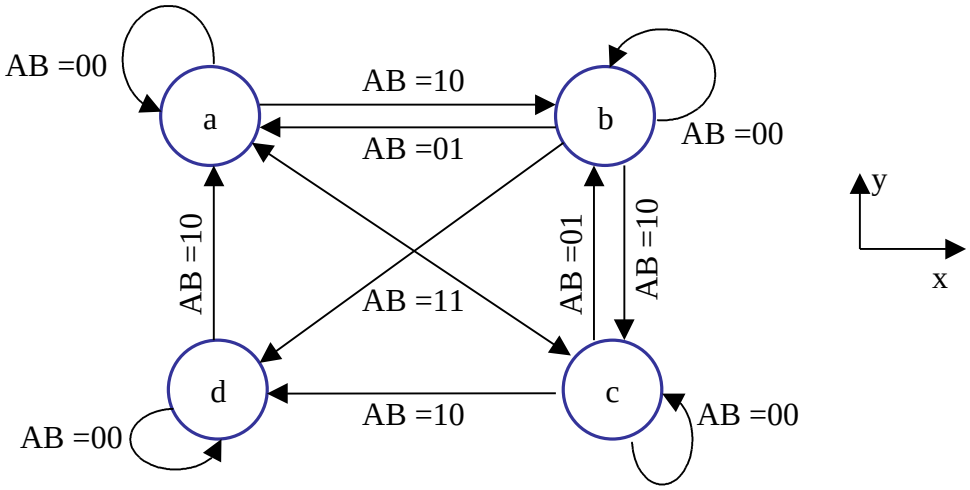
Sorrendi hálózatok olyan hálózatok, melyek kimenete nem csak a bemenetre kapcsolt kombinációtól függ, hanem az előző bementi kombinációk sorrendjétől is.
Pl. egy gyárban egy termék gyártásához négy munkafázis (a, b, c, d) áll rendelkezésre. Többfélét gyártanak a termékből, némelyiknél egy munkafázist kihagynak. Az egyes munkafázisok között a készülő terméket egy kis automata targonca mozgatja, melyet két vezérlőjellel irányítunk:

- „maradj a helyeden”; A = ’0’, B = ’0’.
- „lépj tovább”: A = ’1’, B = ’0’; a→b; b→c; c→d; d→a.
  Ha kész az adott munkafázis akkor A=’1’-el küldhetjük tovább. Ha a d munkafázissal végeztünk, akkor miután a kész darabot elvitték, alaphelyzetbe áll (d→a).
- „lépj vissza”: A = ’0’; B = ’1’; c→b; b→a.
  Ha az előző munkafázist nem elégséges módon hajtották végre. Az utolsó (d) munkafázisból nem léptetjük már vissza (mert pl az utolsó munkafázis olyan, hogy akkor már egyszerűbb a kész terméket feldarabolni és újrafeldogozni), és az elsőből (a) sem mozdul tovább visszaléptetéskor.
- „lépj kettőt”; A = ’1’, B = ’1’; a→c; c→a; b→d.
  Ha kihagyunk egy munkafázist, illetve, ha előről akarjuk kezdeni az egész munkát. A d munkafázisból szintén nem lép már vissza.

Ábrázoljuk irányított gráfként a munkafolyamatot, csúcspontkban az egyes munkafázisokkal:



Az ábrában a kék körökben csúcspontként az egyes munkafázisok vannak melyekek a targonca lehetséges állapotai, a nyilak a munkafolyamatok közötti lehetséges mozgásokat jelzik, a nyilak mellé írtuk azt a bemeneti kombinációt, amire a mozgás létrejön.
Látható, hogy ugyanarra a bemeneti kombinációra más-más mozgással reagálunk, attól függően, hogy épp melyik állapotnál tartunk.
A targonca reakcióit táblázatosan is megadhatjuk (x: az előírt x irányú mozgás; y: az előírt y irányú mozgás):

Eddigi állapot / AB	Következő állapot / x irány y irány			
	00	01	11	10
a	a/0 0	a/0 0	c/1 -1	b/1 0
b	b/0 0	a/-1 0	d/-1 -1	c/0 –1
c	c/0 0	b/0 -1	a/-1 1	d/-1 0
d	d/0 0	d/0 0	d/0 0	a/0 1

Ha a targonca vezérlését digitális úton kívánjuk megvalósítani, akkor az állapotokat és a kimeneteket bináris alakba kell alakítani. Ezt kódolásnak hívjuk.

Pl legyenek az állapotok:

a: 00 b: 01 c: 11 d: 10

Tehát 2 bit (nevezzük Q_1Q_2-nek) elég, hogy megkülönböztessük a lehetséges állapotokat. A lehetséges x jelek száma 3 (-1, 0, 1), ugyanennyi a lehetséges y jeleké is. Az x és az y jel külön-külön 2-2 biten kódolható (legyenek ezek X_1X_2 és Y_1Y_2).

Legyen pl '–1' = '10'; 0 = '00'; '1' = '01'

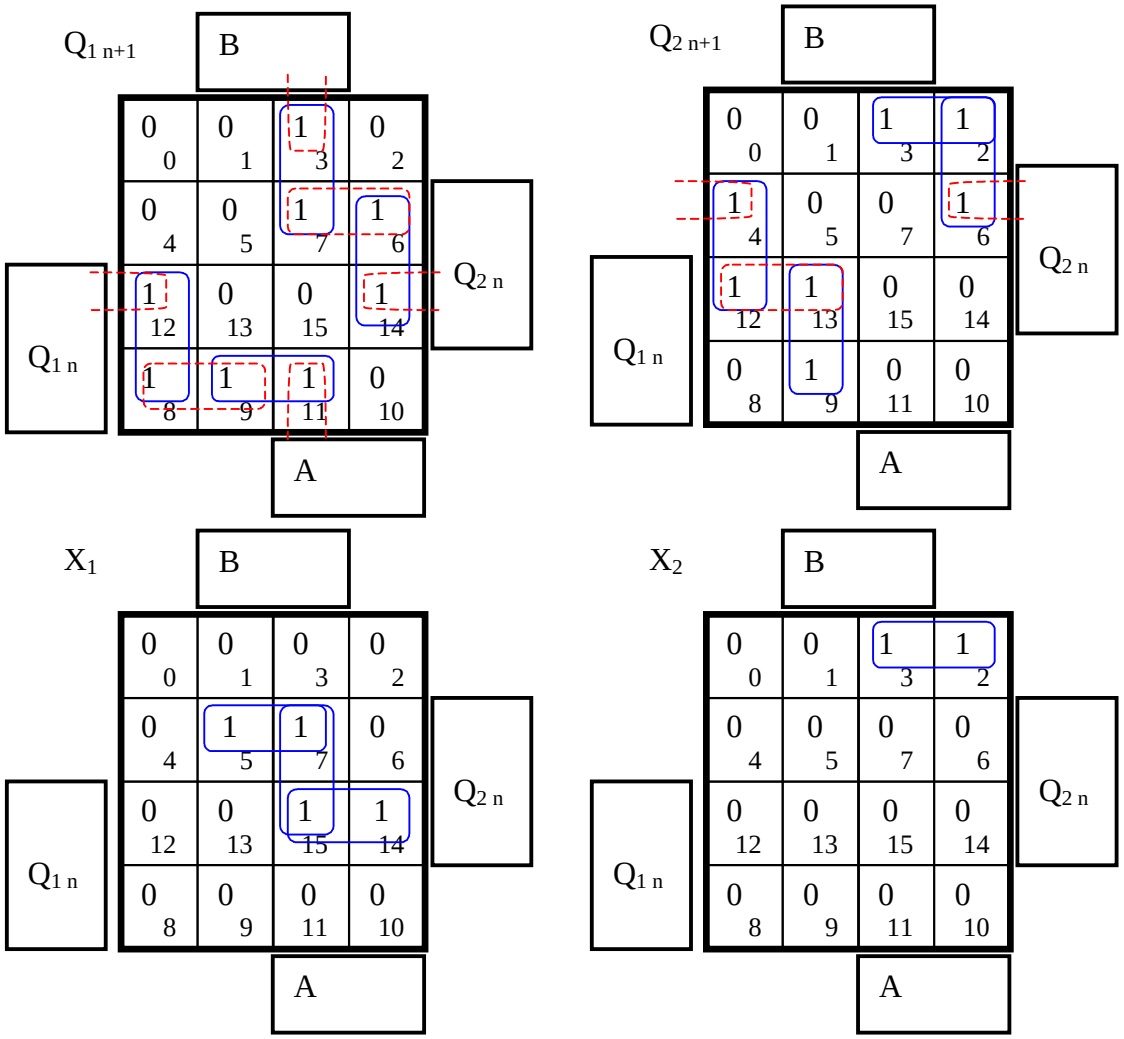
Ebből a bekódolt állapottábla ($Q_{1\,n}Q_{2\,n}$: az előző állapot kódja, $Q_{1\,n+1}Q_{2\,n+1}$: az új állapot kódja):

