II. modul: Sorrendi hálózatok tervezése

2. lecke: Szinkron sorrendi hálózatok tervezése mintapéldákon keresztül I.: az állapotgráf és állapottábla használata feladatmegoldásokban; az ’1’-es súlyú állapotkódolás alkalmazása.

Cél:

A lecke célja, hogy a hallgató megismerje a szinkron sorrendi hálózatok szisztematikus (MEALY, MOORE) tervezési eljárásainak lépéseit, megismerkedjen a hozzájuk kapcsolódó új fogalmakkal. Egy egyszerű szimulációs program (DSCH) segítségével – a leckéhez kapcsolódó mintapéldák és házi feladatok megoldásán keresztül - gyakorlati tapasztalatokat szerezzen a kapuszintű realizáció elkészítésében és az elkészített elméleti feladatmegoldások áramköri tesztelésében.

Követelmények:

Ön akkor sajátította el megfelelően a tananyagot, ha

- ismeri a szinkron sorrendi hálózatok tervezési eljárásainak főbb lépéseit
- tervezőprogram segítségével áramköri realizációt tud építeni
- szimulátor segítségével képes értelmezni a különböző tervezési eljárásokkal megépített áramkörök működési sajátosságait

Időszükséglet:

A tananyag elsajátításához kb. 300 percre van szükség.

Kulcsfogalmak:

- állapotgráf, állapottábla, vezérlési tábla
- szekunder változó, kódolás
- kezdeti állapot: RESET logika
- ’1’-es súlyú kódolás

Tevékenység: tanulmányozza a leckében levezetett feladatmegoldásokat, és készítse el a szimulátor programban a hozzájuk tartozó realizációt, majd ellenőrizze és értelmezze a működésüket!

2.1 Szinkron sorrendi hálózatok tervezése mintapéldákon keresztül I.

Ebben a fejezetben mintapéldákon keresztül mutatjuk be a szinkron hálózatok tervezési eljárásait, megismerjük az egyes tervezési lépések sorrendjét, valamint a megtervezett hálózatok logikai kapuszintű megvalósítását.

2.1.1 Szinkron sorrendi hálózat tervezése: 1. mintafeladat

Példa:

adott egy kétbemenetű (X_1, X_2) és egy kimenettel (Z) rendelkező szinkron sorrendi hálózat. A hálózat az első X_1=X_2 bemeneti kombinációtól kezdve vizsgálja a bemeneteket, és a Z kimenetén jelzi, ha a két bemenet kétszer egymás után azonos logikai szintű. Ha ilyen kombináció sorozat lezajlott, a vizsgálatot újra kezdi. Tervezzük meg a hálózatot JK-MS tárolókkal!

2.1.1.1 A feladat Mealy-típusú hálózat modelljének megoldása

A feladat megoldása során egy szisztematikus módszert követünk, mely eljárás általánosságban használható a sorrendi hálózatok egy adott specifikáció alapján történő elvi és gyakorlati megvalósítására. Ennek általános lépései a következők:

1. *Állapot-átmeneti gráf felrajzolása.*
2. *Előzetes szimbolikus állapottábla felvétele.*
3. *Összevont szimbolikus állapottábla megszerkesztése.*
4. *Kódolt állapottábla elkészítése.*
5. *A specifikációra érvényes vezérlési tábla elkészítése (tárolók használata esetében).*
6. *A szekunder változó(k) és a kimenet(ek) függvényeinek Karnaugh tábla segítségével történő megadása.*
7. *Kezdeti állapotról történő gondoskodás.*
8. *Realizáció logikai kapukkal (és tárolókkal).*

Megjegyzés: a tervezés lépései egyes esetekben kiegészülnek egyéb lépésekkel is (pl. kritikus versenyhelyzet elemzése a szekunder változók kódolásánál, lényeges hazárdok vizsgálata stb.) A tervezés teljes, konkrét lépéssorozatát mindig az adott specifikáció illetve hálózattípus határozza meg. Ezekre látunk példákat majd a következő fejezetekben.)

Megoldás:

1. *Az állapot-átmeneti gráf felrajzolása* (2.1.1.1.a ábra)

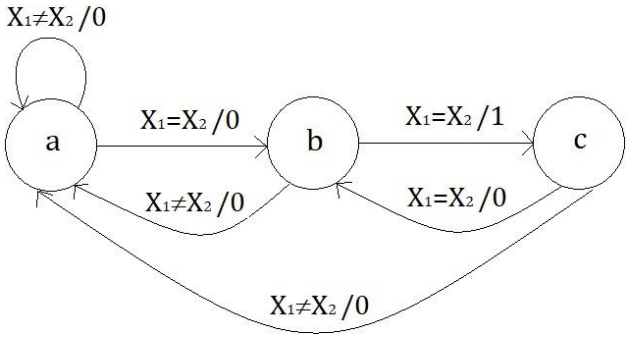
Első lépésként célszerű a hálózat működését egy ún. állapot-átmeneti gráfon, vagy röviden állapotgráfon szemléltetni. Az állapotgráf csomópontjai szimbolikus (legtöbbször betű szimbólumokkal jelölt) állapot kombinációk, röviden állapotok, élei a bemeneti kombinációk. *Mealy-típusú állapotgráfon minden élhez hozzárendeljük a kimeneti kombinációt is.* Az állapotgráf tehát megmutatja, hogy a hálózat adott aktuális állapotból egy adott bemeneti kombináció rákapcsolása után a következő órajelciklus hatására melyik következő állapotba kerül, és az adott aktuális állapothoz adott bemeneti kombináció rákapcsolásakor milyen kimeneti kombináció jelentkezik a kimeneteken. Ez utóbbiakat „/" jellel választjuk el a következő állapot szimbólumától. (*Megjegyzés: a Moore-típusú hálózatok állapotgráfja ettől abban különbözik, hogy a gráf csomópontjaihoz, azaz az állapotokhoz rendeljük hozzá a kimeneti kombinációkat.)*

Az állapotgráf elkészítésének első mozzanataként határozzunk meg, illetve vegyünk fel egy kezdőállapotot, amibe a hálózat bekapcsoláskor kerül. Nevezzük ezt el '*a*' állapotnak.

Az ezután jelentkező első órajelciklus hálózatra gyakorolt hatásának eredménye attól függ, milyen bemeneti bitkombináció érkezik. Ha $\boldsymbol{X_1}$ nem egyenlő $\boldsymbol{X_2}$-vel ($\boldsymbol{X_1 \neq X_2}$, azaz *01* vagy *10*), ez a vizsgálat szempontjából azt jelenti, hogy nem kell előkészíteni a kimenetet a szükséges 0→1 váltáshoz, tehát mindaddig, amíg *01* vagy *10* érkezik a bemenetre, maradunk a kezdeti állapotban(*a*), és a kimenet, $\boldsymbol{Z}$ alacsony szinten marad. Azt viszont, hogy megérkezik az első $\boldsymbol{X_1}$=$\boldsymbol{X_2}$ egyezőség (*00* vagy *11*), a hálózatnak meg kell jegyeznie, hogy a második együttállást jelezni tudja a kimeneten. Vagyis az *a* állapotból a *00* vagy az *11* bemeneti kombinációkra egy másik, *b* állapotba kerül a hálózat, és a *b* szimbólum jelentése, hogy megjött az első, számunkra kedvező bemeneti bitkombináció. A $\boldsymbol{Z}$ kimenet értéke az *a*-ból *b*-be tartó átmenet alatt természetesen *0* marad, hiszen ez még csak az első kedvező bemeneti kombináció.

Ha *b* állapotban tartózkodunk, és a következő órajelciklus a *01* vagy *10* bemeneti kombinációt fogadja, akkor a hálózatot vissza kell irányítani a kezdeti *a* állapotba, a $\boldsymbol{Z}$ kimenet pedig továbbra is alacsony logikai értéken van, a vizsgálat újból kezdődik elölről. Ezzel szemben, ha $\boldsymbol{X_1}$ és a $\boldsymbol{X_2}$ bemeneteken a *00* vagy az *11* bitkombináció érkezik, hálózatunk egy újabb, ezúttal *c* állapotba kerül, és a $\boldsymbol{Z}$=1 értékkel jelzi, hogy megjött a második együttállás. A *c* állapotból való elágazásnál, ha *01* vagy *10* érkezik a

bemeneteken, a kezdőállapotba(*a*) kell mennünk, hiszen újra az első együttállásra kell várni. Érkezhet azonban *00* vagy *11* is a következő órajel mintavételezéskor, és akkor máris a *b* állapotba kell mennünk. Természetesen mindkét esetben a kimenet alacsony szintre vált vissza.



