II. modul: Sorrendi hálózatok tervezése

1. lecke: Elemi sorrendi hálózatok; a sorrendi hálózatok struktúrái és modelljei; aszinkron és szinkron tárolók: SR, D-MS és JK-MS tárolók használata Mealy- és Moore-típusú hálózatok esetén.

Cél:

A lecke célja, hogy a hallgató megismerje a sorrendi hálózatok fogalmát, típusait és azok struktúráit. Ismerje az aszinkron és szinkron hálózatokban alkalmazott tárolókat, felépítésük és viselkedésük sajátosságait, valamint kapuszintű megvalósítási lehetőségeit, és sorrendi hálózatokban történő felhasználásának lehetséges módjait.

Követelmények:

Ön akkor sajátította el megfelelően a tananyagot, ha

- meg tudja határozni a sorrendi hálózatok fogalmát és típusait
- ismeri a Mealy- és Moore típusú hálózatok felépítését
- fel tudja sorolni a sorrendi hálózatokban alkalmazott tárolókat és azok jellegzetességeit

Időszükséglet:

A tananyag elsajátításához kb. 150 percre van szükség.

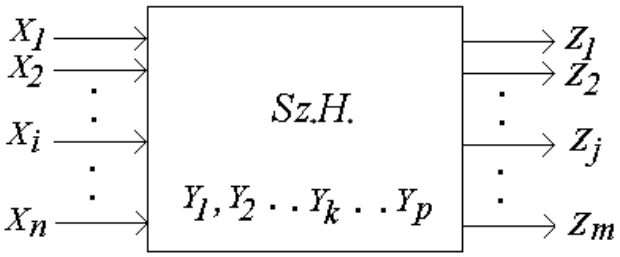
Kulcsfogalmak:

- szekvenciális hálózat
- Mealy-modell, Moore-modell
- aszinkron tároló, szinkron tároló, flip-flop,
- master-slave
- clock, preset, clear
- állapottábla

Tevékenység: tanulmányozza a jegyzetben és rajzolja fel a füzetében a sorrendi hálózatok Mealy- és Moore-modell szerinti struktúráját. Adja meg az aszinkron és szinkron tárolók működésének igazság- és vezérlési tábláját!

1.1 A sorrendi hálózatok „fekete-doboz" modellje

A sorrendi, más néven szekvenciális hálózatok „fekete-doboz" modellje a 1.1.a ábrán látható:



1.1.a ábra[1]

Szekvenciális hálózat „fekete-doboz" modellje

A hálózat működését a bemeneti($X_1, X_2, \ldots X_i, \ldots X_n$) és kimeneti($Z_1, Z_2, \ldots Z_j, \ldots Z_m$) jelek mellett az ún. másodlagos vagy szekunder változók($Y_1, Y_2, \ldots Y_k, \ldots Y_p$) is jellemzik. Ezek jelenléte biztosítja a „szekvencialitást", vagyis a sorrendi hálózatok azon tulajdonságát, mely szerint a kimeneti kombináció nem csak a pillanatnyi bemeneti kombinációtól függ, hanem a korábbi bemeneti kombinációktól, sőt azok sorrendjétől is. A modellt leíró függvénycsoportok a következőként jellemezhetőek:

$$Z_1 = f_{z1}(X_1, \ldots X_i, \ldots X_n, Y_1, \ldots Y_k, \ldots Y_p)$$
$$Z_2 = f_{z2}(X_1, \ldots X_i, \ldots X_n, Y_1, \ldots Y_k, \ldots Y_p)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$Z_j = f_{zj}(X_1, \ldots X_i, \ldots X_n, Y_1, \ldots Y_k, \ldots Y_p)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$Z_m = f_{zm}(X_1, \ldots X_i, \ldots X_n, Y_1, \ldots Y_k, \ldots Y_p)$$

$$Y'_1 = f_{y1}(X_1, \ldots X_i, \ldots X_n, Y_1, \ldots Y_k, \ldots Y_p)$$
$$Y'_2 = f_{y2}(X_1, \ldots X_i, \ldots X_n, Y_1, \ldots Y_k, \ldots Y_p)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$Y'_k = f_{yk}(X_1, \ldots X_i, \ldots X_n, Y_1, \ldots Y_k, \ldots Y_p)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$Y'_p = f_{yp}(X_1, \ldots X_i, \ldots X_n, Y_1, \ldots Y_k, \ldots Y_p)$$

Az első csoport a kimeneteket adja meg a bemenetek és a szekunder változók függvényében, a másik az állapotváltozók új értékeit határozzák meg a bemenetek és az éppen fennálló (aktuális) szekunder változó értékek alapján. (A szekunder változók új értékeit felső vonással(*Y'*) különböztetjük meg az aktuálisaktól.)

Általánosságban a hálózat teljes működésének leírására kétféle függvényt kell definiálni: az első, amit kimeneti függvénynek nevezünk(f_z) a bemeneti kombinációk halmazának és a szekunder változók kombinációiból álló halmaznak a szorzatát képezi le a kimeneti kombinációk halmazába. A második(f_y) ugyanezt a szorzat halmazt a szekunder kombinációk halmazába képezi le. A szekunder változók itt kihasznált kombinációit a hálózat *belső állapotainak*, röviden *állapotainak* nevezzük. Az ***Y*** halmaz neve így állapothalmaz:

$$f_z : \mathbf{X} \times \mathbf{Y} \Rightarrow \mathbf{Z}$$
$$f_y : \mathbf{X} \times \mathbf{Y} \Rightarrow \mathbf{Y}$$
$$\mathbf{X} : a\ bemeneti kombinációk\ halmaza$$
$$\mathbf{Z} : a\ kimeneti kombinációk\ halmaza$$
$$\mathbf{Y} : a\ szekundér\ változók\ halmaza$$

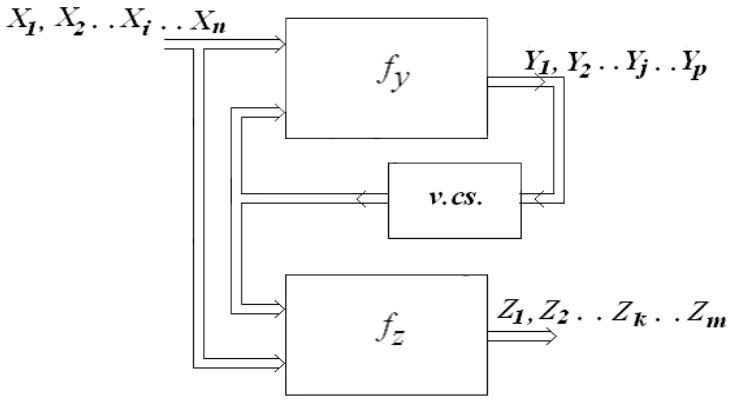
A kimeneti függvény megvalósítása egy olyan kombinációs hálózat, amelynek bemeneteire a hálózat bemenetei és az állapotváltozók csatlakoznak, tehát a bemeneti kombináció és állapot kombináció párok alkotnak egy bemeneti kombinációt az f_z hálózaton. Ha ezeket egy *t* időpontbeli értékkel jellemezzük, akkor a hálózat kimenetein valamilyen tranziens(Δt) után előállnak a hozzárendelt kimeneti kombinációk. Ugyanakkor ugyanez a páros a másik, f_y hálózat bemeneteire érkezve egy másik tranziens után egy új, $t+\Delta t$ időpontbeli állapotot hoz létre, amely visszakerül az f_y-al jellemzett hálózat bemeneteire.

$$f_y : (x_i^t, y_k^t) \rightarrow y_l^{t+\Delta t}$$
$$f_z : (x_i^t, y_k^t) \rightarrow z_j^t$$
$$x_i^t : egy\ bemeneti\ kombináció\ a\ t\ \ pillanatban$$
$$z_j^t : egy\ kimeneti\ kombináció\ a\ t\ \ pillanatban$$
$$y_k^t : egy\ szekundér\ változó\ kombináció\ a\ t\ \ pillanatban$$
$$y_l^{t+\Delta t} : egy\ szekundér\ változó\ kombináció\ a\ (t+\Delta t)\ \ pillanatban$$

1.2 A sorrendi hálózatok strukturális modelljei

1.2.1 Mealy-típusú sorrendi hálózat

A szekvenciális hálózat legáltalánosabb belső struktúrája a 1.2.1.a ábrán látható:



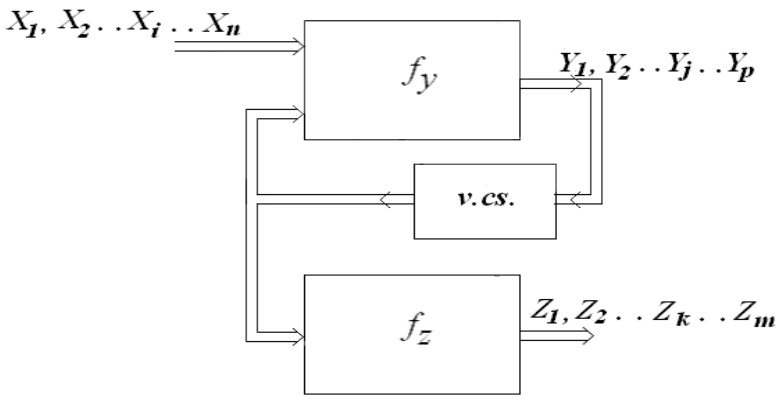
1.2.1.a ábra[1]