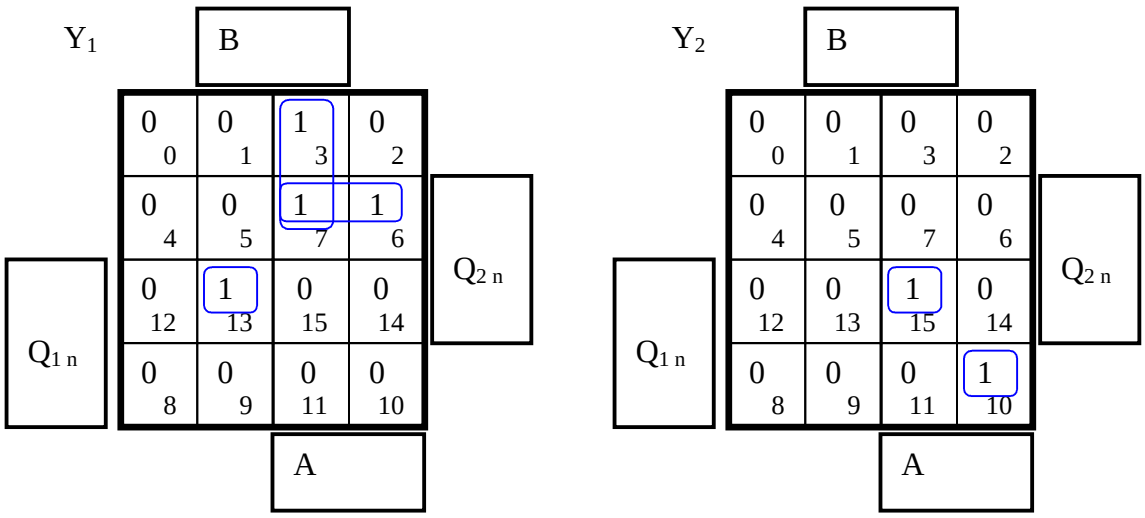
$Q_{1\,n}Q_{2\,n}$ AB	$Q_{1\,n+1}Q_{2\,n+1}$ / $X_1X_2Y_1Y_2$			
	00	01	11	10
00	00/00 00	00/00 00	11/01 10	01/01 00
01	01/00 00	00/10 00	10/10 10	11/00 10
11	11/00 00	01/00 10	00/10 01	10/10 00
10	10/00 00	11/00 00	10/00 00	00/00 01

$Q_{1\,n}$ (sor), $Q_{2\,n}$ (oszlop); **B** (oszlop), **A** (sor)

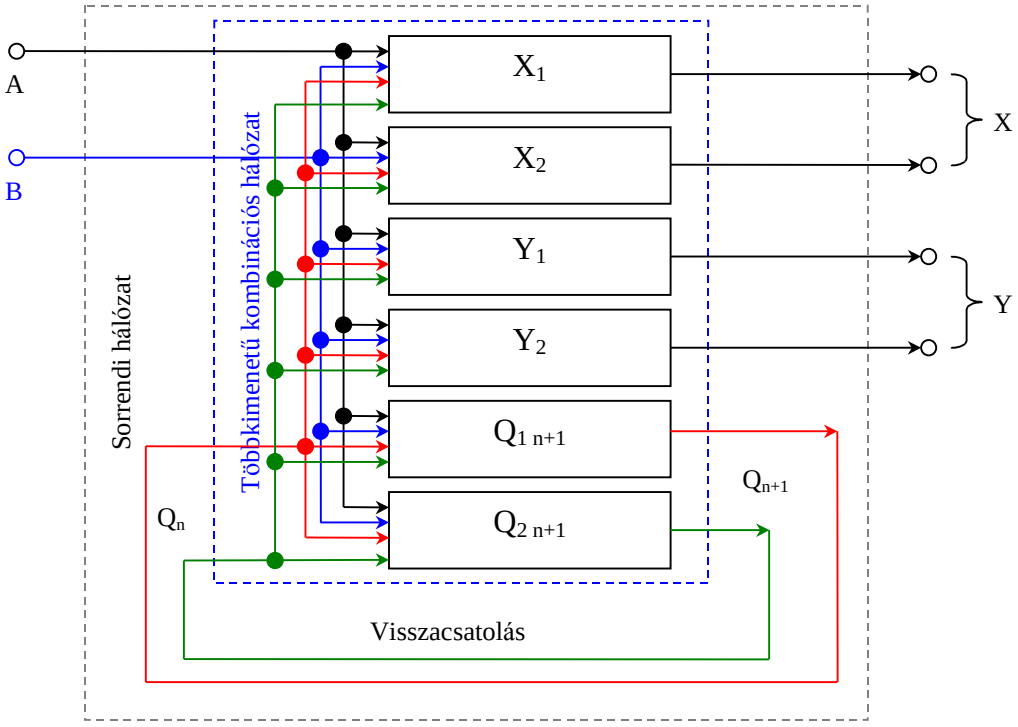
Ha figyelmesen megnézzük, akkor az állapottábla a kékkel berajzolt peremezés szerinti Karnaugh táblával egyezik meg, melybe egyszerre több függvény kimentét beírtuk.

Ebből az egyes változók Karnaugh táblái:





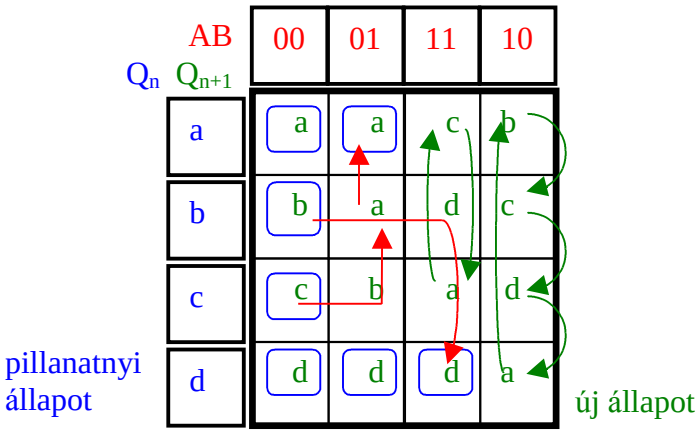
A Karnaugh táblák alapján meghatározhatóak az egyes kimenetek egyszerűsített logikai függvényei. $Q_{1\ n+1}$ és $Q_{2\ n+1}$ Karnaugh táblájában a piros szaggatottal összevont mintermeket hazárdmentesítés céljából célszerű megvalósítani (a hazárdok elméletébe most nem megyek bele részletesen, az egyes kapuelemek késéséből származó hibát okozó tranziens jelenségről van szó).
Az egyes megvalósított kimenetek felhasználásával a megvalósított sorrendi hálózat blokkvázlata a következő:



Látható, hogy a sorrendi hálózat tulajdonképpen egy visszacsatolt többkimenetű kombinációs hálózat. Az állapot a példánkban nincsen külön kivezetve, hanem a bemenő jellel együtt további kombinációs hálózatra vezetve egy az állapotból származtatott kimeneti jelet kapunk. Ez nem általános, van amikor a tulajdonképpeni kimenet maga a pillanatnyi állapot.

Az állapottábla felrajzolásával néhány sorrendi hálózatokkal kapcsolatos fogalmat mutathatunk be:

1. **stabil állapot**: a következő állapotot az AB bemenet és a pillanatnyi állapot határozza meg. Ha a bemeneti kombináció hatására állapotváltozás nem történik, akkor stabil állapotban vagyunk. Ismertetőjele: az van a rublikában, ami a sor elején (kék négyzetek)
2. **gerjedés**: ha az adott bemeneti kombináció és pillanatnyi állapot mellett mellett nem alakulhat ki stabil állapot (zöld nyíl).

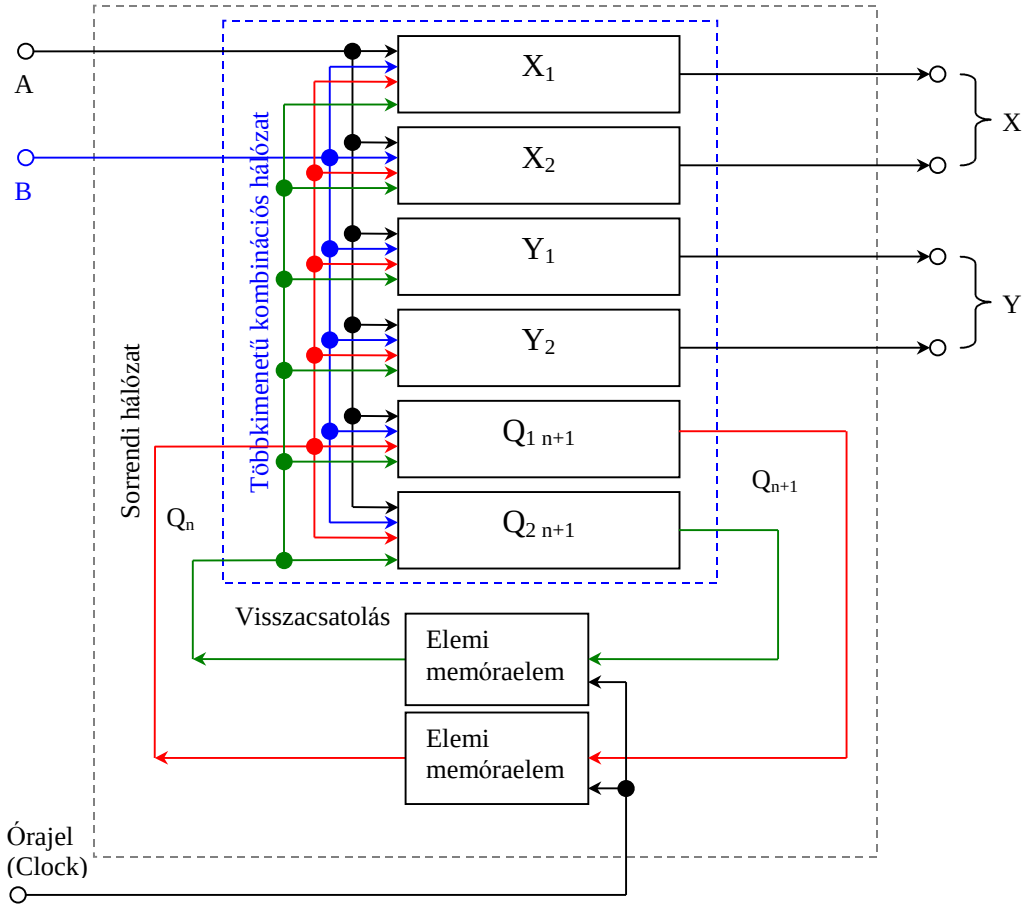


Legyen pl. kiinduláskor AB = 00! Ekkor minden állapot stabil. Legyünk pl a „c” állapotban, és váltson a bemenet AB = 01-re! Ekkor a piros nyíl mentén első lépésben „b” állapotba kerülünk (targonca „c”-ből „b”-be megy). Ha az állapotváltozás után AB 00-ra áll vissza akkor maradunk „b”-ben. Ha a bemenet AB = 01 marad, a piros nyíl mentén „a” állapotba kerülünk (targonca megállás nélkül továbbmegy „a”-hoz). Most már AB akár 00 akár 01 lehet, „a”-ban maradunk (targonca áll). Ha most a AB 11-re vált, akkor „c”-be jutunk (targonca „a”-ből „c”-be megy). Ha az állapotváltozás után AB 00-ra áll vissza akkor maradunk „c”-ben. Ha a bemenet AB = 11 marad, a piros nyíl mentén „a” állapotba kerülünk (targonca megállás nélkül visszamegy „a”-hoz). Ez az oda-vissza ciklus mindaddig fennmarad, amíg AB = 11 (gerjedés). Ha „b”-ből inulunk AB = 00 nál AB = 11-re váltva az új állapot „d” lesz és ott megállunk. Innen addig nem mozdul, amíg AB = 10 bemenet nem érkezik. Tehát a stabilitás és a gerjedés nem csak bemeneti kombinációtól, hanem kiindulási állapottól is függ.

