II. modul: Sorrendi hálózatok tervezése

3. lecke: Szinkron sorrendi hálózatok tervezése mintapéldákon keresztül II.: számlálókkal felépített szinkron sorrendi hálózatok tervezése

Cél:

A lecke célja, hogy a hallgató megismerje a szinkron számlálókkal felépített szinkron szekvenciális hálózatok tulajdonságait, az alkalmazott speciális célarchitektúrát, valamint a tervezés fő lépéseit, továbbá betekintést nyerjen az összetett digitális egységek speciális osztályába tartozó időzítő-vezérlők célorientált alkalmazásába.

Követelmények:

Ön akkor sajátította el megfelelően a tananyagot, ha

- ismeri a szinkron számláló felépítését és működését
- képes megtervezni egy adott specifikáció szerint egy szinkron számláló bázisú időzítő-vezérlő egységet
- szimulátor segítségével meg tudja határozni a hazárdmentes vezérlés szekvenciális folyamatát

Időszükséglet:

A tananyag elsajátításához kb. 150 percre van szükség.

Kulcsfogalmak:

- szinkron számláló, mod-m
- RESET, LOAD, ENABLE
- kettős ugrás
- időzítő-vezérlő egység
- hazárdmentes vezérlés

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

3.1 Szinkron sorrendi hálózatok tervezése mintapéldákon keresztül II.

3.1.1 Szinkron számláló alkalmazása a szinkron sorrendi hálózatokban

Mielőtt az első mintafeladatot ismertetnénk, tekintsük át a szinkron számlálók felépítését és működésük néhány fontos ismérvét!

A számlálók olyan összetett funkciójú digitális egységek, amelyek az eredeti, hagyományos számlálási funkción túl digitális egységek időzítő-vezérlő áramköreiben is felhasználásra kerülnek, mint rögzített funkciójú időzítők.

A számlálókat két csoportra osztjuk: szinkron és aszinkron számlálókra. Mindkét típusú számlálót a *'modulo'* értékkel jellemzünk. A *'modulo-m'* (m-modulusú) számláló számlálási tartománya a természetes számok *'m'*-mel való osztása után kapott lehetséges maradékok tartományán halad át. (*0, 1, 2, . . . m-1*).

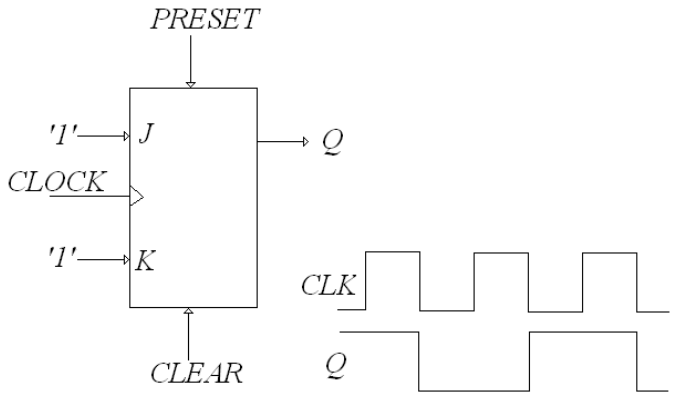
Egy alapvető fogalmat kell még bevezetnünk: a *carry-logika olyan kombinációs hálózat, amely a kimenetén '1'-et ad, ha a számláló kimenetein az (m-1) kódja jelenik meg.*

Megjegyzés: a számlálókkal, mint összetett digitális alapegységekkel egy későbbi fejezetben még foglalkozunk, de most, a szinkron sorrendi hálózatok ismertetése kapcsán csak a szinkron számláló felépítését tárgyaljuk meg részletesebben.

3.1.1.1 A JK-MS flip-flop, mint a számlálók alapeleme

Az aszinkron 'preset' és 'clear' bemenetekkel ellátott JK-MS tárolót korábbi tanulmányainkból már jól ismerjük. Az élvezérelt flip-flop funkciót szinkron számlálókban használjuk ki, míg a kettes-osztó funkciót az aszinkronnak nevezett számlálókban (számláncok) alkalmazzuk. A 3.1.1.1.a. ábra mutatja a kettes-osztót, idő-diagrammal. Belátható, hogy amennyiben a 'mester' tároló beírása az órajel felfutó, a 'szolga' beírása a lefutó élre történik, az órajel frekvenciája megfeleződik, azaz az így kapcsolt JK flip-flop az órajel frekvenciát 2-vel osztja. Megjegyezzük, hogy az aszinkron

számláncokban az órajel logikai szerepet kap, ezért olyan tárolót kell választani, amelyben a 'CLEAR' bemenet közvetlenül és azonnal hat a kimenetre az órajel közreműködése nélkül.



3.1.1.1.a ábra

A JK MS flip-flop kettes-osztó funkciója

A szóban forgó flip-flop számlálókban történő alkalmazása esetén még kiegészül egy ún. engedélyező(E) bemeneti jellel, melynek szerepe, hogy csak akkor engedi a J és K bemeneten megjelenő logikai érték órajelciklus által történő mintavételezését, ha magas szinten van. Ellenkező esetben (alacsony szint) a tároló kimenetén őrzi az előző ciklus végén állandósult logikai értéket.

3.1.1.2 A szinkron számláló modellje

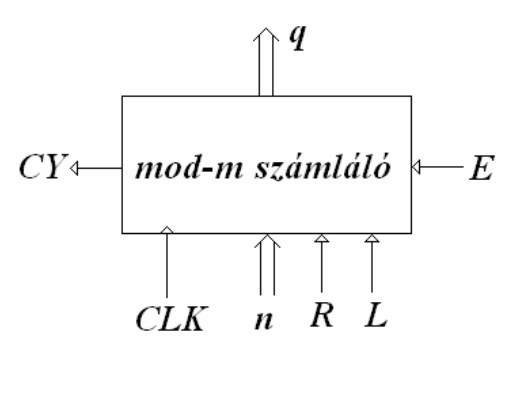
A szinkron számlálókat felépítő JK-MS vagy D-MS flip-flopok órajele azonos. A visszacsatoló hálózat a számláló állapot átmeneti gráfja alapján tervezhető meg azokkal a módszerekkel, amelyeket a szinkron hálózatok tervezésének tárgyalásakor elsajátítottunk. Hangsúlyozandó, hogy számlálók esetén az állapotkód adott. A betölthető(*L*), engedélyezhető(*E*), törölhető(*R*) szinkron *'modulo-m'* számláló vezérlőjelei és funkciói a következők:

- az *R* (RESET) magas szintje a rákövetkező órajel lefutására nullázza a számlálót. Az *R* jelnek a többi vezérlőjellel összehasonlítva abszolút prioritása van!