Mealy-típusú sorrendi hálózat sematikus ábrája

A struktúra az előző pontban bemutatott függvény-kapcsolatokat is tükrözi. *Azt a szekvenciális hálózatot, amelynek f_z kimeneti hálózatára – az egyenletekkel is bemutatott modellnek megfelelően - mind a bemenetek, mind az állapot változók rácsatlakoznak, Mealy-típusú sorrendi hálózatnak nevezzük.*

1.2.2 Moore-típusú sorrendi hálózat

A sorrendi hálózatoknak azt a típusát, amelynek kimeneti kombinációira csak a belső állapotok hatnak, Moore-típusú sorrendi hálózatnak nevezzük. A Moore-féle szekvenciális hálózatban a kimeneti kombinációk a belső állapotok átkódolt formáit szolgáltatják a kimeneteken (1.2.2.a ábra).



1.2.2.a ábra[1]

Moore-típusú sorrendi hálózat sematikus struktúrája

A Mealy- és Moore-típusú hálózatstruktúrák aszinkron és szinkron sorrendi hálózatok esetében egyaránt alkalmazható.

1.3 Aszinkron és szinkron sorrendi hálózatok

1.3.1 Aszinkron sorrendi hálózatok

Az aszinkron sorrendi hálózat f_y hálózatának visszacsatolása órajel nélküli, ún, 'direkt' visszacsatolás. Ezt a direkt visszacsatolást vagy *közvetlenül*, huzalozással hozzuk létre, vagy *S-R tárolókat* helyezünk el a visszacsatoló körben. Mindkét módszer közös sajátossága, hogy a visszacsatolás a hálózat saját késleltetési idejének megfelelően érvényesül. Egy stabil y_j állapot adott x_i bemeneti kombinációra csak akkor áll be, ha az f_y leképezésre igaz, hogy

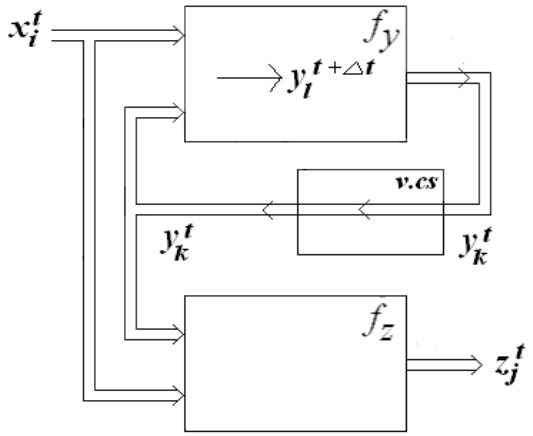
$$f_y(x_i, y_j) = y_j$$

1.3.1.1 Közvetlen visszacsatolású aszinkron sorrendi hálózatok

A 1.3.1.1.a ábra egy közvetlenül, a kimenetről vezetékekkel visszacsatolt aszinkron sorrendi hálózat struktúrájának egy olyan működési fázisát ábrázolja, amikor egy új bemeneti kombinációt már rákapcsoltunk a hálózatra, és az f_y kombinációs hálózat belsejében már megindult a következő állapot kombináció generálása($t+\Delta t$). Az f_y hálózat kimenetein azonban a változás még nem jelent meg(t), az egyelőre még az

aktuális állapot kombinációt őrzi. Az f_z kimenetein ugyancsak átmeneti állapot van, hiszen a bemeneti kombináció már megváltozott, az aktuális állapot kombináció ugyanakkor még uralkodik a bemenetein.

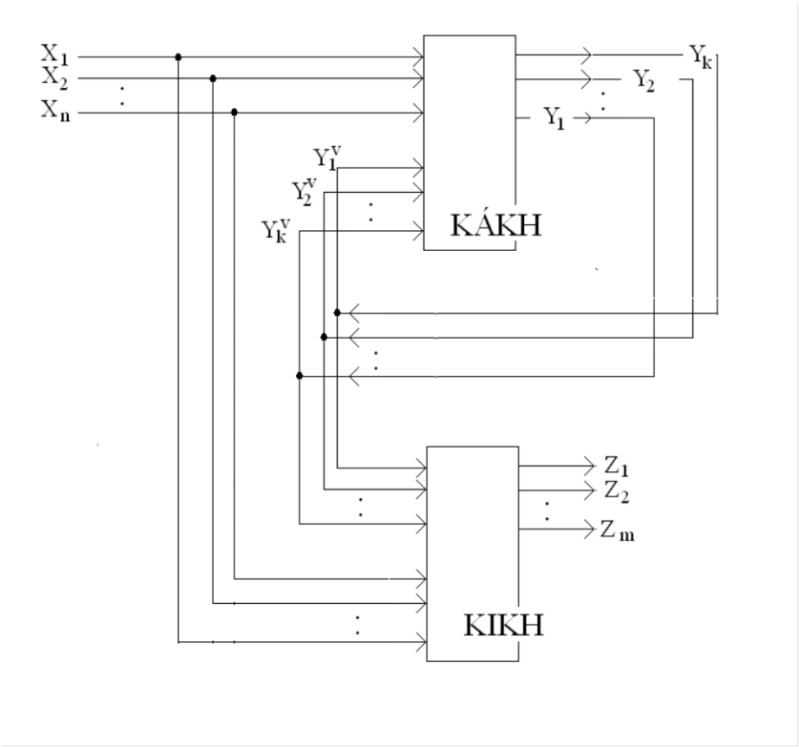
Tegyük fel, hogy a következő állapot kombináció az új bemeneti kombinációval olyan párost alkot az f_y bemenetein, amelyhez az f_y hálózat az ugyanezt a következő állapot-kombinációt rendeli. Ez azt jelenti, hogy az új állapot kombináció *stabilizálódik(t+Δt)*. Így végül a kimeneti kombináció is nyugalomba jut, a stabil új aktuális állapothoz és az eseménysort kiváltó bemeneti kombinációhoz rendelt f_z érték jelenik meg a kimeneteken. Ha ezt követően megváltoztatjuk a bemeneti kombinációt, új tranziens indul meg.



1.3.1.1.a ábra[1]

Közvetlen visszacsatolásos szekvenciális hálózat sematikus ábrája

A 1.3.1.1.b ábrán egy közvetlen visszacsatolással felépített Mealy-típusú aszinkron sorrendi hálózat sematikus ábrája látható. (Az ábrán az *n*-edik időpillanathoz képesti következő, *n+1*-edik időpillanatban bekövetkező állapotot előállító kombinációs hálózat jelölése 'KÁKH', a kimeneti értékeket(Z_m) előállító kombinációs hálózat jelölése 'KIKH'. A visszacsatolt Y_k állapot változók jelölése Y_k^v.)

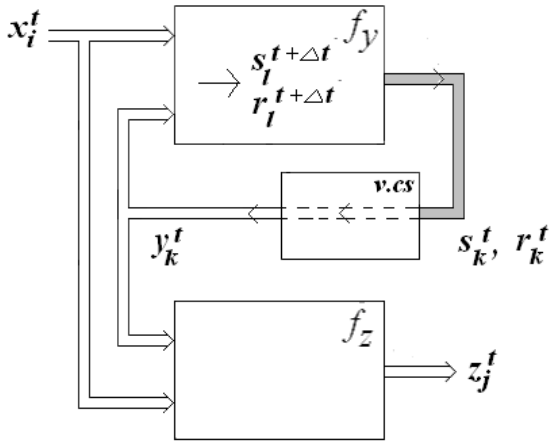


1.3.1.1.b ábra[1]

Mealy-típusú aszinkron sorrendi hálózat felépítése közvetlen visszacsatoló ágakkal

