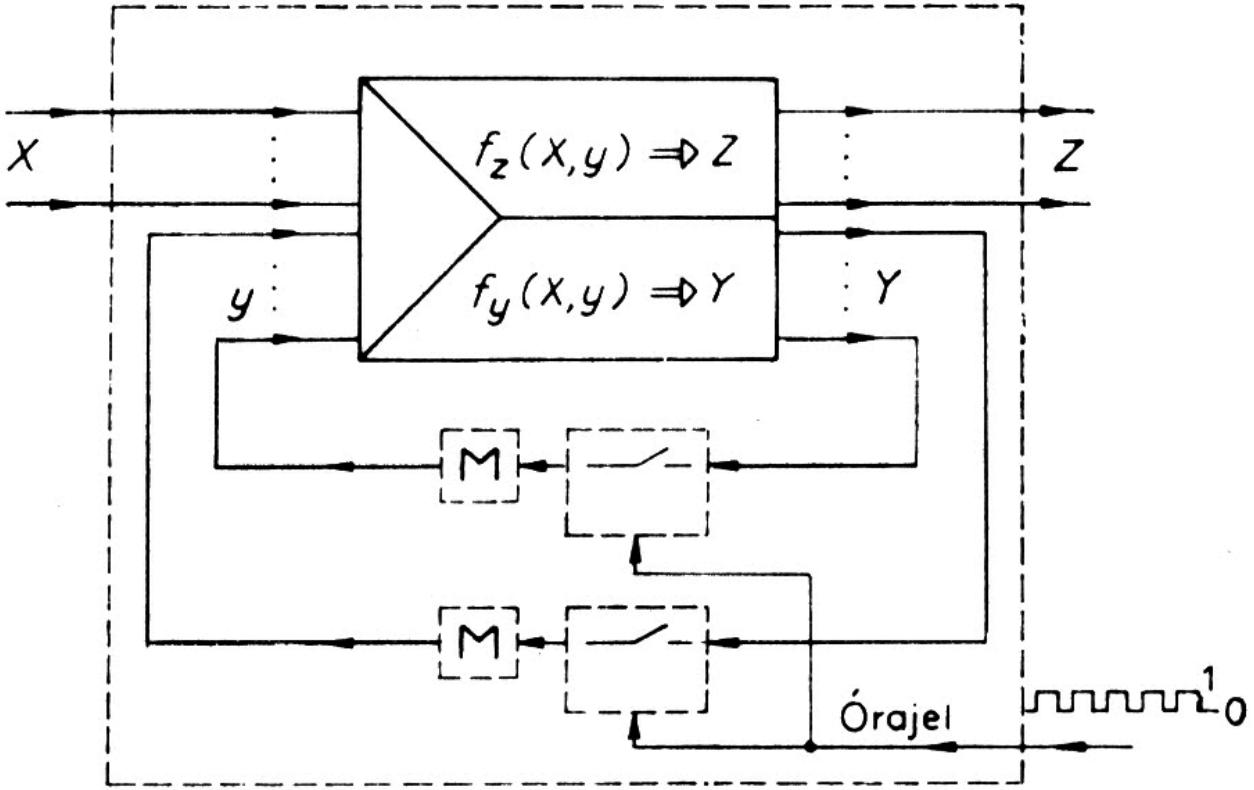
- Ha az instabil állapot alatt éri a S.H.-ot ***X*** bemeneti változás, akkor ennek hatására kialakuló ***Y*** és ***Z*** kombináció attól fog függeni, hogy a bemeneti változás pillanatában éppen melyik instabil állapot állt fenn.
- Ha nem alakul ki stabil állapot (adott bemenetre) és az instabil állapotok állandóan váltják egymást, akkor **oszcillációról** beszélünk.

Aszinkron sorrendi hálózatok

- Aszinkron S.H. minden esetben megvalósíthatók ***visszacsatolt K.H***. megadásával
  - De megvalósíthatók ***szinkron S.H.*** segítségével (pl. elemi tárolók megadásával)
  - Lásd: aszinkron S.H. tervezési lépései
- **Normál aszinkron hálózat**: bármely két stabil állapota közötti átmenet során *legfeljebb egy* instabil átmenet van.
- Több szekunder állapot szükséges ált. a működéshez (szinkronhoz képest).

„Ütemezett” aszinkron S.H.

- Visszacsatoló ágakban (Y) periodikusan nyitjuk/zárjuk a kapcsolókat (CLK). M=Memória: Y szekunder állapot tároláshoz (y=Y -> M[Y]=y)
- X/Z: bemenet/kimenet nem ütemezett!



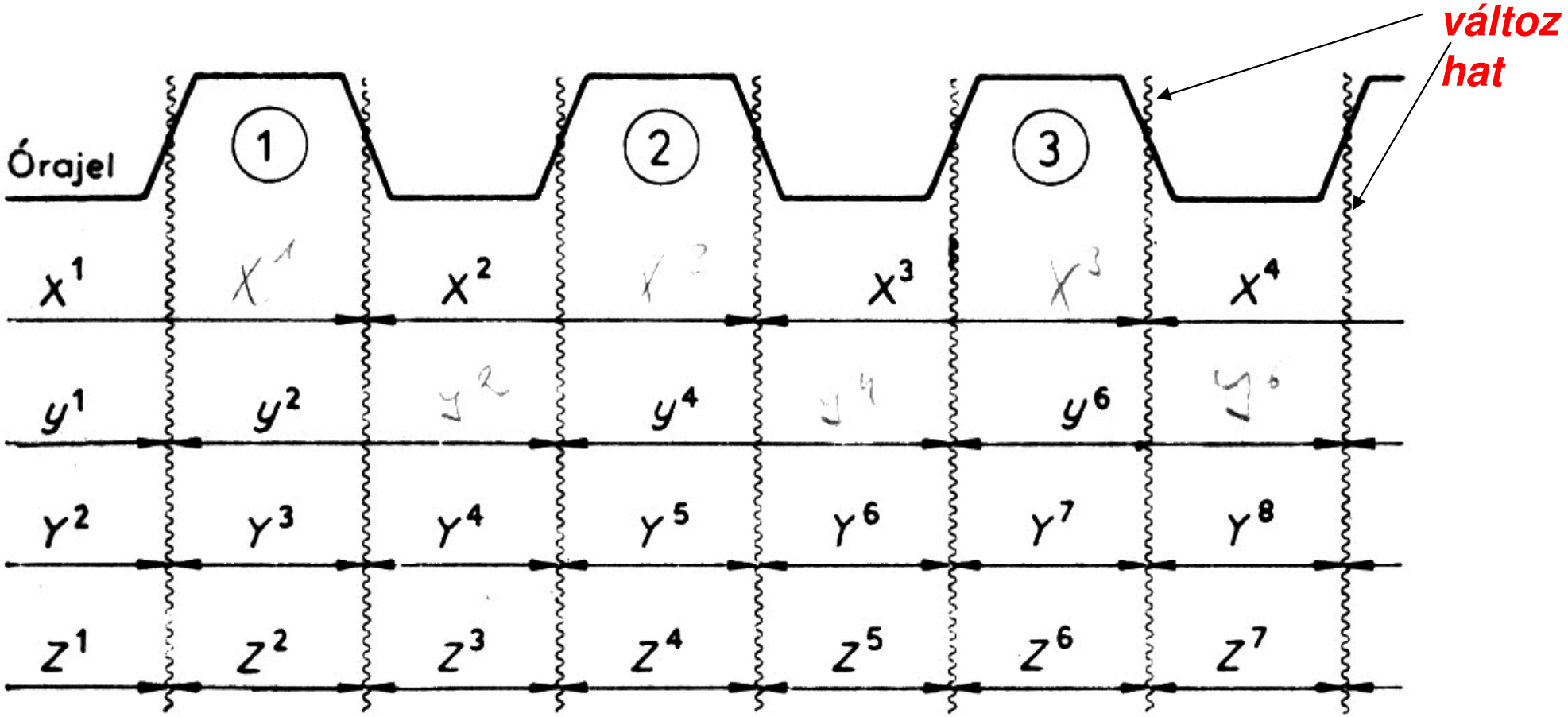
- \- Hálózat sebességét viszont a CLK határozza meg!
- \+ *instabil* állapotok az órajel ütemben következnek egymás után

Szinkron sorrendi hálózatok

- Szinkron S.H: Nem csak a *visszacsatoló* ágak (Y) *szekunder* változói, hanem az X *bemeneti kombinációk* is periodikusan, órajel hatására (CLK) érkezhetnek.
- Minden – stabil / instabil – állapotban érkezik új érvényes bemenet. Ezáltal **nem különböztetjük meg az instabil / stabil állapotokat szinkron esetben!**
- Kevesebb szekunder állapot szükséges a működéshez (aszinkronhoz képest).

Szinkron S.H. működése Mealy modell esetén

$f_z(X, y) \Rightarrow Z$ $f_y(X, y) \Rightarrow Y$



X: lefutó élre vált ↘

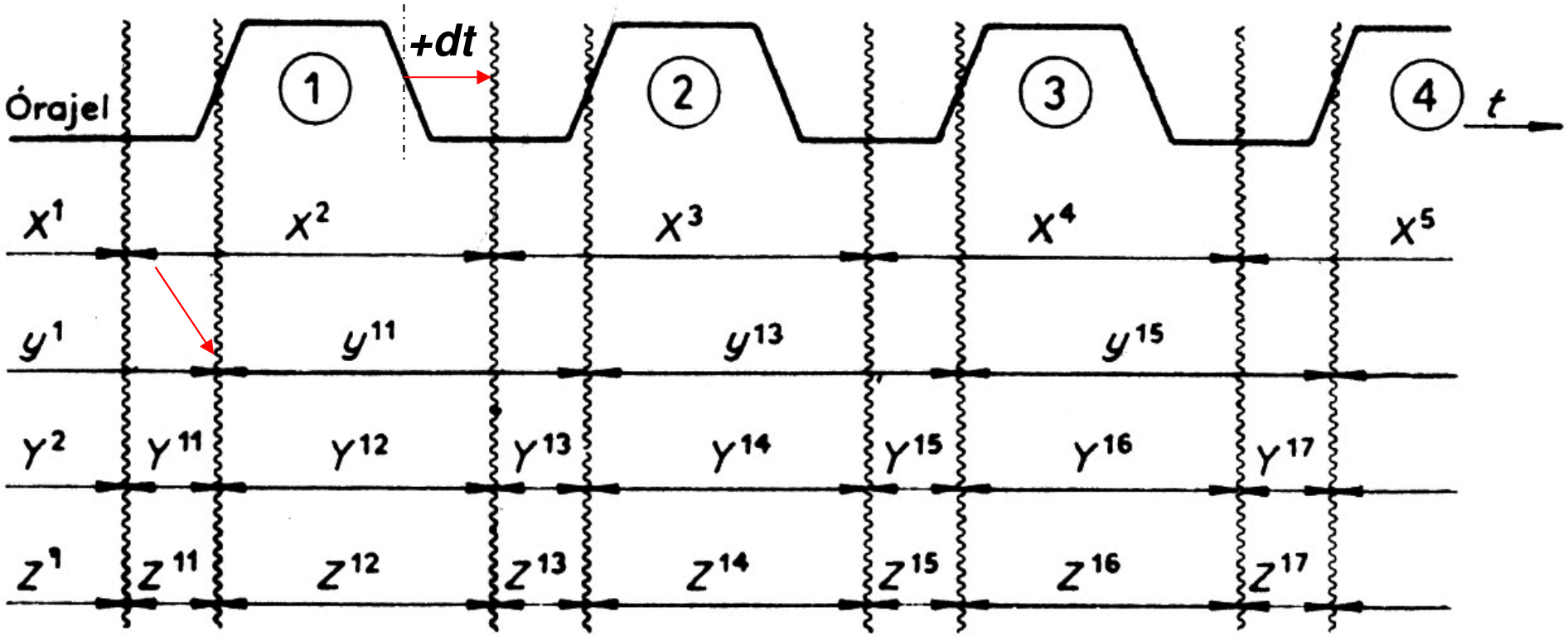
y: felfutó élre vált ↗

index = azonos kombináció

Kezdetben: $f_y(X^1, y^1) \Rightarrow Y^2$

Majd: $f_y(X^2, y^2) \Rightarrow Y^4$ $f_z(X^2, y^2) \Rightarrow Z^3$

Szinkron S.H. működése Mealy modell esetén (‘X’ megelőzi ‘y’ változását)

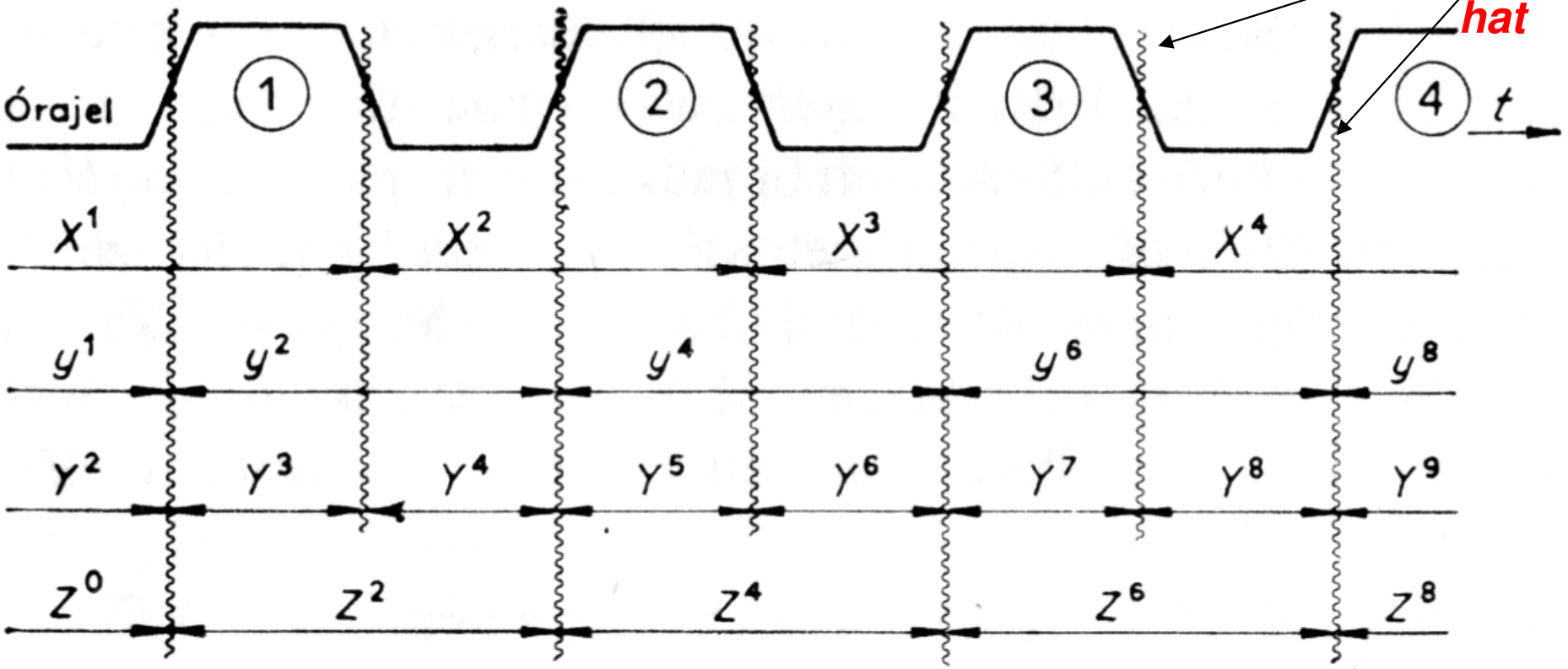


X: lefutó élre vált (+dt) ↘

y: felfutó élre vált ↗

Szinkron S.H. működése Moore modell esetén

$f'_z(y) \Rightarrow Z$ $\quad f_y(X, y) \Rightarrow Y$



X: lefutó élre vált ↘

y: felfutó élre vált ↗

index = azonos kombináció

Kezdetben: $f_y(X^1, y^1) \Rightarrow Y^2$

Majd: $f_y(X^2, y^2) \Rightarrow Y^4$ $\quad f'_z(y^2) \Rightarrow Z^2$

DEF: Szinkronizációs feltételek

- Szinkron működéshez:
  - A Y(y) visszacsatoló ágak órajel-vezérléssel (CLK) periodikusan megszakíthatóak legyenek (tartalmazzanak ‚M' tároló-eleme(ke)t),
  - Az X bemeneti változások az órajellel szinkronban történjenek,
  - Definiálni kell a Z kimeneti kombinációk értelmezési időtartományát.

Összefoglalás:

- Aszinkron S.H. tulajdonságai:
  - Az *instabil* állapotok miatt a szükséges *szekunder változók* száma ált. *több*, mint szinkron esetben, ezért logikai tervezésük is bonyolultabb.
  - A bemeneti változók gyakoriságát, vagyis a működési sebességet csak az *építőelemek jelterjedési* és *megszólalási* késleltetése korlátozza.
  - A logikai tervezésük után *nem kell* szinkronizációs feltételeket biztosítani, azaz megépíthetők tisztán visszacsatolt K.H.-okkal is.