- az *L* (LOAD) jel hatására - amennyiben nincs *R* - a számláló tartalma a következő órajel lefutó élére az *n* adatbemenet(ek)re kapcsolt bitvektor lesz.

- az *E* (ENABLE) jel - amennyiben az *R* és az *L* bemenet is logikai alacsony szinten van - engedélyezi, hogy a következő órajel lefutó élére a számláló tartalma ’1’-el növekedjék (inkrementálás).

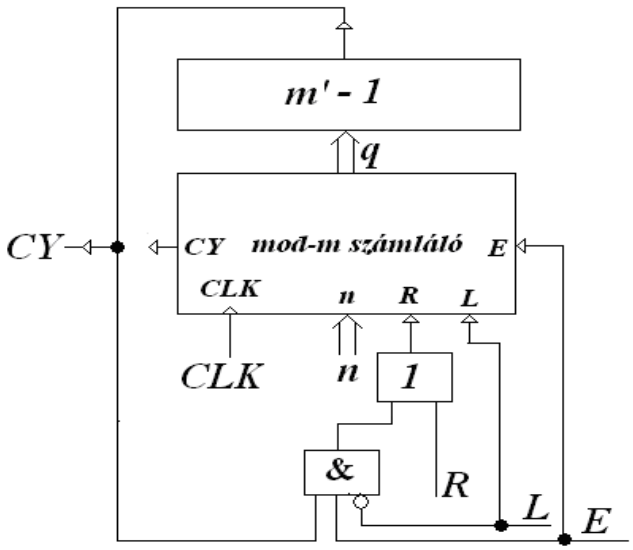
A 3.1.1.2.a. ábrán látható a szinkron számláló általánosan használt szimbóluma:



3.1.1.2.a. ábra

Szinkron számláló általános szimbóluma

Ha egy *m*-modulusú szinkron számlálót át kívánunk alakítani *m’ < m* modulusúvá, akkor új ’carry’ (*CY*) hálózatot kell kialakítani. Az új *CY* detektálja az új *m’ -1* értéket, és a soron következő órajel a számlálót a RESET bemenet segítségével törli. A vezérlő jelek közötti prioritási viszony csak akkor örökíthető át az átalakított számlálóra, ha beiktatjuk az $\bar{L}$ és *R* bemenetű ÉS kaput. Az *L* jel *E* feletti prioritásának ugyanis akkor is érvényesülnie kell, ha a *CY* = 1! (3.1.1.2.b ábra)

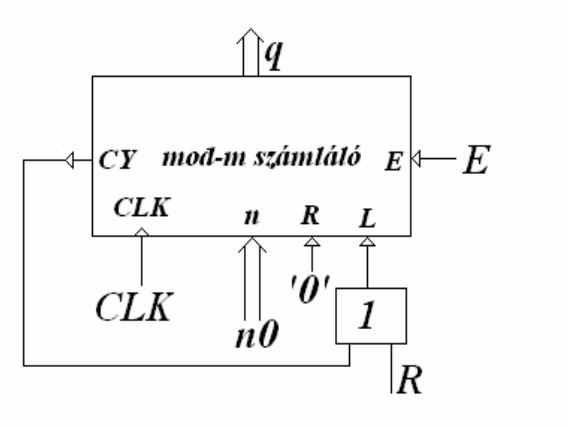


3.1.1.2.b ábra

Egy m-modulusú szinkron számláló átalakítása m' < m modulusú szinkron számlálóvá

Megjegyzés: az átalakított m' < m modulusú számlálónk 'n' bitvektor bemenetén nem íródhat be m'-1-nél nagyobb érték: ezt vagy feltételként szabjuk a felhasználónak, vagy gondoskodni kell egy kiegészítő, ún. szűrőáramkörről (kombinációs hálózat), mely megakadályozza az m'-1-nél nagyobb értékek megjelenését az 'n' adatbemeneteken!

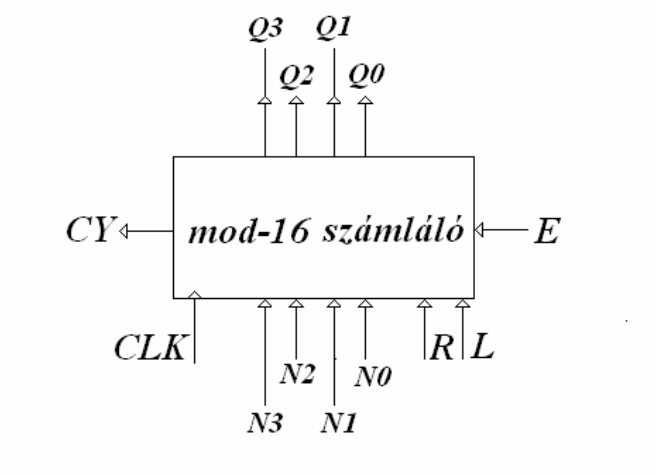
Amennyiben nullától különböző ('n0'), de a modulusnál kisebb kezdőértékkel szeretnénk ciklikusan indítani a számlálást, akkor az eredeti számláló *L* jelének felemelésével, a *CY* =1 feltétellel írjuk az adatbemeneteken keresztül a számlálóba ciklikusan az új kezdőértéket. Az új *R* bemenet ugyancsak az *L* bemenetre hat. Az eredeti *R* bemenetet nem használjuk, azt konstans logikai '0' értékre kötjük (3.1.1.2.c ábra).