2.1.1.1.a ábra

Az 1. mintafeladat Mealy-típusú állapot-átmeneti gráfja

2. *Az előzetes, szimbolikus állapottábla felvétele* (2.1.1.1.b ábra)

Az előzetes szimbolikus állapottáblában ugyanazokat az információkat rögzítjük, mint az állapot-gráfon. Ennek szerkezetét a tárolók, illetve flip-flopok tervezéséből már ismerjük. Ez azonban az állapotokat absztrakt formában tartalmazza, a konkrét állapot kombinációk megállapítását, azaz az állapotkódolást később végezzük el. A állapottáblában tehát bemeneti kombinációnként minden állapothoz megadjuk a következő állapot szimbólumát, és a „/” jellel elválasztva a megkívánt kimeneti kombinációt.

Megjegyzés: az állapotgráf jól szemlélteti grafikus ábrázolásmódon keresztül az adott hálózat mindenkori viselkedését, de a konkrét tervezési folyamathoz nem feltétlenül szükséges, el is hagyható. Amennyiben a specifikáció viszonylag egyszerű és könnyen értelmezhető kritériumokat tartalmaz, akkor elég a megoldás első lépéseként azonnal felírni az előzetes szimbolikus állapottáblát. Ha azonban konkrét áramköri realizáción (vagy szimulációs program segítségével) vizsgáljuk a megvalósított hálózatot, a működés egymást követő fázisai az állapotgráfon keresztül könnyedén leellenőrizhetőek.

aktuális állapot	következő állapot/kimenet	
	$X_1 \neq X_2$	$X_1 = X_2$
a	$a/0$	$b/0$
b	$a/0$	$c/1$
c	$a/0$	$b/0$

2.1.1.1.b ábra

Az 1. mintafeladat előzetes szimbolikus állapottáblája(Mealy-modell)

Ennél a feladatnál egyetlen gráf-él, vagy a tábla egyetlen oszlopa lényegében két bemeneti bitkombinációt képvisel: $\boldsymbol{X_1 \neq X_2}$ (*01;10*) és $\boldsymbol{X_1 = X_2}$(*00;11*). Természetesen megadható lett volna mindkettő állapotonként négy éllel, illetve négy oszloppal is, de célszerű kihasználni az ilyen és ehhez hasonló egyszerűsítési lehetőségeket.

Vegyük észre: a hálózat elágazásait tulajdonképpen egyetlen jel vezérelheti! Ez pedig analógiát mutat egy nevezetes kétváltozós függvénnyel, ami nem más, mint az EXNOR függvény, illetve ekvivalencia kapu. Vezessük be tehát az *E* logikai változót, amelyre így igaz, hogy:

$$E = X_1 X_2 + \overline{X_1}\,\overline{X_2}$$

3. *Összevont szimbolikus állapottábla megszerkesztése* (2.1.1.1.c ábra)

Az állapotgráf szerkesztésekor előfordulhat, hogy akkor is új állapotot veszünk fel, amikor pedig egy már meglévőt is felhasználhatnánk. Legtöbbször éleslátás vagy gyakorlat kérdése, hogy elsőre felismerjük-e a feltétlenül szükséges állapotokat. Ez azonban nem jelent problémát az adott feladat végső megoldását tekintve, hiszen az első, előzetes állapottábla állapotainak számát szisztematikus módszerekkel minimalizálni lehet. Ennek általános megfontolásaival egy későbbi fejezetben részletesen foglalkozunk, most azonban egy magától értetődő kritériumot már megfogalmazhatunk arra vonatkozóan, hogy mikor nem kell megkülönböztetni két állapotot az előzetes állapottáblán: *az előzetes állapottábla két állapotát nem kell*

megkülönböztetni, ezért azok összevonhatók, ha bemeneti kombinációnként megegyeznek a hozzájuk rendelt kimeneti kombinációk, és bemenő kombinációnként ugyanarra a következő állapotra vezetnek.

Keressünk ez alapján összevonható állapotokat az előzetes szimbolikus állapottáblánkban! Az állapottábla alapos vizsgálatából kiderül, hogy a fenti feltétel az *a* és a *c* állapotokra teljesül, hiszen a mindkettőből kiinduló bemeneti kombinációkhoz ugyanazok a kimeneti értékek tartoznak: *E=0*-nál *a*-ból ***Z****=0*, *E=1* esetben ugyancsak, és egyformák a ***Z*** értékek az *E=1* bemenetnél is. A következő állapotok az *E=0*-nál mindkettőre *a*, és *E=1*-re mindkettőre *b*. *Következtetés*: az *a* és *c* állapot megkülönböztetése felesleges, azokat össze lehet vonni. Legyen az új, összevont állapotunk jele *ac* ! Ez alapján elkészíthetjük az összevont szimbolikus állapottáblánkat, amely most már csak a feltétlenül szükséges állapotokat tartalmazza. Ebben az összevont állapotok kerülnek bejegyzésre mindazokon a helyeken, ahol az előzetes táblában az összevont állapotok valamelyike szerepelt.

Megjegyzés: amennyiben nincsenek a feltételrendszer szerint összevonható állapotok, úgy a következő, kódolt állapottáblát előállító lépéssorozat alapja az előzetes szimbolikus állapottábla.

aktuális állapot	**következő állapot/kimenet**	
	$\bar{E}$	E
ac	$ac/0$	$b/0$
b	$ac/0$	$ac/1$

2.1.1.1.c ábra

Az 1.mintafeladat összevont szimbolikus állapottáblája(Mealy-modell)

4. *A kódolt állapottábla elkészítése* (2.1.1.1.d ábra)

Az összevont szimbolikus tábla állapotait bináris kódokkal, azaz állapot kombinációkkal kell ábrázolni. Itt a választott kód hosszúsága egyértelműen meghatározza a visszacsatoló MS tárolók számát. Természetesen legtöbbször minimális számú MS tároló alkalmazására törekszünk, ezért a következő összefüggés alapján kódoljuk az állapotokat:

$$N_{MS} \geq \log_2 N_a$$

Itt N_{MS} az állapotkód hossza, azaz a szükséges MS tárolók száma, N_a pedig a kódolt összevont állapottáblán az állapotok száma. A kódolt állapottábla elrendezése tehát nagyon hasonló, mint az összevont (vagy előzetes) szimbolikus állapottábláé: egyszerűen be kell helyettesíteni az állapotok helyébe a hozzárendelt kódo(ka)t. Példánkban az összevont szimbolikus állapottáblában két állapotot kell megkülönböztetni. Ezekhez következő lépésként rendeljünk egy Q másodlagos változót, és a hozzárendeléskor mindegy, hogy melyik állapothoz rendeljük a Q=0, és melyikhez a Q=1 értéket. Esetünkben az *ac* állapothoz a Q=0, a *b* állapothoz a Q=1 bináris kódértéket rendeltük:

kódolt aktuális állapot, Q^n	**kódolt következő állapot/kimenet**	
	$\bar{E}$ Q^{n+1}/Z	E Q^{n+1}/Z
0	*0/0*	*1/0*
1	*0/0*	*0/1*

4.5.1.1.d ábra

Az 1. mintafeladat kódolt állapottáblája(Mealy-modell)

5. *A vezérlési tábla elkészítése* (2.1.1.1.f ábra)

A tervezési folyamatnak ezen a pontján – amennyiben nincs külön előírás az alkalmazandó tárolók típusáról - általában döntenünk kell, milyen flip-flopokkal valósítjuk meg a hálózatot. A D-MS tárolókkal való megvalósítás kevesebb tervezői munkával jár, hiszen minden flip-flopnak csak egyetlenegy bemenete van, ugyanakkor a kiadódó kombinációs hálózat általában bonyolultabb, mint JK-MS tárolók alkalmazásakor. Ez az előny illetve hátrány persze erősen feladatfüggő. Ezért javasolható mindkét flip-flop típus kipróbálása, és egy jól megválasztott költség-függvény szerinti összehasonlítás. A vezérlési táblák megalkotásához szükségünk van a flip-flopok saját vezérlési tábláira, nevezetesen arra, hogy adott kimeneti változáshoz milyen szinteket kell kapcsolnunk a bemenetekre.