Összefoglalás:

- Szinkron S.H. tulajdonságai:
  - A stabil / instabil állapotok nincsenek külön értelmezve, ezáltal a szükséges *szekunder változók* száma ált. *kevesebb*, mint aszinkron esetben, ezért logikai tervezésük is *egyszerűbb* lehet.
  - A bemeneti változók gyakoriságát, vagyis a működési sebességet a *választott működési frekvencia (órajel)* korlátozza, ezért ált. lassabbak mint az ütemezés nélküli aszinkron sorrendi hálózatok.
  - A logikai tervezésük után a *szinkronizációs feltételeket* biztosítani kell (M).
  - A bemeneti változásokra és a kimeneti kombináció értelmezésére a *szinkronizációs feltételeknek* kell teljesülniük.

Sorrendi (szekvenciális) hálózatok működésének leírása

Sorrendi (szekvenciális) hálózatok működésének leírása - Állapottábla

- Egyértelmű működés leírásához az szükséges, hogy ismerjük az f_z és f_y leképezéseket.
- **Def: Állapottábla**: a táblázat mindenegyes rovatába/cellájába azokat az **Y** és **Z** kombinációkat kell beírni, amelyeket a leképezések az adott cellához rendelt **X** és **y** kombinációk fennállása esetén hoznak létre.

PI: Állapottábla

- X: 4 bemeneti kombináció
- y/Y: 4 szekunder változó (állapot) kombináció
- Z: 4 kimeneti kombináció

	00	01	10	11
y \ X	X^1	X^2	X^3	X^4
00	y^1	$Y^1 Z^1$	$Y^2 Z^4$	$Y^4 Z^2$	$Y^4 Z^4$
01	y^2	$Y^1 Z^4$	$Y^2 Z^3$	$Y^2 Z^2$	$Y^3 Z^1$
10	y^3	$Y^4 Z^3$	$Y^3 Z^1$	$Y^3 Z^3$	$Y^1 Z^2$
11	y^4	$Y^4 Z^2$	$Y^2 Z^2$	$Y^3 Z^3$	$Y^2 Z^8$

PI:

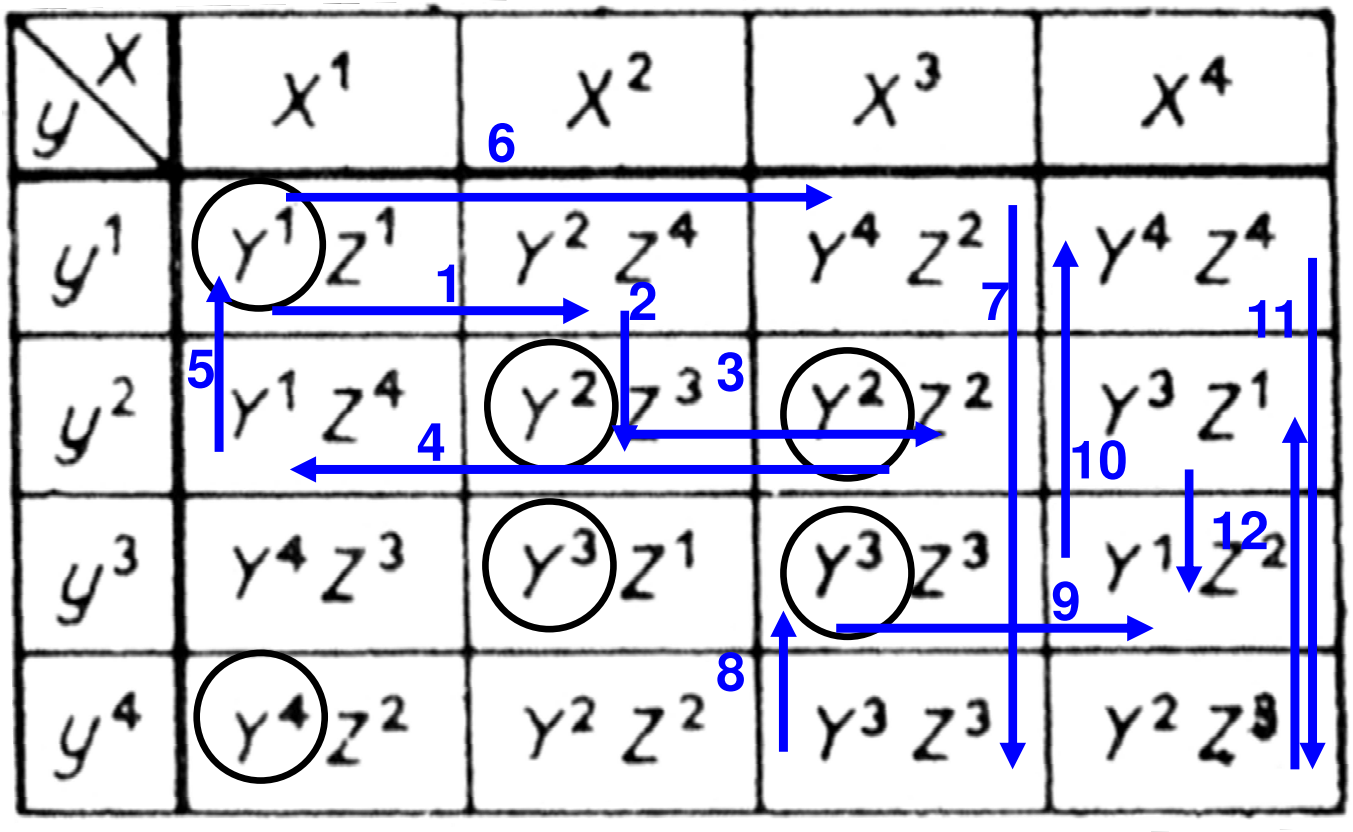
$$f_y(X^1, y^1) \Rightarrow Y^1$$

$$f_z(X^1, y^1) \Rightarrow Z^1$$

Tömör jelölés: felső indexek az egyes kombinációkat jelölik (a megfelelő bináris számok helyett - 00…11)

Példa 1. Aszinkron működés:

- Kezdeti stabil állapot, Tfh: $X=X^1$, $y=y^1$, $Y=Y^1$, $Z=Z^1$
  - Stabil állapotot „**karikázva** jelöljük”: ahol az $Y^i=y^i$ indexekkel azonos
  - **Tfh:** $X^1 \rightarrow X^2 \rightarrow X^3 \rightarrow X^1 \rightarrow X^3 \rightarrow X^4 \rightarrow X^3$ bemeneti változások lesznek



X^1	X^2	X^3	X^1	X^3	X^4	~~X^3~~
$\underline{Z^1}$	Z^4 $\underline{Z^3}$	$\underline{Z^2}$	Z^4 $\underline{Z^1}$	Z^2 Z^3 $\underline{Z^3}$	Z^2 Z^4 Z^3 Z^1....	

Megj: X-el csak akkor lépünk, amikor megvártuk a stabil állapotot.

Hálózat oszcillál! (9-10-11-12 átmenetek után nem kerülünk újból stabil állapotba, hogy X-el léphessünk!)

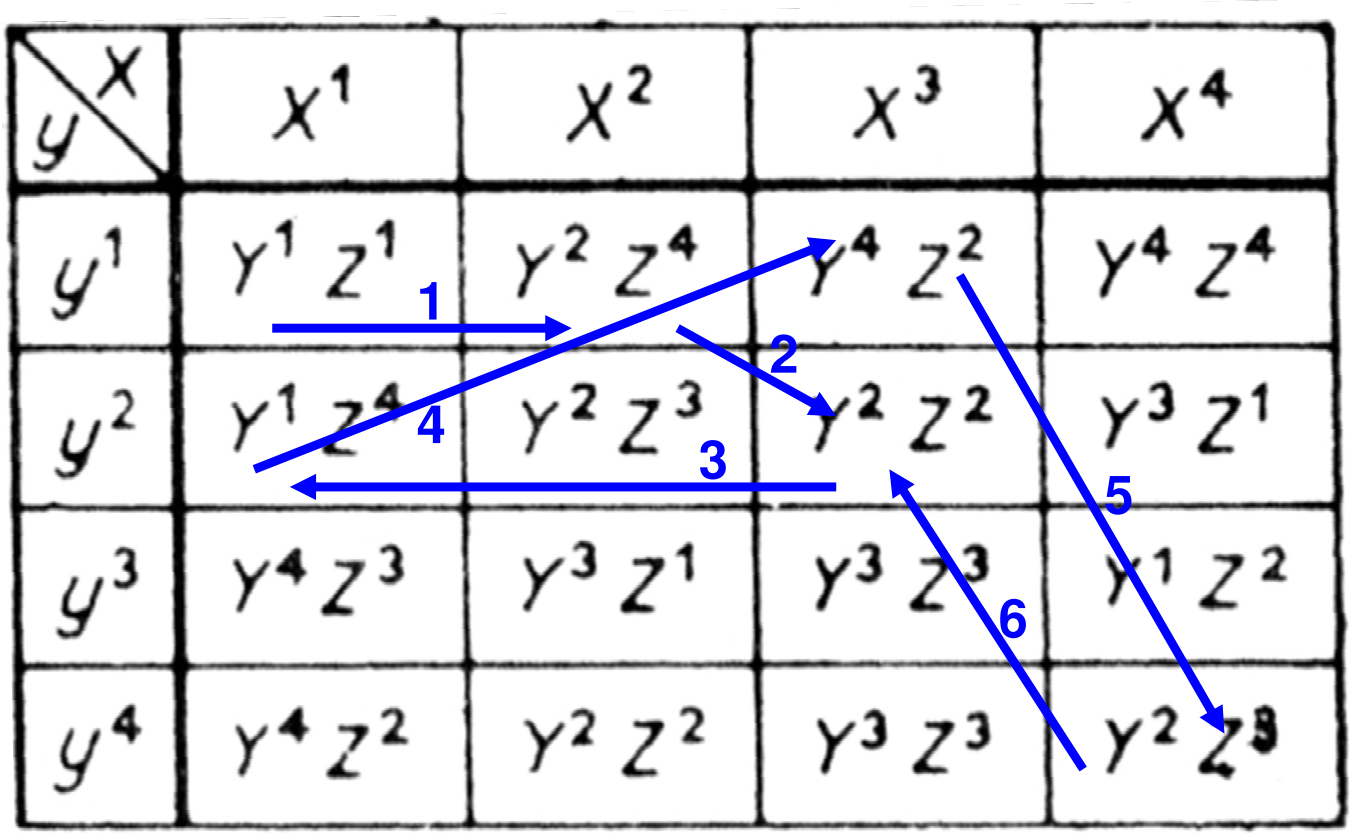
Megj: Aláhúzással jelölve a stabil állapothoz tartozó kimenetet

Példa 1.: (aszinkron következtetés) „Szükséges és elégséges feltétel”

- X^4 fennállása esetén a hálózat oszcillál. Ciklusba kerülünk, amiből nem jutunk stabil állapotba (instabil állapotokban maradunk).
- X^4 jelű oszlopban stabil állapotot nem tudtunk kijelölni kezdetben az állapottáblán!
- Következmény: Oszcilláció
- DEF: Akkor NEM léphet fel oszcilláció **(szükséges feltétel)** ha az állapottábla *minden oszlopában* legalább egy stabil állapot van!
- DEF: Akkor NEM léphet fel oszcilláció, tehát az **elégséges feltételhez** annak is teljesülnie kell, hogy a hálózat minden egyes bemeneti kombinációra, vagyis minden oszlopban *eljusson* egy stabil állapotba!

Példa 1. Szinkron működés:

- Kezdeti állapot, Tfh: $X=X^1$, $y=y^1$, $Y=Y^1$, $Z=Z^1$
  - **Tfh:** $X^1 \rightarrow X^2 \rightarrow X^3 \rightarrow X^1 \rightarrow X^3 \rightarrow X^4 \rightarrow X^3$ bemeneti változások lesznek



X^1	X^2	X^3	X^1	X^3	X^4	X^3
Z^1 Z^1	Z^4 Z^3	Z^2 Z^2	Z^4 Z^1	Z^2 Z^3	Z^3 Z^1	Z^2 Z^2

Megj: 1. CLK-ra az Y^1 válik y^1 –vé ($y^1 = Y^1$).

Majd y^1 visszahatása után fellépő X^2 állapota továbbra is az 1. sorban marad (y1 miatt!), de a 2. oszlop lesz

Megj: minden bemeneti kombinációhoz két kimeneti kombináció is tartozhat, melyből a kimeneti szinkronizáció választja ki a helyeset.

Példa 1.: (szinkron folytatása)

- DEF: Szinkron működés az állapottábla alapján formálisan is megfogalmazható:
  - Egy adott rovatból/cellából kiindulva mindig abba a rovatba/cellába jutunk, amelynek ***i.* sorát** a **kiindulási** rovatban szereplő $\mathbf{Y^i}$ kombináció, ***j. oszlopát*** pedig az **új** $\mathbf{X^j}$ bemeneti kombináció jelöli ki!
- Látható, hogy a szinkron/aszinkron működés eltérő lehet azonos állapottábla és bemeneti kombináció változás esetén is!
- Továbbá X^4 fennállása nem okoz oszcillációt szinkron esetben.

Állapottábla „don’t care” esetre

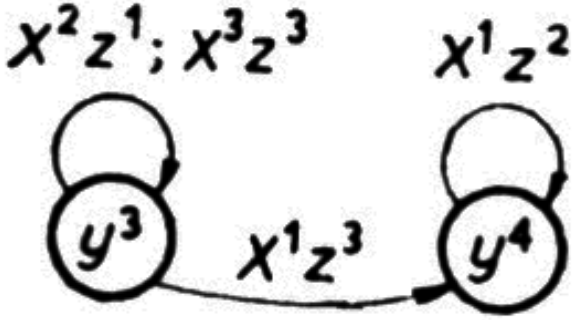
- Állapottáblának lehetnek olyan rovatai is, amelyben nem szerepel előírt Y Z kombináció: *don’t care (közömbös)* állapotok és kimenetek.
- Ekkor intuitív módon, *tetszőlegesen* tölthető ki az állapottábla, úgy hogy a kialakuló S.H. a lehető leggazdaságosabb legyen.
- Ezeket **Nem Teljesen Specifikált Állapottábláknak (NTSÁ)** nevezzük.

DEF: Állapotgráf

- Állapottábla ekvivalens grafikus ábrázolási módja S.H. működésének leírására.
  - A hálózat állapotaira jellemző **y** kombinációknak **állapotokat** (kör), míg
  - a bemeneti **X** kombinációk hatására lejátszódó állapotváltozásokat **állapotátmenetekkel**, vagy **önvisszacsatolásokkal** (hurokkal) feleltetjük meg.

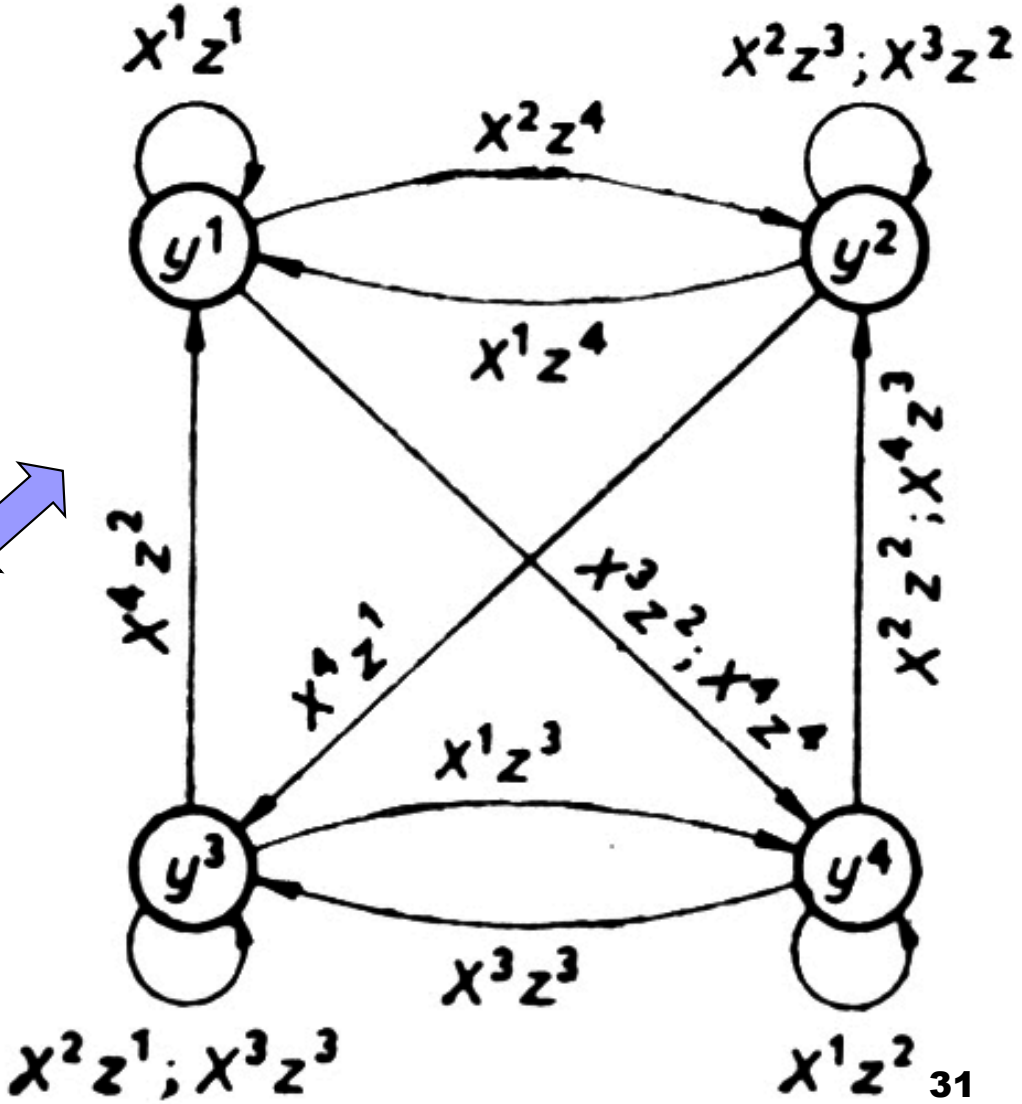
Példák: Állapotgráfok

a.) Tetszőleges példa:



y \ X	X^1	X^2	X^3	X^4
y^1	$Y^1 Z^1$	$Y^2 Z^4$	$Y^4 Z^2$	$Y^4 Z^4$
y^2	$Y^1 Z^4$	$Y^2 Z^3$	$Y^2 Z^2$	$Y^3 Z^1$
y^3	$Y^4 Z^3$	$Y^3 Z^1$	$Y^3 Z^3$	$Y^1 Z^2$
y^4	$Y^4 Z^2$	$Y^2 Z^2$	$Y^3 Z^3$	$Y^2 Z^3$

b.) Korábbi állapottábla alapján felírva:



DEF: Állapotgráf **aszinkron** módú működésének feltételezése

- Csak azok az állapotgráfok értelmezhetők **aszinkron működés** feltételezésével, amelyekben minden egyes átvezető nyílra felírt X^i bemeneti kombináció előbb-utóbb olyan körhöz (állapothoz) vezet, amelyhez tartozó önvisszacsatolás (hurok) címkéjén ugyanaz az X^i bemeneti kombináció szerepel.
- Szinkron esetre nincs kikötés, tetszőleges gráfokra értelmezhető!