3.1.1.2.c ábra

Nullától különböző kezdeti érték beállítása szinkron számlálón

A különböző modulusú számlálók közül a leggyakrabban használt *mod-16*-os számláló szimbólumát a 3.1.1.2.d ábrán látjuk:



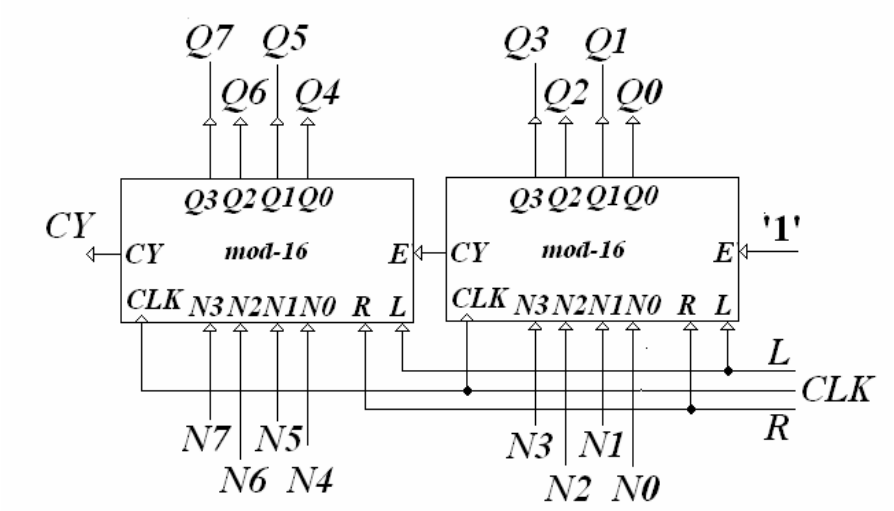
3.1.1.2.d ábra

4-bites mod-16-os szinkron számláló

Megjegyzés: a 3.1.1.2.d ábrán látható alapkiépítés mellet még több funkciót is megvalósítanak egy egységben. Például a felfelé('up') történő számlálás mellett lefelé('down') való számlálást is (UP-DOWN COUNTERS).

Ha a bemutatott mod-16-os számlálót 15-ig ('1111') felszámoltattuk, akkor a *CY* (CARRY) kimeneten magas szint jelenik meg. A következő órajel lefutó élére a számláló ismét '0'-ra áll ('0000'), és a *CY* kimenet is alacsony szintre kerül. Ez a *CY* arra alkalmas,

hogy egy magasabb helyiértékre helyezett számláló *E* bemenetét magas szintre állítsa, így lehetővé téve nagyobb modulusú számlálók kialakítását (kaszkádosítás: 3.1.1.2.e ábra).



3.1.1.2.e ábra

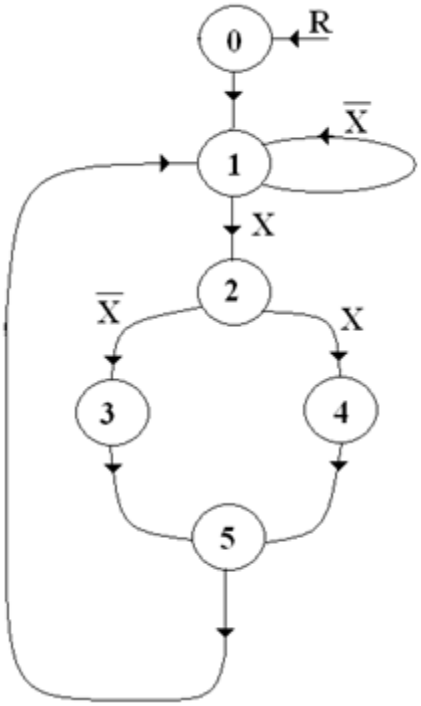
Mod-256-os számláló kialakítása mod-16 modulokból.

A törölhető, engedélyezhető, betölthető szinkron számlálókat szinkron sorrendi hálózatok állapot-regisztereként alkalmazhatjuk. Ez azt jelenti, hogy a hálózat állapotait a számláló kimenetei kódolják. Ha egy állapotgráf lehetséges átmeneteit számba vesszük, belátható, hogy a fenti számláló maradéktalanul képes azokat megvalósítani. Ha egy állapot következő állapota önmaga, akkor ezt a számlálóval úgy realizáljuk, hogy valamennyi vezérlőjelét ’0’ szinten hagyjuk. Ha az állapot átmenet a bináris kód inkrementálása, akkor egyedül az *E* (enable) jelet emeljük fel. Ha az átmenet egy nem önmagára és nem az inkrementált kódú következő állapothoz vezet, akkor az *L*(load) jel felemelésével betöltjük a számlálóba a következő állapot kódját. Az *R* jel funkciója akkor használható ki a legegyszerűbben, ha a szinkron hálózat kezdő állapotához a ’0’ bináris kódját rendeljük. Példánkból látni fogjuk, hogy a számlálók vezérlő bemeneteire multiplexereket csatlakoztatunk, és ezeket programozott logikai hálózatként alkalmazzuk.
Nézzük most meg az első mintapéldán keresztül, hogy melyek a számlálóval kialakított szinkron sorrendi hálózat tervezésének fő lépései!

3.1.2 Szinkron sorrendi hálózat tervezése szinkron számlálóval: 1. mintafeladat

Példa:

valósítsuk meg törölhető (R), betölthető (L) és engedélyezhető (E) számlálóval és multiplexerekkel az alábbi kódolt állapotgráf szerinti szinkron sorrendi hálózatot!(3.1.2.a ábra)



3.1.2.a ábra

Az 1.mintafeladat szinkron számlálóval kialakított sorrendi hálózatának állapotgráfja

A 3.1.2.a ábrán látható az állapot-kimenetű szinkron sorrendi hálózat kódolt állapot-gráfja.

Első lépésként szerkesszünk meg az állapotgráf alapján két táblázatot: az egyik táblázat a *lépések('E')*, azaz az inkrementálások logikai feltételeit, a másik az *ugrások('L')* logikai feltételeit és következő állapotait tartalmazza (3.1.2.b ábra).

LÉPÉS (E=1)		UGRÁS (L=1)		
akt. áll.	feltétel	akt.áll.	feltétel	köv.áll.
0	'1'	2	X	4
1	X	3	'1'	5
2	$\overline{X}$	5	'1'	1
4	'1'			