1.3.1.2 SR tárolókkal visszacsatolt aszinkron sorrendi hálózatok

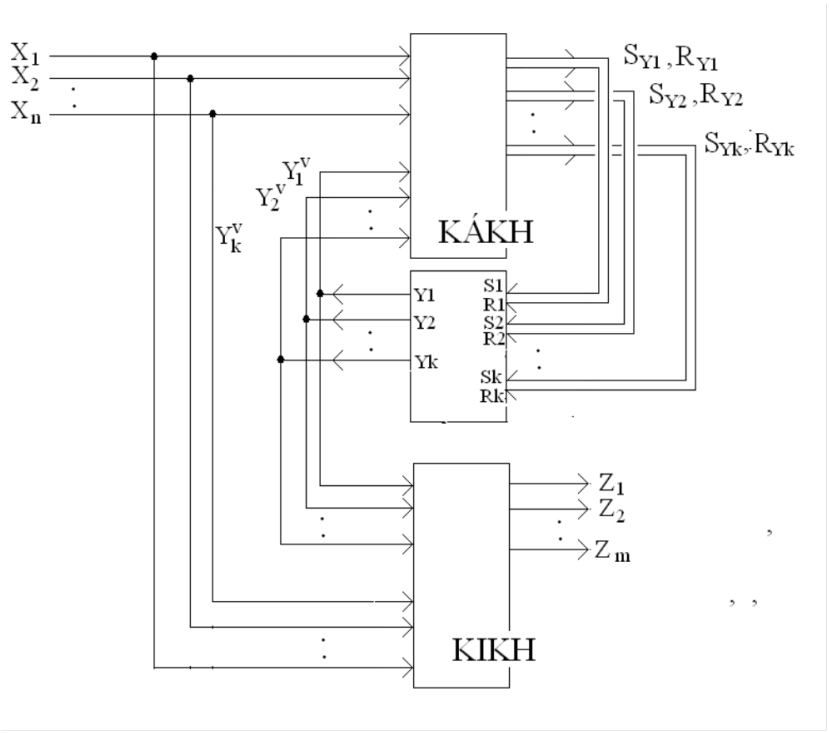
A 1.3.1.2.a ábrán egy olyan SR tárolókkal visszacsatolt aszinkron sorrendi hálózat struktúráját láthatjuk, ami azt a működési pillanatot rögzíti, amikor egy új bemeneti kombinációt már rákapcsoltunk a hálózatra, és az f_y kombinációs hálózat belsejében már megindult a következő állapotot generáló SR kombináció kialakulása ($t+\Delta t$). Az f_y hálózat kimenetein azonban a változás még nem jelent meg (t), az egyelőre még az aktuális állapotot generáló SR kombinációt őrzi. Az f_z kimenetein ugyancsak átmeneti állapot van, hiszen a bemeneti kombináció már megváltozott, az aktuális állapot kombináció ugyanakkor még uralkodik a bemenetein. Tegyük fel, hogy a kialakuló következő állapot az új bemeneti kombinációval olyan párost alkot az f_y bemenetein, amelyhez az f_y hálózat az ugyanezt a következő állapot kombinációt generáló SR kombinációt rendeli. Ez azt jelenti, hogy az új állapot kombináció stabilizálódik ($t+\Delta t$). Így végül a kimeneti kombináció is nyugalomba jut, a stabil új aktuális állapothoz és az eseménysort kiváltó bemeneti kombinációhoz rendelt f_z érték jelenik meg a kimeneteken. Ha ezt követően megváltoztatjuk a bemeneti kombinációt, új tranziens indul meg.



1.3.1.2.a ábra[1]

SR tárolókkal visszacsatolt aszinkron sorrendi hálózat struktúrája

A 1.3.1.2.b ábrán egy olyan Mealy-típusú aszinkron sorrendi hálózat struktúrája látható, melynek visszacsatoló ágaiban SR tárolók találhatók.



1.3.1.2.b ábra[1]

Mealy-típusú aszinkron sorrendi hálózat felépítése SR tárolós visszacsatoló ágakkal

1.3.2 Szinkron sorrendi hálózatok

A szinkron sorrendi hálózat f_y hálózatának visszacsatolása órajellel('clk'), ún. MS (Master-Slave) tárolókon keresztül érvényesül. A gyakorlatban vagy D-MS, vagy JK-MS tárolós visszacsatolást használnak. Mindkét módszer közös sajátossága, hogy az órajel minden állapot kombináció fennállásának időintervallumát egyértelműen kijelöli, függetlenül attól, hogy az f_y visszaadja-e a rákapcsolt aktuális állapot kódot, vagy nem. Így feleslegessé válik az instabil és a stabil állapotok megkülönböztetése.

A szinkron hálózatok az egymást követő állapotokat és kimeneti kombinációkat időben diszkrét sorozatokká alakítják. Ennek megfelelően sajátos jelöléseket alkalmazhatunk, a *t* időbeli kombinációkat inkább az *n* természetes számmal, a *t+Δt* időbeli kombinációkat inkább az *n+1* természetes számmal, mint sorszámokkal jelöljük. A szinkron tárolók (D-MS, JK-MS) kimeneti kombinációinak jelölésére inkább a már megismert *q* szimbólumot használjuk. Megjegyezzük, hogy az ábrákon mindkét jelölésrendszer látható, de a későbbiekben szinkron hálózatok esetén csak a most bevezetett jelölésrendszert alkalmazzuk.

1.3.2.1 Szinkron hálózatok visszacsatoló ága D-MS tárolókkal

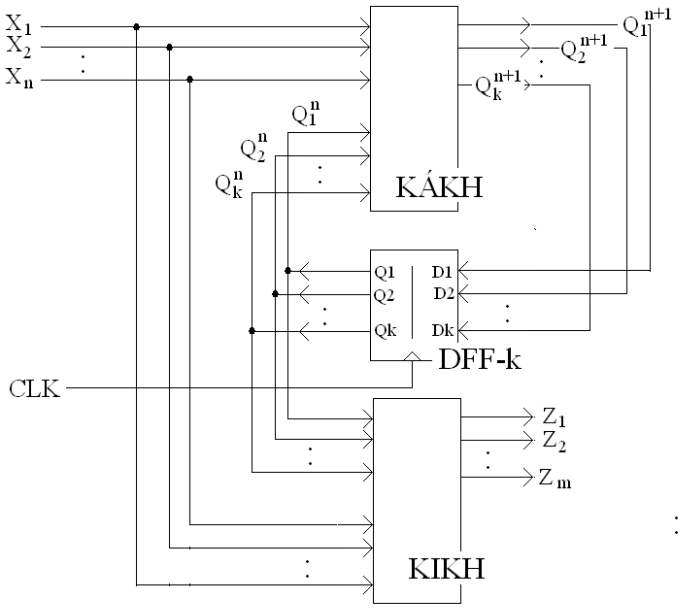
A 1.3.2.1.a és 1.3.2.1.b ábrákon egy-egy szinkron sorrendi hálózat látható Mealy-, illetve Moore-típusú hálózat struktúrában, D-MS tárolókat tartalmazó visszacsatolásokkal.

Értelmezzük a Mealy-típusú hálózat működését a 1.3.2.1.a ábra tanulmányozásával! ezen az ábrán egy olyan időpillanat látható, amikor is az *n-1*-edik órajel ciklus már lejátszódott, és az *n*-edik felfutó él következik. Ennek megfelelően a hálózatra már rákapcsoltuk az *n*-edik ütemnek megfelelő bemeneti kombinációt, és megjelent az ennek megfelelő kimeneti kombináció is. Az f_y kombinációs hálózat a kimenetein előállítja a tárolók bemeneteinek kombinációját. Ezek azonos kombinációk a megkívánt következő állapot kombinációkkal, hiszen a D típusú tárolók kimeneteiken megismétlik a bemeneteikre kerülő értékeket. Míg egy adott $Q_k^{n+1} (= D_k^{n+1})$ kombináció a visszacsatoló flip-flopok bemenetein várakozik, az f_y hálózatra csatlakozó kimeneteik változatlanok, és az aktuális állapot kombinációt, a Q_k^n-et őrzik. Az f_z kombinációs hálózat a bemenetére jutó aktuális állapot kombináció és a bemeneti kombináció eredményeképpen szolgáltatja az aktuális kimeneti kombinációt. Ekkor érkezik az órajel felfutó éle. A következő állapotkód a 'mester' tárolókba kerül, miközben a 'szolga' kimenetek változatlanok.

A következő változás akkor következik be, amikor az órajel lefutó éle megérkezik. Ennek hatására a D-MS tárolók kimenetein megjelenik az eddig következő állapot

kombinációnak nevezett kombináció, és aktuális állapottá válik. Ha ezt követően megváltoztatjuk a bemeneti kombinációt, akkor az f_z kimenetein egy új aktuális kimeneti kombináció, és az f_y kimenetein egy újabb következő állapot kódja jelenik meg.

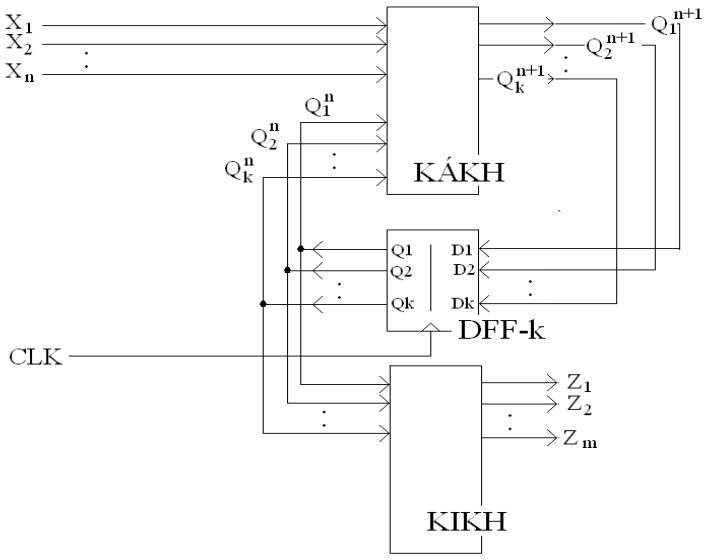
Megjegyzés: a MS tárolók egy speciális csoportját alkotják az élvezérelt tárolók, más néven „flip-flop"-ok. Kialakításukkal és működésükkel a következő, szinkron tárolókat is bemutató fejezetben ismerkedünk meg részletesen.