Ha azt akarjuk elérni, hogy egyszerre csak egy lépés hajtódjék végre, (azaz a targonca meg is álljon minden egyes köztes állapotnál), akkor a hálózatot kívülről szinkronizálni kell.
A külső szinkronizáció azt jelenti, hogy a hálózatba egy periodikusan ismétlődő jelet (órajelet) vezetünk, melynek segítségével a hálózat nem folyamatosan, hanem azonos időközökben vizsgálja meg a bemeneteit és az állapotát, majd ez alapján kerül az új állapotba és adja ki az új kimneti kombinációt.
Szinkronizált hálózatban a ki és bemenetek is csak az órajel által meghatározott időpontban érvényesek, azon kívül bármely értéket felvehetnek (a hálózat illetve az összes többi hálózat ekkor úgysem veszi figyelembe a be- és kimeneteket).
Szinkronizált hálózatban **egy órajelciklus** alatt csak **egy** állapotváltozás történik.

A szinkronizálhatósághoz szükség van az órejel által meghatározott időpontban beálló állapot megjegyzésére, tulajdonképpen a hálózatba memóriaelemeket kell beépíteni.

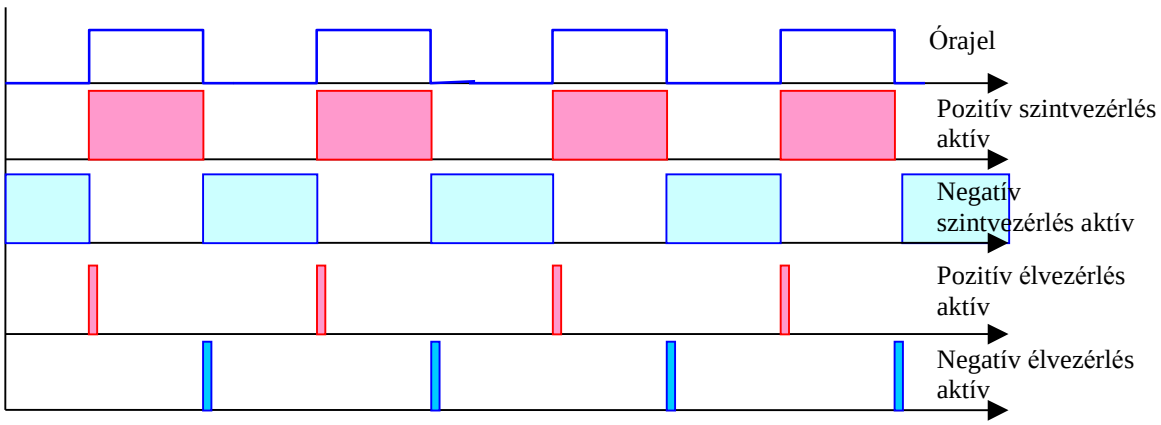
A példahálózatunk szinkronizált változatban:



A memóriaelemek az órajel bizonyos pillanataiban megvizsgálják a bemenetet, és egy perióduson keresztül megőrzik annak értékét. Egy periódus múlva ismét megvizsgálják a bementeiket, és megórzik annak értékét a következő periódusig.

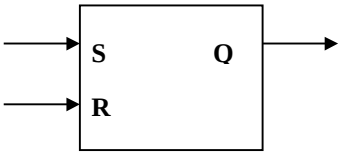
Annak függvényében, hogy az órajel mely pillanataiban vizsgálják meg a bemeneteiket, a szinkron sorrendi hálózatokat a következő főbb osztályokba sorolhatjuk:

- **Szintvezérelt**: Az órajel „1” (pozitív szintvezérlés) vagy „0” (negatív szintvezérlés) értékű félperiódusa alatt a hálózat aszinkron módon üzemel, majd a másik félperiódusban a beállítódott értéket tárolja.
- **Élvezérelt**: Az órajel „0” → „1” (pozitív élvezérlés) vagy „1” → „0” tranziens átmeneti ideje alatt vizsgálja meg a bemenetet, majd a következő órajel-élig tárolja.



Elemi memóriaelemek

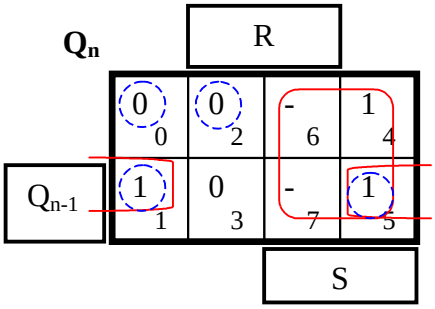
Aszinkron RS tároló (aszinkron RS flip-flop)



Az RS tároló olyan sorrendi hálózat, mely két bemenettel rendelkezik, a kimenete megegyezik az állapotával.
Kiindulás RS = 00. Az S (set: beállít, beír) bemenetre kapcsolt „1” a kimenetet „1”-be állítja, ami az S bemenet „0”-ba visszaállása után is így marad. A kimenetet az R (reset: törlés) bemenetre kapcsolt „1” állítja vissza „0”-ba, ami az R bemenet „0”-ba visszaállása után is így marad. Az RS = 11 vezérlés tiltott vezérlés (egyszerre irjon és töröljön?).
A fentiek alapján az RS flip-flop működési-, igazság- és Karnaugh táblája (Q_{n-1}: az n. ütem előtti kimenet, Q_n: kimenet az n. ütem után):

S	R	Q_n
0	0	Q_{n-1}
0	1	0
1	0	1
1	1	tiltott

S	R	Q_{n-1}	Q_n
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

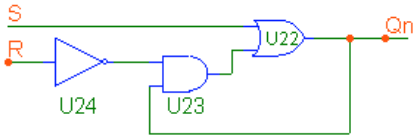


A bejelölhetjük a stabil állapotokat (kék szaggatott kör) a Karanaugh táblán.
A kimenet egyszerűsített logikai függvénye a Karanaugh tábla alapján:

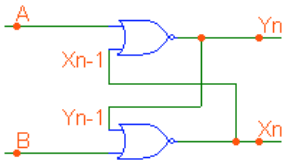
$$Q_n = S + Q_{n-1} \cdot \overline{R}$$

Feltéve ha $A \cdot B = 0$ (azaz AB ≠ 11)

A logikai függvény alapján a hálózat:



A gyakorlatban nem ezt az aszimmetrikus kapcsolást, hanem az alábbi szimetrikus kapcsolást alkalmazzák:



Ha felírjuk a kimenetek logikai függvényeit:

$$Y_n = \overline{A + X_{n-1}} = \overline{A + \left(\overline{B + Y_{n-1}}\right)} = \overline{A + \left(\overline{B} \cdot \overline{Y_{n-1}}\right)}$$ azaz:

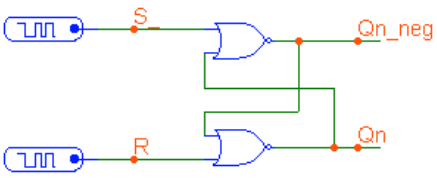
$$\overline{Y_n} = A + \overline{B} \cdot \overline{Y_{n-1}}$$

Ha összehasonlítjuk az RS tároló eredeti logikai függvényével, akkor látható, hogy Q_n szerepét $\overline{Y_n}$, S szerepét A és R szerepét B látja el.

$$X_n = \overline{B + Y_{n-1}} = \overline{B + \left(\overline{A + X_{n-1}}\right)} = \overline{B} \cdot \left(A + X_{n-1}\right) = \overline{B} \cdot A + \overline{B} \cdot X_{n-1}$$

Kihasználjuk, hogy $A \cdot B = 0$, ekkor AB a következő értékeket veheti fel:

AB	Ekkor	$A \cdot \overline{B}$		
0 0	$\Rightarrow$	0·1 =	0 =	A
0 1	$\Rightarrow$	0·0 =	0 =	A
1 0	$\Rightarrow$	1·1 =	1 =	A

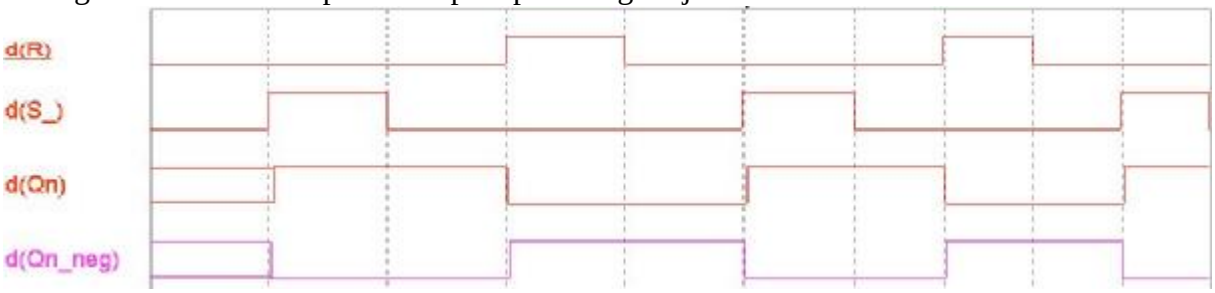


Így a bemenetre tett feltételeink mellett $A \cdot \overline{B} = A$, ezért:

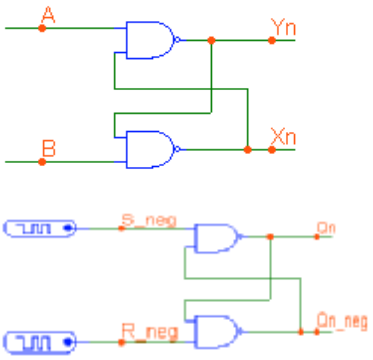
$X_n = A + \overline{B} \cdot X_{n-1}$

Ha ezt hasonlítjuk össze az RS tároló eredeti logikai függvényével, akkor látható, hogy Q_n szerepét X_n, S szerepét A és R szerepét B látja el. Látható még, hogy $X_n = \overline{Y_n}$.