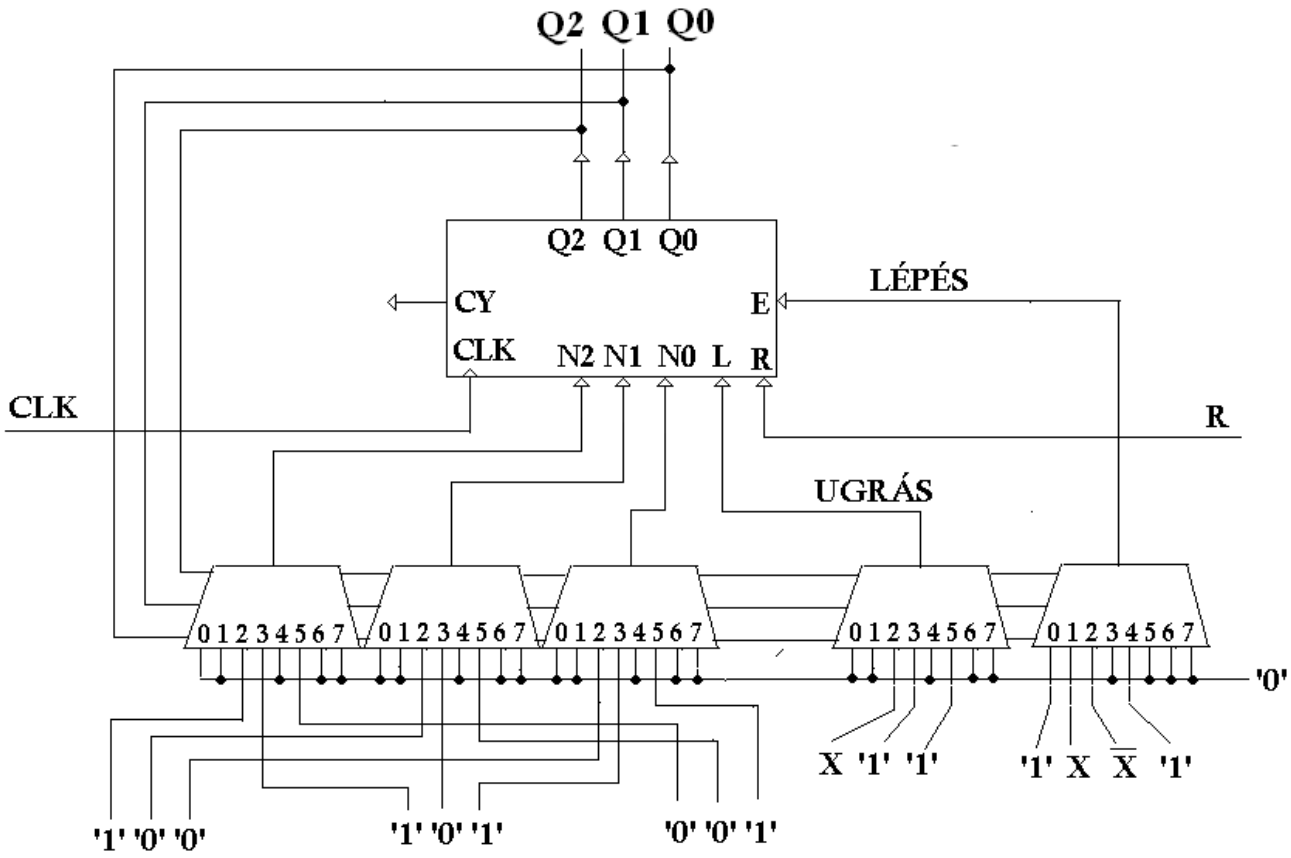
3.1.2.b ábra

Az 1.mintafeladat 'lépés' és 'ugrás' táblázatai

A feladat megoldásának következő lépése a multiplexerek programozása:

- a léptető (*E*) vezérlőbemenetre csatlakozó multiplexert a „LÉPÉS" táblázatban meghatározott feltételek („feltétel" oszlop) szerint kell programozni,
- az ugrást vezérlő bemenetre kapcsolódó multiplexert az „UGRÁS" táblázatban felvett feltételek („feltétel" oszlop) szerint kell programozni,
- az 'n' bitvektor bemenetekre csatlakozó multiplexerek adatbemeneteit pedig szintén az „UGRÁS" táblázatban lefektetett, következő állapot („köv.áll." oszlop) kódjának megfelelő bináris értékekkel kell felprogramozni.

A 3.1.2.c ábrán látható, speciálisan erre a feladatra felrajzolt célarchitektúra mutatja, hogyan kell az aktuális állapot kódja által megcímzett multiplexer bemenetekre a táblázatok szerinti feltételeket, illetve a betöltendő következő állapotkódokat rákapcsolni.



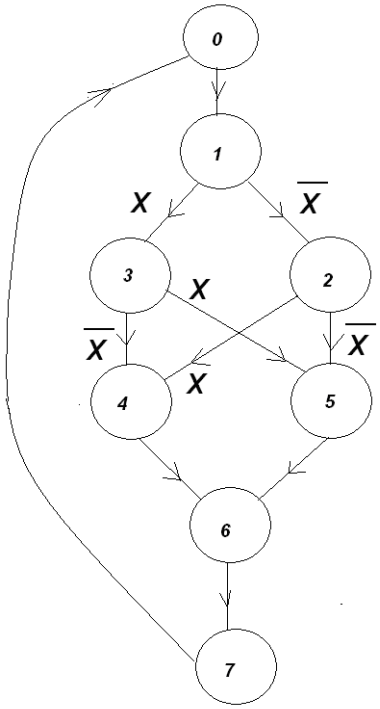
3.1.2.c ábra

Az 1.mintafeladat realizációja specifikált célarchitektúrával, mod-8-as számlálóval és mpx8-1 multiplexerekkel

3.1.3 Szinkron sorrendi hálózatok tervezése szinkron számlálóval: 2.mintafeladat

Példa:

valósítsuk meg törölhető (R), betölthető (L) és engedélyezhető (E) számlálóval és multiplexerekkel az alábbi kódolt állapotgráf szerinti szinkron sorrendi hálózatot!(3.1.3.a ábra)



3.1.3.a ábra

Az 2.mintafeladat állapotgráfja

A feladat megoldását az 1. mintapéldában ismertetettek szerint kezdjük a „LÉPÉS” és „UGRÁS” táblázatok elkészítésével (3.1.3.b ábra):

lépések		*ugrások*		
akt. áll	*feltétel*	*akt. állapot*	*feltétel*	*köv. állapot*
0	*'1'*	*1*	*X*	*3*
1	$\overline{X}$	*2*	$\overline{X}$	*5*
3	$\overline{X}$	*2*	*X*	*4*
5	*'1'*	*3*	*X*	*5*
6	*'1'*	*4*	*'1'*	*6*
7	*'1'*			

