# Digitális technika II

Dr. Göllei Attila, Dr. Holczinger Tibor, Dr. Vörösházi Zsolt



**2014**

A tananyag a TÁMOP-4.1.2.A/1-11/1-2011-0104 "A felsőfokú informatikai oktatás minőségének fejlesztése, modernizációja" c. projekt keretében a Pannon Egyetem és a Szegedi Tudományegyetem együttműködésében készült.


# Előszó

A Pannon Egyetem Műszaki Informatikai Karán tanuló mérnök informatikus alapszakos hallgatók eddig más oktatási intézmények által kiadott jegyzetekből, a kereskedelemben kapható tankönyvekből valamint az Internetről letöltött anyagokból tanulhatták a digitális technikát (BME: [ARATÓP], vagy SZE: [KERESZTP]). Hallgatóink, bár ezekből is tökéletesen elsajátíthatták a tantárgy elméleti részeit, azonban nincs közöttük egyetlen olyan sem, amely a karunkon folyó képzés követelményeihez és tematikájához teljes mértékben igazodna. Jelen jegyzet ezt a hiányt hivatott pótolni.

A jegyzet készítése során fontos célul tűztük ki, hogy a meglévő előadás vázlatokra, fóliákra épülő egységes, jól hasznosítható oktatási segédanyag készüljön a képzésben lévő Digitális Technika I. II., és a Digitális Áramkörök c. tárgyakhoz, amelyet nagy hallgatói létszámmal oktatunk a Pannon Egyetem veszprémi, és nagykanizsai képzési helyein, valamint a Szegedi Tudomány Egyetemen.

A jegyzet mind mérnök informatikus, mind villamosmérnök szakos hallgatók számára korszerű, konkrét, elméleti és gyakorlati ismereteket tartalmaz a digitális áramkörök és rendszerek tervezésének elsajátításához. A témaköröket a nagy terjedelem miatt két jegyzetben publikáljuk: a Digitális technika I. c. segédlet főként a digitális áramkörök bevezetésével, alapfogalmaival és a kombinációs logikai hálózatok szintjével foglalkozik, míg a rá épülő Digitális technika II. c. segédlet a sorrendi logikai hálózatok elméleti és gyakorlati hátterét, valamint azok komplexebb tervezési lépéseit ismerteti.

# Tartalomjegyzék

# Rövidítések

| Kifejezés | Angol | Magyar |
|---|---|---|
| BCD | Binary Coded Decimal | Binárisan Kódolt Decimális számok |
| CAD | Computer-Aided Design | Számítógéppel segített tervezés |
| CG | Carry Propagate | Átvitel terjesztő egység |
| CMOS | Complementer Metal-Oxid Semiconductor | Komplementer-technológiájú fém-oxid félvezetők |
| CP | Carry Generate | Ávitel generáló egység |
| DDL | Diode-Diode Logic | Dióda-dióda logika |
| DMC | Differential Manchester Coding | Differenciális Manchester kódolás |
| DNF | Disjunctive Normal Form | Diszjunktív Normál Alak |
| DTL | Diode-Transistor Logic | Dióda-transzosztor logika |
| ECL | Emitter Coupled Logic | Emitter-csatolású logika |
| EDA | Electronic-design Automation | Elektronikai tervezés-automatizálás |
| FA | Full Adder | Teljes Összeadó |
| FF | Flip-Flop | Élvezérelt tároló |
| FPGA | Field Programmable Gate Arrays | Újraprogramozható Kapuáramkörök |
| GAL | Generic Array Logic | Generikus Tömb Logika |
| HA | Half Adder | Fél Összeadó |
| IC | Integrated Circuit | Integrált Áramkör |
| KH | Combinatorical Network Logic | Kombinációs Hálózat |
| KNF | Conjunctive Normal Form | Konjunktív Normál Alak |
| KV | Karnaugh-Veicht | Karnaugh Veicht diagramm (Karnaugh tábla) |
| LS | Low-power Shottkey | Kis-teljesítményű Shottkey-dióda |
| MC | Manchester Coding | Manchester kódolás |
| NRZ | Non-Return to Zero | Nullára nem-visszatérő (kódolás) |
| NTSH | Non-totally Specified Network | Nem-Teljesen Specifikált Hálózat |
| OC | Open-collector | Nyílt-kollektoros (logikai kapu) |
| PAL | Programmable AND Logic | Programozható ÉS logikai eszköz |
| PCB | Printed Circuit Board | Nyomtatott Áramkör |
| PLA | Programmable Logic Array | Programozható Logikai Tömb |
| PLD | Programmable Logic Devices | Programozható Logikai Eszközök |
| QM | Quine McCluskey | Quine McCluskey (Számjegyes minimalizálás) |
| RCA | Ripple Carry Adder | Átvitelkezelő (több-bites) Összeadó |
| RZ | Return to Zero | Nullára visszatérő (kódolás) |
| SH | Sequential Network Logic | Sorrendi (szekvenciális) Hálózat |

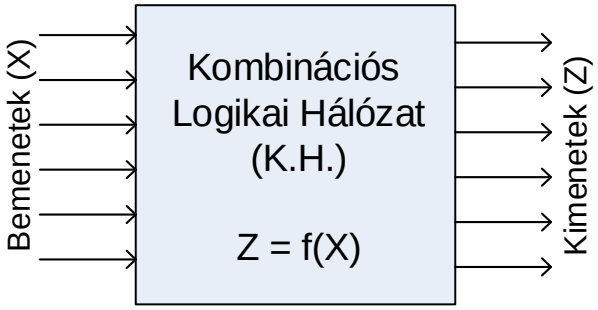
| | | |
|---|---|---|
| TS | Tri-State | Három-állapotú (0, 1, Z) kimenetek |
| TSH | Totally Specified Network | Teljesen Specifikált Hálózat |
| TTL | Transistor-Transistor Logic | Tranzisztor-Tranzisztor Logika |

# 1. Sorrendi hálózatok alapjai

## *Bevezetés*

Ebben a fejezetben a sorrendi hálózatok működésével, felépítésével, fizikailag realizálható modelljeivel (Mealy, Moore modellek) fogunk megismerkedni. A működés leírásának nyomon követésére az állapottáblát, illetve a szemléletesebb, de vele ekvivalens állapot-gráfot fogjuk majd használni. A sorrendi hálózatok működési hátterének ismeretében az egyes alkalmazásokban is fontos „építőköveket", az elemi tárolókat fogunk megvalósítani, illetve más sorrendi hálózatokban felhasználni (pl. szinkron-, és aszinkron hálózatok). A sorrendi hálózatok esetében is – a kombinációs hálózatoknál megismert – működési bizonytalanságok (hazárd) vizsgálata szükséges: egyrészt a késleltetésekből adódó kritikus versenyhelyzet („rendszer hazárd"), másrészt pedig a lényeges hazárd megszüntetése válhat szükségessé.

**Emlékeztető**: (K.H.) Kombinációs logikai hálózatról beszéltünk, ha a mindenkori kimeneti kombinációk (függő változók) értéke csupán a bemeneti kombinációk (független változók) pillanatnyi értékétől függ (tároló „kapacitás", vagy memória nélküli hálózatok voltak). Egyrészt a bemenetek ($X$), kimenetek ($Z$) halmazaival jellemezhetők, másrészt pedig az kimeneti függvény megadásával ($Z = f(X)$).




**1.1 ábra: Kombinációs logikai hálózat szimbóluma**

Hívják még nyílt hatásláncnak is: $f$ ***egyértelmű hozzárendelés (de nem kölcsönösen egyértelmű).***

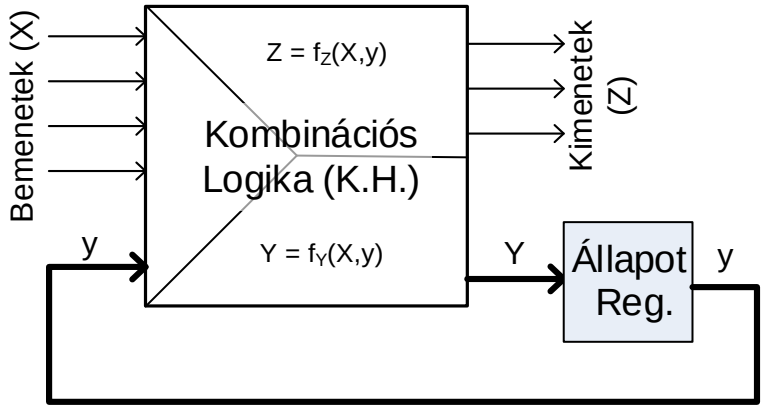
**Def: S.H. – Sorrendi (vagy szekvenciális) hálózatokról** beszélünk**,** ha a mindenkori kimeneti kombinációt, nemcsak a pillanatnyi bemeneti kombinációk, hanem a korábban fennállt bementi kombinációk és azok sorrendje (mint állapot) is befolyásolja. A szekunder/másodlagos kombinációk segítségével az ilyen típusú 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, azaz az állapot (például az Állapot Regiszter tartalma).

Sorrendi hálózatok *fajtái* között két fő típust különböztetünk meg:

1. Szinkron működésű:
    a. Mealy modell
    b. Moore modell
2. Aszinkron működésű:
    - Ütemezetlen (normál) aszinkron működésű
    - Ütemezett aszinkron működésű

A továbbiakban a sorrendi hálózatok helyett praktikus rövidítésként alkalmazzuk az S.H.-t.

Az 1.2-es ábrán egy később tárgyalandó tipikus sorrendi hálózati modell (Mealy modell) blokkszintű felépítése látható. Egyrészt a bemenetek ($X$), kimenetek ($Z$), következő állapotok ($Y$), és az aktuális állapotok ($y$) halmazaival jellemezhetők, másrészt pedig az kimeneti ($f_z(X, y)$), valamint az következő állapot függvény megadásával ($f_Y(X, y)$).




**1.2 ábra: Egy tipikus sorrendi logikai hálózat blokkszintű felépítése**

Hívják még **zárt hatásláncnak** is, a szekunder kombinációk (állapotok), mint visszacsatolás hat a kombinációs hálózat bemenetére.

Előzetesen: a sorrendi hálózatok leképezési szabályai:

Mealy modell $f_Z'(y) \Rightarrow Z$ vagy Moore modell: $f_z(X, y) \Rightarrow Z$

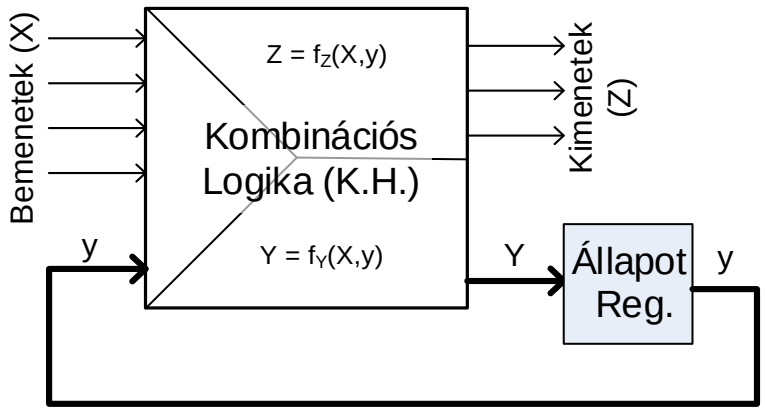
A kimeneti függvény előállítása mindkét modell (Mealy, Moore) esetén azonos: $f_y(X, y) \Rightarrow Y$

## *Sorrendi hálózatok alapmodelljei:*

### Mealy modell

A sorrendi hálózatok egyik alapmodellje. Mivel a hálózatban valós késleltetés van, ezért a kimeneten az eredmény véges időn belül jelenik csak meg! (Visszacsatolás + állapot regiszter elérése = késleltetés). Korábbi értékek (állapotok) visszacsatolódnak a bemenetre: így a kimeneteket nemcsak a bemenetek pillanatnyi, hanem a korábbi állapotai is együtt határozzák meg. Problémák merülhetnek fel az állapotok és bemenetek közötti „*szinkronizáció*" hiánya miatt (változó hosszúságú kimenetet – dekódolás is, amennyiben vezérlő egységek konstruálásában gondolkozunk). Ezért alkalmazzuk legtöbb esetben a második, Moore-féle automata modellt.

- Halmazok:
  - $X$ – a bemenetek,
  - $Z$ – a kimenetek,
  - $Y$ – a következő állapotok halmaza,
  - $y$ – az aktuális állapotok halmaza
- Két leképezési szabály a halmazok között:
  - $f_y(X, y) \rightarrow Y$ : következő állapot függvény (hívják még következő állapot logikának is – „next-state logic"),
  - $f_z(X, y) \rightarrow Z$ : kimeneti függvény.

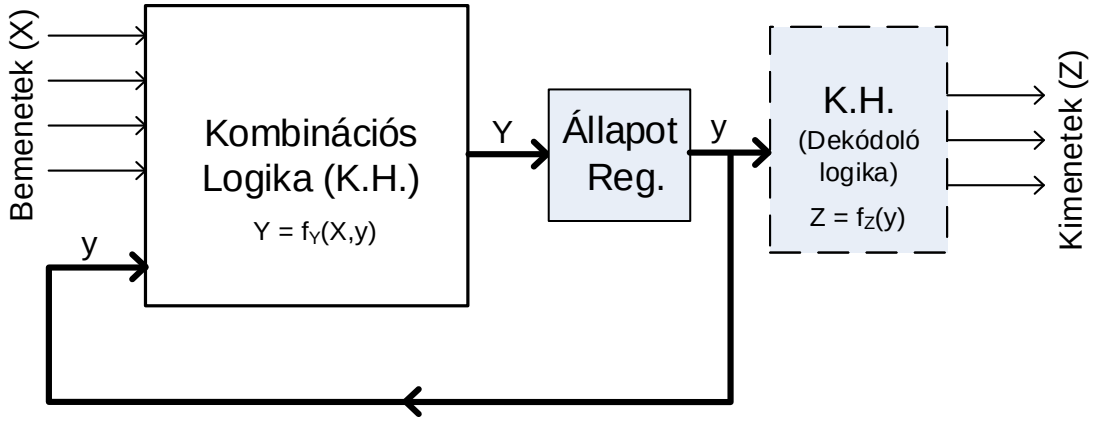



**1.3 ábra: Mealy sorrendi hálózati modell blokkszintű felépítése**

### Moore modell

A sorrendi hálózatok másik alapmodellje. A késleltetési viszonyokról, illetve a hálózat megszólalási idejéről hasonló mondható el, mint a Mealy modell esetében, valamint a korábbi értékek (állapotok) visszahatnak a bemenetre: azonban a kimeneteket mindig az állapotregiszterben tárolt aktuális állapot értékeiből határozzuk meg. Emiatt nem léphet fel „szinkronizációból" adódó probléma az állapotok és bemenetek között (változó hosszúságú kimenetet - dekódolás). Ezért alkalmazzák a legtöbb esetben ezt a második, Moore-féle automata modellt (pl. vezérlő egységek felépítésénél).

- Halmazok:
  - $X$ – a bemenetek,
  - $Z$ – a kimenetek,
  - $Y$ – a következő állapotok halmaza,
  - $y$ – az aktuális állapotok halmaza.
- Két leképezési szabály a halmazok között:
  - $f_y(X, y) \rightarrow Y$ : következő állapot függvény,
  - $f'_Z(y) \rightarrow Z$ : kimeneti függvény (opcionális).




**1.4 ábra: Moore sorrendi hálózati modell blokkszintű felépítése**

Azt is mondhatjuk, hogy a kimenetek közvetlenül csak a pillanatnyi állapottól függenek (bemenettől függetlenek, pontosabban csak közvetett módon függenek).

### Definíciók Aszinkron Sorrendi hálózatok esetén

**Def**: **Nyugalmi állapot** egy adott $X$ bemeneti kombináció mellett csak akkor jöhet létre, ha egy kialakult $Y$ szekunder kombináció a bemenetre $y$-ként visszacsatolódva (azaz $y \leftarrow Y$ visszahatással) az $f_y(X, y)$ leképezés alapján *változatlan* $Y$ kombinációt hoz létre.

**Def**: Amennyiben ilyen feltételek mellett, és változatlan bemeneti kombinációra nem változik az $Y$ ez nyugalmi állapotot (vagy más néven **stabil állapotot** jelent a továbbiakban).

**Def**: Természetesen az $y \neq Y$ szekunder kombináció csak átmenetileg állhat fenn, ezt nevezzük **instabil állapotoknak**). Az instabil állapotok fennállási idejét a visszacsatoló ágak *„jelterjedési"* (propagációs) késleltetése, valamint az $f_y(X, y)$ leképezést megvalósító logikai kombinációs hálózat ún. *„megszólalási" ideje* együttesen határozzák meg. Ha az instabil állapot ideje alatt az $X$ bemeneti kombináció megváltozik, akkor ennek hatására kialakuló $Y$ és $Z$ kombinációk értéke attól fog függeni, hogy az $X$ bemeneti változás pillanatában éppen melyik instabil állapot állt fent.

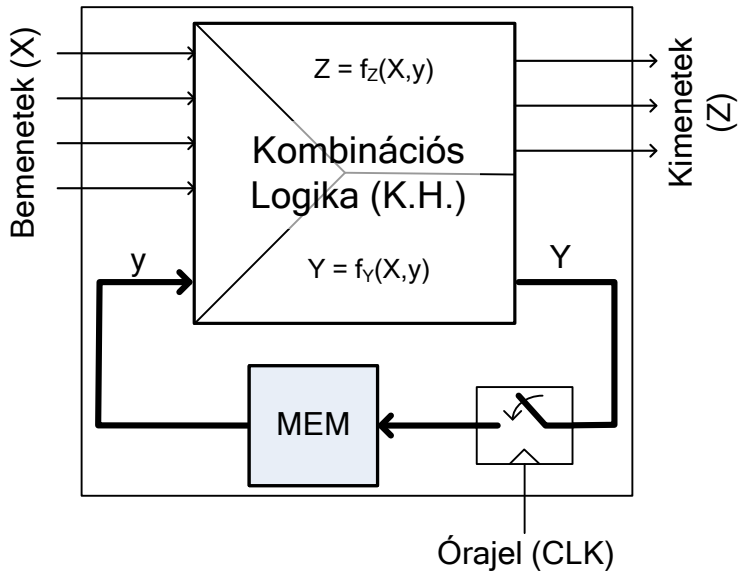
**Def**: Ha nem alakul ki stabil állapot (adott $X$ bemenetre) és az instabil állapotok folyamatosan egymást váltják, akkor **oszcillációról** beszélünk.

**Def**: **Normál aszinkron hálózat**ról beszélünk, ha bármely két *stabil* állapota közötti átmenet során *legfeljebb* egy *instabil* állapot szerepel. Ez kielégíti annak a feltételét, hogy mindig visszatérjünk egy instabil állapotból újabb stabil állapotba.

Aszinkron S.H.-ok minden esetben megvalósíthatók logikai K.H.-ok visszacsatolásának megadásával is, azonban megvalósíthatók szinkron S.H.-ok segítségével (pl. elemi tárolók megadásával – későbbi 5. fejezetben ismertetendő). További részleteket az aszinkron S.H. tervezési lépéseinek vizsgálatakor ismerhetünk meg.

Általánosan elmondható, hogy több szekunder ($Y$) állapot bevezetése szükséges az aszinkron S.H. működésének megadásához (szemben a szinkron S.H.-kal), a stabil, instabil állapotok megkülönböztetésének köszönhetően.

**Def**: Aszinkron hálózatokhoz kapcsolónak olyan megvalósítások, amelyeknél a visszacsatoló szekunder kombinációk változásának visszahatását ütemezett módon befolyásoljuk. Ezeket **ütemezett aszinkron sorrendi hálózatoknak** nevezzük, lásd 1.5 ábra.




**1.5 ábra: Ütemezett aszinkron hálózat blokkszintű felépítése**

Ütemezett aszinkron S.H. Visszacsatoló ágakban ($Y$) periodikusan nyitjuk/zárjuk a kapcsolókat (CLK). M = Memória: $Y$ szekunder állapot tároláshoz ($y = Y \rightarrow \mathrm{MEM}[Y] = y$), azonban a $X/Z$: bemenet/kimenet nem ütemezett! Ebben az esetben hálózat „sebességét" a CLK határozza meg (gyorsabb működésű elemeket lassíthatja is egyben), míg az instabil $Y$ állapotok ezzel a CLK órajellel szinkronban követik egymást.

## Definíciók Szinkron Sorrendi hálózatok esetén:

**Def**: **Állapotok:** Nemcsak a visszacsatoló ágak ($Y$) szekunder változói, hanem az $X$ bemeneti kombinációk is periodikusan, órajel hatására (CLK) érkeznek. A $Z$ kimeneteket is ütemezetten, CLK hatására kapjuk meg hálózatból.

Szinkron esetben *minden* – stabil / instabil – állapotban érkezhet új érvényes bemenet az órajel használata miatt. Ezáltal **nem különböztetjük meg az instabil / stabil állapotokat** szinkron esetben. Emiatt általánosan elmondható, hogy kevesebb szekunder állapot szükséges a működés leírásához (szemben az aszinkron esettel). Ennek következtében a szinkron hálózat tervezése is általában egyszerűbb. Azonban, mivel az órajel határozza meg a hálózat sebességét, általánosan elmondható, hogy lassabbak mind az ütemezett aszinkron hálózatok.

**Def: Szinkron működés „szinkronizációs feltételei":**

- Az $Y(y)$ visszacsatoló ágak órajel-vezérléssel (CLK) periodikusan megszakíthatóak legyenek,
- A visszacsatoló ágak tartalmazzanak (MEM) definiált működésű tároló-elemeket,
- Biztosítani kell, hogy az $X$ bemeneti változások a CLK órajellel szinkronban történjenek,
- Definiálni kell a $Z$ kimeneti kombinációk értelmezési időtartományát, valamint a $Z$ kimeneti változások CLK órajellel szinkronban történjenek.

**Következmény** aszinkron hálózatokhoz: mivel a tervezésük során *nem kell* szinkronizációs feltételeket biztosítani, megépíthetők tisztán visszacsatolt K.H.-ok is.

Szinkron működés szemléltetése idő-diagrammon (Megjegyzés: a mai EDA – Elektronikai Tervezés Automatizálása során ezeket az idő-diagrammokat a logikai hálózat működésének vizsgálatakor, a bemeneti gerjesztésekre adott válaszok alapján automatikusan generálják – működés viselkedési szimulációja PC-n történhet).

a.) Szinkron S.H. – MEALY modell
b.) Szinkron S.H. – MOORE modell

### a.) Idődiagram szinkron Mealy modell esetén:

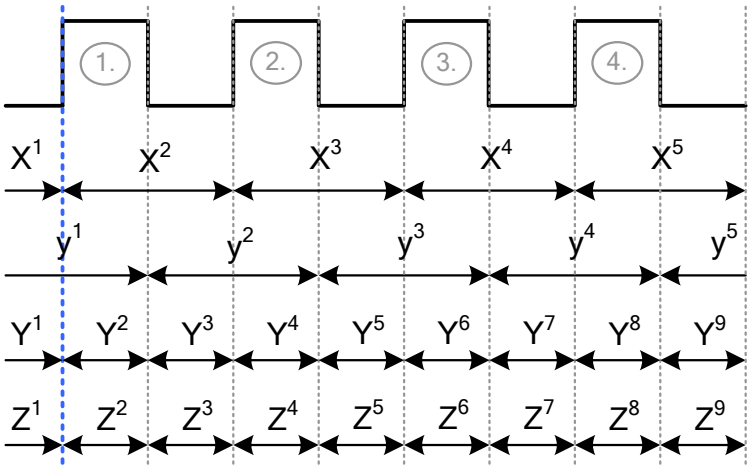
Tekintsük a szinkron sorrendi hálózat Mealy modelljét, amelynél a következő függvények ismertek:

- Következő állapot függvény: $f_Y(X, y) \rightarrow Y$
- Kimeneti függvény: $f_z(X, y) \rightarrow Z$

Azt feltételezzük, hogy az $X$ bemeneti kombinációk az órajel felfutó éleire ( ⌐↑ ), míg az $y$ szekunder állapot (aktuális) kombinációk az órajel lefutó éleire ( ↓¬ ) változnak meg.

Kezdeti feltételként a következő adott: $f_Y(X^1, y^1) \rightarrow Y^1$ , valamint $f_z(X^1, y^1) \rightarrow Z^1$. Felső index a különböző kombinációkat jelöli megkülönböztetésként.

Ekkor a következő idődiagramot lehet felrajzolni**:**




**1.6 ábra: Idődiagram szinkron Mealy-modell esetén**

A fenti ábrán jól látható, hogy a $Z^i$ kimeneti kombinációk, valamint $Y^i$ következő állapot értékek direkt módon mind az $X^i$ bemeneti kombinációk, mind pedig a visszahatott $y^i$ kombinációk értékétől is függenek (változnak).

### b.) Idődiagram szinkron Moore modell esetén:

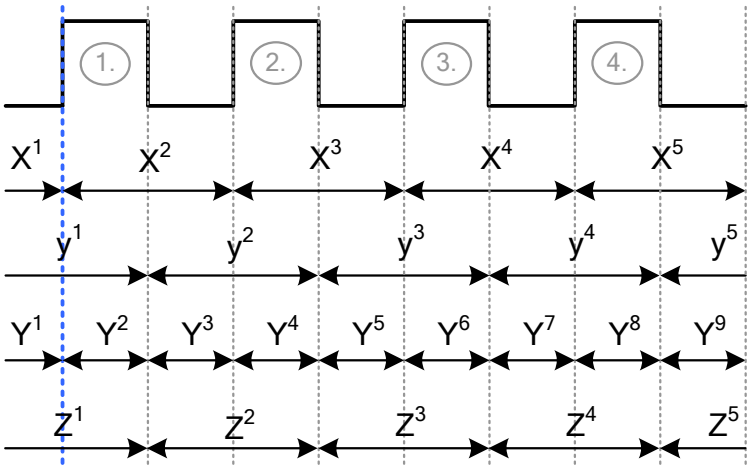
Tekintsük a szinkron sorrendi hálózat Moore-modelljét, amelynél a következő függvények ismertek:

- Következő állapot függvény: $f_Y(X, y) \rightarrow Y$
- Kimeneti függvény: $f_z(y) \rightarrow Z$ (ez a lényeges különbség a Mealy modellhez képest!)

Azt feltételezzük, hogy az $X$ bemeneti kombinációk az órajel felfutó éleire ( ⌐↑ ), míg az $y$ szekunder állapot (aktuális) kombinációk az órajel lefutó éleire ( ↓¬ ) változnak meg.

Kezdeti feltételként a következő adott: $f_Y(X^1, y^1) \rightarrow Y^1$, valamint $f_z(X^1, y^1) \rightarrow Z^1$. Felső index a különböző kombinációkat jelöli megkülönböztetésként.

Ekkor a következő idődiagramot lehet felrajzolni:




**1.7 ábra: Idődiagram szinkron Moore-modell esetén**

A fenti ábrán jól látható, hogy a $Z^i$ kimeneti kombinációk direkt módon mindig csak az $y^i$ aktuális állapotok értékétől függenek (természetesen az $X^i$ bemenetektől indirekt módon ugyan, de függenek), mialatt az $Y^i$ következő állapot értékek mind az $X^i$, mind pedig a visszahatott $y^i$ kombinációk változásaitól egyszerre függenek.

# 2. Sorrendi hálózatok működésének leírása

Az egyértelmű működés leírásához az szükséges, hogy pontosan ismerjük az $f_Z(.)$ és $f_Y(.)$ leképezéseket.

**Def: Állapottábla**: egy olyan Karnaugh táblázathoz hasonlítható felépítésű táblázat, amelynek mindenegyes cellájába (rovatába) azokat az $Y$-következő állapot-, illetve $Z$ kimeneti kombinációkat kell beírni, amelyeket a $f_Z$, $f_Y$ leképezések az adott cellához rendelt $X$ – bemeneti kombinációk, és $y$ aktuális állapot kombinációk fennállása esetén hoznak létre.

## *2.1 Példa: ÁLLAPOT TÁBLA bemutatása aszinkron működést feltételezve*

Legyenek adottak az alábbi paraméterek:

- $X$: bemeneti kombinációk száma 4
- $y/Y$: szekunder változók (állapot) kombinációk száma 4
- $Z$: kimeneti kombinációk lehetséges száma 4

A bemeneti kombinációk kért kiértékelésének (igények) sorrendje legyen a következő:

$$\boldsymbol{X^1 \rightarrow X^3 \rightarrow X^2 \rightarrow X^1 \rightarrow X^3 \rightarrow X^4 \rightarrow X^3}$$

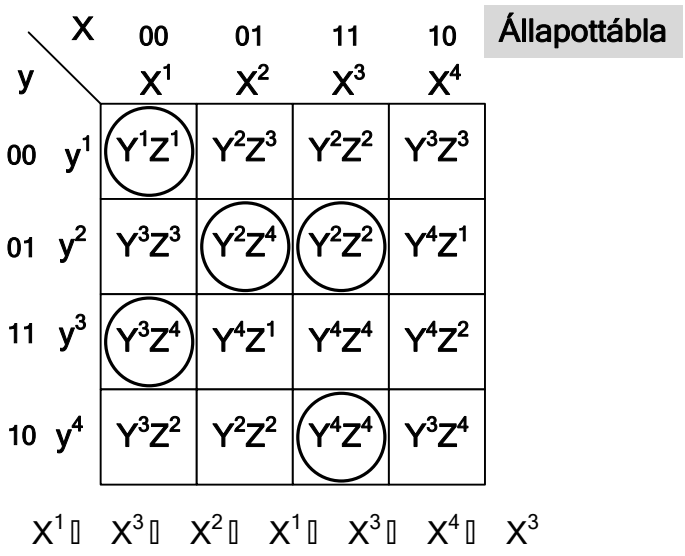
Az állapottábla a fenti paraméterek figyelembe vételével legyen adott a következő formában:

Állapottábla

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

*Megjegyzés*: Tömör jelölésként, a felső indexekben az eltérő $X$, illetve $y$ kombinációkat jelölik (a megfelelő bináris számok helyett).

Első lépésként jelöljük be a stabil állapotokat (kör szimbólum): stabil állapot azokban a cellákban van, amelyekben az $Y^i = y^i$ indexek azonosak (változatlan $X$ mellett tehát azonos állapotkombináció → azonos állapotértéket kapunk az $y \leftarrow Y$ visszahatás során).



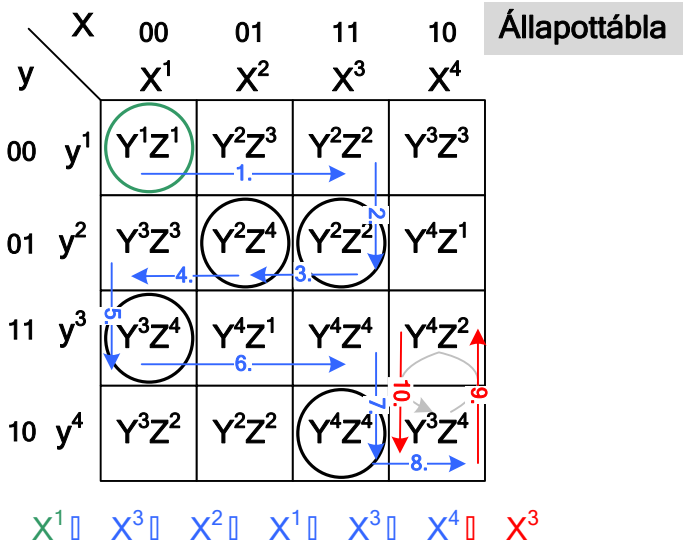

Táblázat alján az $X$ bemeneti kombinációk igényelt kiértékelések sorrendje is fel lett tüntetve.

*Aszinkron működés* során a következő szabályoknak kell érvényesülni:

a.) Jelöljünk ki egy stabil *kiindulási állapotot* (zöld kör, mivel a kezdés sohasem történhet instabil állapotból!):
Tegyük fel, hogy $X = X^1$, $y = y^1$, $Y = Y^1$, $Z = Z^1$.

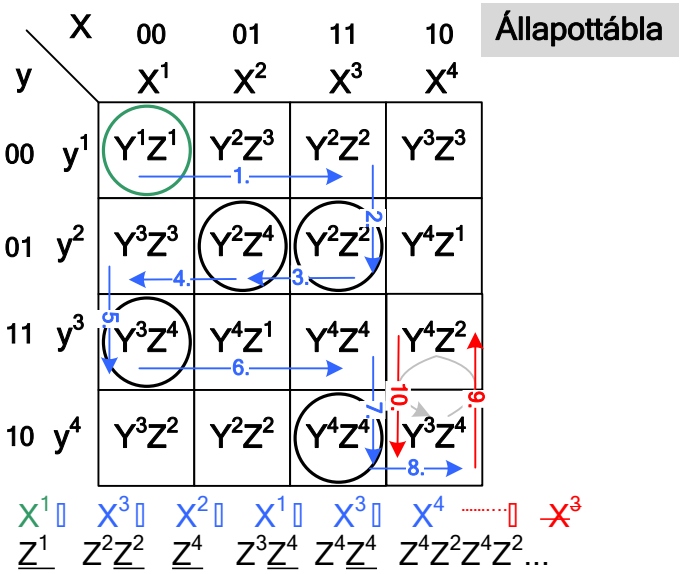
Azaz $f_y(X^1, y^1) \Rightarrow Y^1$, illetve $f_z(X^1, y^1) \Rightarrow Z^1$.

b.) Csak akkor léphetünk vízszintes irányban $X$-el, azaz csak akkor elégíthetünk ki egy új $X$ bemeneti kombinációs igényt, amikor a *stabil* állapotban vagyunk. Egyébként *instabil* állapotban függőlegesen – y irányában – léphetünk csak az instabil állapotban.
A fenti szabályoknak megfelelően az állapottáblán a következő aszinkron működést tudjuk berajzolni, a bemeneti kombinációs igényeket figyelembe véve:




*Következmény*: a fenti ábrán jól látható, hogy az utolsó előtti $X$ bemeneti kombináció igényének érkezéséig ugyan normál aszinkron működésű a hálózatról beszélhetnénk, azonban az utolsó $X^3$ kérést már nem tudjuk kielégíteni, tehát a hálózat **oszcillál** (9-es, és 10-es lépések követik egymást folytonosan, amelyből már nem kerülünk át egy újabb stabil állapotba). Végül nem kerülünk újból a $X^3$-as stabil állapotba.

A működés során leolvashatók a mindenkori stabil/instabil állapotok mellett kapott $Z$ kimeneti kombinációk is, azonban a stabil állapotok mellett előállt kimeneteket tekintjük elfogadhatónak.




A fenti ábrán a stabil y állapot mellett leolvasott érvényes kimeneteket aláhúzással jelöltük (melyek száma lehet 1, vagy több, attól függően, hogy stabil állapotban voltunk, vagy instabil állapotokon lépkedve olvastuk le a mindenkori kimeneti értéket).

**Def: Aszinkron hálózat oszcillációjának szükséges és elégséges feltételei**.

- Abban az esetben NEM léphet fel oszcilláció *(szükséges feltétel),* ha az állapottábla minden oszlopában legalább egy stabil állapot van!
- Akkor NEM léphet fel oszcilláció, tehát *az elégséges feltételhez* annak kell teljesülnie, hogy a hálózat minden egyes $X$ bemeneti kombinációra – vagyis minden oszlopban – eljusson legalább egy stabil állapotba.

*Következmény*: az előző *2.1 Példa* alapján az $X^4$ jelű oszlopban stabil állapotot már az állapottábla felírásakor, kezdetben sem tudtunk kijelölni. Ebből gondolhatunk arra, hogy oszcilláló viselkedésű aszinkron hálózatot fogunk kapni, mivel nem tudunk minden lehetséges $X^i$ bemeneti kérést kielégíteni, és az oszcillációból nem léphetünk ki újabb stabil állapotba.

## *2.2 Példa: ÁLLAPOT TÁBLA bemutatása szinkron működést feltételezve*

Legyenek adottak az alábbi paraméterek:

- $X$: bemeneti kombinációk száma 4
- $y/Y$: szekunder változók (állapot) kombinációk száma 4
- $Z$: kimeneti kombinációk lehetséges száma 4

A bemeneti kombinációk kért kiértékelésének (igények) sorrendje legyen a következő:

$$\boldsymbol{X^1 \rightarrow X^3 \rightarrow X^2 \rightarrow X^1 \rightarrow X^3 \rightarrow X^4 \rightarrow X^3}$$

Az állapottábla a fenti paraméterek figyelembe vételével legyen adott a következő formában:

Állapottábla

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

$X^1$ $X^3$ $X^2$ $X^1$ $X^3$ $X^4$ $X^3$

*Megjegyzés*: Tömör jelölésként, a felső indexekben az eltérő $X$, illetve $y$ kombinációkat jelölik (a megfelelő bináris számok helyett).

Szinkron esetben – definíció szerint – nem különböztetünk meg stabil/instabil állapotokat. Ez azt is jelenti, hogy kiindulási állapot a vizsgálat során bármelyik állapotban lehet.

Táblázat alján az $X$ bemeneti kombinációk igényelt kiértékelések sorrendje is fel lett tüntetve. Itt kell megjegyezni, hogy a későbbiek során a bemeneteknél a nem szomszédos átmeneteket, pl. 00 → 11, vagy 01 → 10 *nem engedjük meg*, a funkcionális hazárdok kiküszöbölése miatt, azonban most az egyszerűség végett ezt a megkötést nem vettük figyelembe.

*Szinkron működés* során a következő szabályoknak kell érvényesülni:

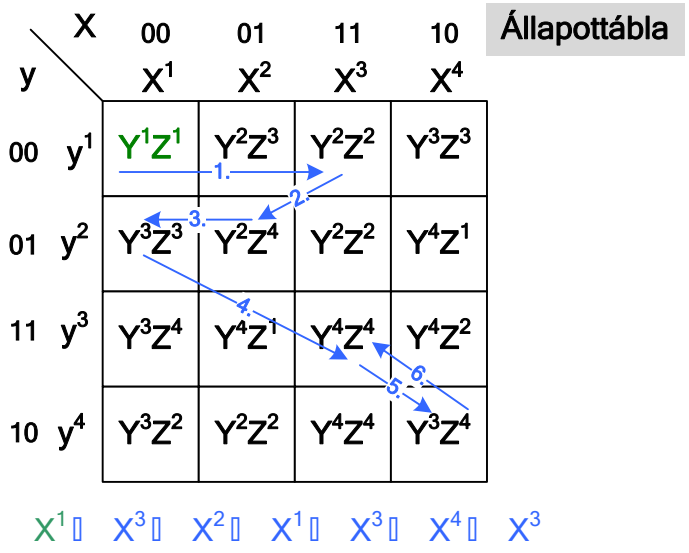
a.) Jelöljünk ki egy *kiindulási állapotot*. Bárhol lehet, nincs rá megkötés!
Tegyük fel, hogy $X = X^1$, $y = y^1$, $Y = Y^1$, $Z = Z^1$.

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

- **Moore modell esetén** $f_z(y^1) \Rightarrow Z^1$, **vagy**
- **Mealy modell esetén**: $f_Z(X^1, y^1) \Rightarrow Z^1$

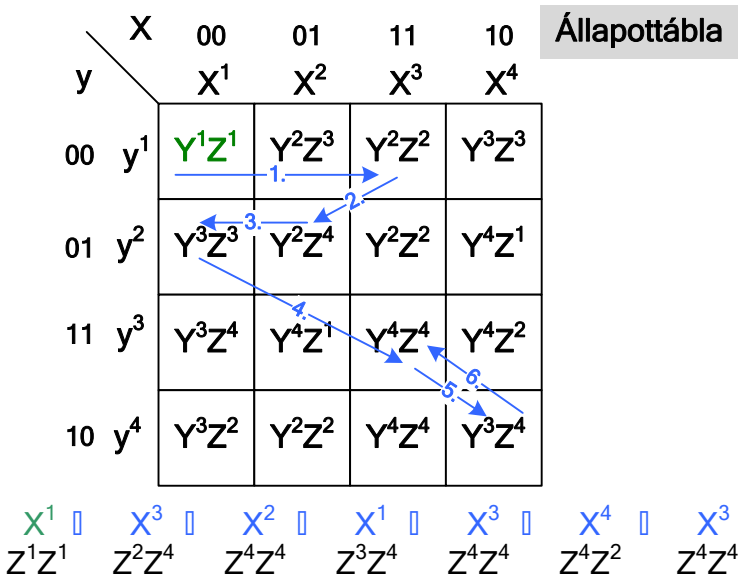
b.) A visszacsatoló ágban minden egyes CLK órajelre! az $Y^1$ válik $y^1$–vé ($y^1 \leftarrow Y^1$ visszahatása során).

c.) Majd $Y^1$ <u>visszahatása után</u> továbbra is az 1. sorban maradunk ($y^1 \leftarrow Y^1$ miatt!), de a 2. oszlopba jutunk, amelyet az $X^2$ bemeneti kombináció jelöl ki
A fenti szabályoknak megfelelően az állapottáblán a következő aszinkron működést tudjuk berajzolni, a bemeneti kombinációs igényeket figyelembe véve:




*Következmény*: mivel szinkron a hálózat az összes bemeneti kombináción végig tudunk lépkedni anélkül, hogy oszcillálna a hálózat.

A működés során leolvashatók a mindenkori állapotok mellett kapott $Z$ kimeneti kombinációk is. Ez látható az állapottábla alatt. *Szinkron* esetben azonban minden $X$ bemeneti kombinációhoz pontosan két $Z$ kimeneti kombináció tartozik, (az *egyik az állapot-visszahatás előtti; a másik pedig visszahatás utáni, de még az aktuális $X$ kombináció mellett leolvasott érték*), amelyek közül a megfelelőt a kimeneti szinkronizációval (CLK működés) választjuk ki.




**Def: Szinkron működés az állapottábla alapján általánosan megfogalmazható**:

Egy adott rovatból/cellából kiindulva mindig abba a rovatba/cellába jutunk, amelynek $i$-dik sorát a kiindulási rovatban szereplő $Y^i$ kombináció, $j$-dik oszlopát pedig az új $X^j$ bemeneti kombináció jelöli ki!

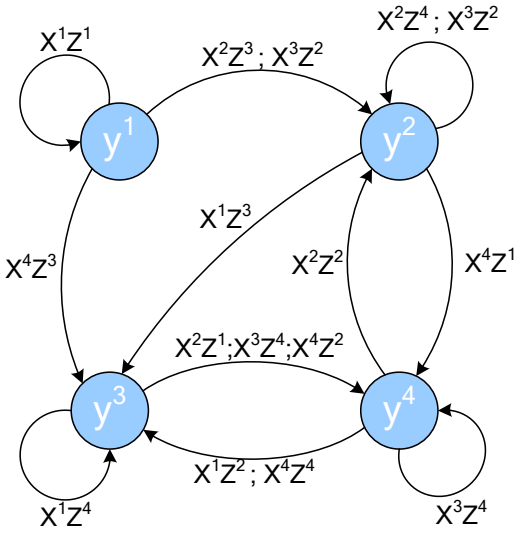
A fenti példákon is jól látható, hogy az *aszinkron*, illetve *szinkron* működés igen eltérő lehet azonos állapottábla és bemeneti kombináció sorozat megadása esetén is!

Azonban az $X^4 \rightarrow X^3$ fennállása nem okoz oszcillációt szinkron esetben.

**Def: Állapottábla megadása „don't care" esetben**

Állapottáblában lehetnek olyan cellák is, amelyekben nem szerepel konkrétan előírt $Y^iZ^j$ kombináció: ezeket don't care ($X$: közömbös) állapotoknak és kimeneteknek tekintjük, valamint hasonlóan járunk el, mint a korábban tanult kombinációs hálózatok esetében.
Ekkor azonban intuitív módon, a lehetőleg optimálisan kellene a kimeneteket és az állapotokat (a felhasznált elemi tárolótól függően, lásd *későbbi 3. fejezet*) megválasztani '0'/'1'-nek.
***Optimális megoldás***: azt jelenti, hogy mind a kimenetek, mind az állapotok értékét úgy választjuk meg, hogy a lehető legkevesebb, legnagyobb, és egyben hazárdmentes lefedést (tömbösítést) biztosítsuk!
Ezeket *Nem Teljesen Specifikált Állapottábláknak (NTSÁ)* nevezzük.
**Def: Állapotgráf:** az állapottábla egy ekvivalens grafikus ábrázolási módja, a S.H. működésének leírására szolgál. A hálózat állapotaira jellemző *y* kombinációknak a *gráf állapotait* (kör), míg a $X$ bemeneti kombinációk hatására lejátszódó állapotváltozásokat a *gráf állapotátmeneteivel*, vagy *önvisszacsatolásokkal* (hurokkal) feleltetjük meg.

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



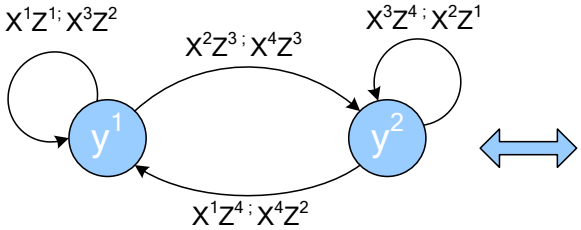

**Aszinkron módú működés feltételezése Állapotgráf vizsgálata alapján.**

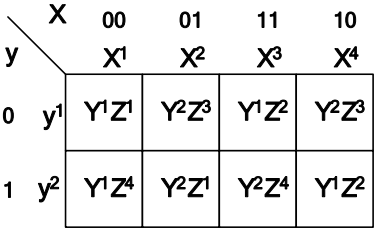
- Csak azok az állapotgráfok értelmezhetők *aszinkron működés* szerint, amelyekben minden egyes állapot átmenetre (átvezető nyílra) felcímkézett $X^i$ bemeneti kombináció végső soron egy olyan csomóponthoz (állapothoz) vezet, amelyhez tartozó önvisszacsatolás (hurok) címkéjén ugyanaz az $X^i$ bemeneti kombináció szerepel! Ilyen esetben biztosan nincs oszcilláció a sorrendi hálózatban, és ekkor aszinkron módon is megvalósítható.
- *Szinkron esetre* nincs ilyen megkötés, tetszőleges gráfokra értelmezhető!

## További példák

### 2.3 Példa: Állapottábla felírása állapotgráf segítségével (2 állapotra)

Adott a következő állapotgráf. Rajzoljuk fel a vele ekvivalens állapottáblát, és jelöljük be a szükséges paramétereket.



| y \ X | 00 $X^1$ | 01 $X^2$ | 11 $X^3$ | 10 $X^4$ |
|---|---|---|---|---|
| 0 $y^1$ | $Y^1Z^1$ | $Y^2Z^3$ | $Y^1Z^2$ | $Y^2Z^3$ |
| 1 $y^2$ | $Y^1Z^4$ | $Y^2Z^1$ | $Y^2Z^4$ | $Y^1Z^2$ |

Megoldás: A fenti ábrán látható, hogy az állapotgráf alapján felírt állapottábla nem tartalmaz don't care értékeket, minden bemeneti kombinációhoz direkt módon meg van adva a következő állapot-,

illetve kimeneti kombináció. Emiatt Teljesen Specifikált Állapottábláról, illetve Állapotgráfról beszélhetünk (TSÁ).
Fontos megjegyezni, hogy az állapottábla $y^1$, illetve $y^2$ állapotainak '0', illetve '1' értéken történő rögzítése a fenti példában ugyan tetszőlegesen történt, azonban a sorrendi hálózatok tervezésekor a pontos rögzítésükről gondoskodni kell, amelynek a megadását a későbbi *8. és 9. fejezetekben* szereplő állapotkódolás során részletesen ismertetünk.
Amennyiben a fenti állapottábla *aszinkron* viselkedését vizsgáljuk, az látható, hogy az $X^4$ bemeneti kombináció fennállásakor (utolsó oszlopban) nem tudunk kijelölni stabil állapotot (a korábbi definíciók szerint).

## *2.4 Példa: Sorrendi hálózat vizsgálata állapottábla alapján*

| y | X: 00 ($X^1$) | 01 ($X^2$) | 11 ($X^3$) | 10 ($X^4$) |
|---|---|---|---|---|
| 00 a | a0 | b1 | b0 | b1 |
| 01 b | d0 | b0 | d0 | d0 |
| 11 c | a1 | d1 | c1 | c1 |
| 10 d | d1 | b1 | c1 | d1 |

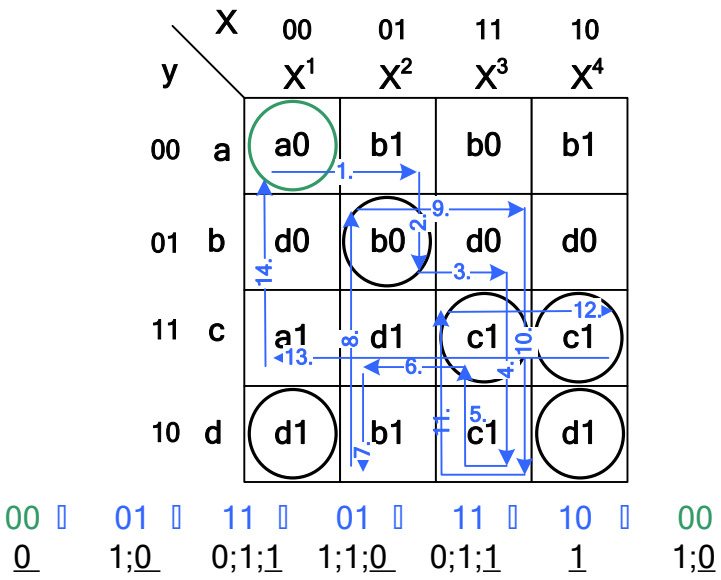
Kérdések:

a.) Milyen szinkron modell szerint működik?
b.) Vizsgálja meg, hogy aszinkron működést feltételezve mi fog történni (normál aszinkron viselkedés vs. oszcilláció)?
c.) A vizsgált lehetséges (szinkron, és aszinkron) működések mellett adja meg a kimeneti jelsorozatot, ha a hálózat kezdetben az 'a' állapotban van, és a bemenetekre az alábbi jelsorozat érkezik:

$X^1$, $X^2$: 00 (kezdetben), majd pedig rendre 01 → 11 → 01 → 11 → 10 → 00 bemeneti kombinációk érkeznek.

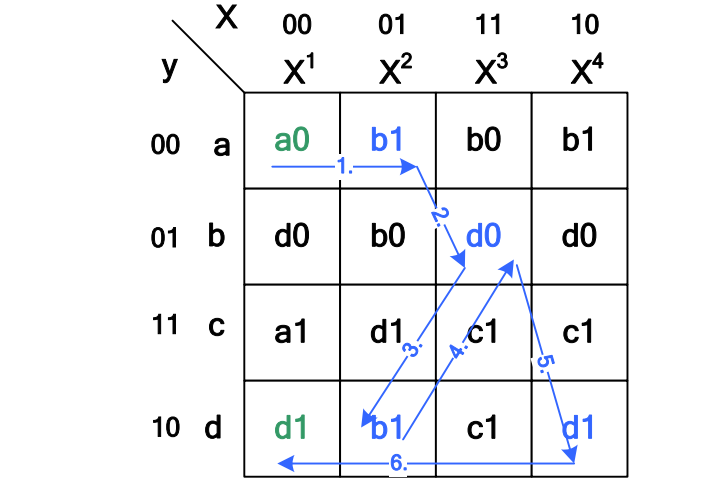
**Megoldás:**

a.) Mivel soronként nem azonosak a kimeneti értékek, ezért szinkron esetben csak Mealy modellről beszélhetünk.
b.) *Aszinkron* működést feltételezve nem lesz oszcilláló a hálózat, mivel minden egyes oszlopába legalább egy helyen tudunk kijelölni stabil állapotot ($y \leftarrow Y$ visszahatás).
c.) A vizsgált bemeneti jelsorozatra *aszinkron* esetben a következőt kapjuk:
Egyrészt ugyan nem oszcilláló a hálózat, azonban normál aszinkron működésű sem, mivel pl. a 2. Stabil állapot után a 3.-4. lépés instabil.
Továbbá a kimeneti értékeknél aláhúzással jelöltük a stabil állapot mellett kapott kimeneti értéket.
Másrészt a bemeneti kombinációk nem szomszédos változásait nem engedélyeztük: eleve így lett megadva a kiértékelés sorrendje. Ennek következtében a sorrendi hálózatban funkcionális hazárd nem alakulhat ki, statikus, illetve dinamikus hazárdot pedig a kombinációs hálózatoknál tanult módon kell megvizsgálni (statikus ← dinamikus).




| 00 | 01 | 11 | 01 | 11 | 10 | 00 |
|---|---|---|---|---|---|---|
| 0 | 1;0 | 0;1;1 | 1;1;0 | 0;1;1 | 1 | 1;0 |

d.) A vizsgált bemeneti jelsorozatra *szinkron* esetben pedig a következőt kapjuk:




| 00 | 01 | 11 | 01 | 11 | 10 | 00 |
|---|---|---|---|---|---|---|
| 0;0 | 1;0 | 0;1 | 1;0 | 0;1 | 1;0 | 1;1 |

A szinkron működés során is leolvashatók a mindenkori állapotok mellett kapott kimeneti kombinációk is. Ez látható az állapottábla alatt. Szinkron esetben azonban minden $X$ bemeneti kombinációhoz pontosan két $Z$ kimeneti kombináció tartozik, (az *egyik az állapot-visszahatás előtti; a másik pedig visszahatás utáni, de még az aktuális X kombináció mellett leolvasott érték*), amelyek közül a megfelelőt a kimeneti szinkronizációval (CLK működés) választjuk ki.

## *2.5 Példa: Sorrendi hálózat vizsgálata állapottábla alapján*

| y \ X | 00 $X^1$ | 01 $X^2$ | 11 $X^3$ | 10 $X^4$ |
|---|---|---|---|---|
| 00 a | a0 | b0 | b0 | b0 |
| 01 b | d0 | b0 | d0 | d0 |
| 11 c | a1 | d1 | c1 | c1 |
| 10 d | d1 | b1 | c1 | d1 |

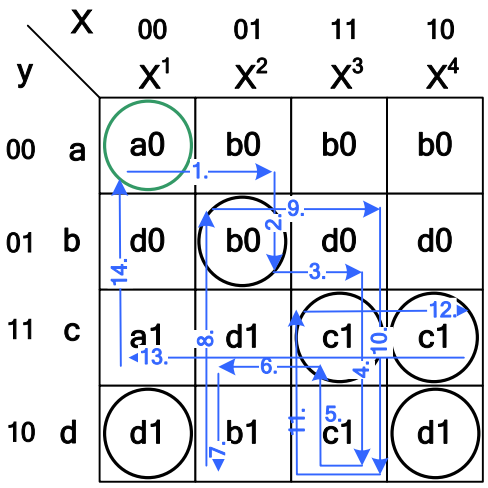
Kérdések:

a.) Milyen szinkron modell szerint működik?
b.) Vizsgálja meg, hogy aszinkron működést feltételezve mi fog történni (normál aszinkron viselkedés vs. oszcilláció)?
c.) A vizsgált lehetséges (szinkron, és aszinkron) működések mellett adja meg a kimeneti jelsorozatot, ha a hálózat kezdetben az 'a' állapotban van, és a bemenetekre az alábbi jelsorozat érkezik:

$X^1$, $X^2$: 00 (kezdetben), majd pedig rendre 01 → 11 → 01 → 11 → 10 → 00 bemeneti kombinációk érkeznek.

**Megoldás:**

a.) Mivel soronként azonosak a kimeneti értékek, ezért szinkron esetben Moore modellről beszélhetünk ($Z = f(X)$).
b.) *Aszinkron* működést feltételezve nem lesz oszcilláló a hálózat, mivel minden egyes oszlopába legalább egy helyen tudunk kijelölni stabil állapotot ($y \leftarrow Y$ visszahatás).
c.) A vizsgált bemeneti jelsorozatra *aszinkron* esetben a következőt kapjuk:
Egyrészt ugyan nem oszcilláló a hálózat, azonban normál aszinkron működésű sem, mivel pl. a 2. Stabil állapot után a 3.-4. lépés instabil.
Továbbá a kimeneti értékeknél aláhúzással jelöltük a stabil állapot mellett kapott kimeneti értéket.
Másrészt a bemeneti kombinációk nem szomszédos változásait nem engedélyeztük: eleve így lett megadva a kiértékelés sorrendje. Ennek következtében a sorrendi hálózatban funkcionális hazárd nem alakulhat ki, statikus, illetve dinamikus hazárdot pedig a kombinációs hálózatoknál tanult módon kell megvizsgálni (statikus ← dinamikus).

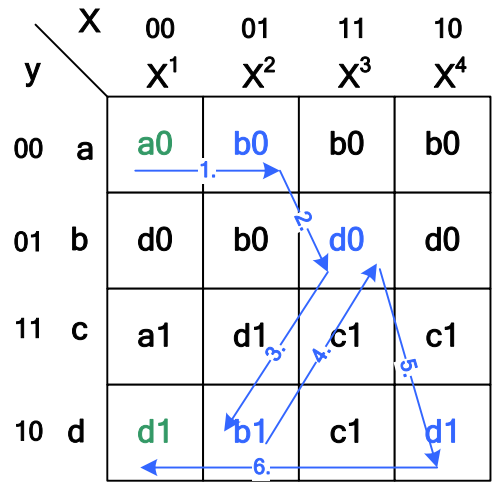



00 ▯ 01 ▯ 11 ▯ 01 ▯ 11 ▯ 10 ▯ 00

0 0;0 0;1;1 1;1;0 0;1;1 1 1;0

ö

d.) A vizsgált bemeneti jelsorozatra *szinkron* esetben pedig a következőt kapjuk:



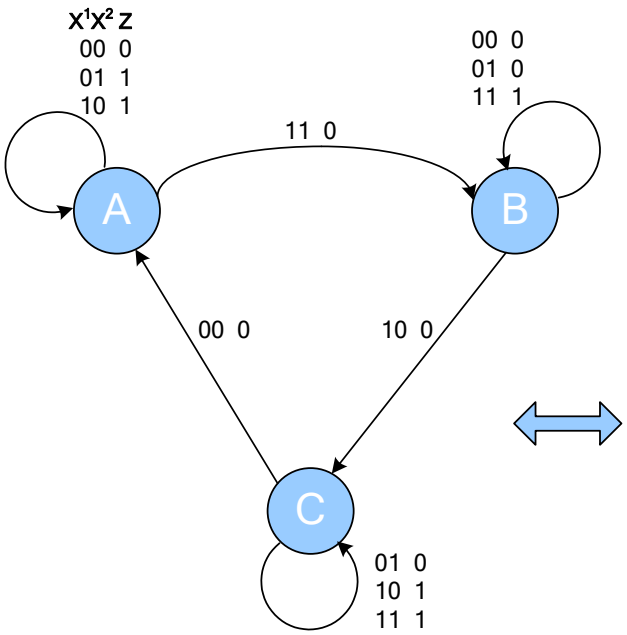

00 ▯ 01 ▯ 11 ▯ 01 ▯ 11 ▯ 10 ▯ 00

0;0 0;0 0;1 1;0 0;1 1;1 1;1

A szinkron működés során is leolvashatók a mindenkori állapotok mellett kapott kimeneti kombinációk is. Ez látható az állapottábla alatt. Szinkron esetben azonban minden $X$ bemeneti kombinációhoz pontosan *két* $Z$ kimeneti kombináció tartozik, (**az *első paraméter az állapot-visszahatás előtti; a második paraméter pedig visszahatás utáni,*** *de még az aktuális $X$ kombináció mellett leolvasott érték*), amelyek közül a megfelelőt a kimeneti szinkronizációval (CLK működés) választjuk ki.

## *2.6 Példa: Állapottábla felírása speciális, don't care állapotot tartalmazó állapotgráf alapján (3 állapotra)*

Adott a következő állapotgráf. Rajzoljuk fel a vele ekvivalens állapottáblát, és jelöljük be a szükséges paramétereket.




| y | X | 00 $X^1$ | 01 $X^2$ | 11 $X^3$ | 10 $X^4$ |
|---|---|---|---|---|---|
| 00 | A | A0 | A1 | B0 | A1 |
| 01 | B | B0 | B0 | B1 | C0 |
| 11 | C | A0 | C0 | C1 | C1 |
| 10 | D | – | – | – | – |

**Megoldás**: Mivel az állapotgráfon 3 állapot (A, B, és C) volt felrajzolva, ezért a negyedik, D állapotot az állapottáblában *don't care*-ként kell rögzítenünk. Az állapotok száma n=3, azonban ezt legalább 2-biten tudjuk csak megfelelően ábrázolni, a következő összefüggés alapján:

$$\lceil \log_2(n) \rceil = \lceil \log_2(3) \rceil = 2bit$$

Ezáltal egy Nem Teljesen Specifikált Állapotgráfról, illetve Állapottábláról beszélhetünk (NTSÁ). A D állapot, don't care-ként való rögzítése sokfajta intuitív tervezési lehetőséget biztosít az optimalizálás során.

Fontos megjegyezni, hogy a fenti állapottábla A, B, C, illetve D állapotainak rendre '00', '01', '11' illetve '10' értékeken történő rögzítése a fenti példában ugyan tetszőleges volt, azonban a későbbi bemutatásra kerülő sorrendi hálózatok tervezésekor a pontos rögzítésükről gondoskodnunk kell, amelynek a megadását a *8. és 9. fejezetekben* szereplő Állapotkódolás során részletesen ismertetünk. A helytelen állapotkódolás további problémákat (hazárdjelenségeket) is okozhat az aszinkron sorrendi hálózatok esetén.

# 3. Flip-flopok, mint a sorrendi hálózatok alapelemei

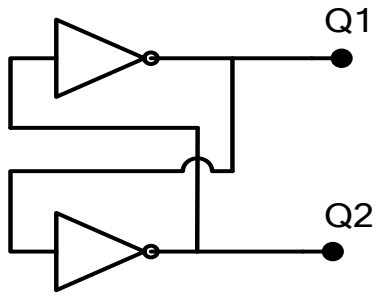
A sorrendi hálózatok megvalósítása során a már megismert logikai kapukon kívül, olyan építőelemekre is szükség lehet, amelyek a hálózat belső állapotának információit tárolni képesek. Kivételek ez alól az aszinkron sorrendi hálózatok, melyek megvalósíthatók csupán a kombinációs hálózatok és hálózati komponensek visszacsatolásával. Ütemezett és szinkron sorrendi hálózatok esetén szükséges az egyes lépések közötti statikus időszakban a hálózat belső állapotát tárolni.

Ezek a tároló elemek egy bit információ tárolására alkalmasak, és a realizációhoz annyi tároló elemre van szükség, ahány független állapota van az adott hálózatnak.

Ezeket a tárolókat **flop-flop-**oknak nevezzük, és a következőkben ismertetjük a leggyakrabban használt tároló elemek típusait, felépítésüket, működésüket.

## *Alapáramkörök*

Az egy bit információ tárolására használható tárolóknak két stabil állapottal kell rendelkezniük. Gyakran hívják őket bistabil billenő áramköröknek is. A stabil állapot tárolásához itt is visszacsatolt áramköröket alkalmaznak, mint annak egyik legegyszerűbb megoldását a következő ábrán láthatjuk.




**3.1. ábra. Bistabil áramkör két inverter felhasználásával.**

A két inverter visszacsatolásával kapott áramkör alkalmas az állapot tárolására, van azonban két alapvető hibája. Az egyik, hogy a tápfeszültség bekapcsolásakor véletlenszerűen, – valójában a két inverter késleltetési idejének arányától függően – alakul ki a stabil helyzet. Ez időben és más paraméterek (pl. hőmérséklet) függvényében még változhat is. A másik dolog, hogy nincsen az áramkörnek bemenete, amelyen keresztül a kívánt állapotot beállíthatnánk. A kapuk bemeneteire közvetlenül kimenetek kapcsolódnak, itt tehát nem lehetséges a bemenetek megvalósítása.

A flip-flop belső állapotának megváltoztatására különböző vezérlő bemeneteket és billenési módokat definiálhatunk. Ezek alapján különböztetjük meg a flip-flopok típusait.

## *Flip-flop-ok típusai*

Működtetés szerint megkülönböztethetünk

- közvetlen vezérlésű,
- kapuzott vezérlésű

tároló elemeket, attól függően, hogy az új állapot közlését és a tárolóba történő beírását ugyanaz a jel, vagy különböző jelek végzik el. Kapuzott vezérlésű flip-flopok esetén megkülönböztethetünk

- együtemű,
- kétütemű

vezérlésű tárolót. A tárolandó állapot beviteli módja alapján megkülönböztethetünk

- R-S

- J-K
- T
- D
- DG

Flip-flopokat. A jelölések a bemenetek számára és elnevezésire utalnak. A flip-flopok billentési módja, tehát az adat beírása szerint megkülönböztethetünk

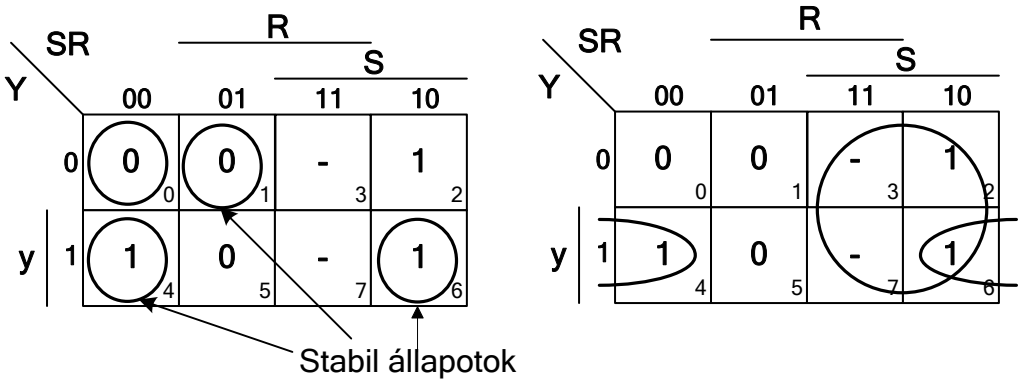
- statikus
- dinamikus

billenésű tárolókat. Működési mód tekintetében a flip-flopok lehetnek szinkron vagy aszinkron rendszerűek. Minden flip-flop típust meg lehet valósítani szinkron módon, de nem mindegyiket lehet aszinkron módon. Erre a konkrét típusok ismertetésénél visszatérünk.

### R-S flip-flop

Gyakran nevezik S-R flip-flop-nak is. Az elnevezés a két vezérlő bemente re utal. $S$ (set, beíró) és $R$ (reset, törlő) bemenetek. Ennek megfelelően a flip-flop állapota '1' értékűre vált, ha a beíró $S$ bemente kap vezérlést és '0' állapotúra vált, ha a törlő bemenetre. Ha mindkét bemenet inaktív, a flip-flop állapota nem változik. Ellentmondásra vezetne, ha a két vezérlő bemenet egyidejűleg lenne aktív, ezért a helyes működés feltétele, hogy a két vezérlő bemenet egyidejűleg nem kaphat aktív vezérlést. A valóságban ilyen esetben az áramkör belső késleltetései határozzák meg, hogy mi lesz a kimenet állapota, tehát ez semmiképpen nem tekinthető specifikusnak.

A fenti működési feltételek alapján megtervezhetjük a flip-flop-ot. Első lépésként írjuk fel a működést leíró vezérlési táblát. A táblázat kitöltését úgy végezzük, hogy a vízszintes sorokban szereplő $y$ értékeket úgy tekintjük, mint a flip-flop előző állapotát, majd a függőleges sorokban szereplő bemeneti kombinációk alapján eldöntjük, hogy mi lesz a flip-flop következő állapota. Tehát, milyen előző állapból, milyen következő állapotba vált át a flip-flop a bemeneti vezérlőjelek hatására. Ezeket a „következő állapot"-okat írjuk be a táblázat rubrikáiba. Az $SR = 11$ oszlopba mindkét előző állapot után közömbös értéket írunk, hiszen ez a bemeneti kombináció tiltott, nem fordulhat elő.

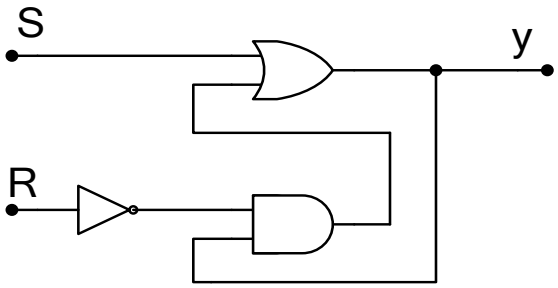



**3.2 ábra R-S flip-flop állapottáblái**

A hálózatot ezután stabilitás szempontjából is megvizsgáljuk. Minden oszlopban kell lennie legalább egy stabil állapotnak, hiszen ellenkező esetben az ehhez tartozó bemeneti kombináció fellépése estén az áramkör oszcillálni kezd. A stabil állapotokat bekarikáztuk. Ilyet most csak az $SR = 11$ oszlopban nem találunk, érthető módon. A következő ábra az egyszerűsítést mutatja, a közömbös állapotokat most '1'-ként rögzítettük. Innen az egyszerűsített függvényalak kiolvasható.

$$f(S, R, y) = S + \bar{R}y$$

Ez alapján felrajzolhatjuk az R-S flip-flop működésének megfelelő hálózatunkat.

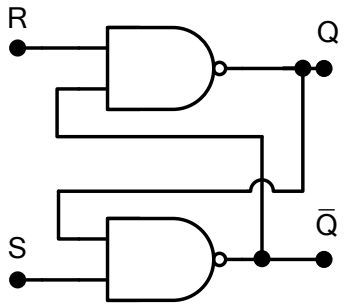



**3.3 ábra R-S flip-flop realizációja**

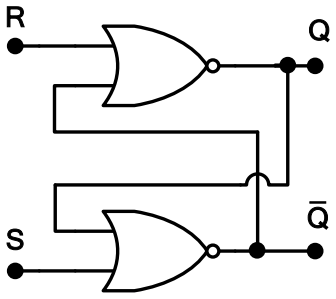
A működést visszacsatolt kombinációs hálózattal valósítottuk meg. A hálózat $y$ kimenete egyben a hálózat belső állapota is. Ezt a változót csatoltuk vissza, ez biztosítja az áramkör állapotának stabil tárolását inaktív bemeneti állapotban is.

A gyakorlatban az R-S flip-flop-ot egyforma kapuáramkörökből valósítják meg, vagy integrált áramköri kivitelben kész flip-flop-ot használnak. Az egyforma kapuáramkörökből történő megvalósítás kétféle működést eredményezhet a választott kaputípustól függően.

R-S0 flip-flop (NAND kapukkal)



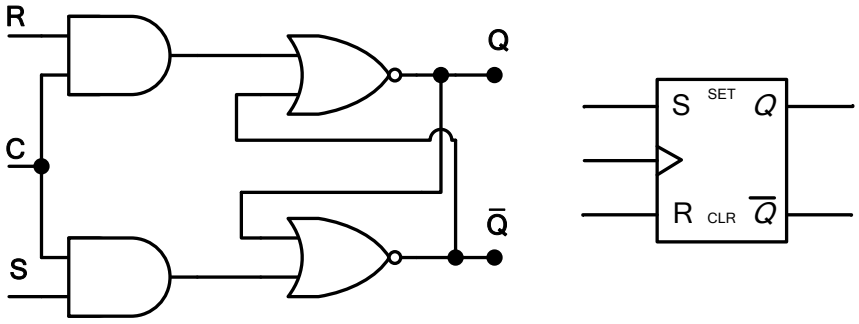

R-S1 flip-flop (NOR kapukkal)




**3.4 ábra R-S0 (NAND) és R-S1 (NOR) flip-flop-ok**

A fenti ábrán látható úgynevezett R-S0 flip-flop NAND kapukkal történő realizációja látható. Az elnevezés ara utal, hogy az áramkör aktív alacsony jelszintre kapcsol. Az R-S1 flip-flop hasonló kapcsolású, de NOR kapukból épül fel. Itt az aktív logikai szint a magas.

A statikus vezérlésű R-S1 flip-flop vezérlőbemeneteit egy beíró órajellel megkapuzva kaphatjuk a kapuzott vagy szinkron vezérlésű RS flip-flop-ot. Ennél a kapuzott változatnál az R és S vezérlőjelek együttes aktív állapota csak a kapuzó-jel aktív állapota mellet tiltott.




**3.5 ábra: szinkron vezérlésű R-S flip-flop**

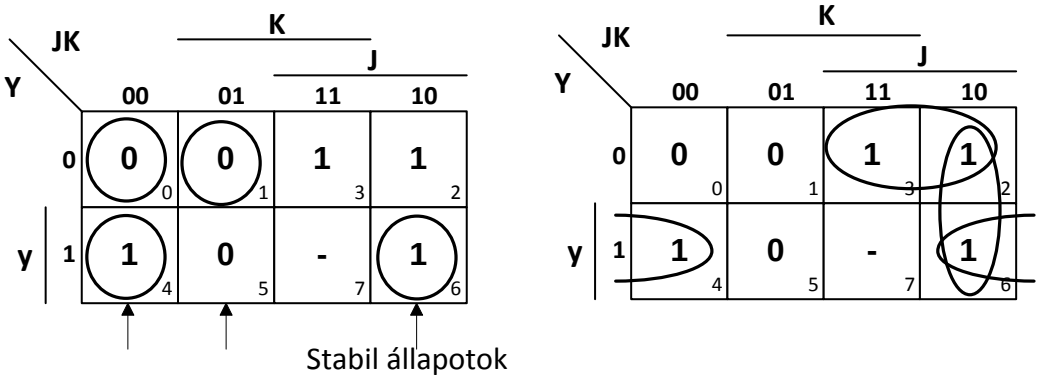
Amennyiben a R-S0 flip-flopot akarunk kapuzott bemenetűvé alakítani, a kapuzást VAGY kapukkal tudjuk megoldani, mivel itt 0 az aktív szint a bemeneteken.

### J-K flip-flop

Az R-S flip-flop azon hibáját, hogy a két bemenetére adott egyidejű aktív vezérlés nem specifikált állapot a J-K flip-flop használatával küszöbölhetjük ki. A J-K flip-flop blokkvázlata hasonló az R-S flip-flop-hoz a $J = S,\ K = R$ jelölések bevezetésével. A J-K flip-flop működése teljes egészében megegyezik az R-S flip-flop-éval, azzal a különbséggel, hogy megengedjük a $JK = 11$ bemeneti állapotot.

Ebben az állapotban definíció szerint a flip-flop invertálja az elő állapotát. Ha eredetileg '0' volt, '11'-re vált és fordítva.

Ennek megfelelően a J-K flip-flop vezérlési állapottáblája a következőképpen alakul.



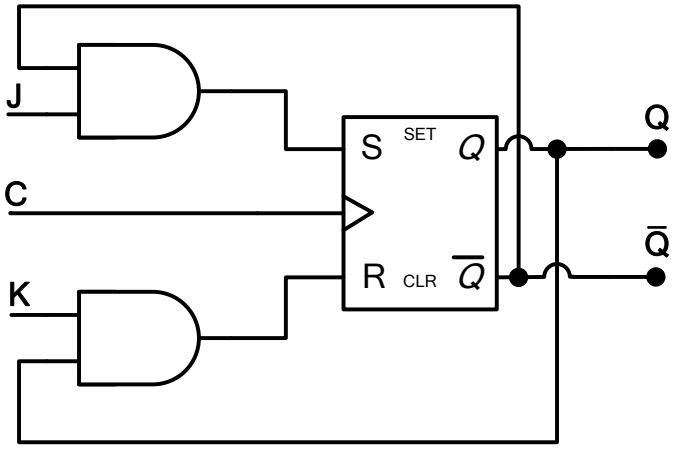

**3.6 ábra J-K flip-flop állapottáblái**

Látható, hogy csak a harmadik oszlopban van eltérés a két táblázat között. A stabil állapotok kijelölésekor azonban észrevesszük, hogy a harmadik oszlopban nincs stabil állapot. A bemeneten kialakuló $JK = 11$ állapot esetén tehát oszcilláció lép fel. Ez egyetlen módon kerülhető el, ha a flip-flop-ot szinkron módon valósítjuk meg, azaz a bemeneti változók által leírt új állapot kialakulásához szükséges még egy szinkronizáló jel órajel is.

Az egyszerűsítés után kiolvasható a flip-flop-ot megvalósító hálózat logikai függvénye.

$$f(J,K,y) = (J\overline{K}) + J\overline{y} + \overline{K}y$$

A zárójelben lévő tag csak a hazárdmentesítés miatt szükséges, de e nélkül a hálózat hajlamos lehet a helytelen működésre. A realizált kapcsolás felrajzolásától most eltekintünk, ezt mindenki maga is elvégezheti. Az így realizált flip-flop még nem teljesen működésképes az előbb már említett instabil állapot miatt. A flip-flop szinkron vagy kapuzott megvalósítása következő ábrán látható.



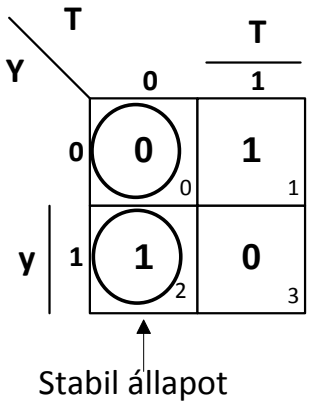
**3.7 ábra kapuzott J-K flip-flop**

Itt a kapuzójelek engedélyezése a kimeneti jelekkel $Q$, $\bar{Q}$ történik. Így biztosítható, hogy a flip-flop magas ’1’ állapotában csak a $K$ törlőbemenet, alacsony ’0’ állapotában csak a $J$ beíró bemenet eredményez állapotváltozást. Tehát mindig a flip-flop előző állapota határozza meg a bekövetkező állapotváltozást. Ez a kapuzási elv lehetővé teszi a $J$ és $K$ bemenetek egyszerre történő vezérlését is.

## T flip-flop

A legegyszerűbb felépítésű flip-flop. Származtatni a J-K flip-flop-ból lehet egyszerűen. Ha egy J-K flip-flop bemeneteit összekötjük, azaz csak 00 vagy 11 bemenőjelekkel vezéreljük akkor a T flip-flop-nak megfelelő működést kapunk.

Ezek alapján a T flip-flop vezérlési táblája a következő.



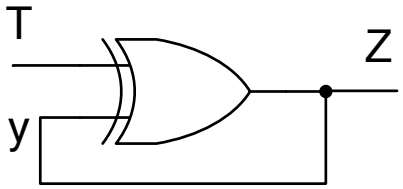

**3.8 ábra T flip-flop állapottáblája**

Vegyük észre, hogy a JK flip-flop-hoz hasonlóan itt is csak egyik oszlopban vannak stabil állapotok. Ez azt jelenti, hogy ezt a flip-flop-ot is csak szinkron módon realizálhatjuk.

