Digitális technika

XIV.

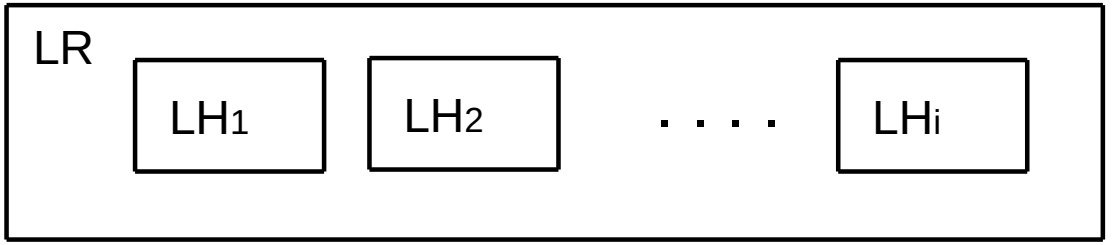
Vezérlő egységek tervezése

Sorrendi hálózatok tervezése II. ***

14.1. Vezérlő egységek

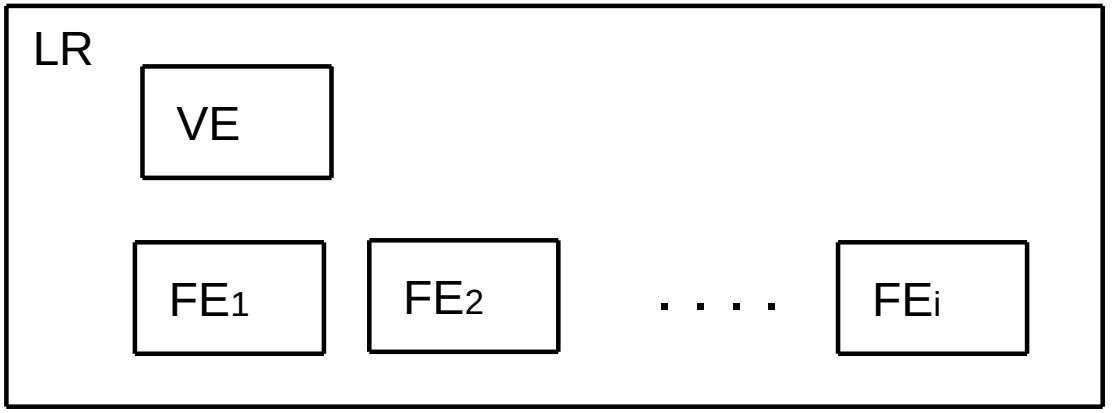
1. Logikai rendszer

- bonyolult logikai feladat esetén nem túl jó megoldás egyetlen bonyolult logikai hálózatot tervezni !
  → célszerű a feladatot felbontani rész feladatokra →
  → több, egyszerű logikai hálózat (LH) együttműködő rendszerét kell kialakítani →
  → logikai rendszer (LR)



- a részfeladatokat, meghatározott funkciókat ellátó logikai hálózatok → funkcionális egységek (FE)

- a funkcionális egységek működését össze kell hangolni, ütemezni kell !! → ezt a speciális feladatot látja el egy speciális funkcionális egység, a vezérlő egység (VE)



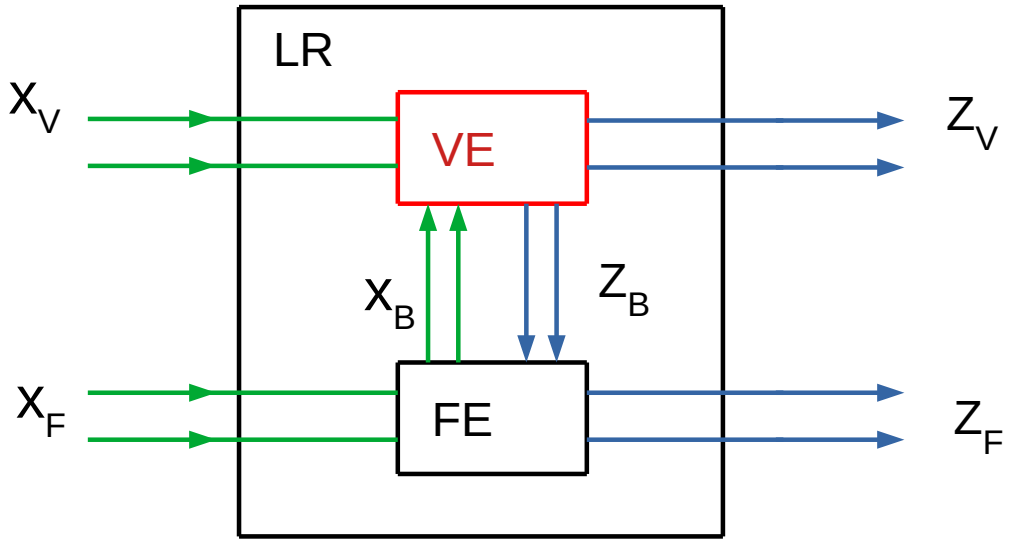
-minden funkcionális egységek szintén bonyolult lehet → felbonthatók szintén kisebb funkcionális egységekre ! →
→ ez a felosztás több szinten (mélységben) folytatható

14.1. Vezérlő egységek

2. Vezérlőegység

- bonyolult szinkron sorrendi hálózat

- vezérlő egység (VE) bemeneti, kimeneti jelei →



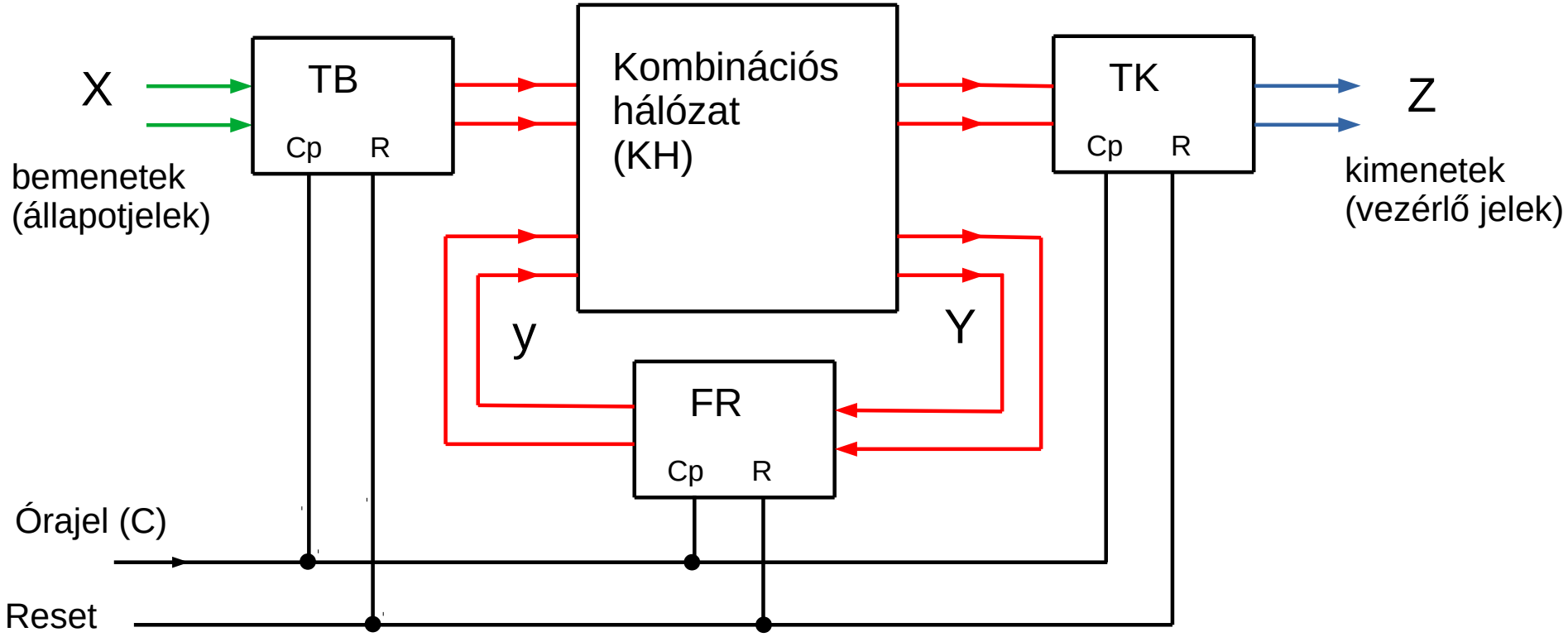
Vezérlő egység típusai

1. huzalozott (hardveres) → gyors, de bonyolult, rugalmatlan (nem módosítható egyszerűen) lehet → a, fázisregiszteres vagy b, számláló rendszerű

2. mikroprogramozott → rugalmas, könnyen módosítható, de lassabb program ROM-ban

14.2. Fázisregiszteres vezérlő

1. általános felépítése



TB – bemeneti időzítők (D tárolók)

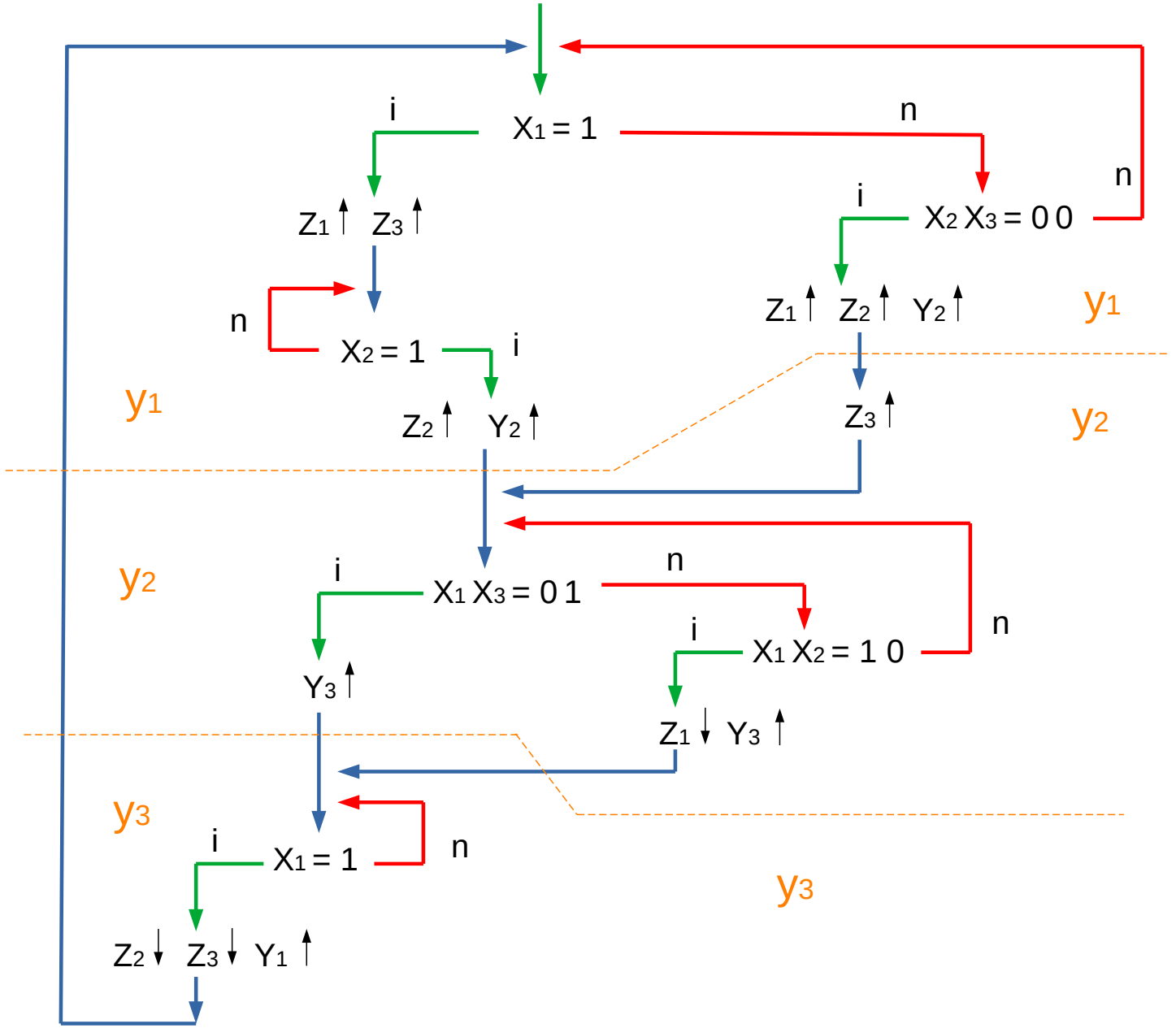
TK – kimeneti időzítők (SR tárolók)

FR – fázisregiszterek → állapot tárolók (reteszelt D tárolók)
ha kevés állapot → N-ből 1 kód
ha sok állapot → kódolt → dekódoló egység is kell !