3.1.3.b ábra

A 2.mintafeladat állapotgráfja alapján felvett 'lépések' és 'ugrások' táblázatai

Az elkészített táblázatokat tanulmányozva egy érdekes jelenségre figyelhetünk fel: az „ugrások” táblázatában azt látjuk, hogy a számlálónk ’2’-es aktuális állapotában (*q=2*) $\overline{X}$ feltétel megléte esetén következő állapotként az 5-ös számlálóállásba kell „ugratnunk” a számlálónkat, míg *X* esetén a 4-es számlálóállásba. Ez azt jelenti, hogy az *X* feltétel, mint logikai változó akár ponált, akár negált értéket vesz fel, a ’2’-es számlálóállás (*q=2*) esetén a következő órajelciklus végén mindenképpen ugranunk kell: vagyis az *L* bemenetre csatlakozó multiplexer 2-es lábára konstans magas szintet kell kapcsolni. Azt, hogy ugyanakkor az *X* feltétel függvényében melyik új számértéket olvastassuk be az ’*n*’ bitvektor bemenetére csatlakozó multiplexereken keresztül - esetünkben a 5-öt (’*101*’) vagy a 4-et (’*100*’) – a számlálónkba, azt egy ún. segédtáblázat elkészítésével tudjuk meghatározni (3.1.3.c ábra).

segédtáblázat a kettős ugrás lekezelésére

	N2	N1	N0
$\overline{X}$	1	0	1
X	1	0	0

Ha q = 2,

$N2(D2) = 1$

$N1(D2) = 0$

$N0(D2) = \overline{X}$

3.1.3.c ábra

A 2.mintafeladat segédtáblázata a „kettős ugrás” kezeléséhez

Mivel a beolvasandó szám értéke *X*-től, mint logikai változótól függ, ezért a táblázat soraihoz *X* lehetséges értékeit rendeljük, az oszlopokhoz pedig az egyes adatbemeneteket (MSB szerint N_2 , N_1 , N_0). Következő lépésként a táblázaton belül beírjuk az egyes *X* feltételek esetén a bemeneteken megjelenítendő számértékeket bináris kód szerint. Vagyis egy olyan táblázatot alkottunk, amiből egyértelműen ki lehet olvasni, hogy *X* függvényében milyen logikai értékeket kell az adott számlálóállás (példánkban *q*=2) esetén felprogramozni a multiplexerek megfelelő adatbemeneteire.

A táblázat N_2 bemenethez tartozó oszlopában azt látjuk, hogy *X* bármely logikai értéke esetén ezen a bemeneten keresztül magas szintet kell beolvasnunk a számlálónkba, így az N_2 bitvektor bemenethez csatlakozó multiplexerünk 2-es adatbemeneti lábára (D_2) konstans magas szintet kell kötnünk.

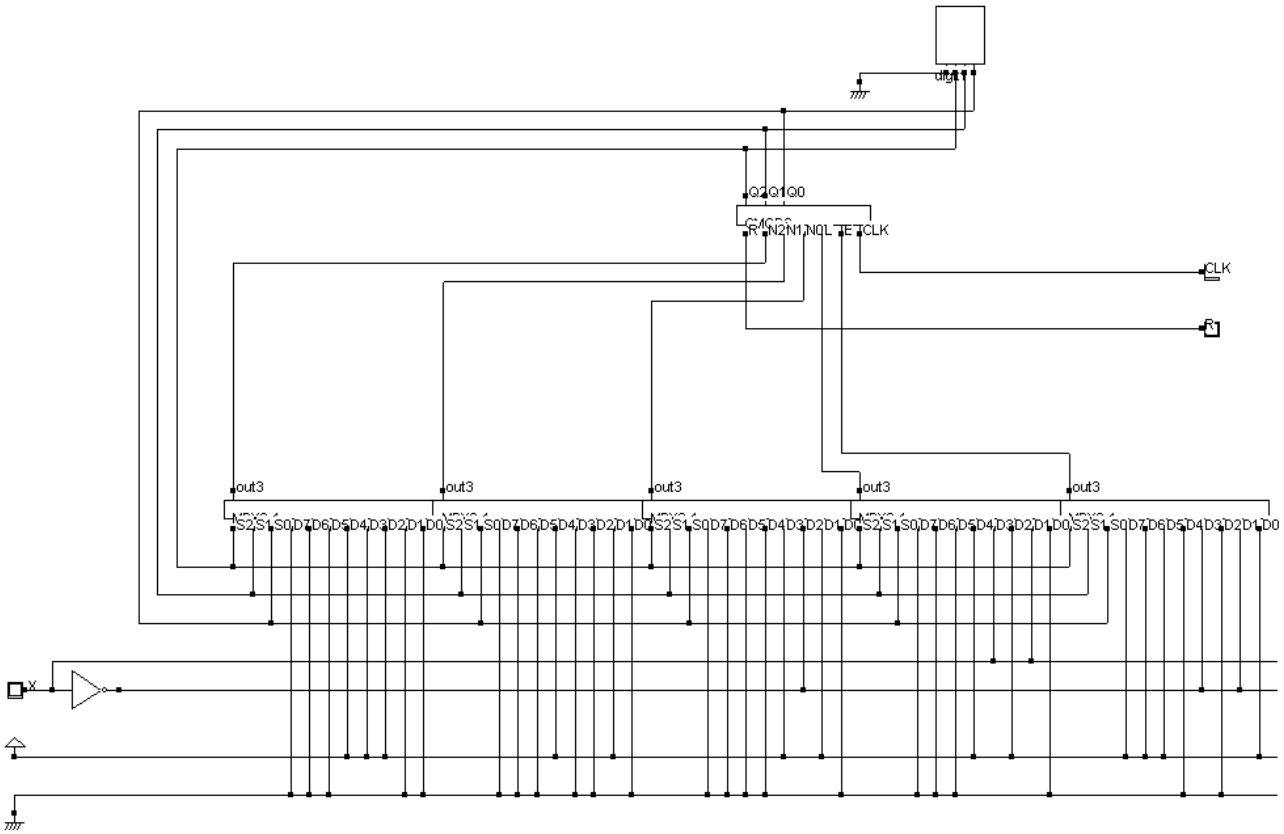
A táblázat N_1 bemenethez tartozó oszlopában pedig azt látjuk, hogy *X* bármely logikai értéke esetén ezen a bemeneten keresztül alacsony szintet kell beolvasnunk a

számlálónkba, így az N_1 bitvektor bemenethez csatlakozó multiplexerünk 2-es adatbemeneti lábára (D_2) konstans alacsony szintet kell kötnünk.

A táblázat N_0 bemenethez tartozó oszlopában található bejegyzések a következő logikát diktálják: ha *X* alacsony szintű, akkor az N_0 bitvektor bemenethez csatlakozó multiplexerünk 2-es adatbemeneti lábára (D_2) magas szintet kell kapcsolnunk, abban az esetben pedig, amikor *X* magas szinten van, ugyanerre a bemenetre alacsony szintet kell biztosítani. A feladatot csak úgy tudjuk megoldani, ha ezt a bemenetet *X* feltételnek a táblázatból adódó függvényeként vezéreljük, vagyis az $\bar{X}$ -at kötjük rá.