Egyszerűsítés után a következő függvényalakot kapjuk,

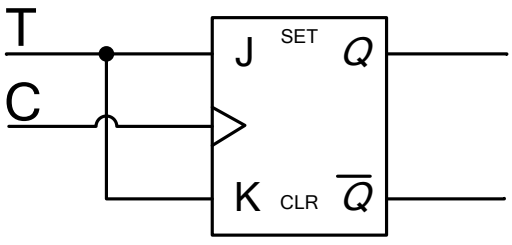
$$f(T,y) = T\bar{y} + \bar{T}y = T \oplus y$$

ami a kizáró-vagy kapcsolat függvénye. A realizáció tehát egyszerűen megoldható egy visszacsatolt logikai kapuval, ami természetesen még nem biztosítja a szinkron vezérlés lehetőségét.




3.9 ábra T flip-flop realizációja

A T flip-flop szinkron vezérelhető változata a következő ábrán látható.



**3.10 ábra T flip-flop megvalósítása J-K flip-flop-ból**

Integrált áramköri változatban ezt a flip-flop típust nem gyártják, mivel J-K flip-flop-ból egyszerűen kialakítható.

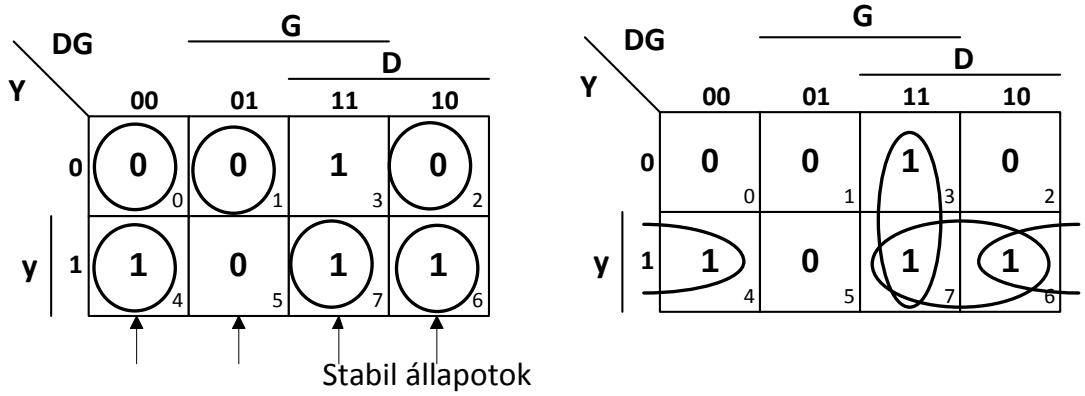
### D-G flip-flop

A D-G flip-flop szintén egy két bemenetű flip-flop. A $D$ (data, adat) és $G$ (gate, kapu) bemenetekre adott vezérlő jelek a következőképpen határozzák meg a flip-flop működését. Ha $G$ magas aktív szintű, akkor a flip-flop kimenete megegyezik a $D$ bemenettel. Ha $G$ alacsony, inaktív akkor a kimenet az inaktívvá válás előtti utolsó $D$ értéket őrzi meg.

$$\text{Ha } G = 1 \Rightarrow y = D$$

$$\text{Ha } G = 0 \Rightarrow y = Y$$

Ezt a flip-flop típust nevezik *latch* áramkörnek is, és mikroprocesszoros környezetben gyakran alkalmazzák pl. a címbusz állapotának tárolására, vagy kimeneti tároló elemként. Működésének ismeretében felírhatjuk a vezérlési tábláját.




**3.11 ábra D-G flip-flop állapottáblái**

Mivel minden oszlopban található legalább egy stabil állapot, aszinkron módon is megvalósítható a flip-flop. Integrált áramköri kivitelben is létezik mind szinkron, mind aszinkron realizációja.

Az egyszerűsítés és az abból kiolvasható függvényalak:

$$f(D, G, y) = \bar{G}y + DG + (Dy)$$

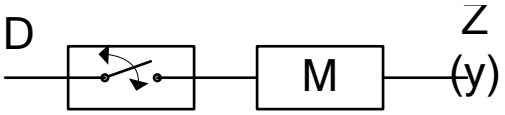
A zárójeles tag a hazárdmentesítés miatt szükséges.

### D flip-flop

Gyakoribb nevén D tároló. Működése nagyon egyszerű, a $D$ bemenetére kerülő adatot tárolja mindaddig, míg újabb adat tárolására szolgáló jel nem érkezik a bemenetére. Így önmagában a $D$ bemenet nem elegendő, szükséges egy beíró bemenet is. Ez a szinkron működést biztosító órajel lesz.

A flip-flop előbb ismertetett módon való tervezése az $y = D$ függvényt adja, aminek realizációja egy $D$-t $y$-al összekötő vezeték.

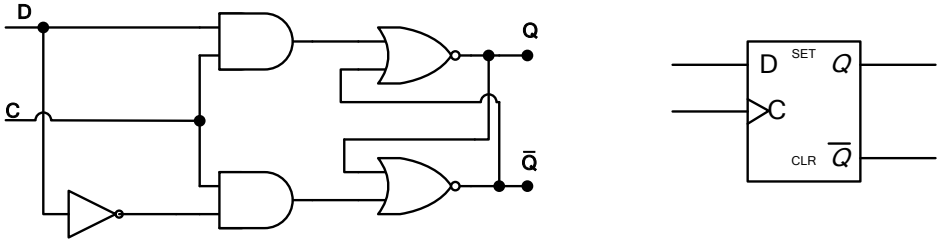

A szinkron működést is megvalósító blokkvázlat a következő ábrán látható. Itt a szinkronizáló órajel kapcsolja a tárolóelem bemenetén lévő kapcsolót.




**3.12 ábra „ aszinkron D flip-flop” realizációja**

Mivel ez a flip-flop is csak szinkron módon realizálható, a szinkron működést megvalósító bemenetet a tervezéskor kell megvalósítani.

A D flip-flop kialakítható kapuzott bemenetű RS flip-flop-ból is a következő ábra szerint.




**3.13 ábra D flip-flop megvalósítása R-S flip-flop felhasználásával**

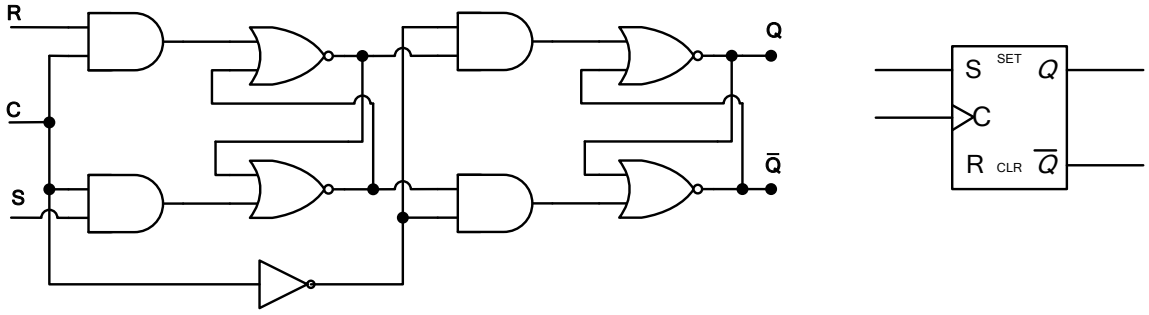
A plusz inverter beépítésével a flip-flop csak $RS = 01$ vagy $RS = 10$ vezérlést kaphat, tehát beírás vagy törlés parancsot. Ez megegyezik a D flip-flop működésével az előbb leírtak szerint.

Fontos megjegyezni, hogy az integrált áramkörként realizált flip-flop-okat nem az itt ismertetett módon tervezik, hanem aszinkron sorrendi hálózatként valósítják meg őket, visszacsatolt logikai hálózatot alkalmazva. Ez a szinkron működést lehetővé tevő órajel bemenet miatt szükséges, amit az itt bemutatott realizációk nem tartalmaznak. Ezek alól csak az aszinkron módon is realizálható RS és DG flip-flop-ok kivételek, amelyek az itt ismertetett kapuáramkörös realizáció esetén is működőképesek.

Bizonyos digitális áramköri feladatokban szükség lehet olyan megoldásokra, amikor a flip-flop kimenete csak akkor veszi fel az új állapot értékét, ha a bemeneti vezérlőjel már inaktív. Ez a feladat közbenső-tárolót alkalmazó flip-flop-al vagy élvezérléssel oldható meg.

### Közbenső tárolós (Master-Slave) flip-flop

A közbenső tárolós flip-flop legegyszerűbb megvalósítása RS flip-flop alkalmazásával, a következő ábrán látható.




**3.14 ábra Közbenső tárolós (Master-Slave) flip-flop**

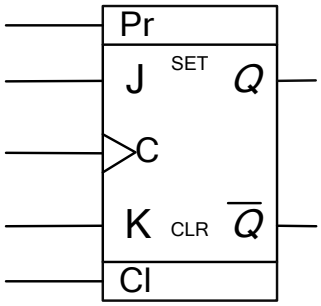
A slave egység a már megismert kapuzott RS flip-flop. Kapuzó órajelét egy inverteren keresztül kapja. A slave egység előtt egy teljesen azonos felépítésű másik RS flip-flop található. Ez az egység kapja közvetlenül a ponált értékű órajelet. A kapuzó órajel aktív állapotára először a master egységbe íródik be a vezérlő bemenetek által definiált új érték. Ekkor a slave egység kapuzó jele inaktív, a kimeneten

nincs változás. A kapuzó órajel inaktívvá válásakor az inverter utáni slave egység kap aktív kapuzójelet és beíródik az adat, mely most már megjelenik a kimenten.

## Közbenső tárolós flip-flop aszinkron billentésének megvalósítása

A többirányú alkalmazhatóság érdekében az integrált áramköri flip-flop-ok gyakran rendelkeznek aszinkron beíró és törlő bemenetekkel is. Ekkor a felhasználó dönti el, hogy melyik módon kívánja az áramkört vezérelni. Természetesen a nem használt bemeneteket inaktív szintre kell kötni. Lehetőség van a szinkron és aszinkron bemenetek egy alkalmazáson belüli együttes használatára is.

Az aszinkron vezérlés esetén a vezérlő jel a flip-flop mindkét tárolóját egyszerre billenti a következő állapotba. Az aszinkron vezérlő bemenetek a $Pr$ (preset) és a $Cl$ (clear).

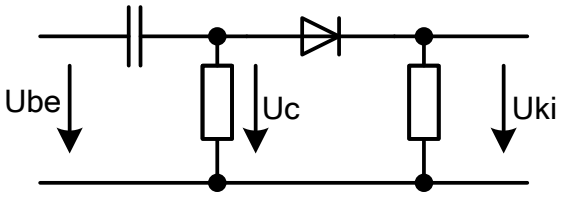



**3.15 ábra Közbenső tárolós (Master-Slave) flip-flop aszinkron beíró bemenetekkel**

## Élvezérelt, dinamikus billentésű flip-flop-ok

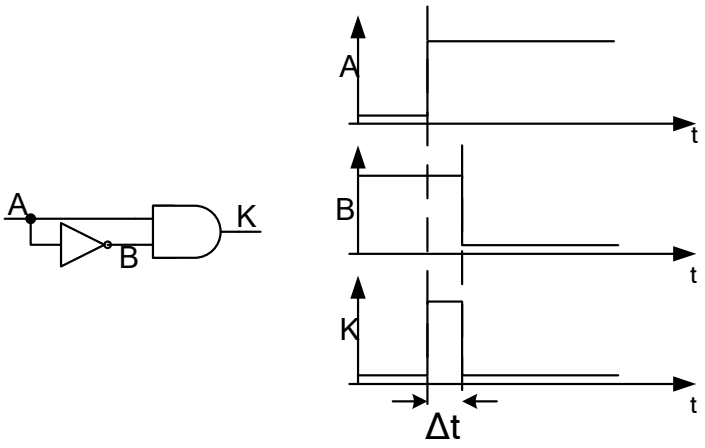
Az eddig ismertetett flip-flop-ok vezérlése statikus jelekkel történt. A flip-flop-ok egy másik nagy csoportja a dinamikus vagy élvezérelt flip-flop-ok. Ezen flip-flop-ok vezérlőjeleit olyan áramköri elemek állítják elő, melyek kimenetén csak bemenetei szintváltozás esetén jön létre logikai jel. Az ilyen áramköröket trigger áramköröknek nevezzük, és megvalósíthatóak diszkrét elemekből, valamint integrált áramköri kapukból is.

Az alábbi ábrán a diszkrét megvalósítású trigger áramkör rajza látható. Ez a legegyszerűbb megoldás, amely csak passzív áramköri egységeket tartalmaz. Legfontosabb eleme a bemeneten található kondenzátor, mely az azt követő ellenállással egy differenciáló tagot alkot. Statikus esetben a kimenet alacsony szinten van. Az Ube ponton csak a bemenetei szintváltozás esetén van jel. A dióda a magas-alacsony jelátmenet esetén kialakuló negatív polaritású feszültséget választja le. Végeredményben az áramkör kimenetén csak az alacsony-magas jelátmenet esetén jön létre rövid idejű jelimpulzus. Hasonló meggondolások alapján készíthető magas-alacsony jelátmenet esetén kimeneti jelet adó áramkör is.




**3.16 ábra Élvezérlés egy lehetséges megvalósítása**

Az élvezérlés integrált áramkörök alkalmazásával is megoldható. Ebben az esetben azt használjuk ki, hogy minden kapuáramkör rendelkezik bizonyos jelkésleltetéssel a bemenete és kimenete között. A kimeneti jelimpulzust tulajdonképpen egy hazárd hozza létra, a következő ábra szerint.




**3.17 ábra Élvezérlés megvalósítása kapukésleltetés felhasználásával**

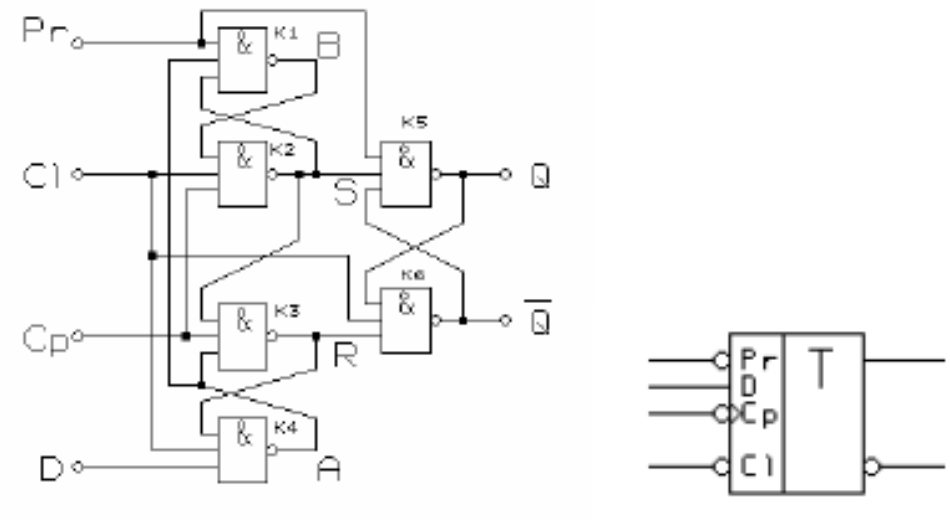
Az impulzus hosszúságát az inverter késleltetési ideje és a logikai kapu belső késleltetései határozzák meg. Hosszabb impulzus előállítása több páratlan számú inverter sorba kapcsolásával valósítható meg.

## *Flip-flop-ok integrált áramköri megvalósításai*

Az eddigi ismereteink alapján megállapíthatjuk, hogy a flip-flop-ok bizonyos típusai egymásba könnyen átalakíthatók vagy működésük nagyon hasonló, ezért egymás feladatát kiválthatják. Például J-K flip-flop-ból könnyen készíthetünk T flip-flop-ot, vagy a J-K esetén az „11” bemeneti kombináció kizárásával az R-S-hez hasonló működést kapunk. A D flip-flop is kialakítható J-K-ból és nagyon elterjedten alkalmazzák. D-G flip-flop-ot Latch elnevezéssel gyártanak külön áramköri elemként. Ezért integrált áramköri formában legelterjedtebbek a D és a J-K flip-flop-ok valamint a Latch tárolók. Integrált áramköri formában az élvezérléshez szükséges trigger áramkör kapacitásai a technológiai méretkorlátok miatt nem valósíthatók meg.

## *Élvezérelt D flip-flop*

Mivel kapacitív tagot tartalmazó differenciáló tagot az integrált áramköri megvalósításokba nem építenek be, a kapuáramkörök késleltetési idejét használják fel a rövid idejű késleltetések, tehát az élvezérlést kiváltó hazárdok létrehozásakor. Az élvezérelt D flip-flop SN7474 típusú integrált áramköri megvalósítását, mint lehetséges áramköri realizációt a következő ábrán láthatjuk.




a. b.

**3.18 ábra Élvezérelt D flip-flop megvalósítása**

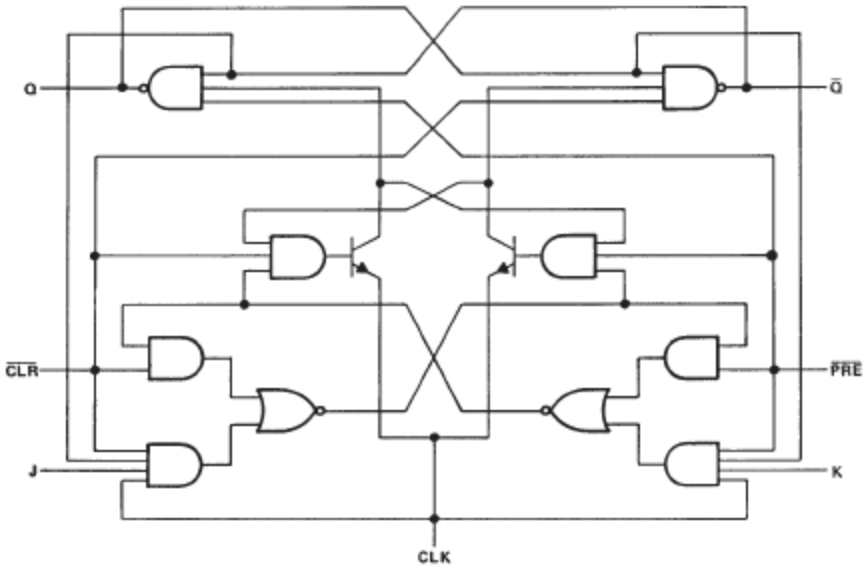
$Pr$ és $Cl$ alacsony aktív aszinkron beíró és törlő bemenetek. $Cp$ bemenet felfutó éle billenti a tárolót, vagyis írja be a flip-flop-ba a $D$ bemeneten levő értéket. A $D$ bemenet közvetlenül a K4, invertálva a K1 jelű ÉS kapura kerül. $Cp$ alacsony értékénél K2 és K3 kimenetei a többi bemenettől függetlenül magasak szintűek. K5, K6 ÉS kapukból felépített tároló bemenetei inaktívak, így az előző értéket tárolják. Ez az úgynevezett előkészítő ütem. A $Cp$ bemenet változásakor a $D$ bemenet értéke a K1, K2, illetve K3, K4 kapukból felépített tárolókból, a K2, K3 kapuk kimeneteinek engedélyezése után átíródik a K5, K6 Kapukból felépített tárolóba. A tároló bemenetei az

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

értékeket veszik fel. A $Cp$ billentőjel változása után, a kapukésleltetések lejártával a $D$ bemenet értéke már nincsen hatással a flip-flop belső állapotára, $D$ megváltozása már nem változtatja meg $S$ és $R$ értékét. A belső késleltetés lezajlásáig tehát fent kell tartani $D$ állapotát a billentőjel felfutása után is, ennek lejártával lesz csak $D$ bemenet állapota hatástalan. A flip-flop újabb állapotváltozása, tehát a $D$ bemeneten levő jel beírása csak a $Cp$ bemeneten fellépő újbóli 0 → 1 állapotváltozáskor következik be.

### Élvezérelt J-K flip-flop

Az SN 7476 típusú MS J-K flip-flop logikai felépítését mutatja a következő ábra. Ez a típus is rendelkezik aszinkron beíró és törlő bemenetekkel, $PRE$ és $CLR$. Az ezekre a bementekre jutó aktív logikai érték közvetlenül beírja vagy törli a Slave R-S flip-flop-ot és tiltja a $J$, $K$ bemenetekhez tartozó Master tárolót. Aktív alacsony szintű bemenetek. Az aszinkron vezérlőjelek inaktív állapota mellett a CLK billentő órajelhez kapcsolódó két közös emitterű tranzisztor gondoskodik a CLK bemenetre jutó 0 → 1 jelátmenetnél a Master és a Slave tárolók elválasztásáról. A CLK bemenet kb. 0,7V-ot meghaladó értéke esetén ugyanis a két tranzisztor lezárás a tranzisztorok bázisait vezérlő logikai kapuk kimenetei hatástalanná válnak. Erre azért van szükség, hogy az ellenütemű vezérlést biztosító áramköri elemek késletetése nagyobb, mint a Slave flip-flop-é, akkor a Slave vezérlésének tiltása előtt beíródhat információ a flip-flop-ba, ami hibás működést eredményezhet.

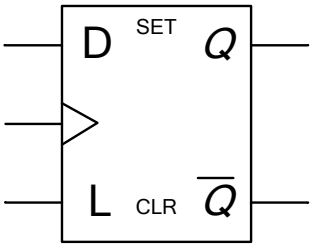



**3.19 ábra Élvezérelt J-K flip-flop megvalósítása**

## *Több bites tároló áramkörök (latch)*

Az előzőekben már említett D-G flip flop alapvető felhasználási területe a több, általában 8 bites adat- vagy címvonalak állapotainak ideiglenes tárolása, például multiplexelt adat és címbuszt használó mikroprocesszoros rendszerek esetén. Valamint adat be- és kivitelek szinkronizációit

biztosító áramkörökben. A gyakorlatban a tárolót statikus (pl. 74LS573) vagy dinamikus (pl. 74LS574) billentésű D tároló elemekkel valósítják meg és nem a flip-flop-ok működését leíró fejezetben foglaltak szerint. A mikroprocesszoros rendszerek buszaihoz való illeszthetőség miatt a tárolók kimeneteit közösen vezérelt engedélyező bemenettel látják el.



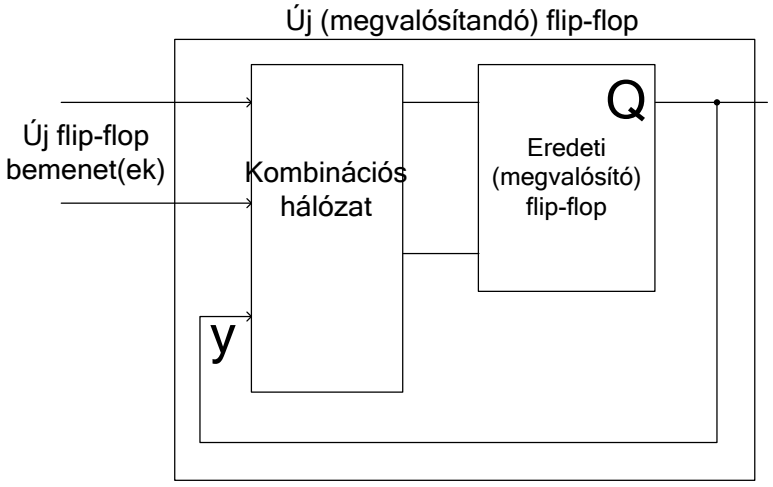

**3.20 ábra 74LS574 D flip-flop**

## *Flip-flop-ok átalakítása másik típusú flip-floppá*

Gyakran előfordul, hogy a meglévő flip-flop típus helyett másfajta flip-flop-ra van szükségünk, vagy hely, illetve tok takarékosság miatt nem célszerű egy újabb áramkör beépítése. Ezek helyett már meglévő, de fel nem használt áramköri egységeket célszerű átalakítani. Például, ha egy két flip-flop-ot tartalmazó tokból csak az egyik flip-flop-ot használjuk fel, de szükségünk lenne még egy másik fajta flip-flop-ra is, célszerű lehet, néhány kapuáramkör felhasználásával ennek átalakítása kívánt flip-flop típusra.

Az átalakítás menete egyszerű és bármilyen flip-flop bármilyen típusúvá átalakítható. Egy dolgot fontos figyelembe venni, hogy szinkron flip-flop-ból csak szinkron flip-flop készíthető és aszinkron flip-flop-ból szinron flip-flop-ot csak a szinkronizáció biztosításával készíthetünk.

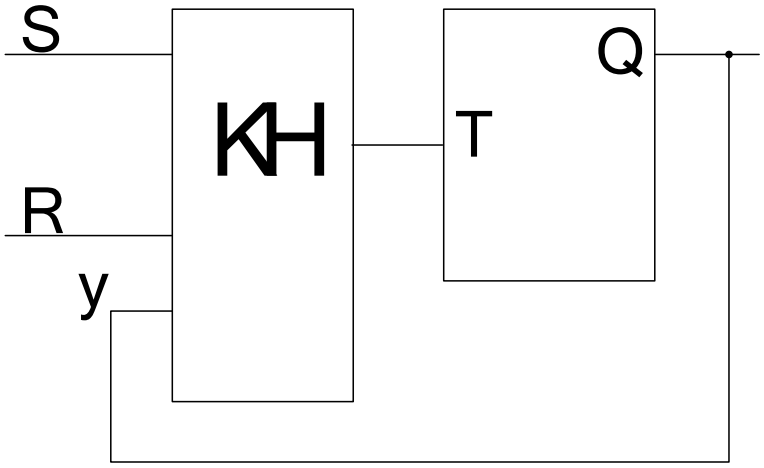
Az átalakítás elve az, hogy a kiindulási (eredeti) flip-flop elé egy kombinációs hálózatot tervezünk, melynek bemenete(i) az új flip –flop bemenete(i) és a kiindulási flip-flop kimenete (y). A kombinációs hálózat a mindenkori állapotot figyelembe véve úgy vezérli a kiindulási flip-flop-ot, hogy annak kimeneti állapota az új flip-flop állapotainak megfelelően viselkedjen.




**3.21 ábra flip-flop átalakítása másik típusú flip-floppá**

Lássunk két egyszerű példát, melyek segítségével könnyen érthetővé válik az átalakítás menete.

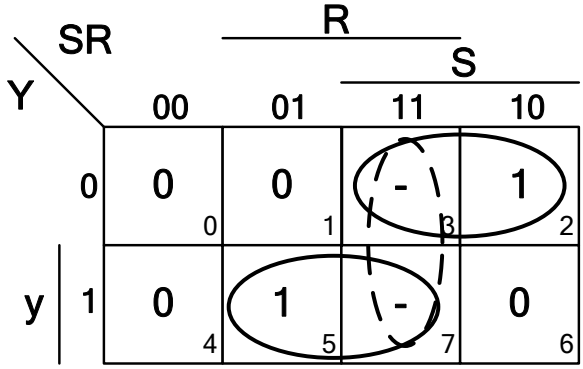
Első példa: T flip-flop átalakítása R-S flip-floppá.




**3.22 ábra T flip- flop ból R-S flip-flop készítése**

A fenti ábrán látható a flip-flop felépítése. Mivel R-S flip-flop-ot készítünk, a vezérlő kombinációs hálózat bemeneteihez fog csatlakozni az új flip-flop $R$ és $S$ bemenete. Az új flip-flop kimenete az eredeti kimenettel megegyező, de visszacsatoljuk a kombinációs hálózat bemenetére is.

A tervezéshez fel kell írnunk az új flip-flop-nak megfelelő vezérlési táblát, és ki kell töltenünk úgy, hogy a kombinációs hálózatunk az új flip-flop működésének megfelelően vezérelje az eredeti flip-flop bementét.



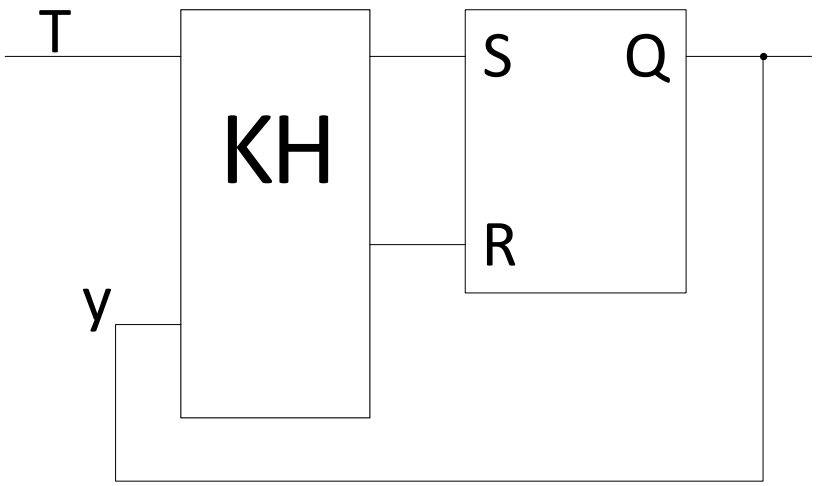

**3.23 ábra Az új R-S flip-flop vezérlési táblájának tervezése**

A táblázat kitöltésének menete a fenti ábrán követhető. Kezdjük a 0-as indexű termel. Az eredeti flip-flop állapotát a kis *y* sora adja meg. Ez most „0”. Ha ekkor $SR = 00$ bemeneti kombináció kerül az RS flip-flop bemenetére ez nem okoz változást a kimeneten. Ahhoz, hogy a T flip-flop se változtassa meg a kimenetét, a bementére 0-t kell adni. Ezért kerül 0 a bal felső (0 indexű) term-be. Hasonló meggondolásból írunk 0-át az 1-es indexű term-be is, hiszen az $SR = 01$ bemeneti kombináció sem változtatja meg a flip-flop állapotát. Az $SR = 11$ bemeneti kombinációhoz közömbös állapotot rendelünk, a már korábban említett okok miatt. Az $SR = 10$ bemenet beírási feltétel, tehát a flip-flop kimenetének most 0-ról 1-re kell változnia. Ehhez a T flip flop bemenetére 1 logikai értéket kell adnunk. A 2-es indexű termbe tehát 1-et írunk. Azonos gondolatmenettel tölthetjük ki a táblázatunk második sorát is, figyelembe véve, hogy most a kiindulási állapotunk az eredeti flip-flop 1-es állapota.

A kitöltött táblázatból az állapot összevonások után a kombinációs hálózat függvénye kiolvasható.

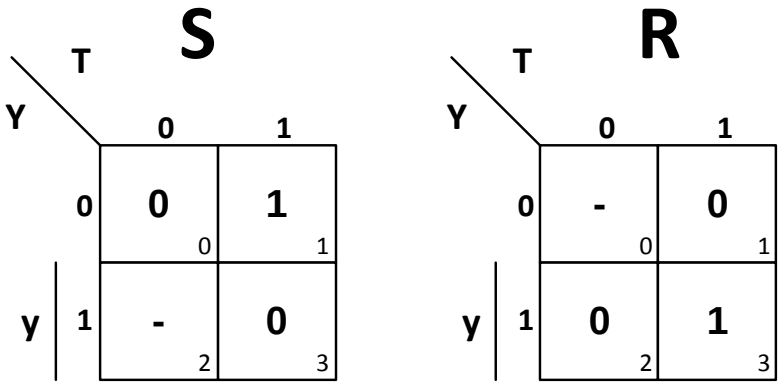
$$f(S,R,y) = Ry + S\bar{y} + (SR)$$

Nézzünk egy másik példát, amikor két bemenetű flip-flop-ból kell kiindulni (eredeti flip-flop) és a tervezendő flip-flop egy bemenettel rendelkezik. Példaként tekintsük most az előző feladat fordítottját, tehát a rendelkezésünkre álló flip-flop egy R-S és ebből szeretnénk T flip-flop-ot készíteni. A megvalósítandó flip-flop blokkvázlatát a következő ábrán láthatjuk.




**3.24 ábra R-S flip-flop-ból T flip-flop készítése**

Ez esetben a megtervezendő kombinációs hálózatunknak egy vezérlő bemenete van a $T$, plusz az $y$ szekunder változó. Az eredeti flip-flop vezérléséhez viszont két kimenettel fog rendelkezni a hálózat. A tervezés menete teljesen megegyezik az előzőekben ismertetett lépésekkel




**3.25 ábra Az új T flip-flop vezérlési táblájának tervezése**

A táblázatunk viszont másképpen néz ki, mivel most két függvényt kell realizálnunk, egyet az $S$, egyet az $R$ flip-flop bemenet számára. A függvényeknek most csak két bemenetük van, ezért a táblázatok csak 4 rubrikát tartalmaznak.

Készíthetünk akár egy táblázatot is, amibe szerepeltetjük az $S$ és $R$ kimeneteket. A táblázatok kitöltését most célszerű egyszerre végezni, mert a következő állapot kialakításában értelem szerűen a flip-flop mindkét bemenete részt vesz. Kezdjük most is a 0-s indexű termmel. Az előző állapotot a $y$ sora jelöli ki, tehát 0 állapotból indulunk ki. Ha ekkor $T = 0$ kombináció (lásd táblázat feletti kis indexek) éri a hálózatot, a T flip-flop kimenete nem változik meg. Ehhez az R-S flip-flop $S$ bemenetére mindenképpen 0 értéket kell adnunk, $R$ bemenetére viszont a 0 és az 1 érték a megfelelő, hiszen mindkét bemenőjel esetén a flip-flop kimenete 0 értékű marad. Ezért kerül ide közömbös bejegyzés. A következő 1-es indexű rubrika fölötti kis 1-es jelzi, hogy most 1 értékű bemenetet kap a T flip-flop. Ekkor kimenetének állapota 0-ról 1-re vált. Ehhez az R-S flip-flop $S$ bemenetére '1', $R$ bemenetére '0' értéket kell adnunk.

A következő sor értelem szerint, 1-es állapotból 0 bemenet esetén a flip-flop kimenete nem változik. Ez $S =$„közömbös", $R = 0$ bemenetet jelent. Az utolsó, 3-as indexű term esetében, 1-es állapotban 1-es logikai bemenet esetén a T flip-flop törlődik, tehát $S = 0$, $R = 1$ a kívánt kimeneti érték. A külön felrajzolt táblázatok előnye a könnyebb állapotösszevonás. A két függvény ebben az esetben egyszerűen kiolvasható.

$$f_S(T,y) = T\bar{y} \qquad\qquad f_R(T,y) = Ty$$

Fontos még egyszer megjegyezni, hogy az áramkörök realizációjakor fokozottan figyelni kell a szinkronizáció biztosítására. Tehát példaként az utóbbi flip-flop csak szinkron R-S flip-flop-ból építhető meg, ahol az eredeti R-S flip-flop szinkronizáló bemente lesz az új flip-flop szinkron bemenete.

# 4. Szinkron sorrendi hálózatok tervezése

Szinkron sorrendi hálózatok tervezési lépései a következők (mindezt ***Mealy* ill. *Moore* modelleken** is külön-külön megvizsgálva, és megvalósítva):

1. **Logikai feladat megfogalmazása:** általában egy szöveges feladat, mely alapján az előzetes állapottábla összeállítása elkezdődhet.
2. **Előzetes állapottábla összeállítása.** Ehhez szisztematikus módszer nem létezik, a szöveges feladat megfelelő értelmezését feltételezve intuitív (sokszor próbálgatásos) jellegű. A feleslegesen megkülönböztetett állapotok a tervezés további lépéseiben összevonhatók, tehát nem jelentenek különösebb nehézségét.
3. **Egyszerűsített (összevont) állapottábla**. Ha az előzetes állapottábla teljesen specifikált (TSÁ), akkor erre a tervezési lépésre szisztematikus módszerek ismeretesek. Ha az előzetes állapottábla nem teljesen specifikált (NTSÁ), akkor pedig ugyanezek a módszerek használhatók, de a segítségükkel kapott eredményből csak intuitív lépések útján kaphatjuk meg a legegyszerűbb összevont állapottáblát.
4. **Állapotkódolás helyes megválasztása**. A sorrendi hálózat egyszerűsége szempontjából kedvező állapot-kódolás megválasztására léteznek módszerek, de egyikről sem bizonyítható az optimalitás.
5. **Alkalmazandó tároló típusának kiválasztása:** Erre a tervezési lépésre sem létezik egyértelmű szisztematikus eljárás, intuitív módon a megvalósító flip-flopok típusát többnyire a rendelkezésre álló építőelem-készlet (mintegy alkatrész bázis) alapján határozzuk meg.
6. **Vezérlési tábla összeállítása**. Ez viszont többnyire teljesen szisztematikus módszer (kivétel a D-G tároló használata). Csak akkor tudjuk a vezérlési táblát összeállítani, ha már pontosan ismerjük a kiválasztott flip-flop-ot. Lényegében a megvalósításra szánt flip-flop-ot vezérlő kombinációs hálózat állapot tábláját, mint vezérlési táblát kell ebben a lépésben megadni.
7. **Működés szemléltetése idődiagramon**. Ez teljesen szisztematikus eljárás. Manapság a komplex, több ezer-, vagy akár több millió kaput tartalmazó programozható rendszerek világában a tervezés nem papíron történik. Integrált fejlesztő eszközök segítik az automatizált tervezést és modellezést. A működés viselkedésének vizsgálatát szimulációs eszközök támogatják, amelyek segítségével idődiagramok is megjeleníthetők.

## *Szinkron sorrendi hálózat (Mealy modell) komplex tervezési feladata:*

### 1. Logikai feladat megfogalmazása:

Tervezzünk egy „soros” összeadót szinkron sorrendi hálózat Mealy megvalósítási modelljének segítségével, melynek a következő paraméterei adottak:

- 2 bemenete ($A$, $B$) és
- egy kimenete van ($F$).
- A bemenetekre, illetve a kimenetre órajel (CLK) ütemezéssel érkezzenek az adatok.

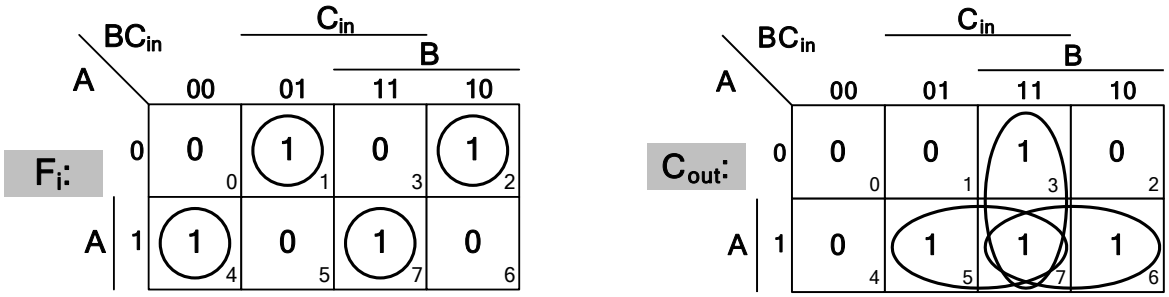
***Elméleti háttér***: A következő rész az összeadó kombinációs hálózat-szintű felépítését és működését segít megérteni, amely alapján a sorrendi hálózatok viselkedésére épülő, úgynevezett „soros” összeadó már könnyebben értelmezhető. A kombinációs hálózatokkal megvalósítható 1-bites összeadóknak két formája ismert:

- *Half Adder (Fél összeadó)*: ha adott $i$ helyiértéken nem kezeljük / generáljuk az átviteli értéket ($C_{out(i)}$), hanem csak a kimeneti értéket ($S_{(j)}$) állítjuk elő
- *Full Adder*: mind az átvitel bemenetet ($C_{in(i)}$), és kimenetet ($C_{out(i)}$) is kezeljük, ill. generáljuk.

  Tekintsük az 1-bites teljes összeadóból (Full Adder) felépített több bites összeadó egy lehetséges kombinációs hálózatként történő realizációját.


**4.1 ábra: 1-bites teljes összeadó szimbóluma**

| $A_i$ | $B_i$ | $C_{in(i)}$ | $F_i$ | $C_{out(i)}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |




$$F_{(i)} = A_{(i)} \oplus B_{(i)} \oplus C_{in(i)} \Rightarrow F_{(i)} = S^{n=3}{}_{1,3}(A_{(i)}, B_{(i)}, C_{in(i)})$$

$$C_{out_{(i)}} = A_{(i)} \cdot B_{(i)} + A_{(i)} \cdot C_{in_{(i)}} + B_{(i)} \cdot C_{in_{(i)}}$$

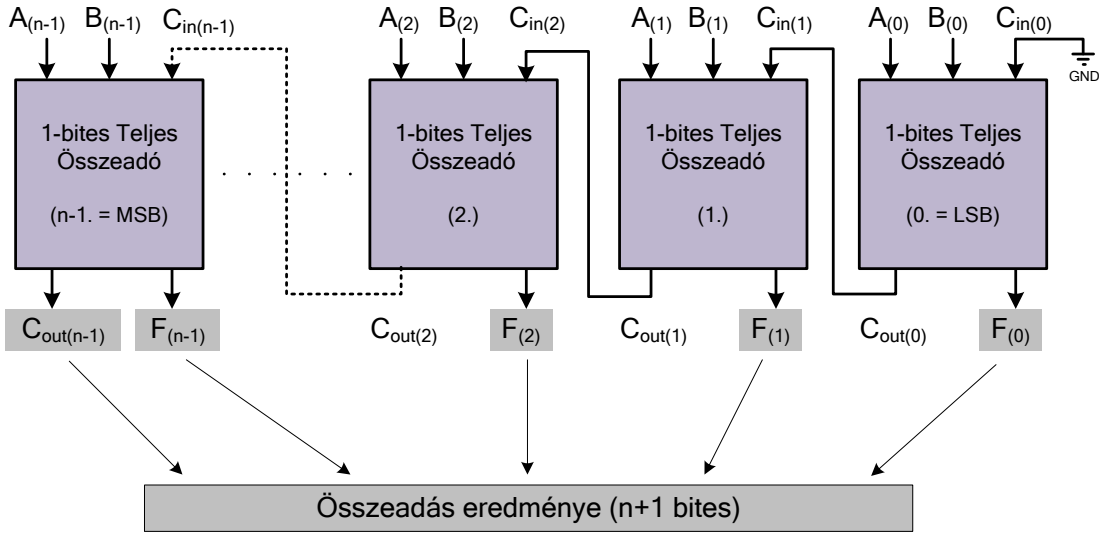
**4.2 ábra: 1-bites teljes összeadó kanonikus igazságtáblázata és Karnaugh táblái, és egyszerűsített DNF alakjai**

A fenti ismeretek birtokában egy N-bites Összeadót könnyen felépíthetünk a következő módon (lásd 4.3 ábra).

- N-db 1-bites Teljes Összeadót (FA) kell elhelyezni úgy, hogy
- Az adott $(i)$ pozícióban lévő Teljes Összeadó $C_{out(i)}$ kimenetét, a következő, eggyel nagyobb helyiértékű $(i+1)$ Teljes Összeadó $C_{in(i)}$ bemenetére kötjük (azaz mindig igaz, hogy $C_{in(i+1)} = C_{out(i)}$). Ezt mindaddig megtesszük, amíg a legkisebb helyirétékű (LSB) Teljes Összeadótól → a legnagyobb helyiértékű (MSB) teljes

összeadóig el nem jutunk. Ezt hívjuk kaszkádosításnak. A legalacsonyabb helyiértéken lévő $C_{in(0)}$ bemenetet a zajvédelem miatt földre (GND) szokás kötni! Az indexelést általánosan $i =$ '0' – tól '$n-1$'-ig definiálják.

- Az így kapott áramkört N-bites Teljes Összeadónak, avagy más néven N-bites *Átvitelkezelő összeadónak (Ripple Carry Adder)* hívjuk. Az összeadásnak természetesen vannak más fizikai megvalósításai, amelyeket most nem tárgyalunk.



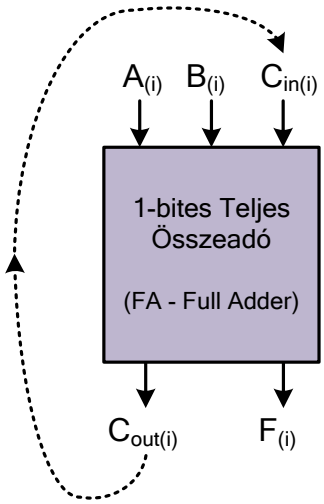

**4.3 ábra: Teljes Összeadókból felépített N-bites Átvitelkezelő felépítése**

**Soros összeadó:**

A sorrendi hálózattal megvalósítható *soros összeadó* a teljes összeadóhoz hasonlóan az MSB ← LSB felé haladva egymás utáni lépésekben adja össze a bináris mintázatokat (mint bináris számjegyek). Azonban nem $n$ db. 1-bites összeadó blokkot használunk fel a megépítéséhez, hanem egyetlen 1-bites összeadót, melynek átvitel vonalait, bemeneteit, kimeneteit megfelelő módon kell kezelni. A legfontosabb, hogy az átvitel kezelését (Carry) meg kell oldani az egyes helyiértékek között: ezért az átvitel kimenetet egy állapotként értelmezzük.

- $A(i)$ és $B(i)$ a két összeadandó bináris szám adott helyiértéken

Carry bemenetet az előző helyiérték felől érkező átvitel határozza meg ($C_{in(i)} = C_{out(i-1)}$)




Carry kimenet a következő helyiértékre csatolódik vissza ($C_{out(i)} = C_{in(i+1)}$). Ez a visszahatás biztosítja a soros összeadó iteratív működését

$$A_{(i)} + B_{(i)} + C_{in(i)} = \boxed{C_{out\,(i)}|F_{(i)}}$$

$0 + 0 + 0 = \boxed{0|0}$

$0 + 1 + 0 = \boxed{0|1}$

$1 + 0 + 0 = \boxed{0|1}$

$1 + 1 + 0 = \boxed{1|0}\,(carry\_out!)$

$0 + 0 + 1 = \boxed{0|1}$

$0 + 1 + 1 = \boxed{1|0}\,(carry\_out!)$

$1 + 0 + 1 = \boxed{1|0}\,(carry\_out!)$

$1 + 1 + 1 = \boxed{1|1}\,(carry\_out!)$

$C_{in(i+1)} \Leftarrow C_{out(i)}$

**4.4 ábra: Soros összeadó felépítése és működése**

**Különbség a teljes összeadó vs. soros összeadó között:** felépítésükből adódóan, míg egy N-bites összeadót N db egymás után kaszkádba kapcsolt 1-bites teljes összeadóból építjük fel, addig a soros összeadó esetén elegendő egyetlen 1-bites teljes összeadó, amelynél az iteratív működést az átvitelek visszacsatolása biztosítja. Ez utóbbinál az N. iteráció után kapjuk meg a végeredményt.

## 2. Előzetes állapottábla összeállítása

A S.H.-nak különbözőképpen kell reagálnia ugyanarra a bemeneti kombinációra a megelőző bemeneti kombinációktól és a hálózat állapotától függően.

Kérdés azonban az, hogy minimálisan mennyi állapot kell a megvalósításhoz? Szinkron Mealy modell esetén a soros összeadóra a következők érvényesek:

- „Előzetes” állapottáblában általában **több** állapot szerepel a szükségesnél.
- Kimenet pillanatnyi értéke: $F_{(i)}$
- Előző helyiértéken keletkezett átvitel (carry) értéke ($C_{in(i+1)} \leftarrow C_{out(i)}$) határozza meg a következő iterációban a carry bemeneti értékét

| **Állapot jele ($y$)** | **Előző helyiértéken keletkezett átvitel értéke: $C_{in(i)} = C_{out(i-1)}$** | **A kimenet pillanatnyi értéke: $F_{(i)}$** |
|---|---|---|
| a | 0 | 0 |

| | | |
|---|---|---|
| b | 0 | 1 |
| c | 1 | 0 |
| d | 1 | 1 |

**4.1 táblázat: az állapotoknak megfelelő szekunder kombinációk megadása**

***Fontos megjegyzés:*** *soros összegzéskor a számított kimeneti érték <u>független</u> a szekunder kombinációkhoz tartozó a pillanatnyi kimeneti értéktől, csak a bemeneti kombinációktól, illetve az előző helyiértéken keletkezett átviteltől <u>függ</u>.*

Előzetes állapottábla összeállítása: az állapottáblának összesen négy oszlopa (A,B bemeneti kombinációk miatt), és 4 sora (a,b,c,d – szekunder kombinációk száma miatt) lesz. Próbáljunk a kitöltés során ügyelni arra, hogy az oszlopok és sorok sorrendje lehetőleg kövesse a Karnaugh táblák peremezését, mivel ez később segíthet pl. a kimeneti függvény könnyebb leolvasásában. Kitöltés menete a következő:

**<u>Kitöltés</u>*: (1. sor)***

- Tfh. $\boldsymbol{y = a}$ és $\boldsymbol{AB = 00} \rightarrow F = A + B + C_{in} = 0$, és $C_{out} = 0 \rightarrow$ Ekkor $Y = a$ állapot és $F = 0$ kimenetre: $\boldsymbol{a0}$
- Tfh. $\boldsymbol{y = a}$ és $\boldsymbol{AB = 01}$ v $\boldsymbol{AB = 10} \rightarrow F = A + B + C_{in} = 1$ és $C_{out} = 0 \rightarrow$ Ekkor $Y = b$ állapot és $F = 1$ kimenetre: **b1**
- Tfh. $\boldsymbol{y = a}$ és $\boldsymbol{AB = 11} \rightarrow F = A + B + C_{in} = 10$ és $C_{out} = 1 \rightarrow$ Ekkor $Y = c$ állapot és $F = 0$ kimenetre: $\boldsymbol{c0}$

**<u>Kitöltés</u>*: (3. sor)***

- Tfh. $\boldsymbol{y = d}$ és $\boldsymbol{AB = 00} \rightarrow F = A + B + C_{in} = 1$, és $C_{out} = 0 \rightarrow$ Ekkor $b$ állapot és $F = 1$ kimenetre: $\boldsymbol{b1}$
- Tfh. $\boldsymbol{y = d}$ és $\boldsymbol{AB = 01}$ v $\boldsymbol{AB = 10} \rightarrow F = A + B + C_{in} = 10$ és $C_{out} = 1 \rightarrow$ Ekkor $c$ állapot és $F = 0$ kimenetre: $\boldsymbol{c0}$
- Tfh. $\boldsymbol{y = d}$ és $\boldsymbol{AB = 11} \rightarrow F = A + B + C_{in} = 11$ és $C_{out} = 1 \rightarrow$ Ekkor $d$ állapot és $F = 1$ kimenetre: $\boldsymbol{d1}$

Előzetes állapottábla

| y | AB: 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 a | Y F<br>a 0 | b 1 | c 0 | b 1 |
| 01 b | a 0 | b 1 | c 0 | b 1 |
| 11 d | b 1 | c 0 | d 1 | c 0 |
| 10 c | b 1 | c 0 | d 1 | c 0 |

Ha megvizsgáljuk a fenti előzetes állapottábla kimeneteit ($F$) tartalmazó oszlopokat, akkor az látható, hogy az egy szekunder kombinációhoz tartozó kimeneti értékek egy-egy soron belül különbözőek. Ez is a *Mealy modell* szerinti működést igazolja.

### 3. Összevont (egyszerűsített) állapottábla

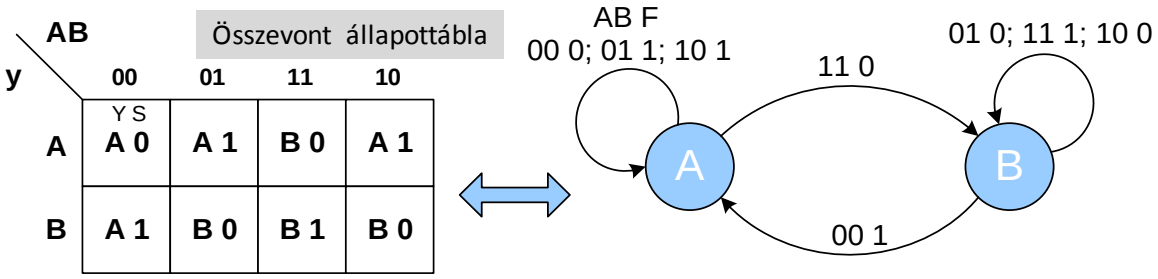
Előzetes állapottábla alapján az a cél, hogy megkeressük a lehető legkevesebb megkülönböztetendő állapotot. Ezek ismeretében kell létrehozni a lehető legkevesebb, de azonos sort nem tartalmazó új állapottáblát, amelyet összevont állapottáblának nevezünk, amelynek így már nem lesz feleslegesen megkülönböztetett állapota (=sora). Fontos tudni, hogy az összevont állapottábla nem azonos a később ismertetésre kerülő állapot kódolási módszerrel!

A fenti előzetes állapottábla következő sorait vonhatjuk össze:

szekunder állapotok

- $A := a, b$ (1.-, és 2. sor azonos)
- $B := c, d$ (4.-, és 3. sor is azonos)

Az összevonás során kapott összevont állapottábla és a vele ekvivalens állapotgráf a következő:




**4.5 ábra: Összevont állapottábla és állapotgráf**

### 4. Kódolt állapottábla

Az összevont állapottábla alapján a szimbolikus összevonásként ($A$, $B$) jelölt állapotoknak meg kell feleltetni egy-egy szekunder kombinációt ($y$).

Az összevont állapottábla sorainak (=állapotainak) számától függ, hogy mennyi különböző szekunder állapotot kell felvenni.

$n$ sor esetén: $\lceil \log_2(n) \rceil$ szekunder állapot szükséges

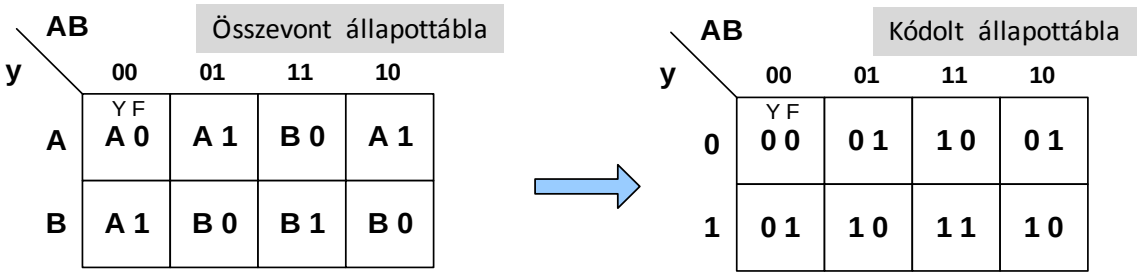
**Választott állapotkódok**: *a feladat szempontjából, valamint a hazárdjelenségek kiküszöbölése miatt egyáltalán nem közömbös, hogy miként rögzítjük az állapotkódokat.* Jelen példában, az egyszerűség kedvéért az állapotkódokat még direkt módon adtuk meg, azonban a későbbi 8. és 9. fejezetben részletesen ismertetjük az állapotkódolás módszereit.

A 2 összevont állapot → 1 szekunder ($y$) változó (két lehetséges értékkel)

- ***Legyen $y := 0$, ha $A$ állapot***
- ***Legyen $y := 1$, ha $B$ állapot***

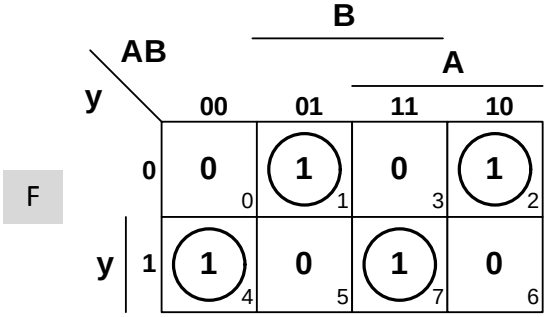
Megjegyzés: természetesen ez a kódolási séma csak az egyik lehetséges eset; lehetett volna akár fordítva is definiálni. Aszinkron esetben a kritikus versenyhelyzet (más néven rendszer hazárd) ismertetésénél tovább vizsgáljuk az állapotkódolást.

Kódolt állapottábla felírása a következő:




**4.6 ábra: Kódolt állapottábla felírása**

Ezután történhet a kimeneti függvény ($F$) Karnaugh táblájának felírása



**4.7 ábra: Az F kimeneti függvény Karnaugh táblájának felírása**

A Karnaugh tábla alapján az összeadó kimenetére kapott DNF alak a következő, amely kifejezhető szimmetrikus függvény segítségével is:

$$F = A \oplus B \oplus y \Rightarrow S^{n=3}{}_{1,3}(A, B, y)$$

Az $F$ függvény statikus, illetve dinamikus értelemben is hazárdmentes, míg a funkcionális hazárdot a nem szomszédos változások kiküszöbölésével kerülhetjük el.

Az $F$ kimeneti függvény fenti előállítása is igazolja a *Mealy-modellnél* korábban tanultakat: nevezetesen, hogy a kimeneti függvényt mind a bemenetek, mind pedig a visszacsatolt állapotérték(ek) együttesen, és direkt módon határozzák meg. Eml: $f_z(X, y) \Rightarrow Z$

## 5. Alkalmazandó tároló típusának kiválasztása

Jelen példában több lehetséges tanult megvalósító tároló típust is megvizsgálunk. Ez a lépés szorosan kapcsolódik a következő, 6. lépéshez. A vezérlési tábla alapján kapható úgynevezett „vezérlő" kombinációs hálózat az alkalmazandó tárolót fogja vezérelni. Itt több lehetséges típust is megvizsgálunk az építőelem készletből:

- J-K (R-S),
- T,
- D-G.

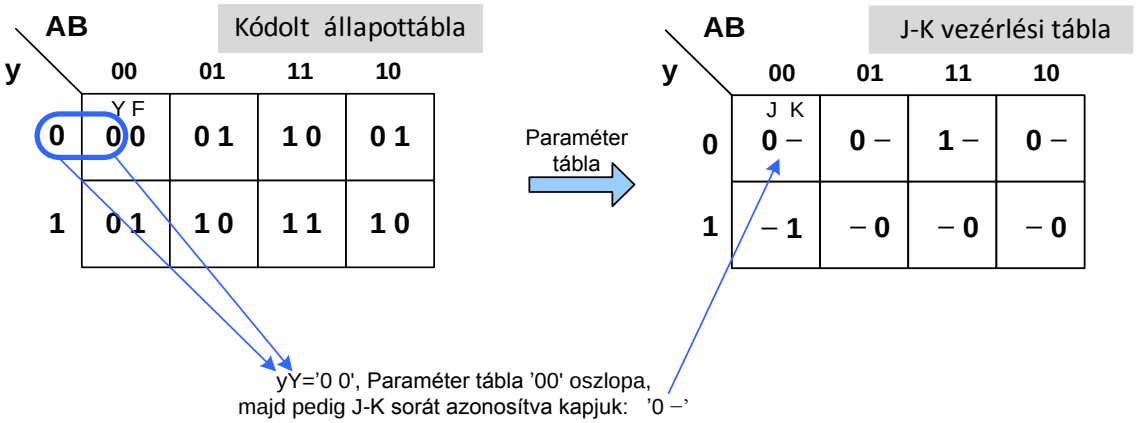
## 6. Vezérlési tábla összeállítása

Minden egyes szekunder ($y$) változóhoz hozzá kell rendelni egy megvalósító flip-flop-ot (FF): például a rendelkezésre álló építőelem készletből. A tárolók működésének ismeretében (*3. fejezet*) határozhatjuk meg, miként kell vezérelni őket egy K.H. segítségével a 4.lépésben felírt kódolt állapottábla alapján.

- Jelen esetben egyetlen egy megvalósító FF-t kell használni, mivel egy szekunder változó van. Így kapjuk meg a vezérlési táblát.
- Abban az esetben azonban, ha több szekunder ($y$) változónk is van, hogy akár különböző FF-kat rendelünk az építőelem készletből egy-egy szekunder változó (állapot) tárolásához.
- A vezérlési tábla felírásához a Paraméter táblázat használata szükséges (*lásd 3. fejezet*).

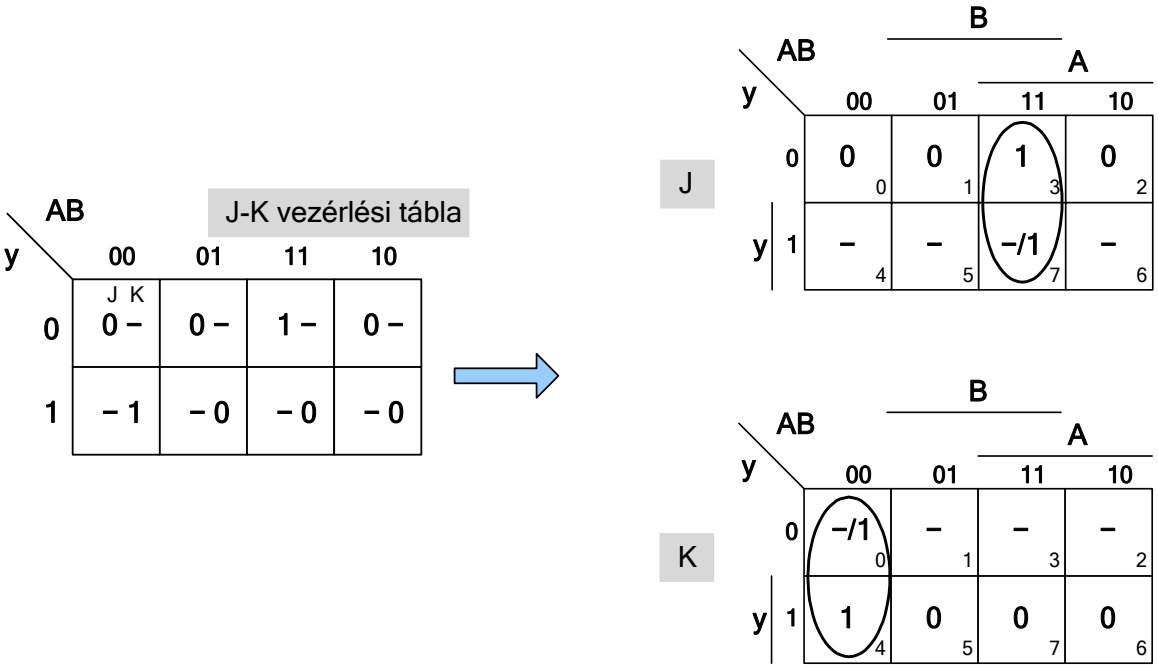
### a. Vezérlési tábla összeállítása J-K (S-R) tároló segítségével

Vezérlési tábla felírása (szekunder kombinációk vizsgálata):




**4.8 ábra: Vezérlési tábla felírása J-K tárolóra**

Vezérlési tábla alapján Karnaugh tábla felírása:




$$J = f_J(A, B, y) = A \cdot B$$

$$K = f_K(A, B, y) = \overline{A} \cdot \overline{B} = \overline{A + B}$$

$$F = A \oplus B \oplus y \Rightarrow S^{n=3}{}_{1,3}(A, B, y)$$

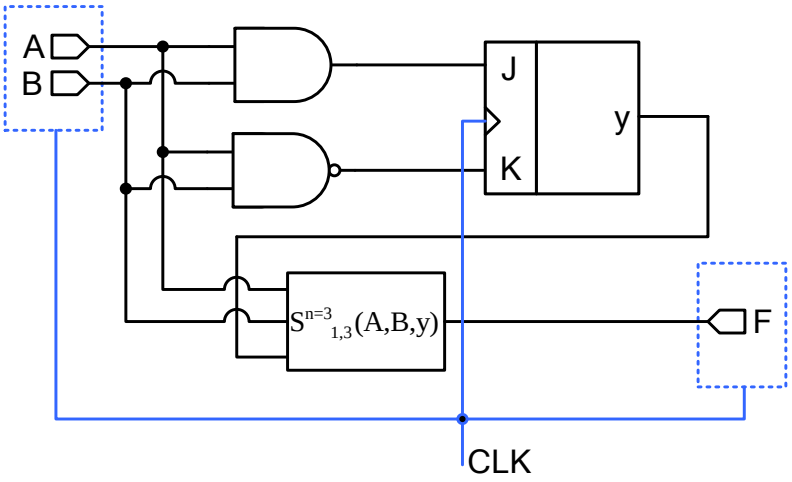
A vezérlő kombinációs hálózatok Karnaugh tábláiban $(J, K)$ arra kell törekedni a tömbösítés során, hogy egyrészt a létező legnagyobb lefedés(eke)t, másrészt ezek közül is a legkevesebb számú lefedés(eke)t vonjuk össze. Itt kell ügyelni arra is, hogy az összevonások során statikus-, illetve dinamikus-értelemben is hazárdmentes alakokat kapjunk (funkcionális hazárd kiküszöböléséről időzítéssel, és a nem szomszédos bemeneti változások tiltásával gondoskodunk). Azokban a cellákban, ahol direkt ’1’-es logikai érték van, az nyilvánvalóan nem maradhat ki a végső megvalósításból, ahol pedig don’t care ’-’ értékek szerepelnek, azokat szabadon választhatjuk meg.

**Megjegyzés**: az S-R tároló használatával történő FF megvalósítás esetén is ugyanezeket a vezérlő kombinációkat kaptuk volna (mivel S-R, illetve J-K értékek rendre, éppen olyan helyeken adódnak ’00’ és ’11’ -re, ahol a vezérlő K.H. a legegyszerűbb DNF alakját nem befolyásolják).

**Elvi kapcsolási rajz: szinkron soros összeadó J-K (vagy S-R) FF felhasználásával felépített elvi logikai rajza.**

A fenti képletek alapján a $K$ függvény esetén ÉS, illetve NEM-VAGY logikai kapukkal is megvalósíthatók, függően a rendelkezésre álló építő elemektől (kapuk az alkatrész készletünkben). Az $F$ kimeneti függvény esetén pedig akár szimmetrikus függvényt is használhatunk, amennyiben rendelkezésre áll.

Ezek alapján a soros összeadó J-K (S-R) tárolóval megvalósított kapcsolási rajza a következő:



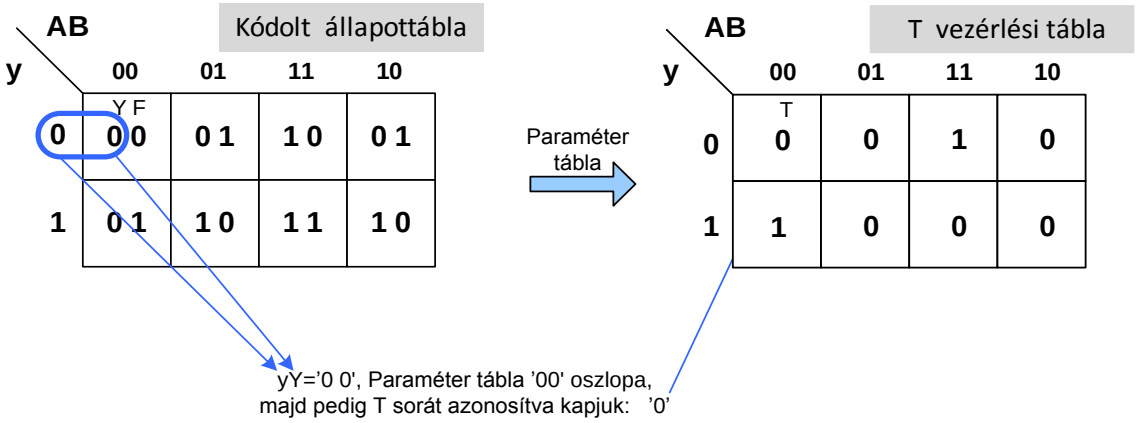

**4.9 ábra: Elvi logikai kapcsolási rajz J-K tárolóval**

Mealy: a tároló és a bemenetek együttesen határozzák meg a kimenetet ($F$).

A szinkronizációs feltételeket biztosítva, órajel (CLK) hatására változik mind a tárolt érték, mind a bemenetek ($A$, $B$) és a kimenet ($F$).

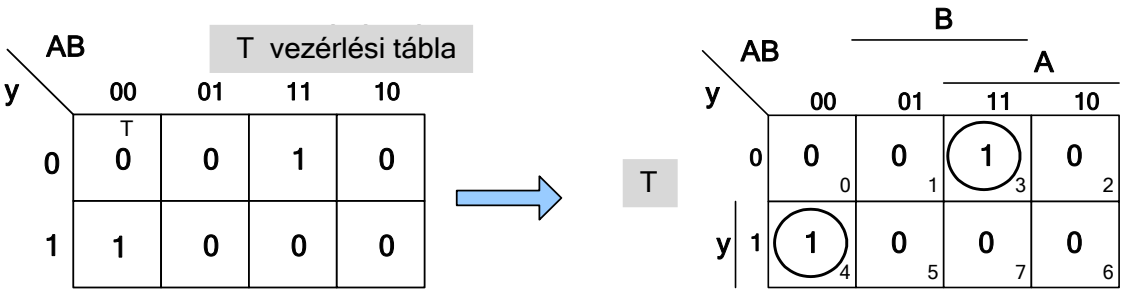
### b. Vezérlési tábla összeállítása T tároló segítségével

Vezérlési tábla felírása (szekunder kombinációk vizsgálata):




**4.10 ábra: Vezérlési tábla felírása T tárolóra**

Vezérlési tábla alapján Karnaugh tábla felírása:




$$T = \overline{AB}y + AB\overline{y}$$

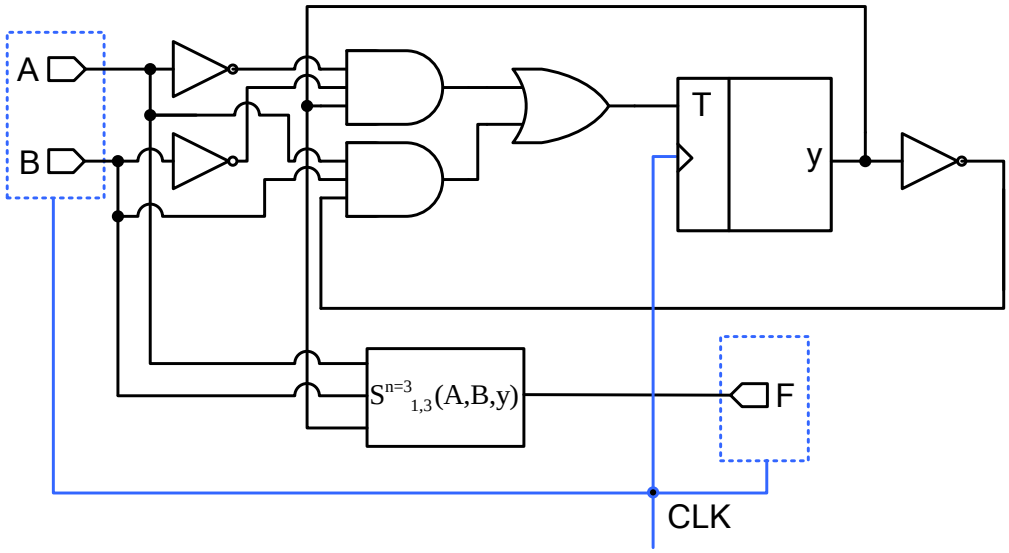
$$F = A \oplus B \oplus y \Rightarrow S^{n=3}{}_{1,3}(A, B, y)$$

A vezérlő kombinációs hálózatok Karnaugh táblájában (T) arra törekszünk, hogy a tömbösítés során, egyrészt a létező legnagyobb lefedés(eke)t, másrészt ezek közül is a legkevesebb számú lefedés(eke)t vonjuk össze. Azonban ebben az esetben, csak két önálló 1-elemű tömb van. *Ezért mondhatjuk azt, hogy a jelen példában a T tároló használata kevésbé lesz optimális, mint J-K, vagy akár S-R esetben (ezért is heurisztikus a módszer).* Az összevonások során statikus-, illetve dinamikus-értelemben is hazárdmentes alakokat nem kapunk (funkcionális hazárd kiküszöböléséről időzítéssel, és a nem szomszédos bemeneti változások tiltásával gondoskodunk). Azokban a cellákban, ahol direkt '1'-es logikai érték van, az nyilvánvalóan nem maradhat ki a végső megvalósításból.

**Elvi kapcsolási rajz: szinkron soros összeadó T - FF felhasználásával felépített elvi logikai rajza.**

A fenti képletek alapján a $K$ függvény esetén ÉS, illetve NEM-VAGY logikai kapukkal is megvalósíthatók, függően a rendelkezésre álló építő elemektől (kapuk az alkatrész készletünkben). Az $F$ kimeneti függvény esetén pedig akár szimmetrikus függvényt is használhatunk, amennyiben rendelkezésre áll.

Ezek alapján a soros összeadó T tárolóval megvalósított kapcsolási rajza a következő:



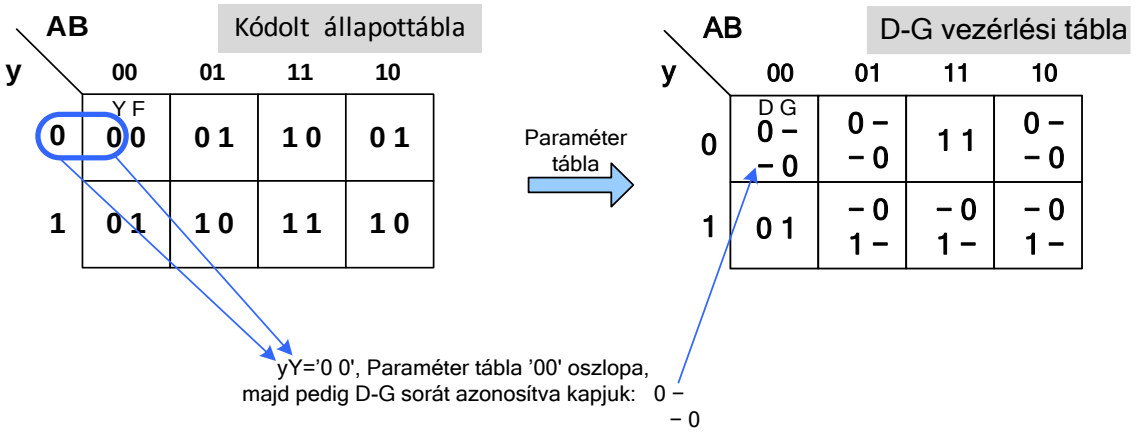

**4.11 ábra: Elvi logikai kapcsolási rajz T tárolóval**

Mealy: a tároló és a bemenetek együttesen határozzák meg a kimenetet ($F$).

A szinkronizációs feltételeket biztosítva, órajel (CLK) hatására változik mind a tárolt érték, mind a bemenetek ($A$, $B$) és a kimenet ($F$).

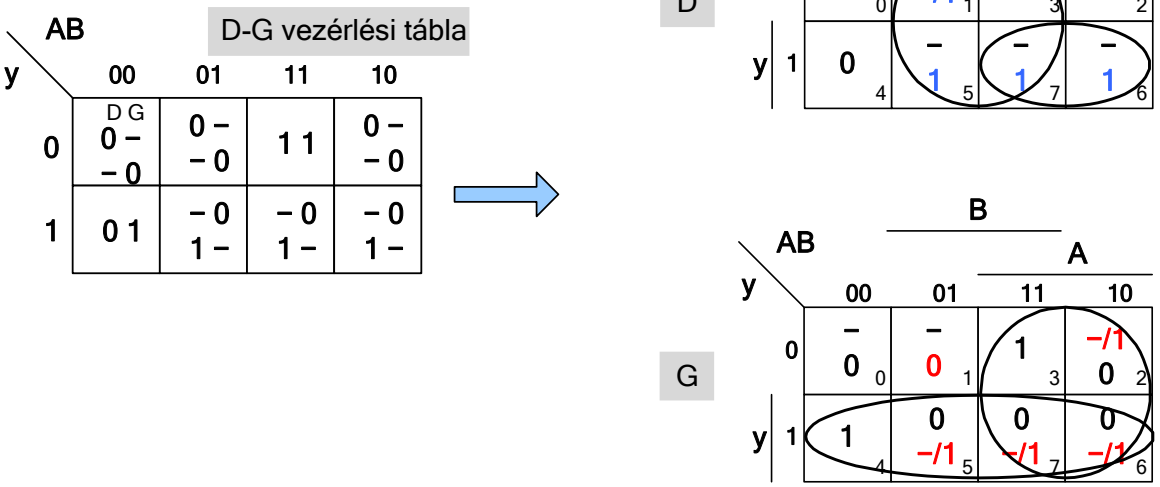
### c. Vezérlési tábla összeállítása D-G tároló segítségével

Vezérlési tábla felírása (szekunder kombinációk vizsgálata):




**4.12 ábra: Vezérlési tábla felírása D-G tárolóra**

Vezérlési tábla alapján Karnaugh tábla felírása:




$$D = Ay + B$$

$$G = A + y$$

$$F = A \oplus B \oplus y \Rightarrow F = S^{n=3}{}_{1,3}(A, B, y)$$

Fontos szabály, hogy a tanult módon egy-egy cellában rendre, vagy csak a felső, vagy csak az alsó logikai érték választható ki. A logikai érték pedig lehet fix '0', '1' vagy '-' don't care is: ez utóbbi esetben további lehetőségek vannak az „optimális kiválasztáshoz". Ezek alapján látható, hogy a nagy tervezői szabadság miatt heurisztikus módszerről beszélünk.

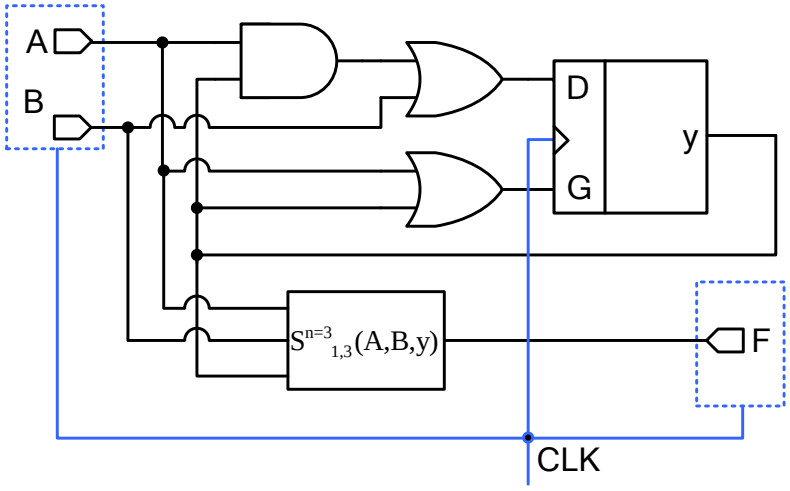
Ezeket a szabályokat figyelembe véve a vezérlő kombinációs hálózatok Karnaugh tábláiban (D, G) arra kell törekedni a tömbösítés során, hogy egyrészt a létező legnagyobb lefedés(eke)t, másrészt ezek közül is a legkevesebb számú lefedés(eke)t vonjuk össze.

A kimenetek előállításánál kell ügyelni arra is, hogy az összevonások során statikus-, illetve dinamikus-értelemben is hazárdmentes alakokat kapjunk (funkcionális hazárd kiküszöböléséről időzítéssel, és a nem szomszédos bemeneti változások tiltásával gondoskodunk).