Mivel általában sok bemeneti jel, kimeneti jel és állapot van →
→ más felépítés és tervezési módszerek mint egy egyszerű sorrendi hálózat esetén → folyamatábra

14.2. Fázisregiszteres vezérlő

2. működés leírása folyamatábrával



$Z_1:1 = y_1*(X_1 + \overline{X}_2*\overline{X}_3)$

$Z_1:0 = y_1 * X_1 * \overline{X}_2$

$Z_2:1 = y_1*(X_2 + \overline{X}_2*\overline{X}_3)$

$Z_2:0 = y_3 * X_1$

$Z_3:1 = y_2 + y_1*X_1$

$Z_3:0 = y_3 * X_1$

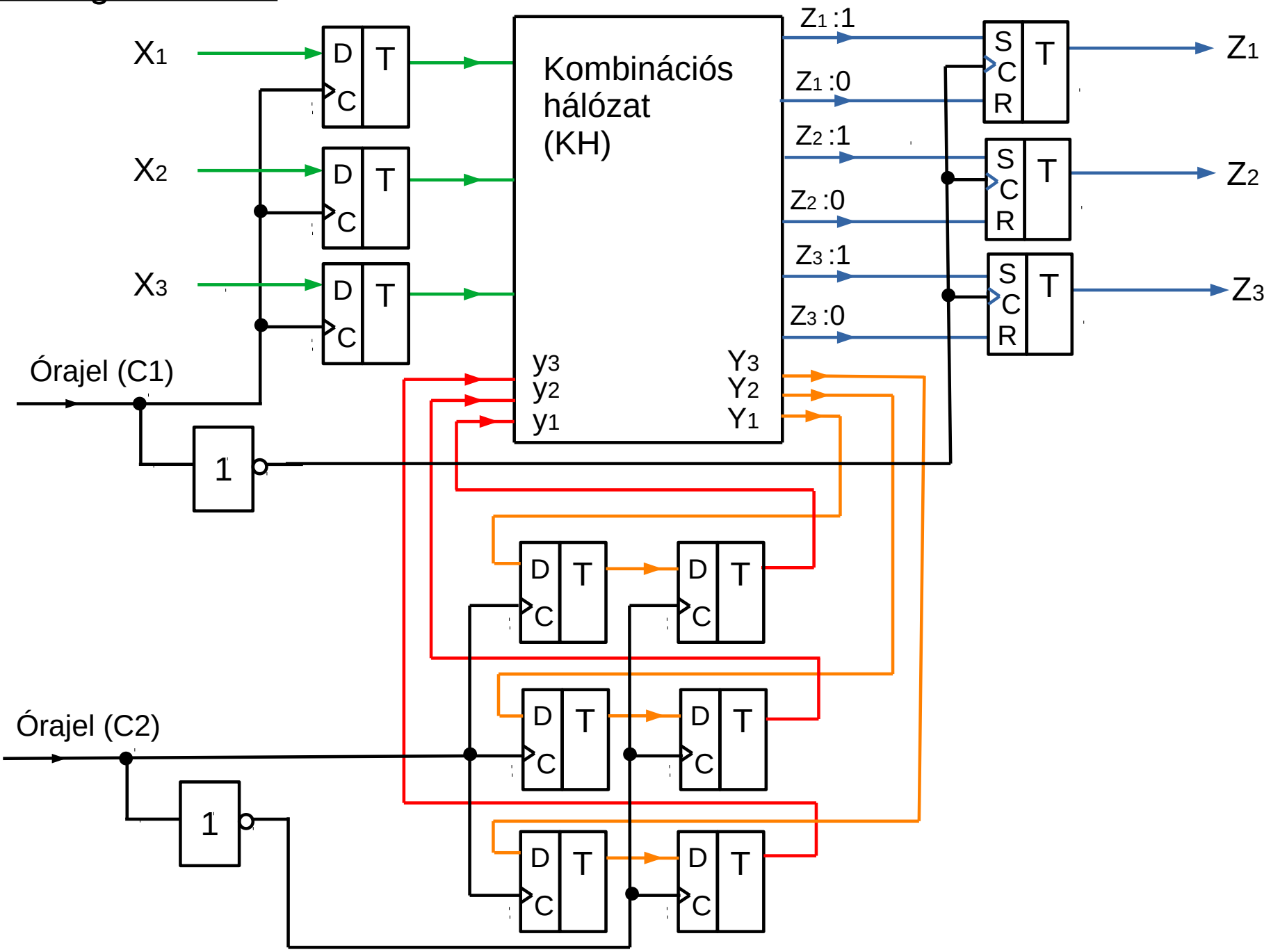
$Y_1 = y_3 * X_1$

$Y_2 = y_1 * (X_2 + \overline{X}_2*\overline{X}_3)$

$Y_3 = y_2 * (\overline{X}_1*X_3 + X_1*\overline{X}_2)$

14.2. Fázisregiszteres vezérlő

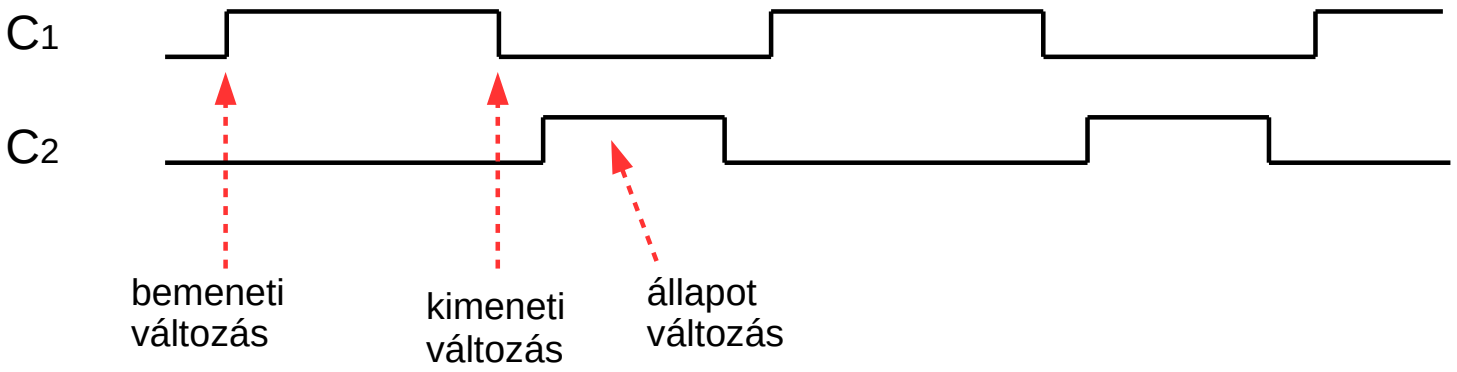
3. megvalósítása



14.2. Fázisregiszteres vezérlő

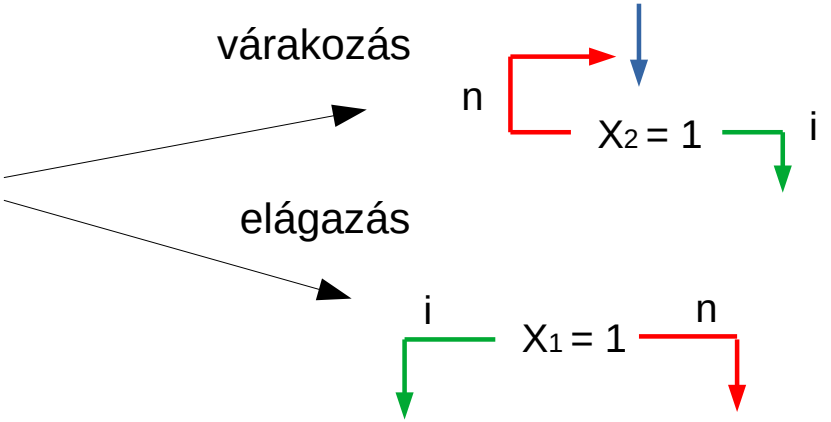
4. magyarázatok

Kétfázisú órajel (C1, C2)



Folyamat ábra műveletei

- bemeneti műveletek → bemenő jel vizsgálata



- kimeneti műveletek
  → kimenő jel értékeinek beállítása

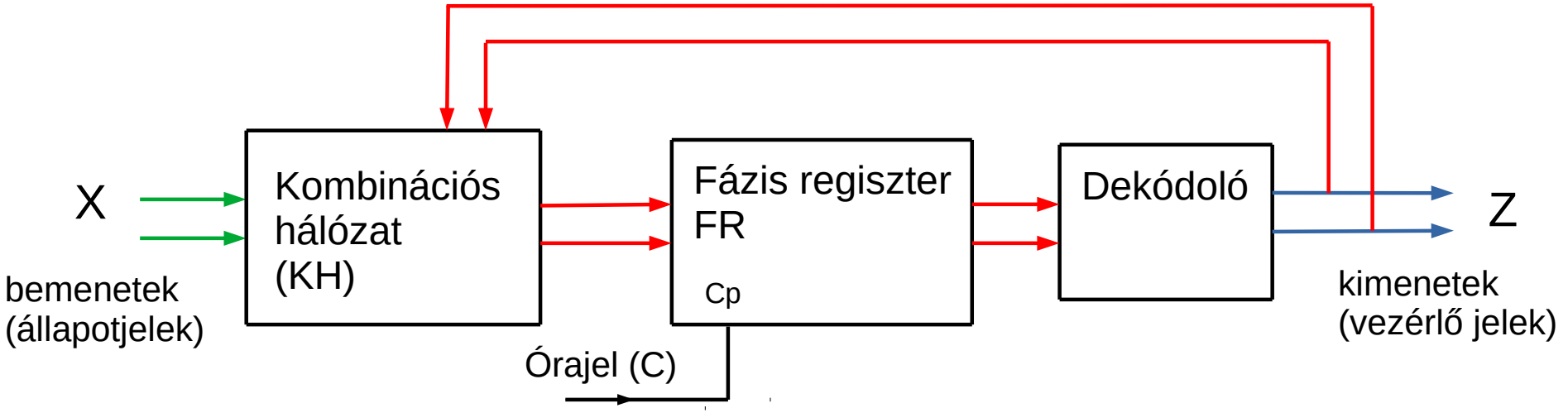
$Z_1\uparrow$ → $Z_1 = 1$ $Z_1\downarrow$ → $Z_1 = 0$

- szekunder műveletek
  → az állapot változók 1 értékét beállító műveletek, az állapot változást (fázis átmenetet) okozzák

$Y_3\uparrow$

14.3. Fázisregiszteres vezérlő 2.

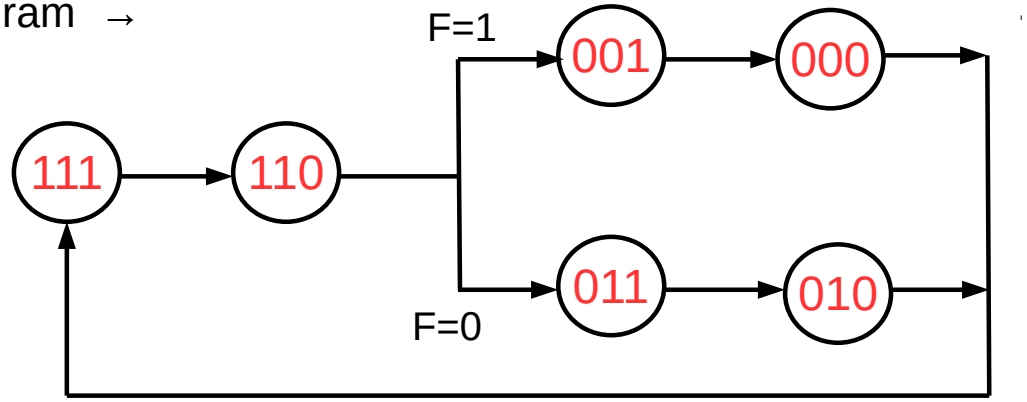
Egy másfajta megközelítés



Tárolósor tartalmazza a vezérlőjelek állapotát kódolt formában

Tervezési példa

állapot diagram →



14.3. Fázisregiszteres vezérlő 2.

Tervezési példa

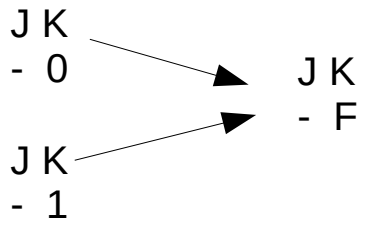
megvalósítás JK tárolókkal, dekódolóval

Állapotátmeneti tábla

				F=0			F=1								
	q_A	q_B	q_C	Q_A	Q_B	Q_C	Q_A	Q_B	Q_C	J_A	K_A	J_B	K_B	J_C	K_C
q_7	1	1	1	1	1	0	1	1	0	-	0	-	0	-	1
q_6	1	1	0	0	1	1	0	0	1	-	1	-	F	1	-
q_1	0	0	1	0	0	0	0	0	0	0	-	0	-	-	1
q_0	0	0	0	1	1	1	1	1	1	1	-	1	-	1	-
q_3	0	1	1	0	1	0	0	1	0	0	-	-	0	-	1
q_2	0	1	0	1	1	1	1	1	1	1	-	-	0	1	-