Általánosságban kimondhatjuk azt a szabályt, hogy kettős ugrás esetén mindig az adott feltételt szolgáltató változó függvényeként kell meghatározni az érintett adatbemeneti értéket.

A konkrét realizációhoz *mod-8*-as szinkron számlálóra és *mpx8-1* multiplexerekre, valamint egy inverterre lesz szükségünk. A DSCH szimulátor programban megszerkesztett architektúra a 3.1.3.d ábrán látható:



3.1.3.d ábra

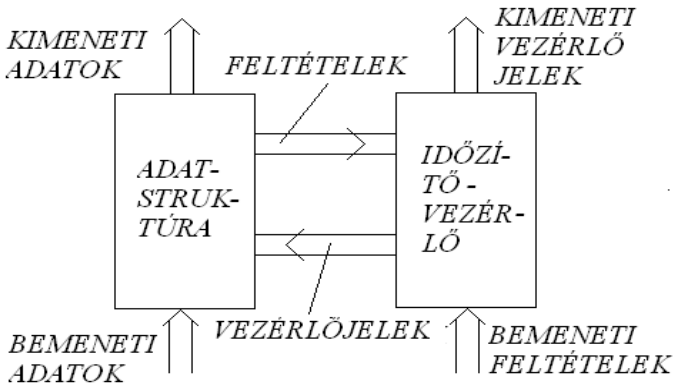
A 2.mintafeladat DSCH szimulátorban előállított realizációja

3.1.4 Szinkron sorrendi hálózatok tervezése szinkron számlálóval: 3.mintafeladat

3.1.4.1 Szinkron számlálók alkalmazása időzítő-vezérlő egységekben

A harmadik mintafeladat ismertetése előtt tekintsük át röviden, mik azok az időzítő-vezérlők, és milyen helyet foglalnak el a digitális architektúrákban.

Digitális berendezések tervezésekor célszerű dekompozícióval meghatározni az egyes hálózati működési funkciókhoz tartozó részstruktúrákat. Ezen elv alapján célszerű elkülöníteni az adatutakat az azok kijelölését és időbeli mozgatását meghatározó időzítő-vezérlő egységtől (3.1.4.1.a ábra).



3.1.4.1.a ábra

Digitális egység dekompozíciója

Az adatstruktúrára és időzítő-vezérlő egységre való felbontás szükségessé teszi a két egység közötti kapcsolódási pontok pontos meghatározását. Az adatstruktúra lényegében regiszterekből, multiplexerekből és funkciós egységekből áll, bemeneti adatokat fogad, és kimeneti adatokat szolgáltat. Ezek az egységek egymással összekapcsolódva különféle adatpályákat tesznek lehetővé. Az, hogy ezek közül adott helyzetben mely pályák valósulnak meg, illetve e pályáknak mely szakaszai aktivizálódnak egy adott pillanatban, ezt az időzítő-vezérlő határozza meg. Az időzítő-vezérlő címzi meg a multiplexerek és forrásait illetve kimeneteit, állítja be funkciós egységek műveletvezérlő kódjait, és felemeli, illetve lebocsátja a regiszterek beíró jeleit. Ezeket nevezzük vezérlőjeleknek. Az időzítő-vezérlő működését azonban az adat-struktúrából rávezetett jelek, feltételek befolyásolják. Ugyanakkor a környezet bemeneti feltételekkel befolyásolja az időzítő-vezérlő működését, és maga az időzítő vezérlő is

szolgáltat kimeneti vezérlő jeleket a környezet számára. A következőkben az időzítő-vezérlő egységet röviden vezérlő egységnek fogjuk nevezni.

A számláló típusú vezérlő alapegysége egy törölhető(R), engedélyezhető(E), betölthető(L) szinkron számláló, amely a kapcsolódó multiplexerekkel együtt egy állapot-kimenetű szinkron sorrendi hálózatot valósít meg. A 3.1.4.1.b ábra szerinti sematikus felépítést tanulmányozva - a számláló vezérlőbemenetei között fennálló elsőbbségi viszonyokat figyelembe véve a működés a következőképpen írható le:

- ha az R bemenetet felemeljük, a számláló tartalma a fennálló helyzettől függetlenül nullázódik. Ezzel szemben, ha

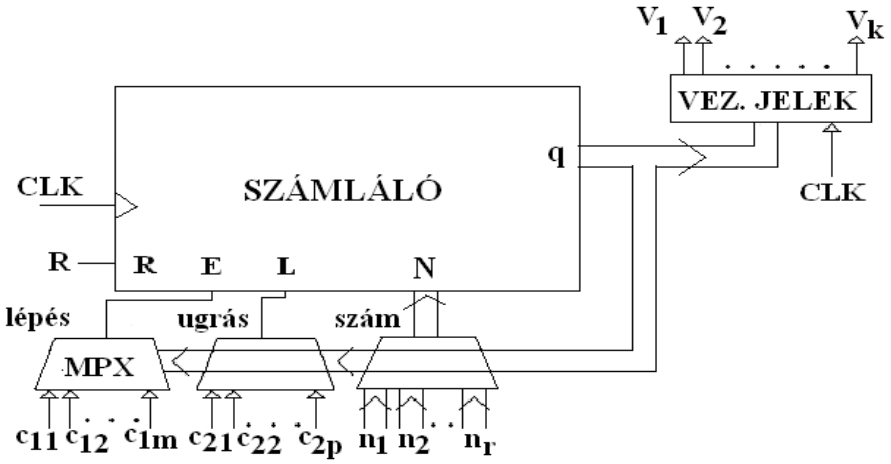
- az R bemenet szintje alacsony, akkor két eset van:

    1. ha a számláló kimenete által megcímzett, a betöltést vezérlő bemenetre kapcsolódó feltétel magas szintű, akkor az általa egyidejűleg megcímzett szám betöltődik a számlálóba. Ezzel szemben áll,

    2. ha a betöltésre kapcsolódó feltétel szintje alacsony. Ekkor ismét két eset van:

        (a) egyik, ha a kiválasztott, az engedélyező bemenetre kapcsolódó feltétel magas. Ilyenkor a számláló tartalma inkrementálódik, ezzel szemben