**Elvi kapcsolási rajz: szinkron soros összeadó D-G FF felhasználásával felépített elvi logikai rajza.**

A fenti képletek alapján a $D$ függvény esetén egy ÉS, illetve egy VAGY logikai kapu, míg a $G$ függvény esetén egy VAGY kaput kell felhasználnunk az építőelem készletből. Az $F$ kimeneti függvény esetén pedig akár szimmetrikus függvényt is használhatunk, amennyiben rendelkezésre áll.

Ezek alapján a soros összeadó D-G tárolóval megvalósított kapcsolási rajza a következő:




**4.13 ábra: Elvi logikai kapcsolási rajz D-G tárolóval**

Mealy: a tároló és a bemenetek együttesen határozzák meg a kimenetet ($F$).

A szinkronizációs feltételeket biztosítva, órajel (CLK) hatására változik mind a tárolt érték, mind a bemenetek ($A$, $B$) és a kimenet ($F$).

## 7. Működtetés szemléltetése idődiagramon (Mealy)

Szinkronizációs feltételként a kimeneti értékek értelmezési tartományát meg kell adni. A valóságban a jelek, így az órajel is véges meredekséggel (felfutó-, lefutó élek) rendelkeznek.

A vizsgálathoz tekintsünk egy tetszőleges bemeneti kombinációváltozást, mely legyen a következő:

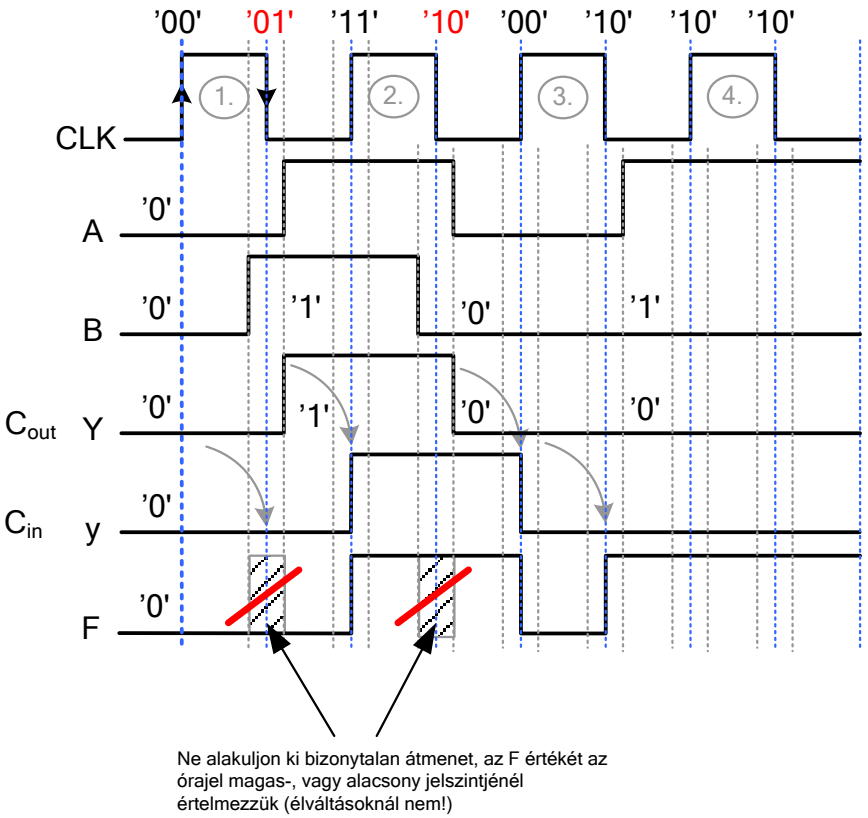
$$\boldsymbol{AB} \coloneqq \mathbf{00} \rightarrow \mathbf{11} \rightarrow \mathbf{00} \rightarrow \mathbf{10}$$

Azt feltételezzük, hogy az A-B bemeneti kombinációk az órajel lefutó éleire ( ), míg az $y$ szekunder állapot (aktuális) kombinációk az órajel felfutó éleire ( ) változnak meg.

A statikus-, dinamikus hazárd megszüntetéséről gondoskodtunk a vezérlő kombinációs hálózat tervezése során. A funkcionális hazárdot csak „szomszédos” bemeneti változások sorozataként tudjuk értelmezni: azaz átmeneti bemeneti kombinációk közbeiktatásával (például ’10’ v. ’01’). Ehhez azt kell még ismerni, hogy a lefutó élhez képest A, vagy B bemenet változzon korábban. Tekintsük ez utóbbi jelet gyorsabban változónak. Ekkor:

$$\boldsymbol{AB} \coloneqq \mathbf{00} \rightarrow \mathbf{01} \rightarrow \mathbf{11} \rightarrow \mathbf{10} \rightarrow \mathbf{00} \rightarrow \mathbf{10}$$

A szinkronizációs feltételek alapján a következő idődiagram rajzolható fel:




**4.14 ábra: Idődiagram szinkron Mealy-modell esetén**

## *Szinkron sorrendi hálózat (Moore-modell) komplex tervezési feladata:*

### 1. Logikai feladat megfogalmazása:

Legyen ugyanaz a feladat, mint szinkron Mealy-modell esetén: tervezzünk egy „soros" összeadót, de most Moore-modellt alkalmazva.

**Soros összeadó:**
A sorrendi hálózattal megvalósítható *soros összeadó* a teljes összeadóhoz hasonlóan az MSB ← LSB felé haladva egymás utáni lépésekben adja össze a bináris mintázatokat (mint bináris számjegyek). Azonban nem n db. 1-bites összeadó blokkot használunk fel a megépítéséhez, hanem egyetlen 1-bites összeadót, melynek átvitel vonalait, bemeneteit, kimeneteit megfelelő módon kell kezelni. A legfontosabb, hogy az átvitel kezelését (Carry) meg kell oldani az egyes helyiértékek között: ezért az átvitel kimenetet egy állapotként értelmezzük.

- $A(i)$ és $B(i)$ a két összeadandó bináris szám adott helyiértéken

### 2. Előzetes állapottábla összeállítása

A S.H.-nak különbözőképpen kell reagálnia ugyanarra a bemeneti kombinációra a megelőző bemeneti kombinációktól és a hálózat állapotától függően.

Kérdés azonban az, hogy minimálisan mennyi állapot kell a megvalósításhoz? Szinkron Moore modell esetén a soros összeadóra a következők érvényesek:

- „Előzetes" állapottáblában általában **több** állapot szerepel a szükségesnél.
- Kimenet pillanatnyi értéke: $F_{(i)}$

- Előző helyiértéken keletkezett átvitel (carry) értéke ($C_{in(i+1)} \leftarrow C_{out(i)}$) határozza meg a következő iterációban a carry bemeneti értékét

| Állapot jele ($y$) | Előző helyiértéken keletkezett átvitel értéke: $C_{in(i)} = C_{out(i-1)}$ | A kimenet pillanatnyi értéke: $F_{(i)}$ |
|---|---|---|
| a | 0 | 0 |
| b | 0 | 1 |
| c | 1 | 0 |
| d | 1 | 1 |

**4.2 táblázat: az állapotoknak megfelelő szekunder kombinációk megadása**

***Fontos megjegyzés #1:*** *soros összegzéskor a számított kimeneti érték független a szekunder kombinációkhoz tartozó* ***a előző/korábbi kimeneti értéktől****, csak a bemeneti kombinációktól, illetve az előző helyiértéken keletkezett átviteltől függ.*

***Fontos megjegyzés #2:*** *az A-B bemeneti kombináció hatása* ***csak azután fog érvényesülni*** *az $F$ kimeneten, miután az általa létrehozott Y visszahat y-ként a bemenetre! (Moore féle szinkronizálás).*

Következmény: **Lényeges eltérés Mealy-Moore modellek működése között**, hogy

- *Mealy* esetén mindig az **új kimeneti** értéket kell bejegyezni **a következő állapot mellé**, míg
- *Moore* esetén **a kiindulási kimeneti** értéket kell bejegyezni a **következő állapot mellé**!

Előzetes állapottábla összeállítása: az állapottáblának összesen négy oszlopa (A,B bemeneti kombinációk miatt), és négy sora (a,b,c,d – szekunder kombinációk száma miatt) lesz. Próbáljunk a kitöltés során ügyelni arra, hogy az oszlopok és sorok sorrendje lehetőleg kövesse a Karnaugh táblák peremezését, mivel ez később segíthet pl. a kimeneti függvény könnyebb leolvasásában. Kitöltés menete a következő:

**Kitöltés*: (1. sor)***

- Tfh. $\boldsymbol{y = a}$ és $\boldsymbol{AB = 00} \rightarrow F = A + B + C_{in} = 0$, és $C_{out} = 0 \rightarrow$ Ekkor $Y = a$ állapot és $F = 0$ kimenetre: $\boldsymbol{a0}$
- Tfh. $\boldsymbol{y = a}$ és $\boldsymbol{AB = 01}$ v $\boldsymbol{AB = 10} \rightarrow F = A + B + C_{in} = 1$ és $C_{out} = 0 \rightarrow$ Ekkor $Y = b$ állapot és **korábbi** $F = 0$ kimenetre: $\boldsymbol{b0}$
- Tfh. $\boldsymbol{y = a}$ és $\boldsymbol{AB = 11} \rightarrow F = A + B + C_{in} = 10$ és $C_{out} = 1 \rightarrow$ Ekkor $Y = c$ állapot és $F = 0$ kimenetre: $\boldsymbol{c0}$

**Kitöltés*: (3. sor)***

- Tfh. $\boldsymbol{y = d}$ és $\boldsymbol{AB = 00} \rightarrow F = A + B + C_{in} = 1$, és $C_{out} = 0 \rightarrow$ Ekkor $b$ állapot és $F = 1$ kimenetre: $\boldsymbol{b1}$
- Tfh. $\boldsymbol{y = d}$ és $\boldsymbol{AB = 01}$ v $\boldsymbol{AB = 10} \rightarrow F = A + B + C_{in} = 10$ és $C_{out} = 1 \rightarrow$ Ekkor $c$ állapot és a **korábbi** $F = 0$ kimenetre: $\boldsymbol{c1}$
- Tfh. $\boldsymbol{y = d}$ és $\boldsymbol{AB = 11} \rightarrow F = A + B + C_{in} = 11$ és $C_{out} = 1 \rightarrow$ Ekkor $d$ állapot és $F = 1$ kimenetre: $\boldsymbol{d1}$

*Moore* esetén **a kiindulási kimeneti** értéket kell bejegyezni a **következő állapot mellé**

Előzetes állapottábla

| y | AB: 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 a | Y F<br>a 0 | b 0 | c 0 | b 0 |
| 01 b | a 1 | b 1 | c 1 | b 1 |
| 11 d | b 1 | c 1 | d 1 | c 1 |
| 10 c | b 0 | c 0 | d 0 | c 0 |

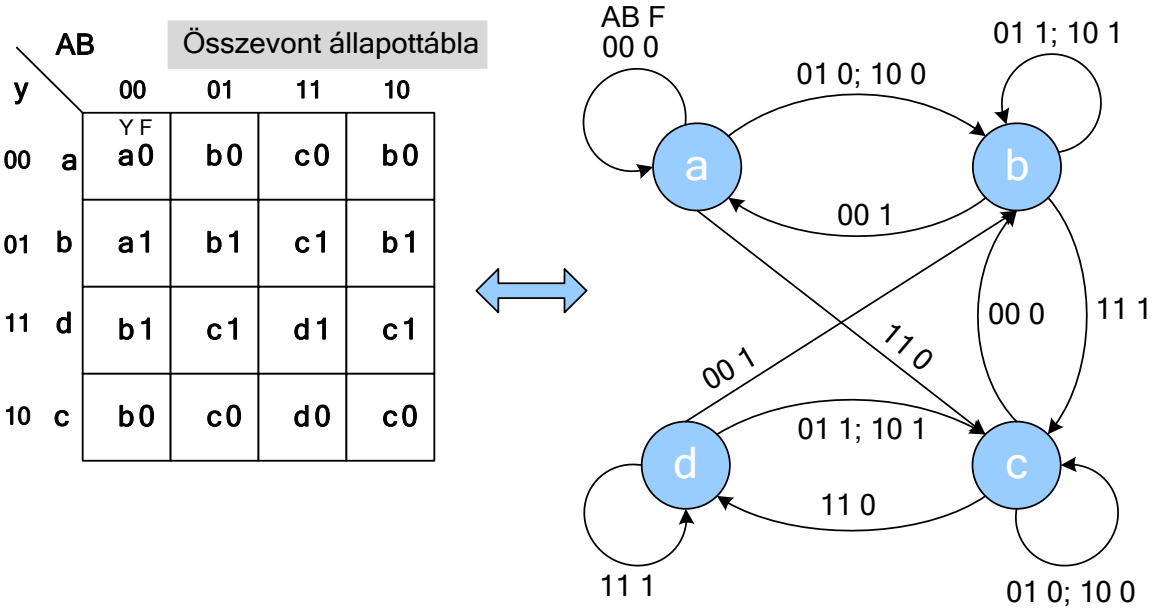
Ha megvizsgáljuk a fenti előzetes állapottábla kimeneteit ($F$) tartalmazó oszlopokat, akkor az látható, hogy az egy szekunder kombinációhoz tartozó kimeneti értékek egy-egy soron belül **azonosak**, emiatt az $A$-$B$ bemenetektől direkt módon nem fog függeni az $F$ kimenet (tömbösítés során a bemeneti változók kiesnek). Ez is a *Moore modell* szerinti működést igazolja.

### 3. Összevont (egyszerűsített) állapottábla

Előzetes állapottábla alapján az a cél, hogy megkeressük a lehető legkevesebb megkülönböztetendő állapotot. Ezek ismeretében kell létrehozni a lehető legkevesebb, de azonos sort nem tartalmazó új állapottáblát, amelyet összevont állapottáblának nevezünk, amelynek így már nem lesz feleslegesen megkülönböztetett állapota (=sora). Fontos tudni, hogy az összevont állapottábla nem azonos a később ismertetésre kerülő állapot kódolási módszerrel!

- Moore esetén a kimeneti ($F$) értéket kizárólag a mindenkori állapot határozza meg ($y$)!
- Az állapotokat tehát a kimenet értéke szerint is meg kell különböztetnünk, emiatt a fenti előzetes állapottáblában nem tudunk sorokat összevonni: *nincsenek feleslegesen megkülönböztetett állapotok*. Így az előzetes állapottábla, egyben összevont állapottábla is.

Az összevonás során kapott összevont állapottábla és a vele ekvivalens állapotgráf a következő:




**4.15 ábra: Összevont állapottábla és állapotgráf**

### 4. Kódolt állapottábla

Az összevont állapottábla alapján az {a,b,c,d} jelölt állapotoknak meg kell feleltetni egy-egy szekunder kombinációt ($y$).

Az összevont állapottábla sorainak (=állapotainak) számától függ, hogy mennyi különböző szekunder állapotot kell felvenni:

$n$ sor esetén: $\lceil \log_2(n) \rceil$ szekunder állapot szükséges.

**Választott állapotkódok**: *a feladat szempontjából, valamint a hazárdjelenségek kiküszöbölése miatt egyáltalán nem közömbös, hogy miként rögzítjük az állapotkódokat.* Jelen példában, az egyszerűség kedvéért az állapotkódokat direkt módon adtuk meg, azonban a későbbi *8. és 9. fejezetekben* részletesen ismertetjük az állapotkódolás módszereit.

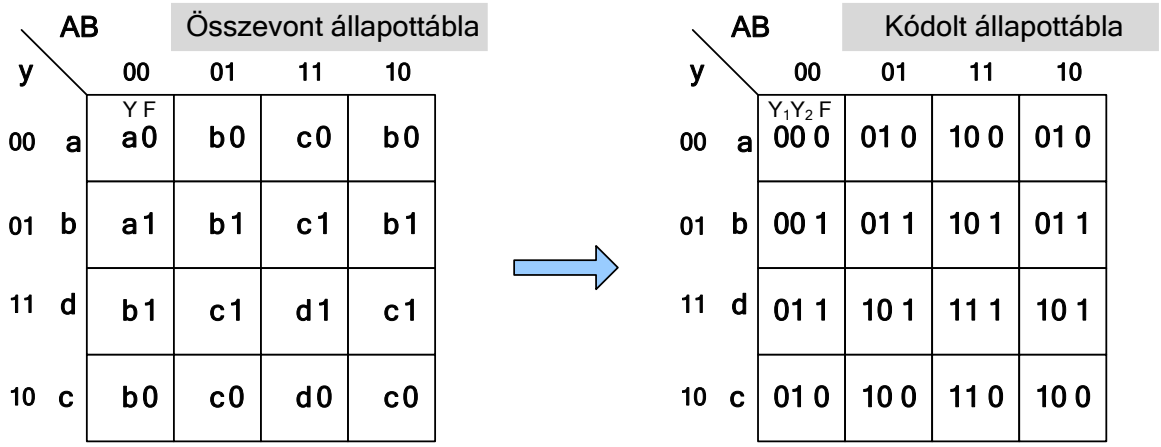
Összevont állapottábla 4 állapota → 2 szekunder ($y_1$ és $y_2$) változó (két-két lehetséges értékkel) kódolt értékekkel:

| összevont állapot | $y_1$ | $y_2$ |
|---|---|---|
| a | 0 | 0 |
| b | 0 | 1 |

| | | |
|---|---|---|
| c | 1 | 0 |
| d | 1 | 1 |

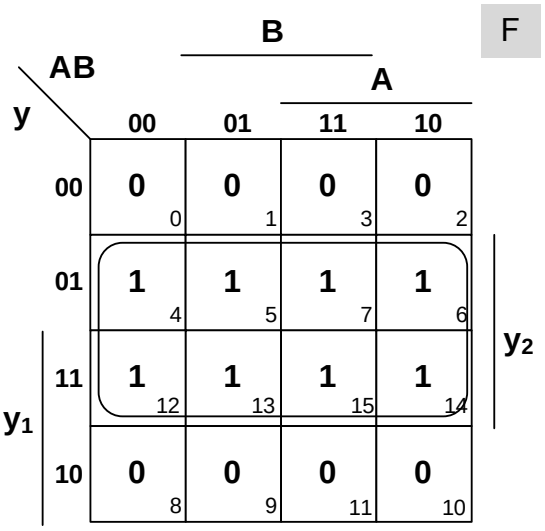
Megjegyzés: természetesen ez a kódolási séma csak az egyik lehetséges eset; lehetett volna akár további eseteket is tetszőlegesen definiálni. Aszinkron esetben a kritikus versenyhelyzet (más néven rendszer hazárd) ismertetésénél tovább vizsgáljuk az állapotkódolást.

Kódolt állapottábla felírása a következő:




**4.16 ábra: Kódolt állapottábla felírása**

Ezután történhet a kimeneti függvény (**F**) Karnaugh táblájának felírása




**4.17 ábra: Az F kimeneti függvény Karnaugh táblájának felírása**

A Karnaugh tábla alapján az összeadó kimenetére kapott DNF alak a következő, amely kifejezhető szimmetrikus függvény segítségével is:

$$F = y_2$$

Az $F$ függvény statikus, illetve dinamikus értelemben is hazárdmentes, míg a funkcionális hazárdot a nem szomszédos változások kiküszöbölésével kerülhetjük el.

Az $F$ kimeneti függvény fenti előállítása is igazolja a *Moore-modellnél* korábban tanultakat: nevezetesen, hogy a kimeneti függvényt direkt módon csak az állapotérték(ek) határozzák meg. Eml:

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

### 5. Alkalmazandó tároló típusának kiválasztása

Jelen példában több lehetséges tanult megvalósító tároló típust is megvizsgálunk. Ez a lépés szorosan kapcsolódik a következő, 6. lépéshez. A vezérlési tábla alapján kapható úgynevezett „vezérlő” kombinációs hálózat az alkalmazandó tárolót fogja vezérelni. Itt több lehetséges típust is megvizsgálunk az építőelem készletből:

- D,
- S-R (J-K).

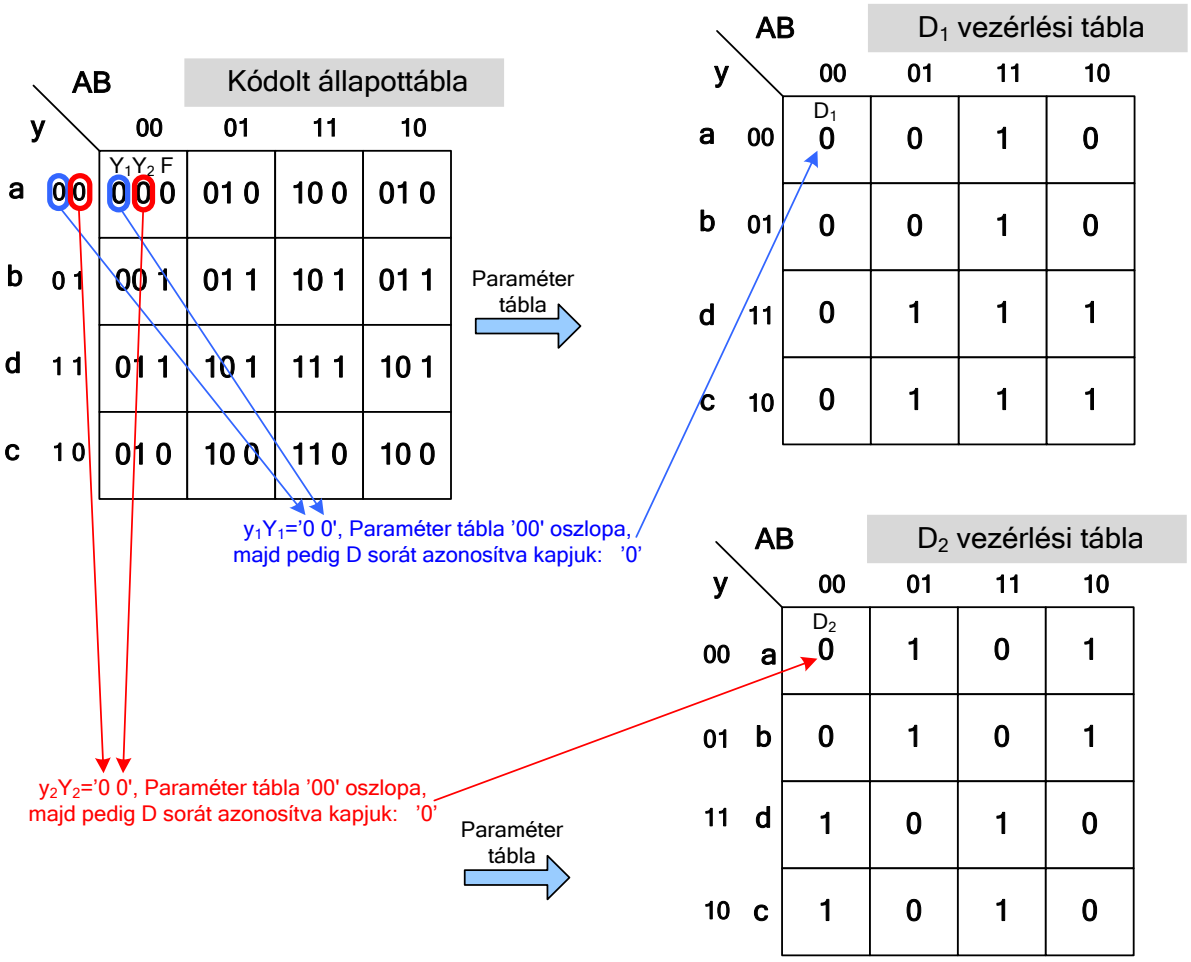
### 6. Vezérlési tábla összeállítása

Minden egyes szekunder ($y$) változóhoz hozzá kell rendelni egy megvalósító Flip-flop-ot (FF): pl. a rendelkezésre álló építőelem készletből. A tárolók működésének ismeretében (*3. fejezet*) határozhatjuk meg, miként kell vezérelni őket egy K.H. segítségével a 4.lépésben felírt kódolt állapottábla alapján.

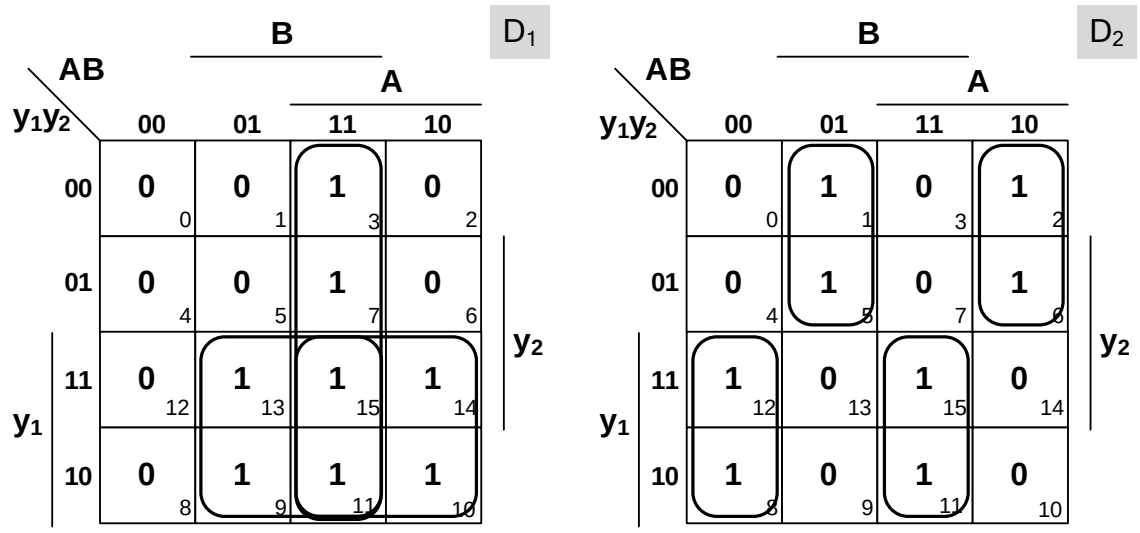
- Jelen esetben 2 megvalósító FF-t kell használni, mivel két szekunder változó van: $y_1$, $y_2$. Így kapjuk meg a vezérlési táblát.
- Megjegyzés: abban az esetben, ha több szekunder ($y$) változónk is van, akár különböző FF-kat rendelünk az építőelem készletből egy-egy szekunder változó (állapot) tárolásához. Jelen példában mindkét változóhoz csak egyfajta tárolót rendelünk.
- A vezérlési tábla felírásához a Paraméter táblázat használata szükséges (*lásd 3. fejezet*).

#### a. Vezérlési tábla összeállítása D tároló segítségével

Vezérlési tábla felírása (szekunder kombinációk vizsgálata):



Vezérlési táblák alapján a Karnaugh táblák felírása külön $D_1$ illetve $D_2$-re :




$$D_1 = AB + Ay_1 + By_1$$

$$D_2 = \overline{AB}y_1 + \overline{A}B\overline{y_1} + A\overline{B}\overline{y_1} + ABy_1 \Rightarrow S^{n=3}{}_{1,3}(A, B, y_1)$$

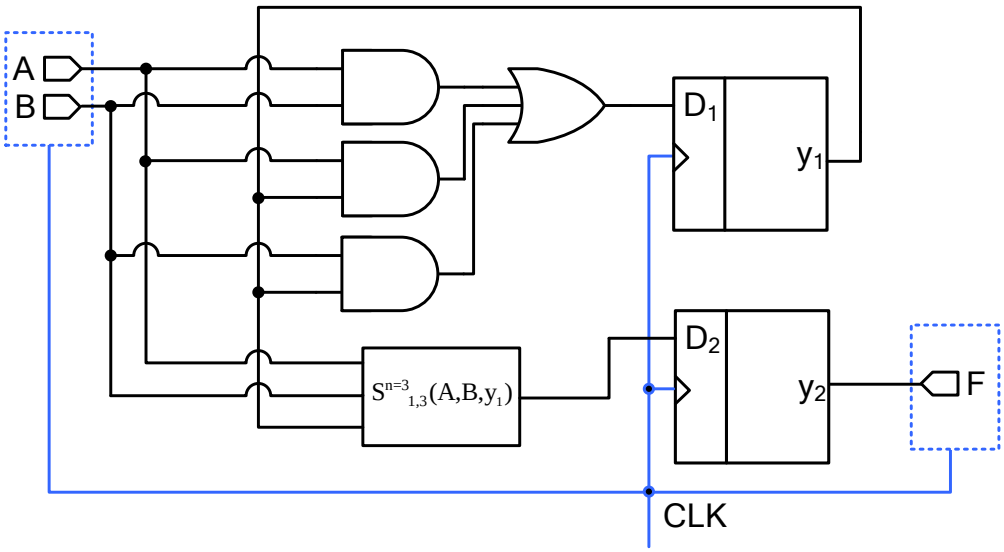
$$F = y_2$$

A vezérlő kombinációs hálózatok Karnaugh tábláiban ($D_1$, $D_2$) arra törekszünk, hogy a tömbösítés során egyrészt a létező legnagyobb lefedés(eke)t, másrészt ezek közül is a legkevesebb számú lefedés(eke)t vonjuk össze. Az összevonások során statikus-, illetve dinamikus-értelemben is hazárdmentes alakokat kapunk (funkcionális hazárd kiküszöböléséről időzítéssel, és a nem szomszédos bemeneti változások tiltásával gondoskodunk).

**Elvi kapcsolási rajz: szinkron soros összeadó D- FF felhasználásával felépített elvi logikai rajza.**

A fenti képletek alapján a $D_1$ függvény esetén 3 db ÉS, illetve 1 db VAGY logikai kapuval valósítható meg, függően a rendelkezésre álló építő elemektől (kapuk az alkatrész készletünkben). A $D_2$ kimeneti függvény azonban sokkal összetettebb, de ez esetben akár szimmetrikus függvényt is használhatunk, amennyiben az rendelkezésre áll. Az $F$ kimeneti függvény pedig csak az $y_2$ változó értékétől függ (független a bemenetektől, illetve az $y_1$-től is).

Ezek alapján a soros összeadó D tárolóval megvalósított kapcsolási rajza a következő:



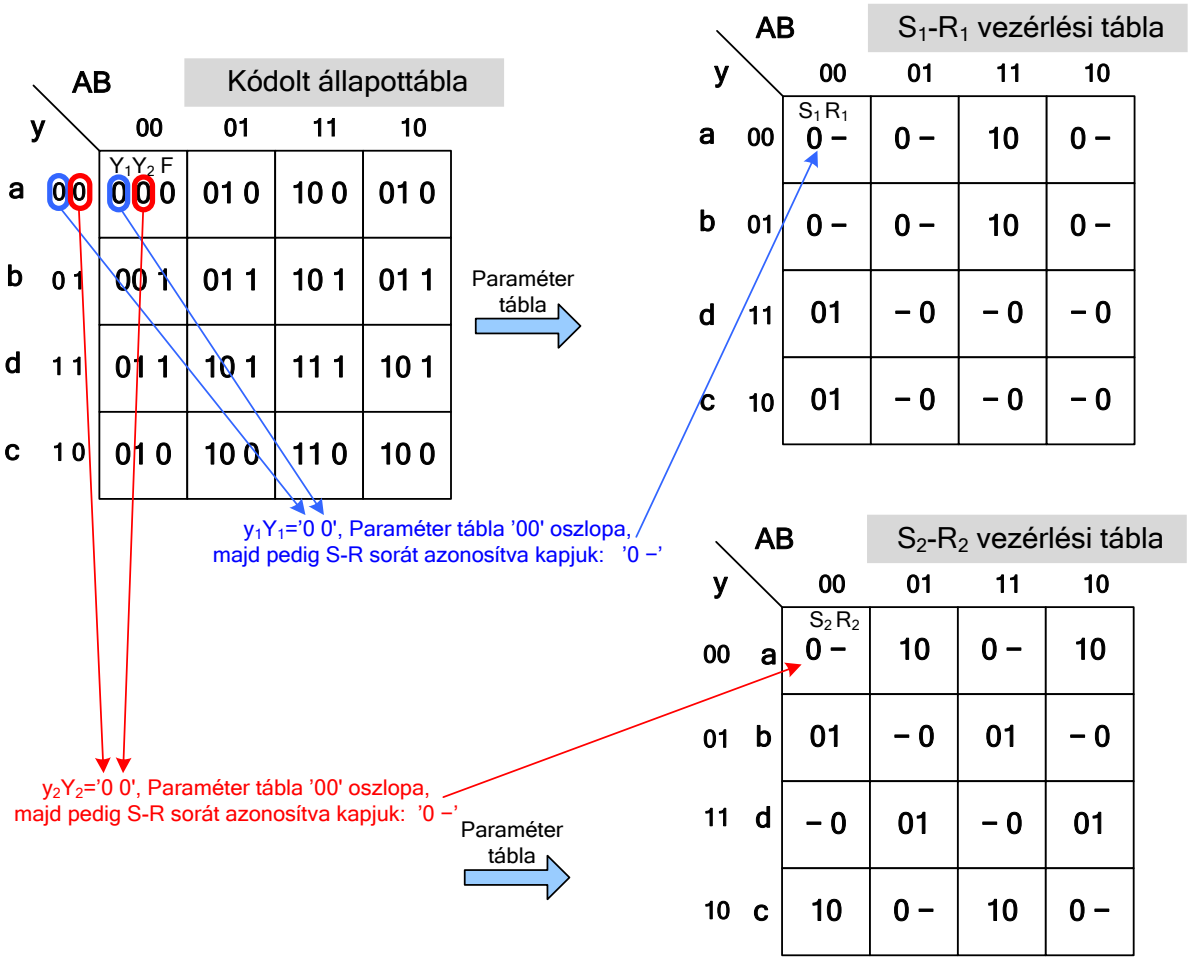

**4.18 ábra: Elvi logikai kapcsolási rajz D tárolókkal**

Moore: az alsó $D_2$ tároló aktuális állapot értéke ($y_2$) direkt módon határozza meg a kimenetet ($F$).

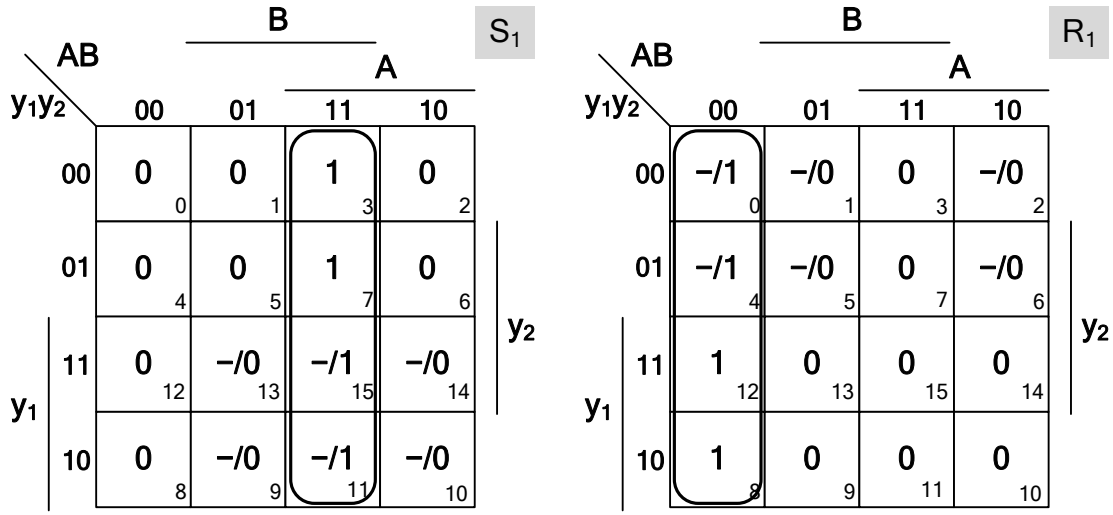
A szinkronizációs feltételeket biztosítva, órajel (CLK) hatására változik mind a tárolt érték, mind a bemenetek ($A$, $B$) és a kimenet ($F$).

### b. Vezérlési tábla összeállítása S-R (vagy J-K) tároló segítségével

Vezérlési tábla felírása (szekunder kombinációk vizsgálata):

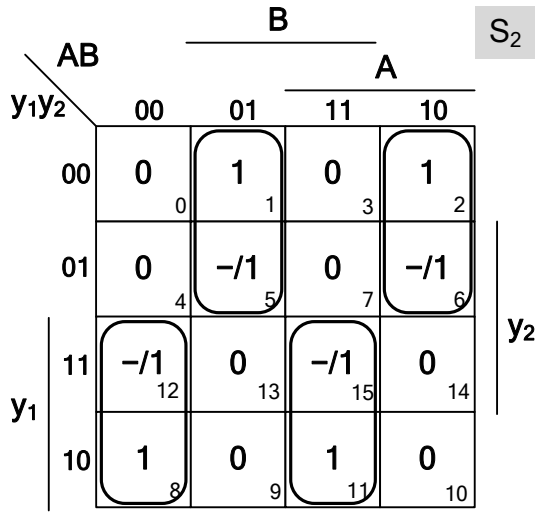


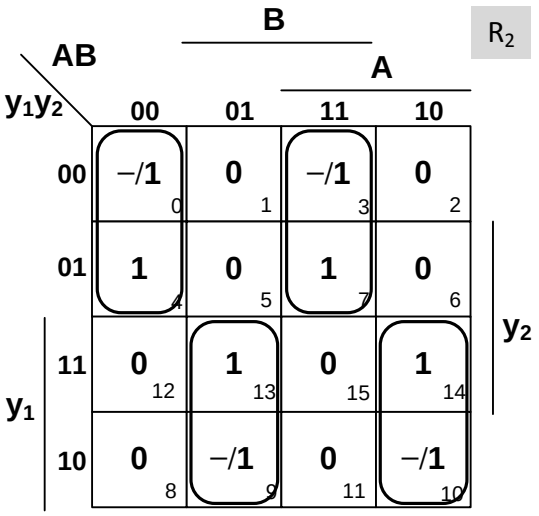

Vezérlési táblák alapján a Karnaugh táblák felírása külön $S_1$ $R_1$-re, illetve $S_2$ $R_2$-re :




$$S_1 = AB$$

$$R_1 = \overline{A} \cdot \overline{B} = \overline{A + B}$$




$$S_2 = \overline{AB}y_1 + \overline{A}B\overline{y_1} + A\overline{B}\overline{y_1} + ABy_1 = A \oplus B \oplus y_1 \Rightarrow {S^{n=3}}_{1,3}(A, B, y_1)$$

$$R_2 = \overline{S_2} \Rightarrow \overline{{S^{n=3}}_{1,3}(A, B, y_1)} = {S^{n=3}}_{0,2}(A, B, y_1)$$

A kimeneti $F$ függvényt már korábban megadtuk, és tudjuk, hogy:

$$F = y_2$$

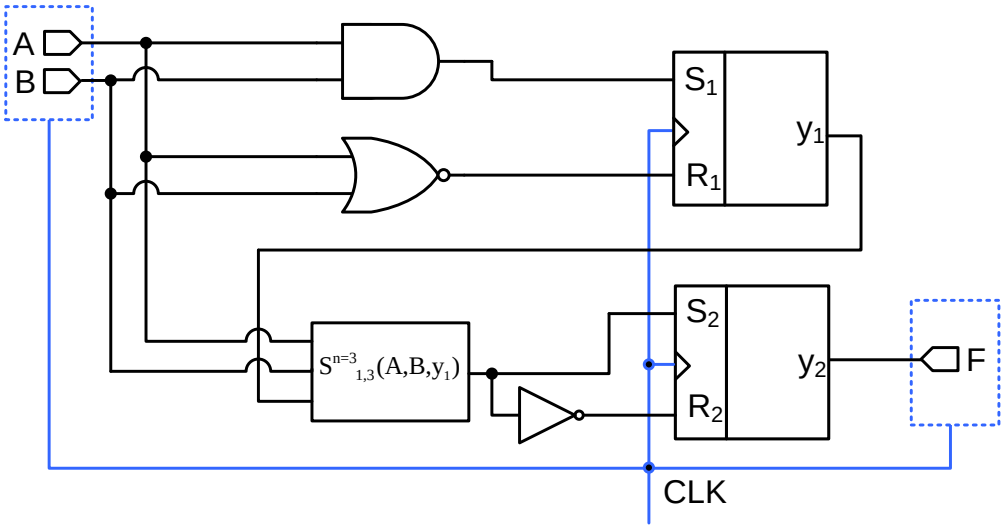
A vezérlő kombinációs hálózatok Karnaugh tábláiban ($S_1$-$R_1$ illetve $S_2$-$R_2$) arra törekszünk, hogy a tömbösítés során, egyrészt a létező legnagyobb lefedés(eke)t, másrészt ezek közül is a legkevesebb számú lefedés(eke)t vonjuk össze. Az összevonások során statikus-, illetve dinamikus-értelemben is hazárdmentes alakokat kapunk (funkcionális hazárd kiküszöböléséről időzítéssel, és a nem szomszédos bemeneti változások tiltásával gondoskodunk).

**Megjegyzés**: az J-K tároló használatával történő FF megvalósítás esetén is ugyanezeket a vezérlő kombinációkat kaptuk volna (mivel S-R, illetve J-K értékek rendre, éppen olyan helyeken adódnak '00' és '11' -re, ahol a vezérlő K.H. a legegyszerűbb DNF alakját nem befolyásolják).

**Elvi kapcsolási rajz: szinkron soros összeadó S-R (vagy J-K) FF felhasználásával felépített elvi logikai rajza.**

A fenti képletek alapján a $S_1$-$R_1$ függvények esetén 1 db ÉS, illetve 1 db NEM-VAGY logikai kapuval valósítható meg, függően a rendelkezésre álló építő elemektől (kapuk az alkatrész készletünkben). Az $S_2$ függvény azonban sokkal összetettebb, de ez esetben akár szimmetrikus függvényt is használhatunk, amennyiben az rendelkezésre áll. Ha észrevesszük, akkor pedig az $R_2$ kifejezhető az $S_2$ függvény negálásával, illetve annak megfelelős szimmetrikus függvény negáltjával. Az $F$ kimeneti függvény pedig csak az $y_2$ változó értékétől függ (független a bemenetektől, illetve az $y_1$-től is).

Ezek alapján a soros összeadó R-S tárolóval megvalósított kapcsolási rajza a következő:




**4.19 ábra: Elvi logikai kapcsolási rajz S-R tárolókkal**

Moore: az alsó $S_2$-$R_2$ tároló aktuális állapot értéke ($y_2$) direkt módon határozza meg a kimenetet ($F$).

A szinkronizációs feltételeket biztosítva, órajel (CLK) hatására változik mind a tárolt érték, mind a bemenetek ($A$, $B$) és a kimenet ($F$).

## 7. Működtetés szemléltetése idődiagramon (Moore)

Szinkronizációs feltételként a kimeneti értékek értelmezési tartományát meg kell adni. A valóságban a jelek, így az órajel is véges meredekséggel (felfutó-, lefutó élek) rendelkeznek.

A vizsgálathoz tekintsünk egy tetszőleges bemeneti kombinációváltozást, mely legyen a következő:

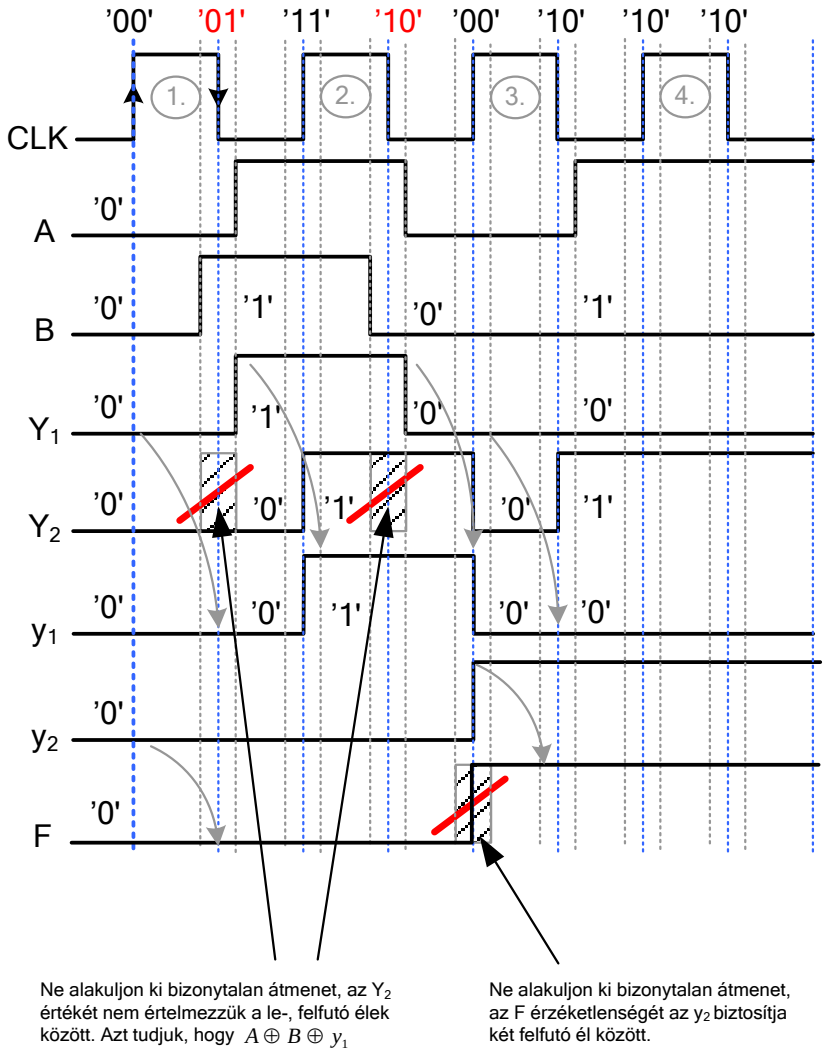
$$\boldsymbol{AB} \coloneqq \mathbf{00} \rightarrow \mathbf{11} \rightarrow \mathbf{00} \rightarrow \mathbf{10}$$

Feltételezzük, hogy az A-B bemeneti kombinációk az órajel lefutó éleire ( ), míg az $y_1$, illetve $y_2$ szekunder állapot (aktuális) kombinációk az órajel felfutó éleire ( ) változnak meg.

A statikus-, dinamikus hazárd megszüntetéséről gondoskodtunk a vezérlő kombinációs hálózat tervezése során. A funkcionális hazárdot csak „szomszédos” bemeneti változások sorozataként tudjuk értelmezni: azaz átmeneti bemeneti kombinációk közbeiktatásával (például ’10’ v. ’01’). Ehhez azt kell még ismerni, hogy a lefutó élhez képest $A$, vagy $B$ bemenet változzon korábban. Tekintsük ez utóbbi jelet gyorsabban változónak. Ekkor:

$$\boldsymbol{AB} \coloneqq \mathbf{00} \rightarrow \mathbf{01} \rightarrow \mathbf{11} \rightarrow \mathbf{10} \rightarrow \mathbf{00} \rightarrow \mathbf{10}$$

A szinkronizációs feltételek alapján a következő idődiagram rajzolható fel:




**4.20 ábra: Idődiagram szinkron Moore-modell esetén**

## *További példák*

### 4.1 Példa: Sorrendi hálózat tervezése TS állapotgráf alapján

Adott a következő Teljesen Specifikált Állapotgráf.

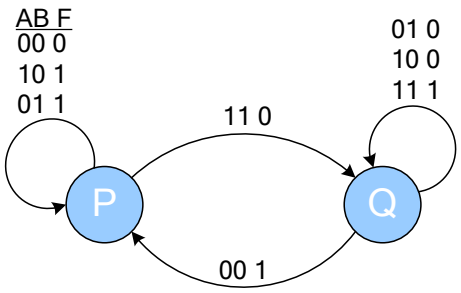
Kérdés:

a.) Tervezzen szinkron sorrendi hálózatot az alábbi állapotgráf és a megadott állapotkódolás alapján szinkron *D-G-tároló* felhasználásával!

b.) Adja meg, hogy mely modell szerint működik a hálózat!

c.) Adja meg a *kódolt állapottáblát*, *vezérlési táblát*, valamint ezek alapján a függvények *Karnaugh* tábláit, elvi *kapcsolási rajzát*.

**Állapotráf:**




**Állapotkódok:**

$P := 0$

$Q := 1$

(Mennyi szekunder változó szükséges?)

Az összevont állapottábla sorainak (=állapotainak) számától függ, hogy mennyi különböző szekunder állapotot kell felvenni:

2 sor esetén: $\lceil \log_2(2) \rceil = 1$, azaz egyetlen szekunder állapot lesz szükséges.

**Megoldás:**

a.) Előzetes állapottábla = összevont állapottábla (nincs megkülönböztethető sora), amelyből a fenti állapotkódok $\{P, Q\}$ behelyettesítésével kapjuk a kódolt állapottáblát.

Összevont állapottábla

| y \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 P | P 0 | P 1 | Q 0 | P 1 |
| 1 Q | P 1 | Q 0 | Q 1 | Q 0 |

Kódolt állapottábla

| y \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 0 | 0 1 | 1 0 | 0 1 |
| 1 | 0 1 | 1 0 | 1 1 | 1 0 |

b.) Az $F$ kimeneti függvény előállítása a következő:

F kimeneti függvény

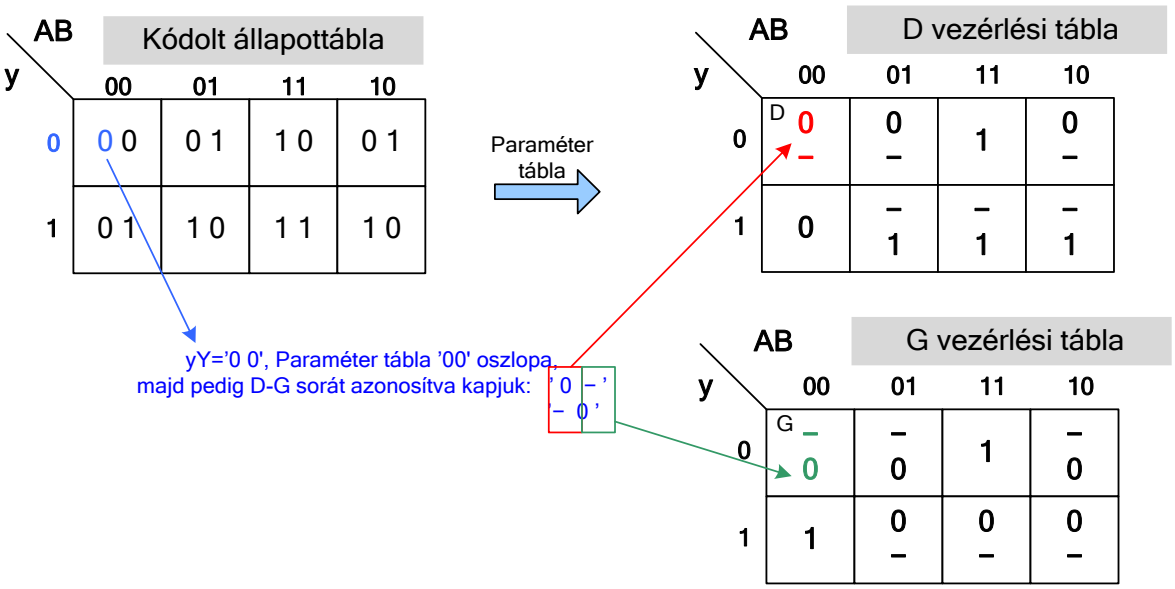
| y \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 (0) | 1 (1) | 0 (3) | 1 (2) |
| 1 | 1 (4) | 0 (5) | 1 (7) | 0 (6) |

$$F = A\overline{B}\overline{y} + \overline{A}B\overline{y} + \overline{A}\overline{B}y + ABy = S_{1,3}^{n=3}(A, B, y)$$

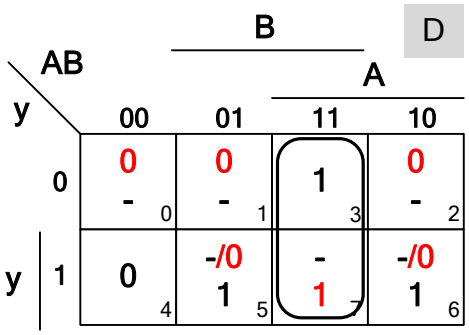
Az $F$ kimeneti függvény esetén további hazárdmentesítő (statikus, dinamikus) hurkokat nem kellet megadni. Látható az is, hogy az $F$ kimeneti függvény megvalósítható $S$ szimmetrikus függvény segítségével is (feltételezve azt, hogy építőelem készletünkben van).

Mivel az $F$ kimeneti függvény direkt módon függ a bemeneti kombinációktól is ($A$, $B$), ezért csak Mealy modellként valósítható meg. Másrészt, mivel a kódolt állapottáblán, illetve az $F$ Karnaugh tábláján is soronként eltérő kimeneteket kapunk, szintén a Mealy modell szerinti működés igazolódik.

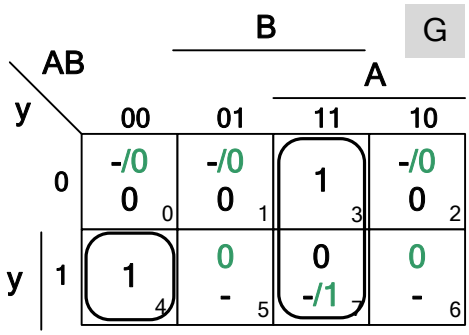
c.) Kódolt állapottábla alapján a Paraméter táblázat szinkron D-G-tároló sorából kapott (D, illetve G) vezérlési táblák a következők (könnyebb leolvasás miatt D, ill. G vezérlési táblák külön kerültek felírásra):




A vezérlési táblákból (D, illetve G) a tanult szabályok szerint kapott Karnaugh táblák (egy lehetséges esete) a következők lesznek:



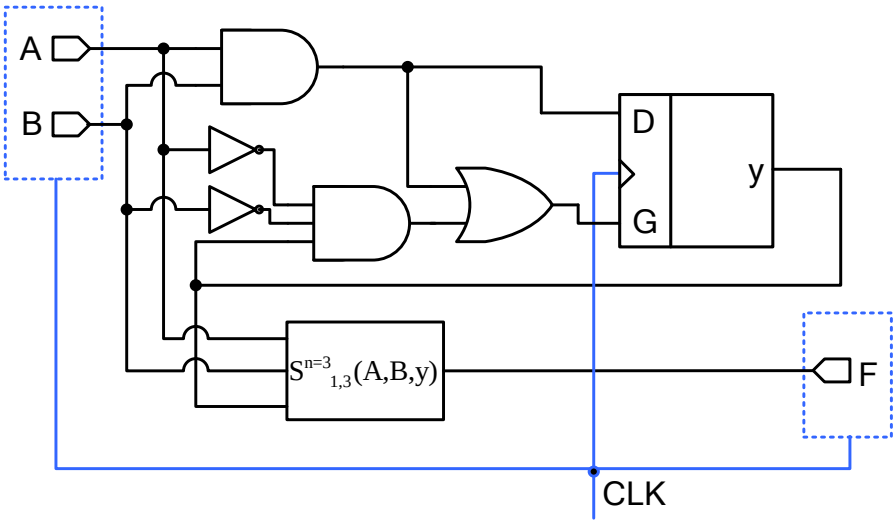

$D = AB$




$G = AB + \overline{AB}y$

A fenti két Karnaugh tábla alapján leolvasott DNF alakban van egy közös prímimplikáns ($AB$), így azt elegendő egyszer megvalósítani. Hazárdmentesítést (statikus, dinamikus) nem kellett végezni.

**Elvi kapcsolási rajz: D-G FF felhasználásával felépített elvi logikai rajza**




Mealy: a tároló és a bemenetek együttesen határozzák meg a kimenetet ($F$).

A szinkronizációs feltételeket biztosítva, órajel (CLK) hatására változik mind a tárolt érték, mind a bemenetek ($A$, $B$) és a kimenet ($F$).

### 4.2 Példa: Sorrendi hálózat tervezése NTS állapotgráf alapján

Adott a következő Nem-Teljesen Specifikált Állapotgráf.

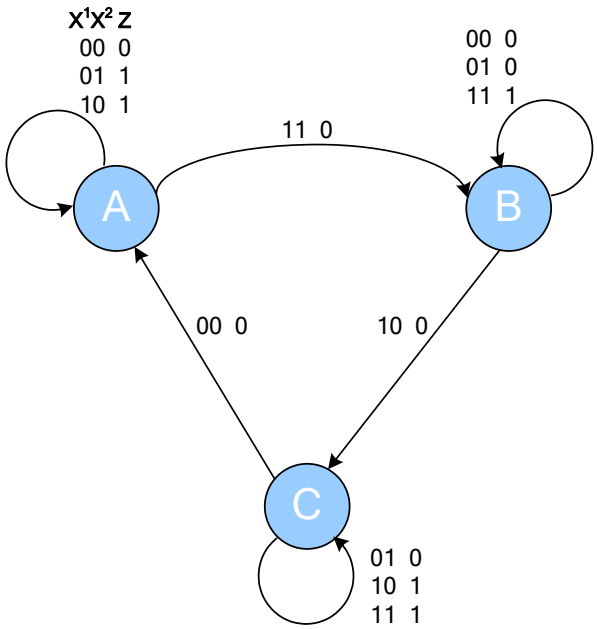
Kérdés:

a.) Tervezzen szinkron sorrendi hálózatot az alábbi állapotgráf és a megadott állapotkódolás alapján szinkron *T-tároló* felhasználásával!

b.) Adja meg, hogy mely modell szerint működik a hálózat!

c.) Adja meg a *kódolt állapottáblát, vezérlési táblát*, valamint ezek alapján a függvények *Karnaugh* tábláit.

**Állapotgráf:**




**Állapotkódok:**

$A := 00$

$B := 10$

$C := 11$

(mennyi szekunder változó szükséges?)

Az összevont állapottábla sorainak (=állapotainak) számától függ, hogy mennyi különböző szekunder állapotot kell felvenni:

3 sor esetén: $\lceil \log_2(3) \rceil = 2$, azaz két független szekunder állapot szükséges.

**Megoldás:**

a.) Előzetes állapottábla = összevont állapottábla (nincs megkülönböztethető sora), amelyből a fenti állapotkódok $\{A, B, C\}$ behelyettesítésével kapjuk a kódolt állapottáblát.

Összevont állapottábla

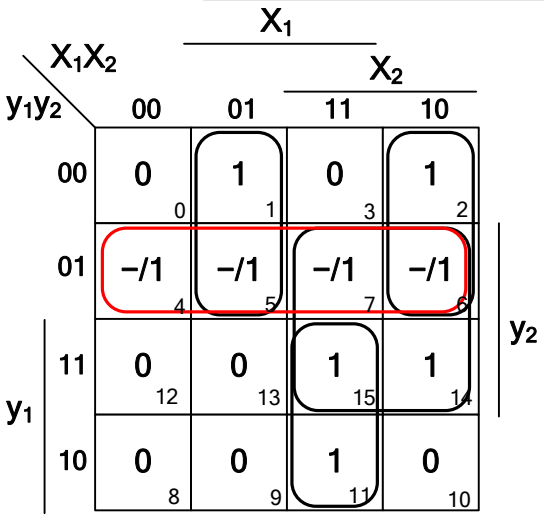
| y \ X | 00 $X^1$ | 01 $X^2$ | 11 $X^3$ | 10 $X^4$ |
|---|---|---|---|---|
| 00 A | A0 | A1 | B0 | A1 |
| 01 – | – | – | – | – |
| 11 C | A0 | C0 | C1 | C1 |
| 10 B | B0 | B0 | B1 | C0 |

Kódolt állapottábla

| $y_1y_2$ \ X | 00 $X^1$ | 01 $X^2$ | 11 $X^3$ | 10 $X^4$ |
|---|---|---|---|---|
| 00 A | 00 0 | 00 1 | 10 0 | 00 1 |
| 01 – | – | – | – | – |
| 11 C | 00 0 | 11 0 | 11 1 | 11 1 |
| 10 B | 10 0 | 10 0 | 10 1 | 11 0 |

b.) A $Z$ kimeneti függvény előállítása a következő:

Z kimeneti függvény




$$Z = X_1y_2 + X_1X_2y_1 + X_1\overline{X_2}\,\overline{y_1} + \overline{X_1}X_2\overline{y_1} + \overline{y_1}y_2$$ (pirossal a hazárdmentesítő hurok)

Mivel a $Z$ kimeneti függvény direkt módon függ a bemeneti kombinációktól, ezért a Mealy-modell lehet. Másrészt a Kódolt állapottáblán, illetve a $Z$ Karnaugh tábláján is egy-egy sorban nem azonos kimeneti értékeket kapunk, ami szintén a Mealy modell szerinti működést igazolja.

c.) Kódolt állapottábla alapján a Paraméter táblázat T-tároló sorából kapott ($T_1$, $T_2$) vezérlési táblák a következők:

$T_1$ vezérlési tábla

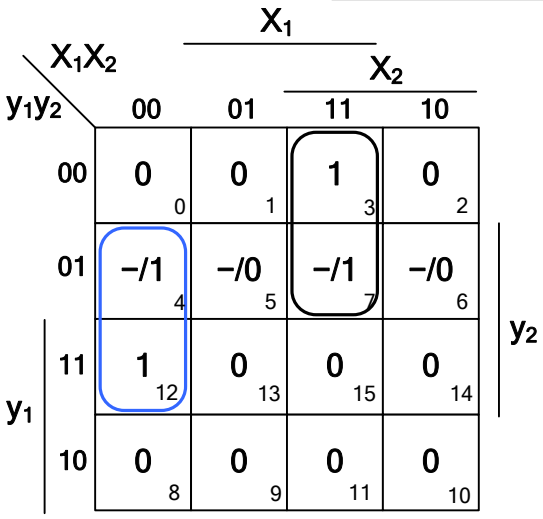
| $y_1y_2$ \ $X_1X_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 ($T_1$) | 0 | 1 | 0 |
| 01 | – | – | – | – |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

$T_2$ vezérlési tábla

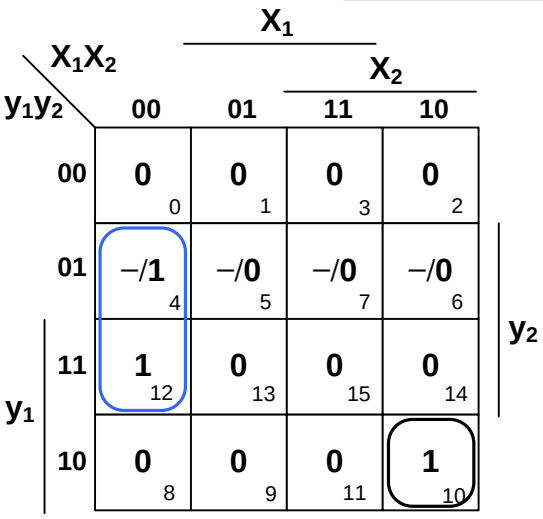
| $y_1y_2$ \ $X_1X_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 ($T_2$) | 0 | 0 | 0 |
| 01 | – | – | – | – |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 1 |

A vezérlési táblákból ($T_1$, $T_2$) kapott Karnaugh táblák a következők lesznek:

$T_1$ függvény




$T_2$ függvény




$$T_1 = \overline{X_1 X_2} y_2 + X_1 X_2 \overline{y_1}$$
$$T_2 = \overline{X_1 X_2} y_2 + X_1 \overline{X_2} y_1 \overline{y_2}$$

(kék jelöli a közös tagokat, elég egyszer megvalósítani)

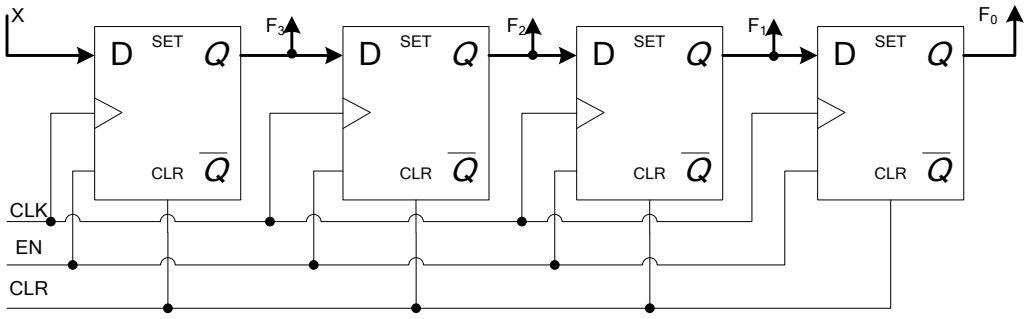
### 4.3 Példa: Sorrendi hálózat állapottáblájának összeállítása (4 bites jobbra léptető regiszter)

Kérdés: Állítsa össze a következő működéssel leírt szinkron sorrendi hálózat előzetes állapottábláját a Moore-modell elve alapján:

- Az áramkörnek egy 4 bites jobbra léptező (shift) regisztert kell realizálnia. Ez azt jelenti, hogy:
  - hálózatnak legyen egyetlen adatbemenete, jelöljük $X$-el
  - hálózatnak legyen 4 kimenete, jelöljük rendre $F_3$, $F_2$, $F_1$, $F_0$-val. (ezeken a kimeneteken keresztül vannak sorba kapcsolva az egyes tároló elemek)
  - az $X$ bemeneten érkező adat először az $F_3$, majd az $F_2$, az $F_1$, végül az $F_0$ kimeneteken jelenik meg (jobbra haladva léptetődik ki), azaz a magasabb helyiértékek felől az alacsonyabb helyiértékek felé haladva shiftelődnek az egymás utáni ciklusokban $X$ bemenetre érkező adatok.
  - Feltételezzük, hogy kezdetben a (0000) állapotban vagyunk, és 0000 a kimenet ($F_3$, $F_2$, $F_1$, $F_0$).

**Megoldás:**

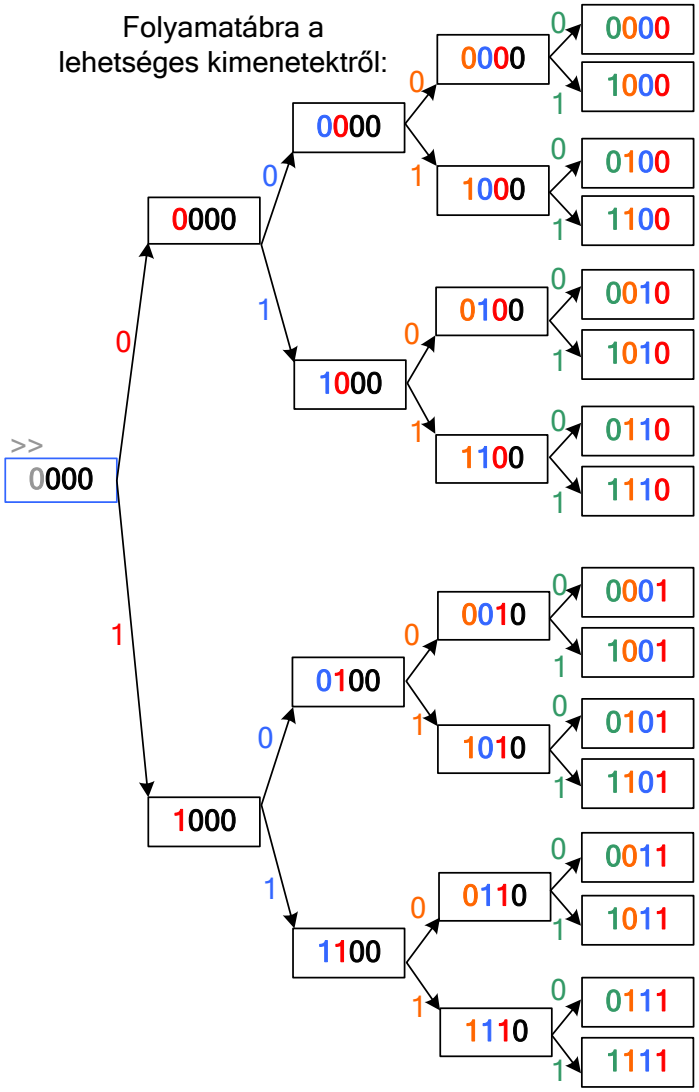
Egy 4-bites **jobbra** léptető/shift (>>) regiszter működését leíró szinkron sorrendi hálózat előzetes állapottáblát kell megvalósítani. Egy olyan sorrendi hálózat lényegében, amely megjegyzi a bemenetére az órajellel szinkronizáltan érkező utolsó 4 értéket, és azokat a kimeneteire helyezi. A megvalósítandó hálózat kereskedelmi forgalomban is kapható. (Katalógus adat: pl. SN54/74LS95)



Előzetes állapottábla összeállítása: A Moore-féle sorrendi hálózat kimenete csak az aktuális állapottól szabad, hogy függjön (bemenettől csak közvetve függ!), azaz a mindenkori kimenet azonos az állapottal. Jobb oldali ábrán az 1-pozícióval történő jobbra shiftelés mechanizmusa látható.

Előzetes állapottábla (Moore modell)

| y \ X | 0 >> | 1 |
|---|---|---|
| | Y $F_3F_2F_1F_0$ | |
| a | a 0000 | i 0000 |
| b | a 0001 | i 0001 |
| c | b 0010 | j 0010 |
| d | b 0011 | j 0011 |
| e | c 0100 | k 0100 |
| f | c 0101 | k 0101 |
| g | d 0110 | l 0110 |
| h | d 0111 | l 0111 |
| i | e 1000 | m 1000 |
| j | e 1001 | m 1001 |
| k | f 1010 | n 1010 |
| l | f 1011 | n 1011 |
| m | g 1100 | o 1100 |
| n | g 1101 | o 1101 |
| o | h 1110 | p 1110 |
| p | h 1111 | p 1111 |




### 4.4 Példa: Sorrendi hálózat állapottáblájának összeállítása (3 bites balra léptető regiszterre)

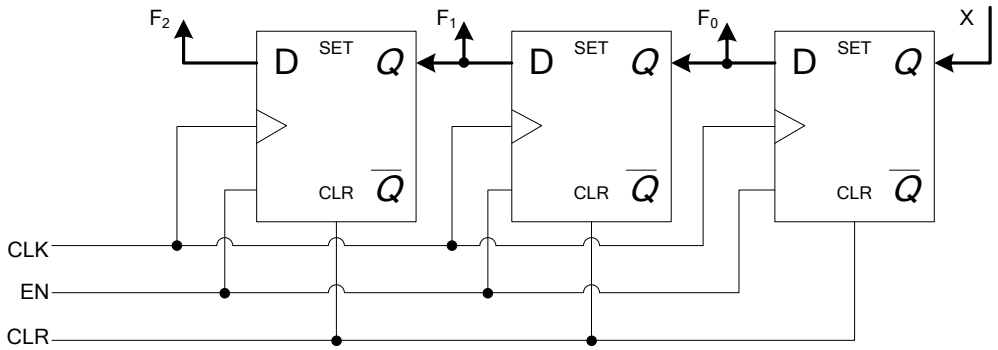
Kérdés:

Állítsa össze a következő működéssel leírt szinkron sorrendi hálózat előzetes állapottábláját mind a Mealy, mind pedig a Moore-modell elve alapján:

- Az áramkörnek egy 3-bites *balra* léptező (shift) regisztert kell realizálnia. Ez azt jelenti, hogy:
  - hálózatnak legyen egyetlen adatbemenete, jelöljük $X$-el
  - hálózatnak legyen 3 kimenete, jelöljük rendre $F_2$, $F_1$, $F_0$-val. (ezek sorba vannak kapcsolva),
  - az $X$ bemeneten érkező adat először az $F_2$, majd az $F_1$, végül az $F_0$ kimeneteken jelenik meg (balra haladva léptetődik ki), azaz a magasabb helyiértékek felől az alacsonyabb helyiértékek felé haladva shiftelődnek az egymás utáni ciklusokban $X$ bemenetre érkező adatok.

**Megoldás:**

Egy 3-bites **balra** léptető/shift regiszter (<<) működését leíró szinkron sorrendi hálózat előzetes állapottáblát kell megvalósítani. Egy olyan sorrendi hálózat lényegében, amely megjegyzi a bemenetére az órajellel szinkronizáltan érkező utolsó 3 értéket, és azokat a kimeneteire helyezi. A megvalósítandó hálózat kereskedelmi forgalomban is kapható. (Katalógus adat: pl. SN54/74LS95 )




Előzetes állapottábla összeállítása: A Moore-féle sorrendi hálózat kimenete csak az aktuális állapottól szabad, hogy függjön, azaz a mindenkori kimenet azonos az állapottal. Jobb oldali ábrán az 1-pozícióval történő „balra léptetés mechanizmusa" látható.

***Fontos megjegyzés Moore modell esetén:*** *az $X$ bemenet hatása* ***csak azután fog érvényesülni*** *az $F$ kimeneteken (elsődlegesen az $F_2$-n), miután az általa létrehozott $Y$ visszahat $y$-ként a bemenetre! (Moore féle szinkronizálás).*

Következmény: **Lényeges eltérés Mealy-Moore modellek működése között**, hogy

- *Mealy* esetén mindig az **új kimeneti** értéket kell bejegyezni **a következő állapot mellé**, míg
- *Moore* esetén **a kiindulási kimeneti** értéket kell bejegyezni a **következő állapot mellé**!

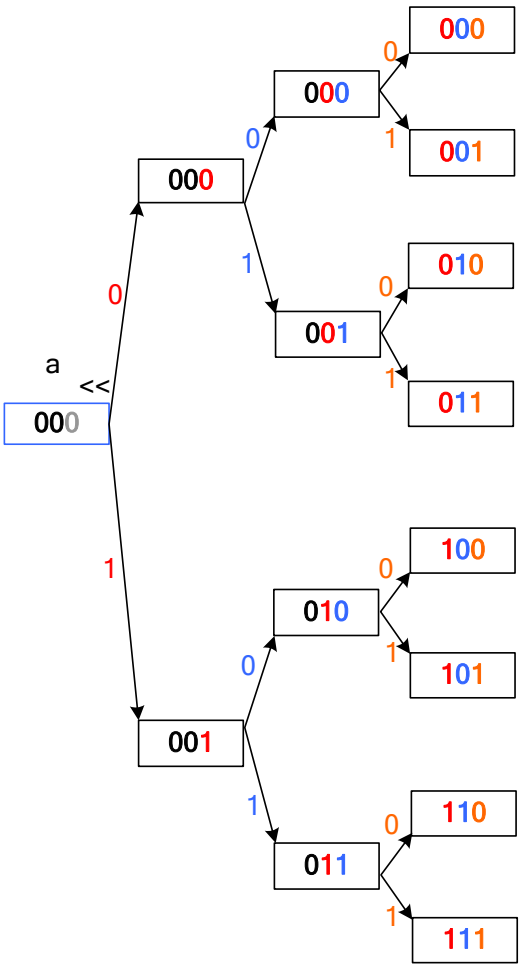
Előzetes állapottábla (MEALY esetben)

| y \ X | 0 | 1 |
|---|---|---|
| 000 a | a 000 | b 001 |
| 001 b | c 010 | d 011 |
| 010 c | e 100 | f 101 |
| 011 d | g 110 | h 111 |
| 100 e | a 000 | b 001 |
| 101 f | c 010 | d 011 |
| 110 g | e 100 | f 101 |
| 111 h | g 110 | h 111 |

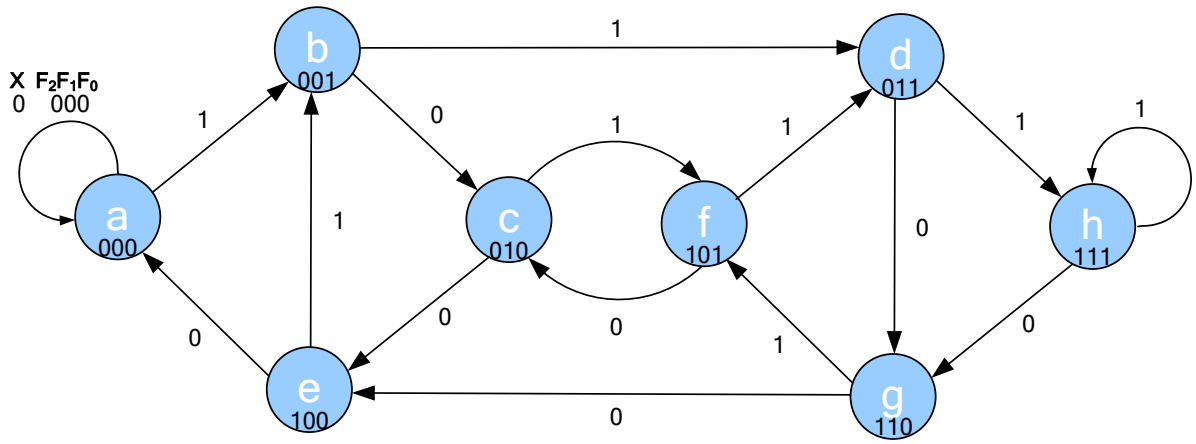
Előzetes állapottábla (MOORE esetben)

| y \ X | 0 | 1 | Y $F_2F_1F_0$ |
|---|---|---|---|
| 000 a | a 000 | b 000 | 000 |
| 001 b | c 001 | d 001 | 001 |
| 010 c | e 010 | f 010 | 010 |
| 011 d | g 011 | h 011 | 011 |
| 100 e | a 100 | b 100 | 100 |
| 101 f | c 101 | d 101 | 101 |
| 110 g | e 110 | f 110 | 110 |
| 111 h | g 111 | h 111 | 111 |

Folyamatábra a lehetséges kimenetekről:




Előzetes állapottáblák alapján felrajzolt állapottábla a következő:




A fenti két előzetes táblát megvizsgálva az látható, hogy a Moore modell esetén nem lehet

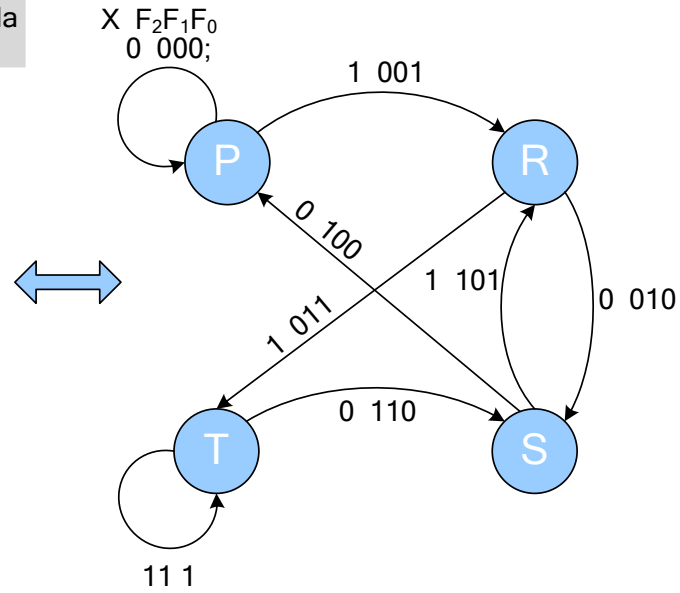
összevonni az előzetes állapottáblát, tehát az egyben összevont állapottábla is. Mealy modell esetén azonban vannak feleslegesen megkülönböztetett sorok, ezeket a következő módon vonhatjuk össze:

szekunder állapotok

- $P \coloneqq a, e$ *(1.-,5. sorok azonosak)*
- $R \coloneqq b, f$ *(2.-,6. sorok azonosak)*
- $S \coloneqq c, g$ *(3.-, 7.. sorok azonosak)*
- $T \coloneqq d, h$ *(4.-,8. sorok azonosak)*

Összevont állapottábla (Mealy esetben)

| y \ X | 0 | 1 |
|---|---|---|
| | Y $F_2F_1F_0$ | |
| P | P 000 | R 001 |
| R | S 010 | T 011 |
| S | P 100 | R 101 |
| T | S 110 | T 111 |




A fenti összevonás is azt a tényt igazolja, hogy a Moore modellek általában mindig több állapottal fejezhetők ki, valósíthatóak meg (összevonás után), mint a Mealy modellek.