1.3.2.1.a ábra[1]

Mealy-típusú szinkron sorrendi hálózat D-MS tárolókkal

A Moore-típusú hálózat struktúra esetében a folyamat hasonlóképpen értelmezhető, de ebben az esetben természetesen a kimeneti értékek(Z_m) előállítása (KIKH) csupán a D flip-flopok által szolgáltatott szekuder változók értékei alapján történik (1.3.2.1.b ábra):

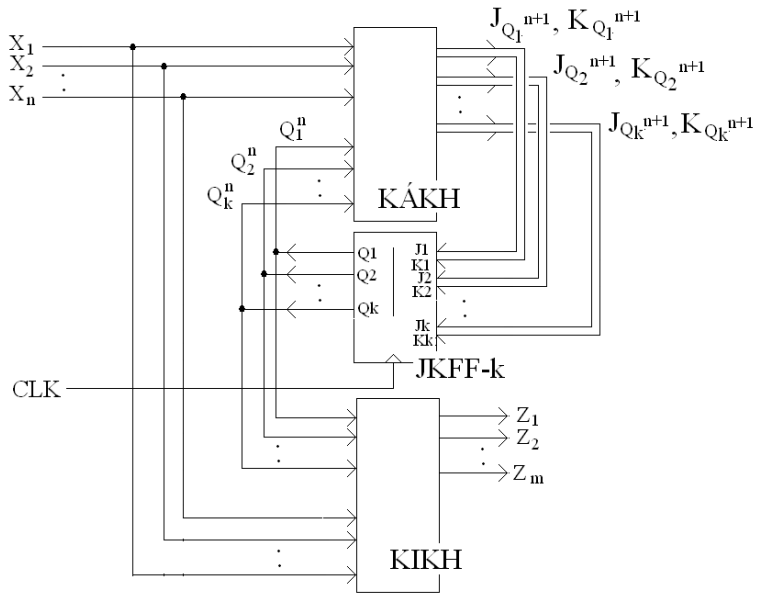


1.3.2.1.b ábra[1]

Moore-típusú szinkron sorrendi hálózat D-MS tárolókkal

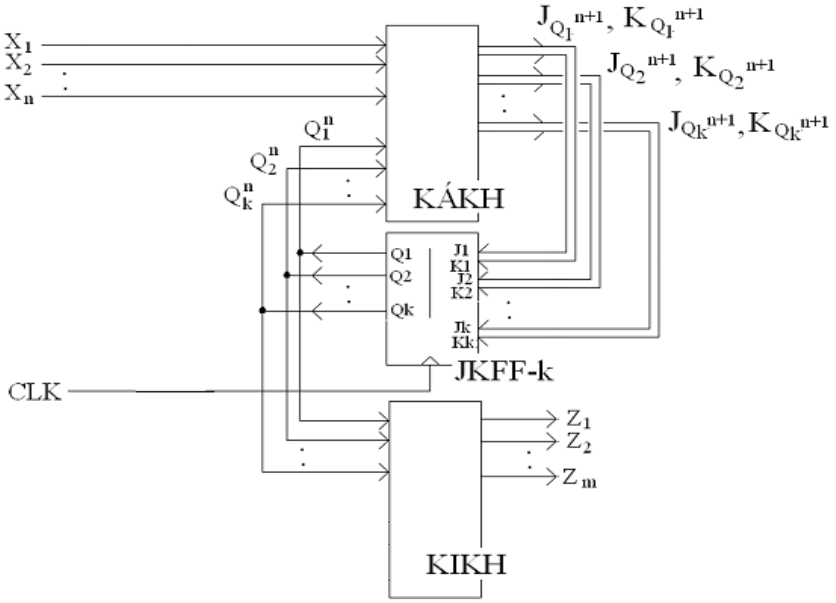
1.3.2.2 Szinkron hálózatok visszacsatoló ága JK-MS tárolókkal

JK-MS tárolóelemek alkalmazása esetében hasonló kialakítású Mealy- és Moore-struktúrákat láthatunk 1.3.2.2.a és 1.3.2.2.b ábrák), mint a D-MS tárolók használatánál, de ebben az esetben természetesen a visszacsatoló ágakban a D bemenetek helyett most külön J és K flip-flop bemeneteink vannak:



1.3.2.2.a ábra[1]

Mealy-típusú szinkron sorrendi hálózat JK-MS tárolókkal



1.3.2.2.b ábra[1]

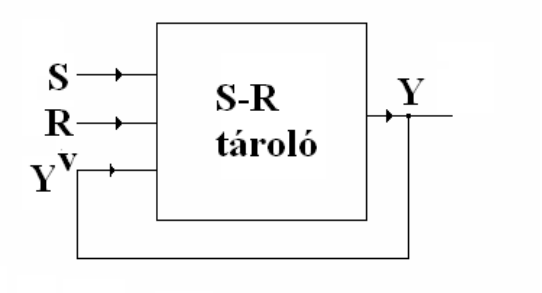
Moore-típusú szinkron sorrendi hálózat JK-MS tárolókkal

1.4 Tárolók sorrendi hálózatokban

1.4.1 Az SR tároló

A sorrendi hálózatokban a szekvencia elvének biztosítására –az előbbiekben megismertek szerint- különböző megoldások léteznek: egyrészt alkalmazhatunk közvetlen visszacsatolást (időbeni késleltetés elve a következő állapot(ok) és a kimeneti érték(ek) előállítására), illetve használhatunk különböző felépítésű, specifikus tárolóelemeket.

Aszinkron sorrendi hálózatokban gyakran használt tárolóelem az aszinkron SR tároló (egyes esetekben RS tárolóként is szokás említeni), melynek két bemenete S(set) és R(reset), kimenete Y, sematikus felépítése a 1.4.1.a ábrán látható:



1.4.1.a ábra[1]

Aszinkron SR tároló sematikus ábrája

Jól látható, hogy lényegében egy olyan kombinációs hálózatról van szó, melynek kimenete a bemenetre, mint „harmadik" bemeneti logikai váltózó érkezik vissza. Működését az egyszerű, illetve összetett igazságtábla alapján lehet értelmezni (1.4.1.b ábra):

egyszerű igazságtábla

S	R	Y
0	0	Y^V
0	1	0
1	0	1
1	1	-

összetett igazságtábla

S	R	Y^V	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

1.4.1.b ábra[1]

Az aszinkron SR tároló igazságtáblái

Értelmezzük az egyszerű igazságtábla alapján az SR tároló működését: ha a tároló mindkét bemenetére ’0’-t kapcsolunk, a kimenet nem változik. Ha csak az R bemenetet emeljük fel, akkor ennek hatására a kimenet aktuális szintjétől függetlenül ’0’-ra áll. Ha csak az S bemenetet emeljük fel, akkor a kimenet aktuális szintjétől függetlenül ’1’-re áll. Mindkét bemenet (S,R) együttes magas szintje tiltott kombiációnak számít („don’t care”).

Vizsgáljuk meg részletesen az első esetet (SR=00), vagyis amikor mind az S, mind pedig az R bemenet is alacsony szinten van! Azt látjuk a táblázatban, hogy ilyenkor a tároló Y kimenete megőrzi a korábbi értékét (Y^V), vagyis attól függően hol ’1’, hol pedig ’0’ logikai szinten van SR=00 esetén. Ez tehát azt jelenti, hogy ebben az esetben egyértelműen egy sorrendi hálózattal állunk szemben, hiszen ugyanarra az adott bemeneti bitkombinációra más-más választ ad a kimenetünk (attól függően, hogy <u>*korábban*</u> milyen más bemeneti értéket kapcsoltunk a hálózatunkra). Példánkban tehát a kimenet saját korábbi értékétől is függ.