Mivel a feladat specifikációja JK-MS tároló(k) alkalmazását írja elő, így ezúttal a JK-MS flip-flop saját vezérlési tábláját kell segítségül hívni. A vezérlés kódolt állapottáblából történő kiolvasásának folyamatára mutat példát a 2.1.1.1.e ábra:

kódolt aktuális állapot, Q^n	kódolt következő állapot/kimenet	
	$\bar{E}$ Q^{n+1}	E Q^{n+1}
0	*0/0*	*1/0*
1	*0/0*	*0/1*

2.1.1.1.e ábra

A vezérlés kiolvasásának folyamata: $Q^n \rightarrow Q^{n+1}$

A kiolvasást követően a 2.1.1.1.f. ábra mutatja a következő lépésként előállítandó táblákat. Mivel egyetlen szekunder változónk van (Q), ezért ennek megvalósításához egyetlen JK-MS tárolóra van szükségünk A baloldali, JK-MS saját vezérlési táblájából kiolvassuk azokat a szükséges J és K vezérléseket, melyek a kódolt állapottáblában meghatározott következő Q^{n+1} értékeket előállítják a tárolónk kimenetén. Az így felírt táblázat tartalmazza a most már a feladatra specifikált J és K bemeneti vezérléseket:

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

$\Rightarrow$

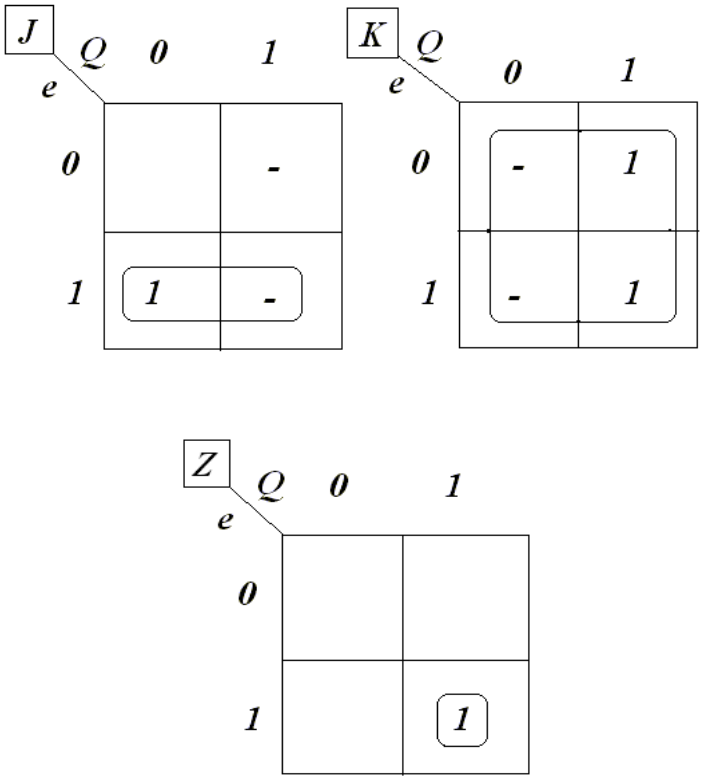
Q	$\bar{E}$ J K	E J K
0	*0* -	*1* -
1	- *1*	- *1*

2.1.1.1.f ábra

Az 1.mintafeladat JK flip-flopjának vezérlési táblája a kódolt állapottáblából és a tároló saját vezérlési táblájából származtatva(Mealy-modell)

6. *A szekunder változó(k) és a kimenet(ek) függvényeinek Karnaugh tábla segítségével történő megadás* (2.1.1.1.g ábra)

A vezérlési táblák megszerkesztése után hozzáfoghatunk a két kombinációs hálózat megtervezéséhez. Ehhez minden adat kiolvasható a táblázatokból. Nyilvánvaló, hogy az általánosabb Mealy-típusú realizációnál mind az f_y, mind az f_z hálózat bemeneteit a teljes hálózat bemenetei és a tárolók kimenetei alkotják, tehát a szükséges Karnaugh táblákat ennek alapján kell felvennünk. A kódolt állapottáblából már látható, hogy esetünkben igen egyszerű, kétváltozós Karnaugh táblákkal megadhatjuk a két kombinációs hálózat logikai függvényeit.



2.1.1.1.g ábra[1]

Az 1.mintafeladat Karnaugh táblái(Mealy-modell)

A táblázatokban a prímimplikánsok kijelölése után kialakult függvényalakok a következő algebrai eredményt adják:

$$J = E, \quad K = 1, \quad Z = Q\,E$$

7. *Kezdeti állapot beállítása*

A realizáció során - amellett, hogy a flip-flop szimbólumokkal és az ismert kapu-szimbólumokkal felrajzoljuk a hálózat struktúráját - gondoskodnunk kell a kezdeti (bekapcsolás utáni) állapot beállításáról is. Ha 'preset' és 'clear' bemenetekkel is rendelkező flip-flopokat választunk, akkor egyszerű dolgunk van: ha a kezdeti állapotban egy adott tároló kimenete '0', a 'clear' bemenetre adunk egy kezdeti állapotba állító impulzust, és a 'preset" bemenetet állandó 0-ba állítjuk. Ha pedig egy adott tárolót kezdeti állapotban '1'-be kell állítani, akkor a 'preset' kimenetre adunk impulzust, és a 'clear' bemenetre konstans '0'-t. A kezdeti állapot beállítását eredményező speciális bemenőjelet '*R*' (RESET) szimbólummal jelöljük. Példánkban a hálózat kezdeti állapotában (*a*) az alkalmazott JK-MS tárolónk kimenetén (a kódolt állapottábla szerint) alacsony szintet kell biztosítani, ezért az '*R*' kezdeti állapot beállító jelet a 'clear' bemenetre kell csatlakoztatni, és a nem használt 'preset' bemenetre konstans alacsony szintet kapcsolunk (2.1.1.1.h ábra).

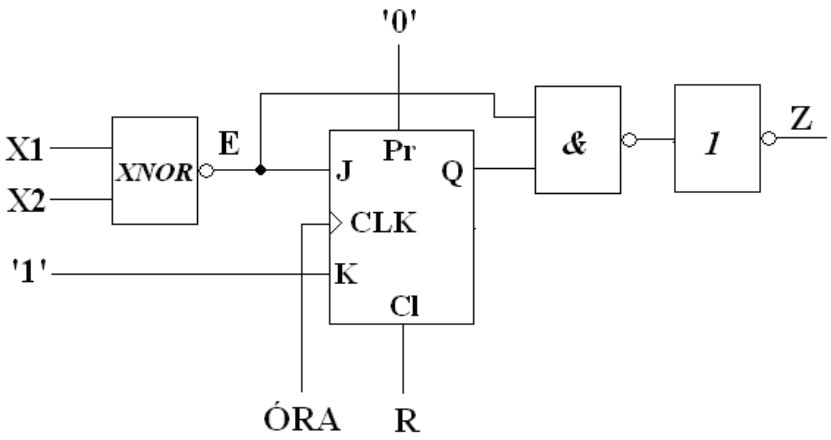
(*Megjegyzés: mindaddig, amíg a kezdeti beállítás más módszereit meg nem ismerjük, csak 'preset' és 'clear' bemenetekkel is rendelkező flip-flopokat használunk.*)

8. *A realizáció logikai kapukkal* (2.1.1.1.h ábra)

A kapott hálózat érdekessége, hogy a realizáció tárolójának kimenete nincs visszacsatolva a következő állapotot generáló hálózat bemenetére, holott a szinkron sorrendi hálózat általános modelljének bemutatásakor ennek lényegét hangsúlyoztuk. Fogadjuk el, hogy egy adott speciális funkció megvalósítása vezethet arra, hogy egyik vagy másik szekunder változó értékétől nem függenek egyik vagy másik flip-flop állapotváltozásai. A Mealy-modell szerint realizált hálózatunk sajátossága, hogy a flip-flop aktuális kimenetétől nem függ annak a következő állapota.