F=0 1→ 1

F=1 1→ 0



$J_A = q_0 + q_2$

$K_A = q_6$

$J_B = q_0$

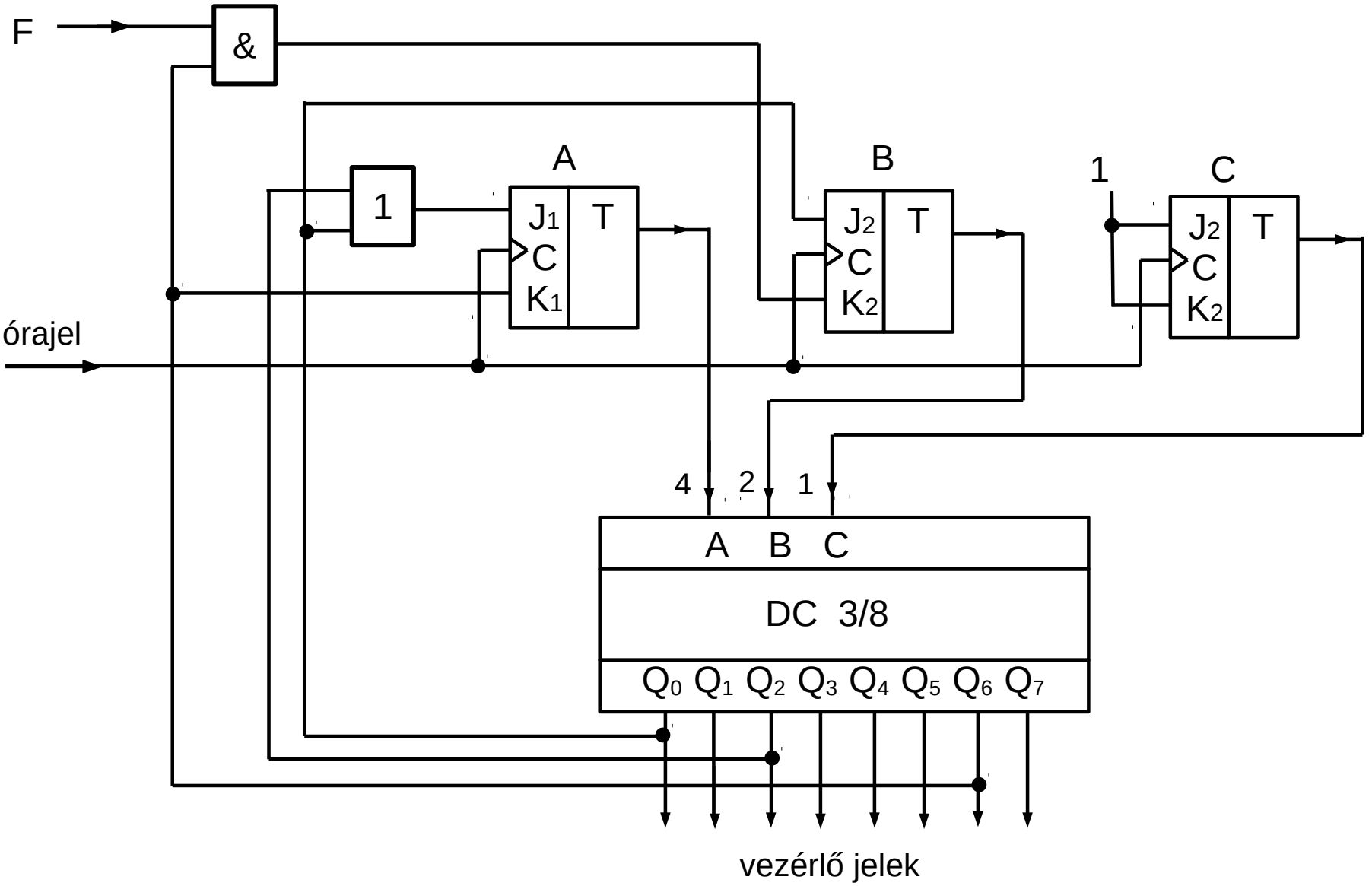
$K_B = q_6 * F$

$J_C = K_C = 1$

14.3. Fázisregiszteres vezérlő 2.

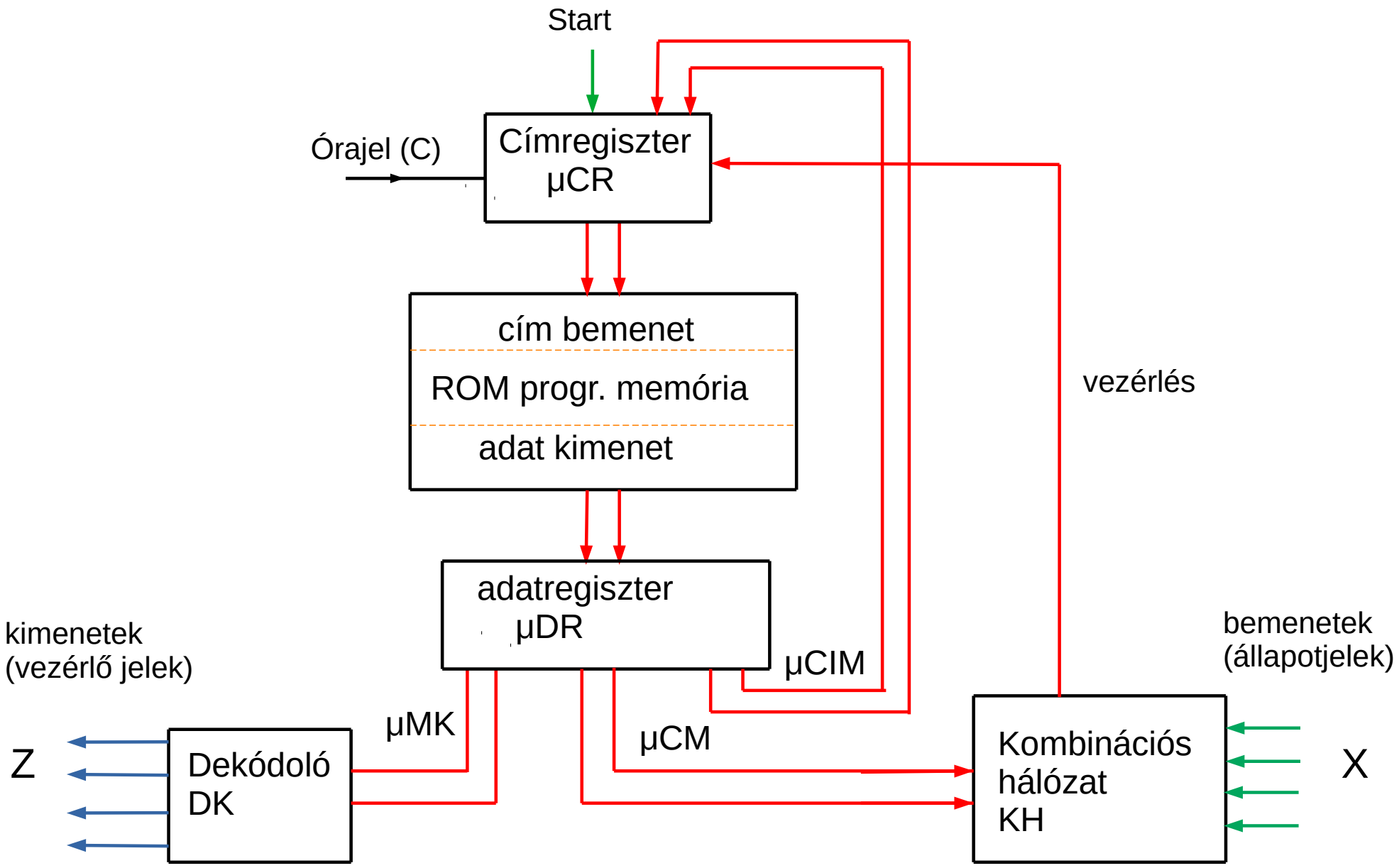
Tervezési példa

megvalósítás



14.4. Mikroprogramozott vezérlő

1. felépítése

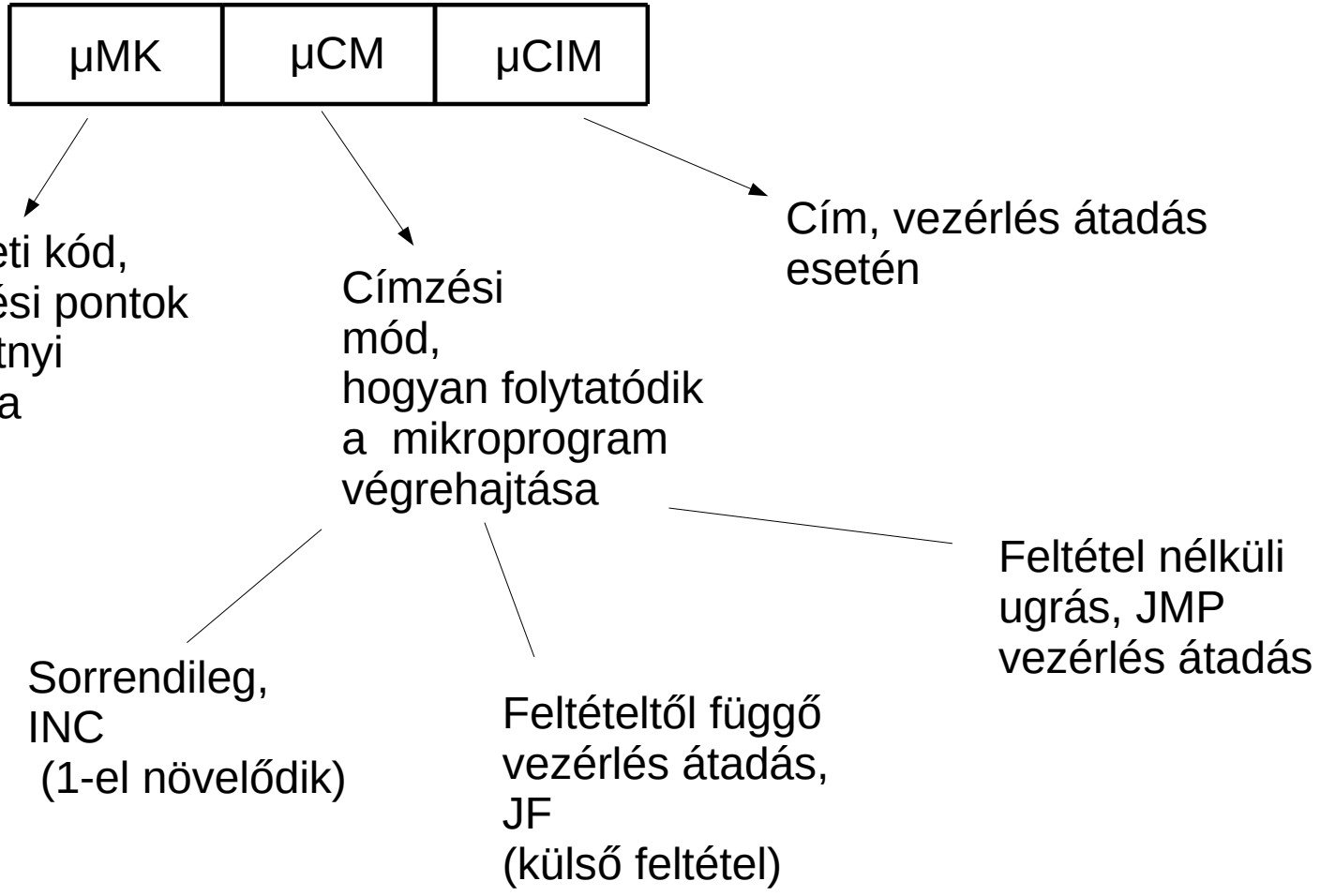


14.4. Mikroprogramozott vezérlő

2. jellemzői

- a művelet végrehajtás lépéseit mikroprogram írja le
  (mikroprogram tárban, memóriában)

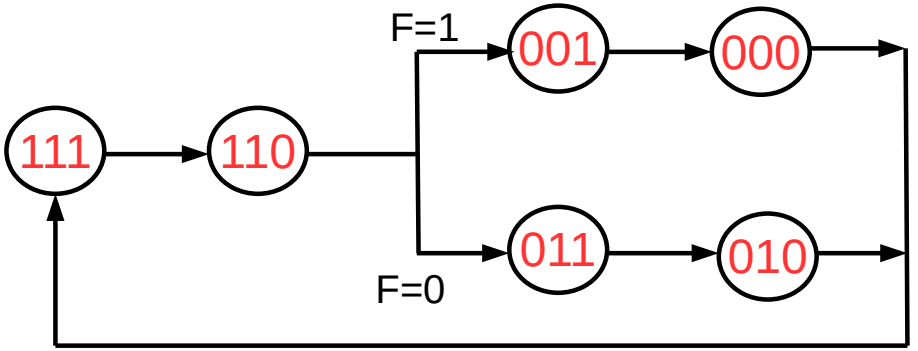
- mikroutasítás felépítése



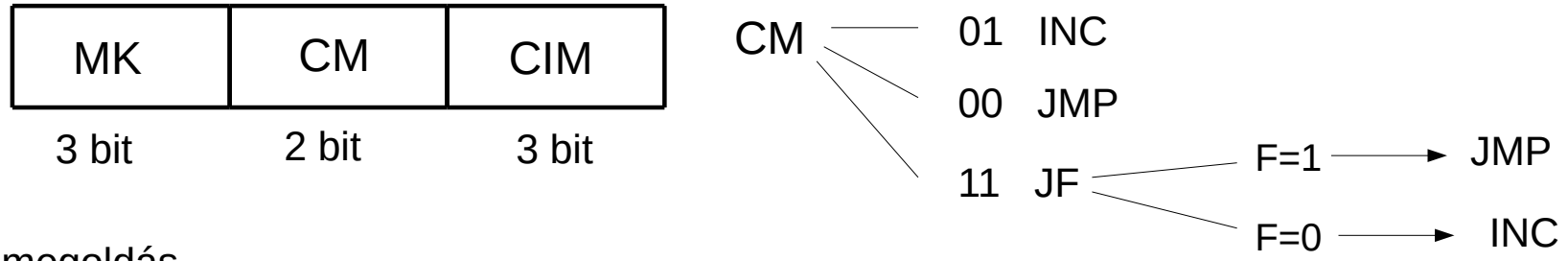
14.4. Mikroprogramozott vezérlő

3. Tervezési példa

állapot diagram



mikroutasítás



megoldás

Memória cím	Állapot, MK	CM	CIM
000	q7	INC	-
001	q6	JF	q1
010	q3	INC	-
011	q2	JMP	q7
100	q1	INC	-
101	q0	JMP	q7