Az egyszerű igazságtáblában tehát a kimenetre vonatkozó következő állapot(érték) oszlopában a határozott logikai értékek mellett szimbólumo(ka)t is találhatunk. A teljes működés értelmezéséhez fejtsük ki az egyes következő kimeneti értékekhez tartozó szimbólumok logikai értékeit a bemenetek, illetve az előző kimeneti érték függvényében: így kapjuk az összetett igazságtáblát. Ezzel egy olyan speciális kombinációs hálózatot tervezünk, amelyen ugyanaz a változó kimenetként és bemenetként is szerepel - azaz a kimenet a bemenetre vissza van vezetve - de az igazságtáblába írt értékek különbözhetnek, hiszen időbeli eltolódás van közöttük. Formálisan meg is különböztetjük a bemenetként szereplő változót a kimenetként szereplő változótól. A

bemeneten az Y szimbólumot ellátjuk egy felső „v” (visszacsatolt) felső indexel. Ennek megfelelően az Y^v szintjeit tekintjük aktuális kimeneti értékeknek, és a Y szintjeit következő kimeneti értékeknek.

Az összetett igazságtábla kitöltésekor fontos, hogy az S=1, R=1 bemeneti kombináció tiltott, ezért ott élhetünk a közömbös kimenet előírással.

1.4.1.1 Az állapot-átmeneti tábla (állapottábla)

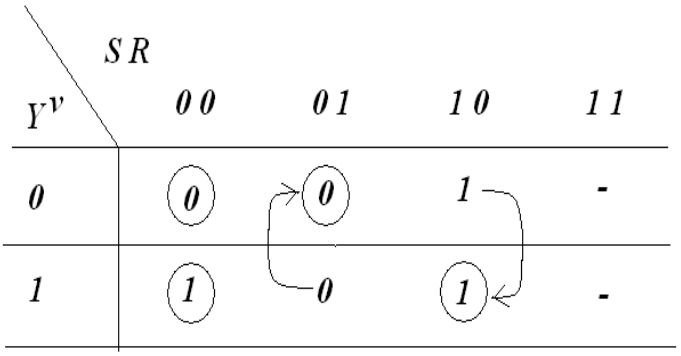
Adjuk meg az SR tároló viselkedésének leírását egy ún. *állapot-átmeneti tábla* segítségével! Ezt a táblát rövidebb meghatározással *állapottáblának* is szokták nevezni. Megadásának módja pedig a következő (1.4.1.1.a ábra):

a táblázat bal szélső oszlopban felsoroljuk az aktuális kimeneti állapotokat, a többi oszlopot pedig a bemeneti kombinációkkal jelöljük. Az összetett igazságtábla értékeit egyszerűen bemásoljuk ebbe az új struktúrájú táblázatba.

Nagyon fontos annak megjelölése, hogy a bejegyzett következő állapot csak átmenetileg jelentkező (tranziens), vagy a rákapcsolt bemeneti kombináció fenntartása mellett nem változik (stabil).

Vizsgáljuk meg, hogyan állapítható meg az állapottáblából a stabilitás vagy az instabilitás ténye: ha a bemenetekre az SR=00 bemeneti kombinációt kapcsoljuk, és az aktuális kimeneti érték 0, azaz $Y^v=0$, akkor a következő állapot is 0, és ez a bemenetre visszakerülve nem változtat az új következő kimeneti értéken. Azaz az SR=00 bitkombinációnál az Y=0 stabil kimeneti állapot. Ezzel szemben az SR=10 és $Y^v=0$ állapot instabil, hiszen az erre következő kimeneti állapot az Y=1. Ez viszont stabilizálódik, hiszen visszakerülve a bemenetre, nem vált ki újabb változást.

Ennek alapján az állapottábla azon kimeneti állapotai stabilak, amelyeknél a bejegyzett következő kimeneti állapot megegyezik az aktuális kimeneti állapottal. A stabil állapotokat a táblázatban bekarikázva jelöljük.

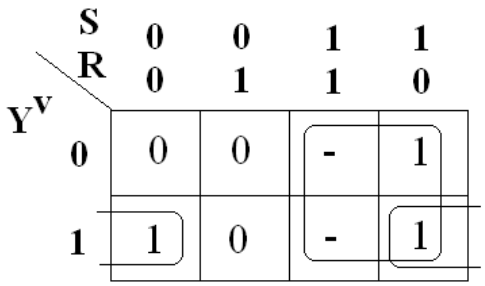


1.4.1.1.a ábra[1]

Az SR tároló állapottáblája

1.4.1.2 Az SR tároló működésének megadása algebrai függvényalakkal és logikai kapuhálózattal

A sorrendi hálózatok viselkedését leíró állapottábla felhasználásával könnyen megadható egy olyan Karnaugh tábla, melynek segítségével felírható – a már megismert minimalizálási eljárás alapján- az adott hálózat legegyszerűbb algebrai alakja. Ehhez nem kell mást tennünk, mint megszerkeszteni a megfelelő számú bemenetekkel rendelkező Karnaugh táblát, és egyszerűen csak át kell másolnunk az állapottáblából az adott kombinációkhoz tartozó következő állapot(kimenet) értékeket. SR tároló esetében ez a következőképpen néz ki (1.4.1.2.a ábra):



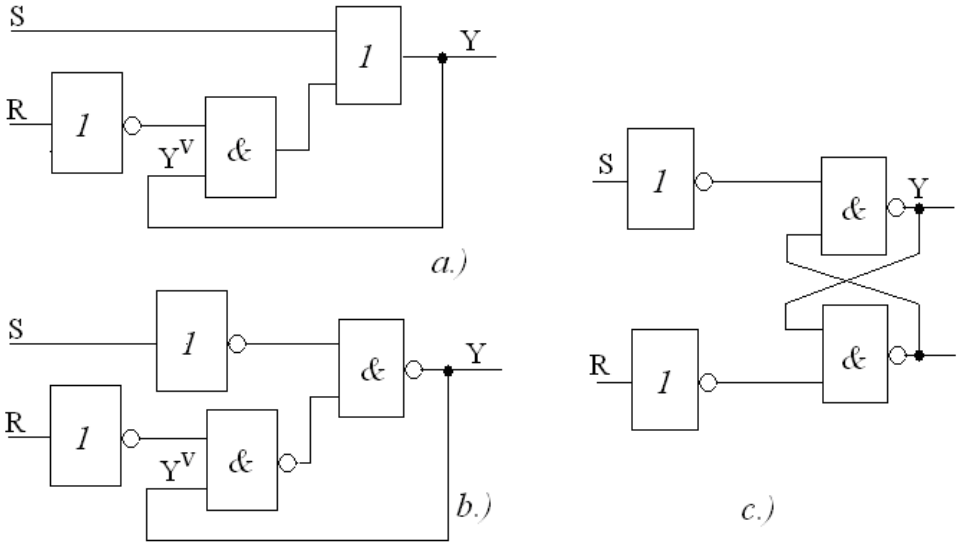
1.4.1.2.a ábra[1]

SR tároló Karnaugh táblája

A minimalizálás érdekében elvégzett lefedések eredményeképpen a következő algebrai alakokat kapjuk:

$$Y = S + \overline{R}\, Y^{v} = \overline{\overline{S + \overline{R}\, Y^{v}}} = \overline{\overline{S}(\overline{\overline{R}\, Y^{v}})}$$

Ezzel a megadási móddal lehetőség nyílik többféle kapurealizációra: AND-OR(1.4.1.2.b ábra *a.)* rajz), illetve NAND-NAND (1.4.1.2.b ábra *b.)* és *c.)* rajz):



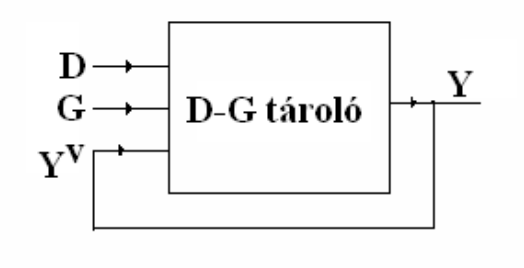
1.4.1.2.b ábra[1]

SR tároló kapuszintű realizációi

Megjegyzés: a korábbiakban már említettük, hogy NAND-NAND realizációkban az inverter, mint „kvázi NAND" elem használata megengedett.

1.4.2 A DG tároló

A DG tároló a 1.4.2.a ábra alapján lényegében egy egybites aszinkron tárolóelemnek tekinthető, mely a G(gate) bemenet felemelésekor írja a kimenetére(Y) a D(data) bemenete aktuális értékét, majd a G lefutásakor ez az érték tárolódik a kimeneten.