A felvázolt NAND-NAND realizáció alapját szolgáló algebrai függvényalakok:

$$J = E, \quad K = 1, \quad Z = \overline{\overline{QE}}$$

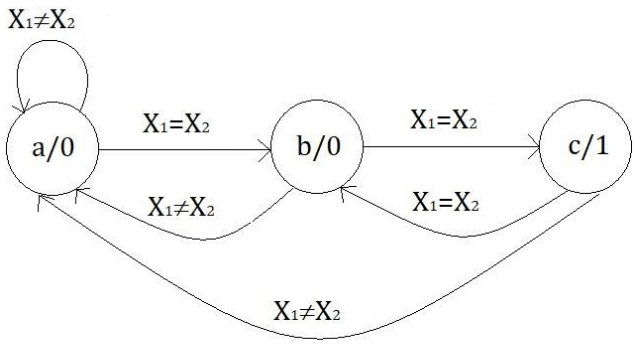


2.1.1.1.h ábra[1]

Az 1.mintafeladat Mealy-típusú NAND-NAND realizációja

2.1.1.2 A feladat Moore-típusú megoldása

A feladat Moore-típusú hálózattal történő megvalósítása során az előbbiekben megismert szisztematikus tervezési módszer – a lépéseket tekintve – sok hasonlóságot mutat a Mealy-modell szerint elvégzett tervezéssel, de lényegét tekintve mégis fontos különbséget mutat. Ezt a különbözőséget már az állapotgráfon az állapotok, valamint az állapot átmenetek jelölése is mutatja (2.1.1.2.a ábra): *Moore-modell esetén a kimeneti kombinációk csak az állapotok függvényei!*



2.1.1.2.a ábra

Az 1.mintafeladat Moore-típusú állapotgráfja

Ez már előrevetíti az állapottábla bejegyzéseinek Mealy-típusú vázlatához képesti különbözőségét (2.1.1.2.b ábra):

aktuális állapot/kimenet	**következő állapot**	
	$X_1 \neq X_2$	$X_1 = X_2$
$a/0$	a	b
$b/0$	a	c
$c/1$	a	b

2.1.1.2.b ábra

Az 1.mintafeladat Morre-modell alapján felállított előzetes szimbolikus állapottáblája

Az előzetes szimbolikus állapottábla alapján elvégezzük az állapot-összevonási lehetőségek vizsgálatát. A szabály hasonló: *két állapot összevonható, ha a hozzájuk tartozó kimeneti kombinációk és a következő állapotok bemeneti kombinációnként azonosak.* Ez alapján a vizsgálat eredménye most azt mutatja, hogy ilyen párok nincsenek, tehát a Moore-típusú hálózatot három állapottal kell megterveznünk. Az állapotok száma alapján legalább két másodlagos változót kell bevezetnünk (Q_1; Q_2) azok bináris kóddal történő megkülönböztetéséhez.

Az előzetes - és immár végleges - szimbolikus állapottábla bejegyzései alapján megszerkeszthetjük a kódolt állapottáblát (2.1.1.2.c ábra). Az egyes állapotokhoz szabadon hozzárendelhetjük a szekunder változók bármely lehetséges bitkombinációját, így erre többféle megoldás is adódhat. Mivel azonban esetünkben csak három állapotot kell kétbites kombinációkkal megkülönböztetni, bármelyik párosítást is alkalmazzuk, mindenképpen marad egy kihasználatlan, „szabad” kombináció. Ez azt jelenti, hogy ehhez a nem használt, „megmaradó” kódhoz tartozó sorban a kimenet, valamint a következő állapotokat előíró vezérlés értékét közömbös bejegyzésként regisztrálhatjuk.

Az egyes állapotok megkülönböztetésére alkalmazzuk a következő kódolást:

	Q_1	Q_2
a :	0	0
b :	0	1
c :	1	0

A párosítás eredményeként adódik, hogy a nem használt kód az *11* szekunder változó bitkombináció, amelynek kódolt állapottábla szerinti sorába kerülnek a fent említett „don't care" bejegyzések.

kódolt aktuális állapot/kimenet	**kódolt következő állapot**	
	$\bar{E}$	E
$Q_1^n\ Q_2^n/Z$	$Q_1^{n+1}\ Q_2^{n+1}$	$Q_1^{n+1}\ Q_2^{n+1}$
0 0/0	*0 0*	*0 1*
0 1/0	*0 0*	*1 0*
1 0/1	*0 0*	*0 1*
1 1/-	*- -*	*- -*

2.1.1.2.c ábra

Az 1.mintafeladat kódolt állapottáblája (Moore-modell)

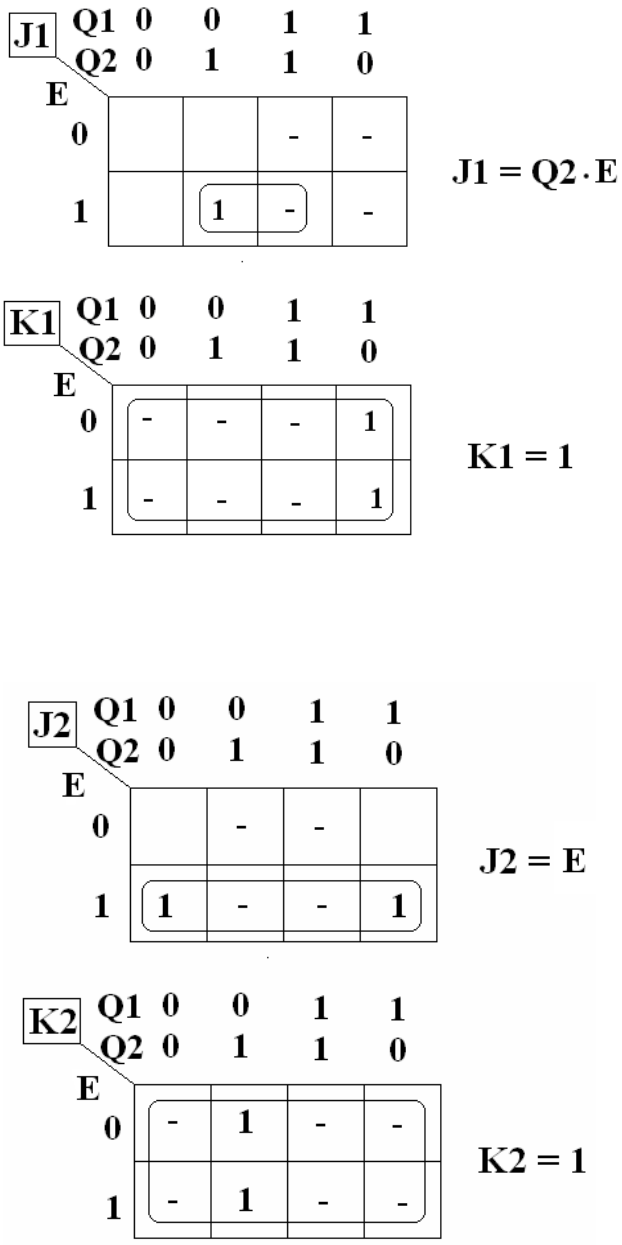
A kódolt állapottábla alapján – a JK-MS tároló saját vezérlési táblájának segítségével – megszerkeszthetjük a feladatra specifikált J és K vezérléseket, esetünkben két tároló bemeneteire (J_1K_1 és J_2K_2) definiálva (2.1.1.2.d ábra):

kódolt aktuális állapot/kimenet	**kódolt következő állapot**			
	$\bar{E}$		E	
$Q_1\ Q_2/Z$	J_1K_1	J_2K_2	J_1K_1	J_2K_2
0 0/0	*0 -*	*0 -*	*0 -*	*1 -*
0 1/0	*0 -*	*- 1*	*1 -*	*- 1*
1 0/1	*- 1*	*0 -*	*- 1*	*1 -*
1 1/-	*- -*	*- -*	*- -*	*- -*

2.1.1.2.d ábra

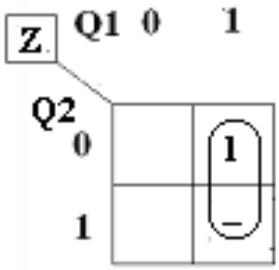
Az 1.mintafeladat JK flip-flopjainak vezérlési táblája a kódolt állapottáblából és a tároló saját vezérlési táblájából származtatva(Moore-modell)

A vezérlési táblázatból kitöltött Karnaugh táblák, az azokban felrajzolt függvény lefedések, valamint az ezekből származtatható algebrai függvényalakok a 2.1.1.2.e és 2.1.1.2.f ábrákon láthatóak. Az állapot kombinációk és a kimeneti kombinációk közötti egyértelmű függés szerint a ***Z*** kimenet csak a *c* állapotban, azaz a Q_1Q_2=*10* állapotkombináció fennállásakor lesz magas szinten.