Elemi sorrendi hálózatok (tárolók / flip-flopok)

Elemi sorrendi hálózatok (tárolók / flip-flopok)

- Építőelem készlet, amely szinkron és aszinkron S.H. felépítésére alkalmas.
  - de tudjuk, hogy **aszinkron** megvalósítható K.H. *visszacsatolásával* is, ill.
  - **szinkron** esetben biztosítani kell a *szinkronizációs feltételeket* is!
- Elemi S.H. egyszerű építőelemek:
  - egyetlen szekunder változóval,
  - bemeneteik 1-, vagy 2- lehet.
  - egyetlen kimenetük van.

Elemi S.H (tárolók)

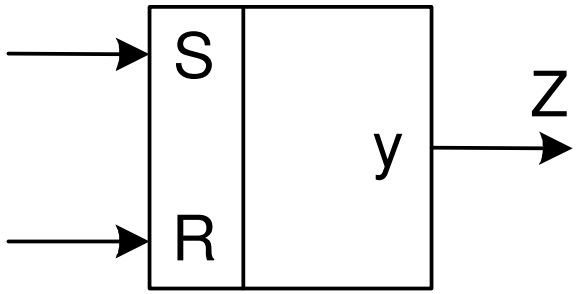
- Az ismertetendő elemi S.H.-k működése a **Moore modell** alapján definiált:

$$f'_z(y) \Rightarrow Z = y$$

  - S-R (Set-Reset Flip-flop)
    - Szinkron/aszinkron
  - J-K FF
    - csak szinkron
  - T FF
    - csak szinkron
  - D-G FF
    - Szinkron/aszinkron
  - D FF
    - csak szinkron

Megj: A kimeneti kombináció azonos a szekunder kombináció bemenetre visszahatott értékével (két állapotot tudunk csak jellemezni '0' v. '1', billenő elemek)

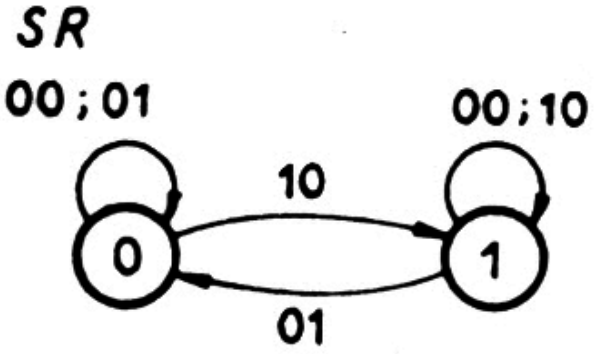
1.) S-R flip-flop



- **Set-Reset tároló (beállítás-törlés)**
  - Szinkron/aszinkron módon is értelmezhető
- **Működése:**
  - `S= '1'` érték az FF állapotát (kimenetét) ‘1’ re állítja
  - `R= '1'` érték az FF állapotát (kimenetét) ‘0’ –ra állítja/reseteli
  - `S=R='0'` esetén az FF állapota (kimenete) nem változik, azaz tárol (Y=y)
  - `S=R='1'` esetén az FF állapota definiálatlan, nem megengedett (dont’ care-el jelöljük)

S-R tárolók: Szinkron / aszinkron működés

Állapotgráf:



a.) Szinkron esetben:

y \ SR	00	01	11	10
0	0	0	-	1
1	1	0	-	1

Y

Ha S=1, írás akkor y: 0->1 lesz

Tárol (R=S=0)

Ha R=1, törlés akkor y: 1->0 lesz

Nem megengedett állapot (R=S=1)

b.) Aszinkron esetben:

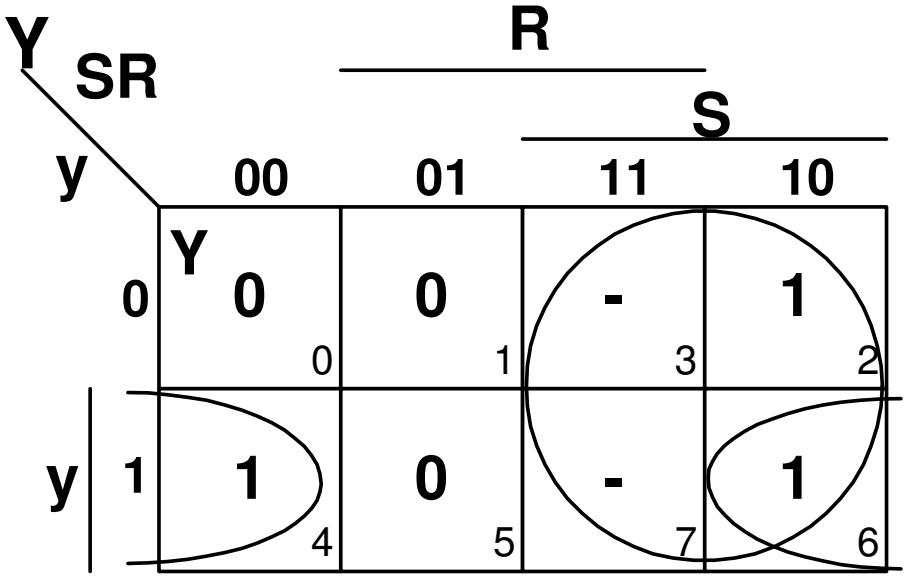
y \ SR	00	01	11	10
0	0	0	-	1
1	1	0	-	1

Y

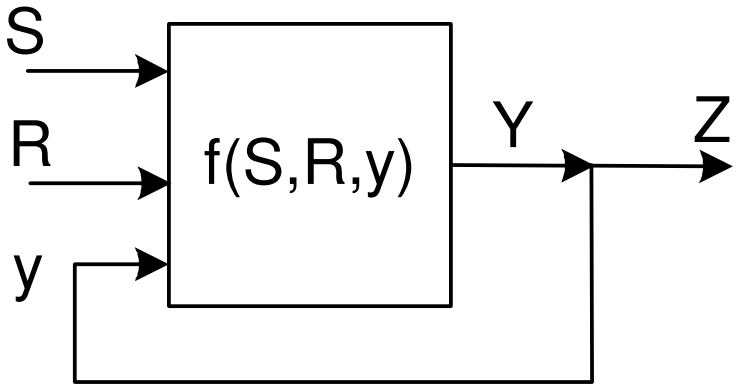
Stabil állapotok

S-R tárolók: Karnaugh tábla és felépítés

Aszinkron esetben:



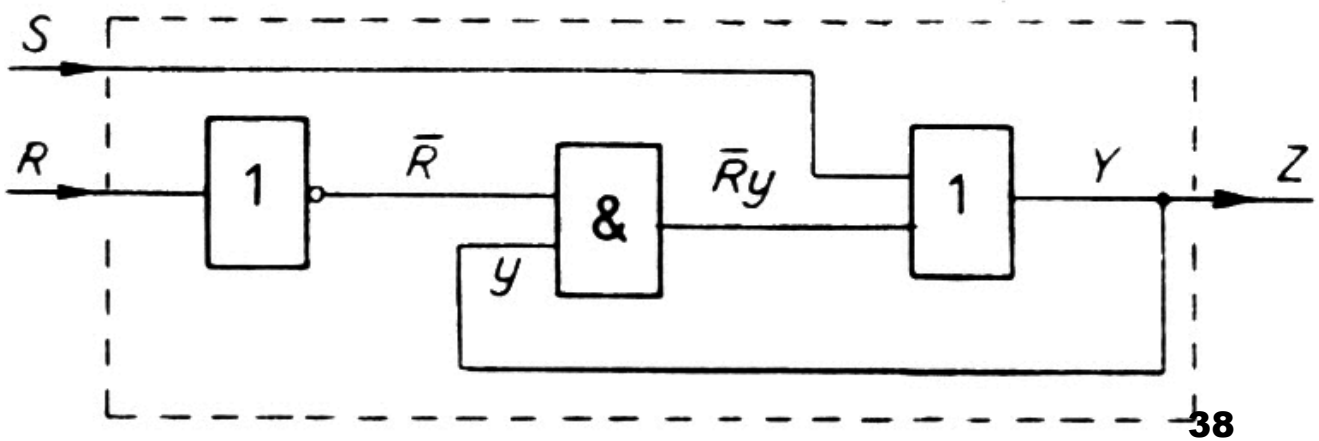
Blokkdiagram: Visszacsatolt K.H-al megadható!



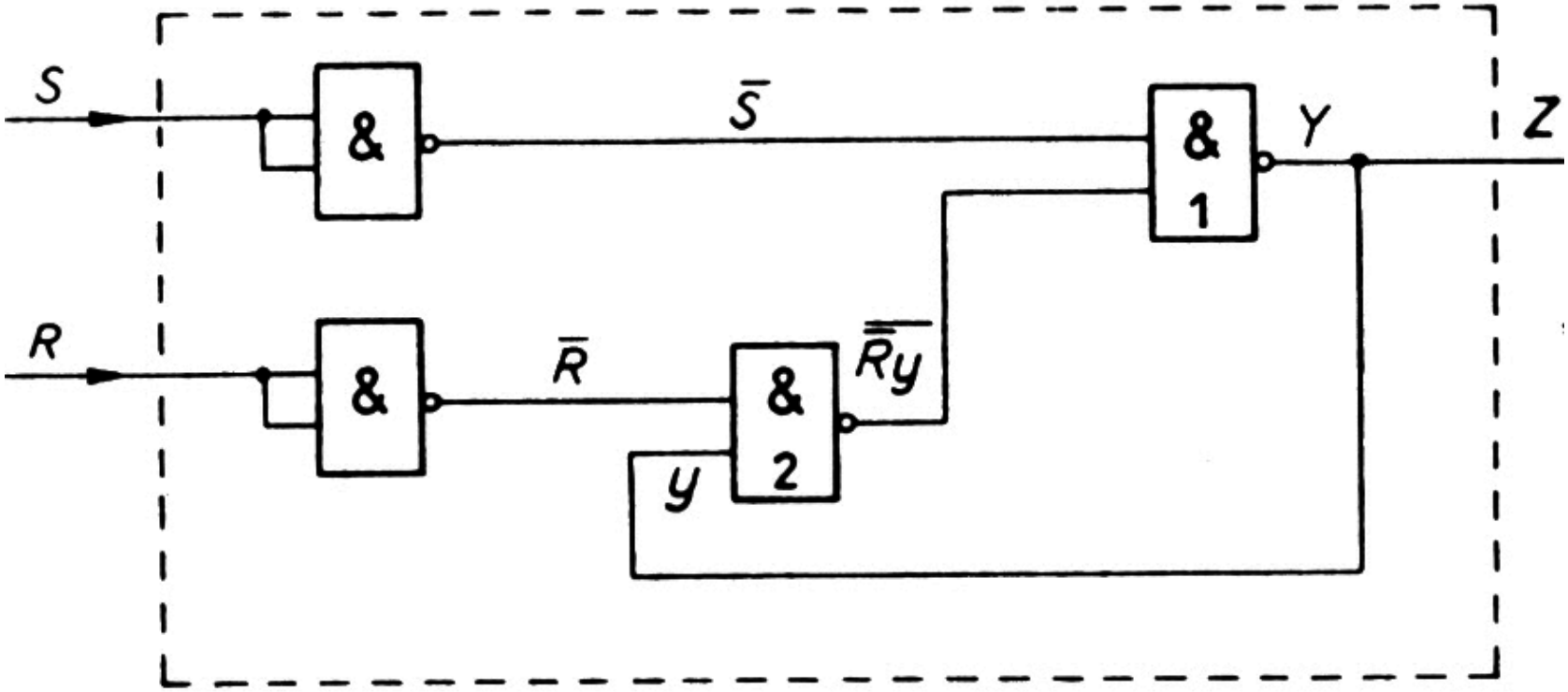
Egyszerűsített DNF alak és elvi logikai rajz:

$$Y = f_y(S, R, y) =$$

$$Z = Y = S + \overline{R}y$$



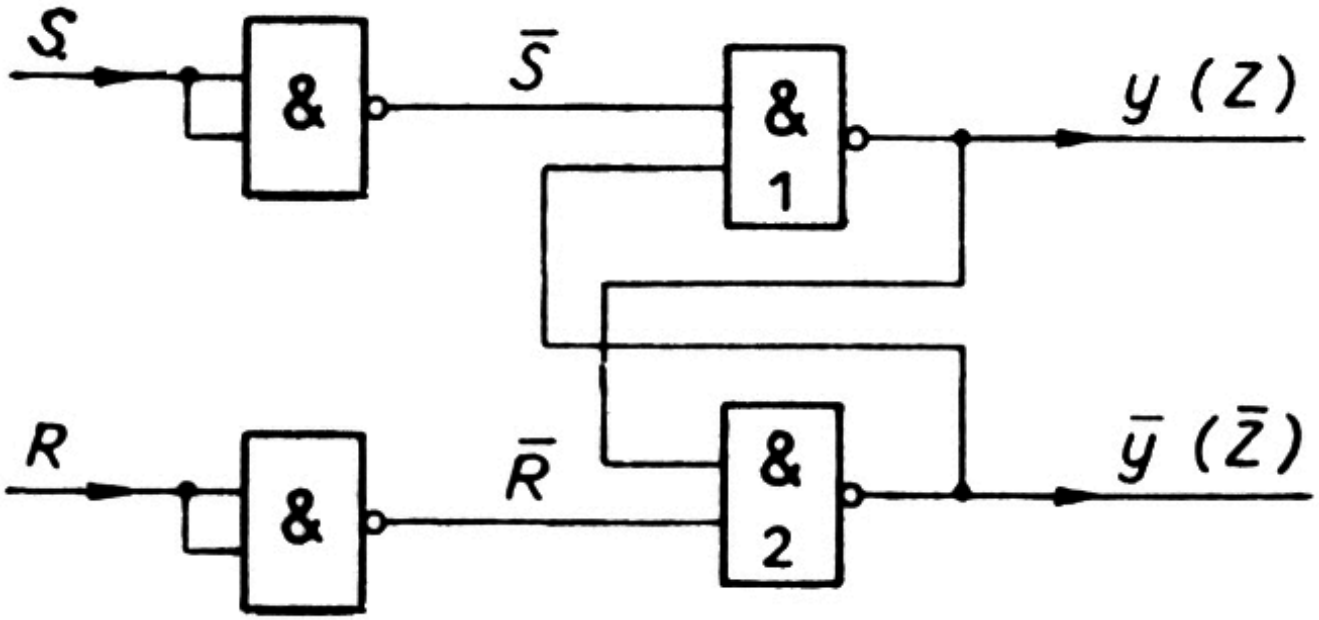
a.) Aszinkron S-R FF megvalósítása visszacsatolt K.H-al. tisztán NAND kapuk felhasználásával