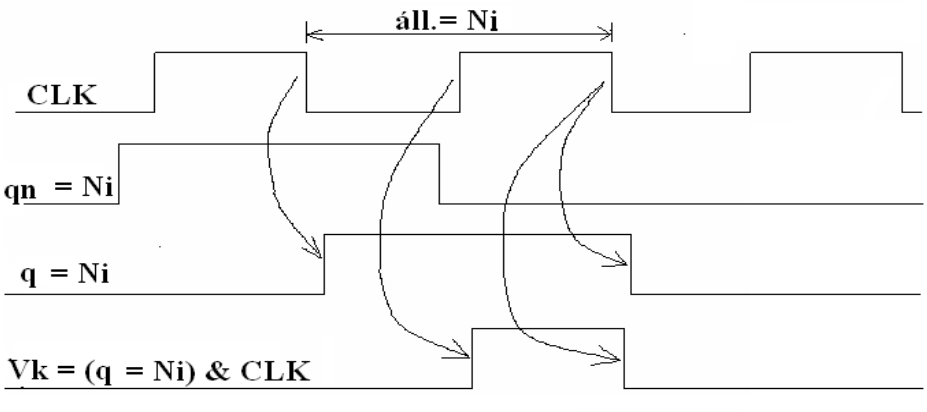
        (b) ha az engedélyező bemenetre kapcsolódó feltétel szintje alacsony, akkor a számláló értéke nem változik.



3.1.4.1.b ábra

Vezérlőegység kialakítása szinkron számláló felhasználásával

A fenti működési fázisokat az előző fejezetben már részleteztük, de a vezérlő másik egységét ott nem tárgyaltuk: ez az egység a környezetnek és az adat-struktúrának szóló vezérlőjeleket generálja. A vezérlőjelek generálásának alapproblémája a hazárdmentesség. Ez azt jelenti, hogy a vezérlőjelek csakis a tervező által meghatározott időintervallumokban lehetnek aktívak, azokon kívül stabilan passzív logikai szinten kell azokat tartani. Ha valamennyi vezérlőjel aktivitását magával az órajellel szinkronizáljuk, akkor a 3.1.4.1.c. ábra szerinti időDiagram igazolja a hazárdmentességet:



3.1.4.1.c. ábra

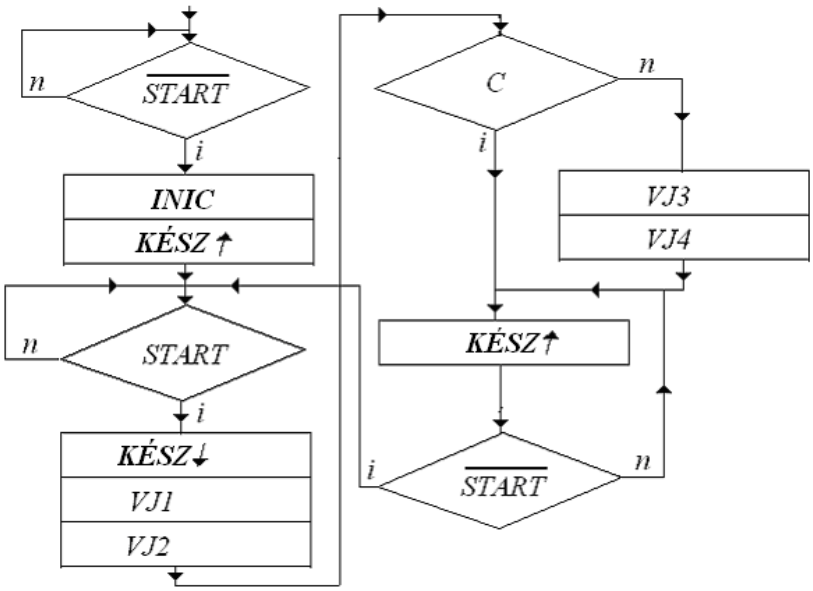
Hazárdmentes vezérlés megvalósítása vezérlőegységben

Legyen q_n a következő számláló állás, q az aktuális számláló állás. Tegyük fel, hogy a q = N_i számláló álláskor, és csakis akkor szeretnénk a V_k vezérlőjel aktivitását kiváltani, az órajellel szinkronban. Belátható, hogy a vezérlőjel bekövetkezése után következő órajel már az időzítési feltétel biztos elmúlása után jelenik meg! Ezzel biztosított a hazárdmentes vezérlés.

3.1.4.2 A 3.mintafeladat

Példa:

tervezzünk időzítő-vezérlő egységet szinkron számláló felhasználásával az alábbi (3.1.4.2.a ábra) folyamatábra alapján!



3.1.4.2.a ábra

A 3.mintafeladat folyamatábrája

A folyamatábrán látható működés szerint az adatstruktúra számára szolgáltatni kell egy *INIC* nevű jelet, valamint a *VJ1, VJ2, VJ3, VJ4* vezérlőjeleket, a környezetből fogadni kell egy *START* jelet, és értelmezzük a *C* nevű jelet az adatstruktúrából eredő feltételnek. Ezeken felül a környezet számára még szolgáltatni kell egy *KÉSZ* jelet is.

A folyamatábra pontos értelmezéséhez a szimbólumok jelentése a következő:

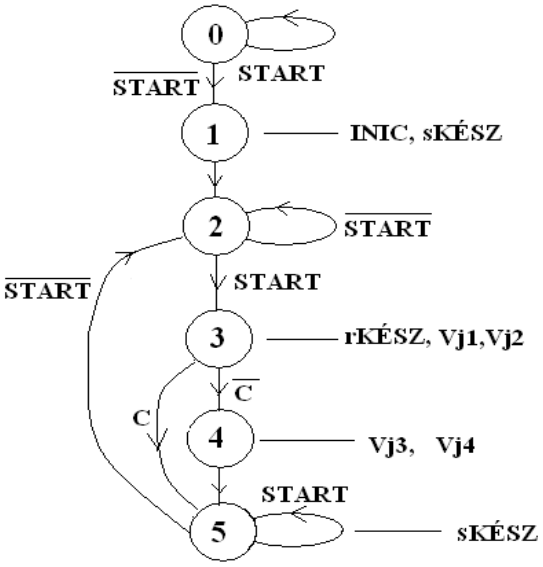
- rombusz, logikai feltétellel: ha igaz, az 'i' kimeneten távozunk
- négyszög, jelnév-bejegyzéssel: órajellel szinkronizált pozitív impulzus a megnevezett jelen
- négyszög jelnév-bejegyzéssel és felfelé vagy lefelé mutató nyíllal: az órajel felfutó élére szinkronizált felfutás vagy lefutás a megnevezett jelen

Megoldás:

Először is értelmezzük és határozzuk meg a hálózat működését a megadott folyamatábra alapján:

az ábra szerint a vezérlőegység a kezdeti állapotból elindulva - amennyiben a *START* alacsony - produkál egy *INIC* impulzust, és a *KÉSZ* felemelésével jelzi a készenlétet. Ezután vár a S*TART* felfutó élére. Ha az megjelenik, visszaejti a *KÉSZ* jelet, és a *VJ1, VJ2* vezérlőjeleken párhuzamosan impulzust produkál. A következő akciók a *C* bemenettől függenek: amennyiben *C* alacsony szintű, akkor a *VJ3, VJ4* impulzusai megjelennek, ha viszont magas, akkor azok kimaradnak. Ezután megint fel kell emelni a *KÉSZ* jelet, és a *START* szintje szerint visszatérni.