# 5. Aszinkron sorrendi hálózatok tervezése

Aszinkron sorrendi hálózatok tervezési és megvalósítási lépései a következők:

1. **Logikai feladat megfogalmazása:** aszinkron esetben is általában egy szöveges feladat, mely alapján az előzetes állapottábla összeállítása elkezdődhet.
2. **Előzetes állapottábla összeállítása.** Ehhez szisztematikus módszer nem létezik, a szöveges feladat megfelelő értelmezését feltételezve intuitív (sokszor próbálgatásos) jellegű. Fontos azonban, hogy aszinkron esetben kizárólag szomszédos bemeneti kombináció-változásokat tételezünk fel, és minden sorban csak egy stabil állapotot írunk. A feleslegesen megkülönböztetett állapotok a tervezés további lépéseiben összevonhatók, tehát nem jelentenek különösebb nehézségét.
3. **Egyszerűsített (összevont) állapottábla**. Ugyanúgy végezhető, mint szinkron esetben. Ha az előzetes állapottábla teljesen specifikált (TSÁ), akkor erre a tervezési lépésre szisztematikus módszerek ismeretesek. Ha az előzetes állapottábla nem teljesen specifikált (NTSÁ), akkor pedig ugyanezek a módszerek használhatók, de a segítségükkel kapott eredményből csak intuitív lépések útján kaphatjuk meg a legegyszerűbb összevont állapottáblát.
4. **Állapotkódolás helyes megválasztása**. az állapotkódolás kialakításakor nem a legegyszerűbb hálózat kialakítása, hanem elsődlegesen a ***'kritikus versenyhelyzet'*** (vagy más néven *'rendszer hazárd'*) kiküszöbölése a cél. Erre szisztematikus eljárások léteznek. Kritikus versenyhelyzet a szekunder kombinációk közötti változás különbségből adódhat.
5. **Alkalmazandó tároló típusának kiválasztása.** Kétféle módszer vizsgálható:
    a. egyrészt tisztán visszacsatolt K.H. segítségével, vagy
    b. aszinkron működésű elemi tárolókból felépítve (pl. S-R, vagy D-G).

    Erre a tervezési lépésre sem létezik egyértelmű szisztematikus eljárás, intuitív módon a megvalósító flip-flopok típusát többnyire a rendelkezésre álló építőelem-készlet (mintegy alkatrész bázis) alapján határozzuk meg.
6. **Vezérlési tábla összeállítása**. Ugyanúgy végezhető, mint szinkron esetben, azonban tisztán visszacsatolással megvalósítva a vezérlési tábla azonos az állapottáblával. Ez is többnyire teljesen szisztematikus módszer (kivétel a D-G tároló használata). Csak akkor tudjuk a vezérlési táblát összeállítani, ha már pontosan ismerjük a kiválasztott flip-flop-ot. Lényegében a megvalósításra szánt flip-flop-ot vezérlő kombinációs hálózat állapot tábláját, mint vezérlési táblát kell ebben a lépésben megadni.
7. **„Lényeges hazárd" ellenőrzése:** a megfelelő szekunder változók késleltetése, a bemenetekhez képest. Ha szükséges, a bemenetek és szekunder változók közötti versenyhelyzet feloldása itt a cél. Szintén kevésbé tekinthető szisztematikus eljárásnak.

**Megjegyzés: Aszinkron sorrendi hálózat szinkron Mealy vs. Moore modell szerinti működése**

- Ha az aszinkron S.H. előzetes állapottáblájának minden sorába csak egyetlen stabil állapotot definiáltunk, még nem döntöttünk annak Mealy, vagy Moore modell-szerű működéséről. Ez csak az *összevont* állapottábla alapján dönthető el!
- Csak akkor tételezhetünk fel Mealy modellt, ha összevonunk olyan állapotokat is, amelyekhez tartozó bemeneti kombinációtól függően eltérő kimeneti értékek tartoznak, illetve az e*gy sorban lévő közömbös kimeneti értékeket egymástól eltérő módon rögzítjük* (választjuk meg), ezáltal soronként nem azonos kimeneti értékeket kapunk.

## *Aszinkron sorrendi hálózat komplex tervezési feladata: D-FF tervezése aszinkron módon*

Tervezéskor a feltételezésünk: egymás után következő bemeneti kombinációk esetén csak szomszédos bemeneti változások megengedettek, azért, hogy a funkcionális hazárdot kiküszöböljük már a tervezés során.

E nélkül ugyanis, az aszinkron S.H. működése nem lenne egyértelműen követhető az állapottáblán.

Példaként. $AB$ = 00 → 11 helyett 00 → 0**1** / **1**0 → 11 lehetséges, attól függően, hogy $A$, vagy $B$ változását érzékeljük előbb (ha ezt nem kötjük ki, akkor ugyan egy másik stabil, de nem az előírt stabil állapotba kerülhet a sorrendi hálózatunk, amely szintén hibás működést eredményezhet)

### a) Logikai feladat megfogalmazása:

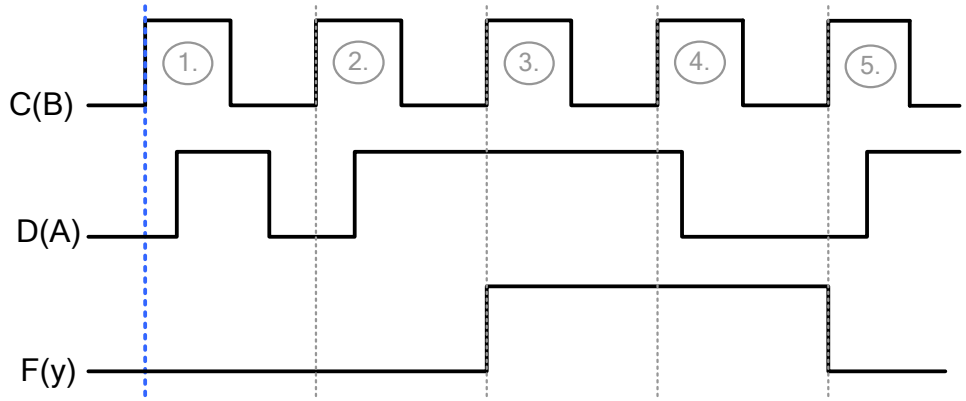
Aszinkron S.H. hálózat tervezése, amely „kívülről" nézve egy szinkron tárolóként viselkedik.

- A megvalósítandó szinkron tároló legyen egy D-FF
- Biztosítani kell a D-FF-re nézve a „Szinkronizációs feltételeket", azaz
- CLK (C) órajelet tekintsük független bemeneti jelnek

Működése: tegyük fel, hogy az $F$ kimenete csak akkor változhat, amikor $C(A)$ értéke '0'→'1' (felfutó élre vált), és ekkor $F$ vegye fel a $D$ bemenetnek a mindenkori pillanatnyi értékét, $F = D(B)$.

Megjegyzés: a (zárójelben) megadott kifejezések a belső felhasznált aszinkron S.H. jeleit definiálják!

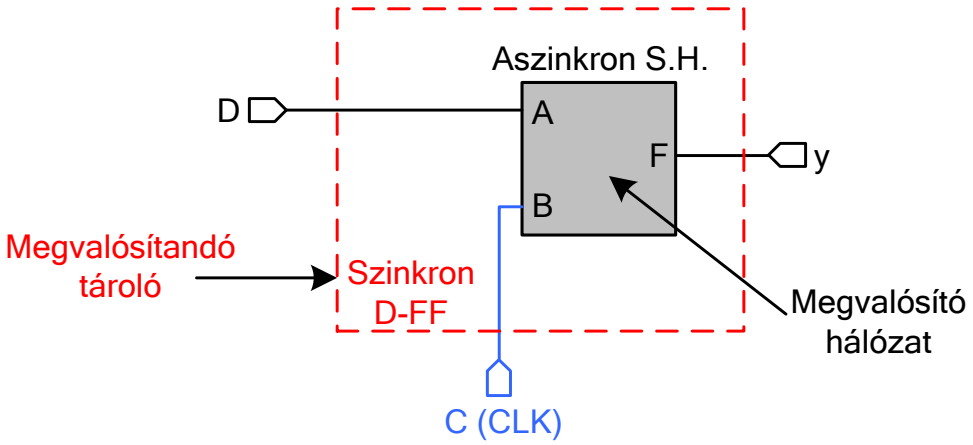
Az aszinkron hálózattal megvalósítható szinkron D-FF működéséhez szükséges jelek szemléltetése:




**5.1 ábra: aszinkron sorrendi hálózat jelei ($A$, $B$) vs. D-tároló jelei ($D$, $C$)**

Látható az idődiagramon, hogy a nem-szomszédos $C(B)$-$D(A)$ bemeneti kombináció-változásokat nem engedélyeztük.

Segítségként a szinkron D tároló, aszinkron sorrendi hálózattal történő elvi megvalósítása látható az alábbi ábrán:




**5.2 ábra: szinkron D tároló felépítése aszinkron sorrendi hálózat segítségével**

### b) Előzetes állapottábla összeállítása

A S.H.-nak különbözőképpen kell reagálnia ugyanarra a bemeneti kombinációra a megelőző bemeneti kombinációktól és a hálózat állapotától függően.

Aszinkron sorrendi hálózat esetén a következők az érvényesek:

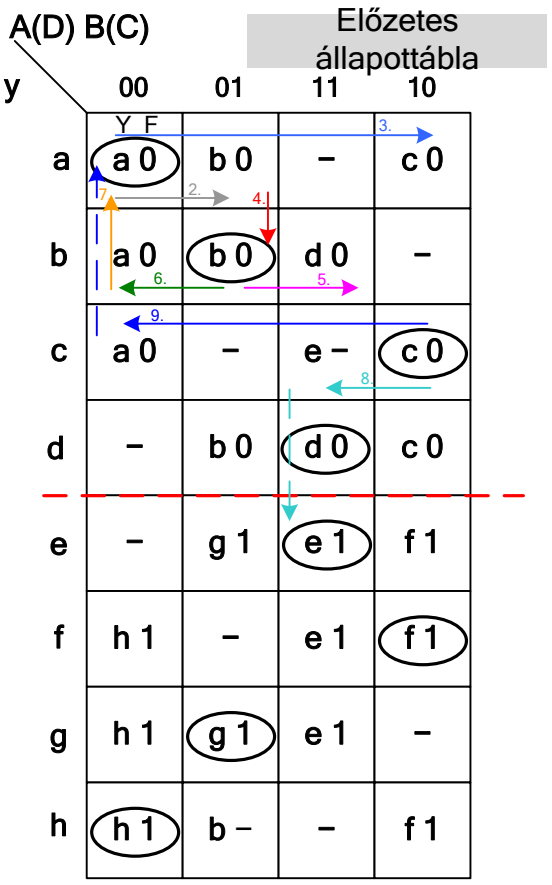
- „Előzetes" állapottáblában általában **több** állapot szerepel a szükségesnél.
- Kimenet pillanatnyi értéke: $F$
- Lehetőség szerint minden sorban csak egyetlen **stabil** állapotot vegyünk fel.
- Adjunk meg stabil kiindulási feltételt-
- A stabil állapothoz képest (karika) csak a szomszédos bemeneti kombináció-változásokat engedélyezünk. A stabil állapothoz képest, amelyik cellában nem-szomszédos a bementi kombináció-változás oda, **don't care** '–' értéket rögzítünk. Ez jelentősen megnövelheti az esetleges állapot összevonást a következő lépésben.
- Ismétlésként: csak stabil állapotból lehet a bemeneti kombinációknak változnia (vízszintesen haladva), különben csak y-mentén haladhatunk (függőlegesen)!

**Kitöltés: *(1. sor)***

1. Tfh. $D(A); C(B) = 00$ stabil **kiindulási állapot**, ahol: $y \coloneqq a$ és $F \coloneqq 0$. Ekkor: $\boldsymbol{a0}$. (stabil $y \Leftarrow Y \coloneqq a$). Ez azt jelenti, hogy $D(A); C(B) = 10$ *közömbösnek* rögzítendő „–" (funkcionális hazárd mentesség).
2. Tfh: $D(A); C(B) = 01$-re vált. (Ekkor $D = 0$, $\boldsymbol{C = 0 \rightarrow 1}$, tehát a kimenet pill. értéke $F = 0$ lesz, felveszi $D$-t). Legyen $b$ instabil állapot. Ekkor: $b0$. ($y \neq Y = b$ instabil)
3. Tfh: $a0$ stabil állapotban vagyunk és $D(A); C(B) = 10$-ra vált. (Ekkor $\boldsymbol{D = 0 \rightarrow 1}$, $C = 0$, tehát a kimenet pill. értéke $F = 0$ marad változatlanul). $c$ instabil állapot. Ekkor: $\boldsymbol{c0}$. ($y \neq Y = c$ instabil). //a 3. sor $c0$ stabil állapotába viheti át//

***(2. sor)***

4. Ha a 2. lépésben kialakuló $b0$ mint $Y => y$ *visszahat* a bemenetre, akkor a 2. sorba jutunk, ahol $\boldsymbol{b0}$ **stabil** állapot lesz ($D(A); C(B) = 01$ ***és*** $y = b$). Ez azt jelenti, hogy $D(A); C(B) = 10$ *közömbösnek* rögzítendő „–" (hazárd mentes).
5. Tfh: $b0$ stabil állapotban vagyunk és $D(A); C(B) = 11$-re vált. (Ekkor $\boldsymbol{D = 0 \rightarrow 1}$, $C = 1$, tehát a kimenet pill. értéke $F = 0$ marad változatlanul). $d$ instabil állapot. Ekkor: $\boldsymbol{d0}$. ($y \neq Y = d$ instabil).
6. Tfh: $b0$ stabil állapotban vagyunk és $D(A); C(B) = 00$-ra vált. (Ekkor $D = 0$, $\boldsymbol{C = 1 \rightarrow 0}$, tehát a kimenet pill. értéke $F = 0$ marad változatlanul). $a$ instabil állapot. Ekkor: $\boldsymbol{a0}$. ($y \neq Y = a$ instabil).
7. Az $a0$ instabil állapotból (2. sor) az $a0$ stabil állapotba kerülhetünk vissza (1. sorba) >>



***(3. sor)***

8. Tfh: a 3. sorban $c0$ stabil állapotban vagyunk azaz $y = c$, és $D(A); C(B) = 11$-ra vált. (Ekkor $\boldsymbol{D = 1}$, $\boldsymbol{C = 0 \rightarrow 1}$, tehát a kimenet pill. értéke $F = 1$ kell legyen). Az '11' oszlopon belül olyan stabil állapotba kell jutni, amelyhez $F = 1$ kimeneti érték tartozik. Ezt jelöljük $\boldsymbol{e}$-vel. **Kimenetként** azért írhatunk **közömbös** '–' értéket, mivel a $c0$ stabil állapotból $\boldsymbol{e}$– instabil állapoton keresztül $e1$ stabil állapotba kerülhet a rendszer, ahol a *kimeneti érték megváltozik* ($\boldsymbol{c0 \rightarrow e1}$). Ilyen esetekben, amikor a stabil – instabil állapothoz tartozó kimeneti érték párok eltérőek, *közömbös kimeneti értéket* kell rögzíteni a köztes instabil állapothoz. Eközben $c0$ stabil állapot azt jelenti, hogy $D(A); C(B) = 01$ *közömbös „–"* (hazárd mentes).
9. Tfh: $c0$ stabil állapotban vagyunk és $D(A); C(B) = 00$-re vált. (Ekkor $\boldsymbol{D = 1 \rightarrow 0}$, $C = 0$, tehát a kimenet pill. értéke $F = 0$ marad változatlanul). $a$ instabil állapot. Ekkor**:** $\boldsymbol{a0}$. ($y \neq Y = c$ instabil). Innen akár az $a0$ stabil állapotba is visszakerülhetünk (1. sorba)
10. És így tovább…

### c) Összevont (egyszerűsített) állapottábla

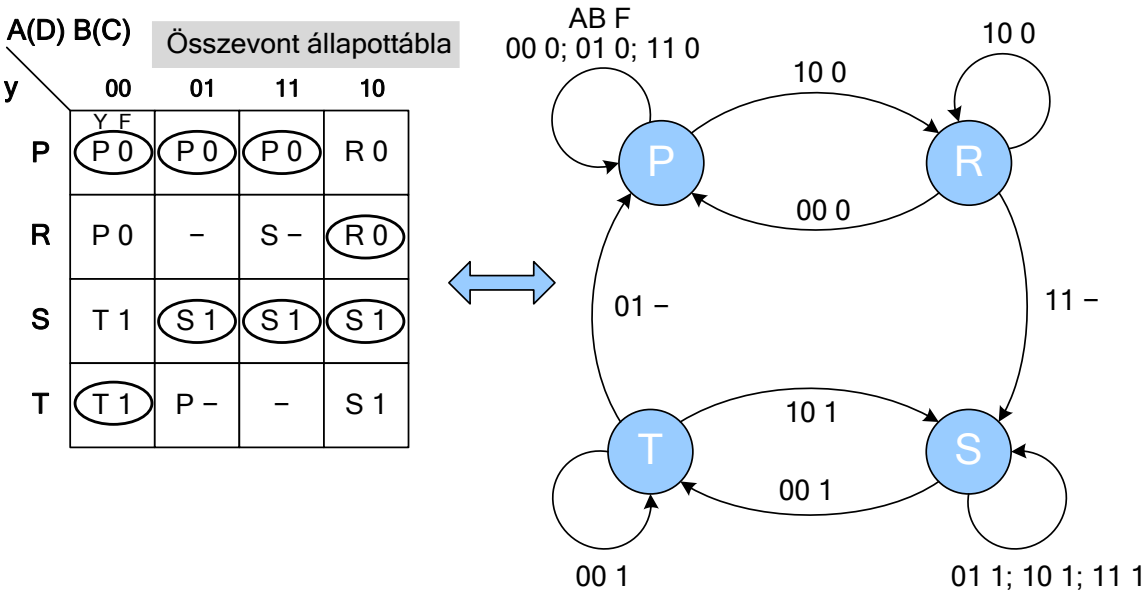
Előzetes állapottábla alapján az a cél, hogy megkeressük a lehető legkevesebb megkülönböztetendő állapotot. Ezek ismeretében kell létrehozni a lehető legkevesebb, de azonos sort nem tartalmazó új állapottáblát, amelyet összevont állapottáblának nevezünk, amelynek így már nem lesz feleslegesen megkülönböztetett állapota (=sora). Fontos tudni, hogy az összevont állapottábla nem azonos a később ismertetésre kerülő állapot kódolási módszerrel!

A fenti előzetes állapottábla következő sorait vonhatjuk össze:

szekunder állapotok

- $P \coloneqq a, b, d$ *(1.-,2.-, és 4. sorok azonosak)*
- $R \coloneqq c$ *(3. sor)*
- $S \coloneqq e, f, g$ *(5.-,6.- és 7. sorok azonosak)*
- $T \coloneqq h$ *(8. sor)*

Az összevonás során kapott összevont állapottábla és a vele ekvivalens állapotgráf a következő:




**5.3 ábra: Összevont állapottábla és állapotgráf**

Megjegyzés: az előzetes állapottáblán, illetve az összevont állapottáblán is, ahogy várható, minden oszlopban rögzítve lett stabil állapot, ami biztosítja a nem oszcilláló (normál aszinkron) működést.

### d) Kódolt állapottábla

Az összevont állapottábla alapján a szimbolikus összevonásként ($A$, $B$) jelölt állapotoknak meg kell feleltetni egy-egy szekunder kombinációt ($y$).

Az összevont állapottábla sorainak (=állapotainak) számától függ, hogy mennyi különböző szekunder állapotot kell felvenni.

$n$ sor esetén: $\lceil \log_2(n) \rceil$ szekunder állapot szükséges

**Választott állapotkódok**: *a feladat szempontjából, valamint a hazárdjelenségek kiküszöbölése miatt egyáltalán nem közömbös, hogy miként rögzítjük az állapotkódokat.* Jelen példában, az egyszerűség kedvéért az állapotkódokat még direkt módon adtuk meg, azonban a későbbi 8. és 9. fejezetekben részletesen ismertetjük az állapotkódolás módszereit.

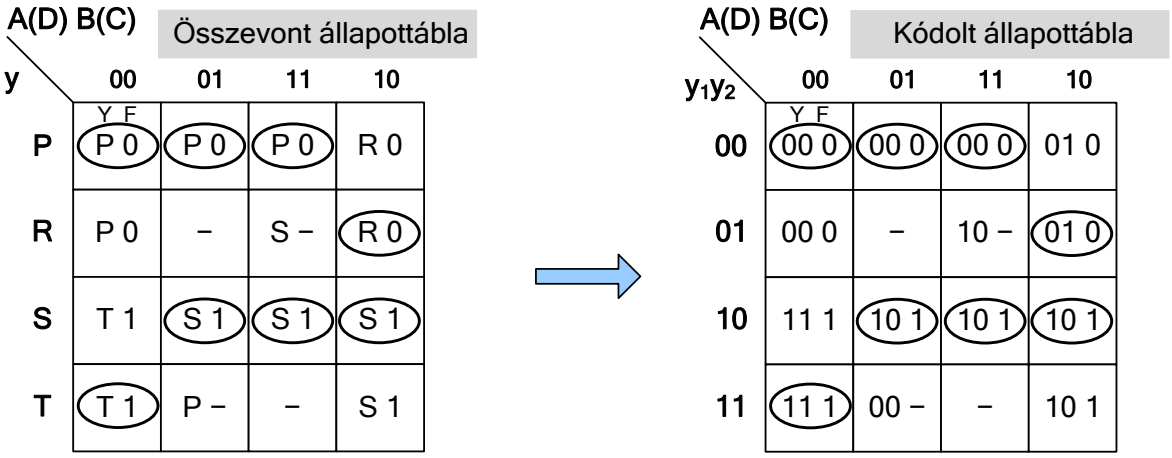
### Állapotkódok: I. módszer (Kritikus versenyhelyzet)

- Összevont állapottábla 4 állapota → 2 szekunder ($y_1$, és $y_2$) változó (két-két lehetséges) kódolt értékekkel:

| összevont állapot | $y_1$ | $y_2$ |
|---|---|---|
| P | 0 | 0 |
| R | 0 | 1 |
| S | 1 | 0 |
| T | 1 | 1 |

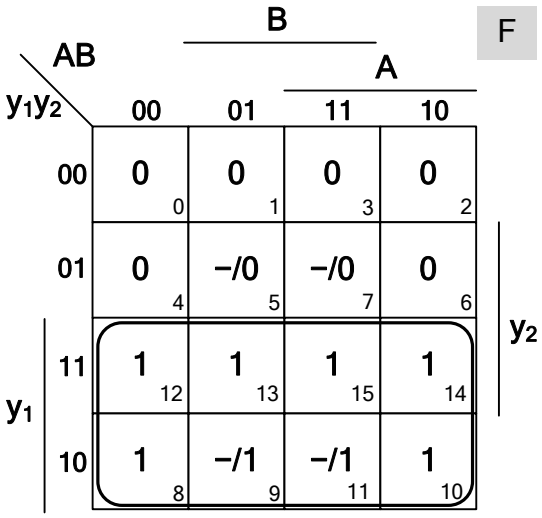
Megjegyzés: természetesen ez a kódolási séma csak az egyik lehetséges eset; lehetett volna akár további eseteket is definiálni. Itt az aszinkron esetben a kritikus versenyhelyzet (más néven rendszer hazárd) ismertetésénél tovább vizsgáljuk az állapotkódolást (I. és II. módszer összevetése).

Kódolt állapottábla (I. módszer szerint) felírása a következő:




**5.4 ábra: Kódolt állapottábla felírása**

Ezután történhet a **kimeneti függvény ($F$)** Karnaugh táblájának felírása:




**5.5 ábra: Az $F$ kimeneti függvény Karnaugh táblájának felírása**

A Karnaugh tábla alapján az $F$ kimenetre kapott DNF alak a következő:

$$F = y_1$$

Az $F$ függvény statikus, illetve dinamikus értelemben is hazárdmentes, míg a funkcionális hazárdot a nem szomszédos változások kiküszöbölésével már elkerültük. Amit viszont nem biztos, hogy elkerültünk, az az ún. *„kritikus versenyhelyzet*", vagy más néven „*rendszer-hazárd",* amelyet meg kell vizsgálni.

### Kritikus versenyhelyzet

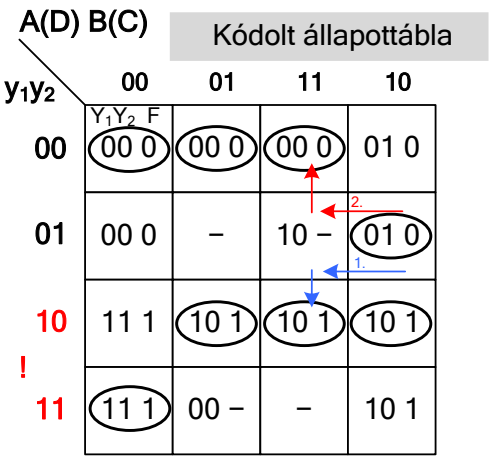
Az állapotkódolással az aszinkron sorrendi hálózat 'helyes', vagy 'hibás' működését befolyásolhatjuk.

Def: Az aszinkron S.H. 'helyes' vagy 'hibás' működése (a hibás stabil állapotba jutás is) az $y$ szekunder változók egymáshoz képesti működési sebességétől függ. Ezt a sebességarányt a hálózatban fellépő jelterjedési késleltetések (visszacsatoló ágban) döntően befolyásolják, ezért a hálózat viselkedése véletlenszerű lehet.

**Kritikus versenyhelyzet (vagy Rendszer Hazárd)**: kettő, vagy több szekunder állapot (esetünkben pl. $y_1$ és $y_2$) közötti verseny, mivel változási sebességük eltérő.

Az állapotkódolás megválasztása nemcsak a megvalósítandó aszinkron S.H. egyszerűségét, hanem annak helyes/hibás működését is döntően befolyásolja.

Példaként vizsgáljuk meg az I. módszer szerinti állapotkódolással megadott kódolt állapottáblát a kritikus versenyhelyzet szempontjából:




**Egyidejű változás:** Tfh. $AB = 10$ mellett az $y_1y_2 = 01$ stabil állapotban van a rendszer (4.oszlop / 2.sor)

1. Változzon a bemenet $AB = 11$-re.
   Ekkor a hálózatnak az '11' oszlopában $y_1y_2$ = '10' stabil állapot van előírva (3. sor), melyet a $y_1y_2\ Y_1Y_2 = 01\ 10$ instabil állapoton keresztül ér el. (azaz $Y = 10$ szekunder kombináció visszahat $y$ –ra). Feltétel ekkor az, hogy a bemeneti változás hatására a szekunder változóknak $y_1 = 0 \rightarrow 1$, míg $y_2 = 1 \rightarrow 0$ **kell egyszerre változnia!**

**Ha mégsem egyszerre változna:**

2. Tfh. $y_2$ **gyorsabban** változik, mint $y_1$.
   Először $y_2 = 1 \rightarrow 0$. Ekkor átmenetileg létrejön az $y_1y_2 = 00$ szek. kombináció. Innen viszont $Y(00) \rightarrow y$ visszahatására az '11' oszlop '00' cellájába (1. sor) kerülünk, amely ugyan szintén stabil, de 'hibás' stabil állapot (az előírt $y_1y_2 = 10$ helyett!)

**5.6 ábra: Kritikus versenyhelyzet vizsgálata**

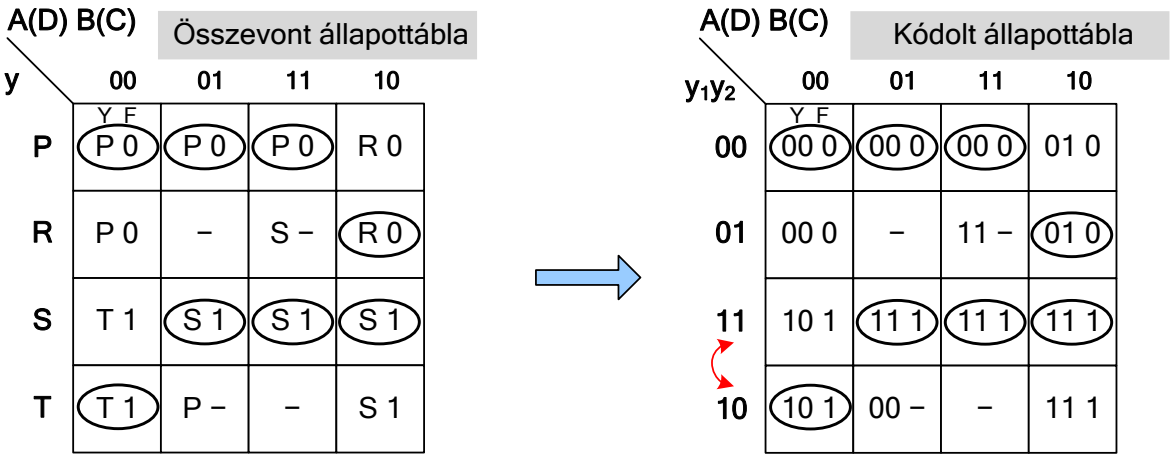
### Állapotkódok: II. módszer (nem-kritikus versenyhelyzet)

- Összevont állapottábla 4 állapota → 2 szekunder ($y_1$ és $y_2$) változó (két-két lehetséges) kódolt értékekkel:

| összevont állapot | $y_1$ | $y_2$ |
|---|---|---|
| P | 0 | 0 |
| R | 0 | 1 |
| S | 1 | 1 |
| T | 1 | 0 |

Megjegyzés: ez a kódolási séma csak az egyik lehetséges eset. Ez a II. kódolási módszer annyiban különbözik az I. módszertől, hogy az $S$, illetve $T$ állapotok értékei fel lettek cserélve és rendre '11', illetve '10' ('10', illetve '11' helyett).

A kódolt állapottábla (II. módszer szerinti) felírása a következő:



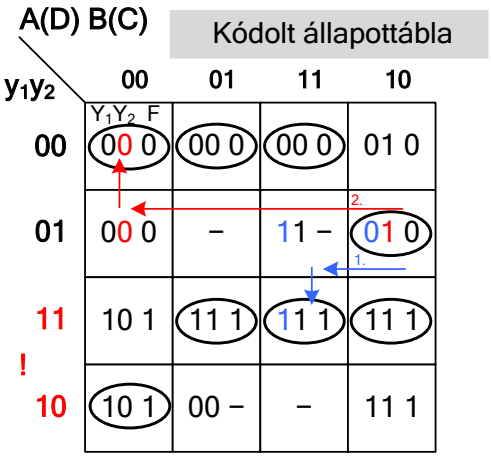
**5.7 ábra: Kódolt állapottábla felírása**

A Karnaugh tábla alapján az $F$ kimenetre kapott DNF alak a II. módszer esetén is azonos: a sorok felcserélése nem változtat a kimeneten (Eml: $F = y_1$).

### Nem-kritikus versenyhelyzet

Ahogy látattuk, az állapotkódolással az aszinkron sorrendi hálózat 'helyes', vagy 'hibás' működését befolyásolhatjuk.

Példaként vizsgáljuk meg a mostani az II. módszer szerinti állapotkódolással megadott kódolt állapottáblát a kritikus versenyhelyzet szempontjából:




Tfh. $AB = 10$ mellett az $y_1 y_2 = 01$ stabil állapotban van a rendszer (4.oszlop / 2.sor)

Előző I. módszerhez képest itt a szekunder változók eltérő sebességarányának változása ellenére **sem alakul ki kritikus versenyhelyzet**, mivel mindig *csak egyetlen szekunder változó* változtatja az értékét ennél a II. módszerbeli állapotkódolásnál!

1. $Y_1$ változik: $01 \rightarrow 11 \rightarrow 11$
2. $Y_2$ változik: $01 \rightarrow 00 \rightarrow 00$

Mindkét módszerrel az előírt helyes stabil állapotba jutunk, azaz nem alakul ki kritikus versenyhelyzet.

**5.8 ábra: Nem-kritikus versenyhelyzet**

### e) Alkalmazandó tároló típusának kiválasztása

Egy aszinkron sorrendi hálózat esetében kétféle módszer alkalmazható:

- **a.) egyrészt tisztán visszacsatolt K.H. segítségével**, ilyenkor a vezérlési tábla azonos az kódolt állapottáblával (mivel $Y_1$-et ill. $Y_2$-t előállító függvényeket kell megvalósítani, majd visszacsatolni), vagy
- **b.) adott aszinkron működésű elemi tárolókból felépítve** (pl. S-R, vagy D-G megvalósító FF közül választhatunk). A tárolók működésének ismeretében (*3. fejezet*) határozhatjuk meg, miként kell vezérelni őket egy K.H. segítségével a 4.lépésben felírt kódolt állapottábla alapján.

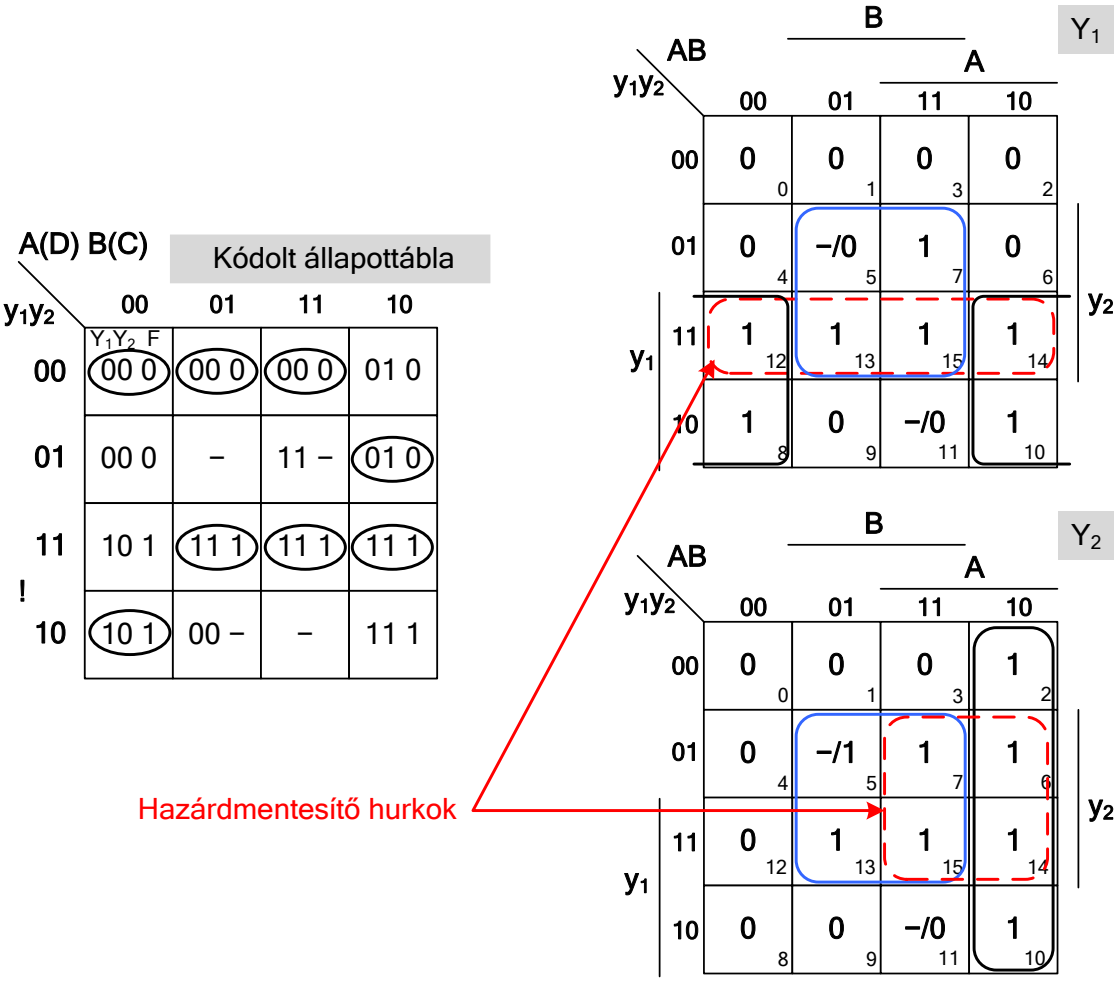
Jelen példában 2 megvalósító FF-t kell használni, mivel két szekunder változó van: $y_1$, $y_2$. Így kapjuk meg a vezérlési táblát.

Megjegyzés: abban az esetben, ha több szekunder ($y$) változónk is van, akár különböző aszinkron viselkedésű FF-kat rendelünk az építőelem készletből egy-egy szekunder változó (állapot) tárolásához. Jelen példában mindkét változóhoz csak egyfajta tárolót rendelünk majd. A vezérlési tábla felírásához pedig a Paraméter táblázat használata szükséges (*lásd 3. fejezet*), hasonlóan, ahogy azt a szinkron hálózatok tervezése esetén megismerhettük.

### f) Vezérlési tábla összeállítása

#### a. Tisztán visszacsatolt kombinációs hálózattal történő megvalósítás

Vezérlési tábla = kódolt állapottábla felírása:




A kódolt állapottábla (mint vezérlési táblák) celláinak megfelelő oszlopaiból képzett Karnaugh táblákból leolvashatók a következő állapot függvényei, külön $Y_1$ illetve $Y_2$-re:

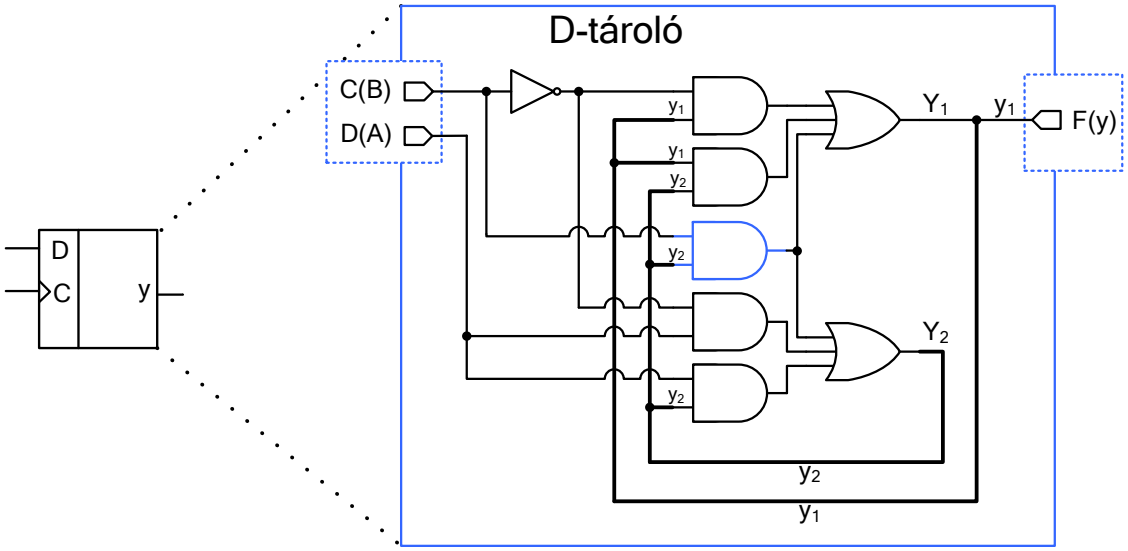
$$Y_1 = By_2 + \overline{B}y_1 + y_1y_2$$

$$Y_2 = By_2 + A\overline{B} + Ay_2$$

$F = y_1$ (korábban kiszámoltuk)

A vezérlő kombinációs hálózatok Karnaugh tábláiban ($Y_1$, $Y_2$) arra törekszünk, hogy a tömbösítés során, hogy egyrészt a létező legnagyobb lefedés(eke)t, másrészt ezek közül is a legkevesebb számú lefedés(eke)t vonjuk össze. Az összevonások során statikus-, illetve dinamikus-értelemben is hazárdmentes alakok megadására törekszünk (pirossal jelölve), a funkcionális hazárd kiküszöböléséről már ez előzetes állapottábla felírása során gondoskodtunk. Az $Y_1$, $Y_2$ következő állapot függvényeknél kékkel a közös prímimplikánsokat jelöltük, amelyet elegendő egyetlen egyszer megvalósítani. Látható, hogy az $F$ kimeneti függvény csak az $y_1$ szekunder állapottól függ, Moore modell adott.

**Elvi kapcsolási rajz: aszinkron sorrendi hálózattal megvalósított szinkron D-tároló megvalósítása tisztán visszacsatolással, és elvi logikai rajz.**



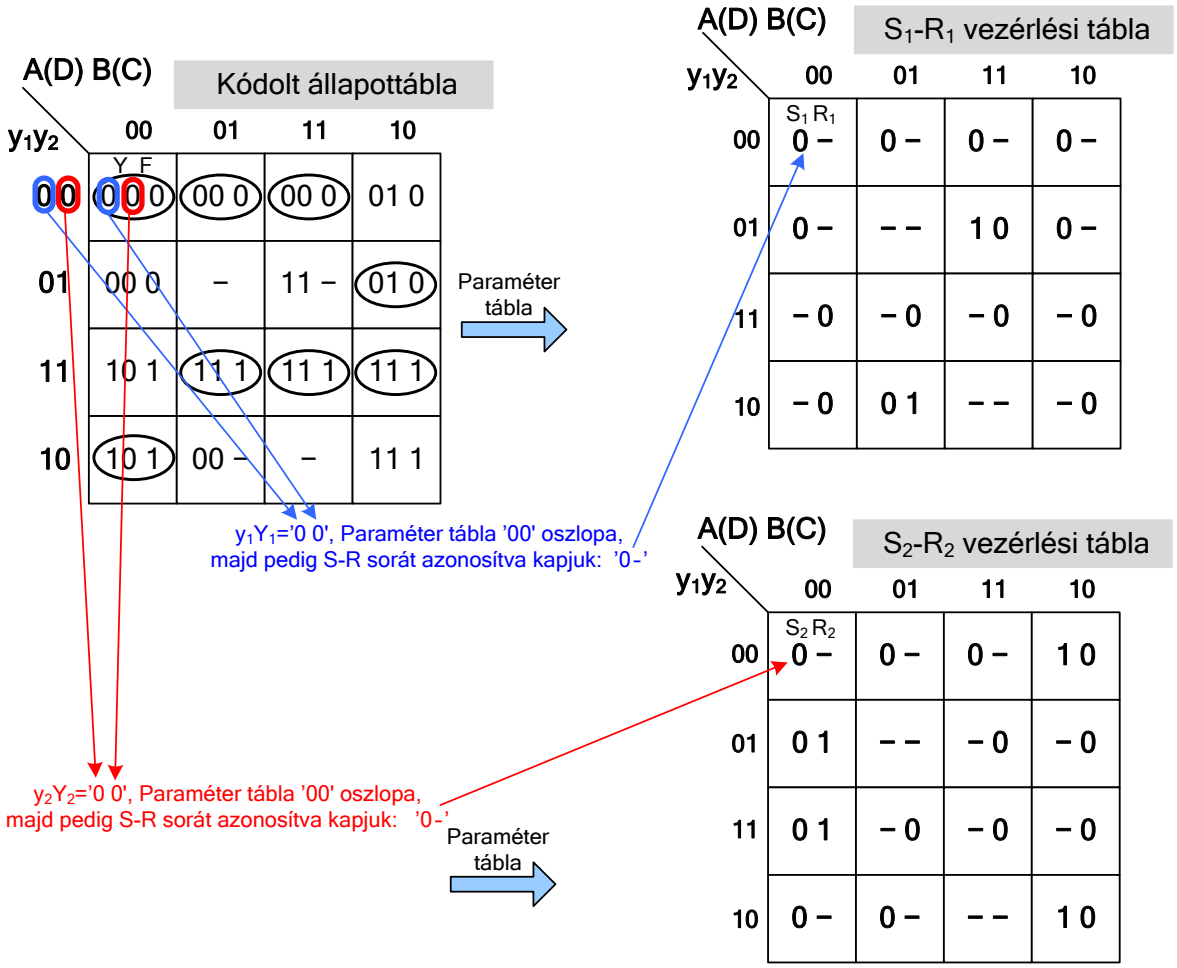

**5.9 ábra: Elvi logikai kapcsolási rajz tisztán visszacsatolással**

Moore: a felső $Y_1$ visszacsatoló ág aktuális állapot értéke direkt módon határozza meg a kimenetet ($F$), és semmi mástól nem függ. Ez alapján a Moore modell viselkedését kapjuk a külső szinkron D tárolót tekintve.

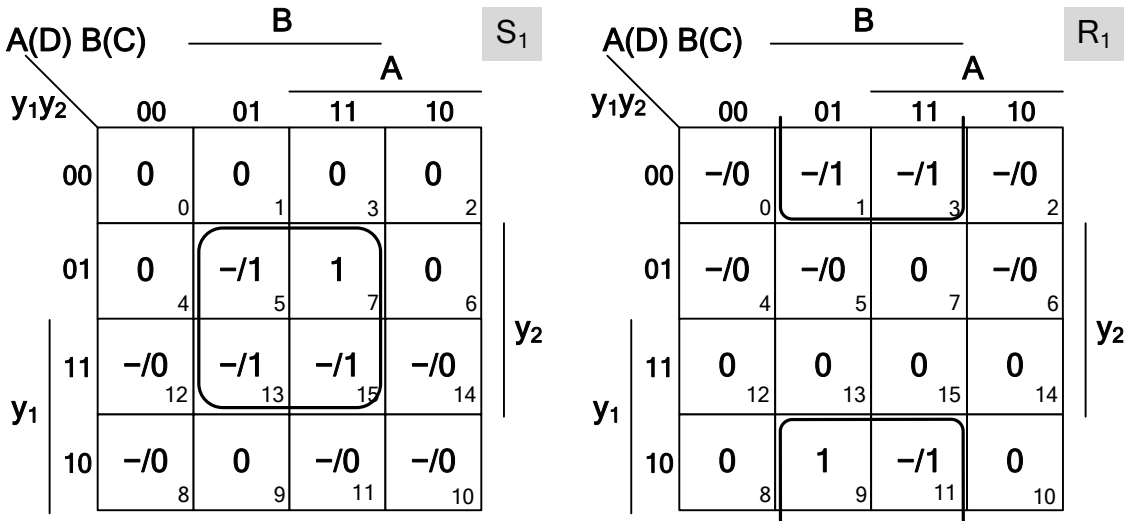
### b. Aszinkron tárolókból történő megvalósítás

Ez a másik lehetséges módszer (tisztán visszacsatolást használó eset mellett) arra, hogy aszinkron sorrendi hálózatot tervezzünk. Ebben az esetben egy megvalósító aszinkron FF-ot választunk a rendelkezésre álló építőelem készletből (pl. a tanult S-R, vagy D-G tárolókat). Ekkor a vezérlési tábla megadható a kódolt állapottáblából a tanult módon a „paraméter tábla” segítségével. Végül pedig a vezérlési táblából képzett Karnaugh táblákból kell az $Y_1$, $Y_2$ illetve az $F$-re vonatkozó függvényeket megvalósítani.

Vezérlési tábla felírása (szekunder kombinációk vizsgálata):

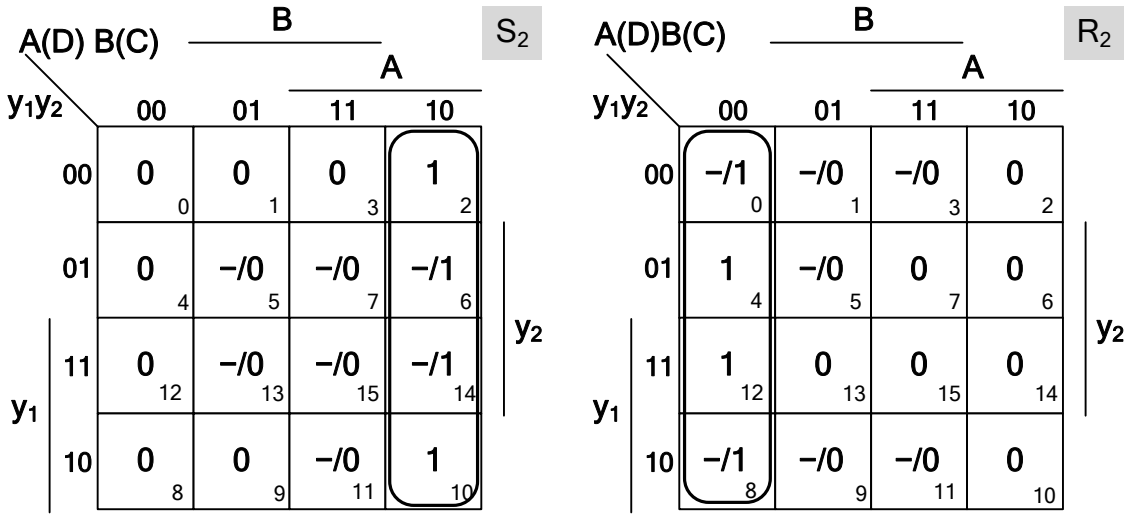



Vezérlési táblák alapján a Karnaugh táblák felírása külön $S_1\ R_1$-re, illetve $S_2\ R_2$-re:




$$S_1 = By_2 = C_{CLK}y_2$$

$$R_1 = B\overline{y_2} = C_{CLK}\overline{y_2}$$



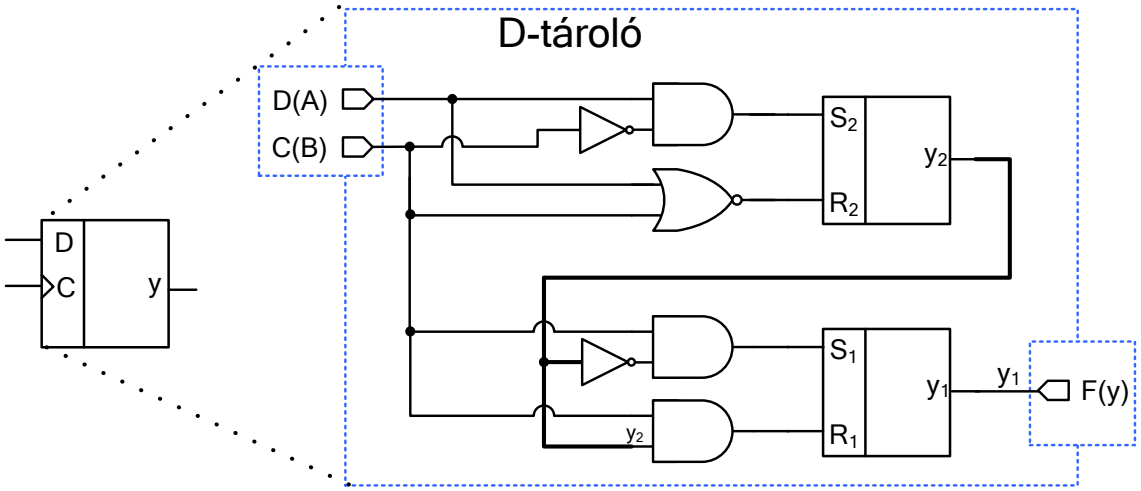

$$S_2 = A\overline{B} = D\overline{C_{CLK}}$$

$$R_2 = \overline{A} \cdot \overline{B} = \overline{D} \cdot \overline{C_{CLK}} = \overline{D + C_{CLK}}$$

Mint látható a statikus, dinamikus hazárd szempontjából hazárdmentes alakokat kaptunk a függvényekre.

A kimeneti függvény értéke nem változott a tisztán visszacsatolt esethez képest (Moore):

$$F = y_1$$




**5.10 ábra: Elvi logikai kapcsolási rajz S-R tárolókkal megvalósítva**

### g) „Lényeges hazárd" ellenőrzése

Az eddigi 1-6. tervezési lépések során megtervezett aszinkron sorrendi hálózatból felépülő szinkron D-tároló még korántsem biztos, hogy biztonságosan (hazárdmentesen) működik. Eddig mindig azt feltételeztük, hogy elsőként a bemeneti kombinációk megváltozása, majd pedig ennek megfelelően a szekunder kombinációk megváltozása jut érvényre. Azonban a sorrendi hálózat tervezésekor, ha nem gondoskodunk ezek késleltetéséről, illetve egymáshoz viszonyított sebesség-arányáról könnyen újabb hazárdjelenséget tapasztalhatunk. Ezt a hazárdjelenséget nevezzük lényeges hazárdnak.

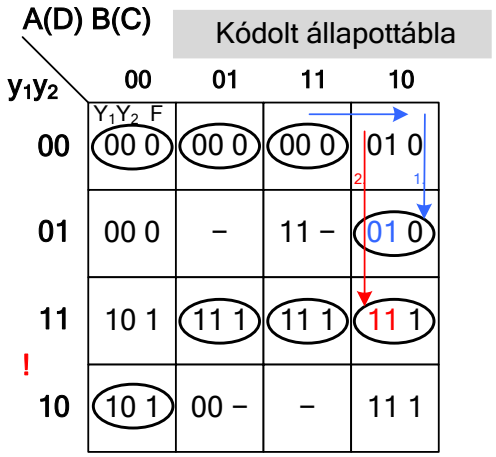
**Definíció: Lényeges hazárd**

Akkor alakulhat ki, ha egy aszinkron S.H. az összevont állapottáblán definiált működéstől eltérően viselkedik: a 'hibás' / 'helyes' működés attól függ, hogy a bemeneti ($X^i$) és a szekunder ($y^i$)

változások egymáshoz képest időben milyen sorrendben játszódnak le, illetve azok sebességaránya befolyásolja. Ha a sorrendi hálózat ettől az *eltérő sebességaránytól* függően *is hibás (nem az előírt) stabil állapotba* kerül – akár tetszőleges állapotkódolást választva – **lényeges hazárd alakul** ki.

**Lényeges hazárd kiküszöbölése:**

- Nem lehetséges az állapotkódolás változtatásával sem (nem úgy viselkedik, mint a kritikus versenyhelyzet).
- Lehetséges viszont a szekunder változók megfelelő késleltetésével: azaz a szekunder változók visszahatását ($y^i \Leftarrow Y^i$) késleltetjük, ahhoz hogy a bemeneti változás kerüljön érvényre először. Pl: a visszacsatoló ág(ak)ba páros számú INVERTER használatával.




Tfh: kiindulásként az $AB = 11$ mellett az $y_1y_2 = 00$ **stabil** állapotban van a rendszer (1.sor)

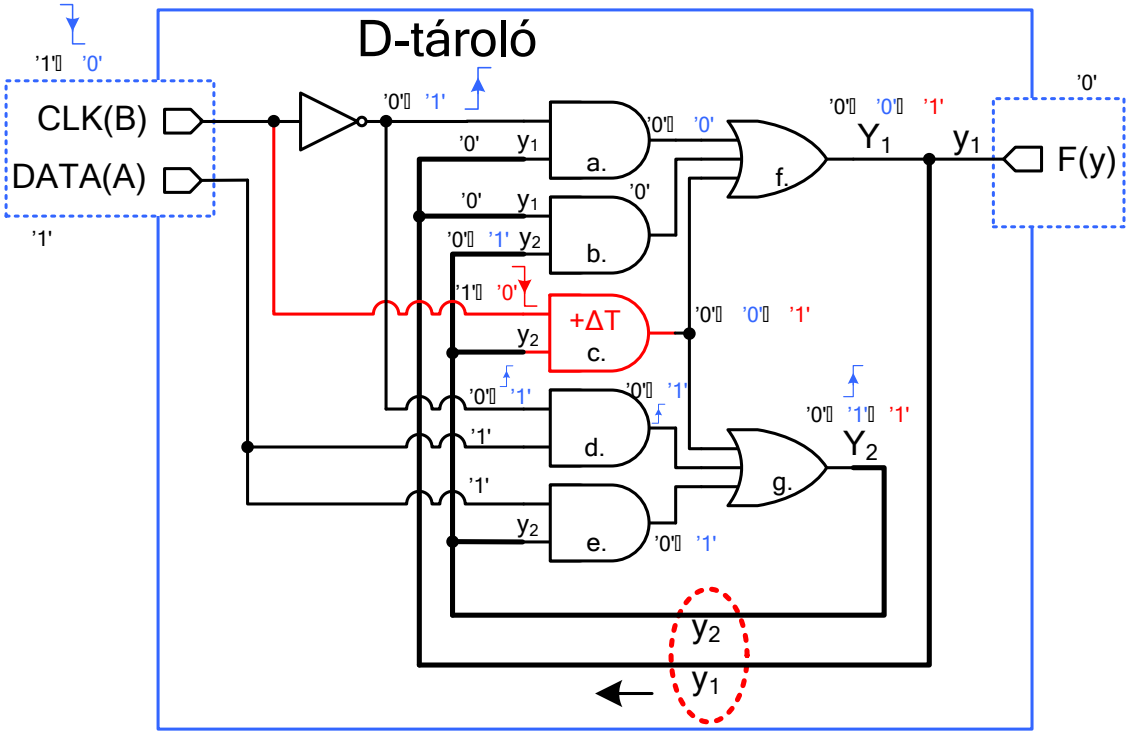
**1.)** Változzon $AB = 10$–ra. Az előírás szerint ennek az oszlopnak az $y_1y_2 = 01$ **helyes stabil** állapotába kell, hogy kerüljön a rendszer.

**2.)** De bizonyos késleltetési viszonyok miatt, akár ugyanezen $AB = 10$ oszlop $y_1y_2 = 11$ **hibás stabil** (stabil, de nem az előírt) állapotába is kerülhetünk

->>

**5.11 ábra: Lényeges hazárd vizsgálata állapottáblán**

A fenti állapottábla alapján most vizsgáljuk meg részletesebben a működést a megtervezett aszinkron hálózat elvi logikai kapcsolását használva (korábbi 5.9 ábra alapján!).




**5.12 ábra: Lényeges hazárd vizsgálata elvi kapcsolási rajzon**

Tegyük fel, hogy kiindulásként az $AB = 11$ (azaz D-CLK) mellett, hogy az $y_1 y_2 = 00$ **stabil** állapotban van a rendszer (1.sor az állapottáblán). Ekkor számítsuk ki milyen szekunder $Y_1 Y_2$ kombinációt kapunk. **Feketével** van jelölve a fenti kapcsolási rajzon. Látható, hogy az $Y_1 Y_2$ kombinációkra = '00' adódik, tehát stabil állapotban vagyunk.

1.) Tegyük fel, hogy az $AB$ bemeneti változásokra '11' → '10' van előírva. Ez azt is jelenti egyben, hogy míg $A$ (azaz $D$, mint Data) változatlanul '1' marad, addig a $B$ (azaz a $C$, mint CLK) bemenet '1'-ről '0'-ra változik. Ez a CLK-en egy lefutó élre történő változást jelent ( ↓ ).
2.) Továbbá azt is feltételezzük, hogy $B$ (CLK) bemenet '1'→ '0' változását először az inverteren keresztül a. és d.-jelű ÉS-kapuk észlelik (megj: további megkötésünk, hogy a c. jelű ÉS-kapu bemenetén van egy nagy $\Delta T$ idejű késleltetés, amelynek hatására csak később változzon a bemenet.). **Kékkel** vannak jelölve ezek a változások, melyeket a CLK ($B$) ilyetén változása okoz. Ekkor az is látható, ha végigkövetjük a kapukon keresztüli változásokat, hogy ennek hatása eljut a g-jelű VAGY-kapura is, és $Y_2 = 1$ jön létre, ami visszahatva $y_2 = 1$-et hoz létre pl. b. vagy d. jelű ÉS- kapuk bemenetén.
3.) Az $Y_2$ visszahatása után, ha pl. a c. jelű ÉS-kapu a felső bemeneten csak $\Delta T$ idejű késleltetés múlva érzékeli a $B$ (CLK) bemenet '1'→ '0' történő korábbi változását, akkor az f. jelű VAGY kapun keresztül $Y_1 = 1$ jön létre. Ez visszahatva $y_1 = 1$-et hoz létre pl. az a. jelű vagy b. jelű ÉS-kapuk bemenetén. Ez egyben azt is jelenti, hogy az $F = y_1$ kimenet értéke '1' lesz. **Pirossal** vannak jelölve a késleltetés utáni új állapot értékek.
4.) Mivel az $Y_2 = 1$-et az $AB = 10$ kombináció a d. jelű ÉS-kapun keresztül állandóan fenntartja, ezért sajnos kialakul a $y_1 y_2\ Y_1 Y_2 = 11\ 11$ **stabil, de hibás (nem a működés szerint előírt állapot).** Ez ugyan teljesen megfelel az állapottábla 3. sorának, de nem ezt volt a helyes működés.

A fenti komplex vizsgálat alapján arra a következtetésre juthatunk, hogy a sorrendi hálózatban lényeges hazárd kialakulása lehetséges. Ennek megszüntetéséről úgy gondoskodhatunk, hogy direkt módon páros számú invertert (vagy valamilyen más késleltető elemet) csatolunk a visszacsatoló $Y$ ágakba, méghozzá akkora jelterjedési késleltetésűt, hogy annak ideje nagyobb legyen, mint a c. jelű ÉS-kaput tartalmazó ágnak a késleltetési ideje!

### Összefoglaló táblázat:

Jelterjedési késleltetések hatása (hiba jelenségek) az *aszinkron sorrendi hálózatokban.*

| Jelenség oka | Jelenség elnevezése | Okozott hiba | Kiküszöbölés módja |
|---|---|---|---|
| A *bemeneti változók* (pl. X1,X2) közötti versenyhelyzet. (Nem szomszédos bemeneti kombináció változásokat tekintve) | **Funkcionális hazárd** | Előre *nem meghatározható* tranziens jelértékek lehetősége miatt a hálózat működése nem definiálható egyértelműen | Szomszédos bemeneti kombináció változások biztosítása, vagy szinkronizáció (feltételezve hogy szomszédos változás esetén statikus v. dinamikus hazárd ne alakuljon ki) |
| A *szekunder változók* (pl y1, y2) közötti versenyhelyzet | **Kritikus versenyhelyzet** | A szekunder változók *változási sebesség-arányától* függően hibás stabil állapot kialakulásának lehetősége | Az állapotkódolás megfelelő megválasztásával |
| A *bemeneti változók* (pl. X1,X2) **és** a *szekunder változók* (pl. y1,y2) közötti versenyhelyzet. | **Lényeges hazárd** | A bemeneti és a szekunder változók *változási sebesség-arányától* függően hibás stabil állapot kialakulásának lehetősége | A megfelelő visszacsatoló ágakba megfelelő késleltető elemek (pl. páros-számú inverter) beépítése |

**5.1 táblázat: Jelterjedési késleltetések hatása aszinkron sorrendi hálózatokban**

# *További feladatok*

### 5.1 feladat: Aszinkron sorrendi hálózat előzetes állapottáblájának felvétele

Kérdés: Egy olyan aszinkron S.H. hálózat tervezése a feladat, amely „kívülről" nézve egy szinkron T tárolóként viselkedik.

- A megvalósítandó szinkron tároló legyen egy *lefutó élre vezérelt T-FF.*
- Biztosítani kell a T-FF-re nézve a „Szinkronizációs feltételeket", azaz
  - CLK ($C$) órajelet tekintsük független bemeneti jelnek

Adjuk meg a következőket:

- T tároló ismert működése alapján a feladat megfogalmazását,
- az előzetes állapottáblát,
- az összevont állapottáblát, valamint az állapotgráfot is!

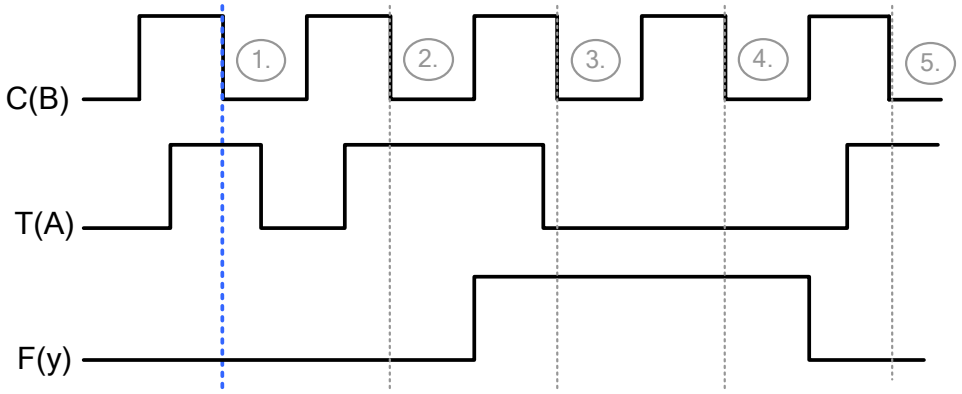
#### a) Logikai feladat megfogalmazása

**Működése**: tegyük fel, hogy

- a sorrendi hálózat $F$ kimenete csak akkor változik, amikor $C(A)$ értéke '1'→'0' (lefutó élre vált).
- Ekkor $F$ kimenet vegye fel a $T$ bemenetnek a mindenkori pillanatnyi *negált* értékét, $F = T(B)$ (ahogyan azt egy T tároló viselkedésétől elvárjuk).

Megjegyzés: a (zárójelben) megadott kifejezések a belső felhasznált aszinkron S.H. jeleit definiálják!

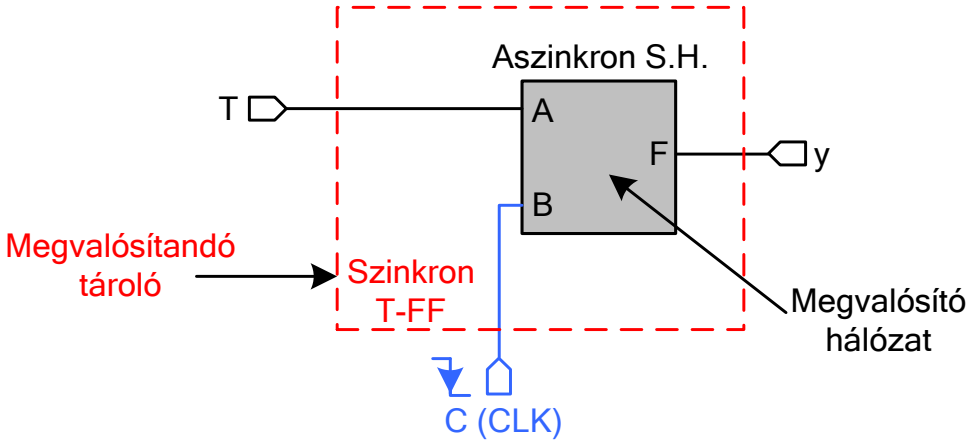
Az aszinkron hálózattal megvalósítható szinkron T-FF működéséhez szükséges jelek szemléltetése:




**5.1 ábra: aszinkron sorrendi hálózat jelei ($A$, $B$) vs. T-tároló jelei ($T$, $C$)**

Látható az idődiagramon, hogy a nem-szomszédos $C(B)$-$T(A)$ bemeneti kombináció-változásokat nem engedélyeztük! (funkcionális hazárd kiküszöbölése miatt)

A szinkron T-FF, aszinkron sorrendi hálózattal történő elvi megvalósítása látható az alábbi ábrán:




**5.2 ábra: szinkron T tároló felépítése aszinkron sorrendi hálózat segítségével**

### b) Előzetes állapottábla összeállítása

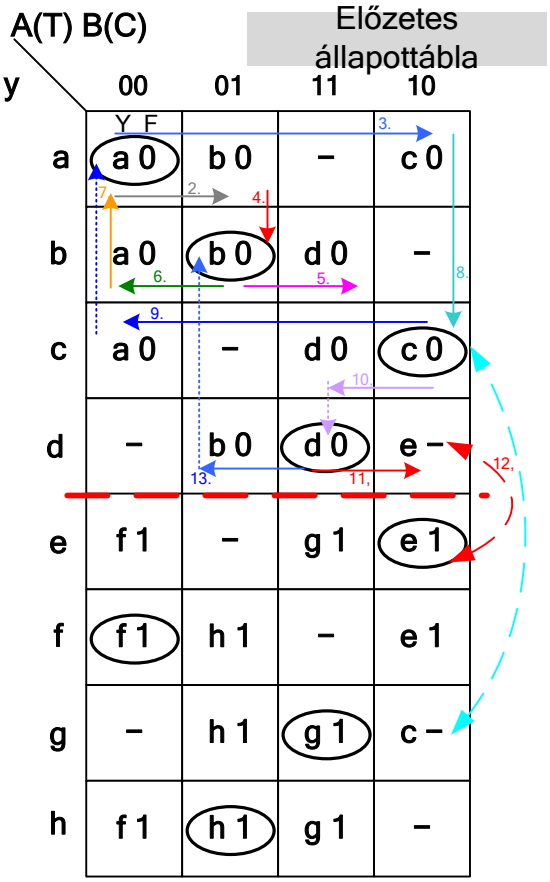
Használjuk fel az előzetes állapottábla összeállítása során tanult szabályokat.

**<u>Kitöltés</u>: *(1. sor)***

1. Tfh. $T(A);C(B)=00$ stabil **kiindulási állapot**, ahol: $y=a$ és $F=0$. Ekkor: $\boldsymbol{a0}$. (stabil $y \Leftarrow Y := a$). Ez azt jelenti, hogy $T(A);C(B)=10$ *közömbösnek* rögzítendő „–” (funkcionális hazárd mentesség).
2. Tfh: $T(A);C(B)=01$-re vált. (Ekkor $T=0$, $C=0\rightarrow 1$, tehát a kimenet pill. értéke $F=0$ marad változatlanul). Legyen $b$ instabil állapot. Ekkor: $b0$. ($y \neq Y = b$ instabil)
3. Tfh: $a0$ stabil állapotban vagyunk és $T(A);C(B)=10$-ra vált. (Ekkor $\boldsymbol{T}=0\rightarrow 1$, $C=0$, tehát a kimenet pill. értéke $F=0$ marad változatlanul). $c$ instabil állapot. Ekkor: $\boldsymbol{c0}$. ($y \neq Y = c$ instabil). //a 3. sor $c0$ stabil állapotába viheti át//

***(2. sor)***

4. Ha a 2. lépésben kialakuló $b0$ mint $Y \Rightarrow y$ *visszahat* a bemenetre, akkor a 2. sorba jutunk, ahol $\boldsymbol{b0}$ **stabil** állapot lesz ($T(A);C(B)=01$ ***és*** $y=b$). Ez azt jelenti, hogy $T(A);C(B)=10$ *közömbösnek* rögzítendő „–” (hazárd mentes).
5. Tfh: $b0$ stabil állapotban vagyunk és $T(A);C(B)=11$-re vált. (Ekkor $\boldsymbol{T}=0\rightarrow 1$, $C=1$, tehát a kimenet pill. értéke $F=0$ marad változatlanul). $d$ instabil állapot. Ekkor: $\boldsymbol{d0}$. ($y \neq Y = d$ instabil).
6. Tfh: $b0$ stabil állapotban vagyunk és $T(A);C(B)=00$-ra vált. (Ekkor $T=0$, $\boldsymbol{C}=1\rightarrow 0$, tehát a kimenet pill. értéke $F=0$ marad változatlanul). $a$ instabil állapot. Ekkor: $\boldsymbol{a0}$. ($y \neq Y = a$ instabil).
7. Az $a0$ instabil állapotból (2. sor) az $a0$ stabil állapotba kerülhetünk vissza (1. sorba) >>



***(3. sor)***

8. Ha a 8. lépésben kialakuló $c0$ mint $Y => y$ *visszahat* a bemenetre, akkor a 3. sorba jutunk, ahol $\boldsymbol{c0}$ **stabil** állapot lesz ($T(A); C(B) = 10$ ***és*** $y = c$). Ez azt jelenti, hogy $T(A); C(B) = 01$ *közömbösnek* rögzítendő „–" (hazárd mentes). Megj: $c0$ stabil állapotba az állapottábla alsó részéből kerülhetünk át ($c$-).
9. Tfh: $c0$ stabil állapotban vagyunk és $T(A); C(B) = 00$-re vált. (Ekkor $T = 1 \rightarrow 0$, $C = 0$, tehát a kimenet pill. értéke $F = 0$ marad változatlanul). $a$ instabil állapot. Ekkor: $\boldsymbol{a0}$. ($y \neq Y = a \neq c$ instabil). Innen az $a0$ stabil állapotba érdemes visszakerülnünk (1. sorba)
10. Hasonlóan $c$ stabil állapotban és $T(A); C(B) = 11$-re váltás esetén ($T = 1 \rightarrow 1$, $C = 0 \rightarrow 1$, tehát a kimenet pill. értéke $F = 0$ marad változatlanul). $d$ instabil állapot. Ekkor: $\boldsymbol{d0}$. ($y \neq Y = d \neq c$ instabil). Innen később az $d0$ stabil állapotba érdemes átkerülni (4. sorba)

***(4. sor)***

11. Tfh: a 4. sorban $d0$ stabil állapotban vagyunk azaz $y = d$, és legyen $T(A); C(B) = 10$-ra váltás. (Ekkor $\boldsymbol{T = 1, C = 1 \rightarrow 0}$ **!!!**, tehát a kimenet pill. értéke pont ekkor kell, hogy megváltozzon az ellentettjére). Az '11' oszlopon belül olyan stabil állapotba kell jutni, amelyhez $F = 1$ kimeneti érték tartozik.
12. Ezt jelölje „$\boldsymbol{e}$**"**. ***Kimenetként*** *azért írhatunk* ***közömbös*** *'– ' értéket*, mivel a $d0$ stabil állapotból $\boldsymbol{e}$ - instabil állapoton keresztül $e1$ stabil állapotba kerülhet a rendszer, ahol a *kimeneti érték megváltozik* ($\boldsymbol{d0 \rightarrow e1}$!). Ilyen esetekben, amikor a stabil–instabil állapothoz tartozó kimeneti érték párok eltérőek, *közömbös kimeneti értéket* kell rögzíteni a köztes instabil állapothoz. Eközben d0 stabil állapot azt jelenti, $T(A); C(B) = 00$ *közömbös „–"* (hazárd mentes).
13. Tfh: $d0$ stabil állapotban vagyunk és $T(A); C(B) = 01$-re vált. (Ekkor $\boldsymbol{T = 1 \rightarrow 0}$, $C = 1$, tehát a kimenet pill. értéke $F = 0$ marad változatlanul). $a$ instabil állapot. Ekkor: $\boldsymbol{b0}$. ($y \neq Y = c$ instabil). Innen akár az $b0$ stabil állapotba is visszakerülhetünk (2, sorba)
14. És így tovább…

**Következmény:**

**Két eset van, amikor a kimenet $\boldsymbol{F}$ értéke megváltozik:**

I. Tfh. $\boldsymbol{d}$ stabil állapot és $T(A)C(B) = 11 \rightarrow 10$, mivel ekkor $C = 1 \rightarrow 0$, azaz $T = 1$ értékének negáltja kellene, hogy kerüljön az $F$ kimenetre, tehát $F = 0$. Ezt technikailag az állapottáblán csak úgy oldhatjuk meg, hogy felveszünk egy köztes állapotot ($e$) bizonytalan don't care kimenet mellett! Ez fog átvezetni minket az „alsó táblázatrészre", ahol a kimenetek '1' –esek (vagy esetleg közömbösek) lehetnek.

II. Tfh. $\boldsymbol{g}$ stabil állapotban vagyunk és $T(A)C(B) = 11 \rightarrow 10$, mivel ekkor $C = 1 \rightarrow 0$, azaz $T = 1$ értékének negáltja kellene, hogy kerüljön az $F$ kimenetre, tehát $F = 0$. Ezt technikailag az állapottáblán csak úgy oldhatjuk, hogy felveszünk egy olyan köztes állapotot ($c$) bizonytalan don't care kimenet mellett, amely átvezet minket a „felső táblázatrészre", ahol a kimenetek '0'-ások (vagy esetleg közömbösek).

Megjegyzés: Aszinkron esetben akár a Moore modell is teljesülhet a közömbös kimenetek helyes megválasztása mellett.

### c) Összevont (egyszerűsített) állapottábla és állapotgráf

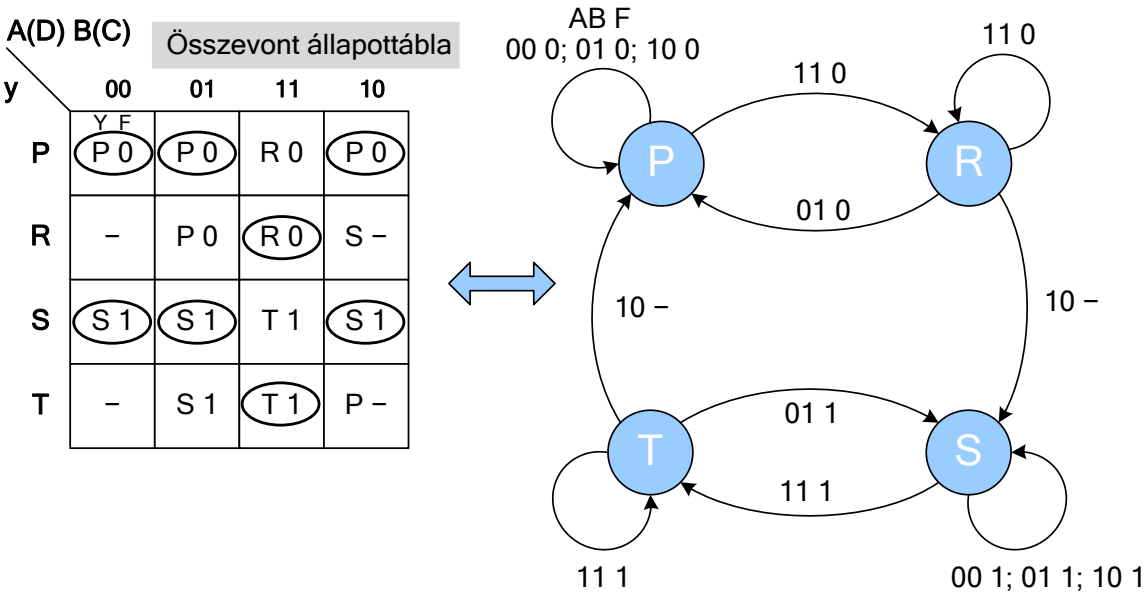
Előzetes állapottábla alapján az a cél, hogy megkeressük a lehető legkevesebb megkülönböztetendő állapotot. Ezek ismeretében kell létrehozni a lehető legkevesebb, de azonos sort nem tartalmazó új állapottáblát, amelyet összevont állapottáblának nevezünk, amelynek így már nem lesz feleslegesen megkülönböztetett állapota (=sora). Fontos tudni, hogy az összevont állapottábla nem azonos a később ismertetésre kerülő állapot kódolási módszerrel!