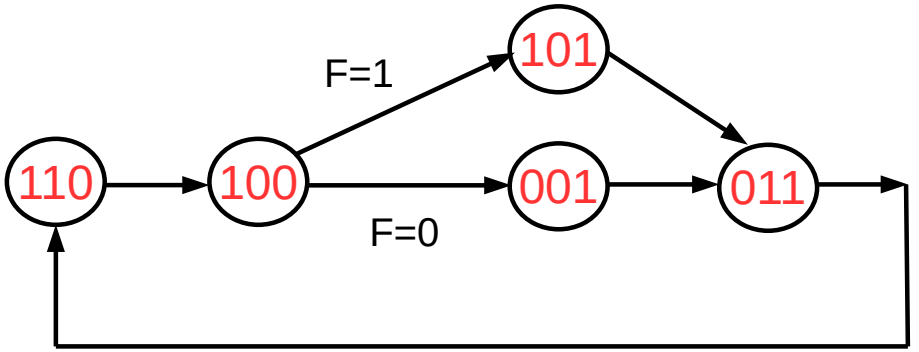
kódolás

Mem. cím	Bináris kód
000	111 01 - - -
001	110 11 100
010	011 01 - - -
011	010 00 000
100	001 01 - - -
101	000 00 000

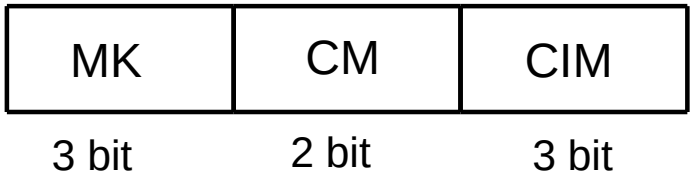
14.4. Mikroprogramozott vezérlő

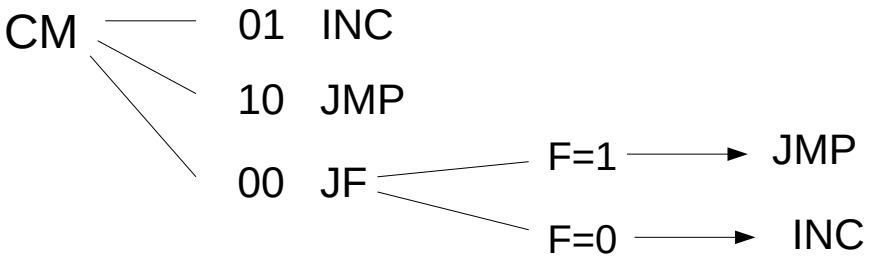
4. Tervezési példa 2.

állapot diagram



mikroutasítás





megoldás

Memória cím	Állapot, MK	CM	CIM
000	q6	INC	-
001	q4	JF	q5
010	q1	INC	-
011	q3	JMP	q6
100	q5	JMP	q3

kódolás

Mem. cím	Bináris kód
000	110 01 - - -
001	100 00 100
010	001 01 - - -
011	011 10 000
100	101 10 011

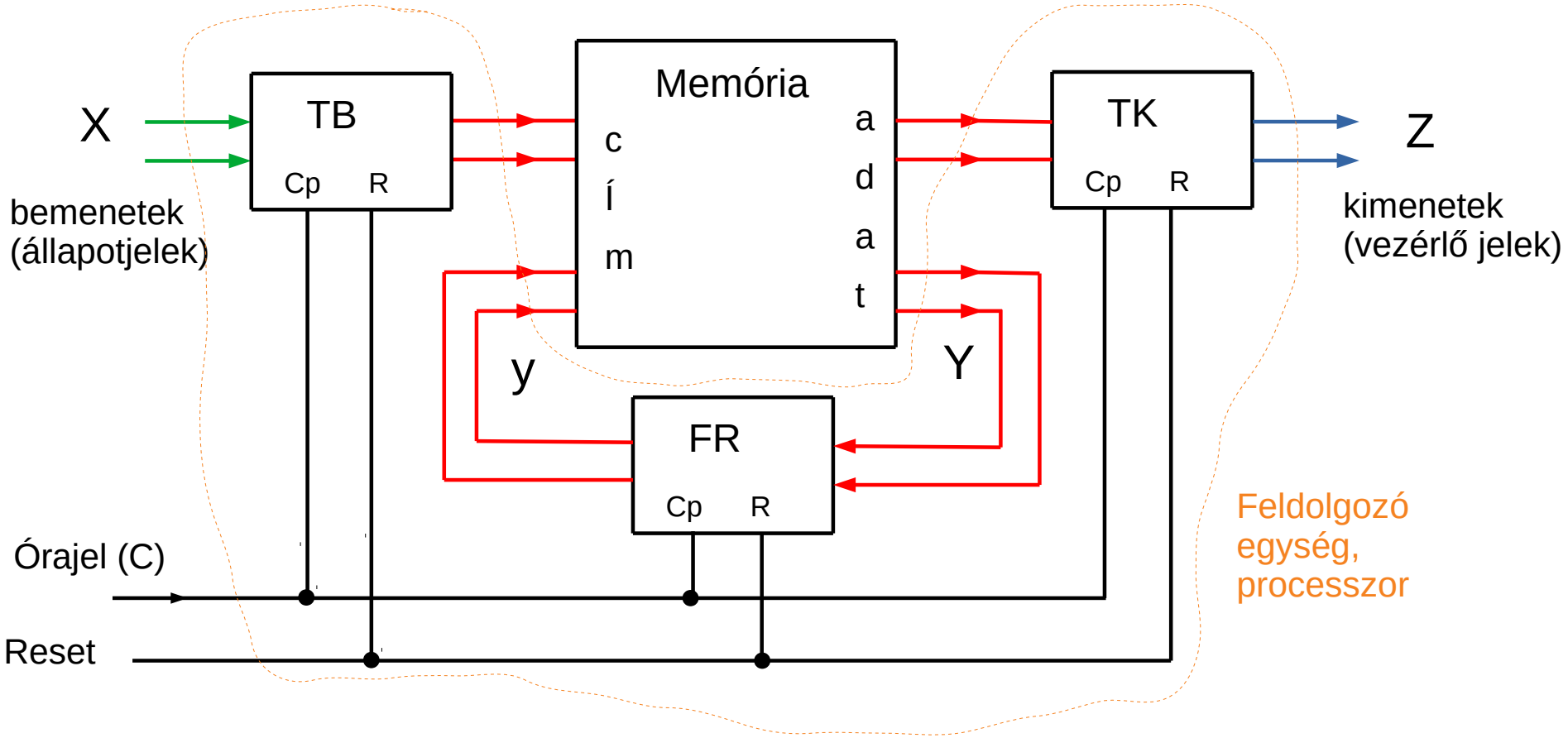
14.5. Mikroprogramozott vezérlő 2.

1. mikroprogramozott vezérlő másképp

Kiindulás fázisregiszteres vezérlőből, de a kombinációs hálózatot memóriával valósítjuk meg !
→ rugalmas, könnyen módosítható vezérlés (a folyamatábra alapján a memóriában lévő programot kell módosítani)

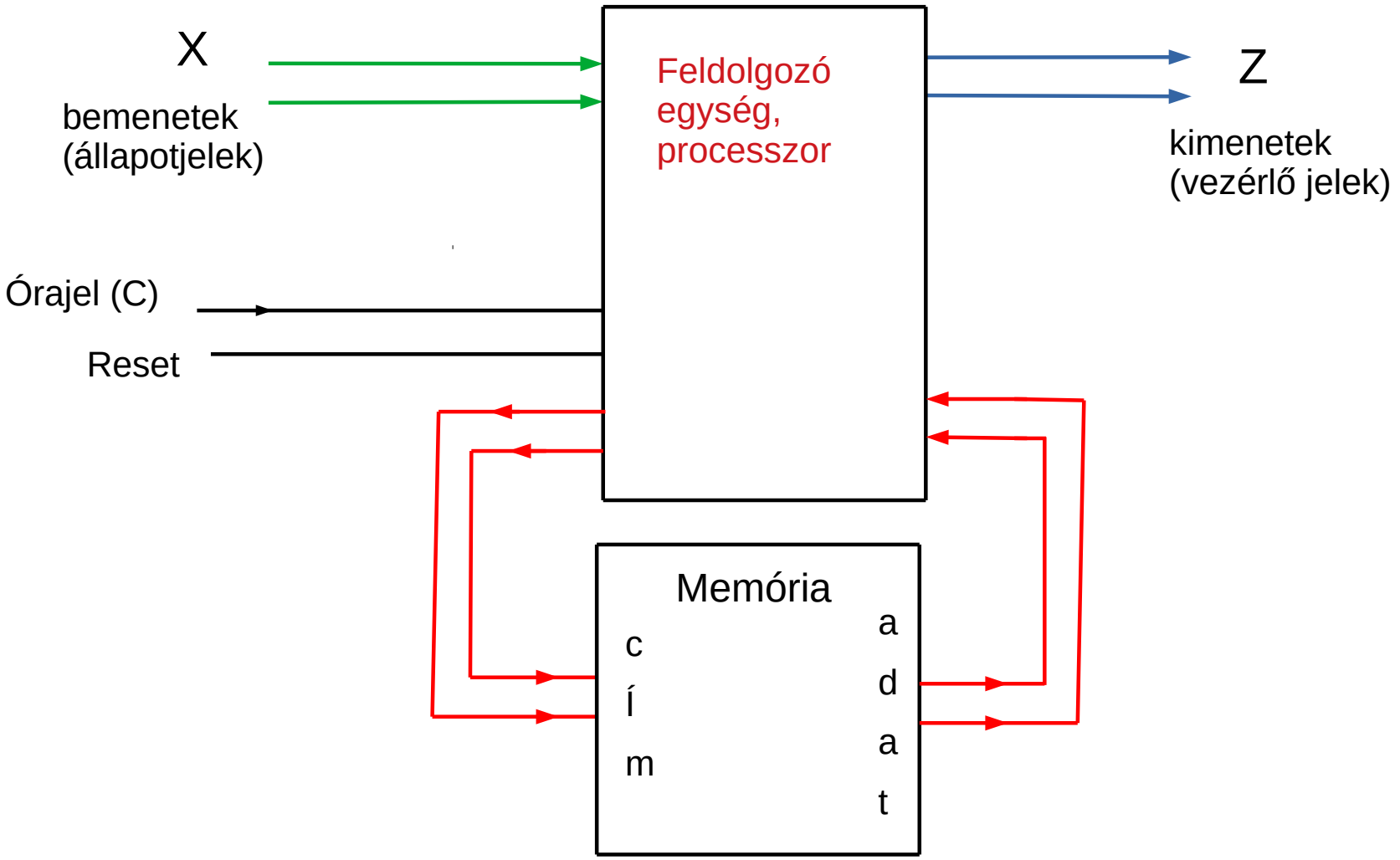
TB – bemeneti időzítők (D tárolók) TK – kimeneti időzítők (SR tárolók)

FR – fázisregiszterek → állapot tárolók (reteszelt D tárolók)



14.5. Mikroprogramozott vezérlő 2.

egyszerűsítve



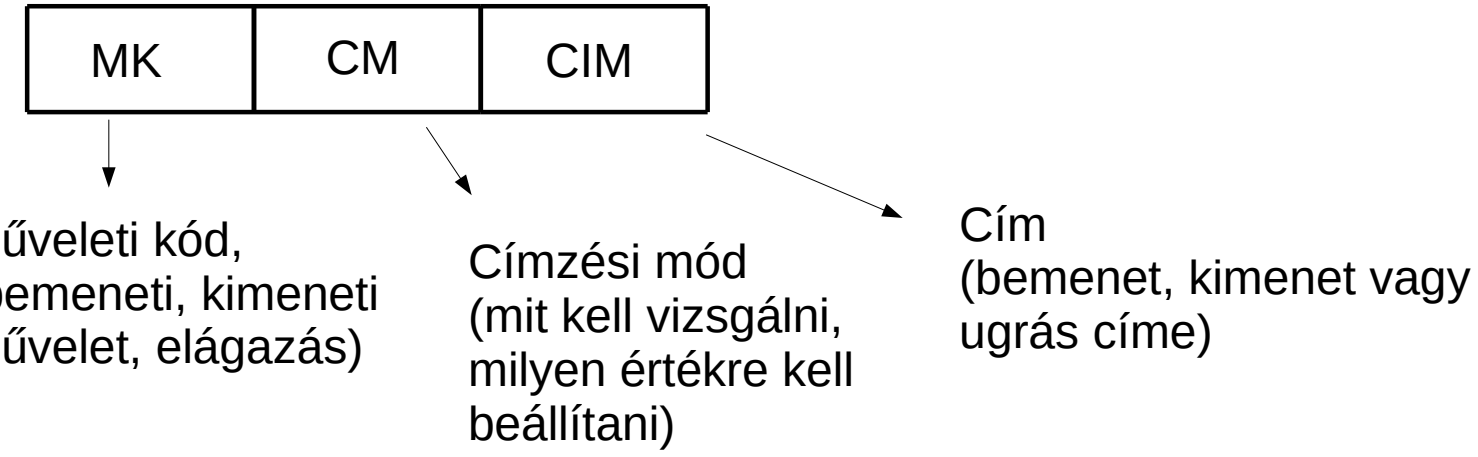
14.5. Mikroprogramozott vezérlő 2.