2.1.1.2.e ábra[1]

Az 1.mintafeladat J és K bemeneteinek Karnaugh táblái, és a lefedések segítségével megadott algebrai alakok(Moore-modell)

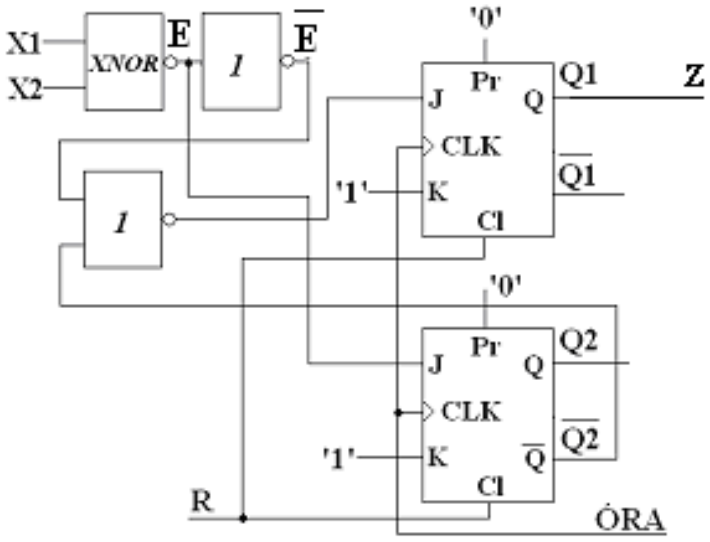


$$Z = Q_1$$

2.1.1.2.f ábra[1]

*Az 1.mintafeladat **Z** kimenetének Karnaugh táblája, és a lefedéssel megadott algebrai alak(Moore-modell)*

A Moore-modell egy lehetséges NAND-NAND realizációját láthatjuk a 2.1.1.2.g ábrán. A stabil kezdeti állapotban (a; $X_1 \neq X_2$) mindkét tároló kiementén alacsony szint kell, hogy megjelenjen, ezért az R beállítójelünket a törlő (CL) bemenetekre kötjük, a 'preset' pedig konstans alacsony szintet kap.



2.1.1.2.g ábra[1]

Az1.mintafeladat Moore-típusú NAND-NAND realizációja

2.1.2 Szinkron sorrendi hálózat tervezése: 2. mintafeladat

Példa:

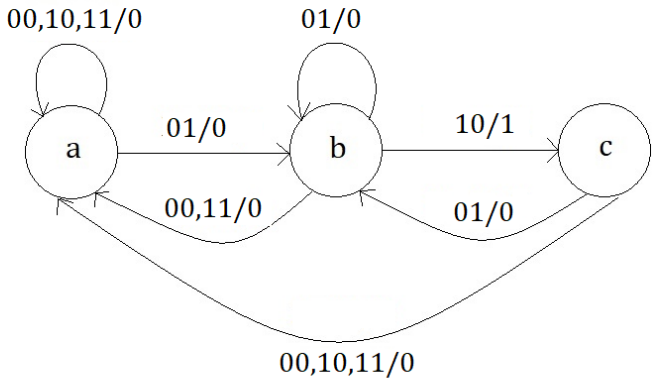
tervezzük meg a lehető legegyszerűbb változatban azt a 'preset' és 'clear' bemenetekkel is rendelkező élvezérelt D-MS tárolókkal felépített Moore-típusú szinkron sorrendi hálózatot, amelynek két bemenete (X_1, X_2) és egy kimenete (Z) van. A hálózat az órajel felfutása előtt fogadja a bemeneti kombinációkat. A hálózat Z kimenete akkor lesz '1', ha a bemenetre az '1 0' kombináció érkezik, de csak abban az esetben, ha az előző óraciklus által fogadott bemeneti bitkombináció '0 1' volt! A kezdeti állapothoz az X_1X_2 = '00' kombináció tartozik.

2.1.2.1 A feladat Mealy-modell szerinti megoldása

Megoldás:

1. *lépés: az állapotgráf felvétele.*

A 2.1.2.1.a ábrán láthatjuk a specifikáció alapján felrajzolt állapotgráfot:



2.1.2.1.a ábra[1]

A 2.mintafeladat Mealy-modell szerinti állapotgráfja

2. *lépés: az előzetes szimbolikus állapottábla felvétele.*

A felrajzolt állapotgráf alapján megszerkesztjük az előzetes szimbolikus állapottáblát a Mealy-típusú tervezési eljárás sajátosságait figyelembe véve (2.1.2.1.b ábra).

aktuális állapot	**következő állapot/kimenet**			
	X_1X_2			
	00	*01*	*10*	*11*
a	$a/0$	$b/0$	$a/0$	$a/0$
b	$a/0$	$b/0$	$c/1$	$a/0$
c	$a/0$	$b/0$	$a/0$	$a/0$

2.1.2.1.b ábra

A 2.mintafeladat előzetes szimbolikus állapottáblája(Mealy-modell)

3. *lépés: az összevont szimbolikus állapottábla megszerkesztése.*

Az állapot összevonás szabályai alapján az előzetes szimbolikus állapottáblán az *a* és *c* állapotok nem megkülönböztethetőek, azaz összevonhatóak, így csupán két állapot marad az összevont szimbolikus állapottáblán: az *ac* (osztály) és a *b* (2.1.2.1.c ábra).

aktuális állapot	**következő állapot/kimenet**			
	X_1X_2			
	00	*01*	*10*	*11*
ac	$ac/0$	$b/0$	$ac/0$	$ac/0$
b	$ac/0$	$b/0$	$ac/1$	$ac/0$

2.1.2.1.c ábra

A 2. mintafeladat összevont szimbolikus állapottáblája(Mealy-modell)

4. lépés: a kódolt állapottábla felvétele.

Az összevont szimbolikus állapottábla alapján csupán két állapotot kell megkülönböztetni, ezért ehhez elég egyetlen szekunder változó(Q) bevezetése. Az *ac* állapothoz rendeljük a *Q=0* kódot, a *b*-hez pedig a *Q=1*-et. Ez alapján a kódolt állapottábla(2.1.2.1.d ábra):

aktuális állapot, Q	**következő állapot/kimenet**			
	X_1X_2			
	00	*01*	*10*	*11*
0	*0/0*	*1/0*	*0/0*	*0/0*
1	*0/0*	*1/0*	*0/1*	*0/0*

2.1.2.1.d ábra

A 2.mintafeladat kódolt állapottáblázata(Mealy-modell)

5. lépés: a vezérlési tábla elkészítése.

A feladat specifikációja D-MS flip-flopok alkalmazását írja elő, ez alapján tehát a vezérlési táblát az egyetlen szekunder változót reprezentáló D-MS tárolónkra kell adaptálni. A tároló saját bemeneti vezérlésének táblája alapján a 2.1.2.1.e ábrán látható a kódolt állapottáblából származtatott vezérlési tábla

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

$\Rightarrow$

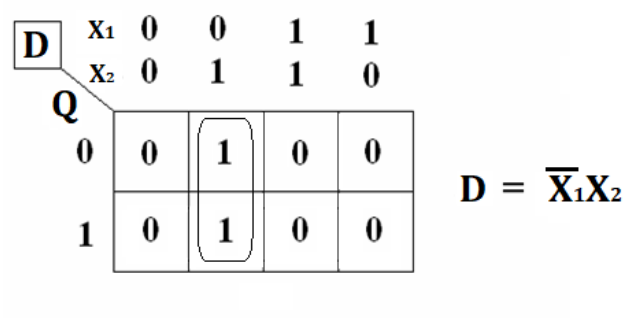
aktuális állapot, Q	D/Z			
	X_1X_2			
	00	*01*	*10*	*11*
0	*0/0*	*1/0*	*0/0*	*0/0*
1	*0/0*	*1/0*	*0/1*	*0/0*

2.1.2.1.e ábra

A 2.mintafeladat vezérlési táblája(Mealy-modell)

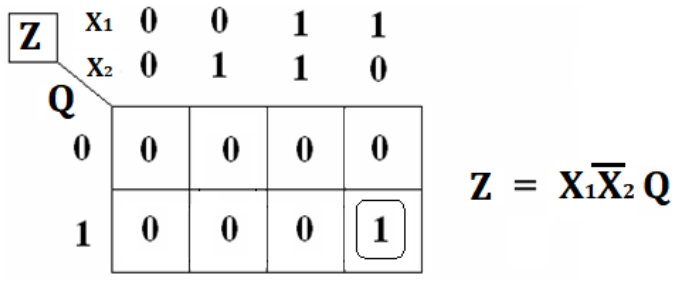
6. *lépés: a szekunder változó(D-MS adatkimenet) és a kimenet függvényeinek Karnaugh tábla segítségével történő megadása.*