A fenti előzetes állapottábla alapján egy lehetséges összevonás a következő:

Megkülönböztetendő szekunder állapotok

- $P \coloneqq a, b, c$ *(1.-,2.-, és 3. sorok azonosak)*
- $R \coloneqq d$ *(4. sor)*
- $S \coloneqq e, f, h$ *(5.-,6.- és 8. sorok azonosak)*
- $T \coloneqq g$ *(7. sor)*

Az összevonás során kapott összevont állapottábla és a vele ekvivalens állapotgráf a következő:




**5.3 ábra: Összevont állapottábla és állapotgráf**

Megjegyzés: az előzetes állapottáblán, illetve az összevont állapottáblán is, ahogy várható, minden oszlopban rögzítve lett stabil állapot, ami biztosítja a nem oszcilláló (normál aszinkron) működést.

### d) Kódolt állapottábla

Az összevont állapottábla alapján a szimbolikus összevonásként ($A$, $B$) jelölt állapotoknak meg kell feleltetni egy-egy szekunder kombinációt ($y$).

Az összevont állapottábla sorainak (=állapotainak) számától függ, hogy mennyi különböző szekunder állapotot kell felvenni.

n sor esetén: $\lceil \log_2(n) \rceil$ szekunder állapot szükséges.

**Választott állapotkódok**: *a feladat szempontjából, valamint a hazárdjelenségek kiküszöbölése miatt egyáltalán nem közömbös, hogy miként rögzítjük az állapotkódokat.* Jelen példában, az egyszerűség kedvéért az állapotkódokat még direkt módon adtuk meg, azonban a későbbi 8. és 9. fejezetekben részletesen ismertetjük az állapotkódolás módszereit.

# 6. Teljesen specifikált sorrendi hálózatok állapotminimalizálása

Egy sorrendi hálózat bonyolultsága két tényezőtől függ. Az állapotgráfban lévő állapotok számától függ, hogy hány darab tárolóra lesz szükség a megvalósításhoz valamint az állapotokhoz rendelt állapotkódok határozzák meg, hogy a tárolókat vezérlő kombinációs hálózat hány darab kaput tartalmaz.

Első lépésben foglalkozzunk az első tényezővel, azaz próbáljuk meg minimalizálni az állapotok számát. Erre azért van szükség, mert a specifikáció alapján felrajzolt „előzetes" állapotgráf nem biztos, hogy minimális számú állapotot tartalmaz.

Jelen fejezetben az úgynevezett teljesen specifikált sorrendi hálózatok állapotminimalizálásával foglakozunk, a nem teljesen specifikált hálózatokról a következő fejezetben lesz szó. A hálózat teljesen specifikáltnak nevezünk, ha minden lehetséges bemenetre és belső állapotra egyértelműen adott, hogy mi lesz a következő állapot és a kimenet. Azaz az állapottáblában nincs határozatlan állapot és kimenet.

Illusztratív példa

A fejezetben a módszerek működésének bemutatásához a következő illusztratív példát használjuk. Legyen egy sorrendi hálózatnak egy $X$-szel jelölt bemenete, egy $Y$-nal jelölt kimenete és tegyük fel, hogy a feladatkiírás alapján a következő előzetes állapottáblát határoztuk meg.

| X / Y | 0 | 1 |
|---|---|---|
| *A* | *C* 0 | *E* 1 |
| *B* | *E* 1 | *D* 0 |
| *C* | *A* 0 | *B* 1 |
| *D* | *D* 0 | *A* 1 |
| *E* | *B* 1 | *D* 0 |

**6.1 ábra: Előzetes állapottábla TSH alakban**

## *Ekvivalens állapotok*

A későbbiekben bemutatandó módszerek nem arra adnak módszert, hogy a specifikációból hogyan lehet minimális állapotszámú állapottáblát felépíteni, hanem arra, hogy egy meglévő állapottáblában hogyan lehet az állapotok számát csökkenteni. Az állapotok számát úgy csökkentjük, hogy állapotokat összevonunk új állapottá. Ezek után a kérdés az, hogy mikor lehet két állapotot összevonni.

„Ránézésre" akkor lehet két állapotot összevonni, ha bármely bemenet hatására a hálózat ugyanolyan állapotba kerül és ugyanaz lesz a kimenete, azaz ha az állapottáblában a két állapot sora megegyezik. Ez az eset elég ritkán fordul elő, de a módszer általánosításával ennél gyakran több állapotot is össze lehet vonni, ezért az összevonásra egy pontosabb feltételt akarunk adni.

Egy hálózatban $A$ és $B$ állapotot „nem-megkülönböztethető"-nek nevezzük, ha a hálózat kimenetén ugyanazt a jelsorozatot kapjuk ugyanarra a bemeneti jelsorozatra függetlenül attól, hogy a hálózat $A$ vagy $B$ állapotból indult. A minimalizálás során a nem-megkülönböztethető állapotokat természetesen össze lehet vonni. Ez a definíció annyival pontosabb, mint a „ránézésre" való összehasonlítás, hogy nem veszi figyelembe, hogy a hálózat a futása során milyen állapotokat vesz

fel, hanem kizárólag a kimenet egyformaságát követeli meg. Ez egy teljesen logikus definíció, mivel a hálózat felhasználása esetén teljesen irreleváns, hogy milyen állapotban van a rendszer, csak a kimeneti jelértékek a fontosak.

Hasonlóképen tudjuk azt is definiálni, hogy egy hálózat két állapota mikor megkülönböztethető. Egy hálózat $A$ és $B$ állapota akkor megkülönböztethető, ha létezik olyan bemeneti jelsorozat, amelyre a kimeneti jelsorozat különbözik $A$-ból vagy $B$-ből indulva.

A nem-megkülönböztethető állapotokat teljesen specifikált hálózatok esetén ekvivalens állapotoknak is nevezzük. A fenti explicit definíció alapján azonban nehéz meghatározni az ekvivalens állapotokat, mivel nem vizsgálhatjuk meg a hálózat összes lehetséges lefutását, ezért egy rekurzív definíciót is megadunk: Egy automata két állapota ekvivalens, ha mindkét állapotban ugyanarra a bemenetre ugyanazt a kimenetet kapjuk, valamint mind a két állapotban ugyanarra a bementre a követező állapotok ekvivalensek. Ha $A$ állapot ekvivalens $B$ állapottal, akkor azok ekvivalencia relációban vannak egymással, melyet jelölése $A \equiv B$.

A megkülönböztethető állapotokat teljesen specifikált hálózatok esetén antivalens állapotoknak nevezzük. Ha két állapot nem ekvivalens, akkor antivalens, ennek jelölése $A <> B$.

Minden ekvivalencia relációra teljesülnek a reflexivitás, a tranzitivitás és a szimmetria tulajdonságok, azaz a most bevezetett ekvivalencia reláció tulajdonságai:

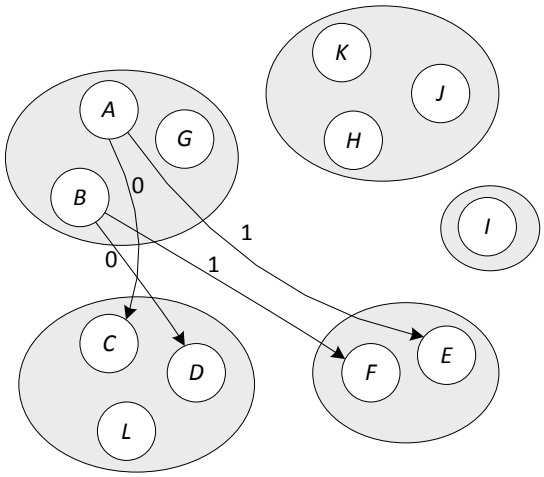
- Reflexív: $A \equiv A$
- Tranzitív: $A \equiv B$ és $B \equiv C \leftrightarrow A \equiv C$
- Szimmetrikus: $A \equiv B \leftrightarrow B \equiv A$

Minden ekvivalencia reláció segítségével a halmazt, amin értelmezzük partícionálni lehet, azaz az általunk bevezetett ekvivalencia reláció az állapotok halmazát diszjunkt részhalmazokra bontja. Az egymással páronként ekvivalens állapotok halmazát ekvivalencia osztálynak nevezzük. Egy ekvivalencia osztály maximális, ha nincs több olyan állapot, amely nincs benne az osztályban és ekvivalens lenne bármely az osztályban lévő állapottal (a tranzitivitás miatt az összes állapottal).

Célunk a maximális ekvivalencia osztályok meghatározása. Mivel minden ekvivalencia osztály egy állapottá összevonható, ezért ha a maximális ekvivalencia osztályokat meghatározzuk ezek jelentik az új állapotokat.

Példa

Az alábbi 6.2 ábrán egy hálózat állapotait ($A, \dots, L$) körökkel és öt maximális ekvivalencia osztályát szürke ellipszisekkel jelöltük. Továbbá élek mutatják, hogy $A$ és $B$ állapotokból $0$ és $1$ bemenetek hatására milyen állapotokba jutunk. Látható, hogy $0$ hatására $A$-ból $C$ állapotba, $B$-ből pedig $D$ állapotba kerülünk, melyek ugyanazon ekvivalencia osztályba tartoznak, valamint $1$ hatására $A$-ból $E$-be, $B$-ből $F$-be kerül a kapcsolás, melyek szintén ekvivalens állapotok. A 6.2 ábrán nincsenek jelölve az automata kimeneti értékei, de természetesen azok is páronként megegyeznek.




**6.2 ábra: Állapotok ekvivalencia osztályok szerinti csoportosítása**

## *Ekvivalencia osztályok előállítása*

Az ekvivalencia osztályok meghatározásához az ekvivalens párokat kell felismerni, melyek egyértelműen megadják a maximális ekvivalencia osztályokat. Az ekvivalencia osztályok meghatározására két módszert mutatunk be, az első a lépcsős táblát, a második partíció finomítást használja.

### Lépcsős tábla

Első lépésben meghatározzuk az ekvivalens állapotpárokat majd ezek alapján generáljuk a maximális ekvivalencia osztályokat.

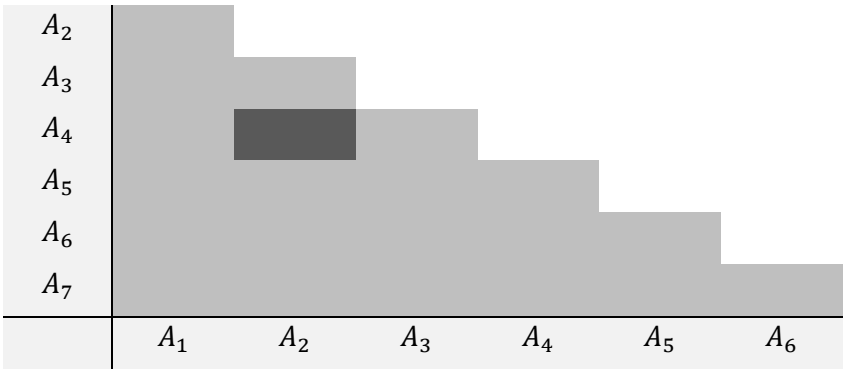
#### *Ekvivalens állapotpárok meghatározása*

Az állapottábla vizsgálata alapján két állapot ekvivalenciájának teljesüléséről a következő állításokat tehetjük.

- Két állapotról „ránézésre” meg tudjuk állapítani, hogy ekvivalensek, ha a két állapothoz tartozó sorok a táblázatban megegyeznek.
- Két állapotról „ránézésre” meg tudjuk állapítani, hogy antivalensek, ha ugyanazon bemenet különböző kimenetet eredményez.
- Ha mind a két állapotra ugyanazok a kimenetek, de a következő állapotok különböznek, akkor nem tudjuk „ránézésre” eldönteni, hogy ekvivalensek vagy antivalensek. Ilyenkor azt mondjuk, hogy a két állapot feltételesen ekvivalens és feljegyezzük, hogy mik az ekvivalencia feltételei, azaz mely állapotoknak kell ekvivalensnek lenniük ahhoz, hogy a két eredeti állapot is ekvivalens legyen. A feltételes ekvivalenciát a feltételekkel jelöljük. Ha például két állapotra feljegyeztük, hogy $(AB, CD)$, akkor ezek az állapotok feltételesen ekvivalensek, és csak akkor ekvivalensek, ha $A \equiv B$ és $C \equiv D$ teljesülnek.

Amennyiben egy bináris reláció értékeit meg akarjuk adni egy $n$ elemű halmazon, akkor azt egy $n \times n$ méretű táblázat segítségével tehetjük meg, ahol az $i$-edik sor $j$-edik eleme megadja, hogy a reláció teljesül vagy nem teljesül, valamint a teljesülés feltételeit tartalmazhatja. Abban az esetben, ha a reláció szimmetrikus, akkor nincs szükség az egész táblázatra, mivel ebben az esetben az $(i, j)$ elem megegyezik a $(j, i)$ elemmel. Ekvivalencia reláció esetében a fő átlóra sincs szükség, mivel minden elem ekvivalens önmagával, azaz $(i, i)$ mindig igaz. Ezek alapján a táblázatot az átló mentén elfelezzük és jellegzetes alakja alapján a továbbiakban lépcsős táblának nevezzük.

Például az alábbi táblázat egy 7 elemű halmaz esetén mutatja a lépcsős táblázatot, ahol az $(A_4, A_2)$ párra vonatkozó bejegyzést tartalmazó cellát sötétszürkével jelöltük. Természetesen ugyanez a cella vonatkozik az $(A_2, A_4)$ párra is.




**6.3 ábra: Lépcsős tábla (kiindulási)**

Egy sorrendi hálózat ekvivalencia relációjához tartozó lépcsős táblát az alábbiak szerint tudunk felírni. Az ekvivalencia reláció értékét minden állapotpárra meg kell adni, ezért akkora táblázatot kell felrajzolni, ahány állapota van az előzetes állapottáblának. A cellák kitöltése a következőképen történik.

- Ha az állapotpárra kimenetek és a következő állapotok is azonosak minden bemeneti kombinációra, akkor az aktuális cellában jelöljük az ekvivalenciát.
- Ha a kimeneti értékek legalább egy bemeneti kombinációra különböznek, akkor antivalencia jelölés kerül a cellába.
- Ha a kimeneti értékek megegyeznek, de a következő állapotok eltérnek valamely bemenetre, akkor feltételes bejegyzés kerül a tábla megfelelő cellájába.

Illusztratív példa

Használjuk a fejezet elején definiált illusztratív példát, melynek az állapottáblája lent látható. Írjuk fel az (6.4 ábra) állapottáblához a lépcsős táblát. Ehhez páronként meg kell vizsgálni, hogy az állapotok ekvivalensek, antivalensek vagy feltételesen ekvivalensek.

| X / Y | 0 | 1 |
|---|---|---|
| *A* | *C* 0 | *E* 1 |
| *B* | *E* 1 | *D* 0 |
| *C* | *A* 0 | *B* 1 |
| *D* | *D* 0 | *A* 1 |
| *E* | *B* 1 | *D* 0 |

**6.4 ábra: Előzetes állapottábla illusztratív példa alapján**

- $AB$ pár antivalens, mivel a kimenet különbözik $X = 0$ és $X = 1$ esetében is.
- $AC$ pár feltételesen ekvivalens, mivel a kimenetek megegyeznek, de a következő állapotok nem azonosak (viszont ekvivalensek még lehetnek). Azaz $A$ és $C$ akkor ekvivalens, ha $X = 0$ eset miatt $C$ és $A$ ekvivalens, valamint $X = 1$ eset miatt $E$ és $B$ ekvivalens. Az $AC$ párt nem írjuk be a feltételek közé, mivel önmagának lenne feltétele, ezt tautológiának nevezzük. Így a táblázat $AC$ cellájában az $EB$ pár szerepel, mint ekvivalencia feltétel.
- $AD$ pár az előző ponthoz hasonlóan feltételesen ekvivalens, a feltétel pedig $CD$ és $AE$.
- $AE$ pár antivalens, mivel a kimenetek különböznek.
- $BC$ pár antivalens.
- $BD$ pár antivalens.

- $BE$ pár ekvivalens, mivel a kimenetek megegyeznek, valamint $X = 1$ esetén a következő állapotok is megegyeznek és $X = 0$ esetén a tautológia lép fel, azaz $B$ akkor lenne ekvivalens $E$-vel, ha $E$ ekvivalens lenne $B$-vel.
- $CD$ pár feltételesen ekvivalens, ahol a feltételek $AD$ és $AB$.
- $CE$ pár antivalens.
- $DE$ pár antivalens.

| | | | | |
|---|---|---|---|---|
| $B$ | <> | | | |
| $C$ | $EB$ | <> | | |
| $D$ | $CD, AE$ | <> | $AD, AB$ | |
| $E$ | <> | ≡ | <> | <> |
| | $A$ | $B$ | $C$ | $D$ |

**6.5 ábra: Első Lépcsős tábla**

Mint a példából is látszik, a lépcsős tábla még nem nyújt elég információt ahhoz, hogy belőle közvetlenül fel tudjuk írni az ekvivalencia osztályokat, mivel nem tudjuk minden állapotpárra, hogy azok ekvivalensek vagy antivalensek. Következő lépésben ebből az úgynevezett első lépcsős táblából készítünk egy második lépcsős táblát úgy, hogy kihasználjuk az ekvivalencia reláció tranzitivitás tulajdonságát.

A második lépcsős táblában az antivalencia bejegyzések hatását vezetjük végig a cellákon. Ha két állapot egymással antivalens, akkor az összes olyan cellában, ahol ennek a párnak az ekvivalenciája a feltétele szerepel ekvivalencia feltételként, azok is antivalensek lesznek. Példánkban az $AB$ pár antivalens és a $CD$ feltételesen ekvivalens pár ekvivalencia feltételei között szerepel az $AB$ pár, így a $CD$ pár is antivalens lesz.

Az algoritmus során az összes antivalencia bejegyzés hatását érvényesíteni kell, azokat is, amelyek más antivalencia bejegyzésből keletkeztek. Azaz addig kell ismételgetni az eljárást, amíg érvényesítetlen antivalencia bejegyzés van a táblázatban, vagy minden állapotpárról ki nem derül, hogy ekvivalens vagy antivalens. Abban az esetben, ha minden antivalencia bejegyzés érvényesítve lett és vannak feltételesen ekvivalens állapotok, akkor azok ekvivalensek lesznek. Ebben az esetben ugyanis a feltételes ekvivalencia feltételei vagy ekvivalens vagy feltételesen ekvivalens állapotok. Ha ekvivalens állapot a feltétel, akkor egyértelmű, hogy a pár is ekvivalens. Ha feltételesen ekvivalens feltételünk van, akkor ezek egymás feltételei.

Illusztratív példa

Az illusztratív példában előállított első lépcsős táblából (a korábbi 6.5 ábra) készítsük el a második lépcsős táblát. A továbbiakban sötétszürke alapon fehér karakterekkel írt cellák segítségével jelöljük azokat az antivalencia bejegyzéseket, melyek hatását már érvényesítettük. Az antivalencia bejegyzések vizsgálatának sorrendje tetszőleges, mi haladjunk oszloponként balról jobbra és az oszlopokon belül pedig fentről lefelé. Így első lépében válasszuk a következő táblázatból (6.6 ábra) az $AB$ antivalens párt.

Mivel a $CD$ pár ekvivalenciájának egyik feltétele az $AB$ ekvivalenciája, de ez antivalens, ezért a $CD$ pár is antivalens lesz. Ez alapján a táblázat az alábbiak szerint módosul. A következő megvizsgálandó antivalens pár az $AE$ pár lesz.

| | | | | |
|---|---|---|---|---|
| $B$ | <> | | | |
| $C$ | $EB$ | <> | | |
| $D$ | $CD, AE$ | <> | <> | |
| $E$ | <> | ≡ | <> | <> |

| | $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|---|

**6.6 ábra: Második lépcsős tábla készítése 1.**

Az $AC$ pár ekvivalenciájának egyik feltétele az $AE$ ekvivalenciája, de ez antivalens, ezért az $AC$ pár is antivalens lesz. Ez alapján a táblázat az alábbiak szerint módosul. A következő megvizsgálandó antivalens pár a $BC$ pár lesz.

| | | | | |
|---|---|---|---|---|
| $B$ | <> | | | |
| $C$ | $EB$ | <> | | |
| $D$ | <> | <> | <> | |
| $E$ | <> | ≡ | <> | <> |
| | $A$ | $B$ | $C$ | $D$ |

**6.7 ábra: Második lépcsős tábla készítése 2.**

A BC pár nem szerepel egyetlen feltételes ekvivalencia feltételei között sem, ezért a táblázatban nem lesz új antivalencia bejegyzés, csak a bejegyzés érvényesítését jelöljük a táblázatban, mint ahogy az alábbiakban látszik.

| | | | | |
|---|---|---|---|---|
| $B$ | <> | | | |
| $C$ | $EB$ | <> | | |
| $D$ | <> | <> | <> | |
| $E$ | <> | ≡ | <> | <> |
| | $A$ | $B$ | $C$ | $D$ |

**6.8 ábra: Második lépcsős tábla készítése 3.**

Ezek után folyamatosan haladva végignézzük az összes antivalencia bejegyzést és azt látjuk, hogy egyik bejegyzés sem generál újabb antivalencia relációt. Ezt az alábbi táblázatban láthatjuk is.

| | | | | |
|---|---|---|---|---|
| $B$ | <> | | | |
| $C$ | $EB$ | <> | | |
| $D$ | <> | <> | <> | |
| $E$ | <> | ≡ | <> | <> |
| | $A$ | $B$ | $C$ | $D$ |

**6.9 ábra: Második lépcsős tábla készítése 4.**

Azt is észrevehetjük, hogy az antivalencia bejegyzések érvényesítése során keletkezett egy olyan új antivalencia bejegyzés, amit a bejárásunk alapján nem vizsgáltunk. Ez példánkban az $AD$ pár antivalencia bejegyzése. Ez a pár sem szerepel egy feltételes valószínűség feltételei között sem, ezért ezt is érvényesítettnek tekinthetjük.

| | | | | |
|---|---|---|---|---|
| $B$ | <> | | | |
| $C$ | $EB$ | <> | | |
| $D$ | <> | <> | <> | |
| $E$ | <> | ≡ | <> | <> |
| | $A$ | $B$ | $C$ | $D$ |

**6.10 ábra: Második lépcsős tábla készítése 5.**

Ha ezt is megtettük, akkor már nem marad több érvényesítetlen antivalencia bejegyzés. Ekkor az összes olyan párt, amelynek a cellájában nincs antivalencia bejegyzés ekvivalensnek tekintünk. Eszerint az $AC$ és a $BE$ párok celláiba szerepel ekvivalencia bejegyzés.

| | | | | |
|---|---|---|---|---|
| $B$ | <> | | | |
| $C$ | ≡ | <> | | |
| $D$ | <> | <> | <> | |
| $E$ | <> | ≡ | <> | <> |
| | $A$ | $B$ | $C$ | $D$ |

**6.11 ábra: Második lépcsős tábla**

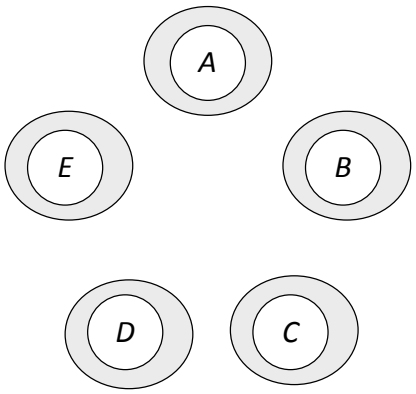
A második lépcsős táblából (6.11 ábra) kiolvashatók az ekvivalens állapotpárok, amelyek jelen esetben $AC$ és $BE$. Az ekvivalens állapotpárok segítségével a maximális ekvivalencia osztályokat meg lehet határozni.

### *Ekvivalencia osztályok generálása*

Kezdetben legyen annyi osztály, ahány állapota van a hálózatnak és minden osztályban pontosan egy állapot szerepeljen. Minden lépésben tekintsünk egy ekvivalens állapotpárt, és ha a két állapot külön osztályban szerepel, akkor azokat összevonjuk. Ezt addig ismételjük, amíg az összes állapotpárt le nem ellenőriztük.

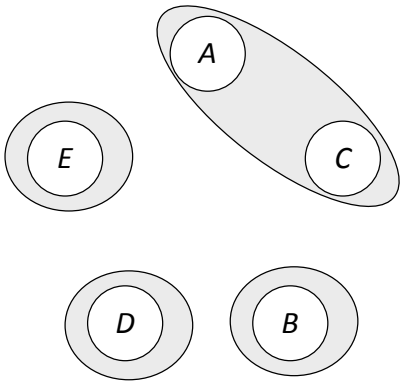
Illusztratív példa

Kezdetben minden állapot külön osztályban van. Mivel öt állapotunk van, ezért öt osztályt veszünk fel. Az alábbi ábrában fehér körökkel vannak jelölve az állapotok és szürke oválissal az egy osztályba lévő állapotok.



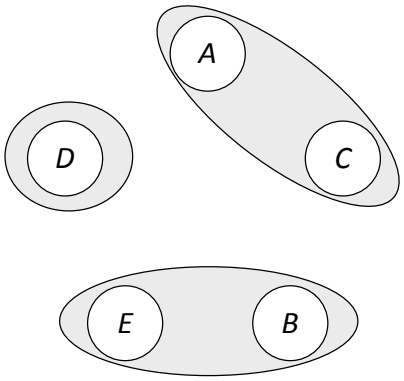

**6.12 ábra: Minden állapot külön osztály**

A második lépcsős táblázat alapján az ekvivalens állapotpárok az $AC$ és $BE$. Először tekintsük az $AC$ párt. Ez alapján összevonható az a két osztály, amelyben $A$ valamint $C$ szerepel. Az összevonás eredményét az alábbi 6.13-as ábrán láthatjuk.




**6.13 ábra: Ekvivalens állapotpárok összevonása**

Ezután a $BE$ állapotpár segítségével végezzük el az összevonást. Ez alapján ismét összevonható két osztály. Mivel több ekvivalens állapotpár nincs, ezért az alábbi 6.14-es ábra megadja a maximális ekvivalencia osztályokat, melyek az $A \equiv C$, $B \equiv E$ és $D$. Ezek alapján össze lehet vonni az $A$ és $C$ állapotokat valamint az $E$ és a $B$ állapotokat, azaz öt állapot helyett három állapot is elegendő a hálózat megvalósításához.




**6.14 ábra: Ekvivalens osztályok a második lépcsős tábla alapján (6.5 ábra)**

### Partíció finomítás

A partíció finomítás módszeréhez nem határozzuk meg az ekvivalens párokat, hogy azokból építsük fel az ekvivalencia osztályokat, hanem a másik irányból próbáljuk közvetlenül meghatározni őket. Azaz feltételezzük, hogy a különböző állapotok azonos ekvivalencia osztályban vannak és megvizsgáljuk, hogy teljesül-e rájuk az ekvivalencia feltételei. Ha nem teljesülnek, akkor a partíciót (partíciókat) szétbontjuk kisebb partíciókra. Ezt addig ismételjük, amíg a feltételek nem teljesülnek.

Két állapot ekvivalenciájának teljesülésére két feltételét fogalmaztuk meg.

- Mind a két állapotban ugyanolyan bemenet ugyanolyan kimenetet eredményez.
- Az állapotokból ugyanazon bemenet hatására ekvivalens állapotokba jutunk.

Az első feltétel teljesülését úgy ellenőrizzük, hogy a kezdeti partíciókat ez alapján generáljuk. Azaz amely állapotokról ránézésre meg lehet állapítani, hogy antivalensek, azok különböző partíciókba kerülnek. Ezt a feltételt a későbbiek során nem kell többet ellenőrizni, mert már a keresés elején teljesül minden azonos partícióba tartozó állapotpárra és minden későbbi partíció ezen partíciók részhalmaza lesz, azaz új „párok” nem keletkeznek.

Most vizsgáljuk meg a második feltétel teljesülését. Tegyük fel, hogy valamilyen partíciók kialakításra kerültek. Minden partícióban minden állapotra megvizsgáljuk, hogy ugyanolyan bemenet hatására milyen partíciókba jutnak. Ha minden állapotra igaz, hogy ugyanabba a partícióba jutnak ugyanolyan

bemenet hatására (azaz teljesül a második feltétel) akkor azok egy partícióban maradhatnak. Ha különböznek az elért partíciók, akkor az állapotokat ez alapján osztjuk új partícióba.

Az újonnan kialakított partíciókon újra meg kell vizsgálni a második feltétel teljesülését, mivel a szétosztás után előfordulhat, hogy már nem teljesül. Például, ha az előző lépésben két állapotból ugyanabba a partícióba jutottunk, de azon belül különböző állapotokba, akkor előfordulhat, hogy az új partíciók esetén ezek az állapotok már nincsenek egy partícióban, mert szétosztottuk az eredetileg őket tartalmazó partíciót is.

Illusztratív példa

Az alábbi előzetes állapottábla alapján először készítsük el a kiinduló partíciókat.

| X / Y | 0 | 1 |
|---|---|---|
| *A* | *C* 0 | *E* 1 |
| *B* | *E* 1 | *D* 0 |
| *C* | *A* 0 | *B* 1 |
| *D* | *D* 0 | *A* 1 |
| *E* | *B* 1 | *D* 0 |

**6.15 ábra: Illusztratív példa előzetes állapottáblája (adott)**

Az állapottáblát megvizsgálva megállapíthatjuk, hogy két partícióba oszthatjuk az állapotokat a kimenet alapján. Az egyik partícióban lesznek azok az állapotok, melyek $X = 0$ hatására $0$ és $X = 1$ hatására $1$ kimenetet adnak ($A$, $C$ és $D$ állapotok). A másik partícióba tartozó állapotok ($B$ és $E$ állapotok) $X = 0$ hatására $1$, $X = 1$ hatására pedig $0$ kimenetet generálnak. Az első partíciót 0-val, a másodikat 1-el jelöljük. Ezek után minden állapotról megnézzük, hogy milyen bement hatására milyen partícióba kerül.

- $A$ állapotból $X = 0$ hatására $C$ állapotba, azaz a 0-s partícióba, $X = 1$ hatására $E$ állapotba, azaz az 1-es partícióba kerülünk.
- $B$ állapotból $X = 0$ hatására $E$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $D$ állapotba, azaz a 0-s partícióba kerülünk.
- $C$ állapotból $X = 0$ hatására $A$ állapotba, azaz a 0-s partícióba, $X = 1$ hatására $B$ állapotba, azaz az 1-es partícióba kerülünk.
- $D$ állapotból $X = 0$ hatására $D$ állapotba, azaz a 0-s partícióba, $X = 1$ hatására $A$ állapotba, azaz a 0-s partícióba kerülünk.
- $E$ állapotból $X = 0$ hatására $B$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $D$ állapotba, azaz a 0-s partícióba kerülünk.

Az eredményeket egy táblázatba (6.1) feljegyezzük, mint az alant látható.

| | 0<br>$ACD$ | 1<br>$BE$ |
|---|---|---|
| $X = 0$ | 000 | 11 |
| $X = 1$ | 110 | 00 |

**6.1 táblázat Partíció finomítás első lépés**

Látható, hogy a 0-s partícióból $X = 1$ hatására két különböző partícióba is eljuthatunk. Ez megsérti az ekvivalencia második feltételét, ezért a 0-s partíciót két részre osztjuk aszerint, hogy az állapotokból milyen partícióba juthatunk. Ez alapján lesz egy $AC$ és egy $D$ partíció, valamint megmarad a $BE$

partíció is. A partíciókat újraszámozzuk, az $AC$ partíció lesz a 0-s, a $D$ partíció az 1-es és a $BE$ partíció a 2-es. Ismét megvizsgáljuk, hogy különböző bemenetek hatására melyik állapotból milyen partícióba jutunk és az eredményeket egy táblázatba jegyezzük fel.

- $A$ állapotból $X = 0$ hatására $C$ állapotba, azaz a 0-s partícióba, $X = 1$ hatására $E$ állapotba, azaz a 2-es partícióba kerülünk.
- $B$ állapotból $X = 0$ hatására $E$ állapotba, azaz az 2-es partícióba, $X = 1$ hatására $D$ állapotba, azaz az 1-es partícióba kerülünk.
- $C$ állapotból $X = 0$ hatására $A$ állapotba, azaz a 0-s partícióba, $X = 1$ hatására $B$ állapotba, azaz a 2-es partícióba kerülünk.
- $D$ állapotból $X = 0$ hatására $D$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $A$ állapotba, azaz a 0-s partícióba kerülünk.
- $E$ állapotból $X = 0$ hatására $B$ állapotba, azaz a 2-es partícióba, $X = 1$ hatására $D$ állapotba, azaz az 1-es partícióba kerülünk.

| | 0<br>$AC$ | 1<br>$D$ | 2<br>$BE$ |
|---|---|---|---|
| $X = 0$ | 00 | 1 | 22 |
| $X = 1$ | 22 | 0 | 11 |

**6.2 táblázat Partíció finomítás második lépés**

Mivel a 6.2 táblázat minden cellájában csak egyféle partíciószám látható, ezért minden partícióra igaz az, hogy ugyanarra a bemenetre ugyanabba a partícióba kerül. Ezzel ki is alakultak az ekvivalencia osztályok, melyek gyakorlatilag megegyeznek a partíciókkal. Így a lépcsős tábla használatához hasonlóan az $A \equiv C$, $B \equiv E$, $D$ partíciós osztályokat kaptuk.

Ahogy az illusztratív példából is látszik a partíció finomítás módszere kevesebb munkával jár, mint a lépcsős tábla használata. Ez a módszer viszont (a lépcsős táblával ellentétben) ebben a formájában kizárólag teljesen specifikált hálózatok állapotminimalizálására használható.

## *Összevont állapottábla*

A maximális ekvivalencia osztályok alapján elvégezhetjük az állapotok összevonását az állapottáblában. Minden ekvivalencia osztályt egy állapotnak tekintünk. Mivel az ekvivalenciának az első feltétele az volt, hogy azonos bement hatására azonos kimenet generálódjon, ezért a kimenet meghatározásához elegendő az ekvivalencia osztály egyetlen állapotát megnézni az előzetes állapottáblában és az ott szereplő kimenetet beírni az összevont állapottáblába. Az ekvivalencia második feltétele az volt, hogy ugyanazon bemenet hatására ugyanazon ekvivalencia osztályba kerüljünk, ezért ehhez is elegendő egy állapotot tekinteni az osztályból és megvizsgálni, hogy milyen bemenet hatására melyik ekvivalencia osztályba kerül és az annak megfelelő új állapotot beírni a táblázatba.

Illusztratív példa

Induljunk ki az előzetes állapottáblából és a korábban meghatározott ekvivalencia osztályokból.

| X / Y | 0 | 1 |
|---|---|---|
| *A* | *C* 0 | *E* 1 |
| *B* | *E* 1 | *D* 0 |
| *C* | *A* 0 | *B* 1 |
| *D* | *D* 0 | *A* 1 |
| *E* | *B* 1 | *D* 0 |

**6.16 ábra Illusztratív példa előzetes állapottáblája (adott)**

Mindegyik ekvivalencia osztályhoz rendeljünk egy-egy állapotot. Legyen az $AC$ ekvivalencia osztályhoz tartozó állapot $S1$-el, a $BE$ osztályhoz tartozó állapot $S2$-vel és a $D$-hez tartozó $S3$-mal jelölve. A teljesség kedvéért vizsgáljuk meg mindegyik ekvivalencia osztályban mindegyik állapotot, hogy lássuk, valóban tetszőlegesen ki lehet egyet választani közülük.

- $S1$ ekvivalencia osztály
  - $A$ állapotból $X = 0$ hatására $C$ állapotba, azaz az $S1$ partícióba kerülünk és a kimenet $0$, $X = 1$ hatására $E$ állapotba, azaz az $S2$ partícióba kerülünk és a kimenet $1$.
  - $C$ állapotból $X = 0$ hatására $A$ állapotba, azaz az $S1$ partícióba kerülünk és a kimenet $0$, $X = 1$ hatására $B$ állapotba, azaz az $S2$ partícióba kerülünk és a kimenet $1$.
- $S2$ ekvivalencia osztály
  - $B$ állapotból $X = 0$ hatására $E$ állapotba, azaz az $S2$ partícióba kerülünk és a kimenet $1$, $X = 1$ hatására $D$ állapotba, azaz az $S3$ partícióba kerülünk és a kimenet $0$.
  - $E$ állapotból $X = 0$ hatására B állapotba, azaz az $S2$ partícióba kerülünk és a kimenet $1$, $X = 1$ hatására $D$ állapotba, azaz az $S3$ partícióba kerülünk és a kimenet $0$.
- $S3$ ekvivalencia osztály
  - $D$ állapotból $X = 0$ hatására $D$ állapotba, azaz az $S3$ partícióba kerülünk és a kimenet $0$, $X = 1$ hatására $A$ állapotba, azaz az $S1$ partícióba kerülünk és a kimenet $1$.

Ezek alapján az összevont állapottábla a következő.

| X / Y | 0 | 1 |
|---|---|---|
| *S*1 | *S*1 0 | *S*2 1 |
| *S*2 | *S*2 1 | *S*3 0 |
| *S*3 | *S*3 0 | *S*1 1 |

**6.17 ábra: Összevont állapottábla**

Az előzetes állapottábla öt állapotot tartalmazott, így annak a realizálásához három állapotváltozó, azaz három tároló lett volna szükséges. Az összevont állapottábla már csak három állapotot tartalmaz, azaz a hálózat kettő tároló segítségével megvalósítható.

## *További példák:*

### 6.1 Példa

Tekintsük a következő előzetes állapottáblát (6.18 ábra) és határozzuk meg a maximális ekvivalencia osztályokat lépcsős tábla és partíciófinomítás segítségével is. Az ekvivalencia osztályokból pedig írjuk fel az összevont állapottáblát.

| X / Y | 0 | 1 |
|---|---|---|
| A | F 0 | H 1 |
| B | F 0 | C 0 |
| C | A 0 | B 0 |
| D | H 0 | C 1 |
| E | A 0 | C 0 |
| F | A 0 | D 1 |
| G | H 0 | B 1 |
| H | D 0 | B 1 |

**6.18 ábra: 6.1 példa előzetes állapottáblája**

Az első lépcsős tábla előállításához páronként meg kell vizsgálni, hogy az állapotok ekvivalensek, antivalensek vagy feltételesen ekvivalensek.

- $AB$ pár antivalens, mivel a kimenet különbözik $X = 1$ esetén.
- $AC$ pár antivalens hasonló okok miatt.
- $AD$ pár feltételesen ekvivalens, mivel a kimenetek megegyeznek, de a következő állapotok nem azonosak (viszont ekvivalensek még lehetnek). Azaz $A$ és $C$ akkor ekvivalens, $F$ és $H$, valamint $H$ és $C$ ekvivalensek, azaz a táblázat $AD$ cellájában az $FH$ és a $CH$ pár szerepel, mint ekvivalencia feltétel.
- $AE$ pár antivalens.
- $AF$ pár feltételesen ekvivalens, az ekvivalencia feltétele $DH$. Az $X = 0$ bemenetből következő $AF$ feltétel tautológia, nem írjuk be a táblázatba.
- $AG$ pár feltételesen ekvivalens, az ekvivalencia feltétele $FH$ és $BH$.
- $AH$ pár feltételesen ekvivalens, az ekvivalencia feltétele $FD$ és $BH$.
- $BC$ pár feltételesen ekvivalens, az ekvivalencia feltétele $AF$.
- $BD$ pár antivalens.
- $BE$ pár feltételesen ekvivalens, az ekvivalencia feltétele $AF$.
- $BF$ pár antivalens.
- $BG$ pár antivalens.
- $BH$ pár antivalens.
- $CD$ pár antivalens.
- $CE$ pár feltételesen ekvivalens, az ekvivalencia feltétele $BC$.
- $CF$ pár antivalens.
- $CG$ pár antivalens.
- $CH$ pár antivalens.

- $DE$ pár antivalens.
- $DF$ pár feltételesen ekvivalens, az ekvivalencia feltétele $AH$ és $CD$.
- $DG$ pár feltételesen ekvivalens, az ekvivalencia feltétele $BC$.
- $DH$ pár feltételesen ekvivalens, az ekvivalencia feltétele $BC$.
- $EF$ pár antivalens.
- $EG$ pár antivalens.
- $EH$ pár antivalens.
- $FG$ pár feltételesen ekvivalens, az ekvivalencia feltétele $AH$ és $BD$.
- $FH$ pár feltételesen ekvivalens, az ekvivalencia feltétele $AD$ és $BD$.
- $GH$ pár feltételesen ekvivalens, az ekvivalencia feltétele $HD$.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $B$ | <> | | | | | | |
| $C$ | <> | $AF$ | | | | | |
| $D$ | $FH, CH$ | <> | <> | | | | |
| $E$ | <> | $AF$ | $BC$ | <> | | | |
| $F$ | $DH$ | <> | <> | $AH, CD$ | <> | | |
| $G$ | $FH, BH$ | <> | <> | $BC$ | <> | $AH, BD$ | |
| $H$ | $FD, BH$ | <> | <> | $BC$ | <> | $AH, BD$ | $HD$ |
| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $G$ |

**6.19 ábra: Első lépcsős tábla (6.1 példa)**

Az első lépcsős táblából (6.19 ábra) az antivalencia kapcsolatok hatásának végigkövetésével elő lehet állítani a második lépcsős táblát, amely már csak ekvivalencia és antivalencia bejegyzéseket fog tartalmazni. Az állapottáblából felírt antivalencia bejegyzések hatása a következő.

- $BD$ antivalenciájából következik $FG$ és $FH$ antivalenciája.
- $BH$ antivalenciájából következik $AG$ és $AH$ antivalenciája.
- $CD$ antivalenciájából következik $DF$ antivalenciája.
- $CH$ antivalenciájából következik $AD$ antivalenciája.

Az új antivalencia bejegyzések hatását is meg kell vizsgálni, de példánkban ezek nem okoznak újabb antivalencia bejegyzéseket. Azokba a cellákba, ahova nem került antivalencia bejegyzés ekvivalencia bejegyzést írunk és így alakul ki a második lépcsős táblánk (6.20 ábra).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $B$ | <> | | | | | | |
| $C$ | <> | ≡ | | | | | |
| $D$ | <> | <> | <> | | | | |
| $E$ | <> | ≡ | ≡ | <> | | | |
| $F$ | ≡ | <> | <> | <> | <> | | |
| $G$ | <> | <> | <> | ≡ | <> | <> | |
| $H$ | <> | <> | <> | ≡ | <> | <> | ≡ |
| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $G$ |

**6.20 ábra: Második lépcsős tábla (6.1 példa)**

A második lépcsős táblából (6.20 ábra) kiolvashatók az ekvivalens párok: $A \equiv F$, $B \equiv C$, $B \equiv E$, $C \equiv E$, $D \equiv G$, $D \equiv H$, $G \equiv H$. Az ekvivalens párok pedig egyértelműen meghatározzák a maximális ekvivalencia osztályokat: $(AF)(BCE)(DGH)$.

Határozzuk meg a maximális ekvivalencia osztályokat a partíciófinomítás módszerével. Az állapottáblát megvizsgálva megállapíthatjuk, hogy két partícióba oszthatjuk az állapotokat a kimenet alapján. Az egyik partícióban lesznek azok az állapotok, melyek $X = 0$ és $X = 1$ hatására is $0$ kimenetet adnak ($B$, $C$ és $E$ állapotok). A másik partícióba tartozó állapotok ($A$, $D$, $F$, $G$ és $H$ állapotok) $X = 0$ hatására $0$, $X = 1$ hatására pedig $1$ kimenetet generálnak. Az első partíciót 0-val, a másodikat 1-el jelöljük. Ezek után minden állapotról megnézzük, hogy milyen bement hatására milyen partícióba kerül.

- $A$ állapotból $X = 0$ hatására $F$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $H$ állapotba, azaz az 1-es partícióba kerülünk.
- $B$ állapotból $X = 0$ hatására $F$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $C$ állapotba, azaz a 0-s partícióba kerülünk.
- $C$ állapotból $X = 0$ hatására $A$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $B$ állapotba, azaz a 0-s partícióba kerülünk.
- $D$ állapotból $X = 0$ hatására $H$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $C$ állapotba, azaz a 0-s partícióba kerülünk.
- $E$ állapotból $X = 0$ hatására $A$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $C$ állapotba, azaz a 0-s partícióba kerülünk.
- $F$ állapotból $X = 0$ hatására $A$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $D$ állapotba, azaz az 1-es partícióba kerülünk.
- $G$ állapotból $X = 0$ hatására $H$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $B$ állapotba, azaz a 0-s partícióba kerülünk.
- $H$ állapotból $X = 0$ hatására $D$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $B$ állapotba, azaz a 0-s partícióba kerülünk.

Az eredményeket a 6.3 táblázatba feljegyezzük, mint az alant látható.

| | 0<br>$BCE$ | 1<br>$ADFGH$ |
|---|---|---|
| $X = 0$ | 111 | 11111 |
| $X = 1$ | 000 | 10100 |

**6.3 táblázat Partíció finomítás első lépés (6.1 példa)**

Látható, hogy az 1-es partícióból $X = 1$ hatására két különböző partícióba is eljuthatunk. Ez megsérti az ekvivalencia második feltételét, ezért az 1-es partíciót két részre osztjuk aszerint, hogy az állapotokból milyen partícióba juthatunk. Ez alapján lesz egy $AF$ és egy $DGH$ partíció, valamint megmarad a $BCE$ partíció is. A partíciókat újraszámozzuk, a $BCE$ partíció lesz a 0-s, az $AF$ partíció az 1-es és a $DGH$ partíció a 2-es. Ismét megvizsgáljuk, hogy különböző bemenetek hatására melyik állapotból milyen partícióba jutunk és az eredményeket egy következő, 6.4-es táblázatba jegyezzük fel.

- $A$ állapotból $X = 0$ hatására $F$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $H$ állapotba, azaz a 2-es partícióba kerülünk.
- $B$ állapotból $X = 0$ hatására $F$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $C$ állapotba, azaz a 0-s partícióba kerülünk.
- $C$ állapotból $X = 0$ hatására $A$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $B$ állapotba, azaz a 0-s partícióba kerülünk.
- $D$ állapotból $X = 0$ hatására $H$ állapotba, azaz a 2-es partícióba, $X = 1$ hatására $C$ állapotba, azaz a 0-s partícióba kerülünk.
- $E$ állapotból $X = 0$ hatására $A$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $C$ állapotba, azaz a 0-s partícióba kerülünk.
- $F$ állapotból $X = 0$ hatására $A$ állapotba, azaz az 1-es partícióba, $X = 1$ hatására $D$ állapotba, azaz a 2-es partícióba kerülünk.

- $G$ állapotból $X = 0$ hatására $H$ állapotba, azaz a 2-es partícióba, $X = 1$ hatására $B$ állapotba, azaz a 0-s partícióba kerülünk.
- $H$ állapotból $X = 0$ hatására $D$ állapotba, azaz az 2-es partícióba, $X = 1$ hatására $B$ állapotba, azaz a 0-s partícióba kerülünk.

| | 0<br>$BCE$ | 1<br>$AF$ | 2<br>$DGH$ |
|---|---|---|---|
| $X = 0$ | 111 | 11 | 222 |
| $X = 1$ | 000 | 22 | 000 |

**6.4 táblázat Partíció finomítás második lépés (6.1 példa)**

Mivel a fenti 6.4 táblázat minden cellájában csak egyféle partíciószám látható, ezért minden partícióra igaz az, hogy ugyanarra a bemenetre ugyanabba a partícióba kerül. Ezzel ki is alakultak a maximális ekvivalencia osztályok, melyek gyakorlatilag megegyeznek a partíciókkal. Így a lépcsős tábla használatához hasonlóan az $(AF)(BCE)(DGH)$ partíciós osztályokat kaptuk.

Végül írjuk fel az összevont állapottáblát, melyben az $(AF)$ ekvivalencia osztályt jelölje $S1$, a $(BCE)$ osztályt $S2$ és a $(DGH)$ osztályt $S3$ állapot.

| X / Y | **0** | **1** |
|---|---|---|
| ***S*1** | ***S*1 0** | ***S*3 1** |
| ***S*2** | ***S*1 0** | ***S*2 0** |
| ***S*3** | ***S*3 0** | ***S*1 0** |

**6.21 ábra Összevont állapottábla (6.1 példa)**

Az állapotok számának minimalizálásával az állapotok száma az eredeti nyolcról háromra csökkenthető volt. Ha az előzetes állapottábla segítségével valósítottuk volna meg a hálózatunkat, akkor három állapotváltozóra lett volna szükségünk, amelyet három tároló segítségével lehetett volna megvalósítani. A minimalizált esetben a három állapot kettő állapotváltozó segítségével kódolható, azaz kettő tároló elegendő a megvalósításhoz.

# 7. Nem teljesen specifikált sorrendi hálózatok állapotminimalizálása

Jelen fejezetben az úgynevezett nem teljesen specifikált sorrendi hálózatok (NTSH) állapotminimalizálásával foglakozunk. A hálózat nem teljesen specifikáltnak nevezünk, ha az előzetes állapottáblában nem minden állapotátmenete és/vagy kimenete van specifikálva. Azaz az állapottáblában van legalább egy határozatlan állapot és/vagy kimenet. A megtervezett áramkör tetszőleges kimenetet adhat és/vagy tetszőleges állapotba léphet valamely bemeneti kombinációra. Normál működés során a nem teljesen specifikált hálózat nem kaphat olyan bemeneti sorozatot, amelynek hatására nem specifikált állapotba lépne, hiszen innentől kezdve nem specifikált a működése.

Az NTSH hálózatok állapotminimalizálása NP-teljes feladat, így egzakt megvalósítása nagyon nagy számítási igényű, ezért erre a gyakorlatban heurisztikus módszereket alkalmaznak. Egy feladathoz több azonos állapotszámú megoldás is létezhet és nem biztos, hogy a későbbiekben bemutatandó módszerek megtalálják az optimális megoldást.

Illusztratív példa

A fejezetben a módszerek működésének bemutatásához a következő illusztratív példát használjuk. Legyen egy sorrendi hálózatnak egy $X$-szel jelölt bemenete, egy $Y$-nal jelölt kimenete és tegyük fel, hogy a feladatkiírás alapján a következő (7.1 ábra) előzetes állapottáblát határoztuk meg.

| X / Y | 0 | 1 |
|---|---|---|
| *A* | *A* 0 | *E* 0 |
| *B* | - 1 | *D* - |
| *C* | *A* - | *B* 0 |
| *D* | *C* - | - - |
| *E* | *C* 1 | - 0 |

**7.1 ábra: Előzetes állapottábla NTSH alakban**

## *Kompatibilis állapotok*

A teljesen specifikált hálózatokhoz hasonlóan itt is olyan állapotpárokat keresünk, amelyek összevonhatóak egy közös állapottá és így az állapotok száma csökkenthető. Itt is megpróbálhatjuk „ránézésre” megállapítani két állapotról, hogy összevonhatóak-e, de ugyanúgy nehézségekbe ütközünk, mint teljesen specifikált hálózatok esetén.

„Ránézésre” akkor összevonható két állapot, ha az állapottáblában a hozzájuk tartozó sorok megegyeznek. NTSH esetében az is problémát okoz, hogy egy nem specifikált állapot/kimenet azonos-e egy specifikált állapottal/kimenettel. Nem mondhatjuk azt sem, hogy egy nem specifikált állapot/kimenet sosem azonos egy specifikálttal, és azt sem, hogy mindig azonos, mert a megvalósítástól függ, hogy a nem specifikált állapot/kimenet milyen értéket fog felvenni.

Ahhoz, hogy szisztematikusan elő tudjuk állítani az összevonható állapotpárokat itt is definiáljuk a „nem-megkülönböztethető” állapotokat, de most a megkülönböztethetőséget definiáljuk. Két szimbolikus állapot az NTSH állapottábláján csak akkor megkülönböztethető, ha létezik legalább egy olyan bemeneti sorozat, amely mindkét állapotra specifikált és legalább egy elemére más kimeneti kombináció adódik. Ha ilyen bemeneti sorozat nem létezik, a két állapot nem-megkülönböztethető.

Ez a definíció annyiban különbözik a TSH esettől, hogy itt a bemeneti sorozatokra azokon a helyeken kell a kimenetnek megegyeznie, ahol az specifikálva van. Itt is arra fektettük a hangsúlyt, hogy a belső állapotok nem számítanak, csak a kimenet, de az is csak akkor, amikor specifikálva van.

A nem-megkülönböztethető állapotokat nem teljesen specifikált hálózatok esetén kompatibilis állapotoknak is nevezzük. Az ekvivalens állapotok rekurzív definíciójához hasonló módon egy automata két állapota akkor kompatibilis egymással, ha mindkét állapotban ugyanarra a bementre ugyanazt a kimenetet kapjuk, valamint mind a két állapotban ugyanarra a bemenetre a következő állapotok kompatibilisek. A nem specifikált kimenetet azonosnak tekintjük bármely kimenettel, továbbá a nem specifikált állapot kompatibilis minden állapottal. Ha $A$ állapot kompatibilis $B$ állapottal, akkor annak jelölése $A \sim B$. Ha $A$ állapot nem kompatibilis $B$ állapottal, azaz inkompatibilisek, akkor annak jelölése $A \backslash\sim B$.

A kompatibilitás reláció tulajdonságai:

- Reflexív: $A \sim A$
- Szimmetrikus: $A \sim B \leftrightarrow B \sim A$
- Nem tranzitív: $A \sim B$ és $B \sim C$ esetén nem következik, hogy $A \sim C$

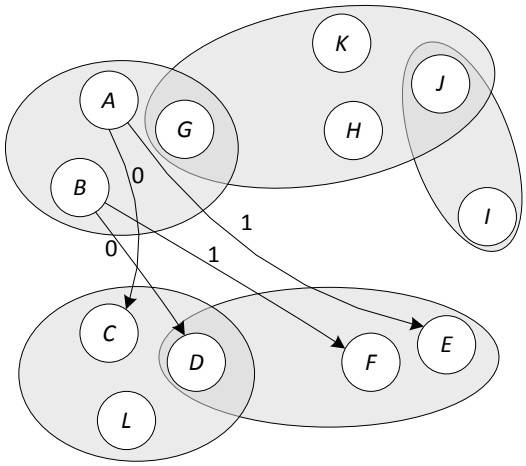
A kompatibilitási reláció segítségével az állapotok halmazát osztályokra lehet bontani, viszont a reláció nem partícionálja az állapotok halmazát, azaz (általában) nem diszjunkt részhalmazokra bontja a halmazt. A reláció által definiált osztályoknak lehetnek közös elemeik. Természetesen minden osztályra igaz az, hogy minden lehetséges állapotpárjára teljesül a kompatibilitás. Ezek az osztályok maximálisak, ha nem bővíthetők egyetlen elemmel sem.

A kompatibilitási osztályok egy adott halmaza zártnak mondjuk, ha bármely osztály tetszőleges két állapotából kiindulva minden olyan bemeneti kombinációra, amely mindkét állapotból specifikált következő állapotot ír elő, a következő állapotok is együtt szerepelnek a halmaz legalább egy osztályában.

Célunk maximális, zárt kompatibilitási osztályok meghatározása. Mivel minden osztály egy állapottá összevonható, ezért ha a maximális, zárt osztályokat meghatározzuk ezek jelentik az új állapotokat.

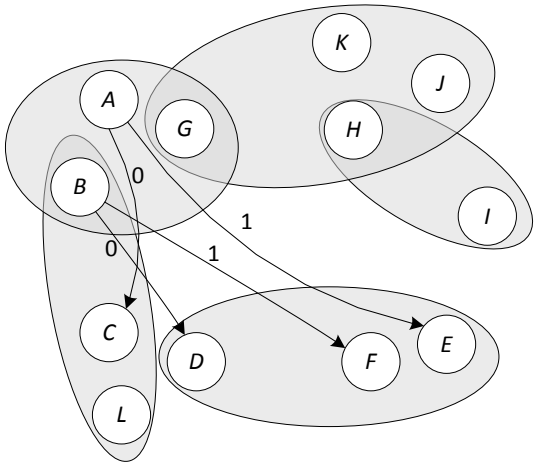
Példa

Az alábbi 7.2 ábrán egy hálózat állapotait ($A, \dots, L$) körökkel és öt maximális, zárt kompatibilitási osztályát szürke ellipszisekkel jelöltük. Továbbá élek mutatják, hogy $A$ és $B$ állapotokból $0$ és $1$ bemenetek hatására milyen állapotokba jutunk. Látható, hogy $0$ hatására $A$-ból $C$ állapotba, $B$-ből pedig $D$ állapotba kerülünk, melyeknek van közös kompatibilitási osztályuk, valamint $1$ hatására $A$-ból $E$-be, $B$-ből $F$-be kerül a kapcsolás, melyek szintén kompatibilis állapotok. Az ábrán nincsenek jelölve az automata kimeneti értékei, de természetesen azok is páronként megegyeznek, ha specifikálva vannak. Az ábrán látszik, hogy léteznek olyan állapotok, melyek több kompatibilitási osztályba is beletartoznak, ilyenek a $D$, $G$ és $J$ állapotok.




**7.2 ábra: Állapotok ekvivalencia osztályok szerinti egyik lehetséges csoportosítása**

Ugyanennek a példának egy másik lehetséges maximális kompatibilitási osztályozása az alábbi 7.3 ábrán látható. Itt a $C$ és $D$ állapotoknak nincs közös kompatibilitási osztályuk, azaz ez a felosztás nem zárt, mivel $A$ és $B$ állapotokból 0 hatására nem jutunk olyan állapotokba, melyek kompatibilisek lennének egymással.




**7.3 ábra: Állapotok ekvivalencia osztályok szerinti másik lehetéses csoportosítása**

A nem teljesen specifikált hálózatok állapotminimalizálásának négy lépése van:

1. Az összes kompatibilis és inkompatibilis állapotpár megkeresése.
2. A maximális kompatibilitási osztályok megkeresése.
3. A minimális számú zárt osztály kiválasztása.
4. A minimalizált állapottábla kitöltése.

## *Kompatibilis és inkompatibilis állapotpárok meghatározása*

A kompatibilis és inkompatibilis állapotpárok halmazát lépcsős tábla segítségével tehetjük meg, ugyanúgy, mint teljesen specifikált hálózatok esetében. Itt is három dolgot állapíthatunk meg egy állapotpárról.

- Két állapotról „ránézésre” meg tudjuk állapítani, hogy kompatibilisek, ha a két állapothoz tartozó sorokban a specifikált állapotok és kimenetek megegyeznek.
- Két állapotról „ránézésre” meg tudjuk állapítani, hogy inkompatibilisek, ha ugyanazon bemenet különböző kimenetet eredményez.

- Ha mind a két állapotra ugyanazok a kimenetek (egyik vagy akár mind a kettő lehet nem specifikált is), de a következő állapotok specifikáltak és különböznek, akkor nem tudjuk „ránézésre” eldönteni, hogy kompatibilisek vagy inkompatibilisek. Ilyenkor azt mondjuk, hogy a két állapot feltételesen kompatibilis és feljegyezzük, hogy mik az kompatibilitás feltételei, azaz mely állapotoknak kell kompatibilisnek lenniük ahhoz, hogy a két eredeti állapot is kompatibilis legyen. A feltételes kompatibilitást az ekvivalenciához hasonlóan a feltételekkel jelöljük. Ha például két állapotra feljegyeztük, hogy ($AB$, $CD$), akkor ezek az állapotok feltételesen kompatibilisek, és csak akkor kompatibilisek, ha $A \sim B$ és $C \sim D$ teljesülnek.

A kompatibilitási reláció teljesülését itt is egy ugyanolyan lépcsős táblában tudjuk feljegyezni, mint teljesen specifikált hálózatok esetén. A fenti tulajdonságokat bejegyezzük az első lépcsős táblába. A cellák kitöltése a következőképen történik.

- Ha az állapotpárra a specifikált kimenetek és következő állapotok is azonosak minden bemeneti kombinációra, akkor az aktuális cellában jelöljük a kompatibilitást. Figyelembe vesszük a fentebb már említett tulajdonságot azaz, hogy a nem specifikált kimenet/állapot azonos minden kimenettel/állapottal.
- Ha a specifikált kimeneti értékek legalább egy bemeneti kombinációra különböznek, akkor inkompatibilitás jelölés kerül a cellába.
- Ha a specifikált kimeneti értékek megegyeznek, de a következő állapotok eltérnek valamely bemenetre, akkor feltételes bejegyzés kerül a tábla megfelelő cellájába.

A második lépcsős tábla generálását a teljesen specifikált esethez hasonlóan végezzük el, itt az inkompatibilis bejegyzések hatását érvényesítjük a többi cellában. Ha egy állapotpárról tudjuk, hogy inkompatibilis, akkor az összes olyan állapotpárra, ahol ez a kompatibilitás feltétele, mondhatjuk, hogy inkompatibilisek. Természetesen az összes inkompatibilis bejegyzés hatását vizsgálni kell, azokat is, amelyek egy másik inkompatibilis bejegyzés hatására váltak inkompatibilissé. Ha minden inkompatibilis bejegyzés hatását megvizsgáltuk, akkor az összes olyan állapotpárra, amely nem inkompatibilis mondhatjuk, hogy kompatibilis függetlenül attól, hogy a cellában kompatibilis vagy feltételesen kompatibilis bejegyzés szerepel, azaz a cellákban a feltételesen inkompatibilis bejegyzéseket kicseréljük kompatibilis bejegyzésre.

Illusztratív példa

Használjuk a fejezet elején definiált illusztratív példát, melynek az állapottáblája lent látható. Írjuk fel az állapottáblához az első majd a második lépcsős táblát. Ehhez először páronként meg kell vizsgálni, hogy az állapotok kompatibilisek, inkompatibilisek vagy feltételesen kompatibilisek.

| X / Y | 0 | 1 |
|---|---|---|
| *A* | *A* 0 | *E* 0 |
| *B* | - 1 | *D* - |
| *C* | *A* - | *B* 0 |
| *D* | *C* - | - - |
| *E* | *C* 1 | - 0 |

**7.4 ábra: Illusztratív példa előzetes állapottáblája (NTSH)**

- $AB$ pár inkompatibilis, mivel a kimenet különbözik $X = 0$ esetében. $X = 1$ esetben a kimeneteket egyezőnek tekintjük, mivel $B$ állapotnál a kimenet nincs definiálva.

- $AC$ pár feltételesen kompatibilis, mivel a kimenetek megegyeznek, de a következő állapotok $X = 1$ esetben nem azonosak (viszont kompatibilisek még lehetnek). Azaz $A$ és $C$ akkor kompatibilis, ha $E$ és $B$ kompatibilis. Így a táblázat $AC$ cellájában az $EB$ pár szerepel, mint kompatibilitási feltétel.
- $AD$ pár az előző ponthoz hasonlóan feltételesen kompatibilis, mivel a kimeneteket egyformának tekintjük a $D$ állapotnál nem definiált kimenetek miatt, valamint $X = 1$ esetben $D$ állapotnál a következő állapot sincs definiálva, ezért azokat kompatibilisnek feltételezzük. Az $X = 1$ bemenet vizsgálatából pedig kiderül, hogy a kompatibilitás feltétele az $AC$.
- $AE$ pár inkompatibilis, mivel a kimenetek különböznek.
- $BC$ pár feltételesen kompatibilis, ahol a feltétel $BD$.
- $BD$ pár kompatibilis, mivel nincs olyan kimenet és állapot sem, amelyik mind a két állapotnál specifikálva lenne.
- $BE$ pár kompatibilis, mivel a következő állapotok és a kimenetek közül kizárólag az $X = 0$ bemenet esetén van a kimenet mind a két állapotra specifikálva és az megegyezik.
- $CD$ pár feltételesen kompatibilis, ahol a feltétel $AC$.
- $CE$ pár feltételesen kompatibilis, ahol a feltétel $AC$.
- $DE$ pár kompatibilis.

| | | | | |
|---|---|---|---|---|
| $B$ | \~ | | | |
| $C$ | $EB$ | $BD$ | | |
| $D$ | $AC$ | ~ | $AC$ | |
| $E$ | \~ | ~ | $AC$ | ~ |
| | $A$ | $B$ | $C$ | $D$ |

**7.5 ábra: Lépcsős tábla (kiindulás)**

A továbbiakban sötétszürke alapon fehér betűkkel írt cellák segítségével jelöljük azokat az inkompatibilis bejegyzéseket, melyek hatását már érvényesítettük. Az inkompatibilis bejegyzések vizsgálatának sorrendje tetszőleges, mi haladjunk oszloponként balról jobbra és az oszlopokon belül pedig fentről lefelé. Így első lépében válasszuk a következő táblázatból az $AB$ inkompatibilis párt. Mivel az $AB$ pár kompatibilitása nem feltétele más állapotpár kompatibilitásának, ezért a (7.5 ábra) táblázat az első lépésben nem módosul.

| | | | | |
|---|---|---|---|---|
| $B$ | \~ | | | |
| $C$ | $EB$ | $BD$ | | |
| $D$ | $AC$ | ~ | $AC$ | |
| $E$ | \~ | ~ | $AC$ | ~ |
| | $A$ | $B$ | $C$ | $D$ |

**7.6 ábra: Második lépcsős tábla készítése 1.**

A fent 7.6 ábrán lévő táblázatban még egy inkompatibilis bejegyzés van az $AE$ állapotpárnál. Ez a pár sem szerepel egyetlen feltételesen kompatibilis bejegyzésnél sem, ezért a táblázat ebben az esetben sem módosul (alábbi 7.7 ábra).

| | | | | |
|---|---|---|---|---|
| $B$ | \~ | | | |
| $C$ | $EB$ | $BD$ | | |
| $D$ | $AC$ | ~ | $AC$ | |
| $E$ | \~ | ~ | $AC$ | ~ |

| | $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|---|

**7.7 ábra: Második lépcsős tábla készítése 2.**

Mivel a fenti lépcsős táblában (7.7 ábra) nem szerepel érvényesítetlen inkompatibilis bejegyzés, ezért az összes feltételesen inkompatibilis bejegyzést kicserélhetjük kompatibilis bejegyzésre. A végleges második lépcsős tábla az alábbi 7.8-as ábrán látható.

| | | | | |
|---|---|---|---|---|
| $B$ | \~ | | | |
| $C$ | ~ | ~ | | |
| $D$ | ~ | ~ | ~ | |
| $E$ | \~ | ~ | ~ | ~ |
| | $A$ | $B$ | $C$ | $D$ |

**7.8 ábra: Második lépcsős tábla készítése 3.**

A második lépcsős táblából kiolvashatók a kompatibilis állapotpárok, ami jelen esetben $AC$, $AD$, $BC$, $BD$, $BE$, $CD$, $CE$ és $DE$. A kompatibilis állapotpárok segítségével a maximális kompatibilitási osztályokat meg lehet határozni.

## *Maximális kompatibilitási osztályok meghatározása*

A maximális kompatibilitási osztályok meghatározására két módszert ismertetünk. Az első kódszer a „kompatibilis párok alapján bővítéssel”, amelyet akkor célszerű használni, amikor kevés a kompatibilis pár. A másik módszer az „inkompatibilis párok alapján szétszedéssel”, amelyet akkor célszerű használni, amikor kevés az inkompatibilis pár. Mind a két esetben először generáljuk az összes kompatibilitási osztályt, majd ezek közül kiválasztjuk a maximálisakat, azaz azokat, amelyeket nem tartalmaz egyetlen másik kompatibilitási osztály sem.

### Kompatibilis párok alapján bővítéssel

Ennél a módszernél azt használjuk ki, hogy egy kompatibilitási osztályban az állapotok páronként kompatibilisek. Ez azt jelenti, hogy három állapotot tartalmazó kompatibilitási osztály generálásához három megfelelő kompatibilis állapotpárra van szükség, négy állapotot tartalmazóhoz hatra, öt állapotot tartalmazóhoz tízre és így tovább. Ha ezt általános formában szeretnénk felírni, akkor egy $n$ elemű kompatibilitási osztály generálásához $n(n-1)/2$ darab megfelelő kompatibilis állapotpárra van szükség.

A megfelelőséget úgy lehet ellenőrizni, hogy a kompatibilis párokra halmazként tekintünk, és e halmazok uniója adja a generálni kívánt osztályt, ezért pontosan annyi elemet tartalmaz, amennyit generálni szeretnénk. Például az $AB$, $AC$ és $BC$ állapotpárok uniója $ABC$ és ez egy három állapotot tartalmazó kompatibilis osztályt eredményez. Az $AB$, $AC$ és $AD$ állapotpárok uniója egy $ABCD$ halmazt eredményezne, de négy állapotot tartalmazó kompatibilitási osztályhoz hat kompatibilis állapotpárra lenne szükségünk, ezért ez a három kompatibilis pár nem eredményez kompatibilitási osztályt.

Ezek alapján a kompatibilitási osztályokat lehet úgy generálni, hogy a megfelelő számú kompatibilis állapotpárt keresünk, melyek összevonása a megfelelő állapotszámú kompatibilitási osztályt eredményezi. Miután generáltuk az összes kompatibilitási osztályt, csak ki kell választani közülük a maximálisakat. Egyszerűen belátható, hogy ha nincs $n$ elemszámú kompatibilitási osztály, akkor nincs $n+1$ elemszámú sem, azaz az algoritmus véges lépésen belül megáll.

Az algoritmus módosítható úgy is, hogy az $n$ elemszámú kompatibilitási osztályt az $n-1$ elemszámú osztályokból generáljuk. Ebben az esetben az $n$ elemszámú osztály generálásához $n$ darab $n-1$ elemszámú megfelelő osztályra van szükség. A megfelelőség a korábban bemutatotthoz hasonló, itt

az $n$ darab kompatibilitási osztály uniójának $n$ darab állapotot kell tartalmaznia új kompatibilitási osztály generálásához. Mi ezt a módosított algoritmust használjuk a továbbiakban.

Az új $n$ elemű kompatibilitási osztályok generálásához a következő módszert érdemes használni. Keresünk minden $n-1$ halmazhoz egy olyan másik osztályt, amely csak egyetlen állapotban különbözik tőle (ha többen különbözne, akkor az uniójuk már a szükségesnél több állapotot tartalmazna). Ellenőrizzük, hogy a két halmaz uniójának minden $n-1$ elemű részhalmaza szerepel-e a kompatibilitási osztályok között, és ha szerepel, akkor ez egy új kompatibilitási osztály. A módszer helyes működéséhez elegendő, ha a két halmaz csak az utolsó állapotban különbözik (feltételezve, hogy a halmaz elemei lexikografikus (ábécé) sorrendben vannak), valamint ha minden állapothoz nála lexikografikusan nagyobb halmazt keresünk. Így nem generálunk egy kompatibilitási osztályt többször.

Illusztratív példa

Induljunk ki az előző fejezetben meghatározott kompatibilis párokból, melyek önmagukban definiálják a kételemű kompatibilitási osztályokat. Ezek lexikografikus listája lent látható. Állítsuk elő első lépésben a háromelemű kompatibilitási osztályokat.

$$AC, AD, BC, BD, BE, CD, CE, DE$$

- Először keressünk az $AC$ halmazhoz egy másik halmazt, amely csak a $C$ állapotban különbözik, ez az $AD$ halmaz lesz. Ennek a két halmaznak az uniója az $ACD$ halmaz, amelynek három darab kételemű részhalmaza van az $AC$, az $AD$ és a $CD$. Mind a három szerepel a listánkban, ezért ez egy új kompatibilitási osztályt eredményez.
- Az $AD$ halmazhoz nincs lexikografikusan nagyobb halmaz, mely csak az utolsó állapotban különbözne tőle. Megjegyzendő, hogy az $AC$ és a $CD$ halmaz is csak egy állapotban különbözik az $AD$ halmaztól, de az $AC$ lexikografikusan kisebb, a $CD$ pedig nem az utolsó állapotban különbözik tőle, és ha ezeket felhasználnák új kompatibilis osztályok generálásához, akkor ugyanazt az osztályt kapnánk, mint az előző esetben.
- A $BC$ halmaztól csak az utolsó állapotban különbözik a $BD$ halmaz, amelyek uniója a $BCD$ halmaz, amely egy új kompatibilitási osztályt jelent, mivel a $CD$ halmaz is elme listánknak.
- A $BC$ halmaztól szintén csak az utolsó állapotban különbözik a $BE$ halmaz. A két halmaz uniója a $BCE$ halmaz, melynek kételemű részhalmazai $BC$, $BE$ és $CE$, amelyek szerepelnek a listánkban, azaz új kompatibilitási osztályt eredményez.
- A $BD$ halmaztól a $BE$ halmaz különbözik az utolsó állapotban, ezek uniója a $BDE$ halmaz, amely egy új kompatibilitási osztály, mivel a $DE$ halmaz is egy létező kompatibilitási osztály.
- A $BE$ halmaznál az utolsó állapotban különböző, nála lexikografikusan nagyobb halmaz nincs a listában.
- A $CD$ halmazhoz a $CE$ halmaz illik, ezek uniója a $CDE$ halmaz, melynek a $CD$, $CE$ és $DE$ részhalmazai is rendelkezésre állnak.
- A $CE$ és a $DE$ halmazokhoz nem találunk megfelelő halmazt.

Ezek alapján öt háromelemű kompatibilitási osztályt találunk, melyek lexikografikus listája lent látható. Próbáljunk ezekből négyelemű kompatibilitási osztályokat generálni.

$$ACD, BCD, BCE, BDE, CDE$$

- Az $ACD$ halmazhoz nincs lexikografikusan nagyobb halmaz, mely csak az utolsó állapotban különbözne tőle.
- A $BCD$ halmaztól csak az utolsó állapotban különbözik a $BCE$ halmaz. Ezek uniója a $BCDE$ halmaz, melynek négy háromelemű részhalmaza van, a $BCD$, a $BCE$, a $BDE$ és a $CDE$, mely halmazok mind elemei a listánknak, azaz egy új kompatibilitási osztályt kapunk.
- A $BCE$, a $BDE$ és a $CDE$ halmazokhoz nem találunk megfelelő halmazt.

Ezek alapján egyetlen négyelemű kompatibilitási osztályt találunk, melye lent látható. Mivel a listánk egy elemű, ezért biztosan nem fogunk öt darab kompatibilis osztályt találni benne, melyből ötelemű kompatibilitási osztályt lehetne generálni.

$$BCDE$$

Így meghatároztuk az összes kompatibilitási osztályt, melyek közül most ki kell választani a maximálisakat. Ezt úgy tehetjük meg, hogy kiválasztunk egyet a legtöbb állapotot tartalmazó osztályokból és az összes olyan kompatibilitási osztályt kitöröljük a listából, amelyet ez tartalmaz. Ezt addig ismételjük, amíg tudunk kiválasztani új kompatibilitási osztályt. Egy osztályt természetesen csak egyszer vizsgálunk meg. Az illusztratív példánkban 14 kompatibilitási osztályt határoztunk meg.

$$AC, AD, BC, BD, BE, CD, CE, DE, ACD, BCD, BCE, BDE, CDE, BCDE$$

A $BCDE$ osztály tartalmazza a legtöbb állapotot, azaz ezt vizsgáljuk először. Az osztály tartalmazza a $BC$, a $BD$, a $BE$, a $CD$, a $CE$, a $DE$, a $BCD$, a $BCE$, a $BDE$ és a $CDE$ osztályokat, ezeket töröljük a listából. Így a listánk hossza 4 elemre rövidül.

$$AC, AD, ACD, BCDE$$

A következő legtöbb állapotot tartalmazó osztály az $ACD$ halmaz. Ez a listában lévő osztályok közül tartalmazza az $AC$ és az $AD$ osztályokat, amelyeket eltávolíthatunk a listából. Így már csak két osztályunk marad. (Az $ACD$ halmaz tartalmazza a $CD$ osztályt is, de azt már korábban eltávolítottuk a listából.)

$$ACD, BCDE$$

Minden a listában még szereplő kompatibilitási osztályt megvizsgáltunk, azaz a listában lévő osztályok a maximális kompatibilitási osztályok.

### Inkompatibilis párok alapján szétszedéssel

Mivel az illusztratív példánkban kevés inkompatibilis pár van, ezért a következőkben bemutatandó módszer erre a példára hatékonyabb lesz, mint a kompatibilis párok alapján szétszedéssel algoritmusa. Természetesen könnyen lehetne olyan példát mutatni, ahol az előző módszer használata lenne a célravezetőbb.

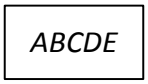
A fejezetben bemutatandó módszer hasonlít a TSH eset partíció finomítás módszeréhez. Ott feltételeztük, hogy az állapotok ekvivalensek egymással és megvizsgáltuk, hogy ez igaz-e (zártság ellenőrzése), és ha nem volt igaz, akkor szétbontottuk a partíciókat kisebb partíciókká. Most feltételezzük, hogy az állapotok kompatibilisek egymással és megvizsgáljuk, hogy ez igaz-e (inkompatibilis párok), és ha nem, akkor szétbontjuk az osztályokat.

Az eljárás során kiindulunk abból, hogy minden állapot kompatibilis minden állapottal és minden állapotot behelyezünk egy osztályba. Kiválasztunk egy inkompatibilis párt és két új osztályt készítünk, ahol mind a két osztály tartalmazza az összes állapotot az eredeti osztályból azzal a kivétellel, hogy az egyik osztály nem tartalmazza az inkompatibilis pár egyik tagját a másik pedig a másikat. Ezek után a két új osztállyal dolgozunk tovább és addig ismételgetjük a szétszedést, amíg van olyan osztály, amelyik megsérti valamelyik inkompatibilitást. A keresést egy bináris fával lehet ábrázolni, ahol

- A fa gyökere az az olyan osztály, amely tartalmazza az összes állapotot.
- Egy csúcs gyerekei a belőle előállított új osztályok.
- A fa levelei a kompatibilis osztályok.

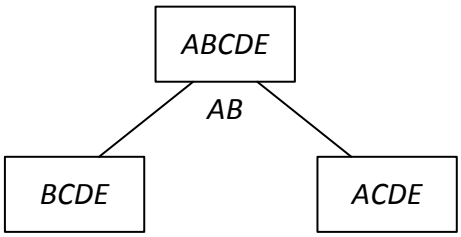
Illusztratív példa

A példánkban öt állapot van ($A$, $B$, $C$, $D$, $E$) és két inkompatibilis pár van, az $AB$ és az $AE$. Ezek alapján határozzuk meg a maximális kompatibilitási osztályokat. Első lépésben feltételezzük, hogy mind az öt állapot kompatibilis egymással és egy osztály képezünk belőlük.