2. jellemzői

- memória tartalom módosításával különböző vezérlési folyamatábrákat valósíthatunk meg
- folyamatábra módosítása → elemi műveletek → egyszerre csak egy bemeneti jel vizsgálata, illetve csak egy kimenet beállítása
- az elemi műveleteknek megfelelő memóriabeli információegységek → mikroutasítások
- folyamatábra → mikroutasítások sorozata → mikroprogram

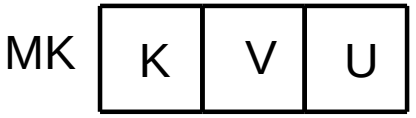
3. mikroutasítás

- meg kell adnia:
  - milyen műveletet kell végrehajtani
  - melyik bemeneti vagy kimeneti jelen
  - elágazás esetén melyik a következő műveletet → ugrási cím



14.5. Mikroprogramozott vezérlő 2.

4. minta

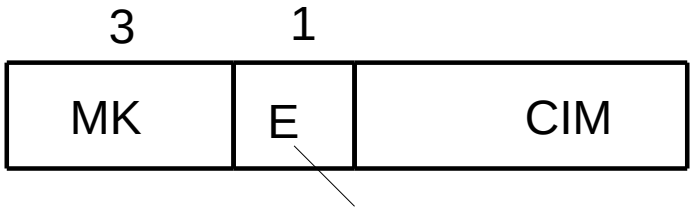


3 bit, 3-ból 1 kód

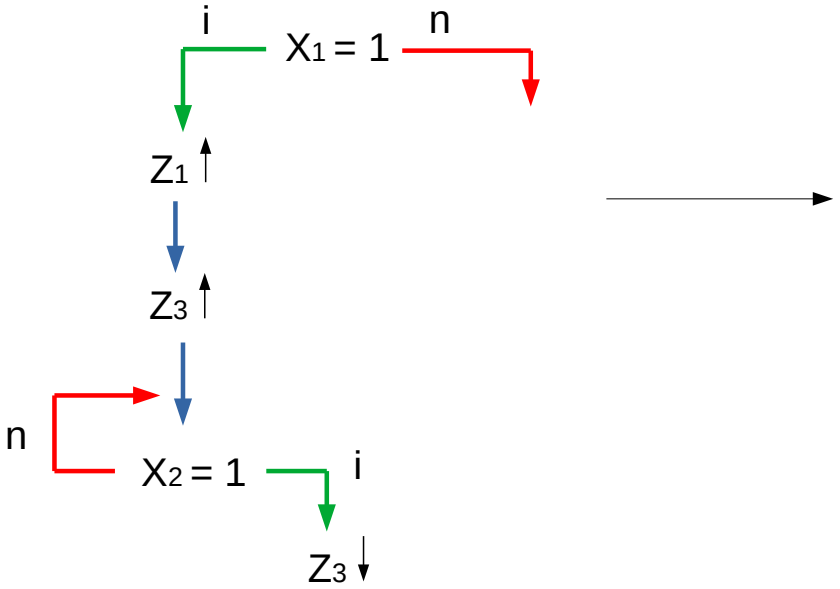
K=1 → kimeneti műveletet

V=1 → bemeneti művelet

U=1 → ugrás



Vizsgálat vagy beállítás értéke




```
CIM1:   V  1  X1
        U  CIM6
        K  1  Z1
        K  1  Z3
CIM2:   V  1  X2
        U  CIM2
        K  0  Z3
```

14.6. Párbeszéd jellegű működés

Hand-shaking (kézfogás)

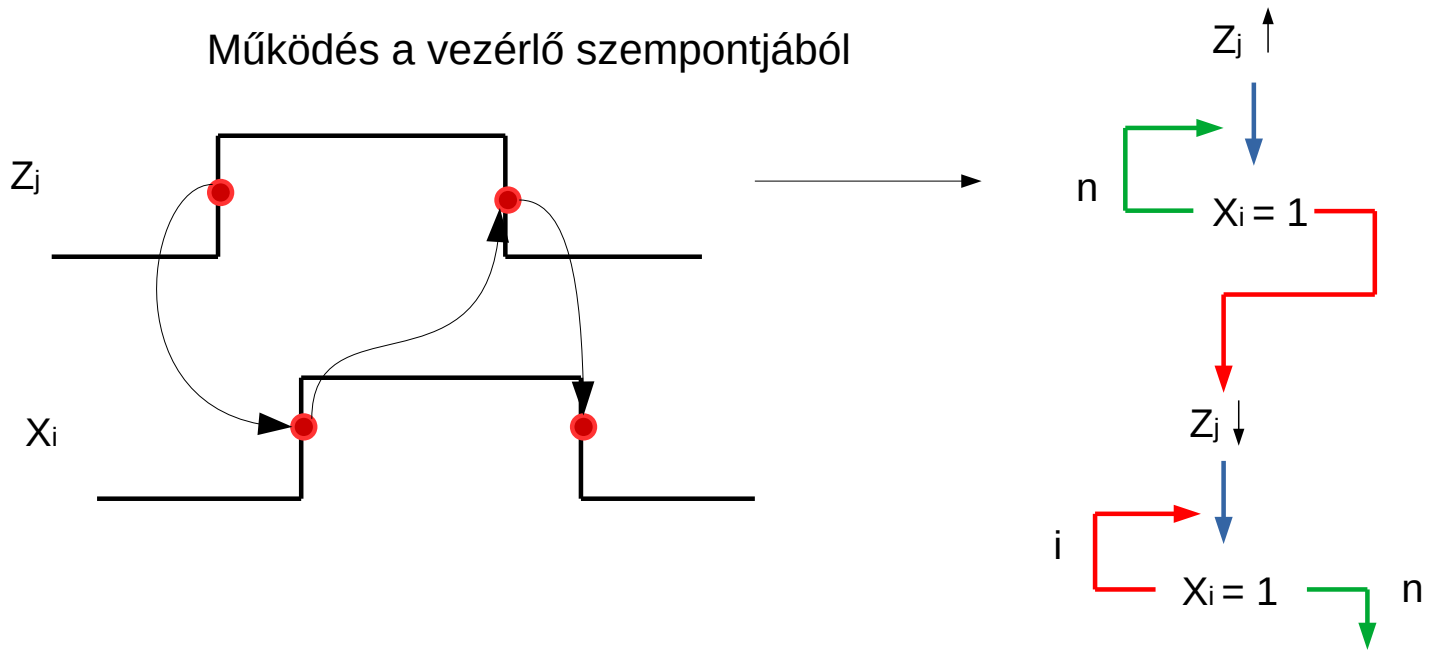
- Minden bemeneti változás létrehoz egy kimeneti változást, és új bemeneti változás csak azután következik be, miután az azt közvetlenül megelőző kimeneti változás már kifejtette hatását →

- bemeneti várakozó műveletek és kimeneti műveletek váltakozva következnek egymás után

- nyugtázó bemeneti kombinációk, minden kimeneti jelnek van egy nyugtázó párja a bemenetek között

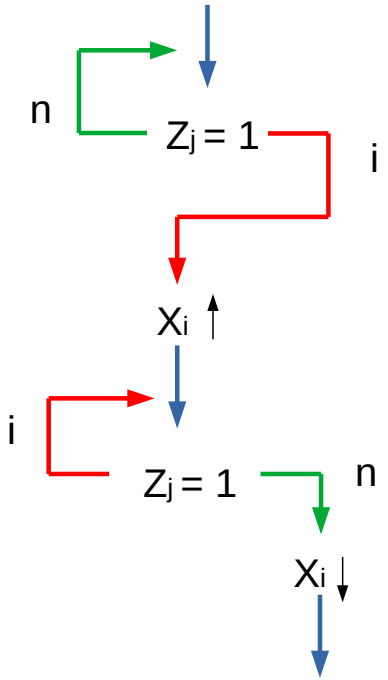
Jelenkénti hand-shaking jellegű működés

Működés a vezérlő szempontjából



14.6. Párbeszéd jellegű működés

Működés a környezet szempontjából



- Előnye: a jelforgalom nagy sebességgel és biztonságosan játszódhat le két egység között

- Akkor célszerű használni, ha viszonylag nagy késleltető hatású összeköttetéseken keresztül történik információ átvitel gyors működésű egységek között

- ha az összeköttetés jel késleltető hatása kicsi akkor kerülni kell !! → különösen aszinkron egységek esetén okozhat ugyanis hazárd jelenségeket

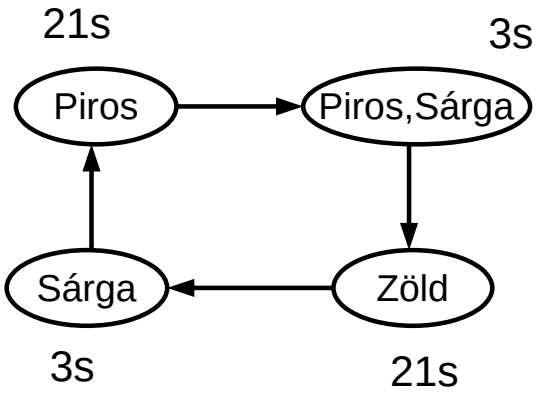
14.7. Sorrendi hálózat tervezése

Tervezés funkcionális elem felhasználásával

1. mintafeladat

közlekedési jelzőlámpa vezérlő áramkör tervezése

állapotdiagram



4 állapot → 16 állapot

piros → piros-sárga → zöld → sárga → piros → …

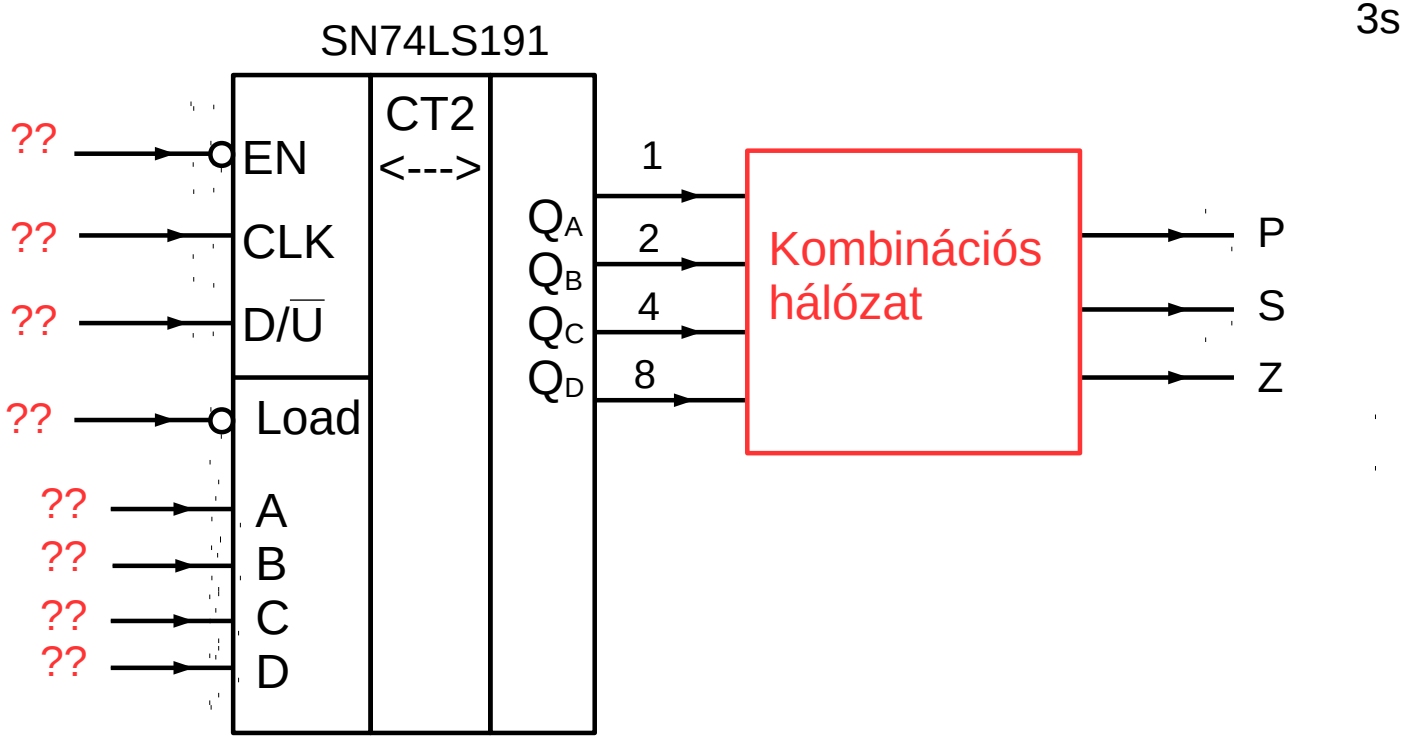
piros-sárga és sárga állapot rövid ideig tart → 1 állapot

piros és zöld állapot hosszabb ideig tart → 7 állapot

Kimenetek: vezérelni kell a három lámpát → P, S és Z

A tervezés → 4 bites szinkron számlálót (16 állapot)