Ebben a lépésben felrajzoljuk és kitöltjük a vezérlési tábla lapján a specifikációhoz illeszkedő Karnaugh táblákat(2.1.2.1.f és 2.1.2.1.g ábra):



2.1.2.1.f ábra[1]

*A 2.mintafeladat D-MS flip-flop **D** adatbemenetének Karnaugh táblája, és a táblázatból kiírt algebrai alakja(Mealy-modell)*



2.1.2.1.g ábra[1]

*A 2.mintafeladat **Z** kimenetének Karnaugh táblája, és a táblázatból kiírt algebrai alakja(Mealy-modell)*

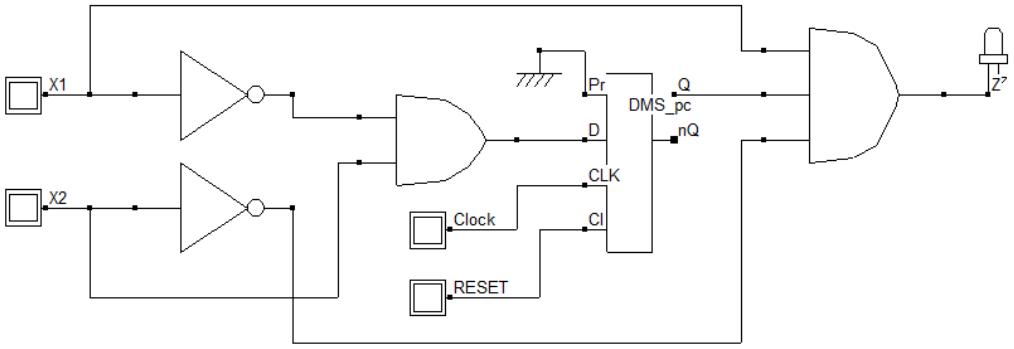
7. *A kezdeti állapot beállításának ellenőrzése*

A stabil kezdeti *a* állapothoz tartozó $\boldsymbol{X_1X_2}$ *= '0 0'* kombináció mellett a ***Z*** kimeneten alacsony logikai szintnek kell megjelenni, ami a függvény szerint teljesül is. A kódolt állapottábla ide vonatkozó bejegyzése szerint a D-MS tárolónk kimenetén is ugyanezt az

értéket kell biztosítanunk, ezért a 2.1.2.1.h ábrán látható realizációban az R('RESET') kezdőérték beállító jelünket a tároló törlő ('Cl') bemenetére kell kötnünk, a 'preset' segédbemenet ('Pr') pedig konstans alacsony szintet('ground') kap.

8. Az áramköri realizáció elkészítése

A feladat logikai kapuszintű realizációja a DSCH gyakorló programban elkészített sematikus áramköri rajzon (2.1.2.1.h ábra) látható:

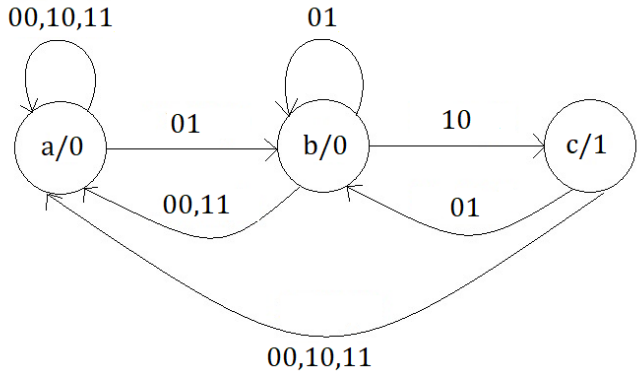


2.1.2.1.h ábra

A 2.mintafeladat Mealy-típusú áramköri realizációja a DSCH program segítségével

2.1.2.2 A feladat Moore-modell szerinti megoldása

A 2.mintafeladat Moore-típusú hálózattal történő megvalósításának első lépéseként rajzoljuk fel a specifikáció alapján az állapotgráfot (2.1.2.2.a ábra.)!



2.1.2.2.a ábra[1]

A 2.mintafeladat Moore-modell szerinti állapotgráfja

Második lépésként szerkesszük meg a hozzá tartozó előzetes szimbolikus állapottáblát(2.1.2.2.b ábra)!

aktuális állapot/kimenet	**következő állapot**			
	X_1X_2			
	00	*01*	*10*	*11*
a/0	*a*	*b*	*a*	*a*
b/0	*a*	*b*	*c*	*a*
c/1	*a*	*b*	*a*	*a*

2.1.2.2.b ábra

A 2.mintafeladat előzetes szimbolikus állapottáblája(Moore-modell)

Következő lépésként vizsgáljuk meg, hogy az állapot összevonás szabályai alapján tudjuk-e az állapotok számát csökkenteni a táblán! Az elemzés ugyan első ránézésre azt mutatja, hogy a kimenet szempontjából az *a* és *b* állapotok összevonhatóak lennének, de

sajnos a második feltétel nem teljesül, hiszen nem minden bemenő bitkombinációnként egyeznek meg a következő állapotok: $\boldsymbol{X_1X_2}$ = *'1 0'* esetében *a* aktuális állapotban maradunk stabilan, *b* aktuális állapotból viszont a *c* állapot felé indulunk. Mindez azt jelenti, hogy a kódolt állapottábla elkészítésének alapja ezúttal az előzetes szimbolikus állapottáblánk.

Mivel három állapotot kell megkülönböztetnünk, így két szekunder változó bevezetése szükséges a bináris kódoláshoz, mely legyen a következő:

	Q_1	Q_2
a :	0	0
b :	0	1
c :	1	0

Ebből pedig a 2.1.2.2.c ábrán látható kódolt állapottáblát készíthetjük el:

kódolt aktuális állapot/kimenet $Q_1^nQ_2^n/Z$	**kódolt következő állapot** X_1X_2			
	00	*01*	*10*	*11*
	$Q_1^{n+1}Q_2^{n+1}$	$Q_1^{n+1}Q_2^{n+1}$	$Q_1^{n+1}Q_2^{n+1}$	$Q_1^{n+1}Q_2^{n+1}$
0 0/0	*0 0*	*0 1*	*0 0*	*0 0*
0 1/0	*0 0*	*0 1*	*1 0*	*0 0*
1 0/1	*0 0*	*0 1*	*0 0*	*0 0*
1 1/-	*- -*	*- -*	*- -*	*- -*

2.1.2.2.c ábra

Az 2.mintafeladat kódolt állapottáblája (Moore-modell)

A $\boldsymbol{Q_1Q_2}$ = *'1 1'* kódkombinációt nem használjuk fel, így a táblázatban a hozzá tartozó bejegyzésekhez a 'don't care' ('-') jelölés kerül.

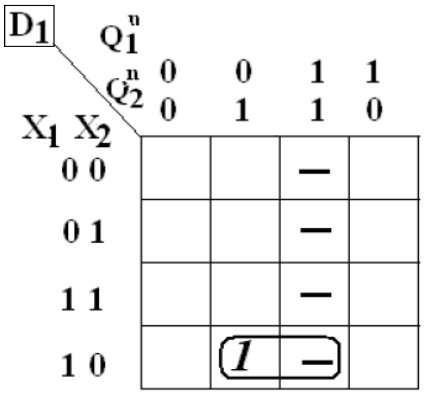
Mivel D-MS tároló felhasználásával kell megoldani a feladatot, ezért erre specifikáljuk a vezérlési táblát. Korábbi tanulmányainkból tudjuk, hogy a D-MS flip-flop alkalmazása esetén a vezérlési táblába a D bemenet vezérlési celláiba csak egyszerűen be kell másolni a kódolt állapottáblában megadott 'következő állapot'-kódot, így a 2.1.2.2.d ábrán látható táblázatot kapjuk. Magától értetődő, hogy a két szekunder változó két D-MS flip-flop (D_1 ; D_2) használatát feltételezi:

kódolt aktuális állapot/kimenet $Q_1^n\ Q_2^n/Z$	X_1X_2 00	01	10	11
	D_1D_2	D_1D_2	D_1D_2	D_1D_2
0 0/0	0 0	0 1	0 0	0 0
0 1/0	0 0	0 1	1 0	0 0
1 0/1	0 0	0 1	0 0	0 0
1 1/-	- -	- -	- -	- -