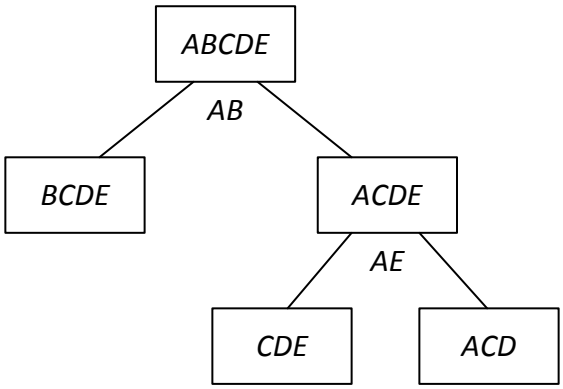

**7.9 ábra: Kompatibilitási osztályok szétszedéssel 1.**

Először vizsgáljuk meg az $AB$ inkompatibilis pár hatását. A módszer szerint ez két új osztályt generál, olyanokat, melyekben minden állapot szerepe, de az egyikből hiányzik az $A$, a másikból a $B$ állapot, azaz egy $BCDE$ és egy $ACDE$ osztályt készítünk.




**7.10 ábra: Kompatibilitási osztályok szétszedéssel 2.**

A két új osztályon vizsgáljuk meg az $AE$ inkompatibilitásának hatását. A $BCDE$ osztályban nem szerepel együtt a két állapot, ezért ezt az osztályt nem bontjuk ketté. Az $ACDE$ osztályban mind a két állapot szerepel, ezért ezt kettébontjuk és egy $CDE$ és egy $ACD$ osztályt kapunk.




**7.11 ábra: Kompatibilitási osztályok szétszedéssel 3.**

Mind a két inkompatibilis pár hatását megvizsgáltuk és a keresés eredményeképpen három kompatibilitási osztályt kaptunk, a $BCDE$, a $CDE$ és az $ACD$ osztályokat. Ezek még nem a maximális kompatibilitási osztályok, még meg kell vizsgálni, hogy nem tartalmazzák-e egymást. Természetesen az előző fejezetben használt maximális kompatibilitási osztályok kiválasztására bemutatott módszer itt is használható lenne, de a példánkban ránézésre is látszik, hogy a $BCDE$ osztály tartalmazza a $CDE$ osztályt, azaz a $CDE$ osztály nem tartozik a maximális kompatibilitási osztályok közé. Így a maximális kompatibilitási osztályaink a $BCDE$ és az $ACD$ osztályok, ugyanúgy, mint az előz fejezetben.

## *Minimális számú zárt kompatibilis osztály meghatározása*

A maximális kompatibilis osztályok száma jóval nagyobb is lehet, mint az eredeti hálózat állapotszáma és nincs is feltétlenül szükségünk az összes osztályra. Az állapotminimalizálás fő nehézsége az, hogy a lépcsős tábla alapján meghatározott maximális kompatibilitási osztályok halmazán kívül létezetnek más, esetleg kedvezőbb zárt osztály-halmazok, ezért az osztályok halmazát megvizsgáljuk, és megpróbáljuk egyszerűsíteni. Az egyszerűsítés jelentheti az osztályok számának vagy az osztályok elemszámának csökkentését. Az osztályok számának csökkentésével lehet elérni az állapotváltozók számának csökkenését, ami a felhasznált tárolók számának csökkentését jelenti. Ha egy osztályban kevesebb változó van, akkor nagyobb valószínűséggel tartalmaz nem specifikált értékeket, melyek a megvalósítás során előreláthatólag egyszerűbb hálózatot eredményeznek.

Az egyszerűsítés során két feltételt kell figyelembe vennünk, a lefedésnek zártnak és teljesnek kell lennie. A zártságot már definiáltuk korábban. Azt mondtuk, hogy az osztályok egy adott halmaza zárt, ha bármely osztály tetszőleges két állapotából kiindulva minden olyan bemeneti kombinációra, amely mindkét állapotból specifikált következő állapotot ír elő, a következő állapotok is együtt szerepelnek a halmaz legalább egy osztályában. A teljesség azt jelenti, hogy minden állapotnak szerepelnie kell legalább egy osztályban.

A használt eljárás fő lépései a következők:

1. A maximális kompatibilitási osztályok közül válasszuk ki a lehető legkevesebbet úgy, hogy minden állapot szerepeljen legalább egy osztályban.
2. Ha a kiválasztott halmaz nem zárt, akkor próbáljuk meg állapotok elhagyásával zárttá tenni. Ha nem sikerül, akkor lépjünk vissza az első pontra és válasszunk ki más osztályokat vagy adjunk hozzá ehhez csoporthoz új osztályt, ami zárttá teszi.
3. Hagyjuk el az osztályokból az elhagyható állapotokat a zártságot megtartva.

Az első pont végrehajtásához szükségünk lesz egy fedési táblázatra, amely nagyban hasonlít az Digitális technika I. jegyzetben használt prímimplikáns táblázatra. A fedési táblázat sorai a maximális kompatibilitási osztályok, oszlopai az állapotok. Ha egy osztály tartalmaz egy állapotot, akkor az adott cellába „X”-et kell írni. Ha van olyan állapot, amelyek oszlopában csak egyetlen „X” található, azaz csak egyetlen kompatibilitási osztály tartalmazza, akkor az ahhoz tartozó osztályt mindenképen ki kell választani. Ezek után megvizsgáljuk, hogy a kiválasztott osztályok lefednek-e minden állapotot, és ha nem, akkor a többi állapothoz is hozzá kell rendelni osztályokat, lehetőleg a minimális számút. A prímimplikáns táblánál megismert $S$ segédfüggvény itt is használható.

Illusztratív példa

A példánkban öt változó és kettő maximális kompatibilitási osztály van, azaz a táblázatnak kettő sora és öt oszlopa lesz. A táblázatban észrevehető, hogy az $A$ állapotot csak az $ACD$, valamint a $B$ és az $E$ állapotokat csak a $BCDE$ osztály fedi le, azaz mind a két kompatibilitási osztályt ki kell választanunk. A két osztály természetesen lefedi az összes állapotot.

| **Osztály** | ***A*** | ***B*** | ***C*** | ***D*** | ***E*** |
|---|---|---|---|---|---|
| $ACD$ | X | | X | X | |
| $BCDE$ | | X | X | X | X |

**7.1 táblázat: Fedési táblázat (illusztratív példa)**

A második pontban leírt zártsági feltétel teljesülését az állapottáblából lehet meghatározni. Meg kell vizsgálni, hogy mik lesznek a következő állapotok minden bemeneti kombinációra. Ha ugyanaz a következő állapot minden bemenetre, akkor azt figyelmen kívül hagyjuk, mivel minden állapot kompatibilis önmagával, valamint, ha a feltételt a kompatibilis önmaga teljesíti azt is figyelmen kívül hagyjuk. A többi bementre megnézzük, hogy az állapotoknak van-e közös kompatibilitási osztálya. Ha van, akkor teljesül a zártság, egyébként nem.

Illusztratív példa

Az alábbi állapottábla alapján ellenőrizzük a kiválasztott kompatibilitási osztályok ($ACD$, $BCDE$) zártságát. Ehhez írjuk fel egy az állapottáblához hasonló táblázatot, ahova be tudjuk jegyezni a következő állapotokat. Ebben a táblázatban a sorok az osztályok lesznek, az oszlopok pedig a bemenetek.

| X / Y | 0 | 1 |
|---|---|---|
| *A* | *A* 0 | *E* 0 |
| *B* | - 1 | *D* - |
| *C* | *A* - | *B* 0 |
| *D* | *C* - | - - |
| *E* | *C* 1 | - 0 |

**7.12 ábra: Előzetes állapottábla (illusztratív példa)**

Vizsgáljuk meg, hogy az $ACD$ majd a $BCDE$ osztály állapotai milyen bemenetre milyen állapotba kerülnek.

- $ACD$ osztály
  - Az $A$ állapot $X = 0$ hatására $A$ állapotba kerül, a $C$ állapot szintén $A$ állapotba kerül, ezért figyelmen kívül hagyjuk, valamint a $D$ állapot $C$ állapotba kerül. Így a táblázat megfelelő cellájába az $AC$ állapotok szerepelnek, amit azért rakunk ott zárójelbe, mert ez önmaga kompatibilise (az osztály tartalmazza az összes felsorolt állapotot), azaz figyelmen kívül hagyjuk.
  - Az $A$ állapot $X = 1$ hatására $E$ állapotba, a $C$ állapot $B$ állapotba, valamint a $D$ állapot határozatlan állapotba kerül. Így a táblázat megfelelő cellájába a $BE$ pár szerepel.
- $BCDE$ osztály
  - A $B$ állapot $X = 0$ hatására határozatlan állapotba, a $C$ állapot $A$ állapotba, a $D$ állapot $C$ állapotba kerül, valamint a $D$ állapot szintén $C$ állapotba kerül ezért figyelmen kívül hagyjuk. Így a táblázat megfelelő cellájába az $AC$ állapotok kerülnek.
  - A $B$ állapot $X = 1$ hatására $D$ állapotba, a $C$ állapot $B$ állapotba, a $D$ állapot határozatlan, valamint a $D$ állapot szintén határozatlan állapotba kerül. Így a táblázat megfelelő cellájába a $BD$ állapotpár kerül, amit zárójelbe rakhatunk, mert ez önmaga kompatibilise.

| ***X*** | **0** | **1** |
|---|---|---|
| ***ACD*** | **(*AC*)** | ***BE*** |
| ***BCDE*** | ***AC*** | **(*BD*)** |

**7.2 táblázat: Zártság vizsgálata (illusztratív példa)**

A 7.2 táblázat felírása után láthatjuk, hogy két esetet kell megvizsgálni, azokat, melyek nincsenek zárójelbe rakva. Ahhoz, hogy zártak legyenek az osztályok az $ACD$ osztályból $X = 1$ hatására egy olyan osztályba kell kerülni, amelyben $B$ és $E$ is szerepel (ez a $BCDE$ osztály), valamint $BCDE$ osztályból $X = 0$ hatására olyan osztályba kell kerülni, amelyben $A$ és $C$ is szerepel (ez az $ACD$ osztály). Mivel mind a két feltétel teljesül, a kiválasztott osztályaink zártak.

A harmadik pont szerint megpróbálhatjuk csökkenteni az állapotok számát az osztályokban. Természetesen csak olyan állapotot távolíthatunk el egy osztályból, amely legalább egy másik osztályban is szerepel, különben nem teljesülne a teljes lefedettség feltétele. Miután egy állapotot kivettünk egy osztályból újra ellenőrizni kell a zártság teljesülését.

Illusztratív példa

Két olyan állapotunk van, amely mind a két osztályunkban ($ACD$, $BCDE$) szerepel, ezek a $C$ és $D$ állapotok. Először vizsgáljuk meg, hogy a $C$ állapot elhagyható-e valamelyik osztályból. Ha az $ACD$ osztályból hagyjuk el a $C$ állapotot, akkor a két osztály az $AD$ és $BCDE$ lesz. A zártsági vizsgálat első lépése az alábbi táblázathoz vezet, amely annyiban különbözik az eredeti táblázatunktól, hogy $ACD$ osztály helyett $AD$ osztályunk van, valamint ennek az osztálynak a sorában nem hagyhatjuk figyelmen kívül az $X = 1$ esethez tartozó $AC$ állapotokat, mivel nem önmaga kompatibilise, viszont figyelmen kívül hagyhatjuk az $X = 1$ esethet tartozó $E$ állapotot, mivel egyedül van, azaz biztos kompatibilis lesz önmagával. A táblázat felírása után láthatjuk, hogy egy esetet kell megvizsgálni, hogy az $A$ és $C$ állapotokhoz tartozik-e közös kompatibilitási osztály. Mivel ez nem teljesül, ezért az osztályok nem zártak, azaz az $ACD$ osztályból nem lehet elhagyni a $C$ állapotot.

| X | 0 | 1 |
|---|---|---|
| ***AD*** | ***AC*** | **(*E*)** |
| ***BCDE*** | ***AC*** | **(*BD*)** |

**7.3 táblázat: Zártság vizsgálata (illusztratív példa)**

Ha az $BCDE$ osztályból hagyjuk el a $C$ állapotot, akkor a két osztály az $ACD$ és $BDE$ lesz. A zártsági vizsgálat első lépése az alábbi táblázathoz vezet. Ebben az esetben a táblázat második sora módosult, és a $BDE$ osztályhoz tartozó mind a két cellában egyetlen állapot szerepel, így mind a kettőt figyelmen kívül hagyhatjuk. A táblázat felírása után láthatjuk, hogy egy esetet kell megvizsgálni, hogy a $B$ és $E$ állapotokhoz tartozik-e közös kompatibilitási osztály. Van ilyen és ez a $BDE$ osztály, azaz az osztályok zártak, így a $BCDE$ osztályból el lehet elhagyni a $C$ állapotot.

| X | 0 | 1 |
|---|---|---|
| ***ACD*** | **(*AC*)** | ***BE*** |
| ***BDE*** | **(*C*)** | **(*D*)** |

**7.4 táblázat: Zártság vizsgálata (illusztratív példa)**

Most vizsgáljuk meg, hogy a $D$ állapot kihagyása az osztályokból milyen eredménnyel jár. Ha az $ACD$ osztályból hagyjuk el a $D$ állapotot, akkor a két osztály az $AC$ és $BCDE$ lesz. A zártsági vizsgálat első lépése az alábbi táblázathoz vezet, amely annyiban különbözik az eredeti táblázatunktól, hogy $ACD$ osztály helyett $AC$ osztályunk van, valamint ennek az osztálynak a sorában nem hagyhatjuk figyelmen kívül az $X = 1$ esethez tartozó $A$ állapotot A táblázat felírása után láthatjuk, hogy az eredeti feladathoz hasonlóan két esetet kell megvizsgálnunk. A $B$ és $E$ állapotok közösen szerepelnek a $BCDE$, valamint az $A$ és $C$ állapotok az $AC$ osztályban, azaz az osztályok zártak, Így az $ACD$ osztályból el lehet hagyni a $D$ állapotot.

| X | 0 | 1 |
|---|---|---|
| ***AC*** | **(*A*)** | ***BE*** |
| ***BCDE*** | ***AC*** | **(*BD*)** |

**7.5 táblázat: Zártság vizsgálata (illusztratív példa)**

Ha az $BCDE$ osztályból hagyjuk el a $D$ állapotot, akkor a két osztály az $ACD$ és $BCE$ lesz. A zártsági vizsgálat első lépése az alábbi táblázathoz vezet. A táblázat felírása után láthatjuk, hogy három esetet kell megvizsgálni, ahol a $B$ és $D$ állapotokhoz nem tartozik közös osztály, azaz az osztályok nem zártak, így a $BCDE$ osztályból nem lehet elhagyni a $D$ állapotot.

| X | 0 | 1 |
|---|---|---|
| ***ACD*** | (***AC***) | ***BE*** |
| ***BCE*** | ***AC*** | ***BD*** |

**7.6 táblázat: Zártság vizsgálata (illusztratív példa)**

Utoljára vizsgáljuk meg, hogy mi történik, ha mind a két állapotot el akarjuk hagyni. Összesen négy lehetséges módon lehetne elhagyni ezeket az állapotokat, de mivel a korábbi vizsgálatokból kiderült, hogy a $C$ állapotot nem lehet elhagyni az $ACD$ és a $D$ állapotot a $BCDE$ osztályokból, ezért egyetlen kombinációt kell megvizsgálnunk, amikor az $ACD$ osztályból a $D$ és a $BCDE$ osztályból a $C$ állapotot hagyjuk el. A táblázat felírása után egyetlen esetet kell megnézni, miszerint a $B$ és az $E$ állapotoknak van-e közös kompatibilitási osztálya. A $BDE$ osztály tartalmazza mind a két állapotot, ezért az osztályaink zártak.

| X | 0 | 1 |
|---|---|---|
| ***AC*** | (***A***) | ***BE*** |
| ***BDE*** | (***C***) | (***D***) |

**7.7 táblázat: Zártság vizsgálata (illusztratív példa)**

## *Összevont állapottábla*

Az előállított kompatibilitási osztályok alapján elvégezhetjük az állapotok összevonását az állapottáblában. Minden kompatibilitási osztályt egy állapotnak tekintünk. Mivel a kompatibilitás egyik feltétele az volt, hogy azonos bemenet hatására a specifikált kimenetek egyezzenek meg, ezért a kimenet meghatározásához elegendő, ha találunk egy olyan állapotot, ahol a kimenet specifikálva van és az ott szereplő értéket beírni az összevont állapottáblába. Ha az osztály egyik állapotnál sincs specifikálva a kimenet, akkor az itt sem lesz specifikálva.

A kompatibilitás másik feltétele az volt, hogy a következő állapotok legyenek kompatibilisek. Az összevont állapottábla következő állapotainak meghatározása hasonlít a zártsági feltétel vizsgálatára. Egy osztályhoz tartozó összevont állapot következő állapota egy olyan osztály állapota lesz, amely tartalmazza az összes olyan állapotot, amelyek az eredeti osztály állapotainak a következő állapotai. Abban az esetben, ha több osztály is tartalmazza az állapotokat, akkor szabadon választható, hogy melyiket írjuk be a táblába, és ha az összes osztály tartalmazza az állapotokat, akkor határozatlan állapot kerülhet az aktuális cellába.

Illusztratív példa

Induljunk ki az előzetes állapottáblából és a korábban meghatározott zárt kompatibilitási osztályokból. Mind a négy zárt kompatibilitási osztályra írjuk fel az összevont állapottáblát.

| X / Y | 0 | 1 |
|---|---|---|
| *A* | *A* 0 | *E* 0 |
| *B* | - 1 | *D* - |
| *C* | *A* - | *B* 0 |
| *D* | *C* - | - - |
| *E* | *C* 1 | - 0 |

**7.13 ábra: Előzetes állapottábla (illusztratív példa)**

Először azt az esetet vizsgáljuk meg, amikor nem csökkentettük az állapotok számát az osztályokban, azaz az osztályaink $ACD$ és $BCDE$. Minden kompatibilitási osztályhoz rendeljünk egy-egy állapotot. Legyen az $ACD$ osztályhoz tartozó állapot $S1$-el és a $BCDE$ osztályhoz tartozó állapot $S2$-vel jelölve. Vizsgáljuk meg, hogy melyik osztályból milyen bemenet hatására milyen osztályba jutunk és mi lesz a kimenet.

- $S1$ kompatibilitási osztály
  - $A$, $C$ és $D$ állapotokból $X = 0$ hatására rendre $A$, $A$ és $C$ állapotokba kerülünk. Mivel ez a két állapot az $ACD$ osztályban van benne együtt, ezért $S1$ állapotból $X = 0$ hatására $S1$ állapotba kerülünk.
  - $A$, $C$ és $D$ állapotokban $X = 0$ hatására a kimenet rendre $0$, $-$ és $-$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S1$ állapotban $X = 0$ hatására kimenet $0$.
  - $A$, $C$ és $D$ állapotokból $X = 1$ hatására rendre $E$, $B$ és $-$ állapotokba kerülünk. Mivel ez a két állapot az $BCDE$ osztályban van benne együtt, ezért $S1$ állapotból $X = 1$ hatására $S2$ állapotba kerülünk.
  - $A$, $C$ és $D$ állapotokban $X = 1$ hatására a kimenet rendre $0$, $0$ és $-$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S1$ állapotban $X = 1$ hatására kimenet $0$.
- $S2$ kompatibilitási osztály
  - $B$, $C$, $D$ és $E$ állapotokból $X = 0$ hatására rendre $-$, $A$, $C$ és $C$ állapotokba kerülünk. Mivel ez a két állapot az $ACD$ osztályban van benne együtt, ezért $S2$ állapotból $X = 0$ hatására $S1$ állapotba kerülünk.
  - $B$, $C$, $D$ és $E$ állapotokban $X = 0$ hatására a kimenet rendre $1$, $-$, $-$ és $1$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S2$ állapotban $X = 0$ hatására kimenet $1$.
  - $B$, $C$, $D$ és $E$ állapotokból $X = 1$ hatására rendre $D$, $B$, $-$ és $-$ állapotokba kerülünk. Mivel ez a két állapot az $BCDE$ osztályban van benne együtt, ezért $S2$ állapotból $X = 1$ hatására $S2$ állapotba kerülünk.
  - $B$, $C$, $D$ és $E$ állapotokban $X = 1$ hatására a kimenet rendre $-$, $0$, $-$ és $0$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S2$ állapotban $X = 1$ hatására kimenet $0$.

| X / Y | 0 | 1 |
|---|---|---|
| **S1** | **S1 0** | **S2 0** |
| **S2** | **S1 1** | **S2 0** |

**7.8 táblázat: Zártság vizsgálata (illusztratív példa)**

Következő esetben vizsgáljuk meg azt az esetet, amikor $BCDE$ osztályból elhagytuk a $C$ állapotot, azaz az osztályaink $ACD$ és $BDE$. Minden kompatibilitási osztályhoz rendeljünk egy-egy állapotot. Legyen az $ACD$ osztályhoz tartozó állapot $S1$-el és a $BDE$ osztályhoz tartozó állapot $S2$-vel jelölve. Ebben az esetben is vizsgáljuk meg, hogy melyik osztályból milyen bemenet hatására milyen osztályba jutunk és mi lesz a kimenet.

- $S1$ kompatibilitási osztály
  - $A$, $C$ és $D$ állapotokból $X = 0$ hatására rendre $A$, $A$ és $C$ állapotokba kerülünk. Mivel ez a két állapot az $ACD$ osztályban van benne együtt, ezért $S1$ állapotból $X = 0$ hatására $S1$ állapotba kerülünk.
  - $A$, $C$ és $D$ állapotokban $X = 0$ hatására a kimenet rendre $0$, $-$ és $-$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S1$ állapotban $X = 0$ hatására kimenet $0$.
  - $A$, $C$ és $D$ állapotokból $X = 1$ hatására rendre $E$, $B$ és $-$ állapotokba kerülünk. Mivel ez a két állapot az $BDE$ osztályban van benne együtt, ezért $S1$ állapotból $X = 1$ hatására $S2$ állapotba kerülünk.
  - $A$, $C$ és $D$ állapotokban $X = 1$ hatására a kimenet rendre $0$, $0$ és $-$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S1$ állapotban $X = 1$ hatására kimenet $0$.
- $S2$ kompatibilitási osztály
  - $B$, $D$ és $E$ állapotokból $X = 0$ hatására rendre $-$, $C$ és $C$ állapotokba kerülünk. Mivel a $C$ állapot az $ACD$ osztályban van benne $S2$ állapotból $X = 0$ hatására $S1$ állapotba kerülünk.
  - $B$, $D$ és $E$ állapotokban $X = 0$ hatására a kimenet rendre $1$, $-$ és $1$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S2$ állapotban $X = 0$ hatására kimenet $1$.
  - $B$, $D$ és $E$ állapotokból $X = 1$ hatására rendre $D$, $-$ és $-$ állapotokba kerülünk. Mivel a $D$ állapot az $ACD$ és a $BDE$ osztályban is benne van, ezért $S2$ állapotból $X = 1$ hatására nem specifikált állapotba kerülünk.
  - $B$, $D$ és $E$ állapotokban $X = 1$ hatására a kimenet rendre $-$, $-$ és $0$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S2$ állapotban $X = 1$ hatására kimenet $0$.

| X / Y | 0 | 1 |
|---|---|---|
| **S1** | **S1 0** | **S2 0** |
| **S2** | **S1 1** | **- 0** |

**7.9 táblázat: Zártság vizsgálata (illusztratív példa)**

A következőkben nem nézzük végig részletesen az összevont állapottábla generálását, csak az állapotokat definiáljuk és utána felírjuk a táblázatot. Most vizsgáljuk meg azt az esetet, amikor az $ACD$ osztályból elhagyjuk a $D$ állapotot, azaz az osztályaink $AC$ és $BCDE$. Minden kompatibilitási

osztályhoz rendeljünk egy-egy állapotot. Legyen az $AC$ osztályhoz tartozó állapot $S1$-el és a $BCDE$ osztályhoz tartozó állapot $S2$-vel jelölve. Az előzetes állapottábla alapján az összevont állapottábla a következő lesz, amely megegyezik az első összevont állapottáblánkkal.

| X / Y | 0 | 1 |
|---|---|---|
| S1 | S1 0 | S2 0 |
| S2 | S1 1 | S2 0 |

**7.10 táblázat: Zártság vizsgálata (illusztratív példa)**

Végül vizsgáljuk meg azt az esetet, amikor az $ACD$ osztályból elhagyjuk a $D$ állapotot és a $BCDE$ osztályból a $C$ állapotot, azaz az osztályaink $AC$ és $BDE$. Minden kompatibilitási osztályhoz rendeljünk egy-egy állapotot. Legyen az $AC$ osztályhoz tartozó állapot $S1$-el és a $BDE$ osztályhoz tartozó állapot $S2$-vel jelölve. Az előzetes állapottábla alapján az összevont állapottábla a következő lesz, amely megegyezik az első összevont állapottáblánkkal.

| X / Y | 0 | 1 |
|---|---|---|
| S1 | S1 0 | S2 0 |
| S2 | S1 1 | S2 0 |

**7.11 táblázat: Zártság vizsgálata (illusztratív példa)**

Az előzetes állapottábla öt állapotot tartalmazott, így annak a realizálásához három állapotváltozó, azaz három tároló lett volna szükséges. Az összevont állapottábla már csak kettő állapotot tartalmaz, azaz a hálózat egy tároló segítségével megvalósítható. A kompatibilitási osztályok négy generált csoportja két lehetséges állapottáblát eredményezet, ahol a két táblázat egyetlen helyen különbözött egymástól, mégpedig a másodikban egy specifikált állapot helyett nem specifikált szerepelt. Ezek alapján az első táblázat segítségével megvalósított hálózat biztosan nem egyszerűbb a második alapján megvalósítottnál, így érdemesebb a másodikat használni.

## *További példák:*

### 7.1 Példa

Minimalizáljuk a következő (7.11 ábra) előzetes állapottáblával leírt hálózat állapotainak a számát.

| X / Y | 0 | 1 |
|---|---|---|
| A | C - | D - |
| B | F 0 | D 0 |
| C | A 0 | - 1 |
| D | E 0 | - 0 |
| E | - 0 | C 1 |
| F | E - | A - |

**7.14 ábra: Előzetes állapottábla (7.1 példa)**

Az első lépcsős tábla előállításához páronként meg kell vizsgálni, hogy az állapotok kompatibilisek, inkompatibilisek vagy feltételesen kompatibilisek.

- $AB$ pár feltételesen kompatibilis, mivel a kimenet „megegyeznek”, de $X = 0$ esetében az állapotok nem azonosak (viszont kompatibilisek még lehetnek). Azaz $A$ és $B$ akkor kompatibilis, ha $C$ és $F$ kompatibilis.
- $AC$ pár kompatibilis, mivel a kompatibilitás feltétele $AC$ lenne, ami tautológia.
- $AD$ pár feltételesen kompatibilis, a kompatibilitás feltétele $CE$.
- $AE$ pár feltételesen kompatibilis, a kompatibilitás feltétele $CD$.
- $AF$ pár feltételesen kompatibilis, a kompatibilitás feltétele $CE$ és $AD$.
- $BC$ pár inkompatibilis, mivel a kimenetek különböznek $X = 1$ esetén.
- $BD$ pár feltételesen kompatibilis, a kompatibilitás feltétele $EF$.
- $BE$ pár inkompatibilis.
- $BF$ pár feltételesen kompatibilis, a kompatibilitás feltétele $EF$ és $AD$.
- $CD$ pár inkompatibilis.
- $CE$ pár kompatibilis.
- $CF$ feltételesen kompatibilis, a kompatibilitás feltétele $AE$.
- $DE$ pár kompatibilis.
- $DF$ pár kompatibilis.
- $EF$ pár feltételesen kompatibilis, a kompatibilitás feltétele $AC$.

| | | | | | |
|---|---|---|---|---|---|
| $B$ | $CF$ | | | | |
| $C$ | $\sim$ | $/\sim$ | | | |
| $D$ | $CE$ | $EF$ | $/\sim$ | | |
| $E$ | $CD$ | $/\sim$ | $\sim$ | $/\sim$ | |
| $F$ | $CE, AD$ | $EF, AD$ | $AE$ | $\sim$ | $AC$ |
| | $A$ | $B$ | $C$ | $D$ | $E$ |

**7.15 ábra: Első Lépcsős tábla (7.1 példa))**

Az első lépcsős táblából az inkompatibilitási kapcsolatok hatásának végigkövetésével elő lehet állítani a második lépcsős táblát, amely már csak kompatibilis és inkompatibilis bejegyzéseket fog tartalmazni. Az állapottáblából felírt inkompatibilis bejegyzések közül csak $CD$ feltétel szerepe a táblázatban, amiből következik $AE$ inkompatibilitása. Az új $AE$ inkompatibilitási bejegyzésből következik $CF$ inkompatibilitása, abból pedig $AB$ inkompatibilitása. Azokba a cellákba, ahova nem került inkompatibilis bejegyzés kompatibilis bejegyzést írunk, így alakul ki a feladat második lépcsős táblája.

| | | | | | |
|---|---|---|---|---|---|
| $B$ | $/\sim$ | | | | |
| $C$ | $\sim$ | $/\sim$ | | | |
| $D$ | $\sim$ | $\sim$ | $/\sim$ | | |
| $E$ | $/\sim$ | $/\sim$ | $\sim$ | $/\sim$ | |
| $F$ | $\sim$ | $\sim$ | $/\sim$ | $\sim$ | $\sim$ |
| | $A$ | $B$ | $C$ | $D$ | $E$ |

**7.16 ábra: Második Lépcsős tábla**

A második lépcsős táblából kiolvashatjuk a kompatibilis és az inkompatibilis párokat, melyek alapján meg lehet határozni a maximális kompatibilitási osztályokat. Kompatibilis párok az $AC$, $AD$, $AF$, $BD$, $BF$, $CE$, $DF$ és $EF$, inkompatibilis párok az $AB$, $AE$, $BC$, $BE$, $CD$, $CF$ és $DE$.

A maximális kompatibilitási osztályok előállítására két módszerünk van, melyek közül az alapján választunk, hogy kompatibilis vagy inkompatibilis párokból van-e kevés. Mivel hét darab inkompatibilis párunk van, ezért az „inkompatibilis párok alapján szétszedéssel" módszer egy hét mélységű bináris fát eredményezne, melynek generálása sok időt venne igénybe, ezért most a „kompatibilis párok alapján bővítéssel" módszer fogjuk alkalmazni. Első lépésként írjuk fel a kompatibilis párok lexikografikus listáját és állítsuk elő belőlük a háromelemű kompatibilitási osztályokat.

$$AC, AD, AF, BD, BF, CE, DF, EF$$

- Először keressünk az $AC$ halmazhoz egy másik halmazt, amely csak a $C$ állapotban különbözik, ez az $AD$ halmaz lesz. Ennek a két halmaznak az uniója az $ACD$ halmaz, amelynek három darab kételemű részhalmaza van az $AC$, az $AD$ és a $CD$. Mivel a $CD$ nem szerepel a listánkban, ezért az $ACD$ nem lesz kompatibilitási osztály.
- Az $AC$ halmazhoz képest az $AF$ halmaz is csak az utolsó állapotban különbözik, de a $CF$ pár sem szerepel a listában, így az $ACF$ sem lesz kompatibilitási osztály.
- Az $AD$ és $AF$ párok uniója az $ADF$ halmazt eredményezi, mely egy új kompatibilitási osztály, mivel a $DF$ pár eleme a listának.
- Az $AF$ halmazhoz képest nincs lexikografikusan nagyobb halmaz, mely csak az utolsó állapotban különbözne tőle.
- A $BD$ halmaztól csak az utolsó állapotban különbözik a $BF$ halmaz, amelyek uniója a $BDF$ halmaz, amely egy új kompatibilitási osztályt jelent, mivel a $DF$ halmaz is elme listánknak.
- A $BF$ halmazhoz képest nincs lexikografikusan nagyobb halmaz, mely csak az utolsó állapotban különbözne tőle.
- A $CF$, $DF$ és $EF$ halmazokhoz egyikéhez képest sincs lexikografikusan nagyobb halmaz, mely csak az utolsó állapotban különbözne tőlük.

Ezek alapján kettő darab háromelemű kompatibilitási osztályt találunk, melyek lexikografikus listája lent látható. Mivel a listánkban kettő elem van, ezért biztosan nem fogunk négy darab kompatibilis osztályt találni benne, melyből négyelemű kompatibilitási osztályt lehetne generálni.

$$ADF, BDF$$

Meghatároztuk az összes kompatibilitási osztályt, melyek közül most ki kell választani a maximálisakat. A példánkban 10 kompatibilitási osztályt határoztunk meg.

$$AC, AD, AF, BD, BF, CE, DF, EF, ADF, BDF$$

Először az egyik háromelemű osztályt választjuk ki, amely most legyen a $BDF$ osztály. Az osztály tartalmazza a $BD$, a $BF$, és a $DF$ osztályokat, ezeket töröljük a listából. Így a listánk hossza hét elemre rövidül.

$$AC, AD, AF, CE, EF, ADF, BDF$$

A következő legtöbb állapotot tartalmazó osztály az $ADF$ halmaz. Ez a listában lévő osztályok közül tartalmazza az $AD$ és az $AF$ osztályokat, amelyeket eltávolíthatunk a listából. Így már csak öt osztályunk marad.

$$AC, CE, EF, ADF, BDF$$

A listánk a már megvizsgáltakon kívül kizárólag kételemű osztályokat tartalmaz, amelyek nem tartalmazzák egymást, azaz a listából nem lehet több elemet eltávolítani, az összes benne szereplő elem egy-egy maximális kompatibilitási osztály. A következő feladtunk ezekből az osztályokból kiválasztani a minimális számú zárt osztályt a fedési táblázat segítségével

A példánkban hat változó és öt maximális kompatibilitási osztály van, azaz a táblázatnak öt sora és hat oszlopa lesz. A táblázatban észrevehető, hogy az $B$ állapotot csak a $BDF$ osztály fedi le, azaz ez a kompatibilitási osztályt mindenképpen ki kell választanunk.

| **Osztály** | ***A*** | ***B*** | ***C*** | ***D*** | ***E*** | ***F*** |
|---|---|---|---|---|---|---|
| $AC$ | X | | X | | | |
| $CE$ | | | X | | X | |
| $EF$ | | | | | X | X |
| $ADF$ | X | | | X | | X |
| $BDF$ | | X | | X | | X |

**7.12 táblázat: Fedési táblázat (7.1 példa)**

A kiválasztott $BDF$ osztály nem fedi le az $A$, $C$ és $E$ állapotokat, ezért itt a számjegyes minimalizálásnál megismert segédfüggvényt fogjuk használni. Először minden osztályhoz hozzárendelünk egy logikai változót, $a$-tól $e$-ig olyan sorrendben, ahogy a táblázatban szerepelnek, azaz $AC \rightarrow e$, $CE \rightarrow b$, $EF \rightarrow c$, $ADF \rightarrow d$ és $BDF \rightarrow e$. A változók értéke 1, ha bevesszük és 0, ha nem. Ha egy állapotot le kell fedni, akkor az állapotot lefedő osztályok közül legalább egyiknek 1-nek kell lennie, azaz logikai VAGY kapcsolat van közöttük, valamint minden állapotot le kell fedni, azaz ezeket a kifejezéseket logikai ÉS kapcsolattal kell összekötni. Ez a következő segédfüggvényt eredményezi $S = (a + d)e(a + b)(d + e)(b + c)(c + d + e)$, amelyből az egyszerűsítés után következő alakot kapjuk $S = abe + ace + bde$, azaz a teljes lefedéshez legalább három osztály szükséges. Az állapotokat le lehet fedni az $AC$, $CE$, $BDF$, vagy az $AC$, $EF$, $BDF$, vagy a $CE$, $ADF$, $BDF$ osztályokkal.

Vizsgáljuk meg például az $AC$, $CE$, $BDF$ osztályokat, hogy zártak-e. Nézzük meg, hogy az $AC$, a $CE$ és a $BDF$ osztály állapotai milyen bemenetre milyen állapotba kerülnek.

- $AC$ osztály
  - Az $A$ állapot $X = 0$ hatására $C$ állapotba, a $C$ állapot $A$ állapotba kerül. Így a táblázat megfelelő cellájába az $AC$ állapotok szerepelnek, amit azért rakunk ott zárójelbe, mert ez önmaga kompatibilise (az osztály tartalmazza az összes felsorolt állapotot), azaz figyelmen kívül hagyjuk.
  - Az $A$ állapot $X = 1$ hatására $D$ állapotba, a $C$ állapot $-$ állapotba kerül. Így a táblázat megfelelő cellájába a $D$ állapot szerepel zárójelben.
- $CE$ osztály
  - Az $C$ állapot $X = 0$ hatására $A$ állapotba, az $E$ állapot $-$ állapotba kerül. Így a táblázat megfelelő cellájába az $A$ állapot szerepel zárójelben.
  - Az $C$ állapot $X = 1$ hatására $-$ állapotba, az $E$ állapot $C$ állapotba kerül. Így a táblázat megfelelő cellájába a $C$ állapot szerepel zárójelben.
- $BDF$ osztály
  - A $B$ állapot $X = 0$ hatására $F$ állapotba, a $D$ állapot $E$ állapotba, az $F$ állapot $E$ állapotba kerül. Így a táblázat megfelelő cellájába az $EF$ állapotok kerülnek.
  - A $B$ állapot $X = 1$ hatására $D$ állapotba, a $D$ állapot $-$ állapotba, az $F$ állapot $A$ állapotba kerül. Így a táblázat megfelelő cellájába az $AD$ állapotpár kerül.

| X | 0 | 1 |
|---|---|---|
| ***AC*** | **(*AC*)** | **(*D*)** |
| ***CE*** | **(*A*)** | **(*C*)** |
| ***BDF*** | ***EF*** | ***AD*** |

**7.13 táblázat: Zártság vizsgálata (7.1 példa)**

Ha a két zárójelbe nem rakott állapotpárt megvizsgáljuk, akkor láthatjuk, hogy sem az $E$ és $F$, sem az $A$ és $D$ állapotoknak nincs közös osztályuk, azaz az osztályaink nem zártak. Ekkor azt tehetjük, hogy állapot(ok) eltávolításával megpróbáljuk zárttá tenni őket. Mivel a $BDF$ osztály az oka, hogy nem zártaz az osztályok, ezért ebből kellene kivenni állapotokat, de az osztály egyik állapotát sem tartalmazza másik osztály, így nem vehetünk ki belőle állapotot, mert különben a teljes lefedés nem teljesülne.

Mivel a lefedés meghatározásánál más osztályokat is meghatároztunk, ezért most próbáljuk ki az $AC$, $EF$, $BDF$ osztályokat, hogy zártak-e. Mivel csak egy osztályban különbözik az előző esettől ($EF$ osztály van $CE$ osztály helyett), ezért elegendő a táblázat felírásakor ezt az egy osztályt megvizsgálni.

- $EF$ osztály
  - Az $E$ állapot $X = 0$ hatására $-$ állapotba, az $F$ állapot $E$ állapotba kerül. Így a táblázat megfelelő cellájában az $E$ állapot zárójelben szerepel.
  - Az $E$ állapot $X = 1$ hatására $C$ állapotba, az $F$ állapot $A$ állapotba kerül. Így a táblázat megfelelő cellájába az $AC$ állapot szerepel.

| X | 0 | 1 |
|---|---|---|
| ***AC*** | **(*AC*)** | **(*D*)** |
| ***EF*** | **(*E*)** | ***AC*** |
| ***BDF*** | ***EF*** | ***AD*** |

**7.14 táblázat: Zártság vizsgálata (7.1 példa)**

Most három zárójelbe rakott állapotpárunk van, melyek közül az $A$ és a $C$ együtt szerepel az $AC$ osztályban, és az $E$ és az $F$ az $EF$ osztályban, de az $A$ és a $D$ nem szerepel együtt egyik osztályban sem. Így ezek az osztályok sem bizonyultak zártnak. A zártság eléréséhez ismét megpróbálhatunk állapotot eltávolítani valamelyik osztályból. Ez az osztály jelenleg is a $BDF$ osztály, mivel itt sérül a zártság. Ebből az osztályból most az $F$ állapotot lehetne eltávolítani, mivel ez szerepel másik osztályban is. Vizsgáljuk meg, hogy ebben az esetben hogyan változik a táblázatnak ez a sora.

- $BD$ osztály
  - A $B$ állapot $X = 0$ hatására $F$ állapotba, a $D$ állapot $E$ állapotba kerül. Így a táblázat megfelelő cellájába az $EF$ állapotok kerülnek.
  - A $B$ állapot $X = 1$ hatására $D$ állapotba, a $D$ állapot $-$ állapotba kerül. Így a táblázat megfelelő cellájába az $D$ állapot kerül zárójelben.

| X | 0 | 1 |
|---|---|---|
| ***AC*** | **(*AC*)** | **(*D*)** |
| ***EF*** | **(*E*)** | ***AC*** |
| ***BD*** | ***EF*** | **(*D*)** |

**7.15 táblázat: Zártság vizsgálata (7.1 példa)**

Láthatjuk, hogy az $F$ állapot eltávolításával a $BD$ osztályból sikerült zárt osztályokat készíteni.

Az algoritmusunk következő lépésében megpróbálhatnánk állapotokat eltávolítani az osztályokból, de mivel minden állapot pontosan egy osztályban szerepel, ezért ezt a lépést kihagyjuk és az előállított kompatibilitási osztályok alapján elvégezhetjük az állapotok összevonását az állapottáblában. Rendeljünk új állapotokat az osztályokhoz: jelölje $S1$ az $AC$, $S2$ az $EF$ és $S3$ a $BD$ osztályhoz tartozó állapotokat. Vizsgáljuk meg, hogy melyik osztályból milyen bemenet hatására milyen osztályba jutunk és mi lesz a kimenet.

- $S1$ kompatibilitási osztály
  - $A$ és $C$ állapotokból $X = 0$ hatására rendre $C$, és $A$ állapotokba kerülünk. Mivel ez a két állapot az $AC$ osztályban van benne együtt, ezért $S1$ állapotból $X = 0$ hatására $S1$ állapotba kerülünk.
  - $A$ és $C$ állapotokban $X = 0$ hatására a kimenet rendre $-$ és $0$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S1$ állapotban $X = 0$ hatására kimenet $0$.
  - $A$ és $C$ állapotokból $X = 1$ hatására rendre $D$ és $-$ állapotokba kerülünk. Mivel ez az állapot a $BD$ osztályban van benne, ezért $S1$ állapotból $X = 1$ hatására $S3$ állapotba kerülünk.
  - $A$ és $C$ állapotokban $X = 1$ hatására a kimenet rendre $-$ és $1$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S1$ állapotban $X = 1$ hatására kimenet $1$.
- $S2$ kompatibilitási osztály
  - $E$ és $F$ állapotokból $X = 0$ hatására rendre $-$, és $E$ állapotokba kerülünk. Mivel ez az állapot az $EF$ osztályban szerepel, ezért $S2$ állapotból $X = 0$ hatására $S2$ állapotba kerülünk.
  - $E$ és $F$ állapotokban $X = 0$ hatására a kimenet rendre $0$ és $-$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S2$ állapotban $X = 0$ hatására kimenet $0$.
  - $E$ és $F$ állapotokból $X = 1$ hatására rendre $C$ és $A$ állapotokba kerülünk. Mivel ez a két állapot az $AC$ osztályban van benne együtt, ezért $S2$ állapotból $X = 1$ hatására $S1$ állapotba kerülünk.
  - $E$ és $F$ állapotokban $X = 1$ hatására a kimenet rendre $1$ és $-$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S2$ állapotban $X = 1$ hatására kimenet $1$.
- $S3$ kompatibilitási osztály
  - $B$ és $D$ állapotokból $X = 0$ hatására rendre $E$, és $F$ állapotokba kerülünk. Mivel ez a két állapot az $EF$ osztályban szerepel együtt, ezért $S3$ állapotból $X = 0$ hatására $S2$ állapotba kerülünk.
  - $B$ és $D$ állapotokban $X = 0$ hatására a kimenet rendre $0$ és $0$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S3$ állapotban $X = 0$ hatására kimenet $0$.

- $B$ és $D$ állapotokból $X = 1$ hatására rendre $D$ és $-$ állapotokba kerülünk. Mivel ez az állapot az $BD$ osztályban van benne együtt, ezért $S3$ állapotból $X = 1$ hatására $S3$ állapotba kerülünk.
- $B$ és $D$ állapotokban $X = 1$ hatására a kimenet rendre $0$ és $0$. Mivel van olyan kimenet, amelyik specifikált, ezért ez lesz a kimenet, azaz $S3$ állapotban $X = 1$ hatására kimenet $0$.

| X / Y | 0 | 1 |
|---|---|---|
| A | C - | D - |
| B | F 0 | D 0 |
| C | A 0 | - 1 |
| D | E 0 | - 0 |
| E | - 0 | C 1 |
| F | E - | A - |

| X / Y | 0 | 1 |
|---|---|---|
| S1 | S1 0 | S3 1 |
| S2 | S2 0 | S1 1 |
| S3 | S2 0 | S3 0 |

**7.14 ábra: Összevont állapottábla (7.1 példa)**

A megvalósítandó kapcsolásban az állapot összevonások után nem 6 állapotot kell reprezentálni, hanem csak 3-at, és így a tárolók száma 3-ról 2-re csökkenthető.

# 8. Szinkron sorendi hálózatok állapotkódolása

Ebben a fejezetben a szinkron sorrendi hálózatok állapotkódolásával fogunk foglalkozni, azaz arról lesz szó, hogy a minimalizált állapottábla melyik állapotaihoz a belső változók milyen kombinációja tartozik. A célunk ezzel az, hogy a lehető legegyszerűbb kapcsolást tudjuk elérni. Az aszinkron hálózatok állapotkódolásáról a következő fejezetben lesz szó, mivel ott a cél nem az egyszerűség, hanem a versenyhelyzet elkerülése lesz.

Mint már korábban láthattuk az állapotokat bináris kóddal tudjuk reprezentálni a hálózatunkban. Tegyük fel, hogy $N$ darab állapot van. Ahhoz, hogy mind az $N$ állapothoz különböző kód tartozhasson, $n = \lceil \log_2 N \rceil$ darab bitre van szükség. Az állapotok bitjei attól függenek, hogy melyik tároló milyen értéket vesz fel, azaz a megvalósításban minimálisan $n$ darab tárolóra van szükség.

Az állapotkódolás jelentős mértékben befolyásolhatja a hálózat költségét. Általában mondhatjuk, hogy ha minimális számú tároló felhasználásával kódoljuk az állapotokat, akkor a vezérlő függvények bonyolultabbak lesznek több bemenetű kapukkal. Ezzel szemben, ha a tárolók számát megnöveljük, akkor egyszerűbb vezérlőfüggvényekhez jutunk kisebb bemenetszámú kapukkal.

A legegyszerűbb hálózatot úgy tudjuk biztosan elérni, ha kipróbáljuk az összes lehetséges állapotkódolást, azaz leszámláljuk a lehetőségeket. Természetesen az állapotkódolás után az összes lehetséges kódoláshoz meg is kell tervezni a hálózatot és ezek közül ki lehet választani a legegyszerűbbet. Ezzel a módszer a gyakorlatban nem megvalósítható, mivel nagyon sok különböző kódolást kell elkészíteni ugyanahhoz az állapottáblához. Ha 6 állapotunk szeretnénk 3 biten kódolni, akkor 420 lényegesen különböző állapotkódolás létezik. Általánosan mondhatjuk, hogy ha $M$ darab állapotot szeretnénk kódolni $n$ biten, akkor a lehetséges állapotkódolások száma

$$\frac{\binom{2^n}{M} M!}{n!\, 2^n}$$

A következőkben négy kódolási módszert mutatunk be, melyek sorban

- Szomszédos állapotkódok választása
- Kódolás helyettesítési tulajdonságú partíciók alapján
- Kódolás kimenet alapján
- Állapotonként egy bit kódolás

Illusztratív példa

A fejezetben a módszerek működésének bemutatásához a következő illusztratív példát használjuk. Legyen egy sorrendi hálózatnak egy $X$-szel jelölt bemenete, egy $Y$-nal jelölt kimenete és a következő állapottáblája.

| X / Y | 0 | 1 |
|---|---|---|
| *A* | *C* 0 | *D* 1 |
| *B* | *F* 0 | *D* 0 |
| *C* | *A* 0 | B 1 |
| *D* | *E* 0 | *B* 0 |
| *E* | *F* 0 | *C* 1 |
| *F* | *E* 0 | *A* 1 |

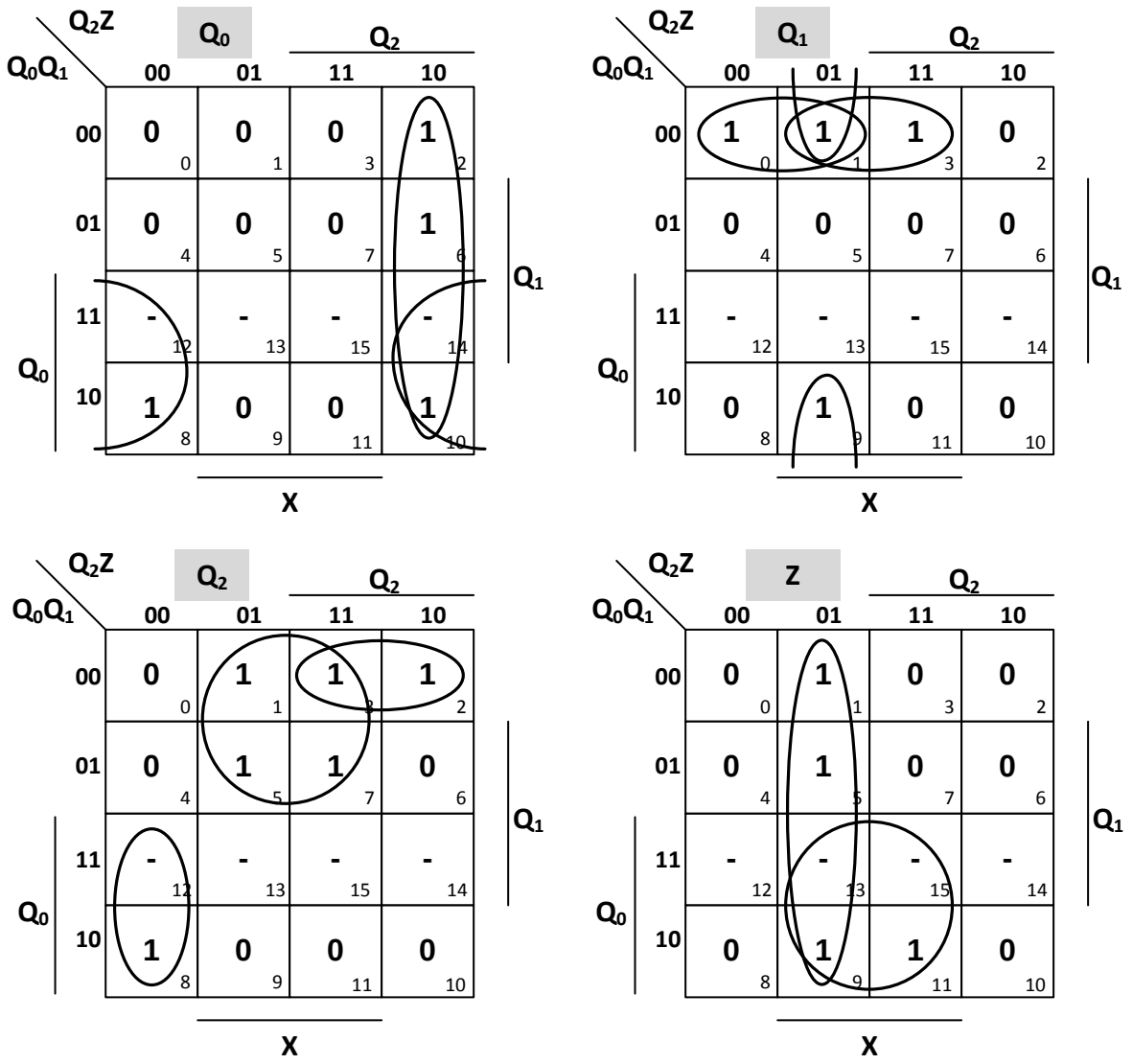
**8.1 ábra: Állapottábla (illusztratív példa)**

Ahhoz, hogy a bemutatandó kódolási módszerek hatékonyságát láthassuk, először kódoljuk az állapotokat „véletlenszerűen” és írjuk fel a vezérlőfüggvényeket. Mivel hat állapot van, ezért legalább három állapotváltozóra van szükség a kódoláshoz ($Q_0Q_1Q_2$). Rendeljük az állapotokhoz növekvő sorrendben a kódokat, azaz $A \rightarrow 000,\ B \rightarrow 001,\ C \rightarrow 010,\ D \rightarrow 011,\ E \rightarrow 100,\ F \rightarrow 101.$ Ezek alapján felírhatjuk a kódolt állapottáblát (8.2 ábra).

| $Q_0Q_1Q_2$ \ $X$ | 0 | 1 |
|---|---|---|
| 000 | 010 0 | 011 1 |
| 001 | 101 0 | 011 0 |
| 010 | 000 0 | 001 1 |
| 011 | 100 0 | 001 0 |
| 100 | 101 0 | 010 1 |
| 101 | 100 0 | 000 1 |
| 110 | --- - | --- - |
| 111 | --- - | --- - |

**8.2 ábra: Véletlenszerűen kódolt állapottábla (illusztratív példa)**

Az egyszerűség kedvéért használjunk mindenhol D tárolót, így az állapottáblából közvetlenül felírhatóak a Karnaugh táblák (8.3 ábra) és leolvashatók a vezérlőfüggvények és a kimenet függvénye.




**8.3 ábra: Karnaugh táblák a vezérlő kombinációs hálózatokra és a kimenetre**

$$Q_0 = Q_2\bar{X} + Q_0\bar{X}$$

$$Q_1 = \overline{Q_0}\,\overline{Q_1}\,\overline{Q_2} + \overline{Q_0}\,\overline{Q_1}X + \overline{Q_1}\,\overline{Q_2}X$$

$$Q_2 = \overline{Q_0}X + \overline{Q_0}\,\overline{Q_1}Q_2 + Q_0\,\overline{Q_2}\bar{X}$$

$$Z = \overline{Q_2}X + Q_0X$$

Ezzel a kódolással látható, hogy a vezérlőfüggvények nem túl egyszerűek, összesen 35 kapubemenetre van szükség a megvalósításhoz. Ezzel fogjuk majd a továbbiakban összehasonlítani a különböző kódolási módszereket.

## *Szomszédos állapotkódok választása*

Ez az algoritmus nem biztos, hogy optimális megoldást fog találni, mivel heurisztikus szabályok alapján működik. A szabályok betartása esetén nagy valószínűséggel jobb kódolást érünk el, mintha véletlenszerűen választanánk az állapotok kódjait.

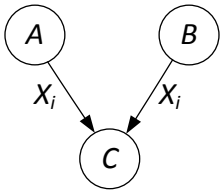
A módszerben megfogalmazott szabályok az állapotkódok szomszédosságát próbálják meghatározni. Azaz azt mondják meg, hogy mely állapotok kódjai különbözzenek legfeljebb egy bitben egymástól.

a. Az állapotok kódja legyen szomszédos, amelyeknek ugyanaz a következő állapota valamilyen bemenetre.
b. Az állapotok kódja legyen szomszédos, amelyek egy állapotnak a következő állapotai.
c. Az „a” szabály az erősebb.

A következőkben vizsgáljuk meg, hogy ezen szabályok alkalmazása miért is vezet jobb kódoláshoz, mint a véletlenszerű kódolás.

### Az „a” szabály

Az „a” szabály szerint szomszédosak azok az állapotok, melyeknek ugyanaz a következő állapota valamely bemenetre. Más szavakkal megfogalmazva, ha egy állapotba két másik állapotból el tudunk jutni ugyanannak a bemenetnek a hatására, akkor azok szomszédos kódokat kapjanak. Ha ezt az állapotgráfon (8.4 ábra) szeretnénk ellenőrizni, akkor, ha a következő részgráfot találjuk benne. Ebben az esetben $A$ és $B$ állapotok kódja az „a” szabály szerint szomszédos lesz.




**8.4 ábra: Állapotgráf az „a” szabály szemléltetésére**

Most nézzük meg, hogy ez a szabály miért is vezethet bennünket jobb állapotkódokhoz. Ehhez nézzünk meg egy példát, ahol két bemenet és négy állapot van. A példából nekünk most elegendő egy olyan részletet kiemelni, ahol a szabály teljesül, az alábbi állapottáblában (8.5-es ábra) is csak azokat az információkat írtuk be, ami jelenleg számunkra fontos. Azaz csak az a két állapotváltás szerepel benne, ahol megegyeznek a következő állapotok és a kimenetet sem jelöltük. A hálózat $A$ és $B$ állapotból is $C$ állapotba jut abban az esetben, ha a bemenet $X_1X_2 = 00$.

| $X_1X_2$ / $Y$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| ***A*** | ***C*** | | | |
| ***B*** | ***C*** | | | |
| ***C*** | | | | |
| ***D*** | | | | |

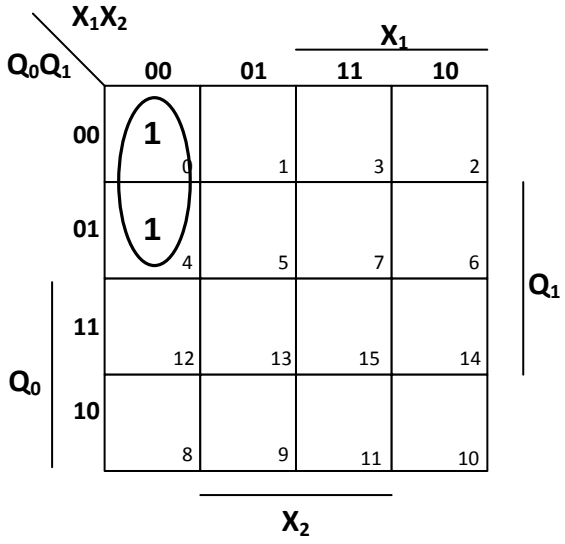
**8.5 ábra: Állapottábla az „a” szabály szemléltetésére**

Mivel négy állapotunk van, ezért két biten, két állapotváltozóval ($Q_0$, $Q_1$) lehet őket kódolni. Az „a” szabály alapján válasszuk a kódjukat szomszédosra. Így legyen például $A$ kódja $Q_0Q_1 = 00$, $B$ kódja $Q_0Q_1 = 01$, $C$ kódja pedig $Q_0Q_1 = 11$. A $D$ állapot kódja $Q_0Q_1 = 10$, de a példa szempontjából ez most közömbös. A részben kódolt állapottábla a következő ábrán látható (8.6 ábra).

| $X_1X_2$ / $Y$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 00 | 11 | | | |
| 01 | 11 | | | |
| 11 | | | | |
| 10 | | | | |

**8.6 ábra: Kódolt állapottábla az „a” szabály szemléltetésére**

Megfigyelhető, hogy $A$ és $B$ szomszédsága miatt $C$ állapot kódbitjei szomszédos rubrikákba kerültek, mivel a bemenet azonos és az állapotkód is csak egyetlen bitben különbözik. Ha ez a szomszédság teljesül, akkor a Karnaugh tábla esetében is szomszédosak lesznek ezek a cellák és ezekben a cellákban ugyanazok az értékek lesznek. Ha minden esetben így járunk el, akkor az 1-esek szomszédos cellákban lesznek, így lefedhetők majd közös hurokkal, a közös hurok pedig egyszerűsítési lehetőséget jelent. Az alábbi ábrán a $Q_0$ és a $Q_1$ állapotváltozók Karnaugh táblájának (8.7-ábra) az a részlete van kitöltve, ahol ezek a szomszédos cellák vannak.

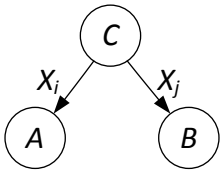



**8.7 ábra: Kimeneti függvény Karnaugh táblája**

Ha összehasonlítjuk a $Q_0$ vagy a $Q_1$ változó Karnaugh tábláját, akkor nem csak azt vehetjük észre, hogy közös hurokkal lefedhetőek ezek a cellák, hanem azt is, hogy ez a hurok mind a két Karnaugh táblában szerepel, azaz csak egyszer kell megvalósítani. Ez a példánkban a $\overline{Q_0}\overline{X_1}X_2$ prímimlikáns. Általánosságban elmondhatjuk, hogy ha a következő állapot kódjában (példánkban a $C$ állapot) több 1-es is van, akkor azok több Karnaugh táblában is ugyanott lesznek, azaz közös prímimplikánsok alakulnak ki.

### A „b” szabály

A „b” szabály szerint szomszédosak azok az állapotok, amelyek egy állapotnak a következő állapotai. Más szavakkal megfogalmazva, ha egy állapotból két másik állapotba el tudunk jutni ugyanannak a bemenetnek a hatására, akkor azok szomszédos kódokat kapjanak. Ha ezt az állapotgráfon szeretnénk ellenőrizni (8.8 ábra), akkor, ha a következő részgráfot találjuk benne. Ebben az esetben $A$ és $B$ állapotok kódja a „b” szabály szerint szomszédos lesz.




**8.8 ábra: Állapotgráf a „b” szabály szemléltetésére**

Most nézzük meg, hogy ez a szabály miért is vezethet bennünket jobb állapotkódokhoz. Ehhez nézzünk meg egy példát, ahol két bemenet és négy állapot van. A példából az előző szabály vizsgálatához hasonlóan elegendő egy olyan részletet kiemelni, ahol a szabály teljesül, az alábbi 8.9— es állapottáblában is csak azokat az információkat írtuk be, ami jelenleg számunkra fontos. A hálózat $C$ állapotból $A$ és $B$ állapotba is eljuthat a bemenettől függően.

| $X_1X_2$ / Y | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| ***A*** | | | | |
| ***B*** | | | | |
| ***C*** | ***A*** | ***B*** | | |
| ***D*** | | | | |

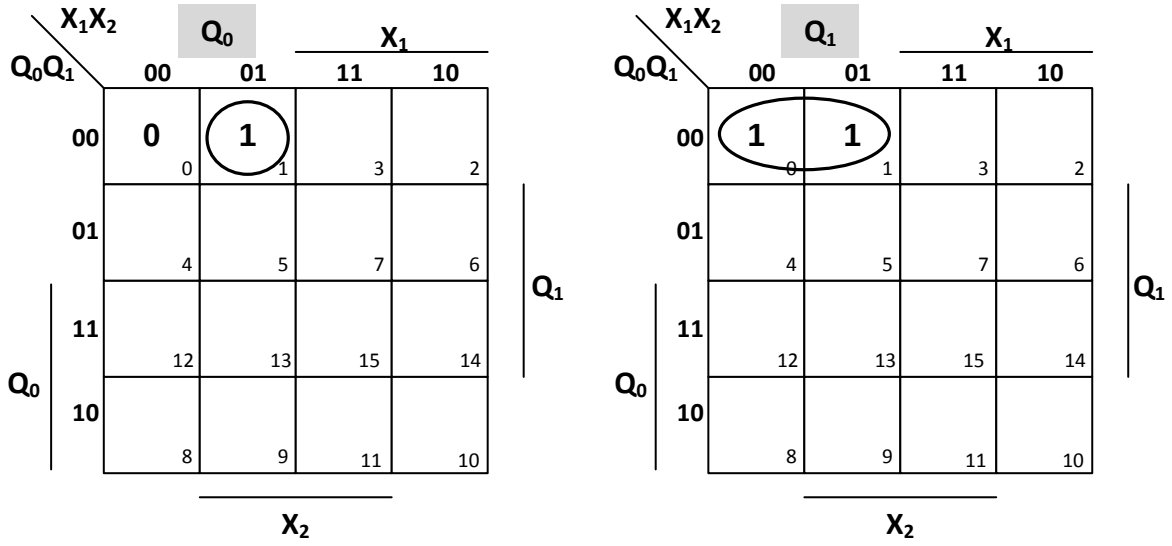
**8.9 ábra: Állapottábla a „b” szabály szemléltetésére**

Mivel négy állapotunk van, ezért két biten, két állapotváltozóval ($Q_0$, $Q_1$) lehet őket kódolni. A „b” szabály alapján válasszuk a kódjukat szomszédosra. Így legyen például $A$ kódja $Q_0Q_1 = 01$, $B$ kódja $Q_0Q_1 = 11$, $C$ kódja pedig $Q_0Q_1 = 00$. A $D$ állapot kódja $Q_0Q_1 = 10$, de a példa szempontjából ez most közömbös. A részben kódolt állapottábla a következő 8.10-es ábrán látható.

| $X_1X_2$ / Y | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| **01** | | | | |
| **11** | | | | |
| **00** | **01** | **11** | | |
| **10** | | | | |

**8.10 ábra: Kódolt állapottábla a „b” szabály szemléltetésére**

Mivel a szekunder változók szomszédosak, ezért a Karnaugh táblákban egyforma értékek ugyanabban a cellában, kivéve egy cellát, ahol a két kód különbözik. Ez alapján sok közös implikáns kialakítására van lehetőség. Az alábbi ábrán a $Q_0$ és a $Q_1$ állapotváltozók Karnaugh tábláinak az a részlete van kitöltve, ahol a részben kitöltött állapottábla is. A két Karnaugh táblában van közös hurok, ami jelen esetben a $\overline{Q_0}\,\overline{Q_1}\,\overline{X_1}\,X_2$, bár ez a $Q_1$ esetben nem prímimplikáns, azaz van nála egyszerűbb összevonási lehetőség is.




**8.11 ábra: Karnaugh tábla a „b" szabály szemléltetésére**

### A „c" szabály

Mivel az előző két szabály vizsgálata során egyértelműen látszik, hogy az „a" szabály nagyobb valószínűséggel vezet egyszerűbb vezérlő függvényekhez, ezért egyértelmű, hogy elsősorban azt célszerű alkalmazni.

### Állapotkódolás

Az „a" és a „b" szabály alapján össze lehet gyűjteni, hogy mely állapotpárok esetén érdemes szomszédos kódokat rendelni hozzájuk. Általában nem lehet az összes szabályt teljesíteni, ezért ki kell választani közülük a lehető legtöbbet úgy, hogy a „c" szabálynak is megfeleljünk. A szabályok kiválasztása nem feltétlenül triviális, a fejezetben megpróbáljuk magunktól kitalálni, hogy mely szabályokat érdemes felhasználni.

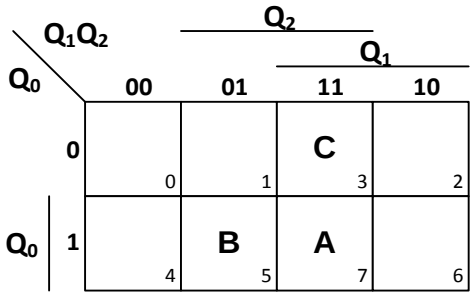
A korábbiakban már megismertünk egy módszert, amely segítségével könnyen lehetett látni a szomszédosságot és ez a Karnaugh tábla volt. Jelen esetben egy átalakított Karnaugh táblát használunk a szomszédosság teljesítéséhez és az állapotkódok kiolvasásához. A táblázatunk mérete attól függ, hogy hány változóval lehet kódolni az állapotainkat. Ha $n$ szekunder változónk van, akkor egy $n$ változós Karnaugh táblát használunk. A Karnaugh tábla peremezése az állapotváltozók szerint történik, és mindegy, hogy melyik perem melyik változót jelöli. A Karnaugh tábla celláiba jelen esetben nem 1-esek és 0-ák kerülnek, hanem állapotok, mégpedig úgy, hogy a szomszédos állapotok szomszédos cellákba kerülnek. Az állapotok kódjainak kiolvasása ezek után a peremezés segítségével könnyen megtörténhet.

Példa

Tegyük fel, hogy hét állapotunk van ($A$, $B$, $C$, $D$, $E$, $F$, $G$), az „a" szabály alapján szomszédosnak kell lennie az $AB$, $AC$, $BC$ és $EF$ állapotpároknak, valamint a „b" szabály alapján szomszédosnak kell lennie az $AB$, $CE$ és $DF$ állapotpároknak. Az állapotok számából következik, hogy három állapotváltozóval lehet kódolni őket, azaz egy háromváltozós Karnaugh táblára van szükségünk $Q_0$, $Q_1$ és $Q_2$ peremezéssel.

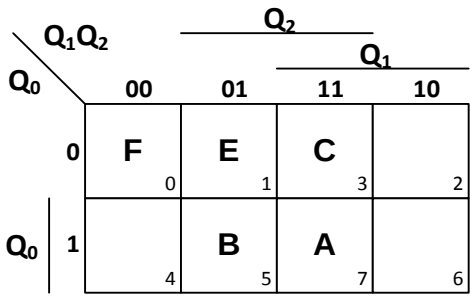
A „c" szabály szerint először az „a" szabály alapján alkotott párokat kell megvizsgálnunk és kiválasztanunk, hogy melyeket használunk fel. Az $AB$, $AC$ és $BC$ szabályok közül egyszerre csak kettő teljesülhet, mivel ha $A$ kódja egy bitben különbözik $B$ kódjától és $B$ kódja egy bitben különbözik $C$ kódjától, akkor $A$ kódja két bitben különbözik $C$ kódjától (vagy egyelőek, de akkor nem lenne különböző a kódjuk). Mivel a „b" szabály állapotpárjai között szerepel $AC$, ezért célszerű ezt kiválasztani, mint szomszédos állapotokat. Emellett szabadon választható, hogy $AC$ vagy $BC$

állapotokat választjuk szomszédosnak. Mi most az $AC$ állapotokat fogjuk kiválasztani. Ezeket a szomszédságot beírjuk a táblázatunk egy tetszőleges helyre, mint az a következő 8.12-es ábrán látszik.



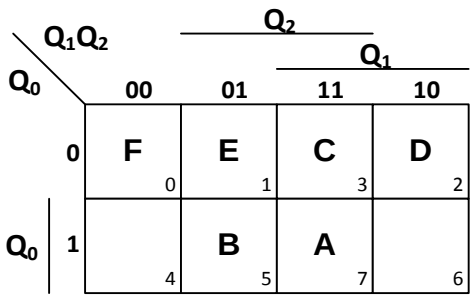

**8.12 ábra: Karnaugh tábla a szabályok alapján 1.**

Az „a” szabály szerint még az $EF$ állapotoknak is szomszédosnak kell lennie, de jelenleg nem tudjuk, hogy ezt hova írjuk be a táblázatba. A „b” szabály viszont azt mondja, hogy a $C$ és az $E$ állapot is legyen szomszédos, így már tudjuk, hogy az $E$ állapotot a $C$ mellé kell írni a táblázatba és a C mellé az $F$-et. Ezt is beírjuk a táblázatunkba (8.13).



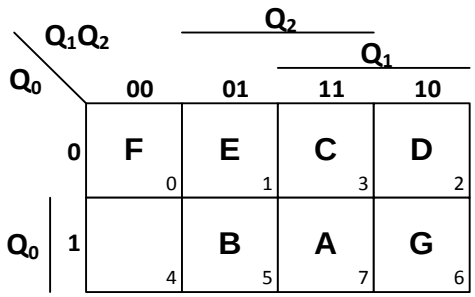

**8.13 ábra: Karnaugh tábla a szabályok alapján 2.**

Most már csak a $DF$ állapotpárt nem írtuk be a táblázat, amely alapján $D$ állapotot az $F$ állapot mellé írjuk.




**8.14 ábra: Karnaugh tábla a szabályok alapján 3.**

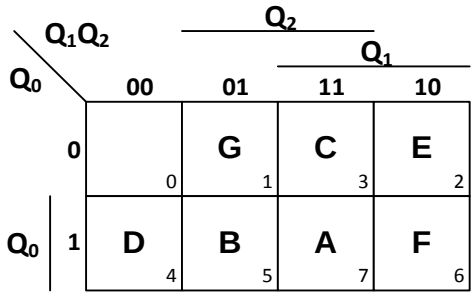
Az „a” és a „b” szabályok által meghatározott szomszédságot kielégítettünk, amit lehetséges volt, de a $G$ állapot nem szerepelt egyik szabályban sem, ezért azt tetszőleges helyre beírhatjuk.




**8.15 ábra: Karnaugh tábla a szabályok alapján 4.**

Minden állapot szerepel a táblázatban, ezért a Karnaugh táblázatnál megismert módon kiolvashatjuk az állapotok ($Q_0Q_1Q_2$) kódjait. Eszerint a következő kódolást hajtjuk végre: $A \rightarrow 111$, $B \rightarrow 101$, $C \rightarrow 011$, $D \rightarrow 010$, $E \rightarrow 001$, $F \rightarrow 000$, $G \rightarrow 010$.

A példából is látszik, hogy a táblázat kitöltése nem egyértelmű, sok olyan megoldás lehet, amely ugyanezeket a feltételeket elégíti ki mégis más kódolást eredményez. Például a következő táblázatban ugyanúgy kezdjük el a táblázat kitöltését, de az $E$ állapotot a $C$ állapot másik szomszédjába írjuk be. Ez csak egy a lehetséges sok megoldás közül, de ez is mutatja, hogy a módszer nem teljesen algoritmizálható, azaz számítógépes megvalósítása nehézségekbe ütközik.




**8.16 ábra: Karnaugh tábla a szabályok alapján (alternatív megoldás)**

Illusztratív példa

Nézzük meg, hogy milyen eredményeket tudunk elérni, ha a fejezet elején az alábbi állapottáblával (8.17-ábra) definiált példánk kódolásához a szomszédos állapotkódok választásának módszerét használjuk.

| X / Y | 0 | 1 |
|---|---|---|
| *A* | *C* 0 | *D* 1 |
| *B* | *F* 0 | *D* 0 |
| *C* | *A* 0 | B 1 |
| *D* | *E* 0 | *B* 0 |
| *E* | *F* 0 | *C* 1 |
| *F* | *E* 0 | *A* 1 |

**8.17 ábra: Állapottábla az illusztratív példához (adott)**

Az „a” szabály alapján azok az állapotok szomszédosak, amelyek ugyanolyan bement hatására ugyanolyan következő állapotba kerülnek. Ez azt jelenti, hogy az állapottáblát függőlegesen járjuk végig.

- Nézzük meg, hogy $X = 0$ bemenet esetén mely állapotpárokra igaz az „a” szabály.
  - A $B$ és az $E$ állapot is $F$ állapotba kerül, így $BE$ feltételünk lesz.
  - A $D$ és az $F$ állapot is $E$ állapotba kerül, így $DF$ feltételünk lesz.
- Nézzük meg, hogy $X = 1$ bemenet esetén mely állapotpárokra igaz az „a” szabály.
  - Az $A$ és a $B$ állapot is $D$ állapotba kerül, így $AB$ feltételünk lesz.
  - A $C$ és a $D$ állapot is $B$ állapotba kerül, így $CD$ feltételünk lesz.

Az „a” szabály így négy feltételt eredményez, az $AB$, $BE$, $CD$ és $DF$ feltételeket.

A „b” szabály alapján azok az állapotok szomszédosak, amelyek ugyanannak az állapotnak a következő állapotai. Ez azt jelenti, hogy az állapottáblát vízszintesen járjuk végig.

- Az $A$ állapot következő állapotai $C$ és $D$, így $CD$ feltételünk lesz.
- A $B$ állapot következő állapotai $F$ és $D$, így $FD$ feltételünk lesz.
- A $C$ állapot következő állapotai $A$ és $B$, így $AB$ feltételünk lesz.
- A $D$ állapot következő állapotai $E$ és $B$, így $BE$ feltételünk lesz.
- A $E$ állapot következő állapotai $F$ és $C$, így $CF$ feltételünk lesz.
- A $F$ állapot következő állapotai $E$ és $A$, így $AE$ feltételünk lesz.

A „b” szabály így hat feltételt eredményez, az $AB$, $AE$, $BE$, $CD$, $CF$ és $FD$ feltételeket.

Következő lépésben töltsük ki a táblázatot az állapotkódok meghatározásához. A hat állapot miatt egy háromváltozós Karnaugh táblára van szükségünk. Először teljesítsük az „a” szabály által meghatározott feltételeket, amely az alábbi táblázatot eredményezi (8.18.).

| $Q_1Q_2$ / $Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | A 3 | C 2 |
| 1 | F 4 | E 5 | B 7 | D 6 |

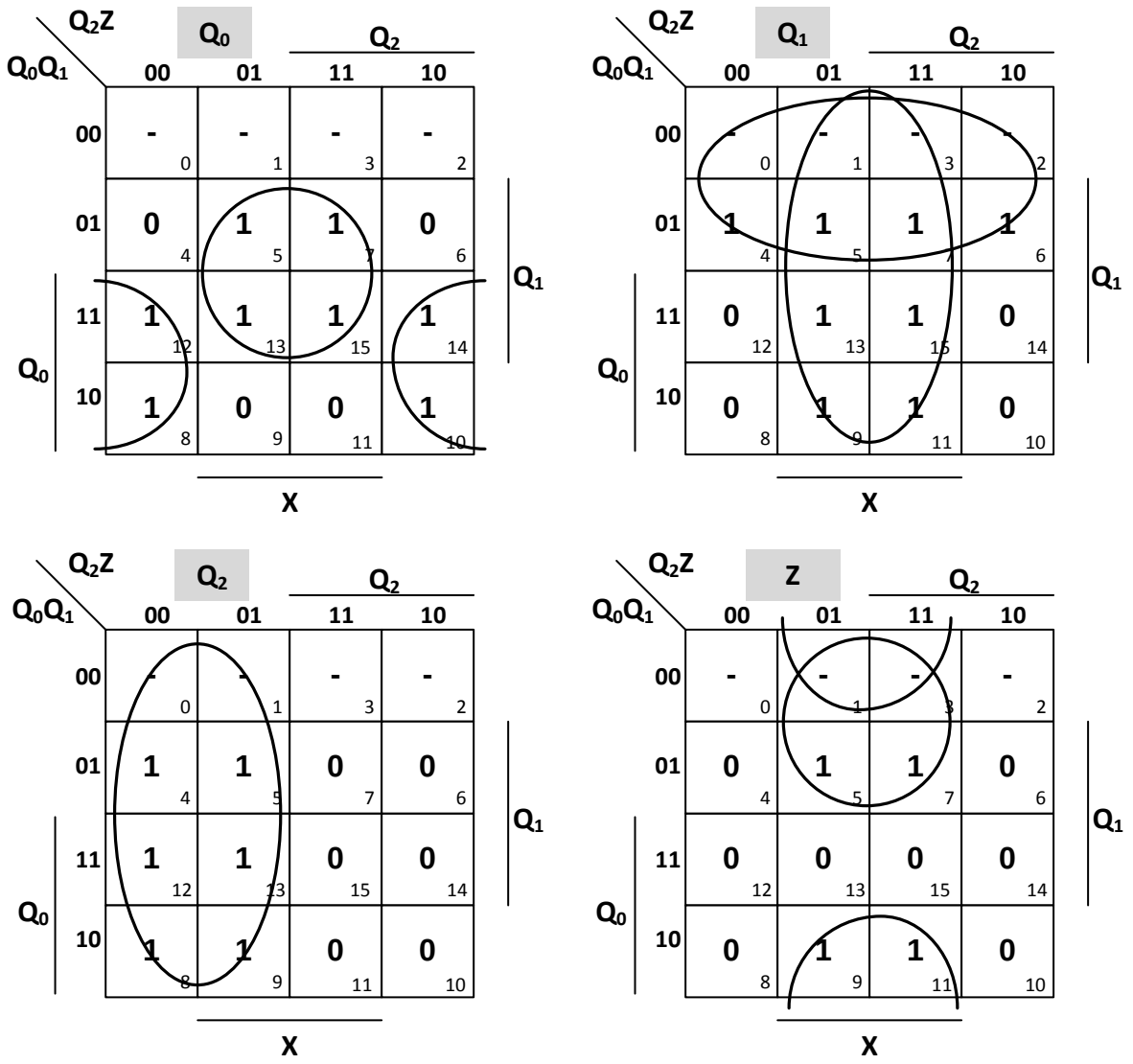
**8.18 ábra: Karnaugh tábla az „a” szabály alapján (illusztratív példa)**

A „b” szabály feltételei közül az $AE$ és a $CF$ feltétel nem teljesül, a többi az „a” szabály feltételei között is szerepel. Az $AE$ feltételt nem lehet együtt teljesíteni az „a” szabályban szereplő $AB$ és $BE$ feltételekkel, valamint a $CF$ feltételt nem lehet együtt teljesíteni a $CD$ és $DF$ feltételekkel. Így több feltételt nem tudunk teljesíteni, továbbá minden állapot szerepel a táblázatban, ezért a Karnaugh táblázatnál megismert módon kiolvashatjuk az állapotok ($Q_0Q_1Q_2$) kódjait. Eszerint a következő kódolást hajtjuk végre: $A \rightarrow 011$, $B \rightarrow 111$, $C \rightarrow 010$, $D \rightarrow 110$, $E \rightarrow 101$, $F \rightarrow 100$. Ezek alapján felírhatjuk a kódolt állapottáblát (8.19).

| $X$ / $Q_0Q_1Q_2$ | 0 | 1 |
|---|---|---|
| 011 | 010 0 | 110 1 |
| 111 | 100 0 | 110 0 |
| 010 | 011 0 | 111 1 |
| 110 | 101 0 | 111 0 |
| 101 | 100 0 | 010 1 |
| 100 | 101 0 | 011 1 |
| 000 | --- - | --- - |
| 001 | --- - | --- - |

**8.19 ábra: Kódolt állapottábla (illusztratív példa)**

Az egyszerűség és azt egységesség kedvéért használjunk ebben az esetben is mindenhol D tárolót, így az állapottáblából közvetlenül felírhatóak a Karnaugh táblák és leolvashatók a vezérlőfüggvények és a kimenet függvénye (8.20 ábra).