+ kiegészítő kapuáramköröket használhatunk



14.7. Sorrendi hálózat tervezése

1. mintafeladat, megoldás

A kombinációs hálózat igazságtáblázata:

	Q_D	Q_C	Q_B	Q_A	P	S	Z
0.	0	0	0	0	1	0	0
1.	0	0	0	1	1	0	0
2.	0	0	1	0	1	0	0
3.	0	0	1	1	1	0	0
4.	0	1	0	0	1	0	0
5.	0	1	0	1	1	0	0
6.	0	1	1	0	1	0	0
7.	0	1	1	1	1	1	0
8.	1	0	0	0	0	0	1
9.	1	0	0	1	0	0	1
10.	1	0	1	0	0	0	1
11.	1	0	1	1	0	0	1
12.	1	1	0	0	0	0	1
13.	1	1	0	1	0	0	1
14.	1	1	1	0	0	0	1
15.	1	1	1	1	0	1	0

A piros 1-es ha $Q_D = 0$

$\rightarrow \quad P = \overline{Q_D}$

A sárga 1-es ha $Q_C = 1$ ÉS $Q_B = 1$ ÉS $Q_A = 1$

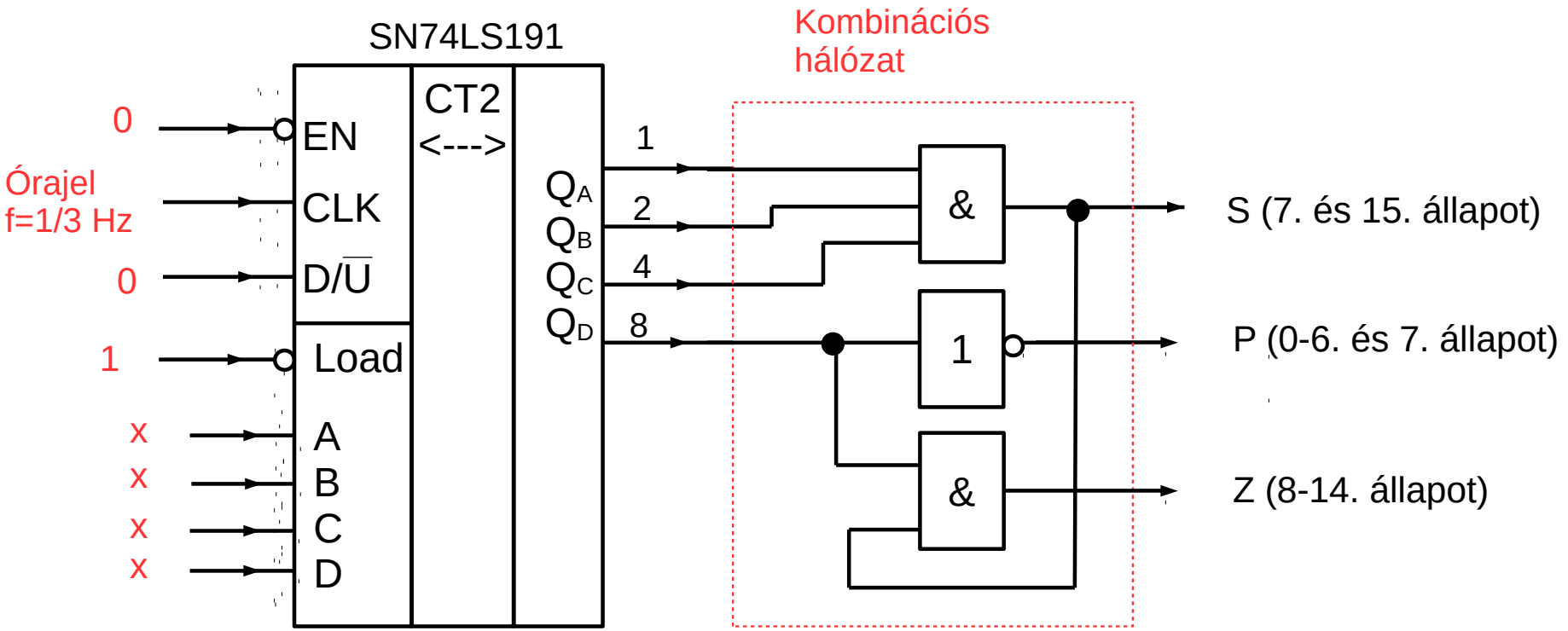
$\rightarrow \quad S = Q_C * Q_B * Q_A$

A zöld 1-es ha $Q_D = 1$ ÉS (NEM sárga)

$\rightarrow \quad Z = Q_D * \overline{S}$

14.7. Sorrendi hálózat tervezése

1. mintafeladat, megoldás

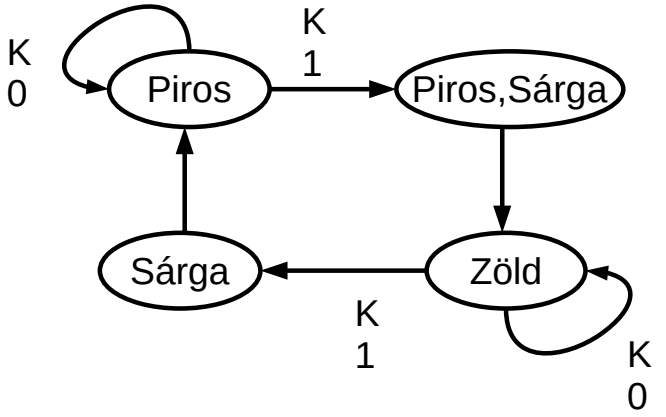


14.7. Sorrendi hálózat tervezése

2. minta feladat: forgalmi jelzőlámpa vezérlő áramkör tervezése másképp

4 állapot van: → 2 állapotváltozó (Q_2 és Q_1)
piros → piros+sárga → zöld → sárga → piros → …
Piros és zöld állapotból csak hosszabb késleltetés után lép tovább !! →
K bemenet jelzi (1-el) hogy a késleltetés letelt

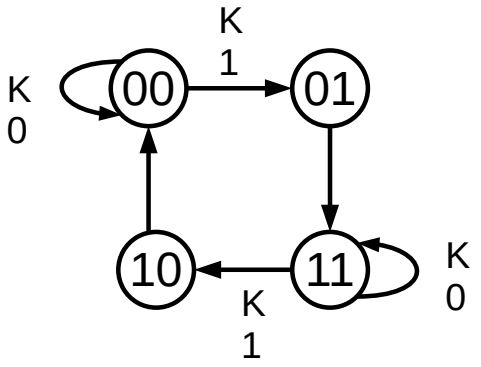
állapotdiagram



Kimenetek:
- vezérelni kell a három lámpát →
  P, S és Z kimenetek
- indítani kell a késleltető áramkört, ha zöld vagy piros állapotba lépünk → I kimenetre 1-es

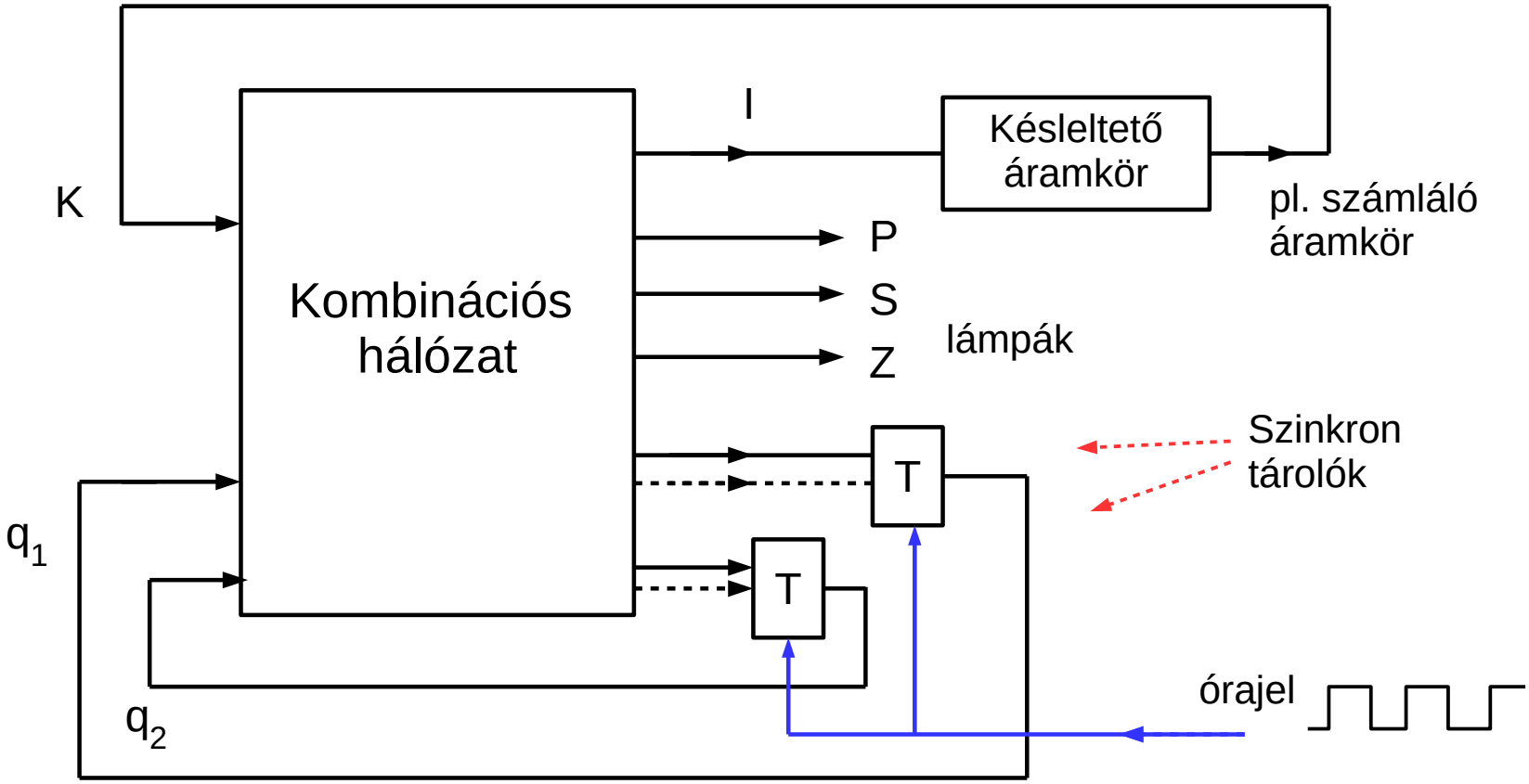
kódolt állapotdiagram

4 állapot → pl. piros=00, zöld=11,
sárga=10, piros+sárga=01



14.7. Sorrendi hálózat tervezése

2. minta feladat, a hálózat felépítése:



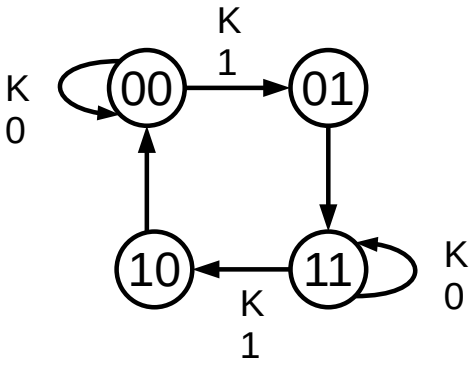
Attól függően, hogy milyen tárolókat használunk → változik a kimenetek száma
→ T vagy D tárolók esetén 2 kimenet kell a két tároló vezérléséhez (T_1, T_2 vagy D_1, D_2)
→ JK tárolók esetén 4 kimenet kell a két tároló vezérléséhez (J_1, J_2, K_1, K_2)
és változik a kombinációs hálózat felépítése is !!

14.7. Sorrendi hálózat tervezése

megvalósítás JK tárolókkal

A JK tárolók vezérlési függvényeinek és a kimenetek függvényeinek meghatározása

kódolt állapotdiagram



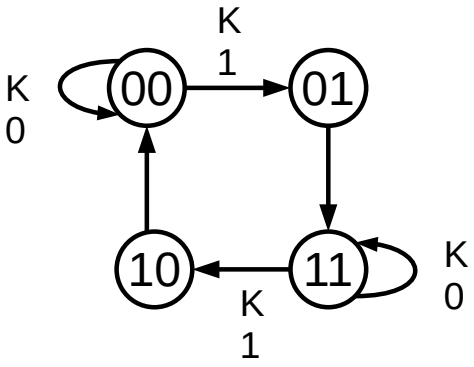
állapot átmeneti tábla

K	q_2	q_1	Q_2	Q_1	J_2	K_2	J_1	K_1	I	P	S	Z
0	0	0	0	0	?	?	?	?	?	?	?	?
0	0	1	1	1	?	?	?	?	?	?	?	?
0	1	0	0	0	?	?	?	?	?	?	?	?
0	1	1	1	1	?	?	?	?	?	?	?	?
1	0	0	0	1	?	?	?	?	?	?	?	?
1	0	1	1	1	?	?	?	?	?	?	?	?
1	1	0	0	0	?	?	?	?	?	?	?	?
1	1	1	1	0	?	?	?	?	?	?	?	?