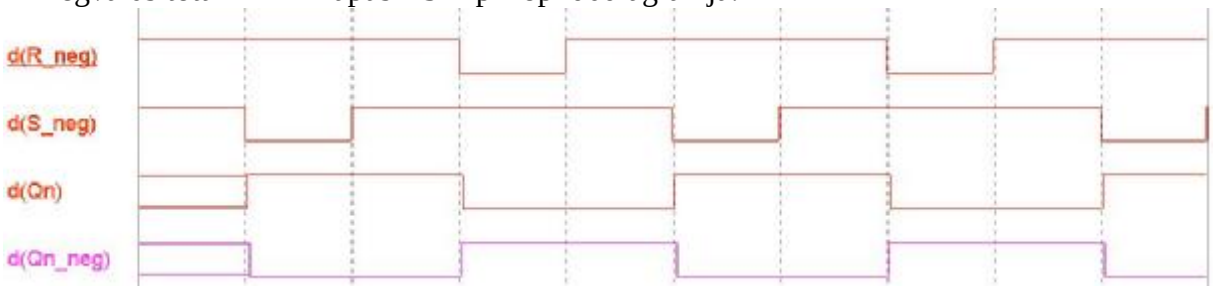
A megvalósított NOR kapus RS flip-flop idődiagramja:



A másik gyakorlatban megvalósított szimmetrikus kapcsolás NAND kapukkal építhető meg. A levezetést az előzőekhez hasonlóan kell megcsinálni, de itt az jön ki, hogy az A az S negáltjának a szerepét tölti be, B az R negáltjának a szerepét, Y_n a Q_n szerepét és X_n a Q_n negáltjának a szerepét.



A megvalósított NAND kapus RS flip-flop idődiagramja:

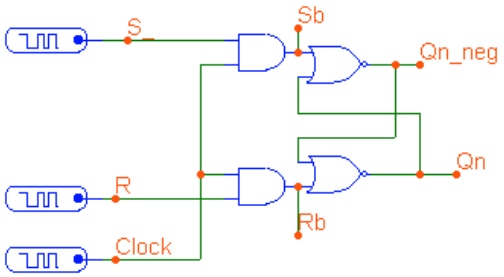


Szinkron RS flip-flop

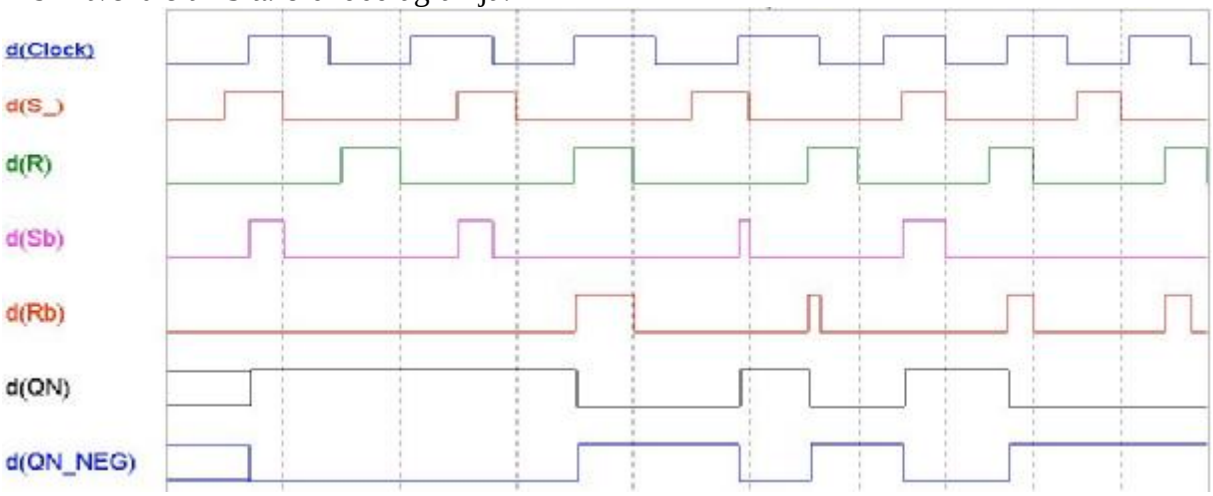
A szinkron RS flip-flop háromféle vezérléssel valósítható meg:

- Szintvezérlés
- Master-Slave
- élvezérlés

Szintvezérelt RS tároló

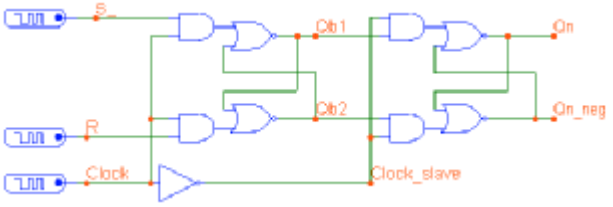


Működése: egy külső órajelet bevezetünk a hálózatba, melyből kikapuzzuk azt az időtartamot, amikor a flip-flop-nak aktívnak kell lennie, azaz reagálni a bemenetre. Az ábrán egy pozitív élvezérelt RS flip-flop látható: a bemeneti AND kapuk az órajel „0” ideje alatt mindkét bemenetre „0”-t adnak.
Így az RS flip-flop az órajel „0” ideje alatt a legutolsó értéket tárolja. A bemeneti AND kapuk az órajel „1” ideje alatt adják a bemeneti jelet az RS flip-flop bemenetére, így minden változás is csak az órajel „1” ideje alatt lehetséges.
A szintvezérelt RS tároló idődiagramja:

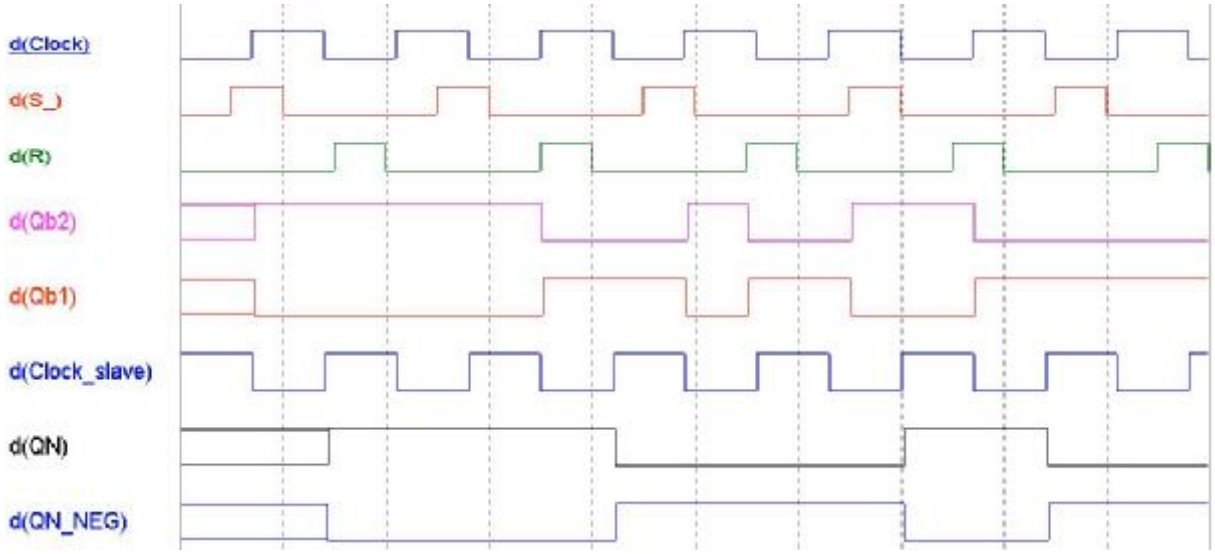


A szinkron hálózatoktól azt várjuk el, hogy egy órajelciklus alatt csak egy állapotváltozást hajtson végre. A diagramban is látható azonban, hogy pl. a negyedik órajelciklusban történik egy ’set’ és még ugyanabban a ciklusban történik egy ’reset’ is. Ezért az élvezérelt flip-flop-kat ott használják, ahol az órajel biztosan gyorsabb a bemenő jeleknél, illetve ott, ahol ez a kettős állapotváltozás nem okoz hibás működést.