1.4.2.a ábra[1]

Aszinkron DG tároló sematikus ábrája

A tároló egyszerű és összetett igazságtáblája a 1.4.2.b ábrán látható:

egyszerű igazságtábla

D	G	Y
0	0	Y^V
0	1	0
1	0	Y^V
1	1	1

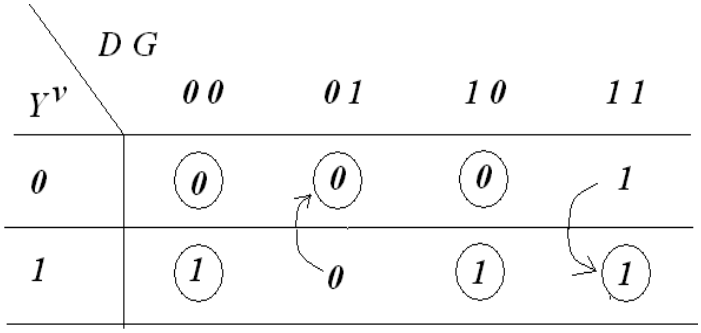
összetett igazságtábla

D	G	Y^V	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

1.4.2.b ábra[1]

Az aszinkron DG tároló igazságtáblái

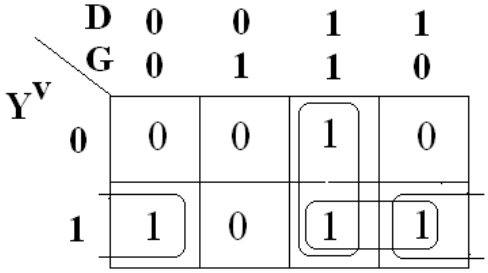
Az összetett igazságtáblából –a korábban megismert módon- könnyen származtatható az állapottábla (1.4.2.c ábra):



1.4.2.c ábra[1]

Az aszinkron DG tároló állapottáblája

A tároló működését leíró függvény felírását – az SR tárolónál alkalmazottakhoz hasonlóan- itt is Karnaugh tábla segítségével tesszük meg (1.4.2.d ábra):



1.4.2.d ábra[1]

Aszinkron DG tároló Karnaugh táblája

A táblában az összes lehetséges prímimplikánst bejelöltük. Ezekből az alábbi függvényalakot felírva elmondhatjuk, hogy az teljes mértékben lefedi a kimeneti ’1’-es

$$Y = D\,G + \overline{G}\,Y^{v}$$

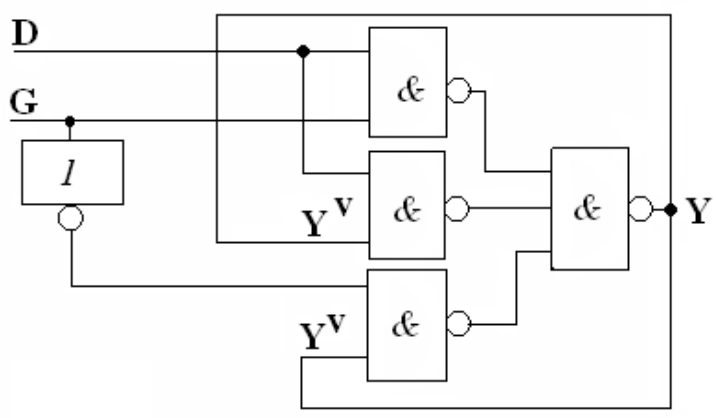
értékeket, vagyis –elméletben- eleget tesz a hálózat kimeneti viselkedésének leírására:

Azonban korábbi tanulmányainkból tudjuk, hogy áramköri realizáció esetében számolnunk kell a kapukésleltetési időkkel, vagyis az előfordulható statikus hazárddal.

Vizsgáljuk meg, hogy mit jelenthet ez a gyakorlatban: tételezzük fel, hogy a D,G, Y^v jelek rendjében ’111’-ben vagyunk, és G alacsony szintre állításával az ’101’ helyzetbe akarunk átmenni. Ha G előbb ’lefut’, minthogy a $\overline{G}$ ’felfutna’, beállhat az ’100’ állapot, és ennek hatására a hálózat ’0’-ban stabilizálódna. A helyes működés érekében kell tehát a statikus hazárdtól való mentesítés, vagyis a redundáns implikáns (DY^v) felírása. Az így kapott mintermes függvényalak:

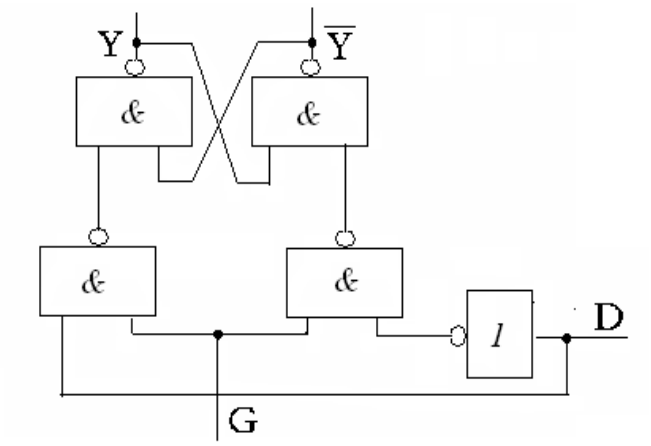
$$Y = D\,G + D\,Y^v + \overline{G}\,Y^v$$

A DG tároló lehetséges NAND-NAND realizációit mutatják a 1.4.2.e és 1.4.2.f ábrák:



1.4.2.e ábra[1]

DG tároló egy lehetséges NAND-NAND realizációja (statikus hazárdmentesített változat)



1.4.2.f ábra[1]

DG tároló NAND-NAND megvalósítása SR tároló sémájából

A DG tároló a digitális hálózatokban leggyakrabban mint klasszikus egybites memóriaelem fordul elő. A korrekt működés érdekében azonban a következő működési fázisok sorrendiségét biztosítani kell: a beírandó adatot a D adatbemeneten előkészítjük, majd a G beíró(kapuzó) bemenetet magas szintre emeljük. A tranziens lejátszódása után a beírt szint az Y kimeneten stabilizálódik. Ezután a D értékét még nem változtatva visszaejtjük a G bemenet szintjét. Az Y kimeneten a beírt érték ott marad. A G lefutása után D hiába változik, a kimenetre ez már hatástalan.

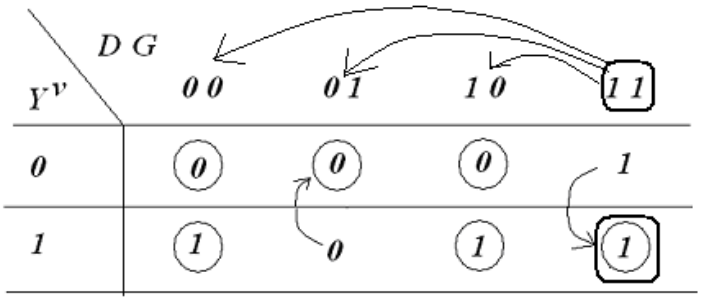
Amennyiben ez a sorrend nem biztosítható minden körülmény között, akkor több probléma is felmerülhet.

Vizsgáljuk meg a DG tároló állapottábláját(1.4.2.g ábra), és kövessük a működést az alább bemenő bitkombináció változás esetén: tételezzük fel, hogy a DGY^v bemenetek ’111’ kombinációja melletti stabil ’1’-es kimeneti állapotból most a DGY^v ’001’ változásának hatására egy másik, szintén stabil ’1’-es kimeneti értéket mutató helyzetbe kerülünk az állapottáblán. Ez azonban két ok miatt sem garantálható teljes mértékben: egyrészt két bemenetet tökéletesen egy időben nem tudunk változtatni, másrészt a két bemeneti változás hatása különböző késleltetésű utakon (áramutakon) érvényesül.

A valóságban tehát nem lehet kizárni, hogy vagy a DG-re nézve az 10, vagy a 01 bemeneti kombináció átmenetileg beáll. Így ha a 01 jelentkezik, azaz D lefutásának hatása előbb érvényesül, a kimeneten beállhat a ’0’ tranziens, amely végül, miután G is ’lefut’, az Y=0 kimenetet stabilizálja.

Ennek a lehetőségnek a kiküszöbölése csak akkor biztosítható, ha megtiltjuk a többszörös bemeneti váltásokat, azaz egyszerre csak egyetlen egy bemeneti jel értéke változhat meg. Ezzel egy olyan általános szabályt is kimondhatunk, mely szerint

visszacsatolt kombinációs hálózatok bemenetei közül egyszerre csak egyet szabad változtatni!



1.4.2.g ábra[1]

Többszörös bemeneti bitváltozás hatása DG tároló esetében ('111'→'001')

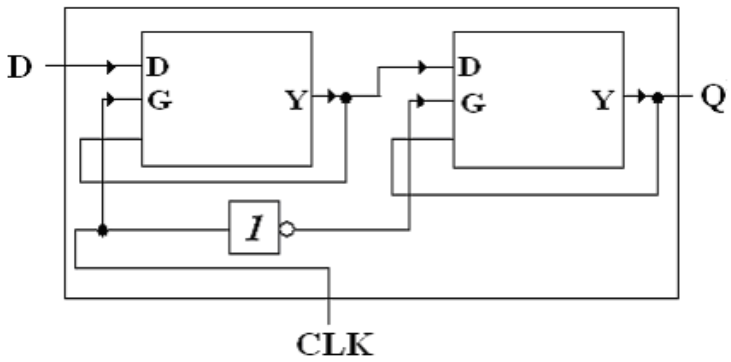
Az aszinkron DG tároló felépítéséből származó további hátránya, hogy a G=1 helyzetben a D-re adott változások kijutnak a kimenetre. *A G=1 helyzetben tehát a tároló a D-bemenet felől „átlátszó" (transzparens).* Ez sok esetben nemkívánatos és további problémák forrását jelentheti. A megoldás ilyenkor olyan tárolók alkalmazása, melyek a bemenet felől „nem átlátszók, azaz nem transzparensek. Erre az igényre nyújtanak megoldást az ún. Mester-Szolga (MS) kialakítású szinkron tárolók.