$$Z = Y = S + \bar{R}y = \overline{\bar{S} \cdot \overline{\bar{R}y}}$$

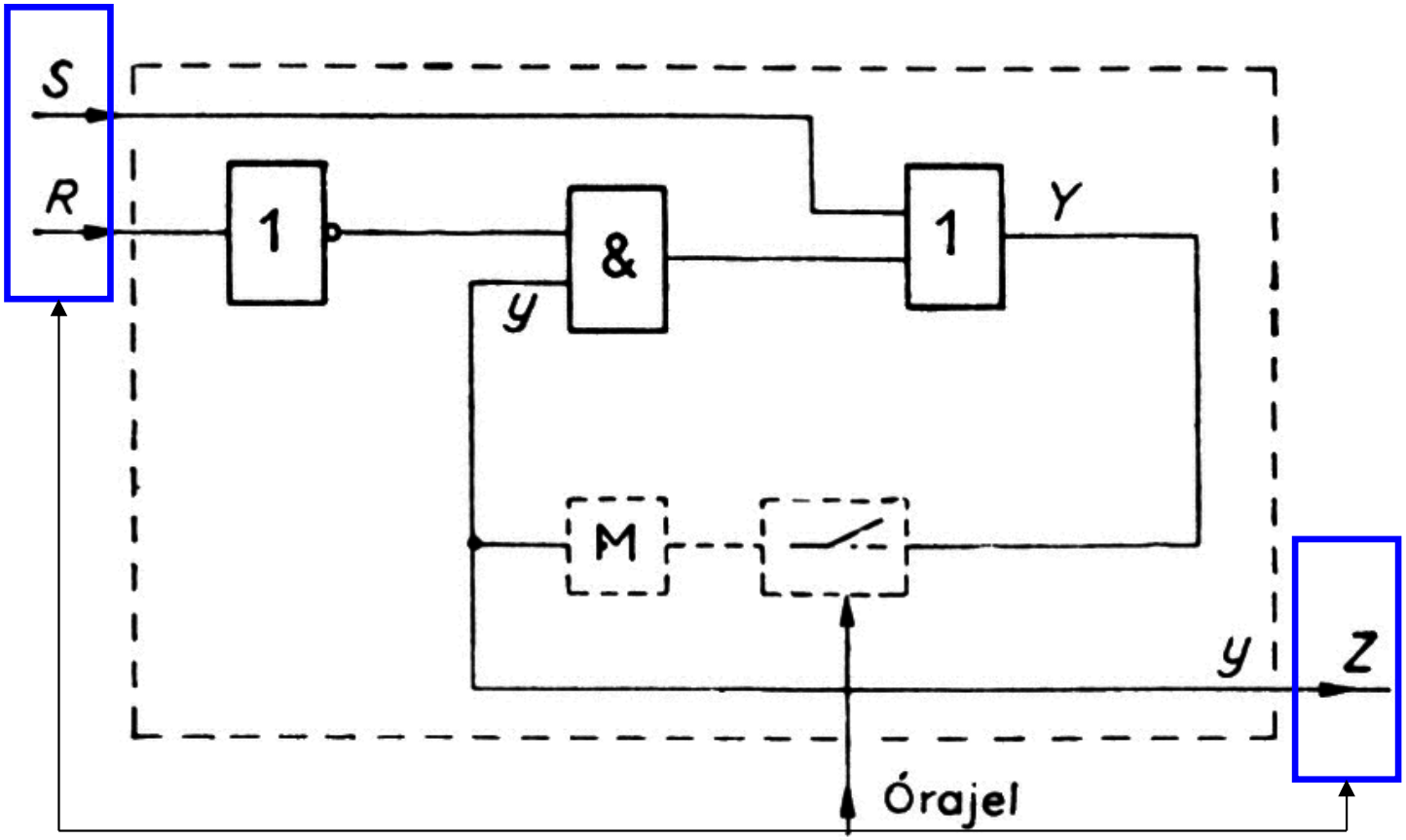
a.) Aszinkron S-R FF megvalósítása visszacsatolt K.H-al. tisztán NAND kapuk felhasználásával

Szokásos szakirodalmi ábrázolás



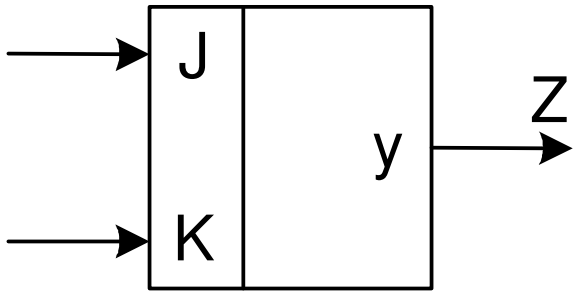
$$Z = Y = S + \overline{R}y = \overline{\overline{S} \cdot \overline{Ry}}$$

b.) Szinkron S-R FF megvalósítása órajel-vezérlésű kapcsoló segítségével



$$Z: y = M[Y] = S + \overline{R}y$$

2.) J-K flip-flop



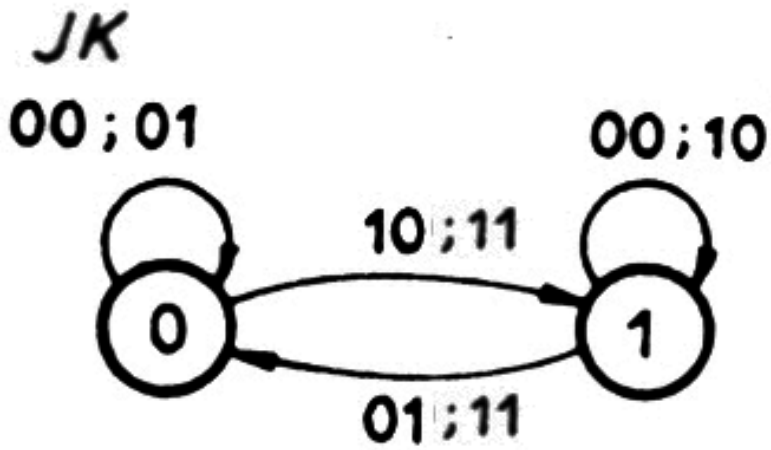
- **J-K tároló (S-R tárolóból származtatva)**
  - Csak szinkron módon értelmezhető
- **Működése:**
  - J=(S)= ‘1’ érték az FF állapotát (kimenetét) ‘1’ re állítja
  - K=(R)= ‘1’ érték az FF állapotát (kimenetét) ‘0’ –ra állítja/reseteli
  - J=K=‘0’ esetén az FF állapota (kimenete) nem változik, azaz tárol (Y=y)
  - **Változás**! J=K=‘1’ esetén az FF állapota is megengedett, de a J-K FF a mindenkori tárolt *állapot* értéket az *ellentettjére* változtatja.

J-K tároló: Szinkron működés

Ha J=‘1’,
írás akkor y:
0->1 lesz

JK / y	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Állapotgráf:



Tárol
(J=K=‘0’)

Ha K=‘1’,
törlés akkor
y: 1->0 lesz

Megengedett állapot (J=K=‘1’),
de ellentett állapot értékkel

(itt nem is jelölhető ki stabil
állapot – csak szinkron)

J-K tárolók: Karnaugh tábla és felépítés

Y

JK \ y	00	01	11	10
0	0 (0)	0 (1)	1 (3)	1 (2)
1	1 (4)	0 (5)	0 (7)	1 (6)

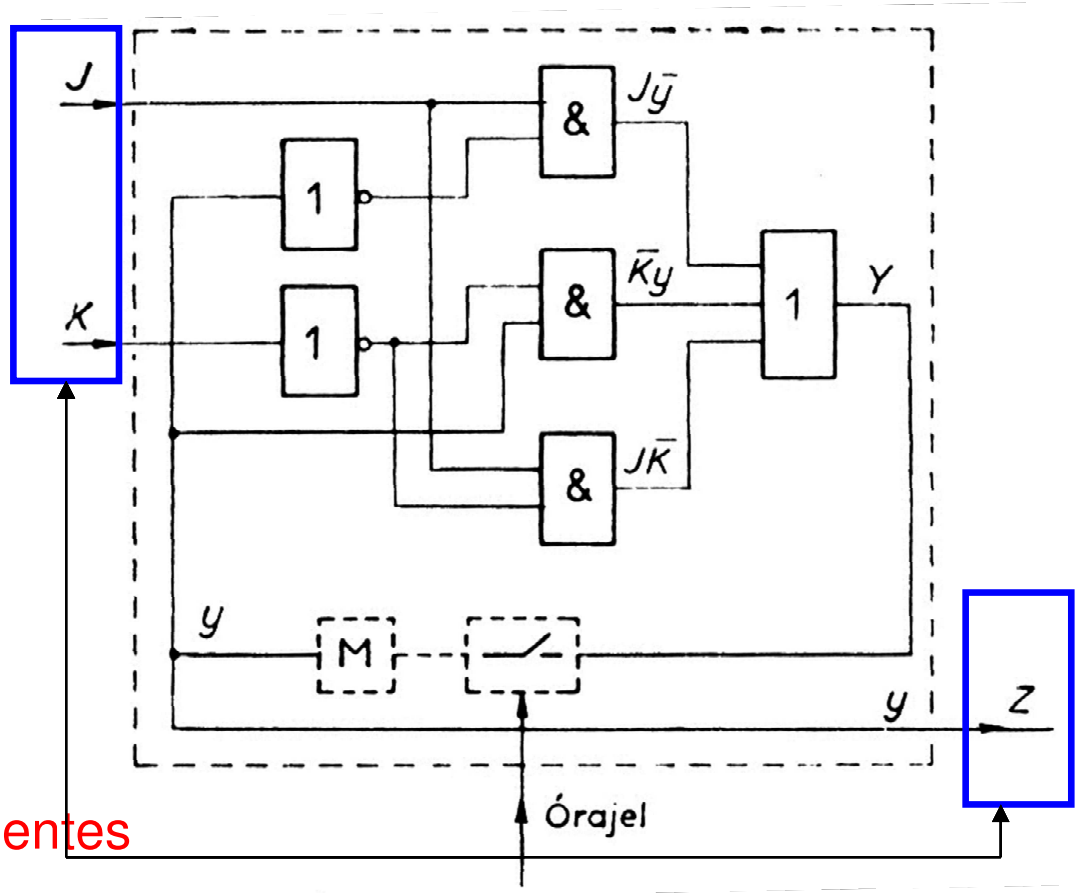
Egyszerűsített DNF alak
és elvi logikai rajz:

$$Y = f_y(J, K, y) =$$

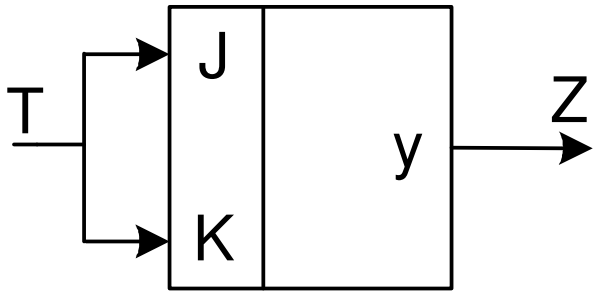
$$Y = J\bar{y} + \bar{K}y + J\bar{K}$$

Hazárdmentes (statikus) hálózathoz

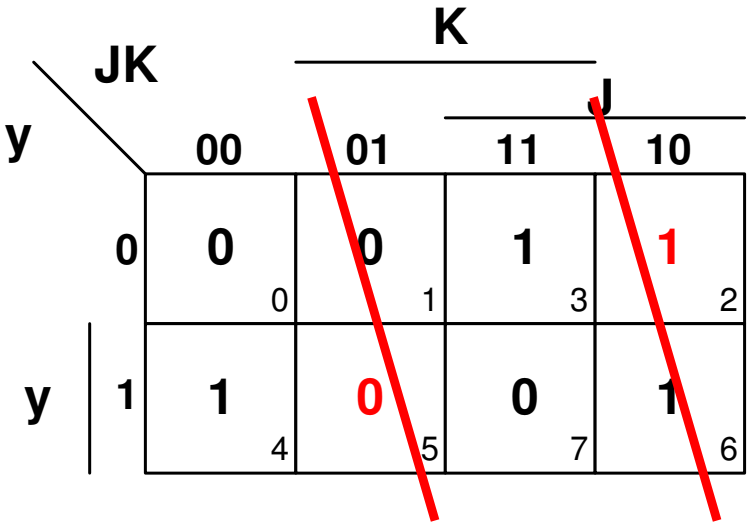
Szinkron J-K FF



3.) T flip-flop



- **T tároló (J-K tárolóból származtatva)**
  - **J=K ; J és K bemenetek összekötve**
  - Csak szinkron módon értelmezhető
- **Működése:**
  - Kizárólag JK=’11’ vagy ’00’ kombinációk megengedettek
  - `T='1' == (JK='11' )` T FF állapota az ellenkezőjére változik
  - `T='0' == (JK='00' )` T FF állapota nem változik (tárol)



T tároló: Szinkron működés

Szinkron: mivel T='1' esetén nem képzelhető el stabil állapot!

Állapottábla:

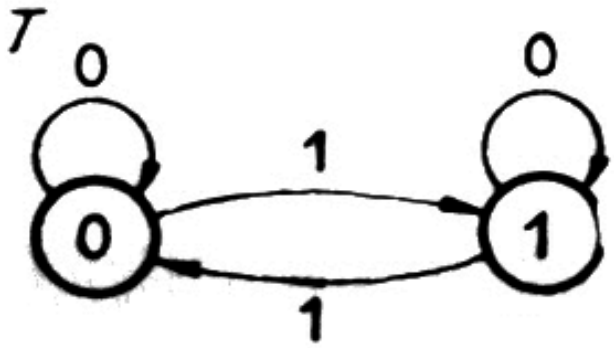
Y \ T	0	1
0	0	1
1	1	0

állapot (tárolási mód)

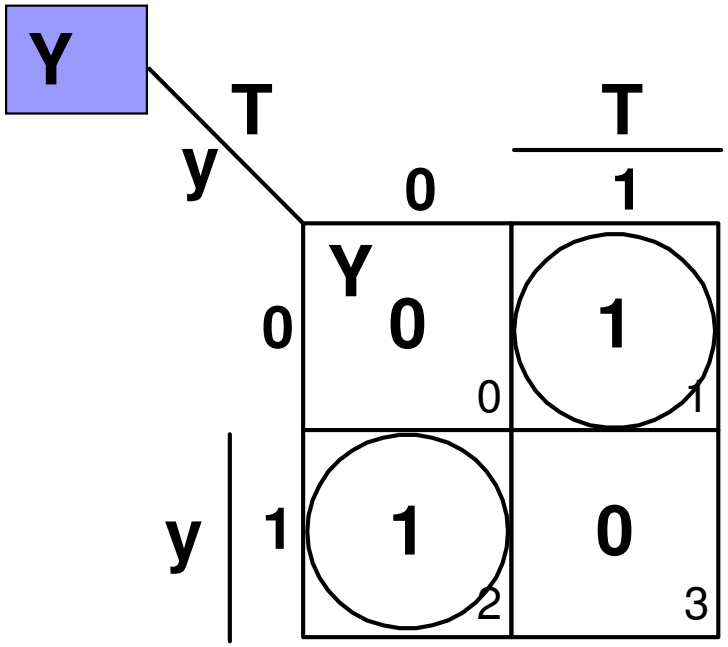
Ha T='1', negálás lesz ($\overline{y}$)

(toggle mód)

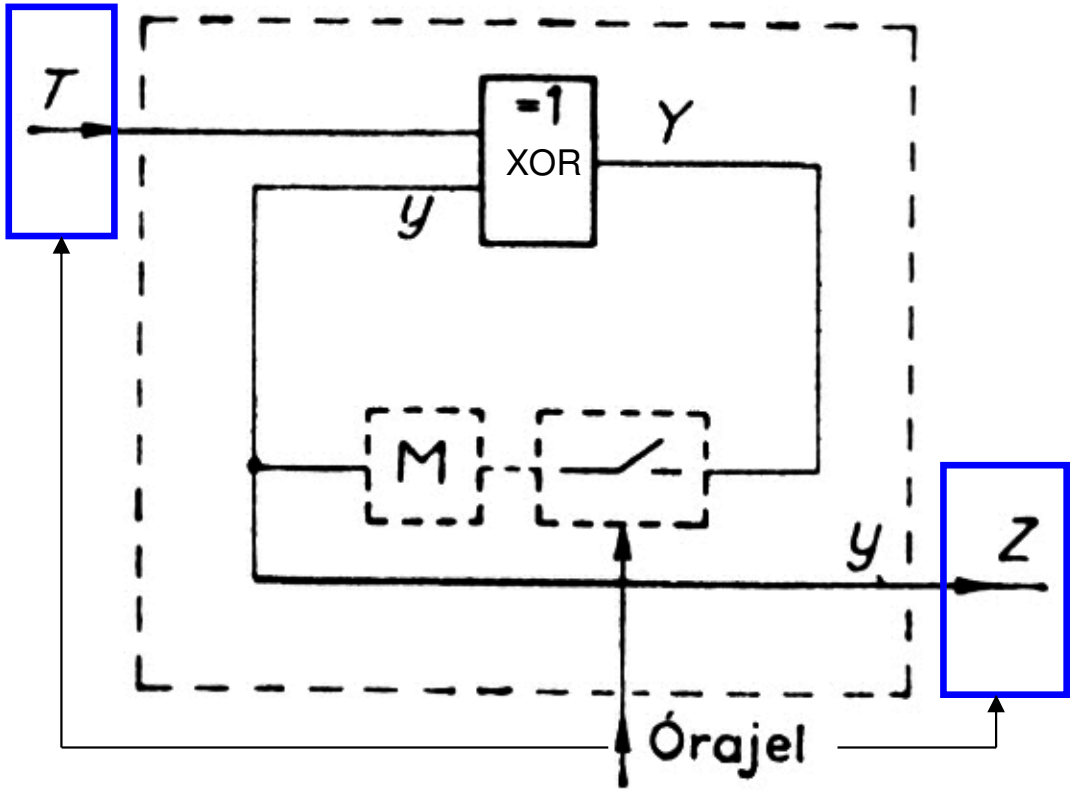
Állapotgráf:



T tároló: Karnaugh tábla és felépítés



Szinkron T FF

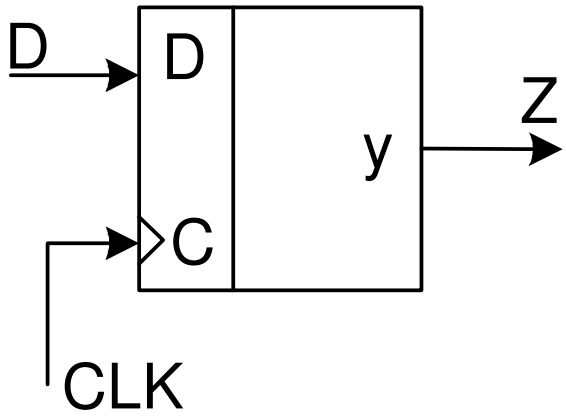


Egyszerűsített DNF alak
és elvi logikai rajz:

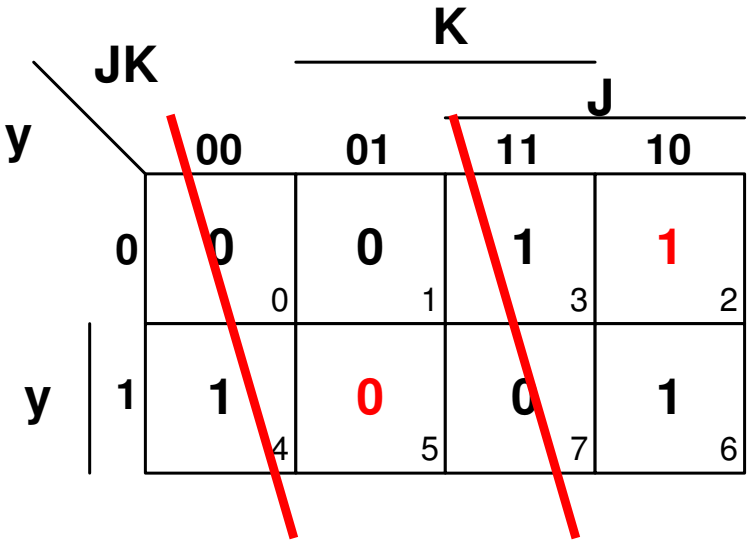
$$Y = f_y(T, y) =$$

$$Y = T\bar{y} + \bar{T}y = T \oplus y = // S_1^2(T, y)$$

4.) D flip-flop



- **D tároló (J-K tárolóból származtatva)**
  - **J≠K különböző bemenetek a megengedett D tároló bemenetek (’01’ v ’10’ kombinációk)**
  - **J=K bemenetek nem megengedettek!**
  - Csak szinkron módon értelmezhető (C)*
- **Működése:**
  - Kizárólag JK=’01’ vagy ’10’ kombinációk megengedettek
  - `D='0' == (JK='01')` D-FF állapota változik (‘0’-t tárol)
  - `D='1' == (JK='10')` D-FF állapota változik (‘1’-et tárol)
  - Tehát egy órajel ciklus ideig tároljuk a bemenetre érkezett értéket, változás a CLK élére történhet



D tároló: Szinkron* működés

***Állapottábla alapján lehetne akár aszinkronként** is értelmezni, mivel D='0' és D='1' bemenet esetén is lehet stabil állapot, DE nem oldaná meg az előírt logikai feladatot a bemeneti kombinációkra. Ezért **csak szinkronként alkalmazzuk különböző S.H.-ok visszacsatoló ágaiban**!*

y \ D	0	1
0	**0**	**1**
1	**0**	**1**

(Y)

Stabil állapotok (tárol)

Ha D='1' és y='0' volt, vagy fordítva, akkor felülírja a bemenet az állapotot

Állapottábla két sora azonos, azaz logikailag 1 állapota lehetne csak → K.H.-ként megvalósítható.

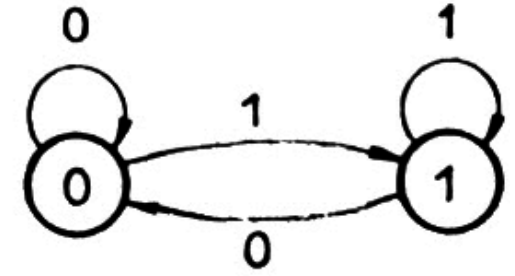
Másrészt:

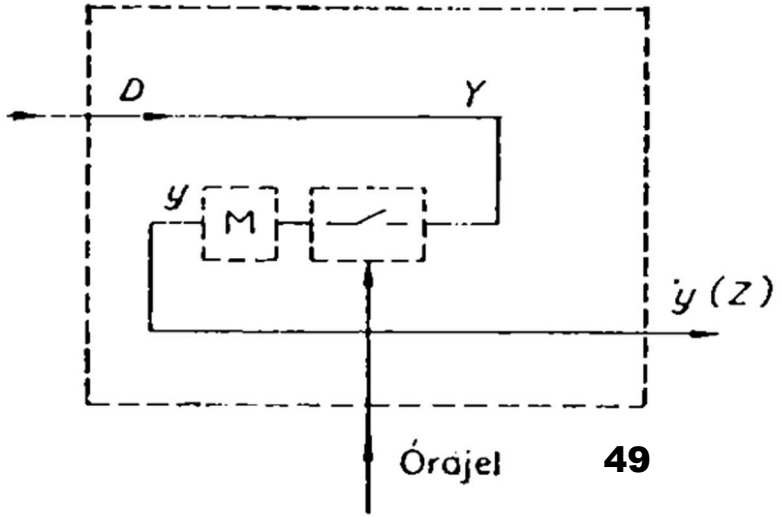
Y ≠ y

Y nem függ y-tól!

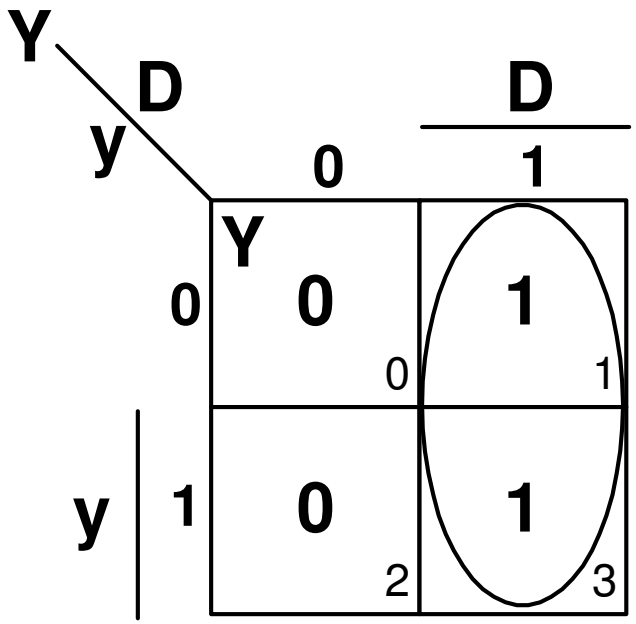
- ***Aszinkron: ---***
- ***Szinkron: CLK***

Állapotgráf:





D tároló: Karnaugh tábla és felépítés

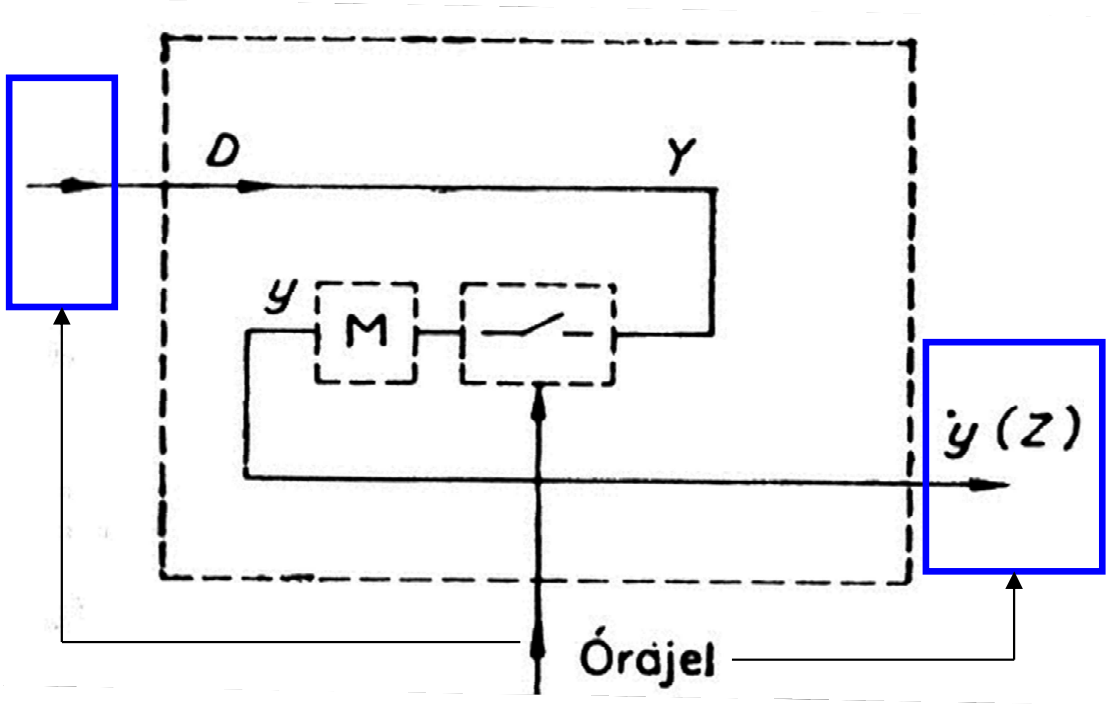


Egyszerűsített DNF alak
és elvi logikai rajz:

$$Y = f_y(D, y) =$$

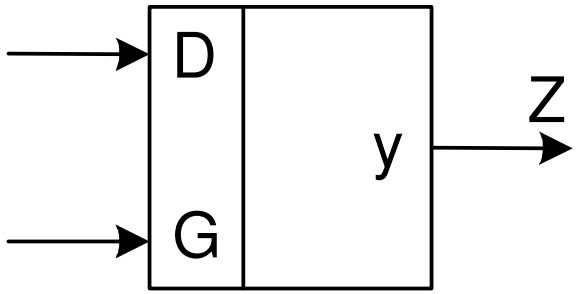
$$Y = D$$ ***y-tól független!!***

Szinkron D(C) FF



D-FF-et a sorrendi hálózatok visszacsatoló ágaiban feltételezett elemként lehet felhasználni!!

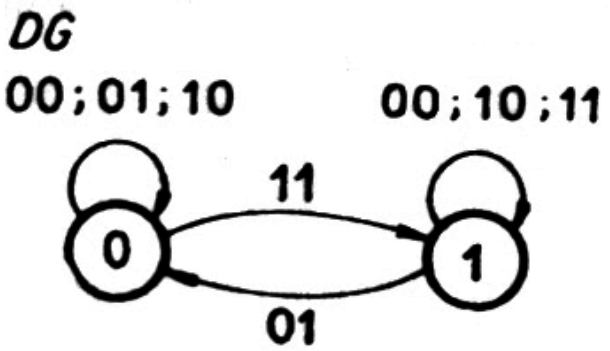
5.) D-G flip-flop



- **D-G tároló (Data-Gate)**
  - Szinkron/aszinkron módon is értelmezhető
- **Működése:**
  - `G='1'` időtartama alatt a D-G FF kimenete (állapota) követi a D bemenetre érkező jelváltozásokat, tehát `Y=D`
  - Ha viszont `G='0'`, akkor egy újabb `G='1'-ig` a D-G FF a D bemeneti értéktől függetlenül `megtartja` a korábbi bemeneti értéket.

D-G tárolók: Szinkron / aszinkron működés

Állapotgráf:



a.) Szinkron esetben:

Ha G=‘0’, akkor korábbi állapot (Y=y)

DG: y \	00	01	11	10
0	0	0	1	0
1	1	0	1	1

Ha G=‘1’, új állapot Y=D, ami lehet ‘0’ v ‘1’

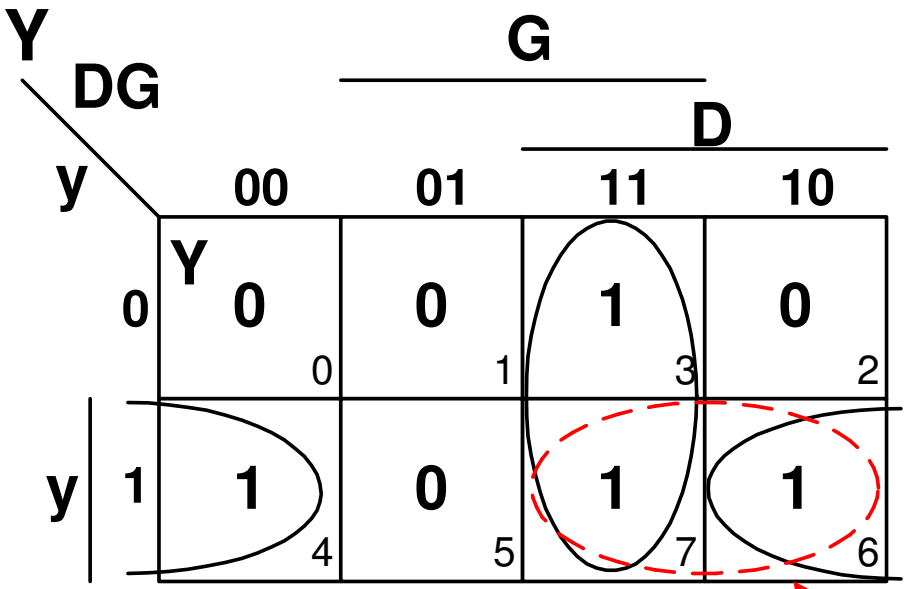
b.) Aszinkron esetben:

DG: y \	00	01	11	10
0	0	0	1	0
1	1	0	1	1

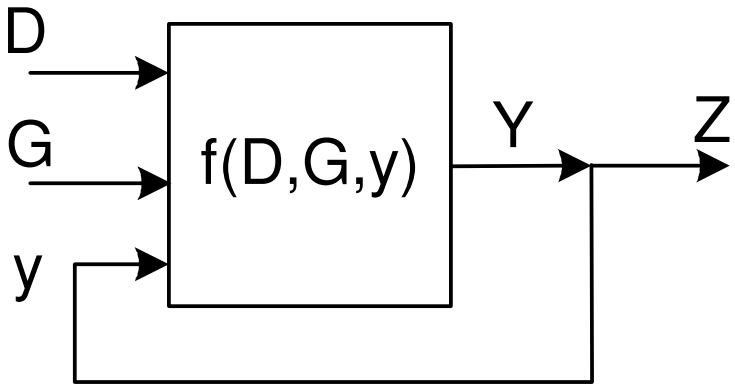
Stabil állapotok

D-G tárolók: Karnaugh tábla és felépítés

Aszinkron esetben:



Blokkdiagram: Visszacsatolt K.H.-al is megadható!