Master-Slave RS tároló

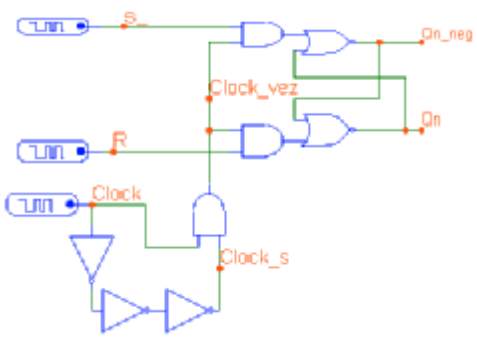


A szintvezérelt RS tároló hátrányát szűri ki két lépcsőben. Az ábrán is látható, hogy tulajdonképpen két egymás után kapcsolt élvezérelt RS flip-flop-ból áll, melyek ellentétes órajelet kapnak. Így amikor a második flip-flop (slave) aktív, az első flip-flop kimenete már állandósult.
A sorba kapcsolás miatt a kimeneten felcserélődnek a Q_n és Q_n negált szerepét ellátó kimenetek.
A kimeneten egy órajel alatt maximum csak egy változás történhet, a Master-Slave RS tároló kimenetén egyáltalán nem jelenik meg az az átmeneti jel-tranziens, ami az élvezérelt esetben a 4. órajelciklusnál előállt.
Ez látható az idődiagramon is:

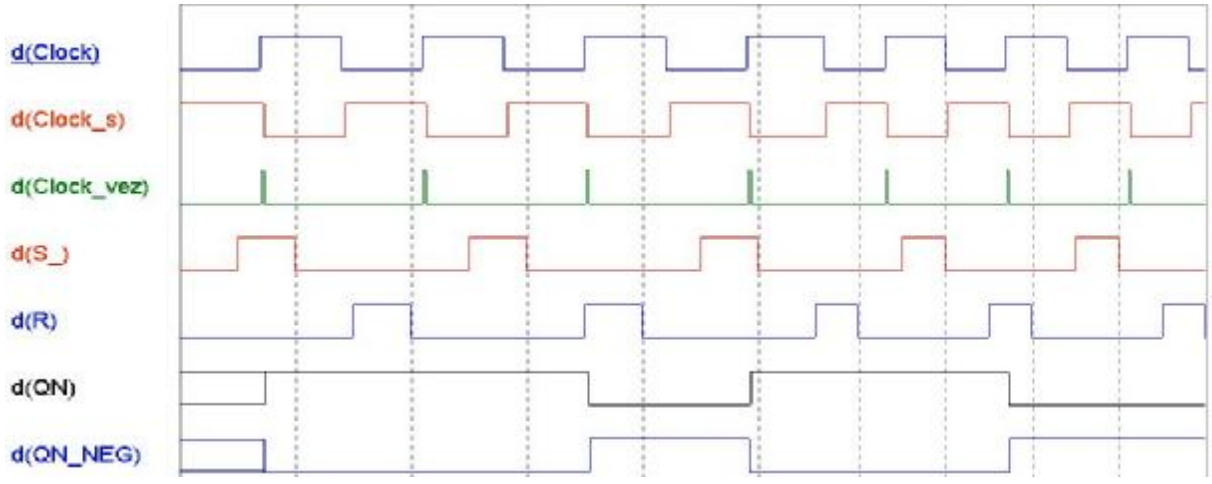


Élvezérelt RS tároló

Működése: tulajdonképpen ugyanúgy működik, mint az élvezérelt RS flip-flop, csak az órajelből egy áramkör a kapukésleltetéseket kihasználva egy nagyon rövid időtartamú kapuzójelet állít elő. Az adott áramkör működése az időtartam alapján követhető vissza könnyen.
(Megj: élvezérlés előállítására nem csak ez a módszer létezik, összetett flip-flopokkal megoldható késleltetőhálózat nélkül is. A késleltetési idő ugyanis függ a környezeti paraméterektől, hőmérséklet stb., ezért az előálló impulzus szélessége változó).



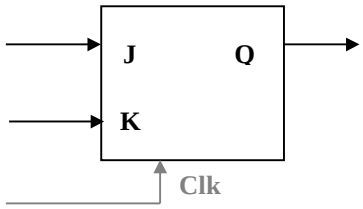
Az időtartam:



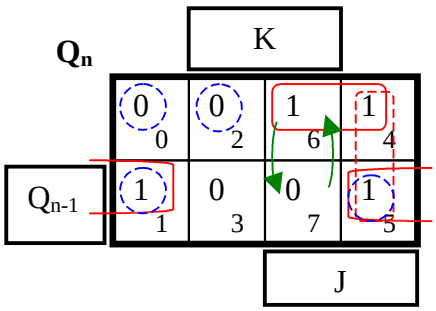
JK tároló (flip-flop)

Működési táblája megegyezik az RS flip-flop-éval, annyi kivétellel, hogy az bemeneti 11 vezérlés is enngedélyezve van, ekkor az új állapota a pillanatnyi állapot negáltja. A kimenet szintén megegyezik az állapotával. A J bemenet a beírás a K bemenet a törlés (talán K, mint kill?, ezekre nem tudok egy használható elnevezést kitalálni)

J	K	Q_n
0	0	Q_{n-1}
0	1	0
1	0	1
1	1	$\overline{Q_{n-1}}$



J	K	Q_{n-1}	Q_n
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



Látható, hogy JK = 11 vezérlés mellett gerjedés lép fel (züld nyilak), ezért a JK flip-flopokat a gyakorlatban csak szinkron változatban alkalmazzák (aszinkron esetben JK = 11 mellett az áramkör sebességétől függő periódusidővel 1-0-1-0 jelsorozatot adna ki, ami a periódusidő környezetfüggése miatt előre meghatározhatatlan kimeneti állapotot idéz elő a JK = 11 vezérlés megszűnte után; míg szinron esetben egy órajelciklus alatt csak egy állapotváltozás történik).
Az RS tárolónál látott módon a J-K tárolót is meg lehet valósítani az egyszerűsített alakjából, de a gyakorlatban RS tárolóból és az RS tároló elé kapcsolt kombinációs hálózatból alakítják ki. A kombinációs hálózat megtervezéséhez fel kell rajzolni a JK tároló működési tábláját, majd ez mellé az RS tároló működési táblájának azon sorait, amelyek az adott Q_{n-1} - Q_n átmenethez tartoznak:

	J	K	Q_{n-1}	Q_n	S	R
0	0	0	0	0	0	-
1	0	0	1	1	-	0
2	0	1	0	0	0	-
3	0	1	1	0	0	1
4	1	0	0	1	1	0
5	1	0	1	1	-	0
6	1	1	0	1	1	0
7	1	1	1	0	0	1

S	R	Q_{n-1}	Q_n
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

A határozatlan bejegyzések olyan állapotváltozásokhoz vannak bejelölve, ahol az előírt állapotváltozást az RS tároló két különböző (de szomszédos, azaz csak az egyik bemenet értékébenben eltérő) bemeneti kombinációja is előidézi. Pl. 1. sor: az előző állapot ’1’ az új

szintén ’1’, ezt az RS = 00 (tartsd a régi állapotot) illetve az RS = 01 (állj ’1’-be) is előidézheti. Tehát mindegy, hogy S milyen, a lényeg, hogy R ’0’ legyen.
A kombinációs hálózat Karnaugh táblái:

S

Q_{n-1} \ K	0	0	1	1
	0	2	6	4
-	0	0	-
1	3	7	5

J

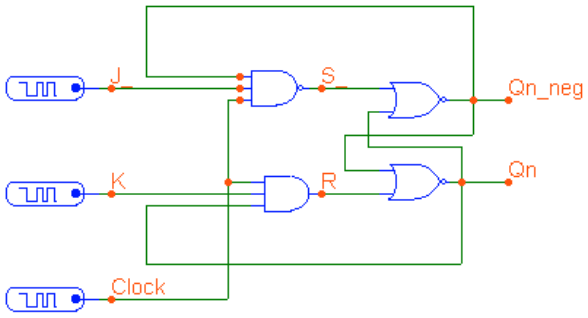
R

Q_{n-1} \ K	-	-	0	0
	0	2	6	4
0	1	1	0
1	3	7	5

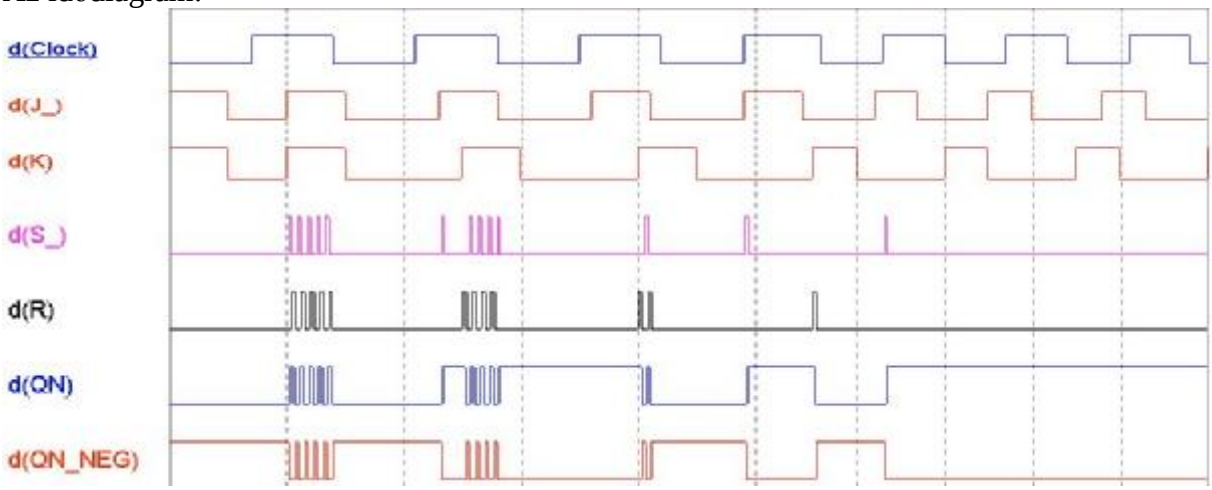
J

Tehát az S bemetre a $J \cdot \overline{Q_{n-1}}$ -t kell vezetni, az R bemenetre $K \cdot Q_{n-1}$-t:

Látható, hogy a megvalósított JK flip-flop „lelke” egy aszinkron RS flip-flop, melyet két hárombemenetű AND kapuval szinkronizálunk és JK tárolóvá alakítunk.
Az S bemenetre vezetett $J \cdot \overline{Q_{n-1}} \cdot Clock$ az órajel ’1’ értéke alatt csak akkor engedélyez írást (S =1-t), ha a kimenet ’0’ (ha a kimenet ’1’ akkor amúgy sem kell írni.).