**8.20. ábra: Vezérlő kombinációs hálózatok és a kimeneti függvény Karnaugh táblái**

$$Q_0 = Q_0\bar{X} + Q_1X$$

$$Q_1 = X + \overline{Q_0}$$

$$Q_2 = \overline{Q_2}$$

$$Z = \overline{Q_0}X + \overline{Q_1}X$$

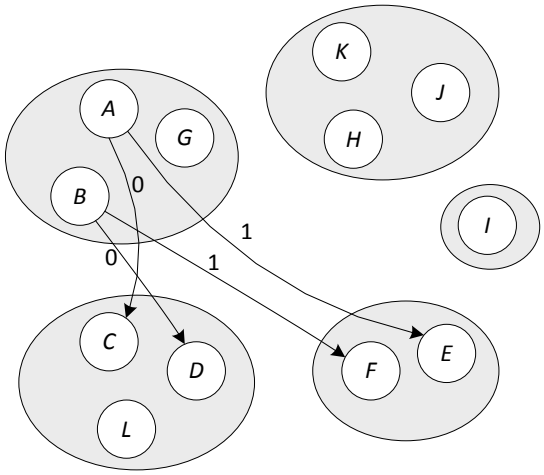
Ezzel a kódolással látható, hogy a vezérlőfüggvények a véletlenszerű kódoláshoz képest sokkal egyszerűbbek. Összesen 14 kapubemenetre van szükség a 35-höz képest.

## *Kódolás Helyettesítési Tulajdonságú (HT) partíciók alapján*

Az előző módszerrel ellentétben ez a kódolási eljárás könnyen algoritmizálható és optimális megoldást ad. Mint a neve is mutatja, ez a módszer azon alapul, hogy az állapotok halmazát partíciókra bontja. Bármely partícióra igaz az, hogy a partícióból bármely állapotot választok ki ugyanarra a bemenetre ugyanabba a partícióba kerülünk belőle, azaz a bemenet ismeretében megmondható, hogy melyik partícióból melyik partícióba kerülünk át. Ezt a tulajdonságot nevezzük helyettesítési tulajdonságnak.

Ez a definíció nagyban hasonlít a teljesen specifikált hálózatok esetén alkalmazott ekvivalencia osztályoknál használtra. Ott is igaz volt, hogy egy osztály minden eleméből ugyanarra a bemenetre

ugyanabba az osztályba tartozó állapotba kerülünk át. A különbség annyi, hogy most a kimeneteknek nem kell megegyezniük.



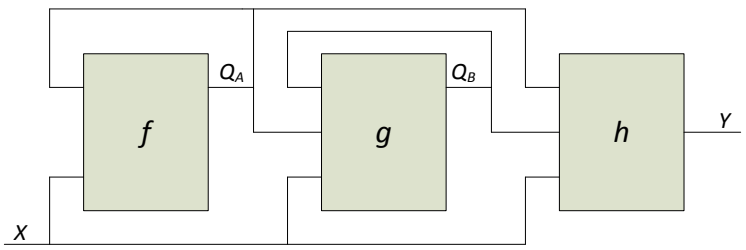

**8.21. ábra: Állapotok és partíciók**

Ez a fajta partícionálás arra lesz jó, hogy külön szekunder változók fogják azonosítani a partíciókat és külön változók a partíción belüli állapotokat. Ezek alapján a szükséges változók számát a partíciók száma és a partíciókban lévő állapotok száma határozza meg. Egy állapot kódját úgy kapjuk meg, hogy a partíciókat megkülönböztető kódok után írjuk a partíción belüli állapotot azonosító kódot.

Jelölje a $\Pi_i$ partícionálás partícióit megkülönböztető változókat $Q_{Ai}$, amelyek száma a szokásos módon meghatározható ($\lceil \log_2 Q_{Ai} \rceil$). Továbbá jelölje a partíciókon belüli állapokat megkülönböztető változókat $Q_{Bi}$, amelyek számát a legtöbb állapotot tartalmazó partíció mérete határoz meg ($\lceil \log_2 Q_{Bi} \rceil$). Azaz egy partícionálás kódolásához $\lceil \log_2 Q_{Ai} \rceil + \lceil \log_2 Q_{Bi} \rceil$ számú állapotváltozóra van szükség.

Mivel a következő partíció meghatározásánál a partícionálás tulajdonsága miatt nem számít, hogy az aktuális partíció mely állapotában vagyunk, ezért a $Q_{Ai}$ változók értéke sem függ a $Q_{Bi}$ változók értékétől, azaz $Q_{Ai}^{t+1} = f(Q_{Ai}^{t}, X)$ teljesül. Ezek miatt a $Q_{Ai}$ változókat önfüggő szekunder változóknak nevezzük. Az önfüggő változók ezen tulajdonsága miatt tudunk egyszerűbb kapcsolást tervezni. Mivel az önfüggő változók kevesebb változótól függenek, ezért a vezérlő függvényeik is egyszerűbbek, ezáltal olcsóbbak lesznek.

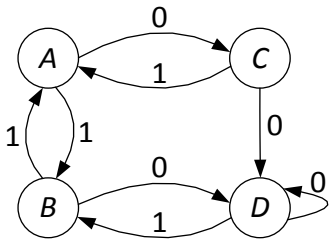
Ha helyettesítési tulajdonságú partíció alapján kódolunk, akkor a megvalósított hálózat a következő 8.22-ees ábrán látható módon néz majd ki. A $Q_A$ önfüggő változók vezérlőfüggvényeit az $f$ részhálózat határozza meg, melynek a bemenetei az $X$ bemenet és a $Q_A$ önfüggő változók. Az így felépülő kapcsolás esetén azt mondják, hogy soros dekompozíció alakul ki, mivel az $f$ és $g$ részhálózat sorba van kötve egymással.




**8.22. ábra: Megvalósítandó hálózat helyettesítési tulajdonságú partíció alapján (soros dekompozíció)**

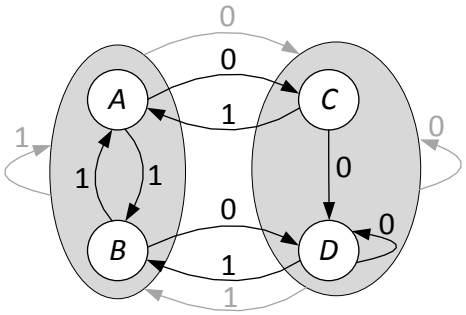
Példa

Tekintsük a következő négy állapottal rendelkező állapotgráfot (8.23).



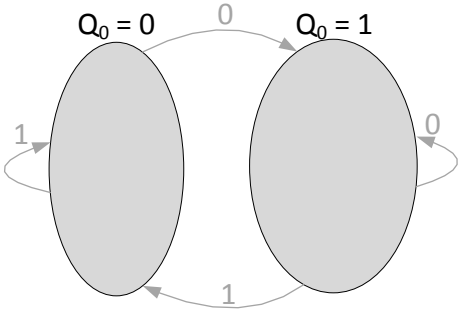

**8.23. ábra: Állapotgráf példa alapján (adott)**

Ha megvizsgáljuk ezt a fenti gráfot, észrevehetjük, hogy két partícióra osztható az állapotok halmaza. Az $A$ és $B$ állapot tartozik az egyik partícióba és a $C$ és $D$ állapot a másikba. Az $AB$ partícióból $X = 0$ hatására a $CD$ partícióba kerülünk, $X = 1$ hatására pedig maradunk az $AB$ partícióban. A $CD$ partícióból $X = 1$ hatására az $AB$ partícióba kerülünk, $X = 1$ hatására pedig maradunk a $CD$ partícióban. Mint látható teljesül a helyettesítési tulajdonság a partíciókra.



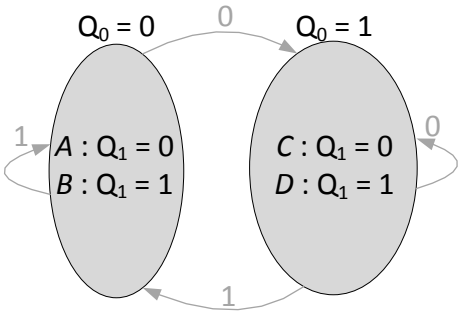

**8.24. ábra: Partíciók és állapotok a helyettesítési tulajdonság alapján**

Mivel két partíciónk van, ezért ezeket egy állapotváltozóval ($Q_0$) meg lehet különböztetni. Jelölje $Q_0 = 0$ az $AB$ partíciót és $Q_0 = 1$ a $CD$ partíciót.




**8.25. ábra: Partíciók a helyettesítési tulajdonság alapján**

Mind a két partícióban kettő elem szerepel, ezért a partíciókban lévő változók megkülönböztetéséhez egy változó ($Q_1$) elegendő. Jelölje az $AB$ partíción belül $Q_1 = 0$ az $A$ állapotot és $Q_1 = 1$ a $B$ állapotot, valamint jelölje a $CD$ partíción belül $Q_1 = 0$ az $C$ állapotot és $Q_1 = 1$ a $D$ állapotot.




**8.26. ábra: Partíciók, mint kódolt állapotok a helyettesítési tulajdonság alapján**

Ezek alapján az állapotok kódolása a következő: $A \rightarrow 00$, $B \rightarrow 01$, $C \rightarrow 10$, $D \rightarrow 11$.

Egy feladathoz természetesen nem csak egy olyan partícionálás létezik, amelyre teljesül a helyettesítési tulajdonság. Minden példához legalább két partícionálás van. Az egyik ($\Pi_0$ partícionálás) esetén az összes állapot külön-külön partícióban szerepel, a másik esetben ($\Pi_1$ partícionálás) pedig csak egyetlen partíció van, az tartalmazza az összes állapotot. Ezeket a partícionálásokat triviális partícionálásoknak nevezzük és ezek nem alkalmasak a kódoláshoz, mivel az első esetben kizárólag önfüggő, a másik esetben pedig csak nem önfüggő változóink lennének. A célunk, hogy a triviálisaktól eltérő összes lehetséges partícionálást előállítsuk, hogy azok közül kiválaszthassuk a legjobbat.

Mivel a partíciók helyettesítési tulajdonsága nagyfokú hasonlóságot mutat a teljesen specifikált hálózatok ekvivalencia osztályainak tulajdonságával, ezért itt is használhatjuk a lépcsős táblát. A módszer lényege ugyanaz, de itt nincsenek ekvivalens és antivalens állapotok, csak feltételek. Minden állapotpárra beírjuk a lépcsős táblába, hogy mikor tartozhatnak egy partícióba. Két állapot akkor tartozik egy partícióba, ha ugyanarra a bemenetre a következő állapotok is egy partícióba tartoznak. Ha nincs feltétel vagy tautológia lép fel, akkor a táblázatba '–' jelet teszünk. Ez abban különbözik az állapotminimalizálástól, hogy a kimeneteket nem kell figyelembe venni.

Miután kitöltöttük a lépcsős táblát, minden állapotpárnál felgöngyölítjük a feltételláncot, azaz ha egy állapotpár feltétele két másik állapot közös partícióba kerülése, akkor az állapotpár feltételei közé rekurzívan beletartoznak a két másik állapot feltételei is. Mivel a helyettesítési tulajdonság az ekvivalenciához hasonlóan tranzitív, ezért a feltételeket partíciókká ugyanúgy össze lehet vonni, mint az ekvivalencia osztályok esetében láttuk. Azok az állapotok, melyek nem szerepelnek egyetlen feltételben sem egyelemű partíciókat fognak alkotni.

Természetesen a partícionálások generálása során egy partícionálást többször is generálhatunk illetve triviális partícionálások is előfordulhatnak. A partícionálások közül az alapján választunk, hogy melyik megvalósításához van szükség kevesebb változóra, amelyet a partíciók száma és a legnagyobb elemszámú parítícióban lévő állapotok száma határoz meg, mint ahogy korábban ezt már megmutattuk ($\lceil \log_2 Q_{Ai} \rceil + \lceil \log_2 Q_{Bi} \rceil$).

Illusztratív példa

Nézzük meg, hogy milyen eredményeket tudunk elérni, ha a fejezet elején az alábbi (8.27 ábra) állapottáblával definiált példánk kódolásához a helyettesítési tulajdonságú partíciók módszerét használjuk.

| X / Y | 0 | 1 |
|---|---|---|
| **A** | **C** 0 | **D** 1 |
| **B** | **F** 0 | **D** 0 |
| **C** | **A** 0 | **B** 1 |
| **D** | **E** 0 | **B** 0 |
| **E** | **F** 0 | **C** 1 |
| **F** | **E** 0 | **A** 1 |

**8.27. ábra: Állapottábla illusztratív példa alapján (adott)**

A lépcsős tábla előállításához páronként meg kell vizsgálni, hogy az állapotok mikor tartozhatnak közös partícióba.

- $AB$ pár akkor tartozhat közös partícióba, ha $C$ és $F$ közös blokkban van, azaz az egy partícióba tartozás feltétele $CF$.
- $AC$ pár akkor tartozhat közös partícióba, ha $C$ és $A$, valamint $D$ és $B$ közös partícióban van. Az $AC$ feltétel tautológia lenne, ezért az egy partícióba tartozás feltétele $BD$.
- $AD$ pár egy partícióba tartozásának feltétele $CE$ és $BD$.
- $AE$ pár egy partícióba tartozásának feltétele $CF$ és $CD$.
- $AF$ pár egy partícióba tartozásának feltétele $CE$ és $AD$.
- $BC$ pár egy partícióba tartozásának feltétele $AF$ és $BD$.
- $BD$ pár egy partícióba tartozásának feltétele $EF$.
- $BE$ pár egy partícióba tartozásának feltétele $CD$.
- $BF$ pár egy partícióba tartozásának feltétele $EF$ és $AD$.
- $CD$ pár egy partícióba tartozásának feltétele $AE$.
- $CE$ pár egy partícióba tartozásának feltétele $AF$ és $BC$.
- $CF$ pár egy partícióba tartozásának feltétele $AE$ és $AB$.
- $DE$ pár egy partícióba tartozásának feltétele $EF$ és $BC$.
- $DF$ pár egy partícióba tartozásának feltétele $AB$.
- $EF$ pár egy partícióba tartozásának feltétele $AC$.

| | | | | | |
|---|---|---|---|---|---|
| $B$ | $CF$ | | | | |
| $C$ | $BD$ | $AF, BD$ | | | |
| $D$ | $CE, BD$ | $EF$ | $AE$ | | |
| $E$ | $CF, CD$ | $CD$ | $AF, BC$ | $EF, BC$ | |
| $F$ | $CE, AD$ | $EF, AD$ | $AE, AB$ | $AB$ | $AC$ |
| | $A$ | $B$ | $C$ | $D$ | $E$ |

**8.28 ábra: Lépcsős tábla**

Most kövessük végig minden állapotpárra a feltételláncot és a feltételek tranzitivitását kihasználva írjuk fel a partíciókat. A konkrét lépéseket az első három párra megnézzük és utána már csak a teljes feltételláncot és a partíciókat írjuk fel.

- $AB$

- Az $A$ és $B$ állapot akkor lehet közös partícióban, ha $C$ és $F$ állapot is közös partícióban van, azaz a feltétellánc jelenleg $AB$, $CF$. Az $AB$ párt már megvizsgáltuk, most nézzük a $CF$ párt. Ezek az állapotok akkor lehetnek egy partícióban, ha $AE$ és $AB$ feltételek is teljesülnek, azaz ezekkel ki kell bővíteni a feltételláncot, de mivel az $AB$ már szerepel a láncban, ezért azt nem rakjuk bele még egyszer. Ezek alapján a feltétellánc $AB$, $CF$, $AE$. A feltételláncban szereplő párok közül csak az $AE$ párt nem vizsgáltuk, ezért most ez következik. Az $AE$ feltétel akkor teljesül, ha teljesül $CF$ és $CD$ feltétel is. Ezek közül $CD$ nem szerepel a láncban, ezért ezzel bővítjük. Ezek után a feltétellánc $AB$, $CF$, $AE$, $CD$. A $CD$ állapotpárhoz a lépcsős táblában az $AE$ feltétel tartozik, de ez már szerepel a láncban, ezért nem bővítjük tovább a láncot. Mivel minden feltételt megvizsgáltunk, így a végleges feltétellánc $AB$, $CF$, $AE$, $CD$ lesz.
- A partíciók kialakításához a feltételláncban szereplő elemeket (párokat) partícióknak tekintjük, és azon elemeket összevonjuk, amelyek tartalmaznak közös állapotot. Az $AB$ és $AE$ párok is tartalmazzák az $A$ állapotot, ezért összevonhatóak, így a feltétellánc $ABE$, $CF$, $CD$ lesz. Hasonlóan a $CF$ és $CD$ párok is tartalmaznak közös állapotot, ezért összevonhatóak, azaz $ABE$, $CDF$ feltételláncot kapunk. Mivel a két elemnek már nincs közös eleme, ezért kaptunk egy új partícionálást, melyet jelöljünk $\Pi_2$-vel, mivel $\Pi_0$ és $\Pi_1$ a triviális parítcionálásnak foglalt. Az eredményül kapott partíciók a következők: $\Pi_2 = (ABE)(CDF)$

- $AC$
  - Az $A$ és $C$ állapot akkor lehet közös partícióban, ha $B$ és $D$állapot is közös partícióban van, azaz a feltétellánc jelenleg $AC$, $BD$. A $BD$ párt még nem vizsgáltuk a feltételláncból, ezért ellenőrizzük azt. A $BD$ feltéttel akkor teljesül, ha $EF$ teljesül, mely feltételt berakjuk a láncba. Ezek alapján a feltétellánc $AC$, $BD$, $EF$. A feltételláncban szereplő párok közül csak az $EF$ párt nem vizsgáltuk. Az $EF$ feltétel akkor teljesül, ha teljesül $AC$ feltétel. $AC$ feltétel már szerepel a láncban, ezért nem rakjuk bele. Mivel minden a láncban szereplő feltételt megvizsgáltunk, így a végleges feltétellánc $AC$, $BD$, $EF$ lesz.
  - Mivel a feltételláncban nincsenek olyan elemek, melyeknek tartalmaznának közös állapotot, ezért nem lehet őket összevonni. Az eredményül kapott partícionálás a következő: $\Pi_3 = (AC)(BD)(EF)$
- $AD$
  - Az $AD$ feltételhez szükség van a $CE$ és $BD$ feltételek teljesüléséhez, így mind a két feltételt hozzáadjuk a feltétellánchoz, azaz a feltétellánc jelenleg $AD$, $CE$, $BD$. A láncból sem a $CE$ sem a $BD$ feltételeket nem ellenőriztük. Mivel mind a kettőt előbb utóbb ellenőrizni kell, ezért szabadon választható, hogy melyikkel folytassuk. Mi a továbbiakban mindig a listában előrébb szereplő nem vizsgált párt fogjuk továbbgöngyölíteni, azaz jelen esetben a $CE$ párt választjuk. Ez akkor teljesül, ha $AF$ és $BF$ is teljesül, amelyeket szintén hozzáadunk a lánchoz, azaz a feltétellánc a következő lesz $AD$, $CE$, $BD$, $AF$, $BF$. A láncban a $BD$ a legelső még nem vizsgált elem, melyhez a lépcsős táblában az $EF$ feltétel tartozik, melyet szintén hozzáfűzünk a láncunk végéhez. Az így kialakult $AD$, $CE$, $BD$, $AF$, $BF$, $EF$ láncból az $AF$ pár teljesüléséhez a $CE$ és $AD$ teljesülése kell, de mind a kettő szerepel a láncban, ezért nem bővítünk. A $BF$ pár hatására $AD$ és $EF$ kerülne bele a láncban, de már szerepelnek benne. Az $EF$ feltétel az $AC$ feltétel láncba való bevételét eredményezi, ezért az új láncunk $AD$, $CE$, $BD$, $AF$, $BF$, $EF$, $AC$. Az $AC$ feltétel a $BD$ párt adná hozzá a lánchoz, de az már szerepel benne. Mivel minden feltételt megvizsgáltunk, így a végleges feltétellánc $AD$, $CE$, $BD$, $AF$, $BF$, $EF$, $AC$ lesz.
  - A partíciók kialakításához először vonjuk össze azokat az elemeket, melyek tartalmazzák az $A$ állapotot, így kapjuk a $ACDF$, $CE$, $BD$, $BF$, $EF$ láncot. Következő lépésben vonjuk össze azokat, melyek a $B$ állapotot tartalmazzák. Ez a $ACDF$, $CE$, $BDF$, $EF$ feltételláncot eredményezi. Ezután vonjuk össze a $C$ állapotot tartalmazó

elemeket, mely a $ACDEF$, $BDF$, $EF$ partíciókat generálja. Majd az $F$ állapotot tartalmazó elemeket is vonjuk össze, mely alapján az összes állapot egy partícióba kerül. Azaz $AD$ állapotpár vizsgálata a $\Pi_1 = (ABCDEF)$ triviális partícionálást eredményezi.

- $AE$
  - Feltétellánc $AE$, $CF$, $CD$, $AB$
  - $\Pi_2 = (ABE)(CDF)$ az $AB$ állapotpár vizsgálatánál már előállított partícionálás
- $AF$
  - Feltétellánc $AF$, $CE$, $AD$, $BC$, $BD$, $EF$, $AC$
  - $\Pi_1 = (ABCDEF)$ triviális partícionálás
- $BC$
  - Feltétellánc $BC$, $AF$, $BD$, $CE$, $AD$, $EF$, $BD$
  - $\Pi_1 = (ABCDEF)$ triviális partícionálás
- $BD$
  - Feltétellánc $BD$, $EF$, $AC$
  - $\Pi_3 = (AC)(BD)(EF)$ az $AC$ állapotpár vizsgálatánál már előállított partícionálás
- $BE$
  - Feltétellánc $BE$, $CD$, $AE$, $CF$, $AB$
  - $\Pi_2 = (ABE)(CDF)$ az $AB$ állapotpár vizsgálatánál már előállított partícionálás
- $BF$
  - Feltétellánc $BF$, $EF$, $AD$, $AC$, $CE$, $BD$, $AF$, $BC$
  - $\Pi_1 = (ABCDEF)$ triviális partícionálás
- $CD$
  - Feltétellánc $CD$, $AE$, $CF$, $AB$
  - $\Pi_2 = (ABE)(CDF)$ az $AB$ állapotpár vizsgálatánál már előállított partícionálás
- $CE$
  - Feltétellánc $CE$, $AF$, $BC$, $AD$, $BD$, $EF$, $AC$
  - $\Pi_1 = (ABCDEF)$ triviális partícionálás
- $CF$
  - Feltétellánc $CF$, $AE$, $AB$, $CD$
  - $\Pi_2 = (ABE)(CDF)$ az $AB$ állapotpár vizsgálatánál már előállított partícionálás
- $DE$
  - Feltétellánc $DE$, $EF$, $BC$, $AC$, $AF$, $BD$, $EC$, $AD$
  - $\Pi_1 = (ABCDEF)$ triviális partícionálás
- $DF$
  - Feltétellánc $DF$, $AB$, $CF$, $AE$, $CD$
  - $\Pi_2 = (ABE)(CDF)$ az $AB$ állapotpár vizsgálatánál már előállított partícionálás
- $EF$
  - Feltétellánc $EF$, $AC$, $BD$
  - $\Pi_3 = (AC)(BD)(EF)$ az $AC$ állapotpár vizsgálatánál már előállított partícionálás

Az összes állapotpár megvizsgálása során összesen kettő nem triviális partícionálást sikerült előállítani, a $\Pi_2$ és a $\Pi_3$ partícionálásokat. Nézzük meg, hogy melyik partícionálás megvalósításához mennyi állapotváltozóra van szükség. A $\Pi_2 = (ABE)(CDF)$ partícionálás kettő partíciót tartalmaz, azaz ezek megkülönböztetéséhez egy darab önfüggő szekunder változóra van szükség, valamint mind a két partíció három állapotot tartalmaz, tehát a partíciókon belüli állapotok megkülönböztetéséhez kettő állapotváltozóra van szükség. A $\Pi_3 = (AC)(BD)(EF)$ partícionálás három partíciót tartalmaz, azaz ezek megkülönböztetéséhez kettő darab önfüggő szekunder változóra van szükség, valamint mind a három partíció kettő állapotot tartalmaz, tehát a partíciókon belüli állapotok megkülönböztetéséhez egy állapotváltozó elegendő. Ezen adatokat a következő táblázatban össze is foglaltuk.

| | Partíciók | Önfüggő szekunder | Maradék szekunder | Összesen |
|---|---|---|---|---|
| $\Pi_2$ | $(ABE)(CDF)$ | 2 partíció → 1 változó | 3 állapot → 2 változó | 3 változó |
| $\Pi_3$ | $(AC)(BD)(EF)$ | 3 partíció → 2 változó | 2 állapot → 1 változó | 3 változó |

Összefoglalva megállapíthatjuk, hogy mind a két partícionálás megvalósításához három darab szekunderváltozóra van szükség, ezért a közöttük való választás tetszőleges. Válasszuk ki a $\Pi_2$ partícionálást. Jelölje $Q_0$ az önfüggő szekunder változót és $Q_1$ és $Q_2$ a maradék szekunder változókat. Az önfüggő szekunder változó értéket úgy kell meghatározni, hogy az értéke egy partícióba tartozó állapotok esetén megegyezzen, de a különböző partíciókba tartozó változók esetén különbözzön. Példánkban legyen a $Q_0$ értéke $0$, az $A$, $B$ és $E$ változók esetén és legyen $1$, a $C$, $D$, és $F$ változók esetén. A maradék szekunder változók esetén arra kel figyelni, hogy az egy partícióhoz tartozó állapotokhoz különböző kombinációk tartozzanak. Így legyen a $Q_1Q_2$ értéke $A$ változó esetén $00$, $B$ esetén $01$ és $E$ esetén $11$, valamint $C$, $D$ és $F$ esetén ezek az értékeke legyenek rendre $00$, $01$ és $11$. Az alábbi táblázatban vonalakkal választottuk el egymástól a partíciókat valamint az önfüggő és a maradék szekunder változókat. Itt vizuálisan is jól látszik, hogy az egy partícióban lévő változóknál $Q_0$ értéke megegyezik, a $Q_1Q_2$ viszont különböző kombinációkat vesz fel.

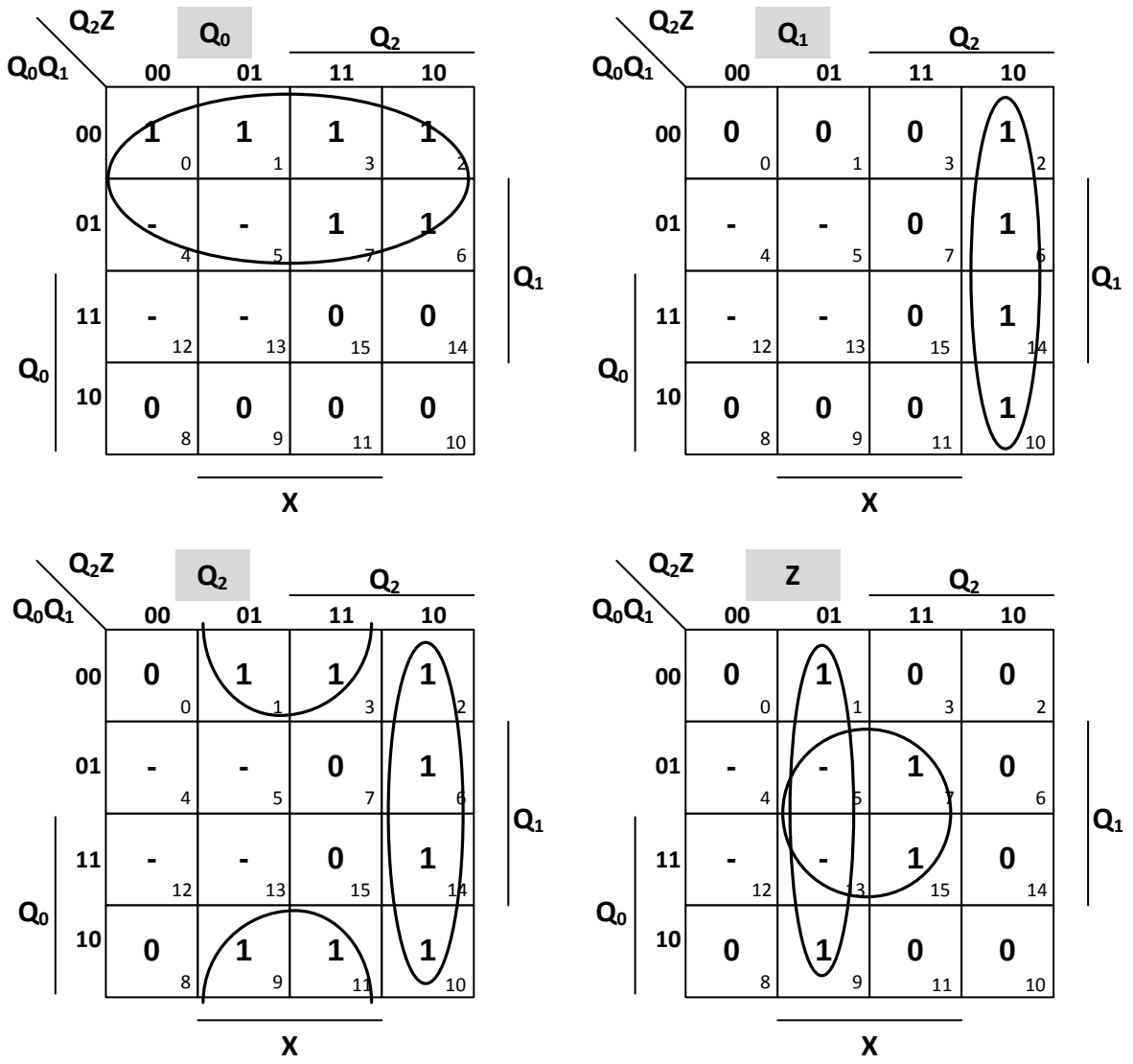
| | $Q_0$ | $Q_1$ | $Q_2$ |
|---|---|---|---|
| $A$ | 0 | 0 | 0 |
| $B$ | 0 | 0 | 1 |
| $E$ | 0 | 1 | 1 |
| $C$ | 1 | 0 | 0 |
| $D$ | 1 | 0 | 1 |
| $F$ | 1 | 1 | 1 |

A táblázatból kiolvashatóak az állapotokhoz tartozó kódok: $A \rightarrow 000$, $B \rightarrow 001$, $C \rightarrow 100$, $D \rightarrow 101$, $E \rightarrow 011$, $F \rightarrow 111$. Ezek alapján felírhatjuk a kódolt állapottáblát.

| $Q_0Q_1Q_2$ \ $X$ | 0 | 1 |
|---|---|---|
| **000** | **100 0** | **101 1** |
| **001** | **111 0** | **101 0** |
| **100** | **000 0** | **001 1** |
| **101** | **011 0** | **001 0** |
| **011** | **111 0** | **100 1** |
| **111** | **011 0** | **000 1** |
| **010** | **--- -** | **--- -** |
| **110** | **--- -** | **--- -** |

**8.29. ábra: Kódolt állapottábla**

Az egyszerűség és azt egységesség kedvéért használjunk ebben az esetben is mindenhol D tárolót, így az állapottáblából (8.29) közvetlenül felírhatóak a Karnaugh táblák és leolvashatók a vezérlőfüggvények és a kimenet függvénye.




**8.30. ábra: Vezérlő kombinációs hálózatok és a kimenet Karnaugh táblái**

$$Q_0 = \overline{Q_0}$$

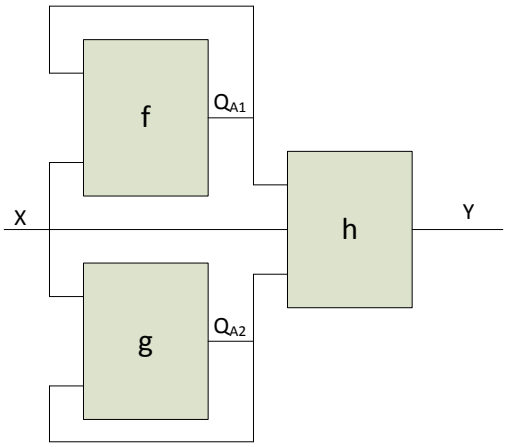
$$Q_1 = Q_2\bar{X}$$

$$Q_2 = Q_2\bar{X} + \overline{Q_1}X$$

$$Z = Q_1X + \overline{Q_2}X$$

Mivel az egyszerűsítés során közös hurkot is találtunk ($Q_2\bar{X}$), ezért a kapcsolás megvalósításához összes 10 kapubemenetre van szükség szemben a véletlenszerű kódolásnál használt 35-tel, valamint a szomszédos állapotkódok módszerénél kapott 14-gyel. A megoldásban az is megfigyelhető, hogy a $Q_0$ önfüggő szekunderváltozó tényleg nem függ a többi szekunderváltozótól, ahogy ennek a soros dekompozíciónál lennie kell.

A helyettesítési tulajdonságú partíciók használata esetében előfordulhat, hogy párhuzamos dekompozíciót is el tudunk érni. Ennek eléréshez egyszerre két partícionálást kell felhasználni. A két partícionálásra teljesülnie kell annak, hogy a két partícionálásból tetszőlegesen egy-egy partícióit kiválasztva legfeljebb egy közös állapotuk van. Ilyenkor azt mondjuk, hogy a két partícionálás ortogonális. Ha van két ilyen partíció, akkor gyakorlatilag két önfüggő szekunder változó csoportunk van, melyek nem függenek egymástól. Az így felépített hálózat struktúrája a következő ábrán látható módon néz ki.




**8.31. ábra: Megvalósítandó hálózat helyettesítési tulajdonságú partíció alapján (párhuzamos dekompozíció)**

Illusztratív példa

A példánkban előállított két partícionálást ellenőrizzük le, hogy ortogonálisak-e. Ezt a legegyszerűbb módon egy táblázat segítségével tehetjük meg, melynek sorai az egyik partícionálás partíciói az oszlopai pedig a másik partíciói. A táblázat celláiba pedig írjuk be, hogy mely állapotok közösek a partíciókban, azaz állítsuk elő a metszetüket. Ha minden cellába legfeljebb egy állapot kerül, akkor a két partícionálás ortogonális.

| | $(ABE)$ | $(CDF)$ |
|---|---|---|
| $(AC)$ | $A$ | $C$ |
| $(BD)$ | $B$ | $D$ |
| $(EF)$ | $E$ | $F$ |

A táblázat kitöltése után láthatjuk, hogy a két partícionálás ortogonális, ezért készíthetünk párhozamos dekompozíciót. A $\Pi_2$ két partíciót tartalmaz, ezért ezek megkülönböztetéséhez egy önfüggő szekunder változót ($Q_0$) vezetünk be. A $\Pi_3$ három partíciót tartalmaz, ezért ezek megkülönböztetéséhez két önfüggő szekunder változót ($Q_1Q_2$) vezetünk be. Ahhoz, hogy az előzőleg már ismertetett feltétel, miszerint az egy partícióba tarozó változók esetében az önfüggő változók legyenek ugyanazok, a különbözőbe tartozók esetén pedig legyenek különbözők, kiterjesztjük a táblázatunkat. Az oszlopokhoz hozzárendeljük a két partíciót megkülönböztető állapotváltozó két lehetséges állapotát, a sorokhoz pedig a három partíciót megkülönböztető állapotváltozók tetszőleges, de különböző kombinációit.

| $Q_1Q_2$ \ $Q_0$ | | 0 | 1 |
|---|---|---|---|
| | | $(ABE)$ | $(CDF)$ |
| 00 | $(AC)$ | $A$ | $C$ |
| 01 | $(BD)$ | $B$ | $D$ |
| 10 | $(EF)$ | $E$ | $F$ |

Ezek után az állapotokhoz tartozó kódok a táblázatból kiolvashatóak, csak össze kell fűzni az oszlophoz tartozó értéket a sorhoz tartozó értékkel. Ezt akár a táblázatba is be lehet írni a változók után, mint ahogy a következő táblázatban látható.

| $Q_1Q_2$ \ $Q_0$ | 0 | 1 |
|---|---|---|
| | $(ABE)$ | $(CDF)$ |

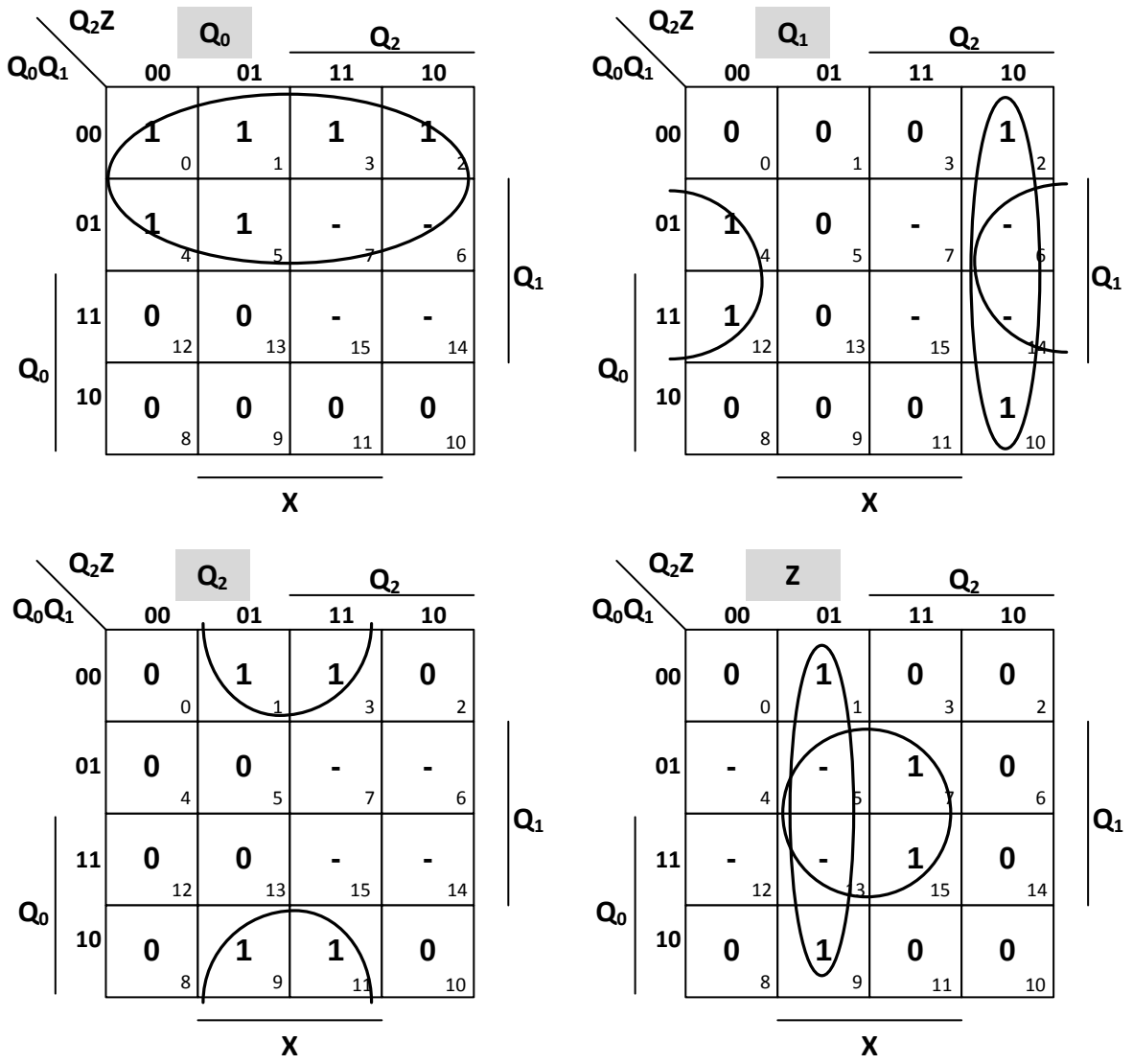
| | | | |
|---|---|---|---|
| 00 | $(AC)$ | $A$ (000) | $C$ (100) |
| 01 | $(BD)$ | $B$ (001) | $D$ (101) |
| 10 | $(EF)$ | $E$ (010) | $F$ (110) |

A táblázatból kiolvashatóak az állapotokhoz tartozó kódok: $A \rightarrow 000$, $B \rightarrow 001$, $C \rightarrow 100$, $D \rightarrow 101$, $E \rightarrow 010$, $F \rightarrow 110$. Ezek alapján felírhatjuk a kódolt állapottáblát.

| $Q_0Q_1Q_2$ \ $X$ | **0** | **1** |
|---|---|---|
| **000** | **100 0** | **101 1** |
| **001** | **110 0** | **101 0** |
| **100** | **000 0** | **001 1** |
| **101** | **010 0** | **001 0** |
| **010** | **110 0** | **100 1** |
| **110** | **010 0** | **000 1** |
| **011** | **--- -** | **--- -** |
| **111** | **--- -** | **--- -** |

**8.32. ábra: Kódolt állapottábla**

Az egyszerűség és azt egységesség kedvéért használjunk ebben az esetben is mindenhol D tárolót így az állapottáblából közvetlenül felírhatóak a Karnaugh táblák és leolvashatók a vezérlőfüggvények és a kimenet függvénye.




**8.33. ábra: Vezérlő kombinációs hálózatok és a kimenet Karnaugh táblái**

$$Q_0 = \overline{Q_0}$$

$$Q_1 = Q_2\bar{X} + Q_1\bar{X}$$

$$Q_2 = \overline{Q_1}X$$

$$Z = Q_1X + \overline{Q_2}X$$

Az egyszerűsítés során nem találtunk közös hurkokat, ezért a szükséges kapubemenetek száma 12-re adódik. A megoldásból látszik, hogy a két szekunderváltozó csoport nem függ egymástól, azaz a $Q_0$ értéke csak $Q_0$-tól, valamint $Q_1$ és $Q_2$ értéke csak $Q_1$-től és $Q_2$-től függ.

## *Kódolás kimenet alapján*

A gyakorlatban gyakran előfordul az a feltétel, hogy valamely szekunder változó/változók értéke közvetlenül adja meg a kimenet/kimenetek értékét. Ilyen esetekben a kimenetet előállító kombinációs hálózatot meg tudjuk spórolni, ezáltal olcsóbb lehet a kapcsolás. További előnye a megoldásnak, hogy a kimenet hazárdmentes lesz, mivel nem kell kapukon áthaladnia a jelnek a kimenetig.

Ez a feltétel gyakran nem teljesíthető könnyen, csak úgy, ha több szekunder változót vezetünk be, így nem biztos, hogy megéri ezt a megoldást választani. Például lehet olyan állapot a hálózatban,

amelynél több kimeneti kombináció is előfordulhat. Ebben az esetben egy állapothoz két kódot kell rendelni, azaz gyakorlatilag egy állapot helyett kettőt kell használni. Ha megvizsgáljuk az illusztratív példánk alábbi állapottábláját, akkor láthatjuk, hogy $A$ állapotba eljuthatunk $C$ állapotból és ekkor $0$ lesz a kimenet és eljuthatunk $F$ állapotba is, amikor $1$ a kimenet értéke. Hasonló probléma lép fel $B$, $C$ és $D$ állapot esetén is, azaz hat állapot helyet tíz állapotot kell kódolni.

| X / Y | 0 | 1 |
|---|---|---|
| A | C 0 | D 1 |
| B | F 0 | D 0 |
| C | A 0 | B 1 |
| D | E 0 | B 0 |
| E | F 0 | C 1 |
| F | E 0 | A 1 |

**8.34 ábra: Előzetes állapottábla**

A kimenet szerinti kódolás előírás bizonyos feladatoknál, mint például a számlálók.

## *Állapotonként egy bit kódolás*

Ez a módszer nem arra fekteti a hangsúlyt, hogy a lehető legegyszerűbb legyen a megtervezett hálózat, hanem arra, hogy a vezérlőfüggvények egyszerűek legyenek. Ez annak a rovására megy, hogy sokkal több állapotváltozót használunk, azaz sokkal több tárolót. Konkrétan minden állapothoz egy külön szekunder változó tartozik, amelynek értéke akkor $1$, ha az adott állapotban vagyunk, különben $0$. Ez azt jelenti, hogy mindig pontosan egy darab tárolóban van $1$ érték. Ezt a faja kódolást tipikusan FPGA áramkörök esetében szokták használni, amikor sok regiszter áll rendelkezésünkre, de a hozzá tartozó kombinációs hálózat bemenetinek száma viszonylag kevés.

A kódolás hátránya, hogy nagyon érzékeny a külső zavarokra, mivel rengeteg olyan állapotkód van, amely nem érvényes. Ha $n$ állapotunk van, akkor $n$ darab állapotváltozónk, amely segítségével $2^n$ állapotkódot lehet leírni és mi ezek közül csak $n$ darabot használunk. Ha a kapcsolásunk bármilyen zavar hatására olyan állapotba kerül, amely nem érvényes, akkor onnantól kezdve hibásan fog működni. Természetesen minden ilyen állapotból megfelelő tervezéssel vissza lehet vezetni a hálózatot érvényes állapotba, de ez a kapcsolást nagyon elbonyolítani, így pont a kódolás előnyét veszítenénk el.

Illusztratív példa

Nézzük meg, hogy milyen eredményeket tudunk elérni, ha a fejezet elején az alábbi állapottáblával definiált példánk kódolásához az állapotonkénti egy bit kódolás módszerét használjuk.

| X / Y | 0 | 1 |
|---|---|---|
| A | C 0 | D 1 |
| B | F 0 | D 0 |
| C | A 0 | B 1 |
| D | E 0 | B 0 |
| E | F 0 | C 1 |
| F | E 0 | A 1 |

**8.35. ábra: Előzetes állapottábla**

Mivel hat állapotunk van, ezért hat állapotváltozónk lesz, amelyeket az állapotok nevei alapján $Q_A$, $Q_B$, $Q_C$, $Q_D$, $Q_E$, $Q_F$ nevekkel jelölünk. Az állapotok kódjai ezek alapján $A \rightarrow 100000$, $B \rightarrow 010000$, $C \rightarrow 001000$, $D \rightarrow 000100$, $E \rightarrow 000010$ és $F \rightarrow 000001$.

A vezérlőfüggvények meghatározásához, fel lehetne írni a kódolt állapottáblát és utána egyszerűsíteni lehetne a függvényeket. Az egyszerűsítéshez jelen esetben a számjegyes minimalizálásra lenne szükségünk, mivel a 6 változós Karnaugh táblákat már nem könnyű kezelni. Mindezek helyett mi egy ennél egyszerűbb módot választunk a függvények felírására, közvetlenül az állapottáblából írjuk fel a függvényeket. A módszert a példán keresztül mutatjuk be.

$A$ állapotba akkor jutunk el, ha $C$ állapotba vagyunk és $X = 0$ a bemenet, ami azt jelenti, hogy $Q_A$ állapotváltozó vezérlőfüggvényében lesz egy $Q_C\bar{X}$ tag. $A$ állapotba $F$ állapotból is el tudunk jutni, ha $X = 1$ a bemenet, azaz egy $Q_FX$ tag is lesz a vezérlőfüggvényben. Mivel másik állapotból nem lehet eljutni $A$ állapotba, ezért a vezérlőfüggvényt a két meghatározott tag VAGY kapcsolata alkotja, azaz $Q_A = Q_C\bar{X} + Q_FX$.

$B$ állapotba akkor jutunk el, ha $C$ állapotba vagyunk és $X = 1$ a bemenet, ami azt jelenti, hogy $Q_B$ állapotváltozó vezérlőfüggvényében lesz egy $Q_CX$ tag. $B$ állapotba $D$ állapotból is el tudunk jutni, ha $X = 1$ a bemenet, azaz egy $Q_DX$ tag is lesz a vezérlőfüggvényben. Mivel másik állapotból nem lehet eljutni $B$ állapotba, ezért a vezérlőfüggvényt a két meghatározott tag VAGY kapcsolata alkotja, azaz $Q_B = Q_CX + Q_DX$.

$C$ állapotba akkor jutunk el, ha $A$ állapotba vagyunk és $X = 0$ a bemenet, ami azt jelenti, hogy $Q_C$ állapotváltozó vezérlőfüggvényében lesz egy $Q_A\bar{X}$ tag. $C$ állapotba $E$ állapotból is el tudunk jutni, ha $X = 1$ a bemenet, azaz egy $Q_EX$ tag is lesz a vezérlőfüggvényben. Mivel másik állapotból nem lehet eljutni $C$ állapotba, ezért a vezérlőfüggvényt a két meghatározott tag VAGY kapcsolata alkotja, azaz $Q_C = Q_A\bar{X} + Q_EX$.

Az összes állapotot hasonlóképpen megvizsgálva a következő vezérlőfüggvényeket kapjuk.

$$Q_A = Q_C\bar{X} + Q_FX$$
$$Q_B = Q_CX + Q_DX$$
$$Q_C = Q_A\bar{X} + Q_EX$$
$$Q_D = Q_AX + Q_BX$$
$$Q_E = Q_D\bar{X} + Q_F\bar{X}$$
$$Q_F = Q_B\bar{X} + Q_E\bar{X}$$

A kimenet vezérlőfüggvényét is közvetlenül az állapottáblából írjuk fel az állapotváltozók vezérlőfüggvényeinek analógiájára. Azaz megnézzük, hogy milyen állapotokban milyen bemenet hatására lesz a kimenet 1 és az ehhez tartozó logikai kifejezések VAGY kapcsolata adja a kimenet vezérlőfüggvényét. A példánk kimeneti vezérlőfüggvénye lent látható.

$$Y = Q_A X + Q_C X + Q_E X + Q_F X$$

Mint a példából is látható a kapcsolásunk 42 kapubemenetet tartalmaz, azaz még a véletlenszerű kódolásnál is többet, de ezzel a módszerrel minden vezérlőfüggvényben biztosan csak kétbemenetű ÉS kapukra van szükség. Így FPGA segítségével az ilyen jellegű kódolások könnyen megvalósíthatók.

# 9. Aszinkron sorendi hálózatok állapotkódolása

A fejezetben az aszinkron hálózatok állapotkódolásáról lesz szó, azaz ugyanúgy, mint az előző 8. fejezetben meg kel határozni, hogy a minimalizált állapottáblában lévő állapotokhoz a belső változók milyen kombinációja tartozzon és egyáltalán hány belső változóra van szükségünk. A célunk viszont a szinkron hálózatok állapotkódolásához képest nem a legegyszerűbb (legolcsóbb) hálózat elérése, hanem az, hogy a hálózatban ne legyen kritikus versenyhelyzet. Ez azért fontos, mert ezen nem a megvalósítandó áramkör ára, hanem működőképessége múlik.

A kritikus versenyhelyzet részletes ismertetésétől most eltekintünk, mivel az már megtörtént az aszinkron hálózatok tervezéséről szóló fejezetben. A probléma azon alapul, hogy aszinkron hálózatokban két érték soha nem változhat pontosan egyszerre és egy állapotváltás során több belső változó változhat egyszerre, akkor a hálózati elemek működési sebességétől függ, hogy melyik változik előbb. Ilyenkor előfordulhat, hogy átkerülünk egy olyan állapotkódba, amely egy másik állapothoz tartozik és ennek az állapotnak a hatása természetesen azonnali a hálózaton és ez egy másik állapotba való átmenetelt indít el. Így az általunk kívánt állapotátmenet helyett egy másik következhet be, azaz a hálózat nem a specifikációnak megfelelően működik.

Az aszinkron hálózatok kódolására két módszert mutatunk be, amelyek közül az első (instabil állapotok beillesztése) tapasztalatot, intuíciót igényel, és nem feltétlenül vezet optimális megoldáshoz. Ezzel szemben a második módszer (Tracey-Unger módszer) algoritmikus és így akár számítógépen is megvalósítható.

## *Instabil állapotok beillesztése*

A legegyszerűbb az lenne, ha a szekunder változók értékeit úgy választanánk meg, hogy minden állapotváltozás esetében azok csak egyetlen változóban különbözzenek egymástól, azaz szomszédosak legyenek. Így elkerülhető lenne az, hogy több változó változzon egyszerre, azaz kritikus versenyhelyzet sem léphetne fel. Ezt azonban nem minden esetben lehet biztosítani. Már az előző fejezetben is láttuk a szomszédos állapotkódok választásának módszerénél, hogy lehetnek olyan szomszédsági feltételek, amelyeket nem lehet egyszerre kielégíteni.

Az ilyen problémákat úgy lehet megoldani, hogy új állapotokat veszünk fel, melyek bevezetéséhez szükség esetén növeljük a szekunder változók számát, azaz a korábban bemutatott $n = \lceil \log_2 N \rceil$ képlettel meghatározott változószámnál több változóra van szükség ahhoz, hogy a kritikus versenyhelyzetet el tudjuk kerülni. Az újonnan bevezetett állapotok instabilak lesznek, ami azt jelenti, hogy a kapcsolás felveszi az állapothoz tartozó állapotkódot, de rögtön át is vált egy másik stabil állapotba anélkül, hogy a bemenet megváltozna. Természetesen úgy kell az instabil állapotokat bevezetni, hogy a hálózat eredeti működése zavartalan maradjon, azaz nem változhat meg a stabil állapotok egymás utáni sorrendje. Továbbá az új instabil állapotok kódolásánál is ügyelni kell arra, hogy a szomszédos állapotok kódjától csak egyetlen szekunder változóban különbözzenek.

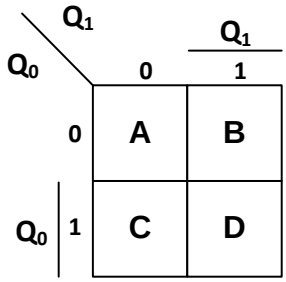
Példa

Tekintsük a következő állapottáblát, amelyben a kimenet értékeitől eltekinthetünk, mivel az nem befolyásolja az állapotok szomszédosságát.

| $X_1X_2$ / $Y$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| ***A*** | ***A*** | ***B*** | ***C*** | ***D*** |
| ***B*** | ***-*** | ***B*** | ***-*** | ***D*** |
| ***C*** | ***A*** | ***-*** | ***C*** | ***D*** |
| ***D*** | ***A*** | ***-*** | ***-*** | ***D*** |

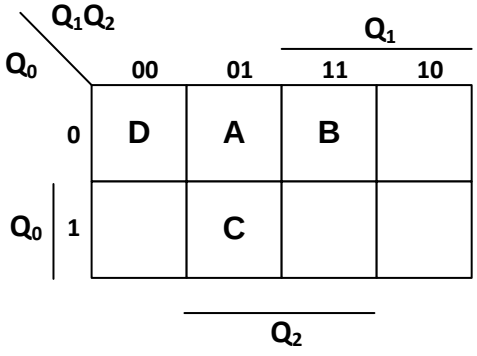
**9.1 ábra: Állapottábla (adott)**

Az előző 8. fejezetben bemutatott szomszédos állapotkódok választása „a” szabályánál használt módszerhez hasonlóan össze lehet gyűjteni a szomszédossági feltételeket. Ez alapján szomszédosnak kell lennie az $AB$, $AC$, $AD$, $BD$ és $CD$ állapotpároknak. Mivel négy állapotunk van, ezért az állapotok kódolásához kettő állapotváltozó elegendő ($Q_0Q_1$), azaz egy kétváltozós Karnaugh táblával megpróbálhatjuk teljesíteni a szomszédsági feltételeket. Ha teljesítjük az $AB$, $AC$ feltételeket, akkor két változó segítségével az $AD$ feltétel már nem teljesíthető. A $BD$ és $CD$ feltételeket ettől függetlenül lehet még teljesíteni, mint ahogy az alábbi 9.2 ábrán látható is, de ebben az esetben nem lesz versenyhelyzet mentes a hálózat.




**9.2 ábra: Karnaugh tábla a szomszédossági feltételek vizsgálatára**

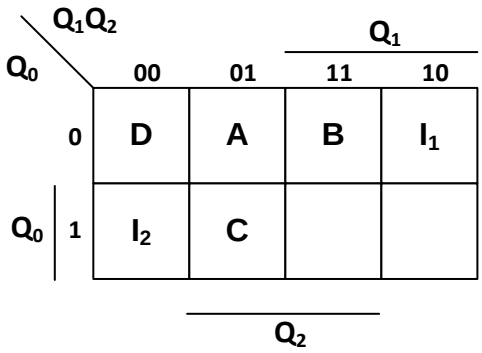
Most próbáljunk meg még egy szekunder változót ($Q_2$) bevezetni és úgy teljesíteni a szomszédsági feltételeket. Ebben az esetben már egy háromváltozós Karnaugh táblázatra van szükségünk. Ebben már teljesíteni tudjuk az $A$ változóhoz tartozó összes szomszédsági feltételt, mint az az alábbi 9.3 ábrán látható.




**9.3 ábra: Nagyobb Karnaugh tábla a szomszédossági feltételek vizsgálatára**

Még két feltételt kell teljesíteni a kritikus versenyhelyzet kizárásához, a $BD$ és a $CD$ állapotpároknak kell szomszédosnak lenniük. Mint a fenti ábrán is látható, ezt közvetlenül nem lehet teljesíteni, hanem szükség van két új instabil állapotra. A $BD$ pár szomszédosságához bevezetünk egy $I_1$

állapotot, melyet a Karnaugh táblában a $B$ és $D$ állapot közé helyezünk el, valamint egy $I_2$ instabil állapot kerül a $C$ és $D$ állapotok közé. A kiegészített Karnaugh tábla a következő 9.4-es ábrán látható.



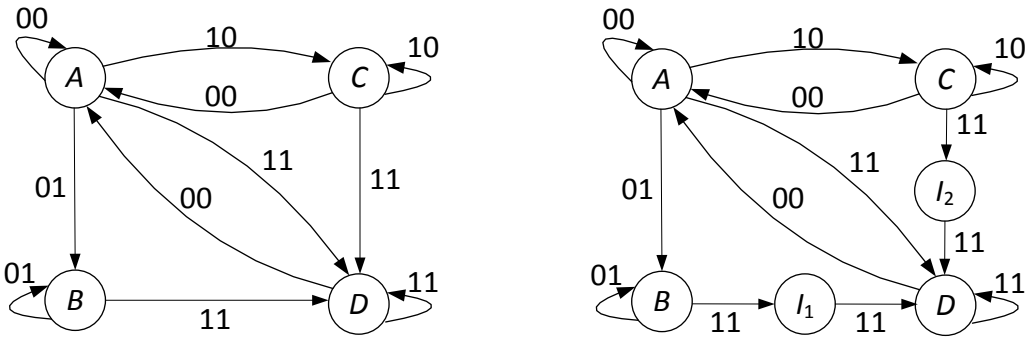

**9.4 ábra: Karnaugh tábla új instabil állapotokkal**

A Karnaugh táblából a korábban ismertettet módszerek segítségével leolvashatóak az állapotok kódjai. $A \rightarrow 001$, $B \rightarrow 011$, $C \rightarrow 101$, $D \rightarrow 000$, $I_1 \rightarrow 010$, $I_2 \rightarrow 100$. Látható, hogy az $A$ állapot kódja valóban csak $Q_1$ változóban különbözik a $B$, $Q_0$ változóban a $C$, és $Q_1$ változóban a $D$ állapot kódjától. Viszont az új állapotok nem szerepelnek az állapottáblában, azért ezeket bele kell őket illeszteni. Mivel az $I_1$ állapotot a $B$ és $D$ állapot közé raktuk be, ezért $B$ állapotból nem jutunk el közvetlenül $D$ állapotba, hanem először $I_1$ állapotba kerülünk és onnan ugyanazon bemenet hatására rögtön tovább a $D$ állapotba. Hasonlóképpen $C$ állapotból $D$ állapot helyett $I_2$ állapotba kerülünk és onnan tovább $D$-be. A módosított állapottábla a következő 9.5-ös ábrán látható.

| $X_1X_2$ / $Y$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| ***A*** | ***A*** | ***B*** | ***C*** | ***D*** |
| ***B*** | - | ***B*** | - | $I_1$ |
| ***C*** | ***A*** | - | ***C*** | $I_2$ |
| ***D*** | ***A*** | - | - | ***D*** |
| $I_1$ | - | - | - | ***D*** |
| $I_2$ | - | - | - | ***D*** |

**9.5 ábra: Módosított állapottábla**

A jobb érthetőség kedvéért érdemes felrajzolni az eredeti és az instabil állapotokkal kiegészített állapotgráfot. A gráfokon az állapottáblához hasonlóan nem jelöltük a kimeneteket, mivel az nem befolyásolja az állapotkódolást. Az 9.6-os ábrán látható, hogy az új állapotok azon állapotok közé kerültek be, ahova a Karnaugh táblába is beraktuk őket, azaz az $I_1$ a $B$ és $D$ közé, az $I_2$ pedig a $C$ és $D$ közé. Az is megfigyelhető, hogy sem az $I_1$ sem az $I_2$ állapot nem stabilizálódik, mivel az 11 bemenet hatására vált ezekbe az állapotokba a hálózat, de ugyanennek a bemenetnek a hatására már tovább is lép $D$ állapotba.




**9.6 ábra: Állapotgráf és kiterjesztett állapotgráf**

## *Tracey-Unger módszer*

A következő módszer nem azt próbálja elérni, hogy a szomszédos állapotok szomszédos kódokat kapjanak, hanem egy előzetes vizsgálat során megnézi, hogy mely állapotátmeneteknél milyen más állapotba juthat a rendszer a versenyhelyzet miatt. Ezeket az állapotokat nevezzük hazárdállapotnak, mivel a hazárdjelenség hatására juthat el ide a rendszer. Ezek után úgy osztja ki a kódokat, hogy az kiinduló és a cél állapotok kódjában legyen egy olyan bit, amelyben ők egyformák, viszont különböznek a hazárdállapotok kódjától. Ezzel azt lehet elérni, hogy hiába nem szomszédosak az állapotkódok, nem fog egy hazárdállapotba átmenni a rendszer, mivel azok a szekunder változók, amelyben egyforma a két állapotkód, nem fognak megváltozni, és így nem tudja felvenni egyik hazárdállapot kódját sem.

A hazárdállapotok felismeréséhez az állapottábla nyújt segítséget. Mivel kritikus versenyhelyzet akkor fordul elő, amikor egy stabil állapotból egy másikba szeretnénk átmenni, ezért a stabil állapotokból kell kiindulni, azaz azokból a cellákból, amelyekben ugyanaz az állapot szerepel, amelyikhez a sor tartozik. Ez után meg kell vizsgálni, hogy ha a bemenet megváltozik, akkor milyen új (stabil) állapotba szeretnénk kerülni, azaz az új bemenet oszlopában és a változó sorában milyen állapot szerepel. A lehetséges hazárd állapotok vagy más néven leskelődők ugyanebben az oszlopban fognak szerepelni és stabil állapotok lesznek, mivel itt van meg az esély arra, hogy másik állapotba kerülünk, mint amelyikbe szeretnénk és ez az állapot stabilizálódhat is a hálózatban. Ezzel a módszerrel az összes stabil-stabil állapotátmenethez meg lehet határozni a leskelődőket.

A feltevésünk szerint az állapotátmenet két állapotkódja egy bitben meg fog egyezni egymással és ez különbözni fog a leskelődőktől, ezért minden stabil-stabil állapotátmenethez összegyűjtöttük a leskelődőket és ezek annyi kódolási szabályt jelentenek számunkra, ahány állapotátmenethez találtunk leskelődőt. Arra viszont figyelni kell, hogy ugyanazok a feltételek többször is generálódhatnak a keresés során. Ebben az esetben a redundanciát meg kel szüntetni, azaz a többszörös feltételeket el kell távolítani. Itt arra is kell figyelni, hogy egy állapotátmenet visszafelé is szerepelhet ugyanolyan leselkedőkkel. Ekkor is eltávolíthatjuk az egyiket, hiszen mind a két feltétel ugyanolyan kódolási szabályt jelent.

Illusztratív példa

Legyen egy aszinkron hálózatunk két bemenettel és öt állapottal, melynek alábbi állapottáblájában bejelöltük a stabil állapotokat, a könnyebb láthatóság miatt.

| $X_1X_2$ / $Y$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| **A** | - - | **B** 0 | **C** 0 | (**A** 0) |
| **B** | **D** 1 | (**B** 0) | - - | **A** 0 |
| **C** | **E** 0 | - - | (**C** 0) | **A** 0 |
| **D** | (**D** 1) | **B** 0 | **C** 0 | - - |
| **E** | (**E** 0) | **B** 0 | **C** 0 | - - |

**9.7 ábra: Állapottábla (illusztratív példa)**

Egy táblázatban fogjuk gyűjteni a megtalált hazárdállapotokat, ahol a bemenet változása alapján fogjuk őket csoportosítani. Mivel egy aszinkron hálózatról beszélünk, ezért az alapfeltevésünk miatt nem változhat egyszerre két bemeneti változó, azaz nincs $00 \to 11$, $01 \to 10$, $10 \to 01$ és $11 \to 00$ változás. Ezek alapján a következő táblázatot kell kitölteni.

| $00 \to 01$ | $00 \to 10$ | $01 \to 00$ | $01 \to 11$ | $10 \to 00$ | $10 \to 11$ | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

- Stabil állapot az $A$ állapot $11$ bemenet esetén.
  - A bemenet megváltozhat $01$-re és ekkor a hálózat $B$ állapotba kell, hogy átváltson. Mivel a $01$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.
  - A bemenet megváltozhat $10$-ra és ekkor a hálózat $C$ állapotba kell, hogy átváltson. Mivel az $10$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.
- Stabil állapot a $B$ állapot $01$ bemenet esetén.
  - Ha a bemenet $00$-ra változik, akkor a hálózat $D$ állapotba kell, hogy kerüljön. A $00$ oszlopban stabil állapot még az $E$ is, ezért ez egy leselkedő hazárdállapot. Ezt az átmenetet és a hozzá tartozó leselkedőt ($B \to D$, $E$) feljegyezzük a táblázat $01 \to 00$ oszlopába.
  - A bemenet megváltozhat $11$-re és ekkor a hálózat $A$ állapotba kell, hogy átváltson. Mivel az $11$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.

| $00 \to 01$ | $00 \to 10$ | $01 \to 00$ | $01 \to 11$ | $10 \to 00$ | $10 \to 11$ | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|
| | | $B \to D \quad E$ | | | | | |

- Stabil állapot a $C$ állapot $10$ bemenet esetén.
  - Ha a bemenet $00$-ra változik, akkor a hálózat $E$ állapotba kell, hogy kerüljön. A $00$ oszlopban stabil állapot még az $D$ is, ezért ez egy leselkedő hazárdállapot. Ezt az átmenetet és a hozzá tartozó leselkedőt ($C \to E$, $D$) feljegyezzük a táblázat $10 \to 00$ oszlopába.
  - A bemenet megváltozhat $11$-re és ekkor a hálózat $A$ állapotba kell, hogy átváltson. Mivel az $11$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.

| $00 \to 01$ | $00 \to 10$ | $01 \to 00$ | $01 \to 11$ | $10 \to 00$ | $10 \to 11$ | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|
| | | $B \to D \quad E$ | | $C \to E \quad D$ | | | |

- Stabil állapot a $D$ állapot $00$ bemenet esetén.

    - A bemenet megváltozhat $01$-re és ekkor a hálózat $B$ állapotba kell, hogy átváltson. Mivel a $01$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.
    - A bemenet megváltozhat $10$-ra és ekkor a hálózat $C$ állapotba kell, hogy átváltson. Mivel az $10$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.
- Stabil állapot az $E$ állapot $00$ bemenet esetén.
    - A bemenet megváltozhat $01$-re és ekkor a hálózat $B$ állapotba kell, hogy átváltson. Mivel a $01$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.
    - A bemenet megváltozhat $10$-ra és ekkor a hálózat $C$ állapotba kell, hogy átváltson. Mivel az $10$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.

Most nézzük meg, hogy a feltételekből hogyan lesz konkrét kód. Mivel egy szabály azt mondja ki, hogy az átmenet két állapotának kódja egy állapotváltozó szempontjából azonosak és különböznek a leselkedők kódjától, ezért logikusan minden szabályhoz rendelhetünk egy szekunder változót, amelyben ezek különbözni fognak egymástól. Ahhoz, hogy az állapotváltozó teljesítse a szabályt, úgy kell megválasztani az állapotait, hogy egyformák legyenek az átmenet két állapotára és ezek különbözzenek a hazárdkódoktól. Ezzel csak azt kötöttük meg, hogy különbözőek legyenek, de azt nem definiáltuk, hogy melyik esetben vesz fel 0 és mikor 1 értéket. Fontos, hogy ha egy állapotra nem ír elő semmit az adott szabály, akkor oda közömbös megjegyzést tehetünk, azaz számunkra mellékes, hogy a többi változó ebben az állapotváltozóban különbözik-e az átmenet állapotaitól vagy sem.

Egyszerűen belátható, hogy bármilyen értékeket választunk a közömbös bejegyzésekhez mindenképpen kritikus versenyhelyzettől mentes kódot kapunk. Az egyetlen probléma az, hogy nagyon sok szekunderváltozóra van szükség az összes szabály kielégítéséhez. Az változók számát csökkenteni tudjuk úgy, hogy összevonunk változókat kihasználva a közömbös bejegyzéseket. Két állapot összevonható, hogy ha értékük megegyezik minden olyan állapotnál, ahol mind a kettő definiált. Ebből következik, hogy a többi állapotnál vagy az egyik vagy mind a két állapothoz közömbös bejegyzés tartozik. Az összevont változót megegyezik a változók értéke közül azzal, amelyiknél nem közömbös bejegyzés szerepel, ha pedig mind a két változónál közömbös bejegyzés szerepel, akkor az összevont állapothoz is közömbös bejegyzést veszünk fel.

Olyan eset is előfordulhat, hogy nem tudunk összevonni állapotot, de ha az egyikhez tartozó bejegyzéseket invertáljuk (az 1-eseket és 0-ákat felcseréljük), akkor már összevonhatóvá válnak. Ezt szabadon megtehetjük, mivel ezzel nem sértjük meg a szabályt, mivel az invertálás nem változtatja meg a különbözőséget. Úgy vehetjük észre az ilyen jellegű összevonás lehetőségét, hogy a normál összevonással ellentétben a változók értéke nem megegyezik, hanem különbözik minden olyan állapotnál, ahol mind a két állapot definiált.

Fontos megjegyezni, hogy a módszer nem biztosítja azt, hogy minden állapotnak külön kódja legyen. Ha például csak egyetlen leselkedőnk (szabályunk) van, akkor a módszer alapján egyetlen változóval lehet már kritikus versenyhelyzet mentes kódot adni. A valóságban ez azt jelenti, hogy ha a megoldásunkban van egy olyan állapotváltozó, amely értéke megegyezik a generált állapotváltozó értékével, akkor nem lesz versenyhelyzet a kapcsolásunkban.

Illusztratív példa

Az elkészült táblázatot vizsgáljuk meg a redundancia miatt, azaz keressünk olyan feltételeket, amelyek többször is szerepelnek akár fordított átmeneti iránnyal. Ilyen szabályokat nem találunk, ezért mind a két szabályt felhasználjuk a kódgeneráláshoz. Mivel kettő szabályunk van, ezért első lépésben feltételezzük, hogy kettő állapotváltozónk lesz, azaz az alábbi táblázatot kell kitölteni.

| | $Y_1$ | $Y_2$ |
|---|---|---|
| $A$ | | |
| $B$ | | |
| $C$ | | |
| $D$ | | |
| $E$ | | |

Egyesével vizsgáljuk meg a szabályokat:

- $B \to D$ átmenet $E$ leselkedővel
  - A szabály szerint a $B$ és a $D$ állapotok kódja egy változóban megegyezik egymással és különbözik $E$ kódjától. Ez a változó legyen most az $Y_1$ és mondjuk azt, hogy $B$ és $D$ kódja ebben a változóban $0$, $E$ kódja pedig $1$. A szabály nem mond semmit $A$ és $C$ változókról, azaz oda közömbös bejegyzéseket tehetünk.

| | $Y_1$ | $Y_2$ |
|---|---|---|
| $A$ | $-$ | |
| $B$ | $0$ | |
| $C$ | $-$ | |
| $D$ | $0$ | |
| $E$ | $1$ | |

- $C \to E$ átmenet $D$ leselkedővel
  - A szabály szerint a $C$ és a $E$ állapotok kódja egy változóban megegyezik egymással és különbözik $D$ kódjától. Ez a változó legyen most az $Y_2$ és mondjuk azt, hogy $C$ és $E$ kódja ebben a változóban $1$, $D$ kódja pedig $0$. A szabály nem mond semmit $A$ és $B$ változókról, azaz oda közömbös bejegyzéseket tehetünk.

| | $Y_1$ | $Y_2$ |
|---|---|---|
| $A$ | $-$ | $-$ |
| $B$ | $0$ | $-$ |
| $C$ | $-$ | $1$ |
| $D$ | $0$ | $0$ |
| $E$ | $1$ | $1$ |

A táblázatban könnyen észrevehető, hogy $Y_1$ és $Y_2$ változók összevonhatóak, mivel az $A$ változónál mindegyik állapotváltozónál közömbös bejegyzés szerepel, $B$ változónál $Y_2$ értéke közömbös, $C$ változónál $Y_1$ értéke közömbös, $D$ változónál mindkét változó értéke $0$, $E$ változónál pedig mind a kettő állapotváltozó értéke $1$. Az összevont változót jelöljük $Y_{12}$-vel, melynek értéke $A$ állapotnál $-$, mivel $Y_1 = Y_2 = -$, $B$ állapotnál $0$, mivel $Y_1 = 0$, $C$ állapotnál $1$ mivel $Y_2 = 1$, $D$ állapotnál $0$, mivel $Y_1 = Y_2 = 0$, valamint $E$ állapotnál $1$, mivel $Y_1 = Y_2 = 1$.

| | $Y_{21}$ |
|---|---|
| $A$ | $-$ |
| $B$ | $0$ |
| $C$ | $1$ |
| $D$ | $0$ |

| $E$ | 1 |
|---|---|

Amit látjuk a módszer szerint egy változó elég ahhoz, hogy a kritikus versenyhelyzetet el tudjuk kerülni a megvalósított hálózatban, de ahhoz nem elegendő, hogy az állapotokat megkülönböztessük egymástól. Ehhez további két állapotváltozó bevezetésére lenne szükség.

Példa

Legyen egy aszinkron hálózatunk két bemenettel és öt állapottal, amelynek alábbi állapottáblájában (9.8 ábra) bejelöltük a stabil állapotokat, a könnyebb láthatóság miatt.

| $Y$ \ $X_1X_2$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| A | (A 0) | B 0 | C 0 | (A 0) |
| B | D 1 | (B 0) | (B -) | A 0 |
| C | A 0 | - - | (C 0) | A 0 |
| D | (D 1) | B 0 | C 0 | E - |
| E | (E 0) | B 0 | C 0 | (E -) |

**9.8 ábra: Állapottábla**

Gyűjtsük ki a táblázatból a stabil-stabil állapotátmenetekhez a lehetséges leskelődőket. Kezdjük el megvizsgálni az állapotátmeneteket minden stabil állapotból kiindulva.

- Stabil állapot az $A$ állapot $00$ bemenet esetén.
  - A bemenet megváltozhat $01$-re és ekkor a hálózat $B$ állapotba kell, hogy átváltson. Mivel a $01$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.
  - Ha a bemenet $10$-ra változik, akkor a hálózat $C$ állapotba kell, hogy kerüljön. Az $10$ oszlopban stabil állapot még a $B$ is, ezért ez egy leselkedő hazárdállapot. Ezt az átmenetet és a hozzá tartozó leselkedőt ($A \rightarrow C$, $B$) feljegyezzük a táblázat $00 \rightarrow 10$ oszlopába.

| $00 \rightarrow 01$ | $00 \rightarrow 10$ | $01 \rightarrow 00$ | $01 \rightarrow 11$ | $10 \rightarrow 00$ | $10 \rightarrow 11$ | $11 \rightarrow 01$ | $11 \rightarrow 10$ |
|---|---|---|---|---|---|---|---|
| | $A \rightarrow C \quad B$ | | | | | | |

- Stabil állapot az $A$ állapot $11$ bemenet esetén.
  - A bemenet megváltozhat $01$-re és ekkor a hálózat $B$ állapotba kell, hogy átváltson. Mivel a $01$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.
  - Ha a bemenet $10$-ra változik, akkor a hálózat $C$ állapotba kell, hogy kerüljön. Az $10$ oszlopban stabil állapot még a $B$ is, ezért ez egy leselkedő hazárdállapot. Ezt az átmenetet és a hozzá tartozó leselkedőt ($A \rightarrow C$, $B$) feljegyezzük a táblázat $11 \rightarrow 10$ oszlopába.

| $00 \rightarrow 01$ | $00 \rightarrow 10$ | $01 \rightarrow 00$ | $01 \rightarrow 11$ | $10 \rightarrow 00$ | $10 \rightarrow 11$ | $11 \rightarrow 01$ | $11 \rightarrow 10$ |
|---|---|---|---|---|---|---|---|
| | $A \rightarrow C \quad B$ | | | | | | $A \rightarrow C \quad B$ |

- Stabil állapot a $B$ állapot $01$ bemenet esetén.
  - A bemenet megváltozhat $00$-ra és ekkor a hálózat $D$ állapotba kell, hogy átváltson. Mivel a $00$ két másik stabil állapot ($A$, $E$) is van, ezért mind a kettő leselkedő állapot.