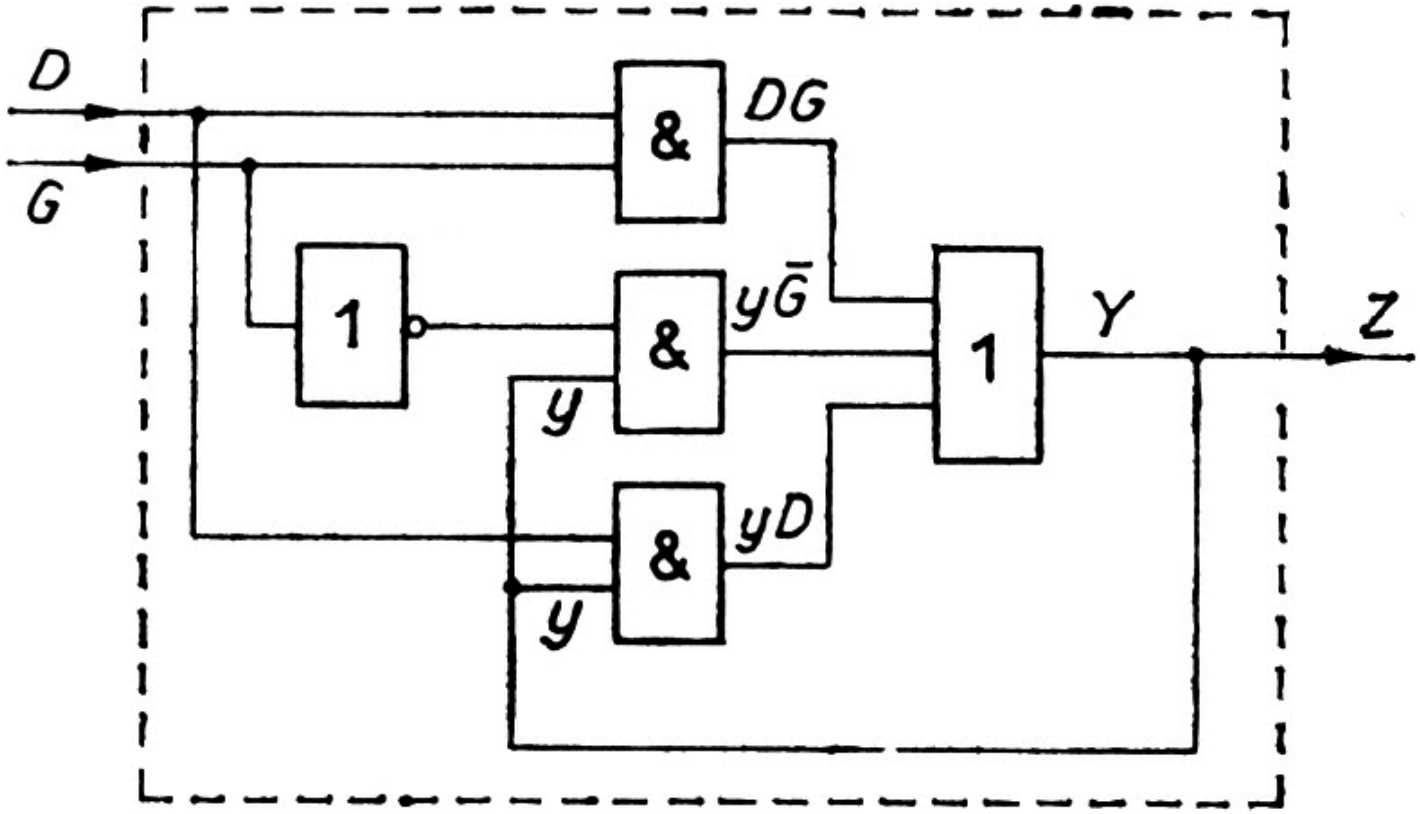
Egyszerűsített DNF alak

$$Y = f_y(D, G, y) =$$

$$Y = Z = DG + \overline{G}y + Dy$$

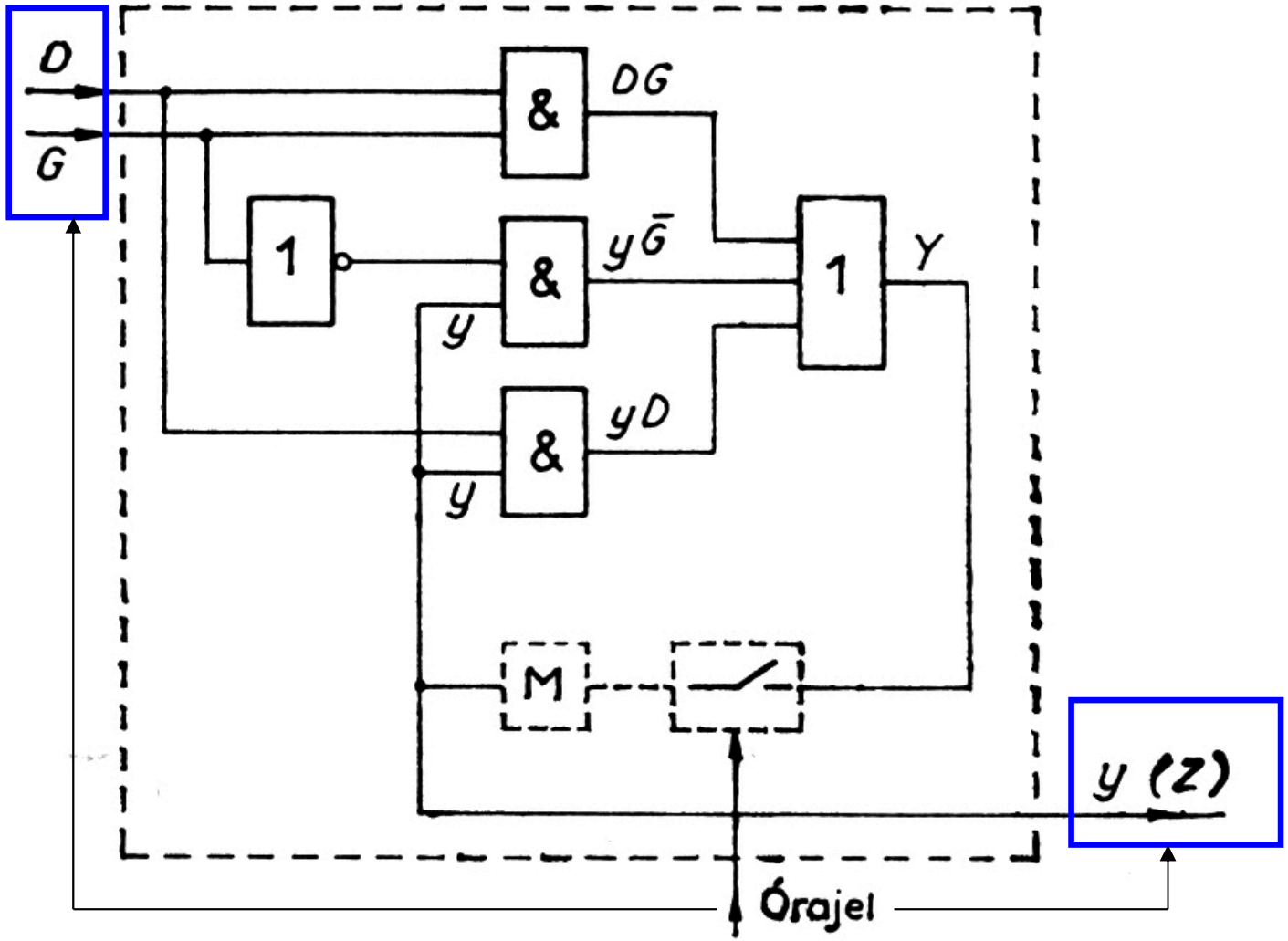
Hazárdmentes (statikus) hálózathoz

a.) Aszinkron D-G FF megvalósítása *visszacsatolt* kétszintű K.H-al. – elvi logikai rajz



$$Y = Z = DG + \overline{G}y + Dy$$

b.) Szinkron D-G FF megvalósítása *órajel-vezérlésű* kapcsoló segítségével



$$Z: y = M[Y] = DG + \overline{G}y + Dy$$

Összefoglaló **„paraméter” táblázat**: flip-flopok működése

Flip-flop megnevezése	Az fy(X,y) leképezést megvalósító logikai függvény	A szükséges bemeneti kombináció, ha yY =				Megjegyzés
		00	01	10	11	
S-R	$Y = S + \overline{R}y$	S R 0 -	10	01	- 0	aszinkron / szinkron
J-K	$Y = J\overline{y} + \overline{K}y(+J\overline{K})$	J K 0 -	1 -	-1	- 0	szinkron
T	$Y = \overline{T}y + T\overline{y}$	T 0	1	1	0	szinkron
D-G	$Y = DG + \overline{G}y(+Dy)$	D G 0 - - 0	11	01	- 0 1 -	aszinkron / szinkron
D-/C/	$Y = D$	D 0	1	0	1	szinkron

** Arató könyv: 180.old 3.31 táblázat*

Paraméter táblázat használata

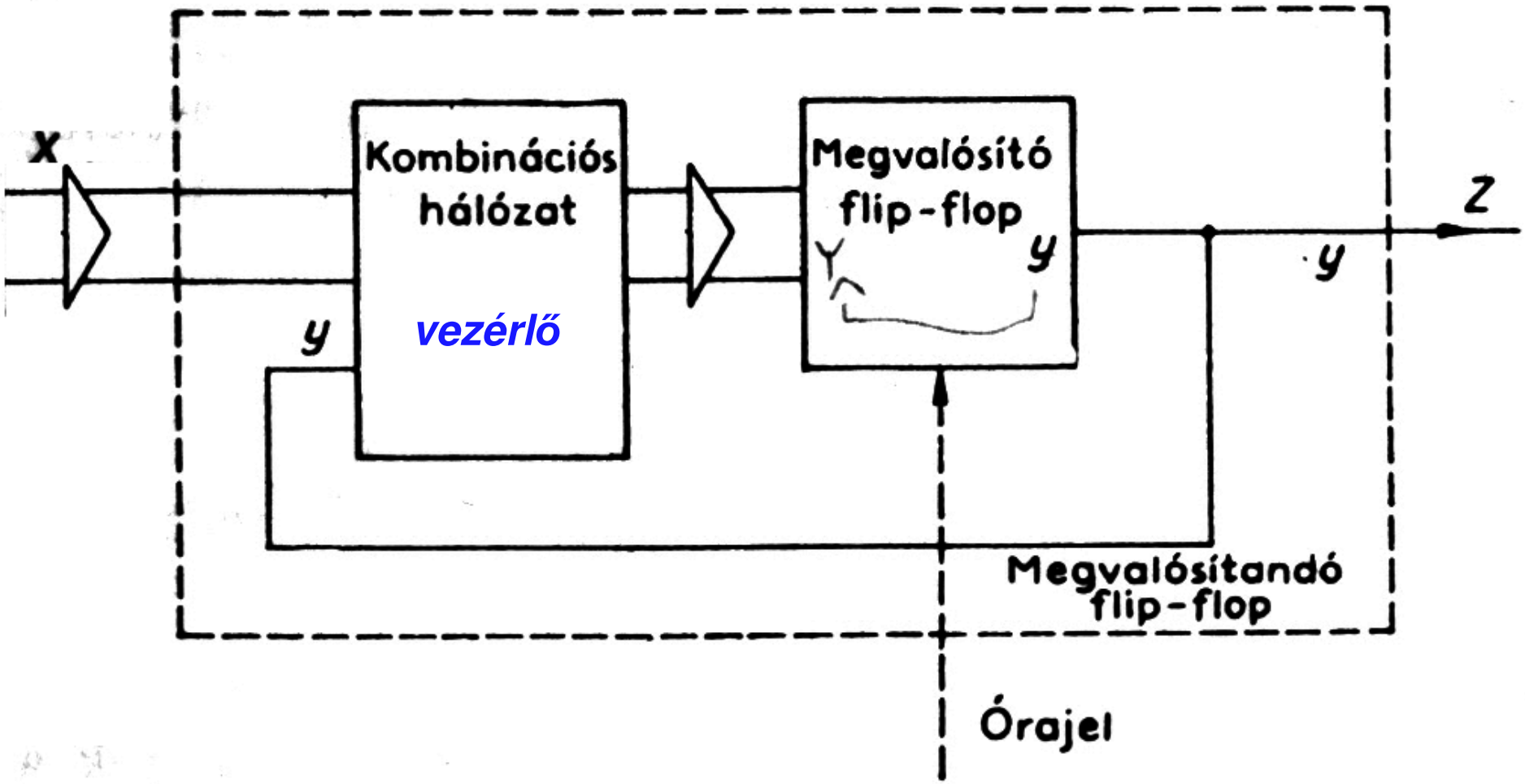
- FF kiválasztása
- Bemeneti kombinációkra az állapottábla `yY` kombinációinak megadása
- Ebből az állapotgráf felírása
- Paraméter táblázat használható! (ZH-n, vizsgán)

Különböző Flip-flopok felhasználása egymás megvalósítására

Különböző Flip-flopok felhasználása egymás megvalósítására

- A kiválasztott elemi tárolók bármelyikével az összes többi flip-flop megvalósítható.
- A *megvalósítandó* FF bemeneti kombinációiból egy K.H. állítja elő/mintegy vezérli a *megvalósító* (felhasznált) FF bemeneti kombinációit (lásd köv. ábra)
- **Fontos**: A sorrendi működést biztosító visszacsatolást nem az y visszavezetése képviseli, hanem a megvalósító / vezérelt FF-ban szükségszerűen meglévő (nem ábrázolt) Y→y belső összeköttetés, csatolás.

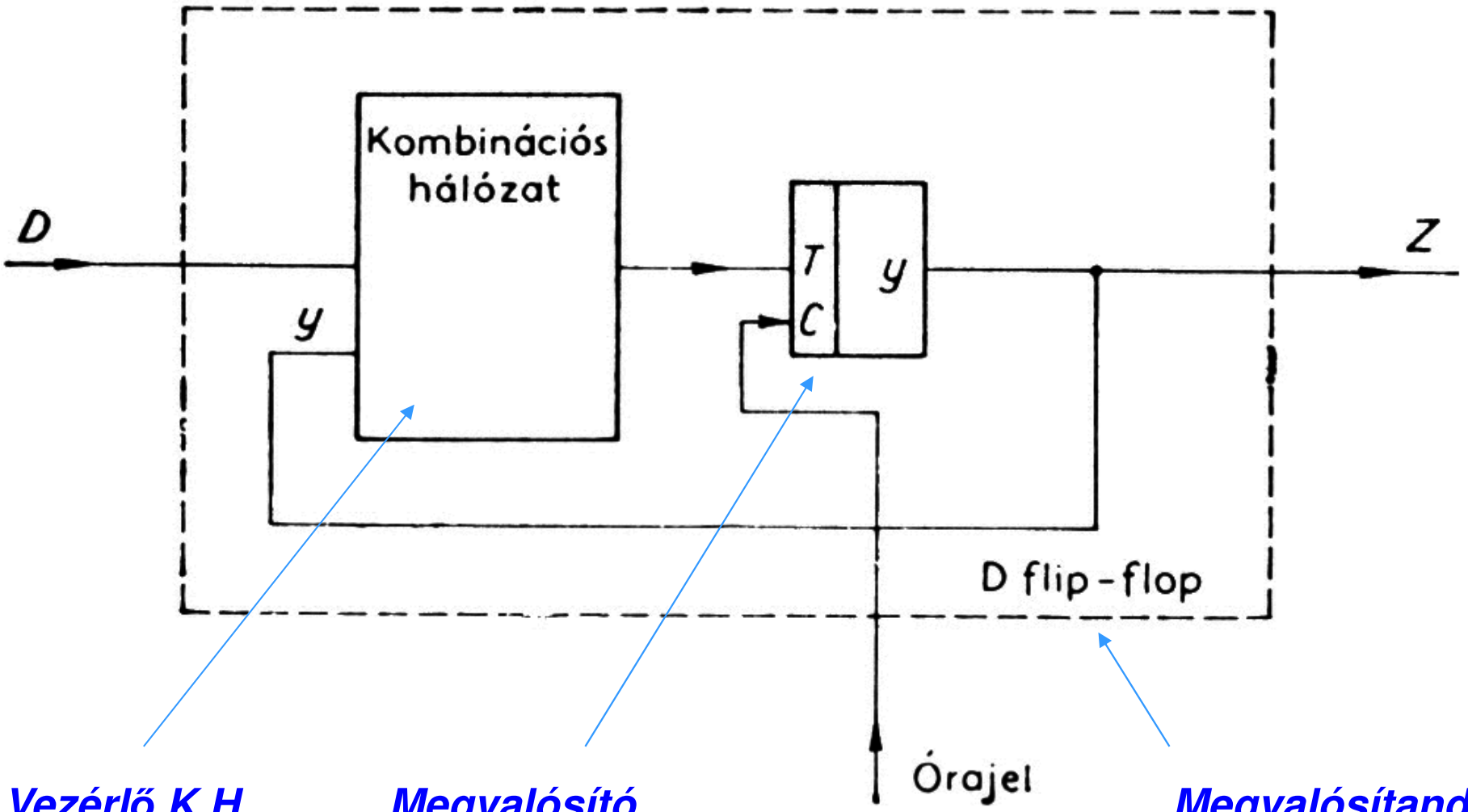
Flip-flop típusok egymás felhasználásával történő megvalósításának vázlata