Az S bemenetre vezetett $K \cdot Q_{n-1} \cdot Clock$ az órajel ’1’ értéke alatt csak akkor engedélyez írást (R =1-t), ha a kimenet ’1’ (ha a kimenet ’0’ akkor amúgy sem kell törölni.).
JK = 11 esetén, ha a kimenet ’0’, akkor RS = 01; ha a kimenet ’1’, akkor RS = 10; így JK = 11 esetén valóban állapotváltás történik.
Az időidiagram:



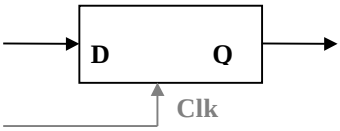
Látható, hogy az S és az R bemenet azonnal visszavált ’0’-ba, amint megtörtént a beírás, vagy törlés. Az is látható, hogy a szintvezérlés előnytelen a JK flip-flopnál, mert az órajel ’1’ ideje alatt aszinkron módon üzemel és JK = 11 esetén gerjed (0→1→0→1→0→1→…), emiatt a JK flip-flopokat általában élvezérelt kivitelben valósítják meg.

D tároló

Olyan egy bemenetű és egy kimenetű sorrendi hálózat, melynek működési táblája a következő (kimenete megegyezik az állapotával):

D	Q_n
0	0
1	1

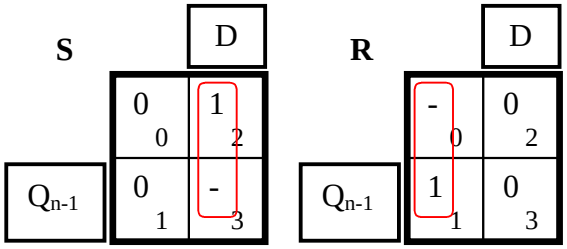
D	Q_{n-1}	Q_n
0	0	0
0	1	0
1	0	1
1	1	1



S	R	Q_{n-1}	Q_n
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

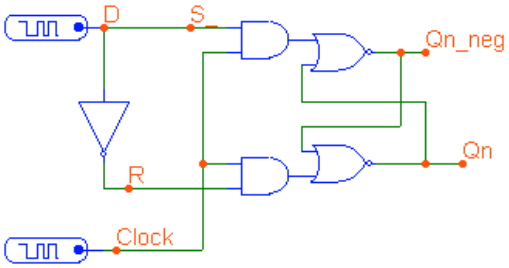
Látható, hogy aszinkron üzemmódban a D flip-flop időkésés nélkül lemásolja a bemenetet, tehát az aszinkron D flip-flop funkcióját maradéktalanul ellátja egy darab vezeték! A gyakorlatban a szinron D flip-flopnak van jelentősége, a már tárgyalt szinkron sorrendi hálózatok állapotvisszacsatolásában ezeket alkalmazzák. Lényeges: a szinkron D flip-flop órajel által engedélyezett időpontokban (időintervallumokban) másolja le a bemenetet, egyébként tartja az utolsó állapotát.
A D tárolót a gyakorlatban szintén RS tárolóval valósítják meg. A bemenetekre a JK tárolónál létott módon a következő kombinációs hálózatot kell kapcsolni (az RS tároló-igazságtáblázat $Q_{n-1} \rightarrow Q_n$-nek megfelelő sorai alpján, lásd fent):

D	Q_{n-1}	Q_n	S	R
0	0	0	0	-
0	1	0	0	1
1	0	1	1	0
1	1	1	-	0

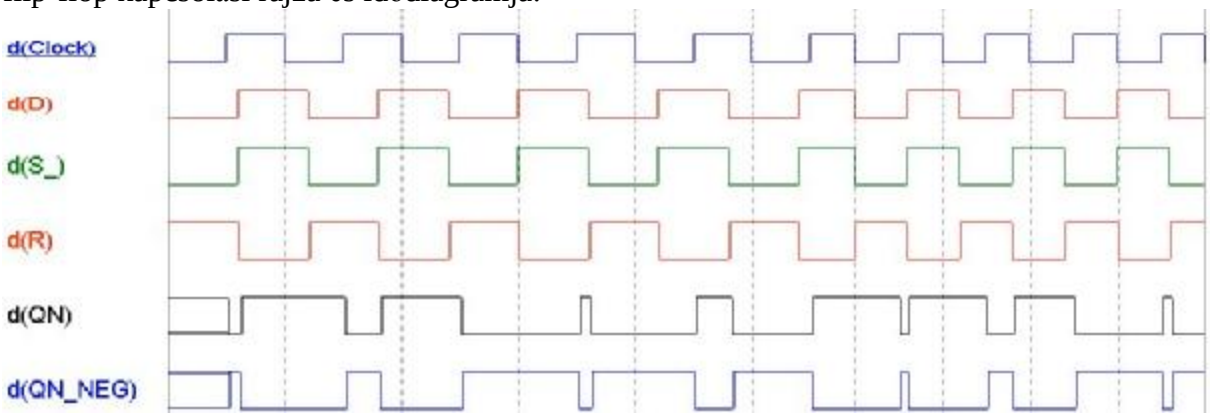


A Karnaugh táblák és az egyszerűsített logikai függvények a minimalizálás után:

$$S = D \qquad R = \overline{D}$$



A megvalósított szintvezérelt szinkron D flip-flop kapcsolási rajza és időddiagramja:

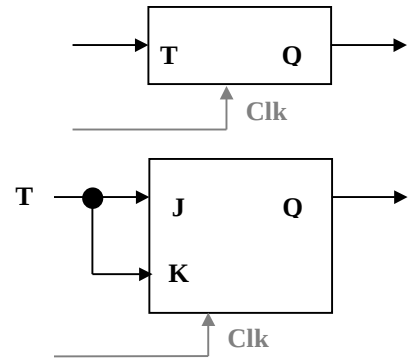


T tároló

Olyan egy bemenetű és egy kimenetű sorrendi hálózat, melynek működési táblája a következő (kimenete megegyezik az állapotával):

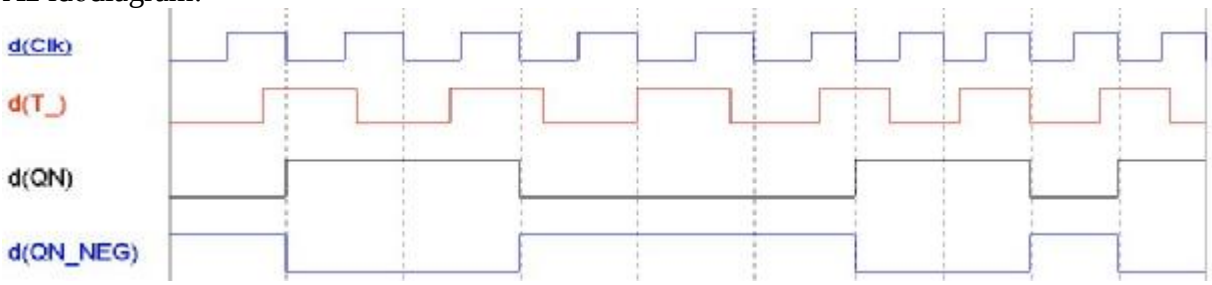
T	Q_n
0	Q_{n-1}
1	$\overline{Q_{n-1}}$

T	Q_{n-1}	Q_n
0	0	0
0	1	1
1	0	1
1	1	0



Aszinkron üzemmódban a T flip-flopot sem lenne értelme használni, mert T = ’1’ esetén, (mint a JK flip-flopnál JK = 11-nél) az áramkör sebességétől függő periódusidejű gerjedés lépne fel a kimeneten. Ezért a T flip-flopot is élvezérelt szinkron üzemmódban alkalmazzák. A gyakorlatban JK flip-floppal valósítják meg, gyanis a JK bemeneteket azonosan vezérelve (rövidre zárva) csak a JK = 00 és a JK = 11 bemeneti kombináció léphet fel; a JK = 00 bemenetre $Q_n = Q_{n-1}$, a JK = 11 bemenetre $Q_n = \overline{Q_{n-1}}$, így a J = T és K = T (fent).

Az időidiagram:

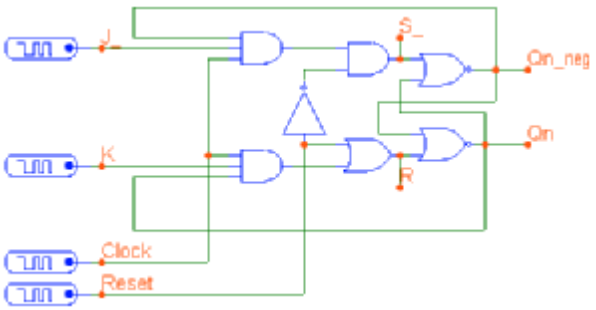


Ez most egy negatív élvezérelt (lefutóélre érzékelő) T flip-flop időidiagramja (ha a kiindulási állapot ’0’).

Minden memóraáramkör az aszinkron RS flip-flopra épül, amely előállítja a kimenet mellett annak negáltját $(\overline{Q_n})$, ezért minden megvalósított flip-flop rendelekzik negált kimenettel is.