14.7. Sorrendi hálózat tervezése

A JK tárolók vezérlési függvényeinek és a kimenetek függvényeinek meghatározása

kódolt állapotdiagram



állapot átmeneti tábla

K	q_2	q_1	Q_2	Q_1	J_2	K_2	J_1	K_1	I	P	S	Z
0	0	0	0	0	0	x	0	x	1	1	0	0
0	0	1	1	1	1	x	x	0	0	1	1	0
0	1	0	0	0	x	1	0	x	0	0	1	0
0	1	1	1	1	x	0	x	0	1	0	0	1
1	0	0	0	1	0	x	1	x	1	1	0	0
1	0	1	1	1	1	x	x	0	0	1	1	0
1	1	0	0	0	x	1	0	x	0	0	1	0
1	1	1	1	0	x	0	x	1	1	0	0	1

14.7. Sorrendi hálózat tervezése

vezérlési táblák

J_1

K \ q_2q_1	00	01	11	10
0	0	x	x	0
1	1	x	x	0

$K*\overline{q_2}$

J_2

K \ q_2q_1	00	01	11	10
0	0	1	x	x
1	0	1	x	x

q_1

K_1

K \ q_2q_1	00	01	11	10
0	x	0	0	x
1	x	0	1	x

$K*q_2$

K_2

K \ q_2q_1	00	01	11	10
0	x	x	0	1
1	x	x	0	1

$\overline{q_1}$

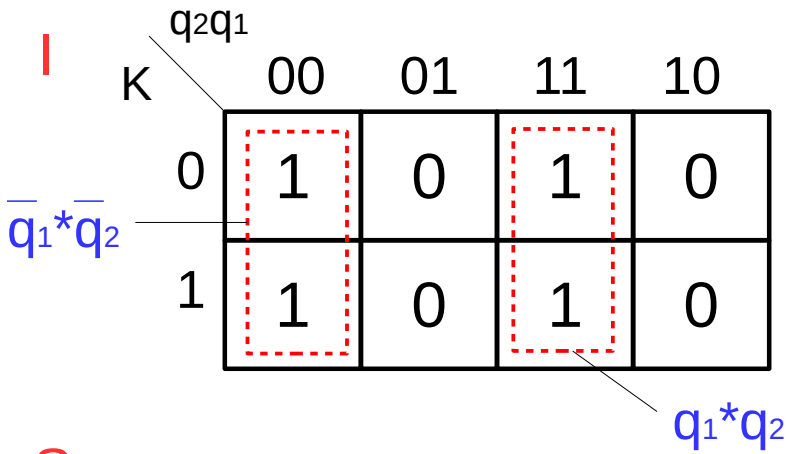
<u>vezérlési függvények</u>

$J_1 = K*\overline{q_2}$ $K_1 = K*q_2$ $J_2 = q_1$ $K_2 = \overline{q_1}$

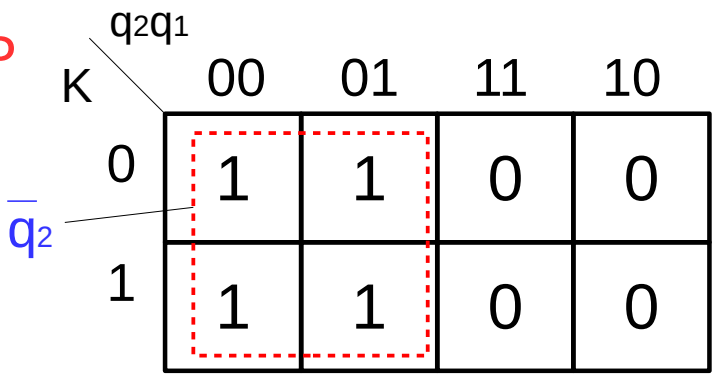
14.7. Sorrendi hálózat tervezése

Kimeneti függvények

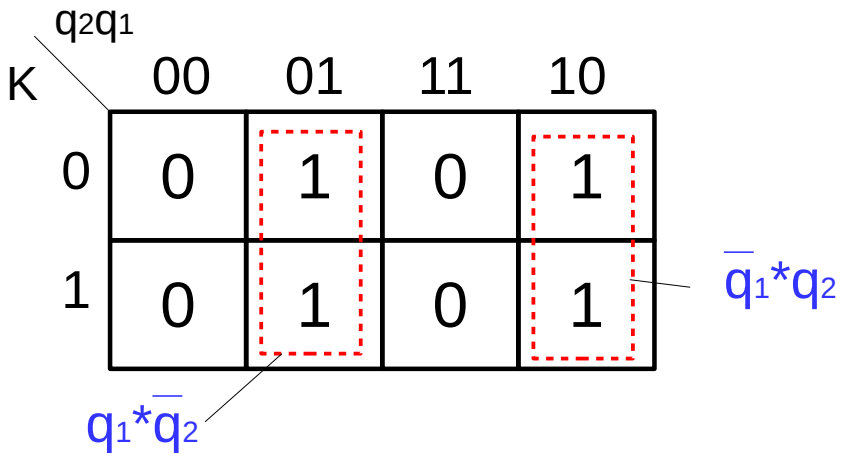
I



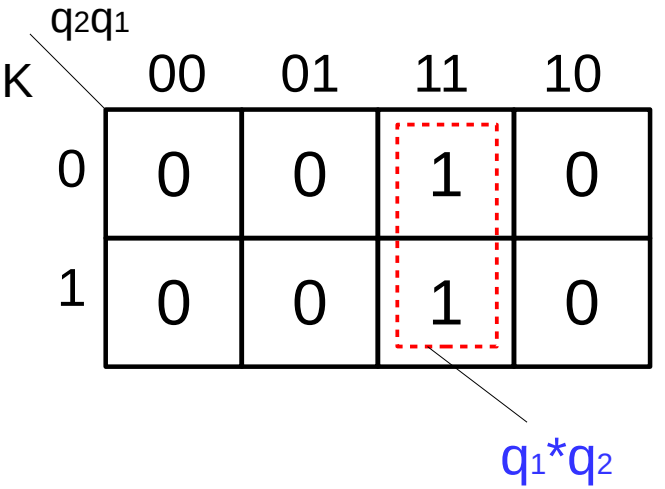
P



S



Z



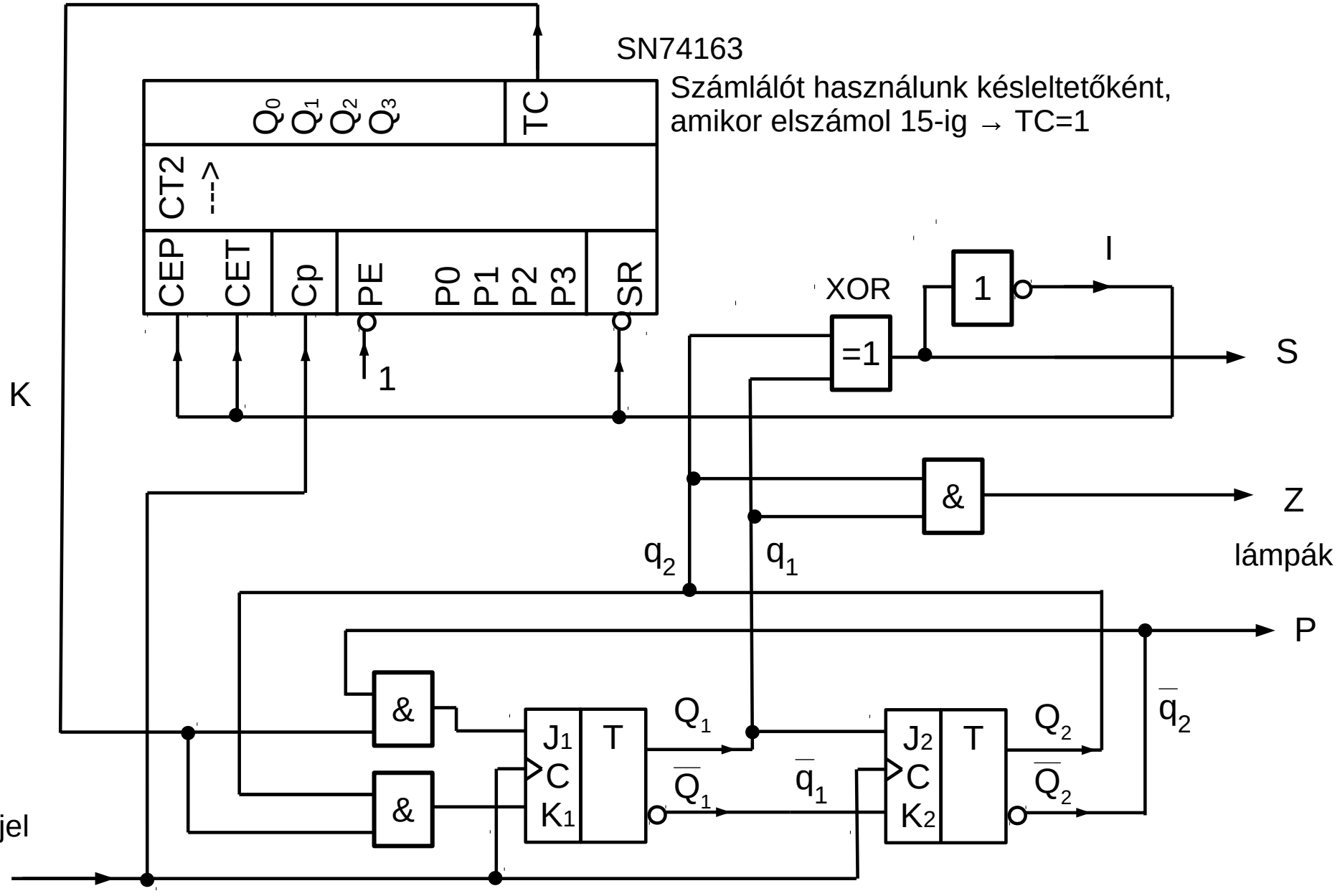
<u>kimeneti függvények</u>

$I = q_1 * q_2 + \bar{q}_1 * \bar{q}_2$ $P = \bar{q}_2$ $S = q_1 * \bar{q}_2 + \bar{q}_1 * q_2 = \bar{I}$ $Z = q_1 * q_2$

XOR

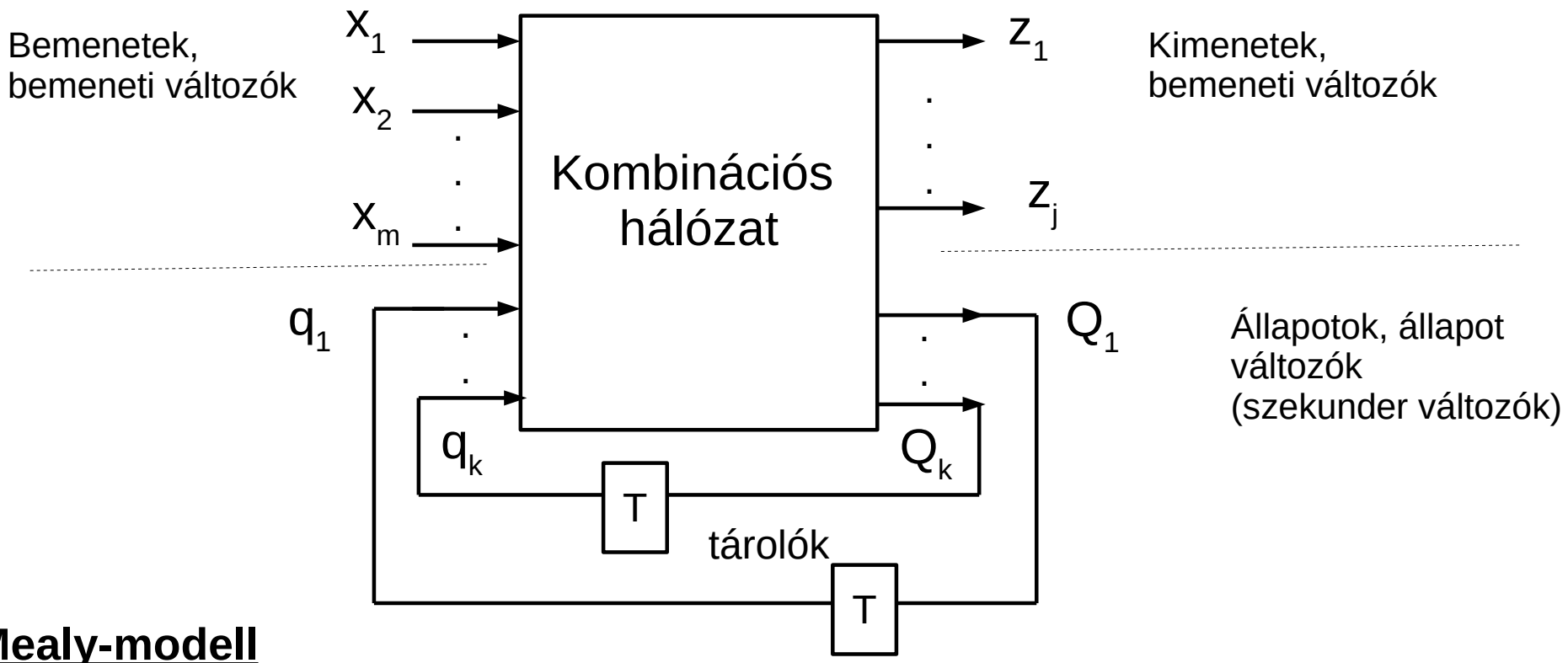
14.7. Sorrendi hálózat tervezése

megoldás



14.8. Sorrendi hálózatok modellezése

Kimenet előállításának módja



Mealy-modell

$Z = f_z (X,q)$

$Q = f_q (X,q)$

A kimenetek és az új állapotok a bemenetektől és a jelenlegi állapotoktól is függenek

Moore-modell

$Z = f_z (q)$

$Q = f_q (X,q)$

A kimenetek közvetlenül csak a jelenlegi állapotoktól függenek !!