A táblázat $01 \to 00$ oszlopába feljegyezzük ezt az átmenetet mind a két leselkedővel együtt ($B \to D, A, E$).

| $00 \to 01$ | $00 \to 10$ | $01 \to 00$ | $01 \to 11$ | $10 \to 00$ | $10 \to 11$ | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|
| | $A \to C \quad B$ | $B \to D \quad A$<br>$E$ | | | | | $A \to C \quad B$ |

- Ha a bemenet $11$-ra változik, akkor a hálózat $A$ állapotba kell, hogy kerüljön. Az $11$ oszlopban stabil állapot még az $E$ is, ezért ez egy leselkedő hazárdállapot. Ezt az átmenetet és a hozzá tartozó leselkedőt ($B \to A, E$) feljegyezzük a táblázat $01 \to 11$ oszlopába.

| $00 \to 01$ | $00 \to 10$ | $01 \to 00$ | $01 \to 11$ | $10 \to 00$ | $10 \to 11$ | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|
| | $A \to C \quad B$ | $B \to D \quad A$<br>$E$ | $B \to A \quad E$ | | | | $A \to C \quad B$ |

- Stabil állapot a $B$ állapot $10$ bemenet esetén.
  - A bemenet megváltozhat $00$-ra és ekkor a hálózat $D$ állapotba kell, hogy átváltson. Mivel a $00$ két másik stabil állapot ($A, E$) is van, ezért mind a kettő leselkedő állapot. A táblázat $10 \to 00$ oszlopába feljegyezzük ezt az átmenetet mind a két leselkedővel együtt ($B \to D, A, E$).

| $00 \to 01$ | $00 \to 10$ | $01 \to 00$ | $01 \to 11$ | $10 \to 00$ | $10 \to 11$ | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|
| | $A \to C \quad B$ | $B \to D \quad A$<br>$E$ | $B \to A \quad E$ | $B \to D \quad A$<br>$E$ | | | $A \to C \quad B$ |

- Ha a bemenet $11$-re változik, akkor a hálózat $A$ állapotba kell, hogy kerüljön. Az $11$ oszlopban stabil állapot még az $E$ is, ezért ez egy leselkedő hazárdállapot. Ezt az átmenetet és a hozzá tartozó leselkedőt ($B \to A, E$) feljegyezzük a táblázat $10 \to 11$ oszlopába.

| $00 \to 01$ | $00 \to 10$ | $01 \to 00$ | $01 \to 11$ | $10 \to 00$ | $10 \to 11$ | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|
| | $A \to C \quad B$ | $B \to D \quad A$<br>$E$ | $B \to A \quad E$ | $B \to D \quad A$<br>$E$ | $B \to A \quad E$ | | $A \to C \quad B$ |

- Stabil állapot a $C$ állapot $10$ bemenet esetén.
  - A bemenet megváltozhat $00$-ra és ekkor a hálózat $A$ állapotba kell, hogy átváltson. Mivel a $00$ két másik stabil állapot ($D, E$) is van, ezért mind a kettő leselkedő állapot. A táblázat $10 \to 00$ oszlopába feljegyezzük ezt az átmenetet mind a két leselkedővel együtt ($C \to A, D, E$).

| $00 \to 01$ | $00 \to 10$ | $01 \to 00$ | $01 \to 11$ | $10 \to 00$ | $10 \to 11$ | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|
| | $A \to C \quad B$ | $B \to D \quad A$<br>$E$ | $B \to A \quad E$ | $B \to D \quad A$<br>$E$<br>$C \to A \quad D$<br>$E$ | $B \to A \quad E$ | | $A \to C \quad B$ |

- Ha a bemenet $11$-re változik, akkor a hálózat $A$ állapotba kell, hogy kerüljön. Az $11$ oszlopban stabil állapot még az $E$ is, ezért ez egy leselkedő hazárdállapot. Ezt az átmenetet és a hozzá tartozó leselkedőt ($C \to A, E$) feljegyezzük a táblázat $10 \to 11$ oszlopába.

| $00 \to 01$ | $00 \to 10$ | $01 \to 00$ | $01 \to 11$ | $10 \to 00$ | $10 \to 11$ | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $A \to C$ | $B$ | $B \to D$ | $A$ | $B \to A$ | $E$ | $B \to D$ | $A$ | $B \to A$ | $E$ | | | $A \to C$ | $B$ |
| | | | | $E$ | | | | $E$ | | | | | | |
| | | | | | | | $C \to A$ | $D$ | $C \to A$ | $E$ | | | | |
| | | | | | | | | $E$ | | | | | | |

- Stabil állapot a $D$ állapot $00$ bemenet esetén.
  - A bemenet megváltozhat $01$-re és ekkor a hálózat $B$ állapotba kell, hogy átváltson. Mivel a $01$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.
  - Ha a bemenet $10$-ra változik, akkor a hálózat $C$ állapotba kell, hogy kerüljön. Az $10$ oszlopban stabil állapot még a $B$ is, ezért ez egy leselkedő hazárdállapot. Ezt az átmenetet és a hozzá tartozó leselkedőt ($D \to C$, $B$) feljegyezzük a táblázat $00 \to 10$ oszlopába.

| $00 \to 01$ | $00 \to 10$ | | $01 \to 00$ | | $01 \to 11$ | | $10 \to 00$ | | $10 \to 11$ | | $11 \to 01$ | $11 \to 10$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $A \to C$ | $B$ | $B \to D$ | $A$ | $B \to A$ | $E$ | $B \to D$ | $A$ | $B \to A$ | $E$ | | $A \to C$ | $B$ |
| | | | | $E$ | | | | $E$ | | | | | |
| | $D \to C$ | $B$ | | | | | $C \to A$ | $D$ | $C \to A$ | $E$ | | | |
| | | | | | | | | $E$ | | | | | |

- Stabil állapot az $E$ állapot $00$ bemenet esetén.
  - A bemenet megváltozhat $01$-re és ekkor a hálózat $B$ állapotba kell, hogy átváltson. Mivel a $01$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.
  - Ha a bemenet $10$-ra változik, akkor a hálózat $C$ állapotba kell, hogy kerüljön. Az $10$ oszlopban stabil állapot még a $B$ is, ezért ez egy leselkedő hazárdállapot. Ezt az átmenetet és a hozzá tartozó leselkedőt ($E \to C$, $B$) feljegyezzük a táblázat $00 \to 10$ oszlopába.

| $00 \to 01$ | $00 \to 10$ | | $01 \to 00$ | | $01 \to 11$ | | $10 \to 00$ | | $10 \to 11$ | | $11 \to 01$ | $11 \to 10$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $A \to C$ | $B$ | $B \to D$ | $A$ | $B \to A$ | $E$ | $B \to D$ | $A$ | $B \to A$ | $E$ | | $A \to C$ | $B$ |
| | | | | $E$ | | | | $E$ | | | | | |
| | $D \to C$ | $B$ | | | | | $C \to A$ | $D$ | $C \to A$ | $E$ | | | |
| | | | | | | | | $E$ | | | | | |
| | $E \to C$ | $B$ | | | | | | | | | | | |

- Stabil állapot az $E$ állapot $11$ bemenet esetén.
  - A bemenet megváltozhat $01$-re és ekkor a hálózat $B$ állapotba kell, hogy átváltson. Mivel a $01$ oszlopban nincs más (stabil) állapot, ezért ebben az esetben nem fenyeget a veszély, hogy más állapotba kerülhetünk.
  - Ha a bemenet $10$-ra változik, akkor a hálózat $C$ állapotba kell, hogy kerüljön. Az $10$ oszlopban stabil állapot még a $B$ is, ezért ez egy leselkedő hazárdállapot. Ezt az átmenetet és a hozzá tartozó leselkedőt ($E \to C$, $B$) feljegyezzük a táblázat $11 \to 10$ oszlopába.

| $00 \to 01$ | $00 \to 10$ | | $01 \to 00$ | | $01 \to 11$ | | $10 \to 00$ | | $10 \to 11$ | | $11 \to 01$ | $11 \to 10$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $A \to C$ | $B$ | $B \to D$ | $A$ | $B \to A$ | $E$ | $B \to D$ | $A$ | $B \to A$ | $E$ | | $A \to C$ | $B$ |
| | | | | $E$ | | | | $E$ | | | | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| $D \to C$ | $B$ | | | | | $C \to A$ | $D$ | $C \to A$ | $E$ | | | | | $E \to C$ | $B$ |
| | | | | | | | $E$ | | | | | | | | |
| $E \to C$ | $B$ | | | | | | | | | | | | | | |

Az elkészült táblázatot vizsgáljuk meg a redundancia miatt, azaz keressünk olyan feltételeket, amelyek többször is szerepelnek akár fordított átmeneti iránnyal.

- Az $A \to C$ átmenet $B$ leselkedővel kétszer is szerepel a táblázatban, egyszer az $00 \to 10$, egyszer az $11 \to 10$ oszlopban, ezért az egyiket törölhetjük. Most töröljük például az $11 \to 10$ oszlopból.

| $00 \to 01$ | | $00 \to 10$ | | $01 \to 00$ | | $01 \to 11$ | | $10 \to 00$ | | $10 \to 11$ | | $11 \to 01$ | $11 \to 10$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $A \to C$ | $B$ | $B \to D$ | $A$ | $B \to A$ | $E$ | $B \to D$ | $A$ | $B \to A$ | $E$ | | | $E \to C$ | $B$ |
| | | | | | $E$ | | | | $E$ | | | | | | |
| | | $D \to C$ | $B$ | | | | | $C \to A$ | $D$ | $C \to A$ | $E$ | | | | |
| | | | | | | | | | $E$ | | | | | | |
| | | $E \to C$ | $B$ | | | | | | | | | | | | |

- Az $E \to C$ átmenet $B$ leselkedővel is kétszer szerepel a táblázatban, ezért az egyiket törölhetjük.

| $00 \to 01$ | | $00 \to 10$ | | $01 \to 00$ | | $01 \to 11$ | | $10 \to 00$ | | $10 \to 11$ | | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $A \to C$ | $B$ | $B \to D$ | $A$ | $B \to A$ | $E$ | $B \to D$ | $A$ | $B \to A$ | $E$ | | |
| | | | | | $E$ | | | | $E$ | | | | |
| | | $D \to C$ | $B$ | | | | | $C \to A$ | $D$ | $C \to A$ | $E$ | | |
| | | | | | | | | | $E$ | | | | |
| | | $E \to C$ | $B$ | | | | | | | | | | |

- A $B \to D$ átmenet $A$ és $E$ leselkedőkkel kétszer szerepel a táblázatban, ezért az egyiket törölhetjük.

| $00 \to 01$ | | $00 \to 10$ | | $01 \to 00$ | | $01 \to 11$ | | $10 \to 00$ | | $10 \to 11$ | | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $A \to C$ | $B$ | $B \to D$ | $A$ | $B \to A$ | $E$ | $C \to A$ | $D$ | $B \to A$ | $E$ | | |
| | | | | | $E$ | | | | $E$ | | | | |
| | | $D \to C$ | $B$ | | | | | | | $C \to A$ | $E$ | | |
| | | $E \to C$ | $B$ | | | | | | | | | | |

- A $B \to A$ átmenet $E$ leselkedővel kétszer szerepel a táblázatban, ezért az egyiket törölhetjük.

| $00 \to 01$ | | $00 \to 10$ | | $01 \to 00$ | | $01 \to 11$ | | $10 \to 00$ | | $10 \to 11$ | | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $A \to C$ | $B$ | $B \to D$ | $A$ | $B \to A$ | $E$ | $C \to A$ | $D$ | $C \to A$ | $E$ | | |
| | | | | | $E$ | | | | $E$ | | | | |
| | | $D \to C$ | $B$ | | | | | | | | | | |
| | | $E \to C$ | $B$ | | | | | | | | | | |

Ha tovább vizsgáljuk az átmeneteket, akkor azt is láthatjuk, hogy a $C \to A$ átmenet szerepel $D$ és $E$ leselkedőkkel a $10 \to 00$ oszlopban valamint szerepel $E$ leselkedővel a $10 \to 11$ oszlopban. Egyszerűen belátható, hogy az első feltétel szigorúbb és tartalmazza a második feltételt, ezért a másodikat törölhetjük a táblázatból.

| $00 \to 01$ | $00 \to 10$ | $01 \to 00$ | $01 \to 11$ | $10 \to 00$ | $10 \to 11$ | $11 \to 01$ | $11 \to 10$ |
|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A \to C$ | $B$ | $B \to D$ | $A$ | $B \to A$ | $E$ | $C \to A$ | $D$ |
| | | | $E$ | | | | $E$ |
| $D \to C$ | $B$ | | | | | | |
| $E \to C$ | $B$ | | | | | | |

Generáljunk kódot a példánkhoz. Mivel hat szabályunk van, ezért első lépésben feltételezzük, hogy hat állapotváltozónk lesz, azaz az alábbi táblázatot kel kitölteni.

| | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ |
|---|---|---|---|---|---|---|
| $A$ | | | | | | |
| $B$ | | | | | | |
| $C$ | | | | | | |
| $D$ | | | | | | |
| $E$ | | | | | | |

Egyesével vizsgáljuk meg a szabályokat:

- $A \to C$ átmenet $B$ leselkedővel
  - A szabály szerint az $A$ és a $C$ állapotok kódja egy változóban megegyezik egymással és különbözik $B$ kódjától. Ez a változó legyen most az $Y_1$ és mondjuk azt, hogy $A$ és $C$ kódja ebben a változóban $0$, $B$ kódja pedig $1$. A szabály nem mond semmit $D$ és $E$ változókról, azaz oda közömbös bejegyzéseket tehetünk.

| | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ |
|---|---|---|---|---|---|---|
| $A$ | $0$ | | | | | |
| $B$ | $1$ | | | | | |
| $C$ | $0$ | | | | | |
| $D$ | $-$ | | | | | |
| $E$ | $-$ | | | | | |

- $D \to C$ átmenet $B$ leselkedővel
  - A szabály szerint a $D$ és a $C$ állapotok kódja egy változóban megegyezik egymással és különbözik $B$ kódjától. Ez a változó legyen most az $Y_2$ és mondjuk azt, hogy $D$ és $C$ kódja ebben a változóban $0$, $B$ kódja pedig $1$. A szabály nem mond semmit $A$ és $E$ változókról, azaz oda közömbös bejegyzéseket tehetünk.

| | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ |
|---|---|---|---|---|---|---|
| $A$ | $0$ | $-$ | | | | |
| $B$ | $1$ | $1$ | | | | |
| $C$ | $0$ | $0$ | | | | |
| $D$ | $-$ | $0$ | | | | |
| $E$ | $-$ | $-$ | | | | |

- $E \to C$ átmenet $B$ leselkedővel
  - A szabály szerint az $E$ és a $C$ állapotok kódja egy változóban megegyezik egymással és különbözik $B$ kódjától. Ez a változó legyen most az $Y_3$ és mondjuk azt, hogy $E$ és $C$ kódja ebben a változóban $1$, $B$ kódja pedig $0$. A szabály nem mond semmit $A$ és $D$ változókról, azaz oda közömbös bejegyzéseket tehetünk.

| | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ |
|---|---|---|---|---|---|---|
| $A$ | $0$ | $-$ | $-$ | | | |
| $B$ | $1$ | $1$ | $0$ | | | |
| $C$ | $0$ | $0$ | $1$ | | | |
| $D$ | $-$ | $0$ | $-$ | | | |
| $E$ | $-$ | $-$ | $1$ | | | |

- $B \rightarrow D$ átmenet $A$ és $E$ leselkedővel
  - A szabály szerint a $B$ és a $D$ állapotok kódja egy változóban megegyezik egymással és különbözik $A$ és $E$ kódjától. Ez a változó legyen most az $Y_4$ és mondjuk azt, hogy $B$ és $D$ kódja ebben a változóban $0$, $B$ kódja pedig $1$. A szabály nem mond semmit $C$ változóról, azaz oda közömbös bejegyzést tehetünk.

| | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ |
|---|---|---|---|---|---|---|
| $A$ | $0$ | $-$ | $-$ | $1$ | | |
| $B$ | $1$ | $1$ | $0$ | $0$ | | |
| $C$ | $0$ | $0$ | $1$ | $-$ | | |
| $D$ | $-$ | $0$ | $-$ | $0$ | | |
| $E$ | $-$ | $-$ | $1$ | $1$ | | |

- $B \rightarrow A$ átmenet $E$ leselkedővel
  - A szabály szerint a $B$ és az $A$ állapotok kódja egy változóban megegyezik egymással és különbözik $E$ kódjától. Ez a változó legyen most az $Y_5$ és mondjuk azt, hogy $B$ és $A$ kódja ebben a változóban $0$, $E$ kódja pedig $1$. A szabály nem mond semmit $C$ és $D$ változókról, azaz oda közömbös bejegyzéseket tehetünk.

| | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ |
|---|---|---|---|---|---|---|
| $A$ | $0$ | $-$ | $-$ | $1$ | $0$ | |
| $B$ | $1$ | $1$ | $0$ | $0$ | $0$ | |
| $C$ | $0$ | $0$ | $1$ | $-$ | $-$ | |
| $D$ | $-$ | $0$ | $-$ | $0$ | $-$ | |
| $E$ | $-$ | $-$ | $1$ | $1$ | $1$ | |

- $C \rightarrow A$ átmenet $D$ és $E$ leselkedővel
  - A szabály szerint a $C$ és az $A$ állapotok kódja egy változóban megegyezik egymással és különbözik $D$ és $E$ kódjától. Ez a változó legyen most az $Y_6$ és mondjuk azt, hogy $C$ és $A$ kódja ebben a változóban $1$, $D$ és $E$ kódja pedig $0$. A szabály nem mond semmit $B$ változóról, azaz oda közömbös bejegyzést tehetünk.

| | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ |
|---|---|---|---|---|---|---|
| $A$ | $0$ | $-$ | $-$ | $1$ | $0$ | $1$ |
| $B$ | $1$ | $1$ | $0$ | $0$ | $0$ | $-$ |
| $C$ | $0$ | $0$ | $1$ | $-$ | $-$ | $1$ |
| $D$ | $-$ | $0$ | $-$ | $0$ | $-$ | $0$ |
| $E$ | $-$ | $-$ | $1$ | $1$ | $1$ | $0$ |

A táblázatban könnyen észrevehető, hogy $Y_1$ és $Y_2$ változók összevonhatóak, mivel az $A$ változónál $Y_2$ nem definiált, $B$ változónál mind a kettőnek $1$ az értéke, $C$ változónál mind a két változó értéke $0$, $D$ változónál $Y_1$ közömbös, $E$ változónál pedig mind a kettő állapotváltozónál közömbös bejegyzés szerepel. Az összevont változót jelöljük $Y_{12}$-vel, melynek értéke $A$ állapotnál $0$, mivel $Y_1 = 0$, $B$ állapotnál $1$, mivel $Y_1 = Y_2 = 1$, $C$ állapotnál $0$, mivel $Y_1 = Y_2 = 0$, $D$ állapotnál $0$, mivel $Y_2 = 0$, valamint $E$ állapotnál $-$, mivel $Y_1 = Y_2 = -$. Hasonló módon látható, hogy $Y_3$ és $Y_4$ változók is összevonhatóak.

| | $Y_{12}$ | $Y_{34}$ | $Y_5$ | $Y_6$ |
|---|---|---|---|---|
| $A$ | $0$ | $1$ | $0$ | $1$ |
| $B$ | $1$ | $0$ | $0$ | $-$ |
| $C$ | $0$ | $1$ | $-$ | $1$ |
| $D$ | $0$ | $0$ | $-$ | $0$ |
| $E$ | $-$ | $1$ | $1$ | $0$ |

A példában $Y_5$ és $Y_6$ is összevonható, ha az egyik változóhoz tartozó oszlopot invertáljuk. Emlékeztetőül két állapot akkor vonható össze invertálás segítségével, ha minden olyan állapotnál értékük különbözik, ahol mind a kettő nem közömbös bejegyzést tartalmaz. Jelen esetben $A$ állapotnál $Y_5 = 0$ és $Y_6 = 1$, valamint $E$ állapotnál $Y_5 = 1$ és $Y_6 = 0$. Minden más állapotnál valamelyik változóhoz közömbös bejegyzés tartozik. A kritikus versenyhelyzet szempontjából tetszőleges, hogy melyik oszlopot invertáljuk, mi most válasszuk $Y_6$-ot.

| | $Y_{12}$ | $Y_{34}$ | $Y_5$ | $Y_6$ |
|---|---|---|---|---|
| $A$ | $0$ | $1$ | $0$ | $0$ |
| $B$ | $1$ | $0$ | $0$ | $-$ |
| $C$ | $0$ | $1$ | $-$ | $0$ |
| $D$ | $0$ | $0$ | $-$ | $1$ |
| $E$ | $-$ | $1$ | $1$ | $1$ |

Az invertálás után már összevonhatóak $Y_5$ és $Y_6$ a már megszokott módszerrel $Y_{56}$ változóvá.

| | $Y_{12}$ | $Y_{34}$ | $Y_{56}$ |
|---|---|---|---|
| $A$ | $0$ | $1$ | $0$ |
| $B$ | $1$ | $0$ | $0$ |
| $C$ | $0$ | $1$ | $0$ |
| $D$ | $0$ | $0$ | $1$ |
| $E$ | $-$ | $1$ | $1$ |

További összevonást nem lehet végrehajtani a táblázatban, mert bármely változópárt vizsgáljuk, mindig találunk olyan állapotokat, amelyben különböznek és olyat is, amelyben megegyeznek. Például $Y_{12}$ különbözik $Y_{34}$-től $A$, $B$ és $C$ állapotoknál, de megegyezik $D$ állapotnál. Hasonlóan igazolható, hogy $Y_{12}, Y_{56}$ valamint $Y_{34}, Y_{56}$ állapotpárokra nem összevonhatóak.

A táblázatban szerepel egy közömbös bejegyzés, amit tetszőlege módon be lehet állítani $0$-ra vagy $1$-re, mi most válasszuk az $1$-et.

| | $Y_{12}$ | $Y_{34}$ | $Y_{56}$ |
|---|---|---|---|
| $A$ | $0$ | $1$ | $0$ |
| $B$ | $1$ | $0$ | $0$ |

| | | | |
|---|---|---|---|
| $C$ | 0 | 1 | 0 |
| $D$ | 0 | 0 | 1 |
| $E$ | 1 | 1 | 1 |

Ha egyesével megnézzük a szabályainkat, akkor láthatjuk, hogy mindegyik teljesül. Példaként nézzük meg az első szabályt, ami $A \to C$ átmenet volt $B$ leselkedővel. Hogy ne kerülhessünk $B$ állapotba $A \to C$ átmenet közben, arról az $Y_{12}$ állapotváltozó gondoskodik, mivel az átmenet közben ennek a változónak nem kell megváltoznia (mind a két állapotnál $0$ az értéke), így nem kerülhetünk $B$ állapotba, mert annak értéke ennél a változónál $1$.

Észrevehetjük, hogy megoldásunkban $A$ és $C$ kódja megegyezik. A kódolás során ugyanis csak azt figyeltük, hogy ne alakulhasson ki kritikus versenyhelyzet és a szabályokban $A$ és $C$ sehol nem szerepelt a szabály két különböző oldalán. Ez azt jelenti, hogy kritikus versenyhelyzet szempontjából $A$ és $C$ soha nem leselkedője egymásnak, így kialakulhatott egyforma kód a két állapotra. A kódolás szabályai szerint azonban az állapotokat meg kell egymástól különböztetni, amit egy új változó bevezetésével lehetne megoldani.

# 10. Kódolási eljárások

A kódolás kifejezés hallatán a legtöbb embernek valamilyen titkosítással, rejtjelezéssel kapcsolatos dolog jut elsőre eszébe. Nem is tévednek nagyot, hiszen a kódolási eljárások nagy részét manapság adatvédelmi és titkosítási célokra használják. A kódolási eljárások felhasználásának másik fontos területe az adatbiztonság, amely azt jelenti, hogy a különböző átviteli csatornákon továbbított adatok sérülése esetén könnyen és gyorsan felismerhető legyen a meghibásodott információ. A hibás információ újraküldésével, esetleg a vételi oldalon történő kijavításával a hiba korrigálható.

E szerint a kódolás célja kettős:

- Adatvédelmi kódolás, a továbbított adatok hibáinak felderítésére, javítására
- Adatok titkosítása, illetéktelen hozzáférés elleni védelme

## *Adatvédelem, adatbiztonság*

Ha egy csatornán (adatvezetéken, optikai szálon, IR – inrfa-red adatkapcsolaton, RF – rádiófrekvenciás összeköttetésen) adatokat továbbítunk azok meghibásodhatnak. A hiba oka lehet külső zavar, az átviteli csatorna sérülése, meghibásodása vagy más, előre nem látható probléma. Ha megsérült adat érkezik a vevő egységbe és az azt érvényes adatként értelmezi és kezeli az, az adatgyűjtő, irányító stb. rendszer hibás működéséhez vezet. A hibás működés következtében rossz adatok, információk, sérült file-ok tárolódhatnak el, vagy egy irányítórendszer hibás adatok alapján végzi a feladatát. Mindkettő akár súlyos működésbeli és/vagy anyagi problémákat okozhat, elkerülésük tehát fontos feladat.

A részletesebb magyarázatok előtt nézzünk néhány elméleti alapfogalmat:

- **Kód**: két szimbólumhalmaz egyértelmű egymáshoz rendelésének rendszere.
- **Kódolás**: a szimbólumok / szimbólumhalmazok egymáshoz rendelése meghatározott szempontok szerint (két szimbólumhalmaz egyértelmű egymáshoz rendelésének rendszere)
- **Dekódolás**: a kódolás ellentétes művelete – visszatérés az eredeti halmazra
- **Jelkészlet**: azon jelek összessége, amelyeket meghatározott szabályok szerint a kódszavak felépítéséhez használhatnak.
- **Kódszó**: a jelkészlet elemeiből meghatározott szabályok szerint felépített értelmes üzenetet jelentő egybefüggő jelsorozat.
- **Kódszó készlet**: a kódolásra meghatározott szabályok szerint felhasználható egybefüggő kódszavak összessége.
- **Tiltott kódszó**: olyan kódszavak, amelyek képzési szabályok szerint képezhetőek, de nem alkalmazhatók az adott kódkészletben.
- **Bit**: az információ legkisebb egysége.
- **Redundancia** az információelméletben az információ vagy üzenet átvitelnél használt bitek számának és az aktuális információ vagy üzenet bitjei számának a különbsége.
  Információelméleti értelemben tehát a redundancia több jel felhasználása, mint amennyire adott információ közvetítéséhez feltétlenül szükség van. Az üzenet közvetítéséhez minimálisan szükséges és az aktuálisan felhasznált jelek mennyisége alapján a redundancia mértéke számszerűen is kifejezhető.
- Az **adattömörítés** egy lehetséges mód a nem kívánt redundancia csökkentésére, a különféle ellenőrzőösszegek pedig hibajavítás céljából növelik a redundanciát, ha az átvitel egy zajos csatornán folyik, ahol a zaj csökkenti az átviteli kapacitást.
- **Hamming-távolság** alatt két azonos hosszúságú kódszó eltérő bitjeinek, illetve karaktereinek a számát értjük. Tehát azt, hogy egy kódszóban hány bitet kell az ellenkezőjére változtatni, hogy egy másik érvényes kódszót kapjunk.

- Egy **kódrendszer Hamming-távolsága** alatt a kódszó készlet elemei között észlelt legkisebb Hamming távolságot értjük. Ezt rendszerint d-vel jelöljük. A felfedezhető hibák száma d-1, a javítható hibák száma pedig d/2-nél kisebb egész szám.

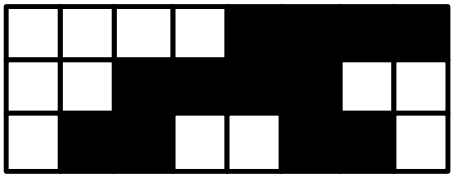
## *Kódtípusok:*

### Pozíció kódok:

Elfordulás, elmozdulás tehát pozíció változás detektálására használják őket. Az egymásután következő pozíciók kódja egy Hamming távolságú. Így a pozíció érzékelők (pl. optikai érzékelők) a pozíció határ átmenetnél nem adnak hibásan "távoli" pozíciót jelentő kódot, ahogy az több Hamming távolságú kód estén előfordulhatna.

### *Gray-kód*

Legismertebb pozíciókód a **Gray-kód**:



A fenti ábra egy 8 szakaszra osztott vízszintes mozgás detektálására alkalmas 3 bites Gray-kódot mutat.

A kombinációs hálózatok grafikus egyszerűsítésénél használt Karnaugh táblák peremezése is Gray kódú, mivel a feladat ott is az egymás mellette elhelyezkedő sorok-oszlopok szomszédos kódolása.

Nagyobb bitszámú pozíciókódot tükrözéses módszerrel lehet kisebb bitszámú pozíció kódból készíteni. Például a 2 bites Gray kódból (most a kisebb helyfoglalás miatt az egymás alatti számok adják a kódot, 00,01,11,10):

0011

0110

Folytassuk a kódok felírását fordított sorrendben, hogy az előző felírás tükörképét kapjuk:

0011 | 1100

0110 | 0110

Végül a régi kódok elé (fölé) írjunk 0-át, a tükrözött kódok elé pedig 1-et:

0000 1111

0011 1100

0110 0110

Így megkaptuk az előbbi ábrának megfelelő Gray kódot.

### *Johnson kód*

Ez szintén egy pozíció kód, vagyis az egymást követő kódok Haming távolsága =1.

- 3 bites Johnson kód: 000, 001, 011, 111, 110, 100
- 4 bites Johnson kód: 0000, 0001, 0011, 0111, 1111, 1110, 1100, 1000

Képzése: a csupa 0-t tartalmazó $N$ bites kódszóból kiindulva jobbról először 1-esekkel, majd 0-ákkal töltjük fel a bitpozíciókat. Ezért az $N$ bites Johnson kód kódszavainak száma $2N$.

***Súlyozott kódok***

A súlyozott kód a kódszavak helyi értékéhez valamilyen valós számot (súly) rendel. Ilyenkor a kódszó információtartalmát rendszerint a kódszó egyeseinek megfelelő súlyok összege adja.

$$N_\omega = \sum_{i=1}^{k} W_i S_i$$

Ahol $W_i$ az $i$-edik pozíció súlyozása, $S_i$ – az $i$-edik pozíció bináris értéke (0, 1) és $k$ a kódszó elemeinek a száma.

## *BCD kódok*

A száminformációk továbbítására leggyakrabban a binárisan kódolt decimális kódokat (BCD) használják. Tíz számjegy kettes számrendszerbeli ábrázolásához minimálisan 4 helyi értékre van szükség. Viszont a 4 változó lehetséges 16 kombinációjából csak 10-et használunk ki.

- A decimális számok feldolgozásánál leginkább a normál BCD kódot (NBCD) alkalmazzák → súlyozása: 8 4 2 1
- AIKEN feltöltése: mindig a kisebb helyi értékűeket tölti fel először, majd a kettesek közül mindig az elsőt → súlyozása: 2 4 2 1
- STIBITZ (háromtöbbletes vagy élőnullás kód), minden értékhez 3-at kell adni
- WHITE itt is a kisebb helyi értékűt tölti fel először; súlyozása: 5 2 1 1

A következő táblázat a leggyakrabban használt BCD kódok grafikus szemléltetését mutatja:

| ELNEVEZÉS | | NBCD | | | | AIKEN | | | | STIBITZ | | | | 2421 I. | | | | 2421 II. | | | | 4221 | | | | 5421 | | | | 5221 | | | | 5311 | | | | WHITE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SÚLY | | 8 | 4 | 2 | 1 | 2 | 4 | 2 | 1 | NINCS | | | | 2 | 4 | 2 | 1 | 2 | 4 | 2 | 1 | 4 | 2 | 2 | 1 | 5 | 4 | 2 | 1 | 5 | 2 | 2 | 1 | 5 | 3 | 1 | 1 | 5 | 2 | 1 | 1 |
| DECIMÁLIS ÉRTÉK | 0 | | | | | | | | | | | ● | ● | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 1 | | | | ● | | | | ● | | ● | | | | | | ● | | | | ● | | | | ● | | | | ● | | | | ● | | | | ● | | | | ● |
| | 2 | | | ● | | | | ● | | | ● | | ● | ● | | | | | | ● | | | | ● | | | | ● | | | | ● | | | ● | | | ● | | | ● |
| | 3 | | | ● | ● | | | ● | ● | | ● | ● | | | | | ● | | | ● | ● | | | ● | ● | | | ● | ● | | | ● | | ● | | | | ● | | | ● |
| | 4 | | ● | | | | ● | | | | ● | ● | ● | ● | | | | | ● | | | | ● | ● | | | ● | | | | ● | ● | | | ● | | | | ● | | ● |
| | 5 | | ● | | ● | ● | | ● | ● | ● | | | | ● | | ● | ● | | ● | | ● | | ● | | ● | ● | | | | ● | | | | ● | | | | ● | | | |
| | 6 | | ● | ● | | ● | ● | | | ● | | | ● | ● | ● | | | | ● | ● | | ● | ● | | | ● | | | ● | ● | | | | ● | | | | ● | | | ● |
| | 7 | | ● | ● | ● | ● | ● | | ● | ● | | ● | | ● | ● | | ● | | ● | ● | ● | ● | ● | | ● | ● | | ● | | ● | | ● | | ● | | ● | | ● | | ● | ● |
| | 8 | ● | | | | ● | ● | ● | | ● | | ● | ● | ● | ● | ● | | ● | ● | ● | | ● | ● | ● | | ● | | ● | ● | ● | | ● | ● | ● | ● | | | ● | ● | | ● |
| | 9 | ● | | | ● | ● | ● | ● | ● | ● | ● | | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | | | ● | ● | ● | | ● | ● | | ● | ● | ● | ● | ● |

**10.1 ábra BCD kódok grafikus ábrázolása**

**Excess kódok:**

Az excess kódok közvetetten súlyozott kódok. A szám képzése:

$$N_\omega = \frac{\sum_{i=1}^{k} W_i S_i - e}{q}$$

$e$ és $q$ : két különböző szám, a kódolástól függően kell megadni

- STIBITZ, ha $e = 3; q = 1$
- NIDDING, ha $e = 5; q = 2$
- DIAMOND, ha $e = 6; q = 27$

## *Hibafelfedő-, és javító kódok*

Az előzőekben ’zajmentes’ csatornákat feltételeztünk, de ’zajos’ csatorna esetén a küldött kódszó megváltozhat, azaz a kódszóban ’0’ → ’1’ vagy ’1’ → ’0’ csere jöhet létre. A gyakorlatban annak a legnagyobb a valószínűsége, hogy csak egy helyen változik meg a kódszó.

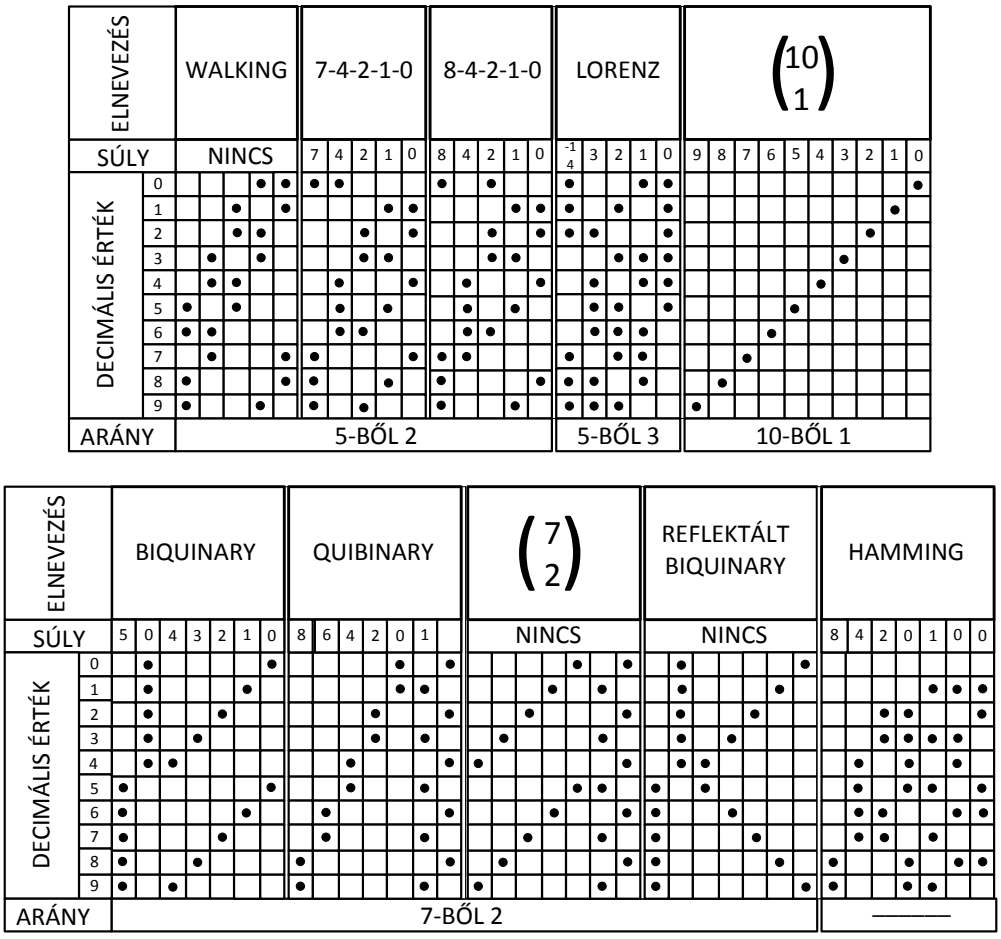
Egyetlen hiba úgynevezett paritásos ellenőrzéssel felfedhető: azaz minden kódszóval még egy ellenőrző helyi értéket (paritás bit) is átviszünk a csatornán. Ennek értékét úgy választjuk meg, hogy a teljes kódszó az ellenőrző helyi értékkel együtt páros vagy páratlan ’0’-kat ill. ’1’-eseket tartalmazzon. Ezt a járulékos helyi értéket paritás bitnek nevezik.

A TTL rendszerben a paritás bit előállítása, ill. ellenőrzése az SN 74180 típusú bővíthető 8-bites áramkörrel könnyen elvégezhető.

A paritásos ellenőrzést igen gyakran alkalmazzák, mert így tetszőleges kódtípus felruházható hibafelfedő képességgel. A paritás generátor figyeli az adó kimeneteit és előállítja a paritás bitet, amivel kiegészíti az átviendő jelet. A vevő oldalon a paritásvizsgáló figyeli a kiegészített információt, és ha nem megfelelő számú ’1’-est észlel, akkor jelzést ad.

Egyszerűbb hibafelfedő áramkört igényelnek az aránykódok. Ezek alapvető jellemzője, hogy az egyes kódszavakban szereplő ’1’-esek és’ 0’-k aránya állandó. A leggyakrabban használt aránykód a 7-4-2-1-0 súlyozású.

- WALKING-kód: 7-4-2-1-0-kód, 8-4-2-1-0-kód
- LORENZ-kód: 10-ből 1 –kód, 7-ből 1 –kód
- HAMMING-kód

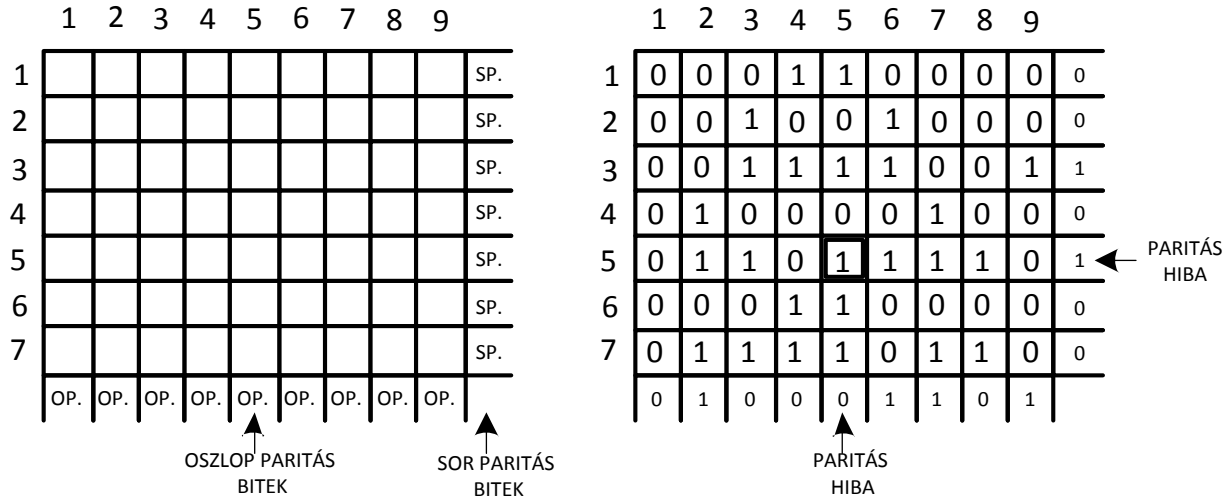



**10.2 ábra Aránykódok grafikus ábrázolása**

A hibafelfedés feltétele, hogy a Hamming távolság (d) 1-nél nagyobb legyen. Ha ugyanis két kódszó között a Hamming távolság csak 1, akkor egy hiba egy másik érvényes kódszót is eredményezhet. Maximálisan tehát d-1 hiba fedezhető fel. (Ez pl. egy redundancia lehet)

Az eddigiekben csak a hiba létének kimutatásával foglalkoztunk. A hiba javításához a hiba helyét is pontosan ismerni kell.

A hibajavítást blokkszerű adatátvitel esetén sor- és oszlop paritás ellenőrzésével is elvégezhetjük. Ha az adó kódszavait szisztematikusan egymás alá írva képzeljük el, akkor egy-egy paritás bitet minden sorhoz és minden oszlophoz rendelhetünk. Ily módon egyetlen hiba a hibás sor és oszlop metszéspontjában van. Így tudjuk a hiba helyét, tehát értékcserével (’0’ → ’1’ vagy ’1’ → ’0’) a hiba javítható.

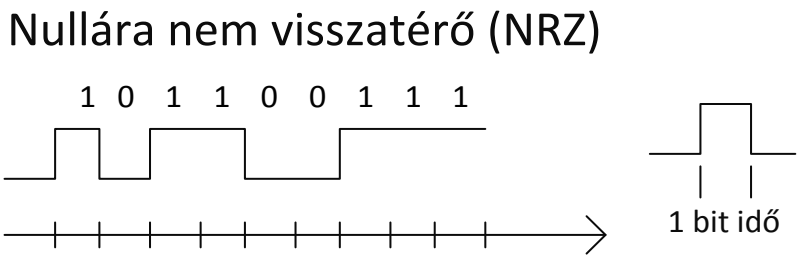



**10.3 ábra Sor-oszlop paritáskód**

A bináris kódok alkalmazásának hátránya, hogy ha egy jelsorozat ’0’ értékkel kezdődik, a vevő oldalon nem lehet meghatározni, hogy hány 0 volt az első előtt. Továbbá az egymás utáni több 0 vagy 1 érték kerül átvitelre, akkor nincs jelátmenet, amit a vevő a szinkronizációhoz felhasználhasson. Ennek a problémának a megoldására több lehetőség is kínálkozik.

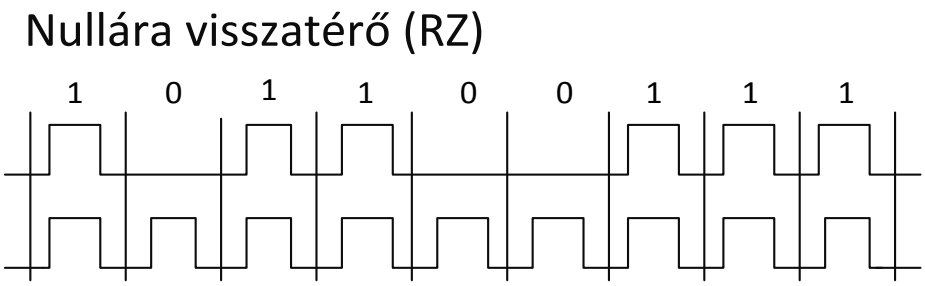
## *NRZ-RZ kódolás*

NRZ (Non Return to Zero) kódolás esetén a több egymás utáni 1-es nem különül el. Ez a leginkább gyakori, természetes jelforma. Ha egy bit 1-es, akkor a feszültség teljes bit idő alatt H-szintű, ha 0-s, akkor L szintű. Két vagy több egymás utáni 1-es bit esetén a feszültség megszakítás nélkül H-marad a megfelelő ideig, az egyesek között nem tér vissza 0-ra.




**10.4 ábra NRZ kód**

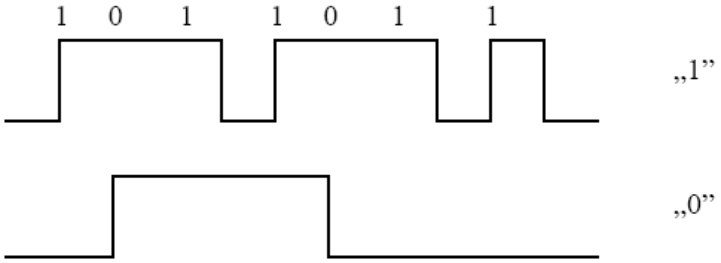
RZ (Return to Zero) kódolás jellegzetessége, hogy minden 1-es bit után visszatér 0-ra, de a ’0’ bitek esetén az NRZ-hez hasonlóan folyamatosan nulla a jelszint. Azért az egyes értékek esetén van csak visszatérés, mert nagyon sok jeltovábbító egység stand-by állapota magas logikai szintű, a jelek a vonalra invertálás után kerülnek. RZ jelkódolást használva el tudjuk választani a stand-by állapotot az egymás utáni folyamatos egyes bitektől.




**10.5 ábra RZ kód**

## *Fáziskódolt jelátmenet*

Fáziskódolt (PE – phase encoded) jelátmenet - az ugrás reprezentálja a biteket attól függően, hogy az átmenet milyen irányú 0 → 1 vagy 1 → 0. Két csatorna van: az egyik az ’1’-eseket, a másik a ’0’-kat reprezentálja



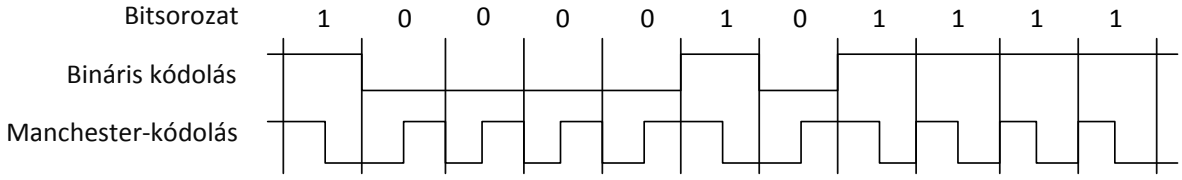

**10.6 ábra Fáziskódolt jelátmenet**

Ilyen kódolást találunk a mágnesszalagos jelrögzítő rendszerek esetén. Nagy előnye, hogy a két csatorna jelátmeneteinek ÉS kapcsolata az órajelet adja, tehát nincsen szükség külön szinkronizációt biztosító órajel csatornára. Megfigyeltük már, hogy mindegy milyen lassan vagy gyorsan húzzák el a mágneskártyánkat leolvasó egységben?

## *Manchester kódolás*

A kódolási eljárás lényege, hogy minden bit idejét két egyenlő részre osztják és a bitidőn belül egy jelátmenetet alakítanak ki a következők szerint:

- Bináris ’1’ küldése esetén az első fél bitidőben a jel magas szintű, a másodikban alacsony
- Bináris ’0’ küldése esetén az első fél bitidőben a jel alacsony szintű, a másodikban magas




**10.7 ábra Manchester kódolás**

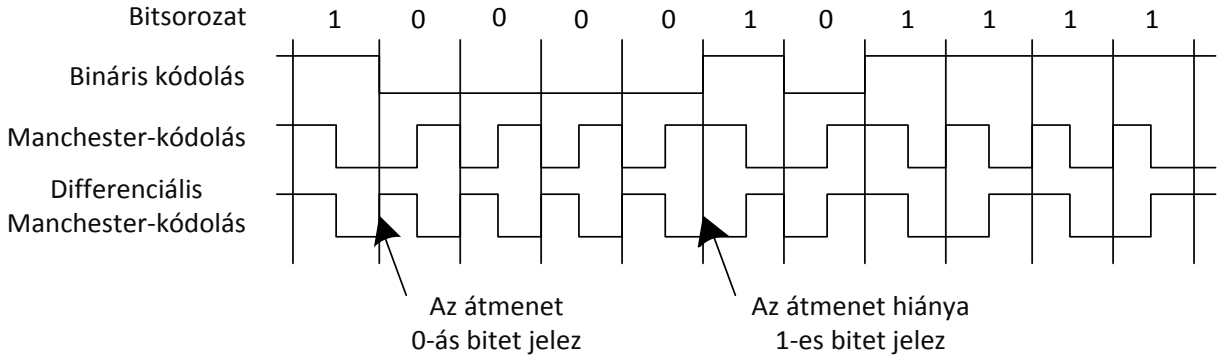
Így minden továbbított bit tartalmaz egy jelátmenetet, ami alkalmas a szinkronizálásra. A módszer hátránya, hogy az átvitel kétszer akkora sávszélességet igényel. Az alapsávi rendszerek közül nagyon sok használja a Manchester kódolást az egyszerűsége miatt.

## *Differenciális Manchester kódolás*

A differenciális Manchester kódolás esetén a ’0’ bitet a bitidő elején fellépő jelátmenet jelzi, míg az ’1’ bitet, ennek hiánya. Egymást követő ’0’ bitek esetén, a bitidőn belül egy le-, majd egy felfutó él

ismétlődik. Egymást követő ’1’-esek esetén viszont a bitek határán nincs jelátmenet, csak a bitidő közepén.

Mindkét Manchester kódolás esetén a bitidő közepén található jelátmenet, ami a bitek detektálását segíti. A differenciális Manchester kód előállítása és detektálása bonyolultabb berendezést igényel, viszont jobb zavarvédettséget biztosít. Az összes alapsávú 802.3 szabványú rendszer, Manchester-kódolást használ. A magas szintnek +0,85 volt, míg az alacsonynak -0,85 volt felel meg, így az egyenáramú (DC) komponens 0 volt lesz.




**10.8 ábra Differenciális Manchester-kódolás**

## *Adatok titkosítása*

A kódolási és titkosítási eljárások egészen a 1980-as évekig szinte kizárólag a katonai és nemzetvédelmi területeken kerültek alkalmazásra. Ennek alapvető oka, hogy a polgári alkalmazások terén addig csak kevés igény merült fel és ezeket is jelentősen korlátozta az akkori technológiai fejlettségi szint. A polgári alkalmazások elterjedésének lehetőségét az un. nyilvános kulcsú titkosítás elterjedése alapozta meg.

### Titkosítás (kriptológia)

A titkosítás vagy rejtjelezés a kriptográfiának azaz eljárása, amellyel az információt un. nyílt szöveget egy algoritmus, titkosító eljárás segítségével olyan szöveggé alakítjuk, ami olvashatatlan olyan személyek számára, aki nem rendelkezik az olvasáshoz szükséges speciális tudással. Ezt a „tudást”, információt nevezzük kulcsnak. A kulcson kívül természetesen szükség van még a kulcs alkalmazásának ismeretére is. Az eredmény a titkosított információ, a titkosított szöveg. Sok titkosító eljárás egy az egyben, vagy egyszerű átalakítással használható megfejtésre is, azaz a titkosított szöveget újra olvashatóvá alakítsa. A titkosításnak fontos katonai szerepe volt, de igazán fontos, és nyilvános kutatások a XX. sz. második felében indultak meg, mivel ekkorra már előtérbe került a titkosítás személyi használatára való igénye (egyre több adat kerül átvitelre nyilvános csatornákon – pl. banki, egészségügyi, személyi adatok)

A titkosítást régóta használja a katonaság és a kormányok, azért, hogy megkönnyítsék a titkos kommunikációt például államtitkok vagy fontos nemzetgazdasági információk továbbítására, tárolására. Napjainkban sokféle területen használják, polgári rendszerekben is. Ilyenek például a számítógép-hálózatok, Internet, mobiltelefonok, vezeték nélküli hálózatok, és a bankautomaták.

A kriptoanalízis tudománya a titok „feltörésére” tartalmaz eljárásokat. Ezzel a témakörrel azonban most nem foglalkozunk.

Néhány a titkosítással kapcsolatos alapvető fogalom:

- A **titkosság** – egy titkos adat lehet hozzáférhető, de titkossága miatt illetéktelenek számára értelmezhetetlen.
- A **védettség** – nem más, mint a hozzáférés korlátozása.

- A **hitelesség** – azt jelenti, hogy ténylegesen az-e az információ forrása, a küldő, akire számítok, aki ilyen módon meg van jelölve.

Vizsgáljunk meg néhány egyszerű titkosító eljárást.

## *Szimmetrikus kulcsú kódolások*

A titkosítás során a nyílt adatból (nem titkos) blokkokat készítünk: $x = (x_1, x_2, \dots, x_n)$ – ez a nyílt üzenet. A kódoló ebből állítja elő az $y = (y_1, y_2, \dots, y_n)$ titkosított üzenetet, egyértelmű leképezéssel:

$$y = E_k(x),$$

ahol $E_k$ $k = (k_1, k_2, \dots, k_n)$ paraméterű kódoló transzformáció. A $k$ vektor a titkosítás kulcsa. Inverz transzformáció, vagy dekódolás: $x = D_k(y)$.

Egy példa: Az angol ábécé betűit felírjuk, és betűnkénti helyettesítést alkalmazunk, ahol a felső sorbeli karaktert az alatta lévőre cseréljük. A helyettesítő tábla lehet olyan, hogy a sort egy értelmes szóval kezdjük, majd utána felsoroljuk az ábécé fennmaradó betűit. A kód ekkor ez a szó lesz, és ennek ismeretében végezhetjük el a dekódolást.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

BOCIADEFGHJKLMNPQRSTUVWXYZ

A kódolást végezhetjük permutációval is, ez esetben a nyílt szöveget permutáció hosszú blokkokra bontjuk, majd kódoljuk. A kód itt a permutáció.

Például a

MINDEN TITKOS UGYNOK HAZUDIK

Szöveget bontsuk 8 karakteres blokkokra és kódoljuk az alábbi permutáció alkalmazásával:

$$\Pi = \begin{pmatrix} 12345678 \\ 54376821 \end{pmatrix}$$

Az eredmény:

EDNTNIIMUSOYGNKTZAHDUIKOK

A szóközöket kihagytuk a most fennmaradó 25. karaktert önmagával kódoltuk. A dekódolás az inverz permutálással történik.

Julius Caesar nevéhez kapcsolódik a Caesar-kód néven ismert titkosítás, ahol a kódolt üzenet ugyanúgy betűkből áll, mint az elküldendő üzenet, a nyílt szöveg. A galliai hadjáratról küldött leveleiben, ha a császár titkos dolgokat akart közölni, akkor az ábécében egyszerűen eltolta a betűket három egységgel, vagyis az A helyett D, a B helyett E betű szerepelt, és így tovább. Nincs róla adat, hogy ezt a titkosítást Julius Caesar korában feltörték volna. Különben Augustus nem használta volna évtizedekkel később ugyanezt a módszert azzal a különbséggel, hogy három helyett egy egységnyi eltolást alkalmazott.

Eredetileg a latin ábécé 21 betűt tartalmazott: A, B, C, D, E, F, Z, H, I, K, L, M, N, O, P, Q, R, S, T, V, X. Mivel a Z és S betű között jelentéktelennek ítélték a különbséget, az i. e. 4. században a Z-t kihagyták az ábécéből. Később, az i. e. 1. században azonban újra felvették, de mivel az eredeti helyét új betű, a G foglalta el, a Z a sor végére került. Julius Caesar korában a latin ábécé tehát így nézett ki: A B C D E F G H I J K L M N O P Q R S T V Z X Y.

Kódolás sorát az ábécé betűit feleltessük meg számoknak az A=0, B=1, C=2 helyettesítéssel.

Legyen $x = (x_1, x_2, x_3, \dots, x_n)$ nyílt üzenet, és $y = (y_1, y_2, y_3, \dots y_n)$ rejtett üzenet. A nyílt üzenet rejtett üzenetbe konvertálása a

$$x_i = y_i + k_i (\mathrm{mod}\ 26)$$

moduló összeadással történik, ahol $k = (k_1, k_2, k_3, \dots, k_n)$ vektor a kulcs. $i = (1,2,3, \dots, n)$ A vektor elemei a $\{0,1,2, \dots, n\}$ halmazból veszik az értékeiket.

A dekódolás az $x_i = y_i - k_i (\mathrm{mod}\ 26)$ művelettel történik.

Vannak más mechanizmusok is a szövegek kódolására. Ezekkel az eljárásokkal kódolt adatokat csak nehezen vagy egyáltalán nem lehet feltörni.

Az eljárási szabály: az ábécé minden betűjét a harmadik baloldali szomszédjával cseréljük ki, miközben minden betű után szúrjuk be az ábécé egy-egy betűjét az ábécé vége felöl kezdve. Ez már egy sokkal bonyolultabb eljárás, de még sokkal nehezebb kódolásokat is ki lehet találni.

A számítógépeken alkalmazott kódolási eljárások során az egész művelet hasonlóképpen megy végbe. A különbség annyi, hogy a számítógép csak nullákat **("0") és egyeseket ("1") ismer**. Így szöveg helyén 0-kból és 1-kből álló számsorozat található. Például így néz ki egy számsorozat: 0101010010111001001

Hogy ezt a számsort titkosítsuk, egy számtani műveletet alkalmazunk, ami átvitel nélküli összeadás. A művelet legyen a következő: 0+0=0; 1+0=1; 0+1=1; 1+1=0. (A logikai műveletek közül ez a művelet az úgynevezett "kizáró vagy"). A betűk helyett itt a számokat kódoljuk. Van egy kódolandó számsorunk és egy matematikai eljárásunk, milyen számsorozattal végezzük el az összeadást?

Természetesen egy kulccsal, amely maga is egy számsor.

Például, legyen a kulcsként használt számsor a következő: 0011010000100010011.

Adjuk össze ezt a két számsort:

| | | |
|---|---|---|
| | 0101010010111001001 | üzenet |
| + | 0011010000100010011 | kulcs |
| = | 0110000011110111010 | kódolt üzenet |

Vegyük észre, hogy itt a kulcs ugyanolyan hosszú, mint az eredeti üzenet. Ezért sok variációs lehetőség adódik a kulcsok kiválasztására. Minél hosszabb egy kulcs, annál többfajta kulcs létezik, s annál nehezebb a titkos üzenetet megfejteni.

Ilyen kulcsfüggő eljárások esetén a kódolási szabályt akár mindenki ismerheti, csak a kulcsnak kell titokban maradnia.

Ha a kódolt üzenetet ismét az eredeti üzenetté szeretnénk változtatni, akkor vagy a kódolt üzenetből kell kivonnunk a kulcsot, vagy egyszerűen csak az összeadási műveletet kell még egyszer elvégezni.

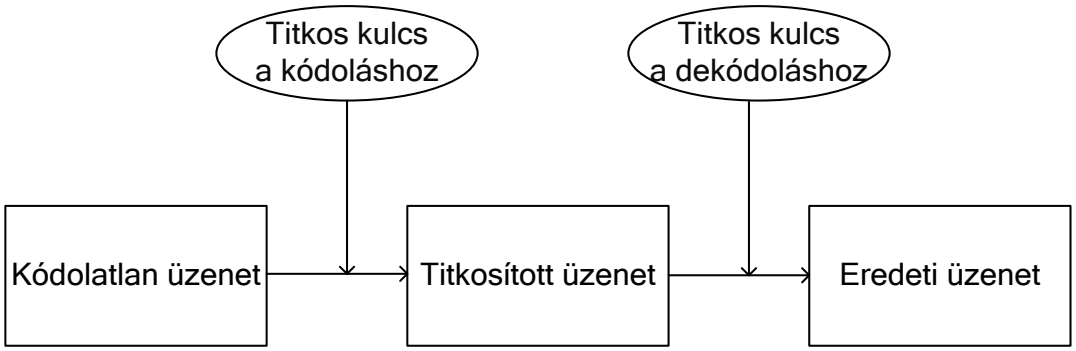
Az üzenet sikerének a „kulcsa”, hogy a küldőnek és a fogadónak is ugyanazzal a kulccsal kell rendelkeznie. De a kérdés az, hogy hogyan egyezzenek meg a kulcsban anélkül, hogy egy jogosulatlan harmadik személy kifigyelje, vagy ellophassa azt. Ez a szimmetrikus kódolási eljárások legnagyobb problematikája.

Egy másik probléma abból adódik, ha pl. sok személy kommunikál egymással. Tételezzük fel, hogy 12 személy szeretne egymással kódolt üzenetet váltani. Két embernek azonban lehetnek közös titkai, amit nem szeretnének a másik tíz emberrel megosztani. Hány kulcsot kell ilyen esetben alkalmazni?

Az első embernek 11 kulcsra van szüksége. Saját maga számára nem kell kulcs, de minden más személlyel történő kapcsolattartáshoz szükséges egy-egy kulcs. A másodiknak is 11 kulcsra van szüksége, de egyet az elsőnél már beszámoltunk, és így tovább. Ha így számolunk tovább, a következő eredmény születik:11+10+9+8+7+6+5+4+3+2+1=66. Ez nem kevés. 50 személy esetén ez a szám 1225, 1000 személy esetén majdnem 500000 különböző kulcs kellene. El lehet képzelni azt az esetet, ha például egy közösségi oldalon mindenki így szeretne üzenni a másiknak, miközben az üzenetek illetéktelenek számára nem hozzáférhetőek. A szimmetrikus kódolás e két problémáját az aszimmetrikus kódolás oldotta meg.

## *Konvencionális titkosítás*

Az eddig bemutatott konvencionális vagy szimmetrikus kulcsú kódolás hátránya, hogy mind a küldő, mind a fogadó oldalon rendelkezni kell a rejtjelezéshez használt kulccsal. Ilyenkor a legnagyobb problémát a kulcs biztonságos eljuttatása jelenti adóból vevőbe. Ez történhet rejtett úton, vagy nyilvános csatornán titkosítva. Utóbbi esetben visszakanyarodunk az előbbi problémához. A szimmetrikus kulcsú titkosító rendszer blokkvázlata az alábbi ábrán található:




**10.9 ábra Szimmetrikus kulcsok alkalmazása**

A szimmetrikus kulcsú titkosítás hátrányát részben megoldotta A. Shamir az általa kidolgozott érdekes eljárással. A módszer segítségével a kulcs cseréje nélkül történhet titkosított üzenetváltás partnerek között. Az esetleges behatolóról feltételezzük, hogy csak a csatornán folyó üzenet lehallgatására képes, azaz passzív támadó. Ez a módszer előrelépést jelent még a nyilvános kulcsú algoritmusokhoz képest is, mert itt a kulcs nyilvános részét sem kell a partnerek tudomására hozni.

Az algoritmus megértéséhez egy egyszerű példa legyen a következő:

**A** titkos üzenetet kíván küldeni **B**-nek. **A** az üzenetét egy lezárható (lelakatolható) dobozba teszi és elküldi **B**-nek. **B** nem tudja kinyitni a zárat, hiszen nincsen hozzá kulcsa. Ehelyett **B** is lezárja (lakatolja) a dobozt a saját zárjával és visszaküldi **A**-nak. **A** megkapja a kétszeresen is lezárt ládát. Ezt követően leveszi róla saját lakatját és visszaküldi **B**-nek. Most már csak **B** lakatja van a dobozon, amit **B** természetesen a saját kulcsával kinyit. **A** láda nyitva és **B** elolvashatja az üzenetet.

Amennyiben a lakatok helyett elektronikus kódolásra kívánunk áttérni, néhány megkötéssel kell élnünk. A felhasználók mindegyikének olyan titkosító algoritmussal kell rendelkeznie, hogy a rendszerben alkalmazott kódoló algoritmus kommutatív legyen, azaz a többszöri kódolás eredménye független legyen a kódolás sorrendjétől.

Tehát $E_k(x)$ kódoló transzformáció és $k_A$, $k_B$ kulcspár esetén

$$E_{k_A}\left(E_{k_B}(x)\right) = E_{k_B}\left(E_{k_A}(x)\right)$$

**A** tehát 3 lépésben tudja elküldeni üzenetét **B**-hez:

1. A => B: $y_1 = E_{k_A}(x)$
2. B => A: $y_2 = E_{k_B}\left(E_{k_A}(x)\right)$
3. A => B: $y_3 = D_{k_A}\left(E_{k_B}\left(E_{k_A}(x)\right)\right) = E_{k_B}$

A megoldás tökéletesnek tűnhet, hiszen a „láda” felnyitásához szükséges kulcsok sohasem kerülnek ki **A** és **B** „kezéből”, tehát a ládát senki illetéktelen nem tudja felnyitni.

Könnyen belátható azonban, hogy a nyilvános csatornán továbbított üzenet egyszerű lehallgatásával (passzív támadás) is könnyedén feltörhető.

A három kódolt üzenetet a következő

$$y_1 = x + k_A, \qquad y_2 = x + k_A + k_B, \qquad y_3 = x + k_B,$$

alakba írva láthatjuk, hogy ha a titkosítást moduló összeadással végeztük, akkor egyszerűen a három üzenet összegét képezve kapjuk, hogy

$$y_1 + y_2 + y_3 = x.$$

Tehát megfejtettük a nyit üzenetet. Egyszerű módon a kulcsokhoz is hozzájuthatunk a

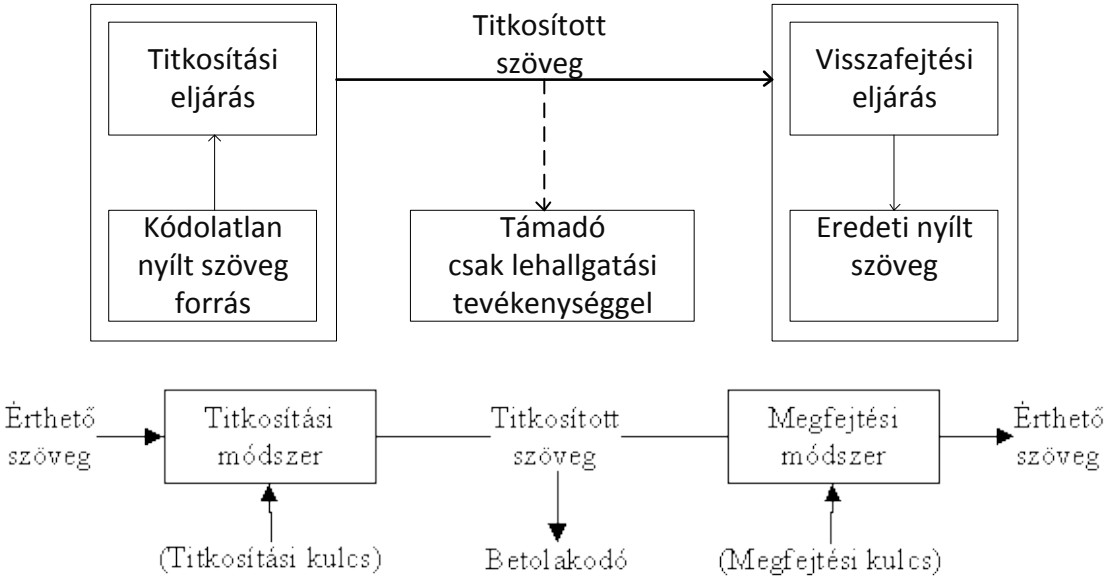
$$k_B = y_1 + y_2 \qquad k_A = y_2 + y_3$$

összefüggések kiszámításával.

Rejtett kulcsú kódolás: a rejtett üzenetet egy nyilvános csatornán, míg a kulcsot egy titkos csatornán továbbítjuk. A csatorna védettsége attól függ, hogy a csatorna nyitott-e vagy titkos.

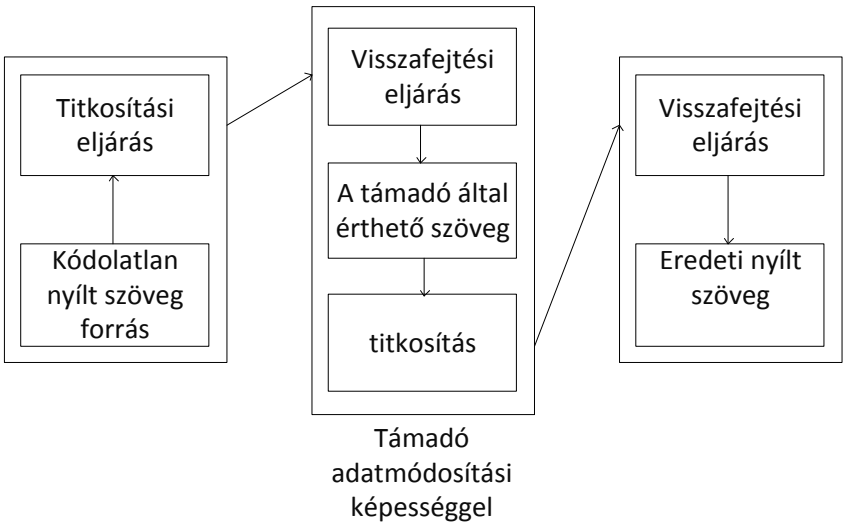
A támadó célja lehet az üzenet, vagy a kulcs megszerzése (feltételezve, hogy a kódoló, illetve dekódoló algoritmust ismeri). A titkosító algoritmus által nyújtott védettség nem haladhatja meg a kulcsa védettségének mértékét.

A támadás lehet aktív, vagy passzív. Passzív támadás a lehallgatás, azaz amikor a támadó a nyilvános csatornán keresztül valamilyen információhoz hozzájut, és ez alapján próbálja megfejteni a kulcsot (rejtjelfejtés). A támadás lehet: rejtett szövegű (a támadó birtokában van az azonos kulccsal kódolt üzeneteknek); nyílt szövegű (nyílt és rejtett üzenet párok birtokában); választható nyílt szövegű (a támadó megválaszthatja, hogy mely nyílt üzenet párját szeretné látni).




**10.10 Passzív támadás**

Aktív támadásról beszélünk akkor, ha a támadó kivonja a rejtett üzenetet a csatornából, és számára kedvezően módosítva továbbítja. Másik aktív módszer, amikor a támadó egy legális felhasználó szerepét próbálja eljátszani. Az üzenet titkos, ha csak a legális partner tud hozzájutni a tartalmához. Az üzenet hiteles, ha olyan valaki generálja, aki a kulcs legális birtokában van.




**10.11 Aktív támadás**

A támadó feltörte az algoritmust, ha „gyorsan” meg tudja állapítani az üzenet tartalmát. A „gyorsan” azt jelenti, hogy időben fel tudja használni céljára, még mielőtt a megszerzett adatok aktualitásukat vesztik.

A modern titkosítási tudomány már nem azt tűzi ki célul, hogy egy adott rejtjelezett szöveg elméletileg feltörhetetlen legyen, hanem a gyakorlati feltörhetetlenségre törekednek. Általánosan elmondható, hogy olyan nehézségű titkosítást kell választanunk, hogy egy esetleges feltörési kísérlet erőforrás igénye (pénz, idő, emberi erőforrás) nagyobb legyen, mint a feltört információból elérhető haszon. Megfelelően biztonságos rejtjelező algoritmus választása esetén a feltörés annál több időt, illetve egyéb erőforrást vehet igénybe, minél hosszabb kulcsot választunk a rejtjelezéshez. Tehát ha nem ismerjük a titkosító kulcsot, akkor a szöveg megfejtéséhez szükséges erőforrások (jellemzően: idő, ráfordított költség) arányosak (exponenciálisan arányosak!) a lehetséges kulcsok számával. Ha az algoritmust megfelelően írták meg, nincs lehetőség az algoritmus visszafejtésére. Az egyetlen lehetséges feltörési mód a kulcsok próbálgatása, melyet angolul brute-force-nak nevezünk. Egy átlagos számítógép másodpercenként 10 ezres nagyságrendű kulcsot tud kipróbálni a visszafejtendő adatokon. 128 bites kulcsok használatával a kipróbálandó kulcsok száma akkora, hogy a föld összes számítási kapacitását igénybe véve is hosszabb időre lenne szükség, mint ahány éve a Világegyetem létezik.

## *Üzenethitelesítő kódok*

Az üzenethitelesítés feladata a kommunikációs csatornán átküldött üzenetek hitelességének és integritásának biztosítása. Pontosabban, az üzenethitelesítés lehetővé teszi az üzenet vevője számára a küldő identitásának ellenőrzését és az átvitel során az üzenetben bekövetkezett változások (melyek származhatnak véletlen hibából vagy szándékos módosításból) detektálását.

Az üzenethitelesítést leggyakrabban üzenethitelesítő kódok alkalmazásával valósítják meg. Egy üzenethitelesítő kódra gondolhatunk úgy, mint egy kriptográfiai ellenőrző összegre, amit a küldő az üzenet elküldése előtt kiszámít, és az üzenethez csatol. A csatornán átvitelre kerül az üzenet és az üzenet ellenőrző összege is. A vevő mindkettőt veszi, majd ellenőrzi az ellenőrző összeget. Fontos megjegyezni, hogy az üzenethitelesítő kód értéke nemcsak magától az üzenettől függ, hanem egy a küldő és a vevő által megosztott titkos kulcstól is. A támadó ezen titok hiányában nem tud fabrikált vagy módosított üzenetekhez érvényes üzenethitelesítő kódot előállítani. A vevő tehát meg lehet győződve arról, hogy minden helyes üzenethitelesítő kóddal vett (és nem saját magától származó) üzenet csakis a vélt (például az üzenetben megjelölt) küldőtől származhat, és annak integritása sértetlen.

## *Nyilvános, aszimmetrikus kulcsú kódolás – az RSA algoritmus*

Az RSA algoritmus publikációja számos helyen megtalálható az interneten (ebből is látszik, hogy bár az eljárás publikus, mégsem fejthető vissza a vele titkosított üzenet). Az alábbi összefoglaló Ködmön József műve [KÖDMÖNJ] alapján készült.

Az RSA-titkosításhoz egy nyílt és egy titkos kulcs tartozik. A nyílt kulcs mindenki számára ismert, s ennek segítségével kódolhatják mások nekünk szánt üzeneteiket. A nyílt kulccsal kódolt üzentet csak a titkos kulccsal tudjuk "megfejteni". Az eljárás első lépése a nyílt-titkos kulcspár legenerálása. Az RSA-eljáráshoz a következő módon generáljuk a kulcsokat:

Ez az algoritmus **Fermat kis tételén** alapul.

E szerint a tétel szerint, ha $\boldsymbol{p}$ prímszám, és nem osztója $\boldsymbol{a}$ egésznek, akkor

$$\boldsymbol{a^{(p-1)}-1}$$

osztható $\boldsymbol{p}$-vel. A tétel alapján, ha $\boldsymbol{p}$ és $\boldsymbol{q}$ különböző prímszámok, és $\boldsymbol{a}$-nak egyik sem osztója, akkor mind $\boldsymbol{p}$, mind pedig $\boldsymbol{q}$ osztója $\boldsymbol{a^{(p-1)(q-1)}-1}$ értéknek, ami képlettel leírva:

$$qp|a^{(p-1)(q-1)}-1$$

Ez nem más, mint a kis Fermat tétel, csak a tételbeli képletben a helyére egyszer $\boldsymbol{a^{p-1}}$, egyszer pedig $\boldsymbol{a^{q-1}}$ kerül rendre a $\boldsymbol{q}$-val, illetve $\boldsymbol{p}$-vel való oszthatóságot felírva.

Mivel $\boldsymbol{p}$ és $\boldsymbol{q}$ különböző prímek, ezért a szorzatukkal is osztható $\boldsymbol{a^{(p-1)(q-1)}-1}$

Legyen $\boldsymbol{n=qp}$.

Ekkor $\boldsymbol{a^{(p-1)(q-1)}+1}$ pont $\boldsymbol{a}$ maradékot ad $\boldsymbol{n}$-nel osztva, ha $\boldsymbol{a}$ kisebb, mint $\boldsymbol{n}$.

Legyen $\boldsymbol{ef=(p-1)(q-1)+1}$ szorzat alakban felírva. Ekkor az

$$\boldsymbol{a^{ef} \bmod n = a}$$

egyenlethez jutottunk, ahol a **mod** a maradékképzést jelenti.

Legyen a nyilvános kulcs az $\boldsymbol{e,n}$ számpáros, a titkos kulcs pedig az $\boldsymbol{f}$ szám.

A kódolás során az üzenetet először számokká alakítjuk át olyan módon, hogy a számok mindegyike kisebb legyen, mint $\boldsymbol{n}$. Ezután az egyes $\boldsymbol{m}$ számokat az

$$\boldsymbol{M = m^e \bmod n}$$

képlettel kódoljuk előállítva a rejtjelezett $\boldsymbol{M}$ üzenetet, és ezt az üzenetet az

$$\boldsymbol{M = m^f \bmod n}$$

képlet alapján lehet dekódolni.

A felhasznált számoknak olyan nagyoknak kell lenniük, hogy az $\boldsymbol{n}$ számot ne lehessen prímtényezőkre bontani. Ha ugyanis az $\boldsymbol{n}$ számot fel tudjuk bontani $\boldsymbol{n=qp}$ alakra, akkor $\boldsymbol{e}$ alapján a maradékos inverz megkeresésével lehet meghatározni $\boldsymbol{f}$-et.

A prímtényezős felbontásra pillanatnyilag *nem áll rendelkezésre hatékony algoritmus*, bár az sem bizonyított, hogy ilyen algoritmus nem létezik. Mivel az alapvető aritmetikai műveletek, mint szorzás, összeadás, hatványozás hatékonyan elvégezhetőek, ezért lehetséges olyan nagy $\boldsymbol{p}$ és $\boldsymbol{q}$ használata, amely esetén $\boldsymbol{n}$ rövid idő alatt nem bontható fel szorzattá.

# Irodalomjegyzék

[ARATÓP] Dr. Arató Péter: Logikai rendszerek tervezése (Műegyetemi Kiadó, 1992)

[KERESZTP] Dr. Keresztes Péter: Digitális Hálózatok (Széchenyi Egyetem, 2006)

[KÖDMÖNJ] Ködmön, J.: Kriptográfia (ComputerBooks, 1999)