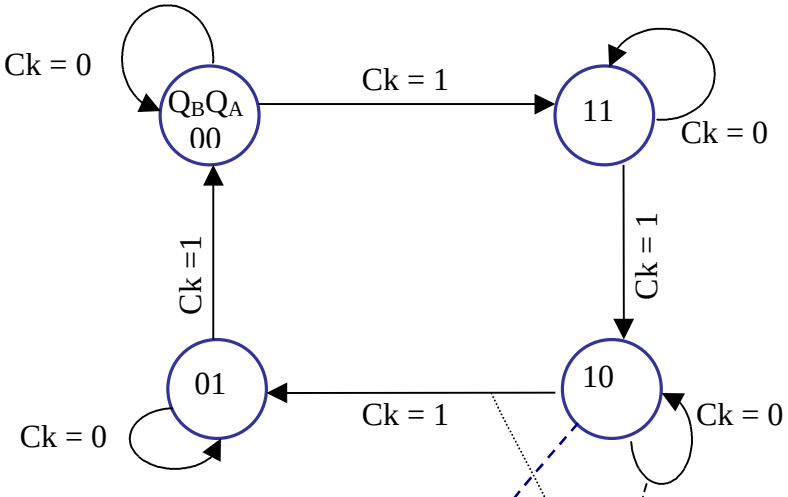
Bizonyos alkalmazásokban szükség lehet az egyes alkalmazott flip-flopok működésétől független törlésére illetve írására (ez azt jelenti, hogy a bemeneti jelektől független). (Pl. a T flip-flop kiindulási állapota csak így állítható be.) Ezért a gyakorlatban megvalósított flip-flopok további bemeneteket kapnak: Reset = törlés (nem azonos az RS flip-flop R bemenetével) és estleg Preset = írás bementeket. Ezt olyan további kombinációs hálózattal valósítják meg ami pl. Reset = ’1’ re leválasztja a belső RS flip-flopot a bemeneteiről, és RS = 10 bemenetet ad neki.

Pl. JK flip-flop Reset bemenettel:
(az S bemenet előtt álló AND kapu csak akkor engedi tovább a J bemenetről érkező jelet, ha a Reset bemenet ’0’, az R bemenet előtt álló OR kapu mindenképpen ’1’-t ad R-re, ha a Reset bemenet ’1’.) A Preset bemenet hasonlóképpen valósítható meg.

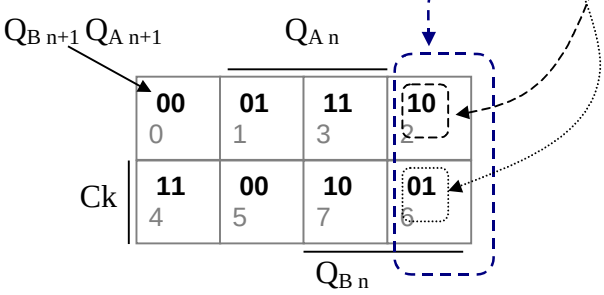


Példa: realizáljuk a következő állapottáblát megvalósító áramkört:

a. alapkapukkal
b. T tárolókkal

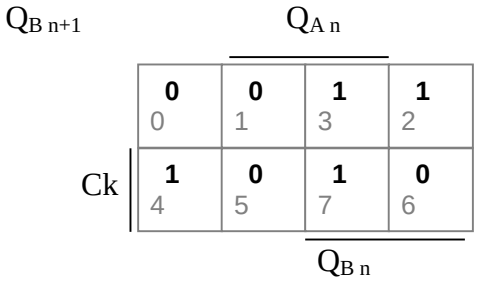


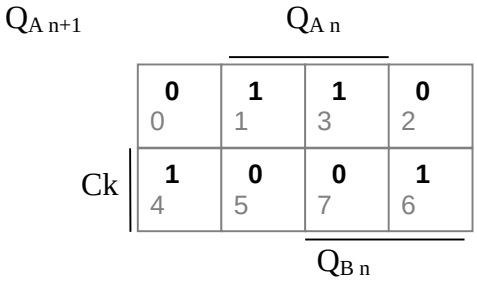
Két állapotváltozónk van, +egy bemeneti változónk => háromváltozós V-K táblákat kell felírnunk, melyben a változók:
Ck
Q_B pillanatnyi állapota, $Q_{B\,n}$
Q_A pillanatnyi állapota; $Q_{A\,n}$
az eredmények pedig
Q_B következő állapota, $Q_{B\,n+1}$
Q_A következő állapota $Q_{A\,n+1}$.



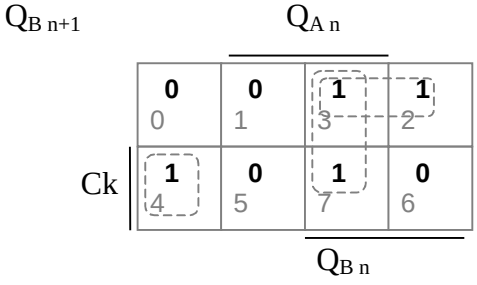
Vegyük fel először a hálózat kódolt állapottábláját (*a két eredmény közös V-K táblája*), kitöltjük az állapotgráf alapján.

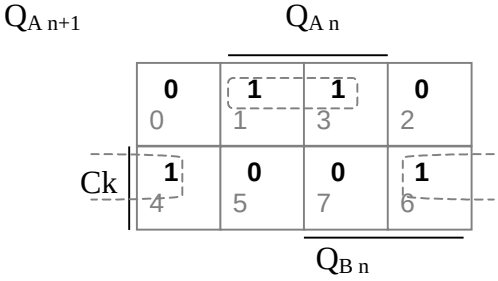
Majd bontsuk szét a két változó szerint.





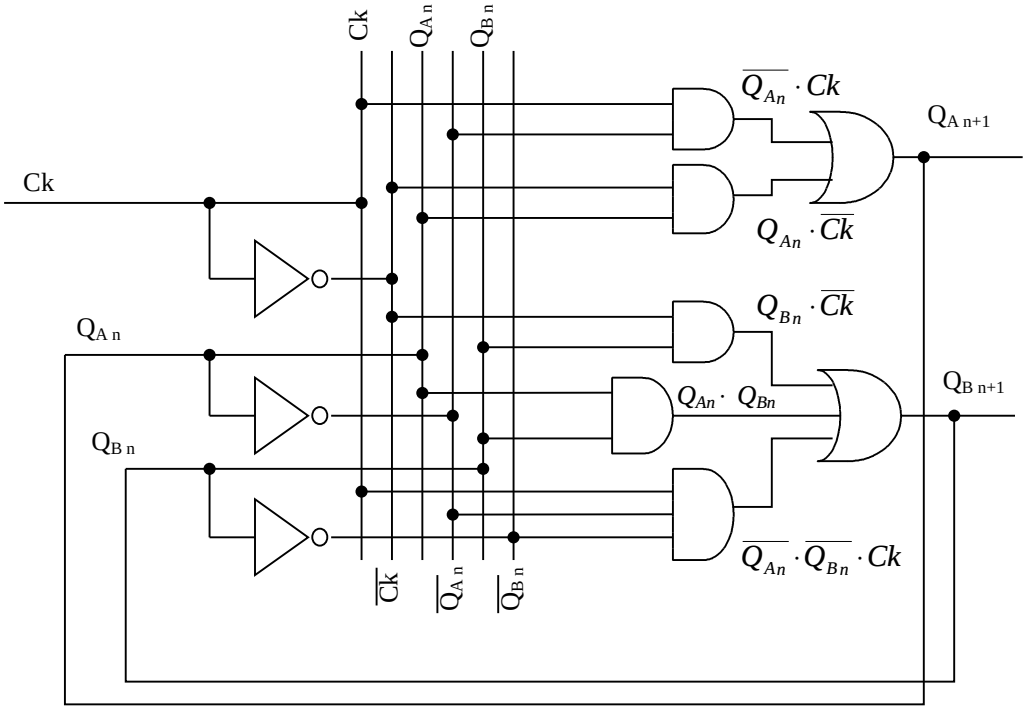
a. minimalizáljuk a V-K táblák alapján



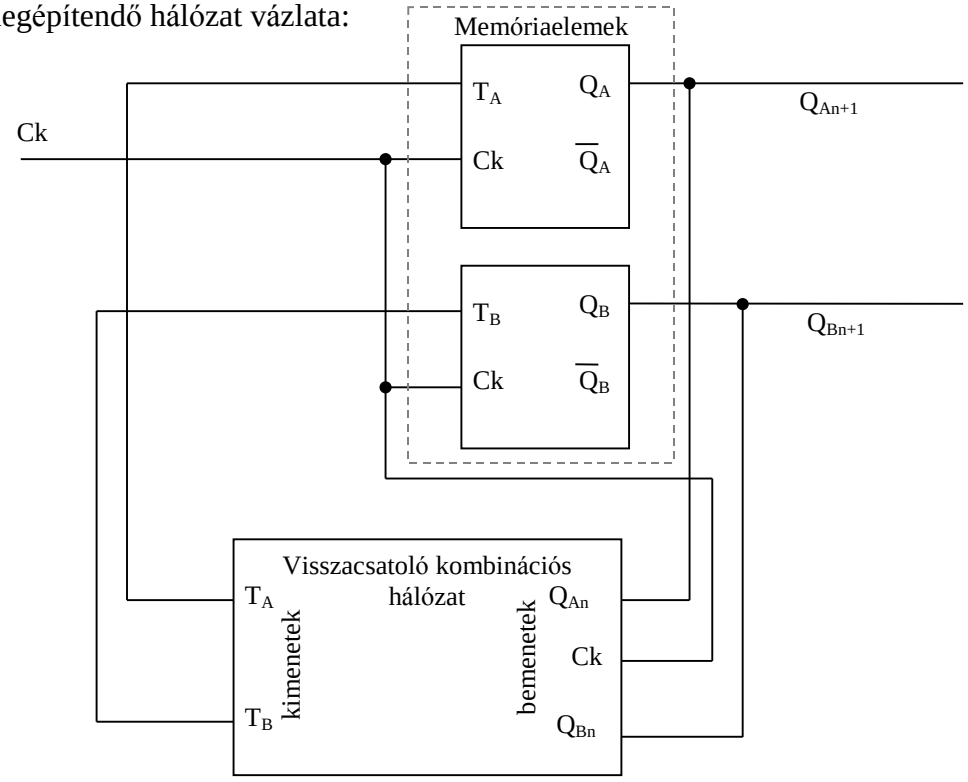


$$Q_{Bn+1} = \overline{Q_{An}} \cdot \overline{Q_{Bn}} \cdot Ck + Q_{An} \cdot Q_{Bn} + Q_{Bn} \cdot \overline{Ck}$$

$$Q_{An+1} = Q_{An} \cdot \overline{Ck} + \overline{Q_{An}} \cdot Ck$$

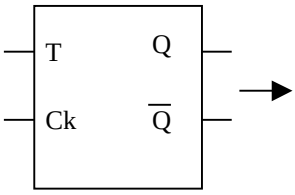


b. A megépítendő hálózat vázlata:



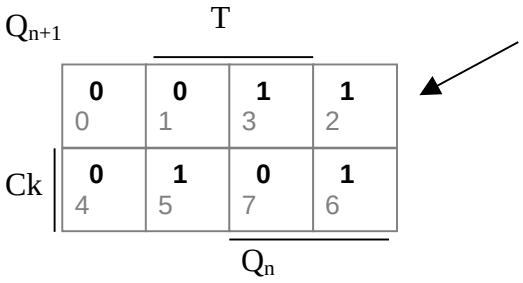
A tulajdonképpeni feladat a visszacsatoló kombinációs hálózat meghatározása, amelyit az elvárt működési módnak megfelelően vezérli a T tárolók bemeneteit.