14.9. Jelterjedési késleltetések hatása

- aszinkron hálózatok esetén lehetnek problémák
- statikus és dinamikus hazárd (kapuk késleltetése miatti hibák) kiküszöböltnek tekintett

1. funkcionális hazárd

- a bemeneti változók közötti versenyhelyzet okozza
- nem szomszédos bemeneti kombináció változás

→ hiba lehetőség,
a hálózat állapota nem definiálható egyértelműen

- megoldás → szinkronizáció
→ szomszédos bemeneti kombináció változások biztosítása

2. versenyhelyzet

- nem jön létre hibás stabil állapot csak az változhat, hogy a helyes stabil állapotot milyen instabil állapotok közbeiktatásával éri el a hálózat

3. kritikus versenyhelyzet

- a szekunder változók (állapot változók) közötti versenyhelyzet
- ha a szekunder változók nem egyszerre változnak → hibás állapot-átmenet jöhet létre ! → esetleg hibás stabil állapot is kialakulhat !!
- kiküszöbölése → állapot kódolás megfelelő megválasztása (egyszerre csak egy állapot változó változzon)

14.9. Jelterjedési késleltetések hatása

4. lényeges hazárd

- a bemeneti változók és a szekunder változók közötti versenyhelyzet okozza (késleltetési viszonyok miatt)

  ↘

  hiba lehetőség,
  Olyan instabil állapot jöhet létre, amely hibás stabil állapotba vezeti a hálózatot

- kiküszöbölése

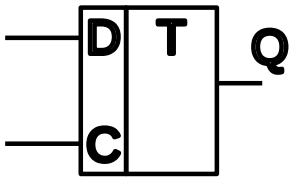
  ↘

  - a megfelelő visszacsatoló ágakba megfelelő értékű késleltetés beépítése

  - aszinkron hálózatoknál feltételeztük ugyanis, hogy a hálózatot először a bemeneti változás éri, majd ezután érvényesül a szekunder változók változása

14.10. Aszinkron hálózatok tervezése

Minta feladat 1.: pozitív él-vezérelt D tároló tervezése
Megvalósítás visszacsatolt kombinációs hálózattal → C rendes bemenet)



Amikor a C bemenet 0→ 1 átmenetet csinál akkor Q = D
(a kimenet felveszi a bemenet értékét),
egyébként Q = q (tárolja az előző állapotot)

C – clock ---> órajel bemenet

1. előzetes állapottábla →

○ stabil állapot

Minden sorban egy stabil állapot legyen !
Nem szomszédos bemeneti változás ne legyen egy sorban ! → határozatlan

y \ DC	00	01	11	10
a	a 0	b 0	-	c 0
b	a 0	b 0	d 0	-
c	a 0	-	e -	c 0
d	-	b 0	d 0	c 0
e	-	g 1	e 1	f 1
f	h 1	-	e 1	f 1
g	h 1	g 1	e 1	-
h	h 1	b -	-	f 1

→ Következő állapotot → c és a kimenet → 0

2. állapot összevonás

a, b, d→ A

e, f, g→ C

c→ B

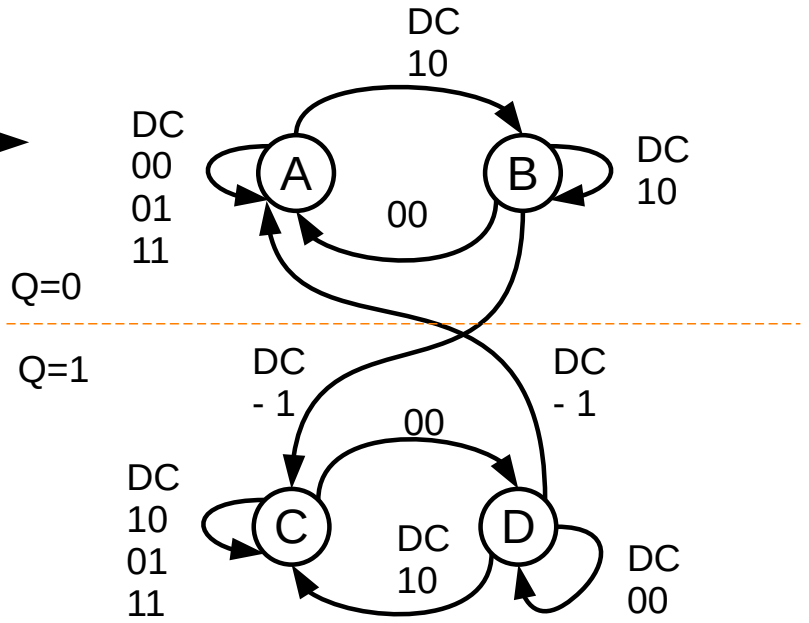
h→ D

14.10. Aszinkron hálózatok tervezése

3. összevont állapottábla

y \ DC	00	01	11	10
A	(A) 0	(A) 0	(A) 0	B 0
B	A 0	-	C -	(B) 0
C	D 1	(C) 1	(C) 1	(C) 1
D	(D) 1	A -	-	C 1

állapotdiagram



4. állapot kódolás

4 állapot → 2 állapot változó
Minden állapotváltozás esetén egyszerre csak egy állapot változó változzon !!

	y_1	y_2
A	0	0
B	0	1
C	1	1
D	1	0

kódolt állapot tábla

y_1 y_2 \ DC	00	01	11	10
00	(00) 0	(00) 0	(00) 0	01 0
01	00 0	-	11 -	(01) 0
11	10 1	(11) 1	(11) 1	(11) 1
10	(10) 1	00 -	-	11 1

Y_1Y_2Q

14.10. Aszinkron hálózatok tervezése

5. vezérlési tábla összeállítása

Megvalósítási lehetőségek:

- aszinkron tárolókkal → vezérlési tábla
- visszacsatolt kombinációs hálózattal → vezérlési tábla = kódolt állapot tábla

$y_1\ y_2$ \ DC	00	01	11	10
00	00 0	00 0	00 0	01 0
01	00 0	-	11 -	01 0
11	10 1	11 1	11 1	11 1
10	10 1	00 -	-	11 1

6. logikai függvények egyszerűsítése

Y_1

$y_1\ y_2$ \ DC	00	01	11	10
00	0	0	0	0
01	0	-	1	0
11	1	1	1	1
10	1	0	-	1

$Y_1 = C*y_2 + \overline{C}*y_1 + y_1*y_2$

Y_2

$y_1\ y_2$ \ DC	00	01	11	10
00	0	0	0	1
01	0	-	1	1
11	0	1	1	1
10	0	0	-	1

$Y_2 = C*y_2 + \overline{C}*D + D*y_2$

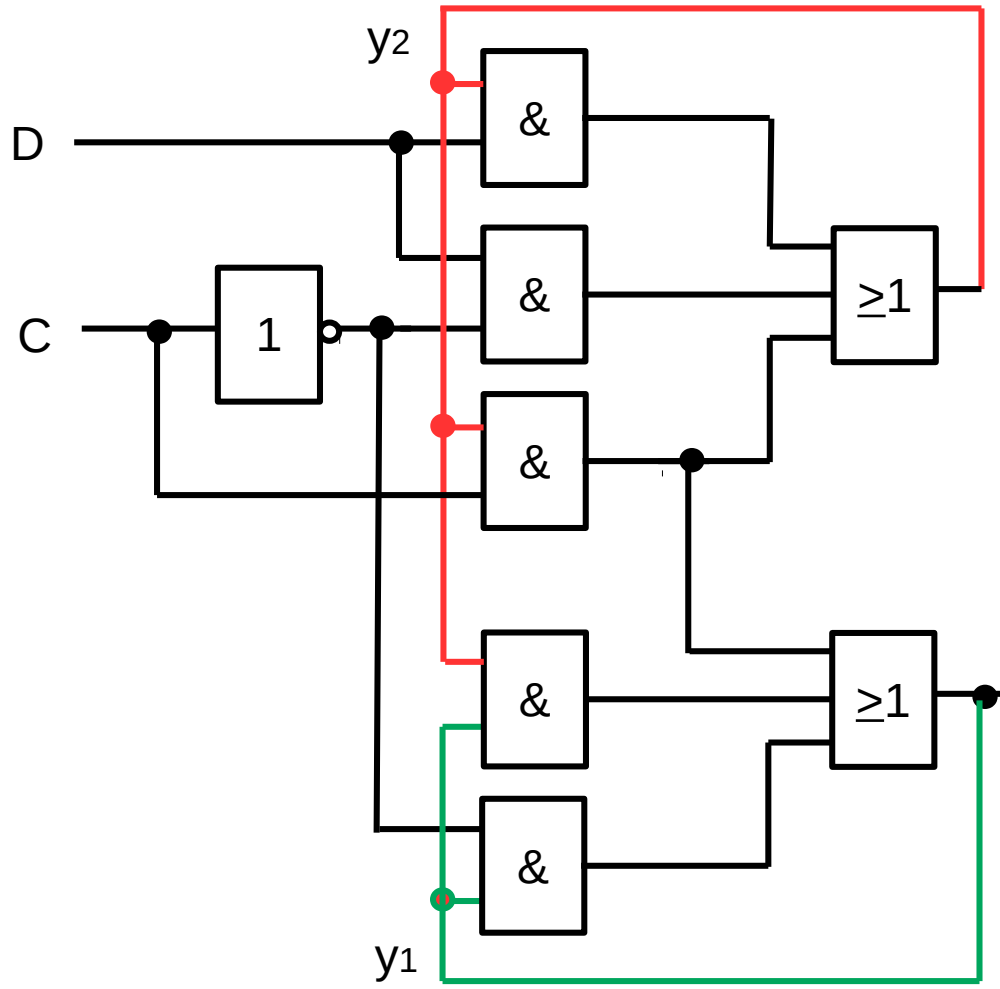
Q

$y_1\ y_2$ \ DC	00	01	11	10
00	0	0	0	0
01	0	-	-	0
11	1	1	1	1
10	1	-	-	1

$Q = y_1$

14.10. Aszinkron hálózatok tervezése

7. kapcsolási rajz



$Y_2 = C*y_2 + \overline{C}*D + D*y_2$

$Q = y_1$

$Y_1 = C*y_2 + \overline{C}*y_1 + y_1*y_2$

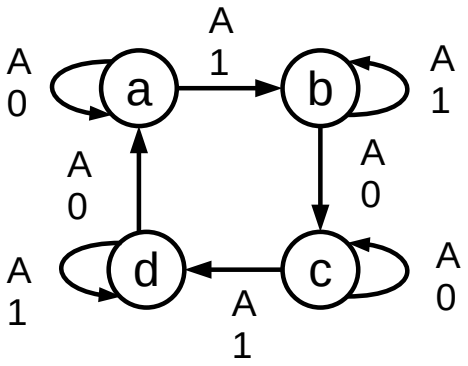
14.10. Aszinkron hálózatok tervezése

Minta feladat 2.:

A bemenet (A) állapot változásait számoljuk két biten (Z_2 Z_1)

4 állapot a, b, c, d (0, 1, 2, 3), ahol éppen tart a számolásban

1. állapotdiagram



2. előzetes állapottábla

y \ A	0	1
a	a 00	b 10
b	c 01	b 10
c	c 01	d 11
d	a 00	d 11

Y Z_1 Z_2

cellákban levő értékek

stabil állapot

Minden sorban egy stabil állapot legyen !

3. állapot összevonás

Most nem lehet állapotokat összevonni

14.10. Aszinkron hálózatok tervezése

4. állapot kódolás

Minden állapotváltozás esetén egyszerre csak egy állapot változó változzon !!
→ ellenkező esetben hibás állapot-átmenet lehetséges (kritikus versenyhelyzet)

	y_2	y_1
a	0	0
b	0	1
c	1	1
d	1	0

5. kódolt állapottábla

y_2 y_1 \ A	0	1
00	00 00	01 10
01	11 01	01 10
11	11 01	10 11
10	00 00	10 11

Egy cella tartalma:
$Y_2\ Y_1\ Z_1\ Z_2$

6. vezérlési tábla

Először dönteni a megvalósításról:
aszinkron tárolókkal, vagy visszacsatolt kombinációs hálózattal

Ha visszacsatolt, akkor
vezérlési tábla = kódolt állapot tábla

Karnaugh táblákba átírni az $Y_2\ Y_1\ Z_1\ Z_2$ értékeket

14.10. Aszinkron hálózatok tervezése

7. függvények egyszerűsítése

Y_2

y_2 y_1 \ A	0	1
00	0	0
01	1	0
11	1	1
10	0	1

Y_1

y_2 y_1 \ A	0	1
00	0	1
01	1	1
11	1	0
10	0	0

Z_1

y_2 y_1 \ A	0	1
00	0	1
01	0	1
11	0	1
10	0	1

Z_2

y_2 y_1 \ A	0	1
00	0	0
01	1	0
11	1	1
10	0	1

$$Y_2 = \overline{A}*y_1 + y_1*y_2 + A*y_2$$

$$Y_1 = \overline{A}*y_1 + y_1*\overline{y}_2 + A*\overline{y}_2$$

$$Z_1 = A$$

$$Z_2 = \overline{A}*y_1 + y_1*y_2 + A*y_2$$

$$Z_2 = Y_2$$