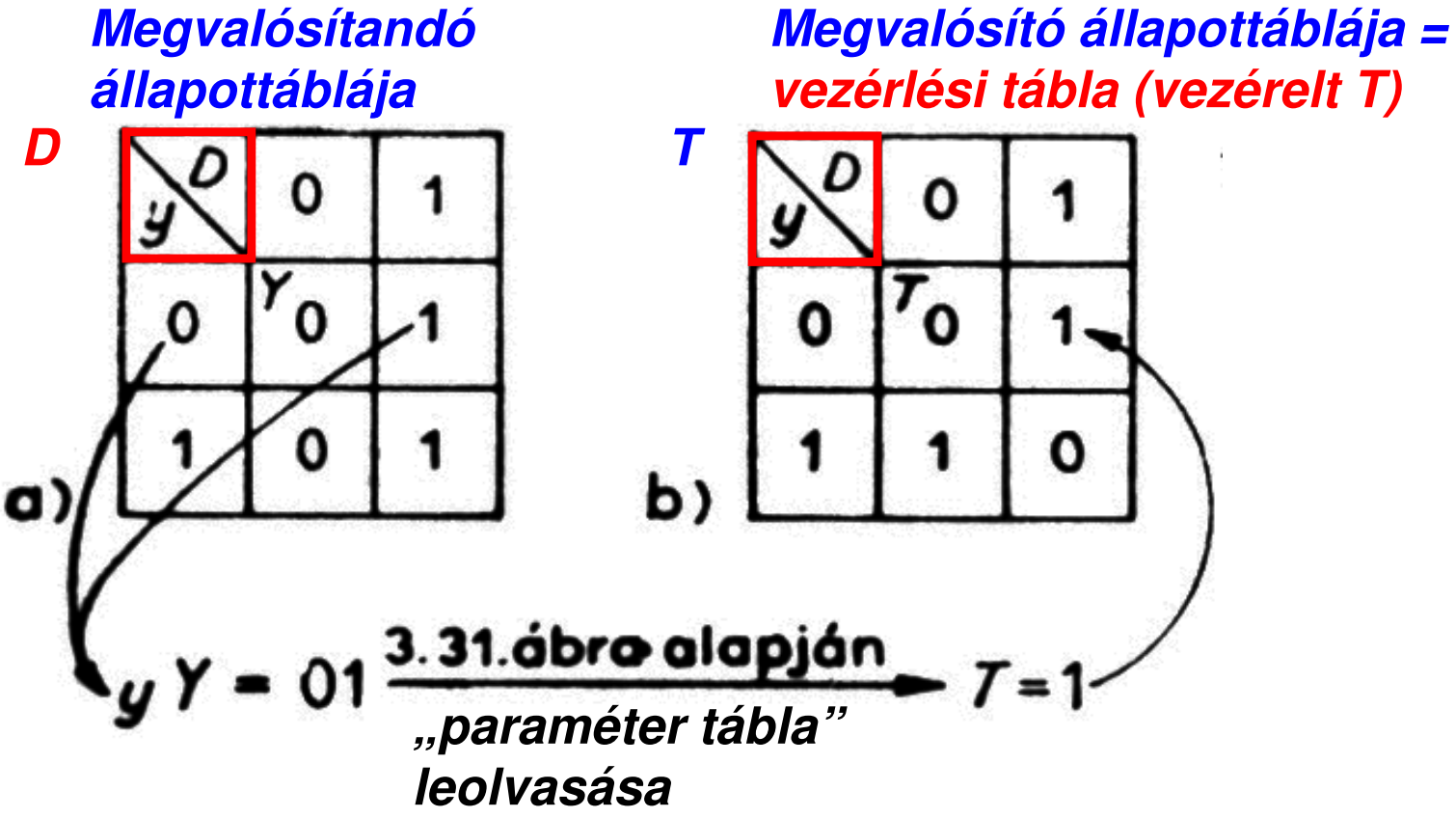
DEF: Vezérlési tábla

- **Vezérlési tábla** egy olyan speciális „állapottábla” amely a vezérlő K.H. kívánt működését írja le táblázatos formában: (pl. T -> D)
  - **Fejlécei** azonosak a **megvalósítandó** FF (pl. D*) állapottáblájának fejlécével,
  - Az egyes **celláiba/rovataiba** viszont nem Y értékeket írunk be, hanem azt a bemeneti kombinációt, amelyet a **megvalósító/vezérelt** FF-ra (pl. T*) kell juttatni ahhoz, hogy a megvalósítandó FF (pl. D*) állapottáblájának adott celljához tartozó Y állapot létrejöjjön.
    - *zárójelben az előző példa alapján kapott FF-okat írtam, természetesen *bármilyen*, a megvalósításhoz felhasznált FF-páros lehet
  - A vezérlési tábla a megvalósító FF Karnaugh tábláját fogja megadni → Minimalizáció

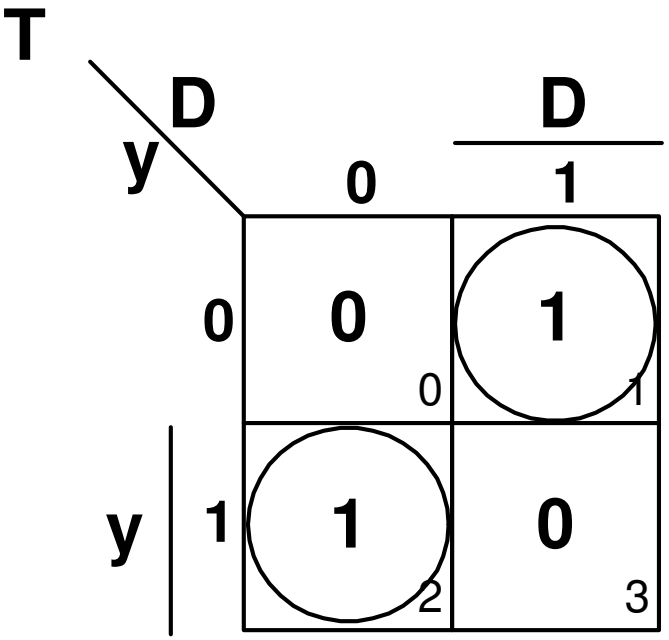
Példa 1.) D-FF ← T-FF-al történő megvalósítása – „rendszertechnikai vázlat”



Példa 1.) D-FF ← T-FF-al történő megvalósítása



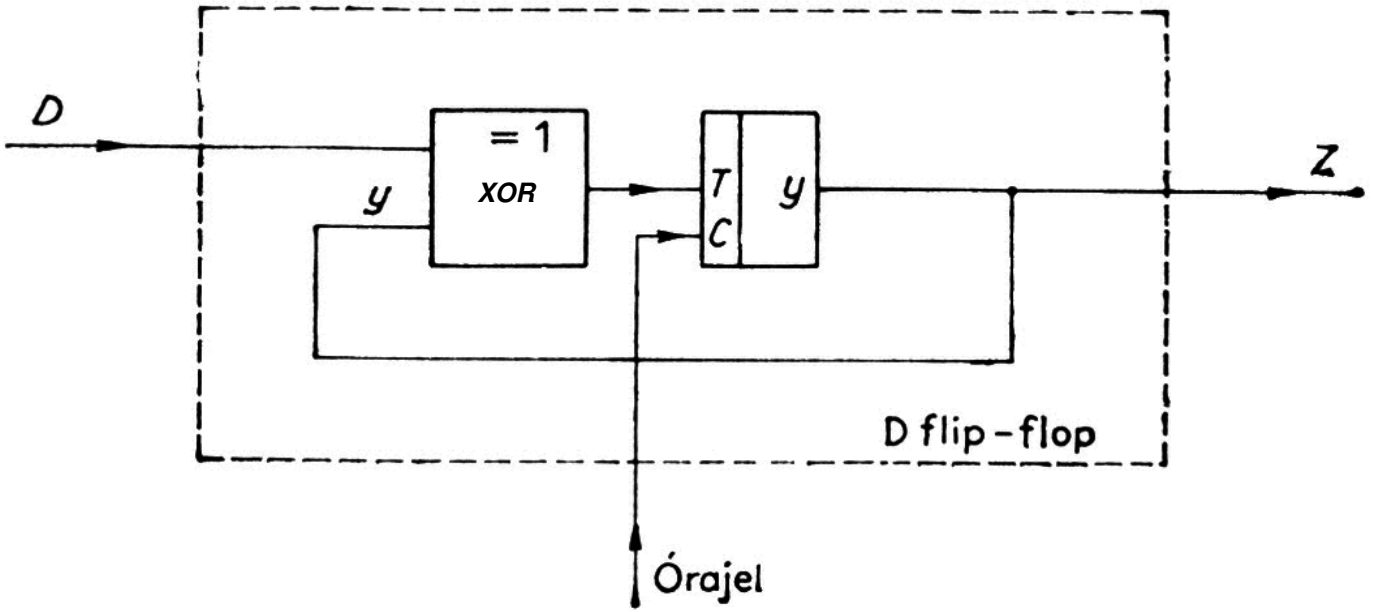
Példa 1.) D-FF ← T-FF-al történő megvalósítása



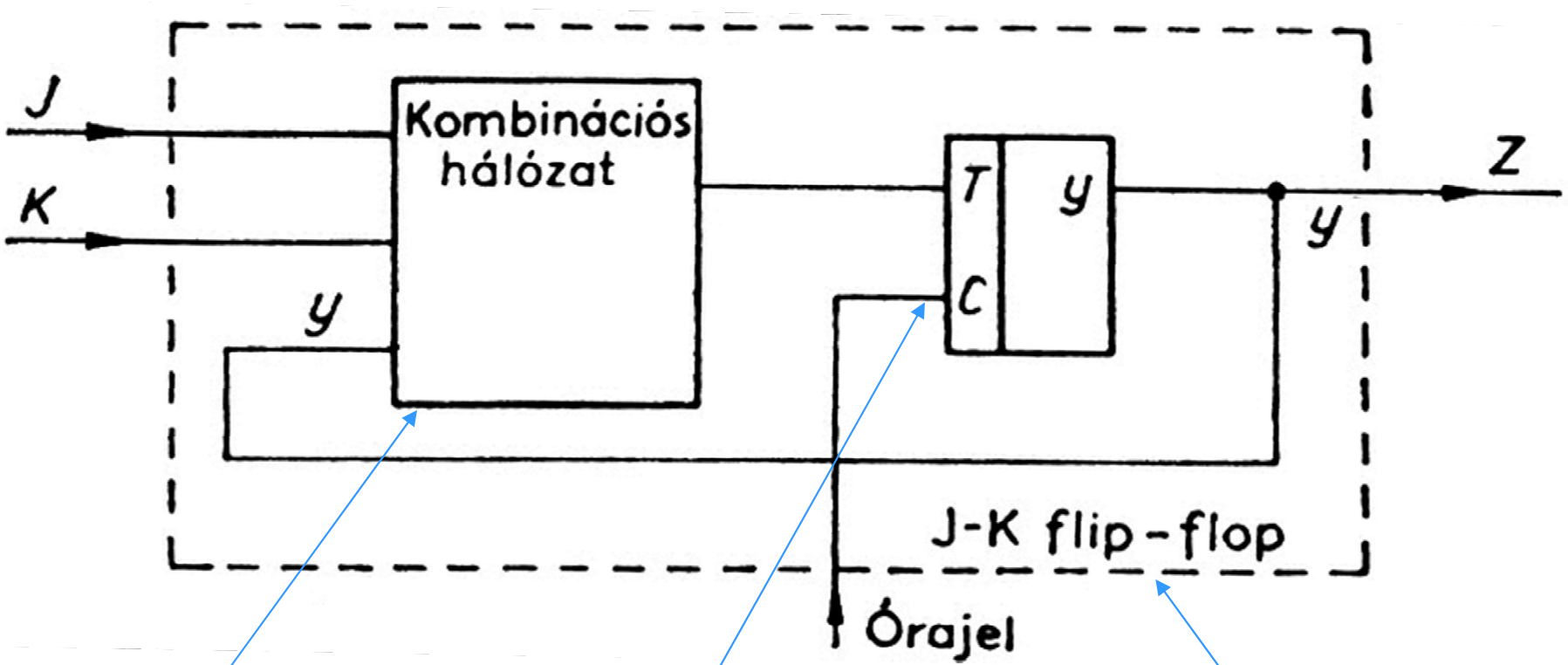
Egyszerűsített DNF alak
és elvi logikai rajz:

$$T = f(D, y) =$$

$$= D\bar{y} + \bar{D}y = D \oplus y$$



Példa 2.) J-K-FF ← T-FF-al történő megvalósítása – „rendszertechnikai vázlat”

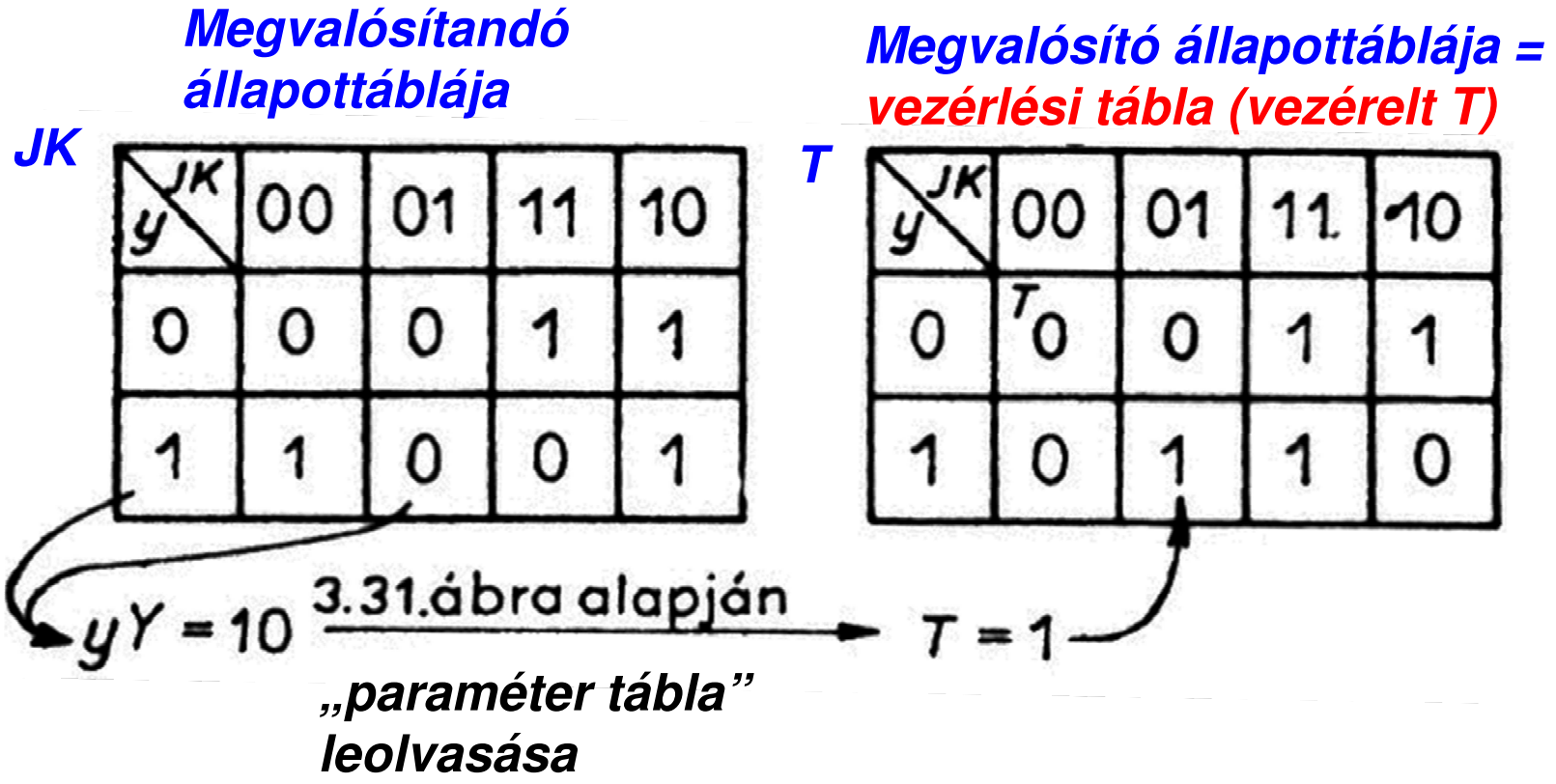


Vezérlő K.H

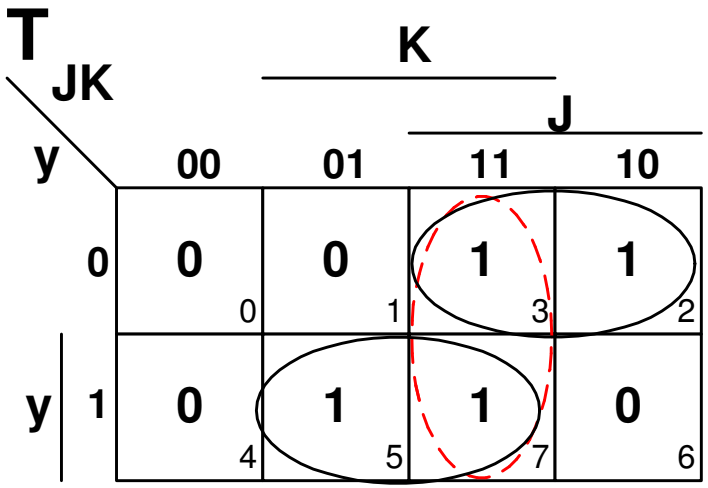
Megvalósító /vezérelt FF (pl. T)