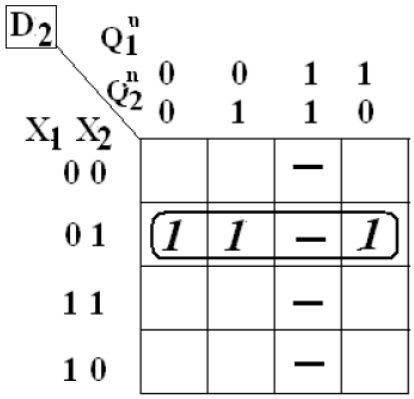
2.1.2.2.d ábra

A 2.mintafeladat vezérlési táblája(Moore-modell)

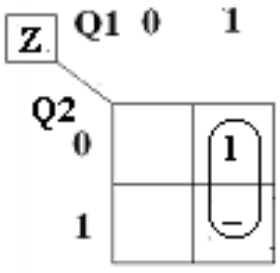
A D-MS tárolók és a kimenet függvényének felírásához szerkesszük meg a szükséges Karnaugh táblákat (2.1.2.2.e ábra):



$$D1 = Q_2^n \, X1 \, \overline{X2}$$



$$D2 = \overline{X1} \, X2$$

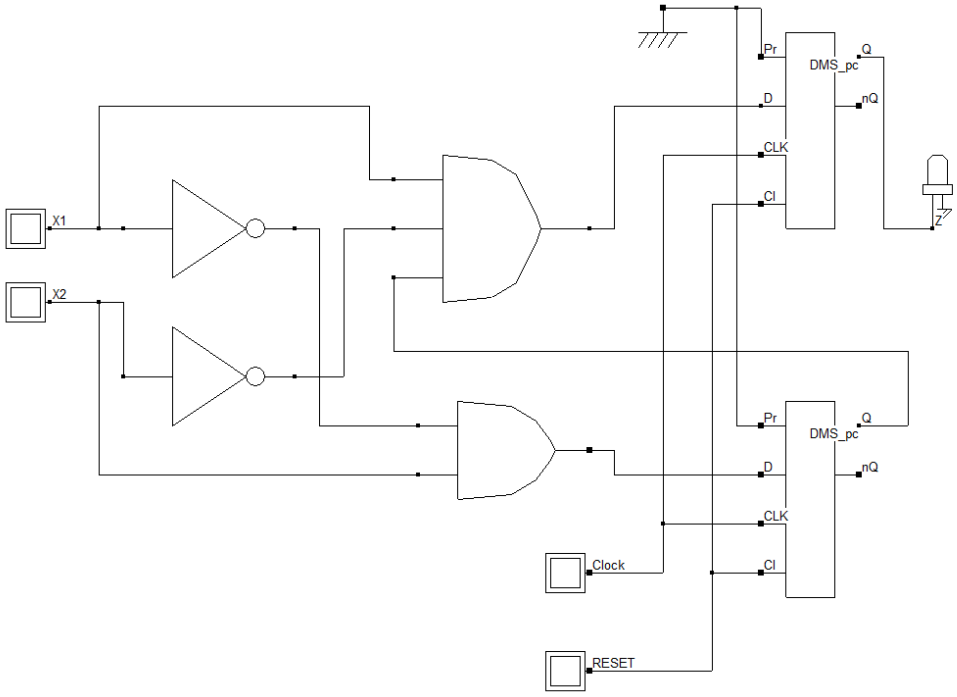


$$Z = Q_1$$

2.1.2.2.e ábra[1]

*A 2.mintafeladat D-MS flip-flop **D** bemeneteinek, és a **Z** kimenetnek a Karnaugh tábla alapján kiírt algebrai függvényalakja(Moore-modell)*

A Moore-tipusú kapurealizáció a 2.1.2.2.f ábrán látható:



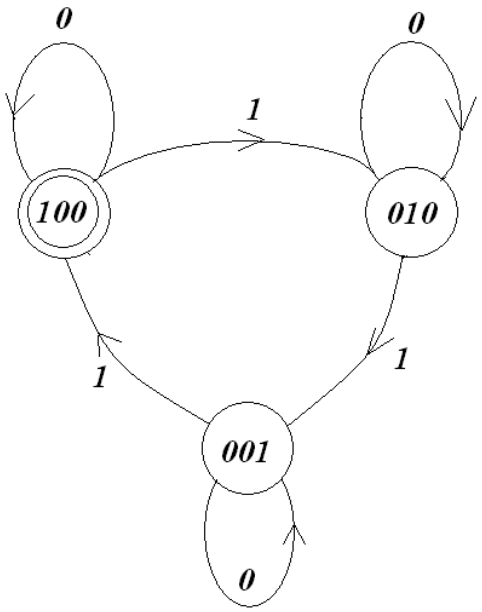
2.1.2.2.f ábra

A 2.mintafeladat Moore-típusú áramköri realizációja a DSCH program segítségével

2.1.3 Szinkron sorrendi hálózat tervezése: 3.mintafeladat

Példa:

tervezzük meg szinkron 'preset' és 'clear' bemenetekkel is rendelkező, élvezérelt D-MS tárolókkal a 2.1.3.a ábrán látható állapotgráf szerinti állapot-kimenetű szinkron sorrendi hálózatot a lehető legegyszerűbb változatban! A kezdeti állapotot a gráfon dupla kör jelzi.



2.1.3.a ábra[1]

A 3.mintafeladat állapotgráfja

2.1.3.1 A feladat megoldása Karnaugh táblás egyszerűsítés segítségével

A feladat specifikációjában egy új meghatározással találkozunk: az *állapot-kimenetű hálózat* fogalmával.

Állapot-kimenetű hálózat:

ha megnézzük a feladathoz tartozó állapotgráfot, akkor megállapíthatjuk, hogy a korábbi jelölésrendszerek (Mealy, Moore) alapján a kimenet értékére vonatkozó (külön) feljegyzést nem látunk. A gráfon most csak a bemenet (egy bit), illetve az egyes állapotokhoz rendelt hárombites kódok értékei vannak feltüntetve. Ilyenkor a kimenet értéke(i) maga az állapot, azaz maga az állapotot reprezentáló bináris kódszó. Ez egyben azt is jelenti, hogy az adott hálózatnak annyi kimenete van, ahány bit határozza meg az egyes állapotok kódját. Az ilyen struktúrájú sorrendi hálózatokat állapot-kimenetű hálózatoknak nevezzük.

Mivel a specifikációban a megadott állapotgráfon már a kódolt állapotokkal jellemzett hálózat egyes (adott pillanatbeli) működési fázisait láthatjuk, ezért kézenfekvő, hogy a feladat megoldását a kódolt állapottábla felvételével kezdjük (2.1.3.1.a ábra):

kódolt aktuális állapot,			kódolt következő állapot					
			X = 0			X = 1		
Q_1^n	Q_2^n	Q_3^n	Q_1^{n+1}	Q_2^{n+1}	Q_3^{n+1}	Q_1^{n+1}	Q_2^{n+1}	Q_3^{n+1}
1	0	0	1	0	0	0	1	0
0	1	0	0	1	0	0	0	1
0	0	1	0	0	1	1	0	0
0	0	0	-	-	-	-	-	-
0	1	1	-	-	-	-	-	-
1	0	1	-	-	-	-	-	-
1	1	0	-	-	-	-	-	-
1	1	1	-	-	-	-	-	-

2.1.3.1.a ábra

A 3.mintafeladat kódolt állapottáblája

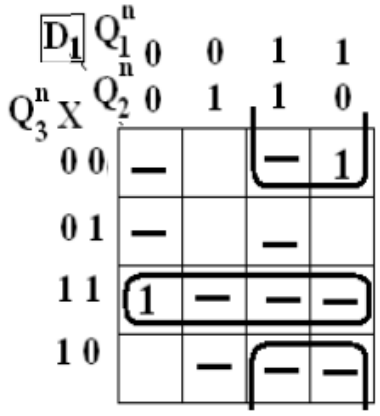
D-MS tárolók használata esetén a szekunder változókhoz rendelt flip-flopok vezérlési táblája a 2.1.3.1.b ábra szerint adódik:

kódolt aktuális állapot,			kódolt következő állapot					
			X = 0			X = 1		
Q_1^n	Q_2^n	Q_3^n	D_1	D_2	D_3	D_1	D_2	D_3
1	0	0	1	0	0	0	1	0
0	1	0	0	1	0	0	0	1
0	0	1	0	0	1	1	0	0
0	0	0	-	-	-	-	-	-
0	1	1	-	-	-	-	-	-
1	0	1	-	-	-	-	-	-
1	1	0	-	-	-	-	-	-
1	1	1	-	-	-	-	-	-