1.4.3 Mester-szolga (MS) tárolók

1.4.3.1 D-MS tároló

Az aszinkron DG tároló felhasználásával alakítsunk ki olyan tároló komplexumot, mellyel kiküszöbölhető a transzparencia jelensége egy tárolóban! Ennek érdekében a következők szerint járjunk el: kössünk sorba egymás után két aszinkron DG tárolót a 1.4.3.1.a ábra szerinti kialakításban, azaz az első tároló Y kimenetét kössük a második tárolónk D bemenetére, és ennek a második tárolónak az Y kimenetét nevezzük el Q kimenetnek. Az így kapott kétfokozatú tárolónk bemenete legyen tehát az első fokozat D adatbemenete, a kimenete pedig a második fokozat Q kimenete. Továbbá azt a bemenetet, amelyről az első tároló G bemenetét vezéreljük, és amelynek negált változója a másik tároló beírását vezérli, speciális funkcióval ruházzuk fel: órajel ('clock' vagy 'clk') lesz a neve, illetve funkciója. Ezt az órajelet szinkronizáló jelnek is nevezik: innen ered a „szinkron hálózat" elnevezés. Az ilyen szinkronizáló jelet nem tartalmazó áramköröket pedig aszinkron áramköröknek nevezik.

A működést analizálva beláthatjuk, hogy az órajel magas értékénél a D bemenet szintje beíródik az első DG tárolóba, de eközben a második tároló kimenete változatlan, hiszen a G bemenetére ’0’ jut. Az órajel lefutásakor az első fokozat ’átlátszatlanná’ válik, a kimenetének értéke azonban átíródik a második tároló kimenetére. A beírás egy órajelciklussal megtörtént, de úgy, hogy a teljes tároló ezalatt átlátszatlan maradt, hiszen egyik fokozata mindkét órajel helyzetben átlátszatlan volt. Az ilyen két ütemben beírható tárolókat nevezzük mester-szolga (master-slave) tárolóknak, mivel az első fokozatba beírt értéket a második szolgai módon bemásolja. A D-MS tároló működését tehát az órajel két fázisra bontja: az első a D bemenet mintavételezése és a mintavételezett érték tárolása, miközben a Q kimenet változatlan, őrzi az utolsóként beállt értéket. A második a Q kimenetre a mintavételezett érték rákapcsolása és tárolása, mi-közben a D bemenet változásai már hatástalanok maradnak.



1.4.3.1.a ábra[1]

D-MS tároló aszinkron DG tárolók és központi vezérlőjel(órajel) alkalmazásával

A D-MS tároló igazságtáblája a 1.4.3.1.b ábrán látható:

egyszerű igazságtábla

D	Q^{n+1}
0	0
1	1

1.4.3.1.b ábra

A D-MS tároló igazságtáblája

Az állapottáblában az órajel-vezérelt tárolóknál a következő kimeneti állapot jelölésére az (*n+1*) felső indexet szokás alkalmazni, mivel az aktuális kimeneti állapotot az *n*-edik órajel ciklus eredményének tekintjük, és így *n* felső indexet alkalmazunk a jelölésre. Azaz az aktuális kimenet jele Q^n, a következőé Q^{n+1} .

A 1.4.3.1.b ábrán látható, hogy a D-MS tároló következő kimeneti állapota nem függ az aktuális kimeneti állapottól, csakis a D bemenettől. Az összetett igazság-tábla tehát azonos az egyszerűvel.

A tároló –adott előírás szerinti- működtetéséhez szükségünk van egy ún. *vezérlési táblára*, amely segítséget nyújt az alkalmazott MS tárolónak a hálózat specifikáció szerinti viselkedésének biztosításához. A D-MS tároló vezérlési táblája az összetett igazságtábla alapján származtatható. A 1.4.3.1.c ábrán a vastag határvonalakkal feltüntetett táblázat baloldala mutatja a flip-flop kimenetének lehetséges változásait, a jobboldali oszlop pedig azt, hogy az adott változás végbemeneteléhez milyen logikai értéket kell a D bemenetre kapcsolni. Nyilván ez igen egyszerű, hiszen mindig a megkívánt új értékkel azonos értéket kell a D-re kapcsolnunk.

.

D	Q^n	Q^{n+1}	$Q^n \rightarrow Q^{n+1}$		D
0	0	0	0	0	0
0	1	0	0	1	1
1	0	1	1	0	0
1	1	1	1	1	1

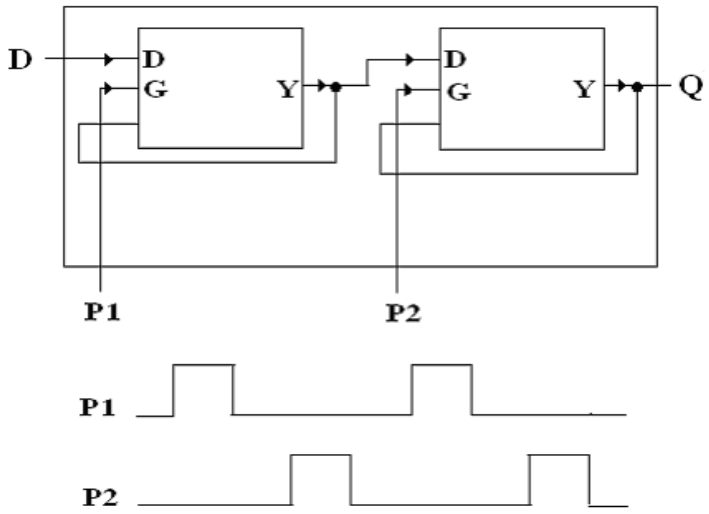
1.4.3.1.c ábra[1]

A D-MS tároló vezérlési táblájának származtatása az összetett igazságtáblából

1.4.3.2 Órajel vezérlések a MS tárolókban

Master-slave tárolókban a két fokozat működésének időbeli elválasztására gyakran kétféle vezérlőjel kialakítást is alkalmaznak.

Az egyik az ún. *kétfázisú, nem átlapolt órajel* használata. Erre látunk példát D-MS tárolóban a 1.4.3.2.a ábrán:

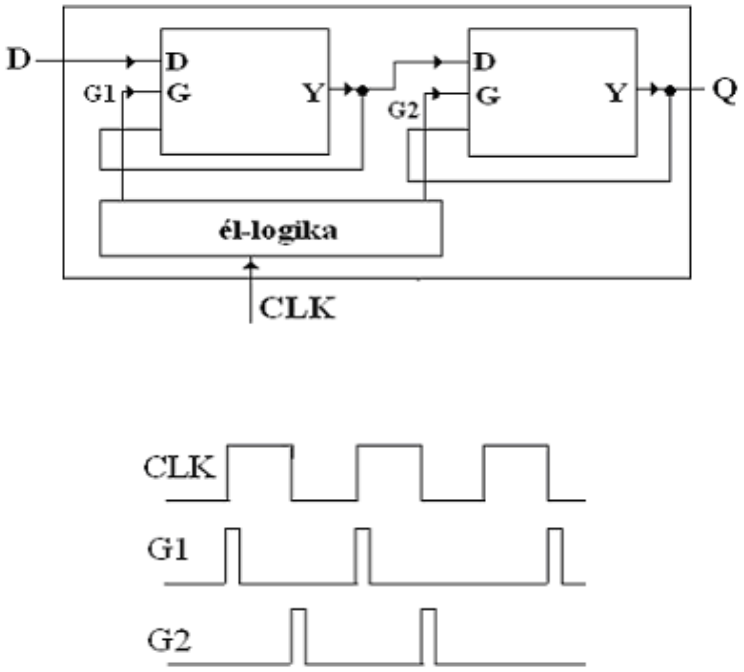


1.4.3.2.a ábra[1]

Kétfázisú órajel használata D-MS tárolóban

A két, P1 és P2 órajel-impulzus időben nem átlapoltan érkezik a mester és a szolga fokozatra, ezzel biztosítva a biztonságos elválasztást.