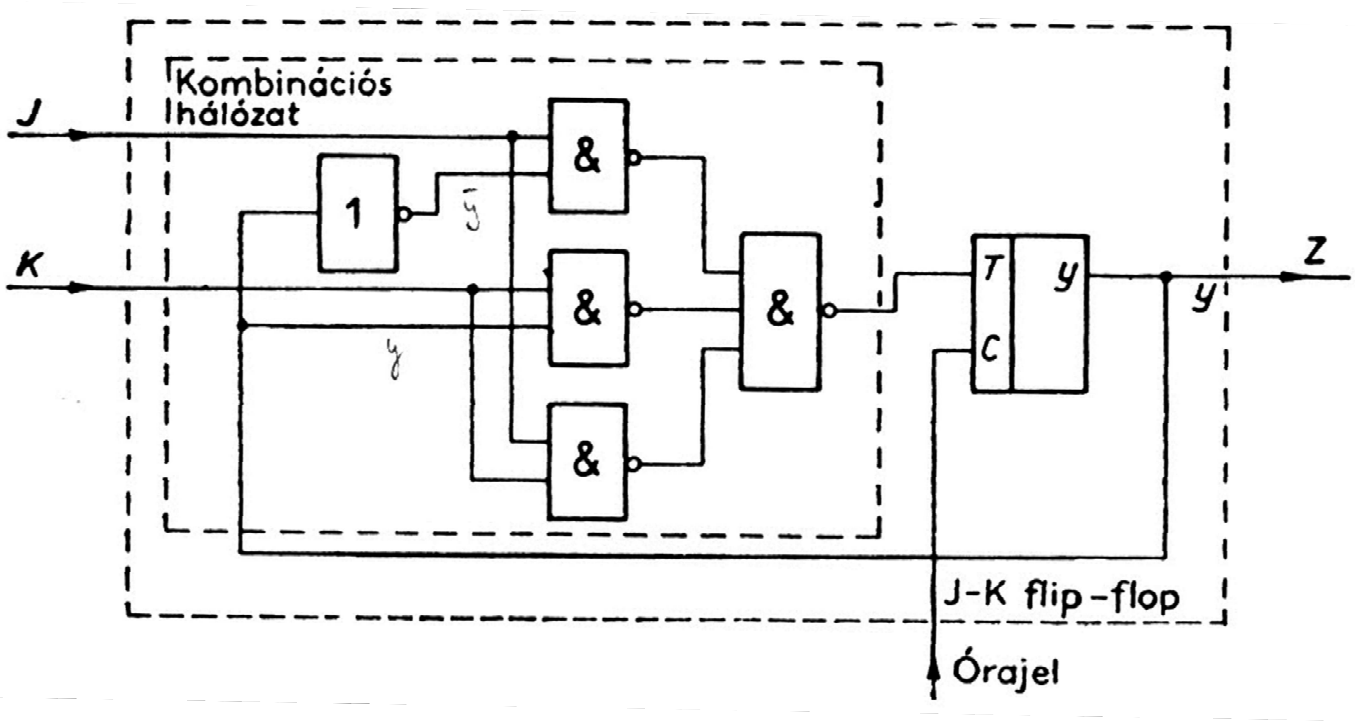
Megvalósítandó FF (pl. J-K)

Példa 2.) J-K-FF ← T-FF-al történő megvalósítása



Példa 2.) J-K-FF ← T-FF-al történő megvalósítása



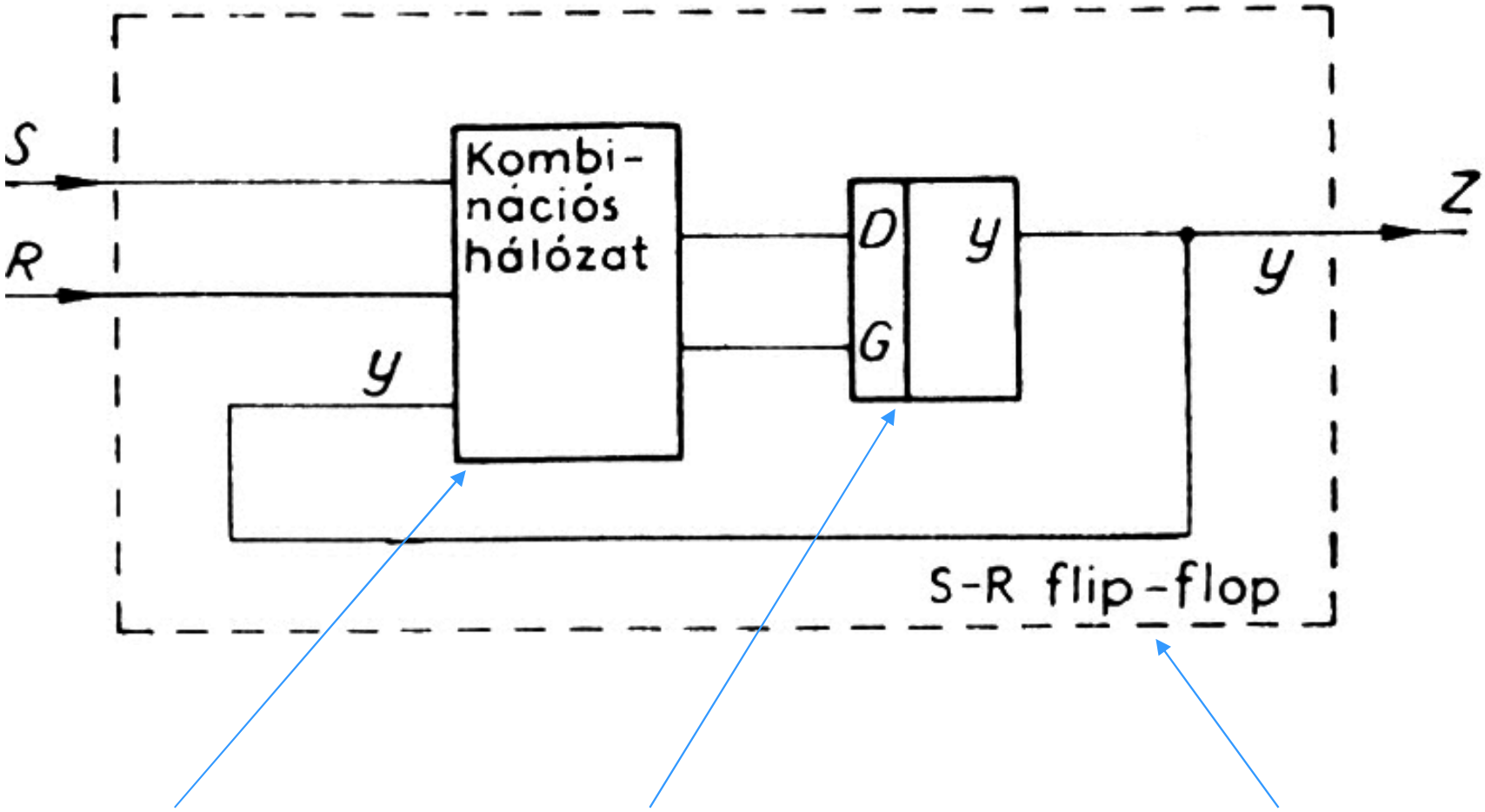


Egyszerűsített DNF alak és elvi logikai rajz:

$$T = f(J, K, y) =$$

$$= J\bar{y} + Ky + (JK)$$

Példa 3.) S-R FF ← D-G FF-al történő megvalósítása – „rendszertechnikai vázlat”



Vezérlő K.H

Megvalósító /vezérelt FF (pl. D-G)

Megvalósítandó FF (pl. S-R)

Példa 3.) S-R FF ← D-G FF-al történő megvalósítása

Megvalósítandó állapottáblája

S-R

y \ SR	00	01	11	10
0	Y 0	0	—	1
1	1	0	—	1

Megvalósító állapottáblája = vezérlési tábla (vezérelt D-G)

D-G

y \ SR	00	01	11	10
0	DG 0 – – 0	0 – – 0	—	11
1	– 0 1 –	01	—	– 0 1 –

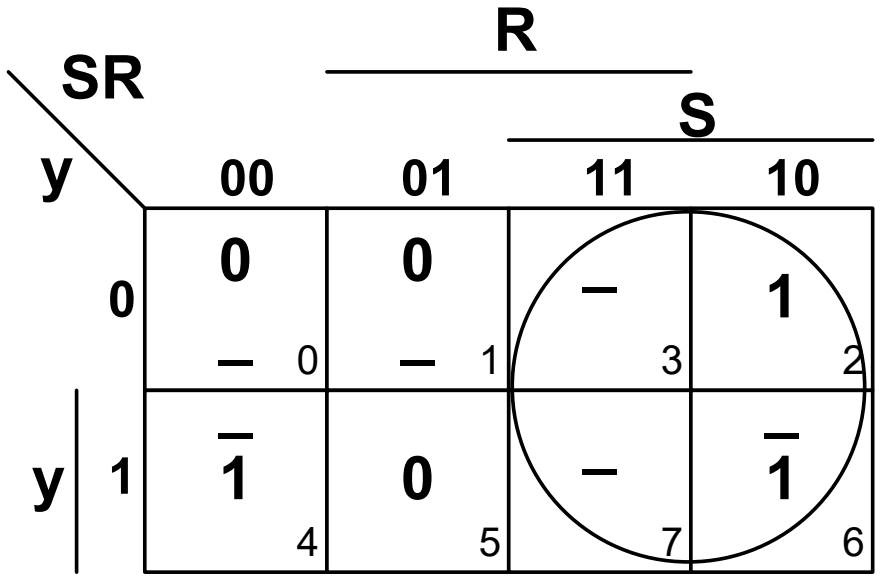
$yY = 11$ — 3.31 ábra alapján → $DG = \begin{matrix} -0 \\ 1- \end{matrix}$

„paraméter tábla” leolvasása

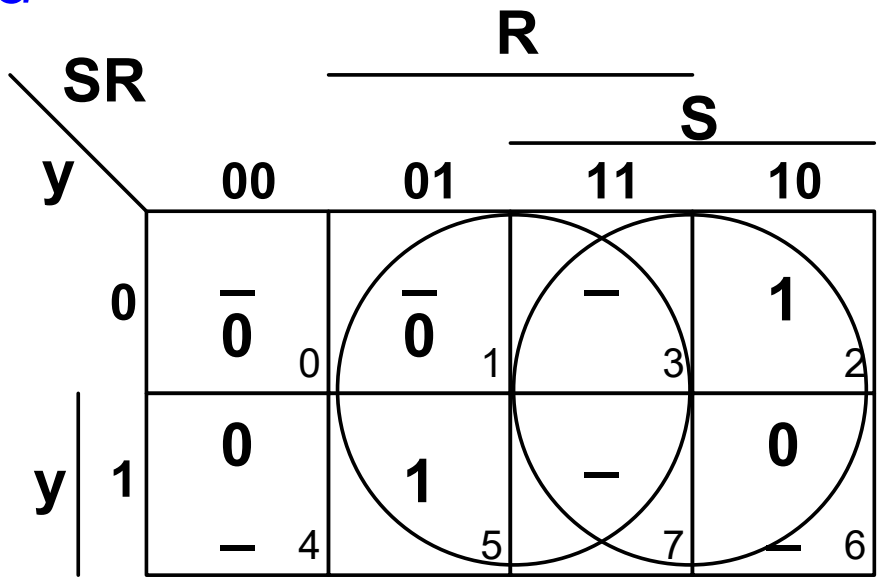
Példa 3.) S-R FF ← D-G FF-al történő megvalósítása – Karnaugh táblák

külön D és G szerint felírva

D



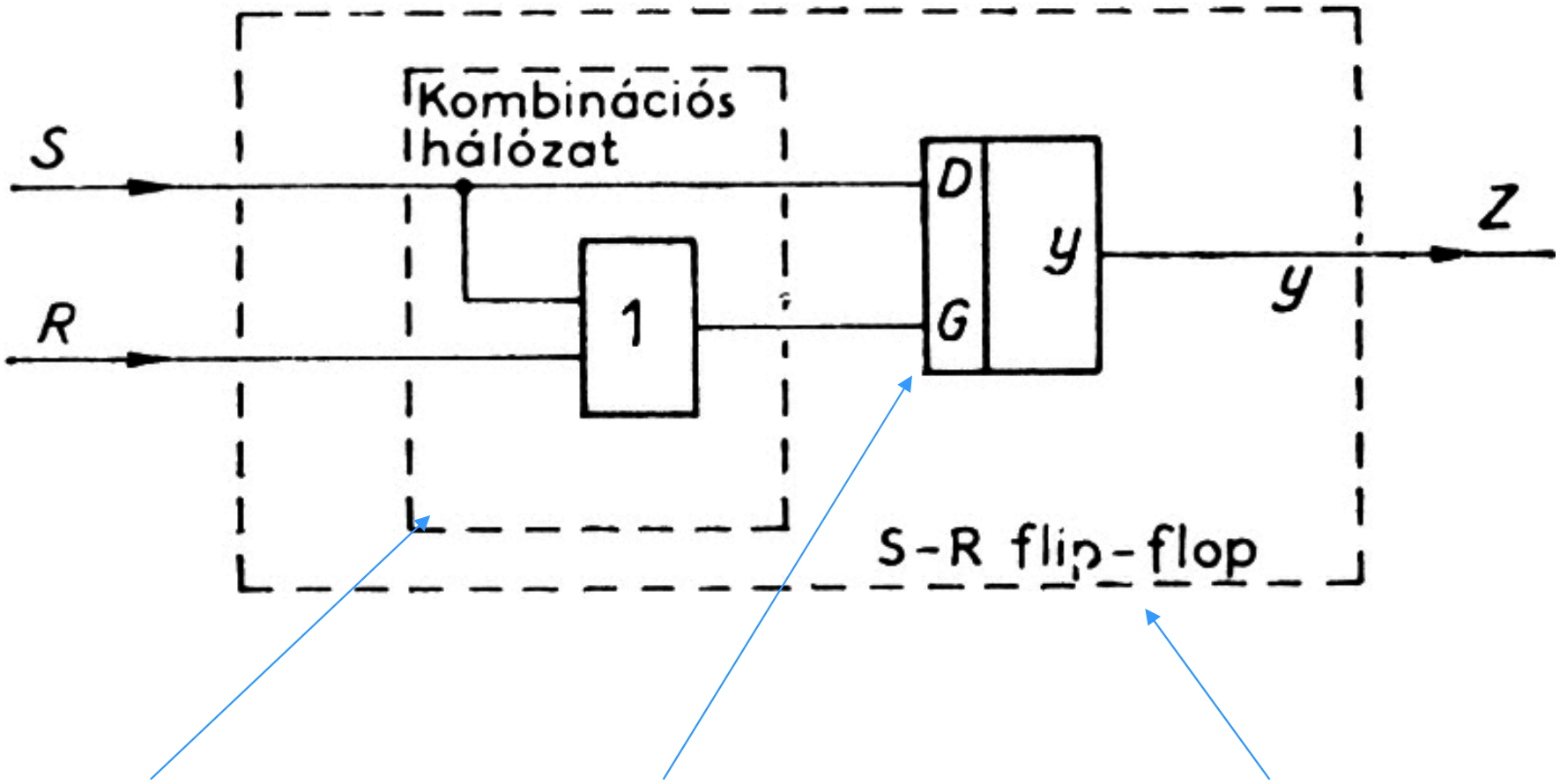
G



Egyszerűsített DNF alak: $D = S$ $G = S + R$

! Leolvasás D-G FF esetén: legegyszerűbb alak meghatározása próbálgatással, oly módon, hogy a D és G Karnaugh táblák **minden egyes cellájában/rovatban haladva egyaránt vagy csak alsó / vagy csak felső bejegyzések szerint végezzük az összevonást! (A következő cellában szintén vagy csak alsó/felsőt választunk!)**

Példa 3.) S-R FF ← D-G FF-al történő megvalósítása – „elvi logikai rajz”

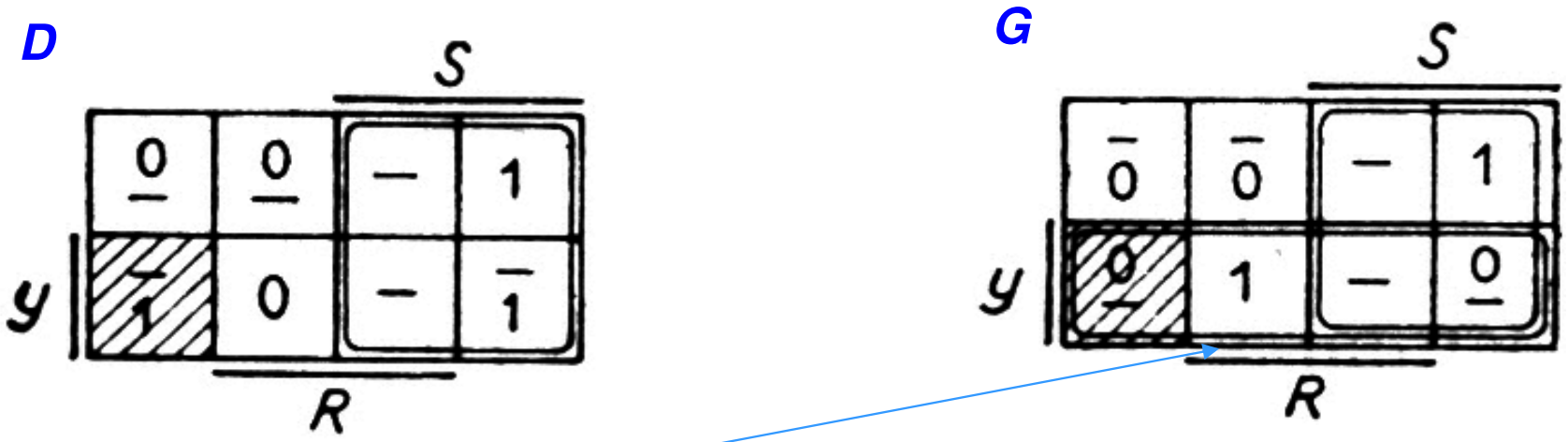


Vezérlő K.H

Megvalósító /vezérelt FF (pl. D-G)

Megvalósítandó FF (pl. S-R)

Megj: példa hibás összevonásra a D-G tároló esetén (előző példa)



Hiba: *Ha G = y szerint vonnánk össze, és alsó dont'care-t választanánk 1-esnek a hurok képzéséhez,*

ekkor viszont az S táblán is csak az alsó 1-est választhatnánk ki és vonhatnánk össze (nem pedig a felső dont'care-t '0'-nak választva).