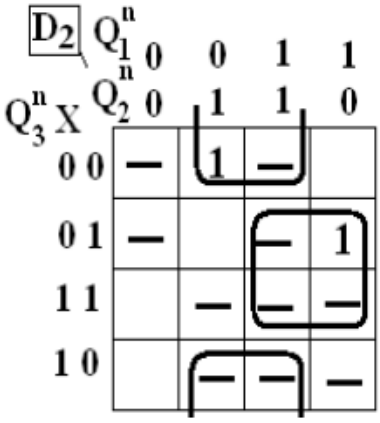
2.1.3.1.b ábra

A 3.mintafeladat vezérlési táblája

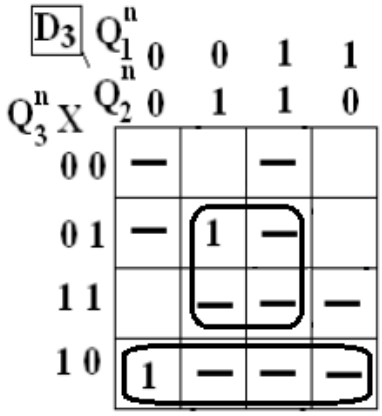
A vezérlési tábla adatai alapján megszerkesztett és kitöltött Karnaugh táblákon az egyszerűsítéseket elvégezve adódnak a D adatbemenetek logikai függvényei (2.1.3.1.c ábra):



$$D_1 = Q_1^n \overline{X} + Q_3^n X$$



$$D_2 = Q_2^n \overline{X} + Q_1^n X$$



$$D_3 = Q_2^n X + Q_3^n \overline{X}$$

2.1.3.1.c ábra[1]

*A 3.mintafeladat D-MS flip-flop **D** bemeneteinek Karnaugh táblái, és a táblázatokból kiírt algebrai függvényalakok*

2.1.3.2 Egy másik megoldás: az ’1’-es súlyozású kódolás kihasználása

A feladat állapotgráfját tanulmányozva szembeötlő, hogy a három állapotot ezúttal három szekunder változóval különböztettük meg, holott elég lenne kettő is. Ugyanakkor azt is felfedezhetjük, hogy minden egyes kódszóban csak egyetlen ’1’-es szerepel. *Az olyan kódolást, melyben csak egyetlen bit helyén szerepel ’1’-es érték (a többi ’0’), 1-es súlyú kódolásnak nevezzük (’bit per state’).*

Megjegyzés: szinkron hálózatok VLSI (Very Large Scale Integration= nagy integráltságú elemek) megvalósításakor gyakran igen gyorsan célravezető egy olyan állapotkód, amikor minden egyes szimbolikus állapothoz egy D–MS flip-flopot rendelünk. Ilyenkor ez a flip-flop ’aktív’, vagyis ’1’-es kimeneti értékű. Ahhoz azonban, hogy az állapotokat meg is különböztessük a hozzájuk rendelt kódszavakkal, éppen olyan hosszúságú (bitszámú) kódot kell alkalmazni, ahány állapotunk van. Ez biztosítja az 1-es súlyú kódolás helyes alkalmazását is.

A különböző állapotokhoz tetszőleges helyiérték szerint adhatjuk meg az egyetlen egy ’1’-esünk pozícióját, így többféle kódkiosztás is – a specifikáció által elvárt - helyes működési megoldáshoz vezet.

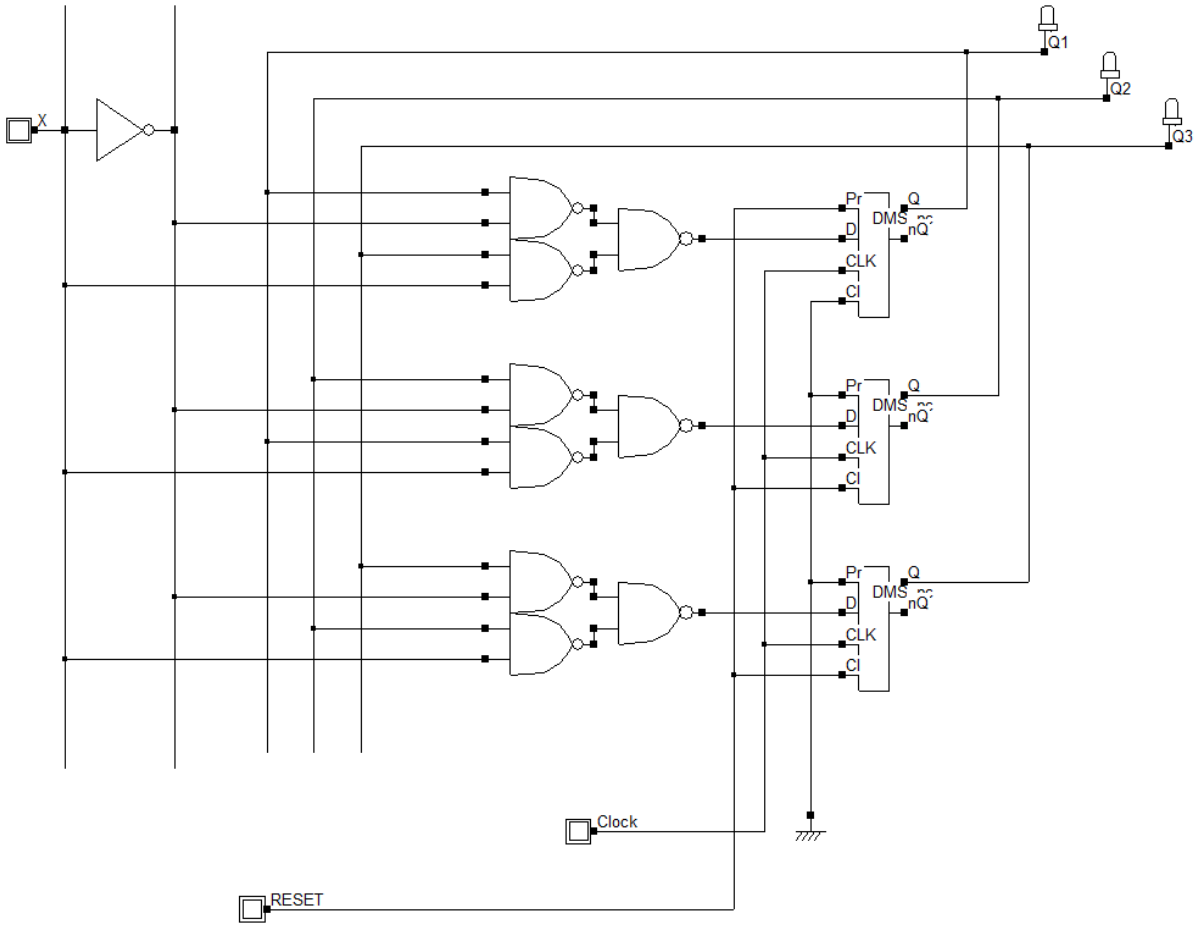
A D bemenetek vezérlésének megvalósítása ilyenkor rendkívül egyszerű, és az állapotgráf alapján elvégezhető: minden egyes D bemenetre akkor és csak akkor kell ’1’-et kapcsolni, amikor az általa reprezentált tároló kimenetnek ’0’-ról ’1’-re kell változnia, azaz amikor az adott flip-flop által reprezentált állapotnak be kell állnia. Ez pedig az állapotgráfból egyértelműen kiolvasható. Az így kapott algebrai függvényalak megegyezik a Karnaugh táblák alapján meghatározottakkal, azaz

$$D_1 = Q_1^n \overline{X} + Q_3^n X$$

$$D_2 = Q_2^n \overline{X} + Q_1^n X$$

$$D_3 = Q_2^n X + Q_3^n \overline{X}$$

Ne feledkezzünk meg a kezdeti állapot helyes beállításáról sem! Példánkban ez azt jelenti, hogy a kezdeti állapot ($D_1 D_2 D_3$ = ’100’) beállítását végző ’R’ jelünket ezúttal a D_1 tárolónk ’preset’ (Pr) segédbemenetére, a D_2 és D_3 flip-flop esetében pedig a ’clear’ (Cl) segédbemenetre kell kötni (2.1.3.2.a ábra):



2.1.3.2.a ábra

A 3.mintafeladat NAND-NAND architektúrájú áramköri realizációja a DSCH program segítségével

Felhasznált irodalom:

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