A másik megoldás az ún. *élvezérelt órajel* alkalmazása. Ennek kialakítását láthatjuk a 1.4.3.2.b ábrán, D-MS tároló esetében:



1.4.3.2.b ábra[1]

D-MS flip-flop élvezérelt órajellel

Az alkalmazás előnye, hogy maga a vezérlő órajel(CLK) egyfázisú, és egy speciális, néhány logikai kapuból álló kiegészítő áramkör alkalmazásával állítják elő az él-logikának megfelelő impulzusokat (G1 és G2). Az ábrán szereplő élvezérlés esetünkben azt jelenti, hogy az órajel(CLK) felfutó élére az egyik(mester), a lefutó élre a másik(szolga) tároló működik. Az ilyen él-logikával megvalósított MS tárolókat *lefutó élre működő tárolóknak* nevezik, és a gyártók ezt a specifikációban szereplő, a működési fázisokat megadó ún. funkciótáblában (function table) az órajelnél egy '↓' szimbólummal szokták jelölni. *Felfutó élre működő MS tárolók* esetében az élvezérlés éppen fordított, ilyenkor a funkciótáblában az órajelhez rendelt szimbólum is ennek megfelelően a '↑' jel (1.4.3.4.b ábra). Az élvezérelt tárolókat más néven „flip-flop"-oknak is nevezik.

A fentiekben bemutatott órajel ütemező megoldások természetesen más (JK,T) MS tárolók esetében is azonos kialakítással történnek.

1.4.3.3 JK-MS tároló

A JK-MS tároló egy olyan mester-szolga tároló, melynek egy J és egy K adatbemenete és egy Q adatkimenete van. A tároló működését leíró egyszerű és összetett igazságtáblát a 1.4.3.3.a ábrán láthatjuk:

egyszerű igazságtábla

J	K	Q^{n+1}
0	0	Q^n
0	1	0
1	0	1
1	1	$\overline{Q^n}$

összetett igazságtábla

J	K	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

1.4.3.3.a ábra[1]

A JK-MS tároló igazságtáblái

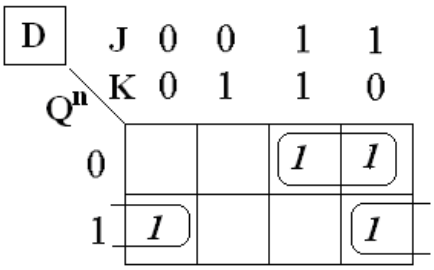
JK-MS tároló megvalósításához kézenfekvő megoldásnak tűnik a D-MS tároló felhasználása, hiszen annak vezérlési táblája (ami megfeleltethető az egyszerű igazságtáblájának) alapján a D bemenet vezérlése könnyen felírható: egyszerűen behelyettesítve a JK-MS összetett igazságtáblájában a következő állapotot meghatározó Q^{n+1} helyébe a D bemenet vezérlését (1.4.3.3.b ábra), a táblázatból kiolvashatjuk a szükséges hálózatot leíró algebrai függvényalakot (1.4.3.3.c ábra). Ehhez utolsó lépésként a szükséges Karnaugh táblát kitöltjük az igazságtábla alapján, és a kijelölt prímimplikánsokból felírt függvényalakból származtatható a D-MS tárolóval realizálható hálózat (1.4.3.3.d ábra).

Megjegyzés: a kitöltött Karnaugh táblát tanulmányozva, korábbi tanulmányaink alapján feltételezhetjük, hogy statikus hazárd veszélye állhat fent a kijelölt két prímimplikáns ($J\overline{Q^n}$, $\overline{K}Q^n$) átmeneti mintermjei között. A hazárdmentesítés azonban ebben az esetben nem szükséges, mivel szabályszerűen az órajelet csak akkor lehet felemelni, ha a D-t meghajtó hálózat tranziensei befejeződtek, és D már nyugalmi állapotba került.

J	K	Q^n	D
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

1.4.3.3.b ábra[1]

D-MS tároló vezérlése a JK-MS megvalósításához



1.4.3.3.c ábra[1]

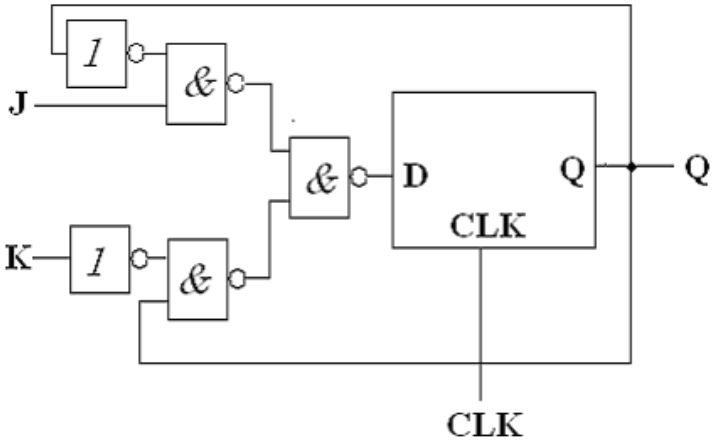
A D-MS tároló felhasználásával megvalósított JK-MS tároló Karnaugh táblája a kijelölt prímimplikánsokkal($J\overline{Q^n}$, $\overline{K}Q^n$)

A D bemenetre felírt algebrai alak:

$$D = J\overline{Q^n} + \overline{K}Q^n$$

illetve ennek NAND-NAND változata:

$$D = \overline{\overline{(J\overline{Q^n})} \cdot \overline{\overline{K}Q^n}}$$



1.4.3.3.d ábra[1]

JK-MS tároló NAND-NAND realizációja D-MS tároló felhasználásával

A JK-MS tároló vezérlési táblájának az összetett igazságtáblából származtatható alakja a 1.4.3.3.e ábrán látható. Az összetett igazságtáblában minden lehetséges változáshoz van egy olyan bemenet a kettő közül, amelynek szintje lehet ’0’ is és lehet ’1’ is, azaz közömbös. Ezek a „don’t care” bejegyzések a következő állapot kombinációt generáló kombinációs hálózat egyszerűsítéséhez vezetnek:

J	K	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$Q^n \rightarrow Q^{n+1}$		J	K
0	0	0	—
0	1	1	—
1	0	—	1
1	1	—	0

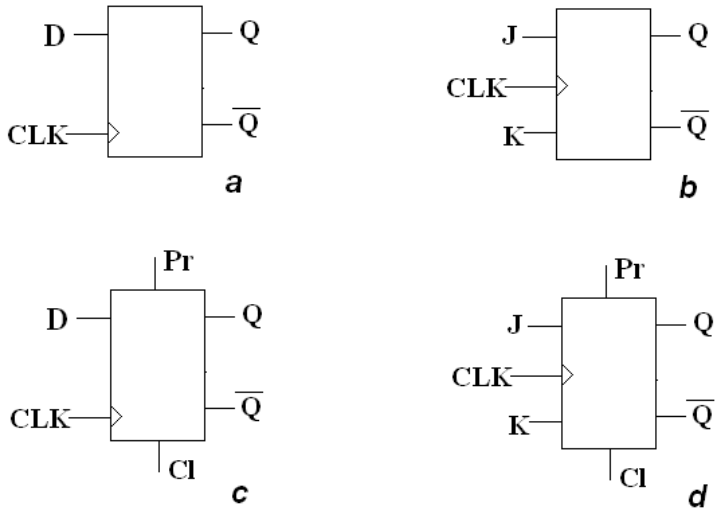
1.4.3.3.e ábra[1]

A JK-MS tároló vezérlési táblájának származtatása az összetett igazságtáblából

1.4.3.4 Mester-szolga tárolók segédbemenetei

A MS tárolók esetében a szükséges vagy előírt kezdeti állapot beállítását ún. segédbemenetekkel (’Pr’; ’Cl’ vagy ’Clr’) lehet elvégezni. Ezek legtöbbször a gyártók által eleve kialakításra kerülnek egy tárolóelemet tartalmazó integrált áramköri tokban, de ennek hiányában – egy, az adatbemenetekre kapcsolódó - kiegészítő áramkör segítségével is biztosítható a funkció (ennek megvalósításával a későbbiekben részletesen is foglalkozunk)(1.4.3.4.a ábra). A ’preset’ (’Pr’) bemenet a tárolót a funkcionális bemenetektől függetlenül logikai magas szintre, a ’clear’ (’Cl’ vagy ’Clr’) a funkcionális bemenetektől függetlenül logikai alacsony szintre állítja.

Megjegyzés: a segédbemenetek egyes flip-flopoknál az órajeltől függetlenül állítják be a kimenetet, másoknál az órajel valamelyik élének hatására (ez az adott tároló működését specifikáló funkciótáblából egyértelműen kiolvasható(1.4.3.4.b ábra)). Előbbieket aszinkron segédbemeneteknek, utóbbiakat szinkron segédbemeneteknek nevezzük.



1.4.3.4.a ábra[1]

D-MS és JK-MS tárolók szokásos szimbólumai segédbemenetekkel, és azok nélkül

INPUTS				OUTPUTS	
PRE	CLR	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H	H
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q^n	$\overline{Q}^n$

1.4.3.4.b ábra

A Texas Instruments által gyártott HC(T)7474 tokozású D-MS flip-flop funkciótáblázata

Felhasznált irodalom:

[1] Dr. Keresztes Péter: Digitális hálózatok (HEFOP elektronikus Jegyzet)