Ha a T tároló működési tábláját kiegészítjük a Ck órajellel, megkapjuk az igazságtábláját, amiből felírhatjuk az állapot tábláját (*tulajdonképpen a V-K tábláját*):

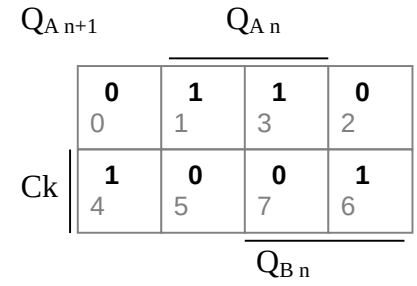


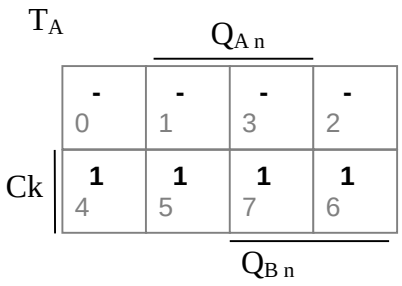
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

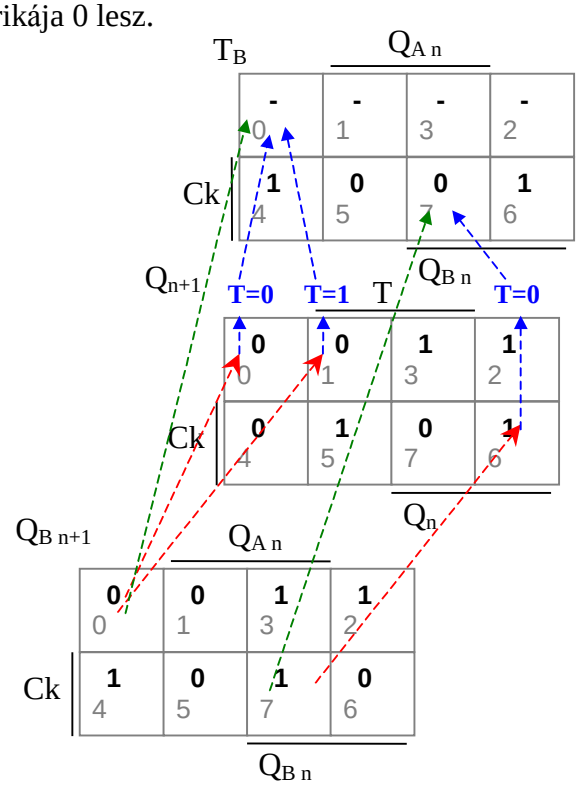
Ck	T	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



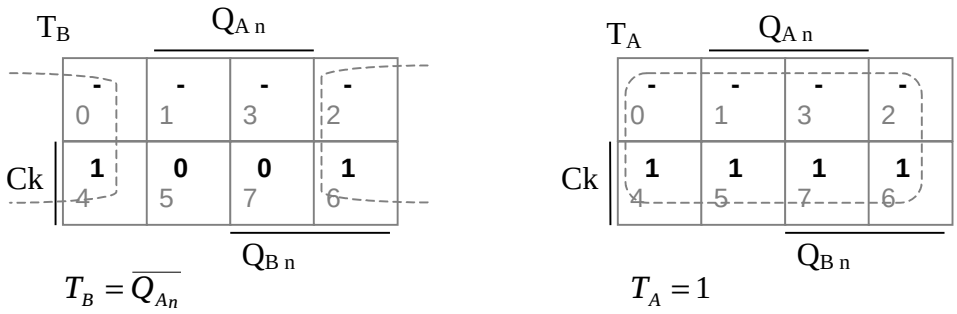
A kapott V-K táblát vessük össze az állapot-változók V-K tábláival.
Pl. $Q_{B\,n+1}$ –nek megfelel Q_{n+1}, $Q_{B\,n}$-nek megfelel Q_n, Ck-nak Ck.
Látható, hogy pl. $Q_{B\,n+1}$ ’0’ rublikája (*Ck=0, $Q_{B\,n}$=0, $Q_{B\,n+1}$=0*) megfelel a Q_{n+1} ’0’ és ’1’ rublikájának (*Ck=0, Q_n=0, Q_{n+1}=0*). Ez alapján T_B ’0’ rubrikája lehet 0 is 1 is, nincs hatással a $Q_{B\,n+1}$ kimenetére az adott $Q_{Bn}Q_{An}Ck$ bemeneti kombináció mellett. Ezért az határozatlan ’-’ lesz.
Vagy pl. $Q_{B\,n+1}$ ’7’ rublikája (*Ck=1, $Q_{B\,n}$=1, $Q_{B\,n+1}$=1*) megfelel a Q_{n+1} ’6’ rublikájának (*Ck=1, Q_n=1, Q_{n+1}=1*). Ez alapján T_B ’7’ rubrikája 0 lesz.
Hasonló módon vetjük össze $Q_{A\,n+1}$ V-K táblájáta T tárolóV-K táblájával.



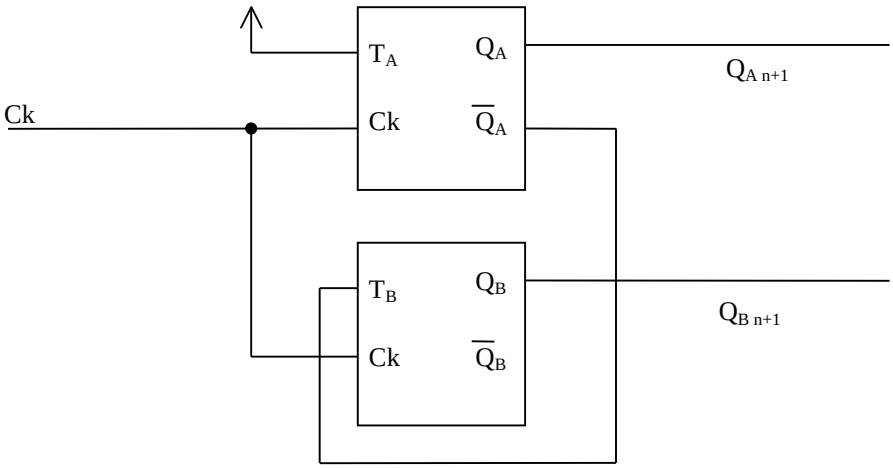




Minimalizálhatunk:



A realizált T tárolós hálózat:



A feladat megoldható táblázatosan is. Először írjuk fel a realizálandó hálózat igazság-tábláját, mellé a T tároló működési tábláját. Egyszerűsítésként megálapítható, hogy mind a megvalósítandó hálózat, mind a T tárolók Ck = 0 alatt nem változtatják meg az állapotukat, ezért a tárolókat vezérlő kombinációs hálózatban nincs értelme Ck-t is figyelembe venni.

A hálózat igazságtáblája úgy kapható meg a legegyszerűbben, ha először lekövetjük az állapotgráfon, hogy melyik állapotból melyik állapotba kell kerülni, majad az így kapott táblázat sorait átrendezzük:

$Q_{B\,n}$	$Q_{A\,n}$
0	0
1	1
1	0
0	1

$Q_{B\,n+1}$	$Q_{A\,n+1}$
1	1
1	0
0	1
0	0

Átrendezés után a működési táblák:

$Q_{B\,n}$	$Q_{A\,n}$
0	0
0	1
1	1
1	0

$Q_{B\,n+1}$	$Q_{A\,n+1}$
1	1
0	0
1	0
0	1

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	1	0
1	0	1

Most emeljük ki $Q_{B\,n}$ és $Q_{B\,n+1}$ oszlopait, másoljuk mellé a T tároló azonos állapotban lévő Q_n és Q_{n+1} oszlopait, valamint az utóbbiakhoz tartozó T bemenet oszlopát. Ennek

segítségével megállapítható a kombinációs hálózat T_B kimenete, a bemeneteit ($Q_{A\,n}$ *és* $Q_{B\,n}$) is felvéve megkapjuk a T_B kimenet igazságtábláját:

$Q_{B\,n}$	$Q_{A\,n}$	$Q_{B\,n+1}$
0	0	1
0	1	0
1	1	1
1	0	0

Q_n	Q_{n+1}	T
0	1	1
0	0	0
1	1	0
1	0	1

T_B	$Q_{B\,n}$	$Q_{A\,n}$
1	0	0
0	0	1
0	1	1
1	1	0

T_B-t a kombinációs hálózatoknál tanult módon minimalizálhatjuk:

$$T_B = \overline{Q_{An}}$$

Ugyanígy járjunk el $Q_{A\,n}$ és $Q_{A\,n+1}$ oszlopaival:

$Q_{B\,n}$	$Q_{A\,n}$	$Q_{A\,n+1}$
0	0	1
0	1	0
1	1	0
1	0	1

Q_n	Q_{n+1}	T
0	1	1
1	0	1
1	0	1
0	1	1

T_A	$Q_{B\,n}$	$Q_{A\,n}$
1	0	0
1	0	1
1	1	1
1	1	0

$$T_A = 1$$