Következő lépésként célszerű a vezérlés folyamatának ütemezését, azaz egy vezérlési szekvencia leírását egy számláló-állásokkal jelölt állapotgráfban elkészíteni (3.1.4.2.b ábra):



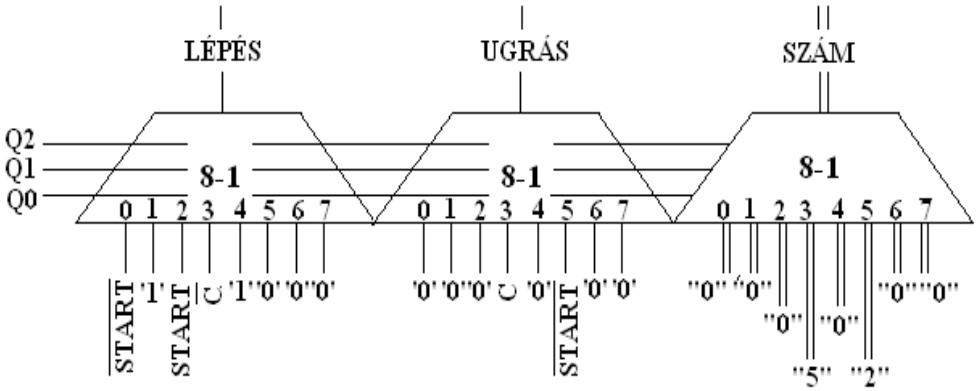
3.1.4.2.a ábra

A 3.mintafeladat állapotgráffal megadott vezérlési szekvenciája

A számlálóállásokhoz rendelt akciókat a gráf jobboldalán soroljuk fel. Az akciórész lehet üres. Ilyen például az első ütem, amikor a *START* értékétől függően várakozunk, vagy egy ütemmel továbblépünk. Figyeljük meg, hogy itt a jel emeléseket és ejtéseket '*s*' vagy '*r*' előtagokkal jelöljük, érzékeltetve ezzel, hogy az ilyen típusú jeleket *SR* bistabil tárolókkal állítjuk elő. Így minden ilyen jelhez két impulzus típusú jelet rendelünk. Ezzel azt is elérjük, hogy az ütemezett vezérlési szekvencia már csak impulzus típusú jelekre hivatkozik.

A teljes szekvencia leírásához szükséges állapotok számából adódik, hogy a realizáláshoz mod-8-as számlálót kell használni, ami azt is jelenti, hogy 3 darab mpx8-1 multiplexerre lesz szükségünk.

A multiplexerek programozását is a vezérlési szekvencia leírásából lehet meghatározni. Az ehhez szükséges táblázatok elkészítése már rutinmunka: a „*LÉPÉS*" multiplexer megtervezésekor azokat az ütemeket gyűjtjük össze az ide tartozó táblázatban, amelyekben inkrementálás található. Az ütemek az inkrementáláshoz tartozó feltételeket címzik. Feltétel nélküli inkrementálási ütemhez nyilvánvalóan konstans logikai '1' tartozik. A másik szükséges táblázat felrajzolásánál hasonlóan gyűjtjük össze az „*UGRÁS*" multiplexer címeit és bemeneteit is. Ugyanezekhez a címekhez rendeljük a „*SZÁM*" multiplexer bemeneteit is, amelyekhez az ugrásokhoz beírt ütem-számokat rendeljük. Az így meghatározott multiplexer-hálózat programozása a 3.1.4.2.b ábrán látható, ahol a címzőbemenetekként megjelenő Q_1, Q_2 *és* Q_3 a mod-8-as számlálónk számláló(adat) kimeneti bitjei:



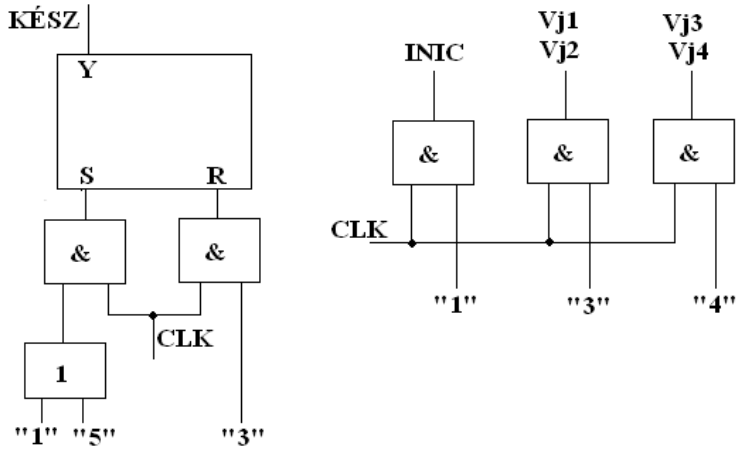
3.1.4.2.b ábra

A 3.mintafeladat multiplexereinek programozása

A komplett realizáció elkészítéséhez még szükség van a vezérlőjel-generátorok megtervezésére.

Az *INIC* valamint a *VJ1, VJ2, VJ3, VJ4* vezérlőjelek előállítása NEM-VAGY kapukkal történhet, a számlálónk kimenetéről leolvasott és dekódolt ütemszámok felhasználásával. Mivel a kapuk bemenetére a negáltakat kell kapcsolni, a dekódolt ütemszámok negáltját tüntettük fel a kapuk bemenetein. A *KÉSZ* jel előállítása bistabil SR tárolón keresztül történik. Látható, hogy az alkalmazott tárolón is az órajellel szinkronban történik meg a változás! Az egység kapuszintű sematikus vázlata a 3.1.4.2.c

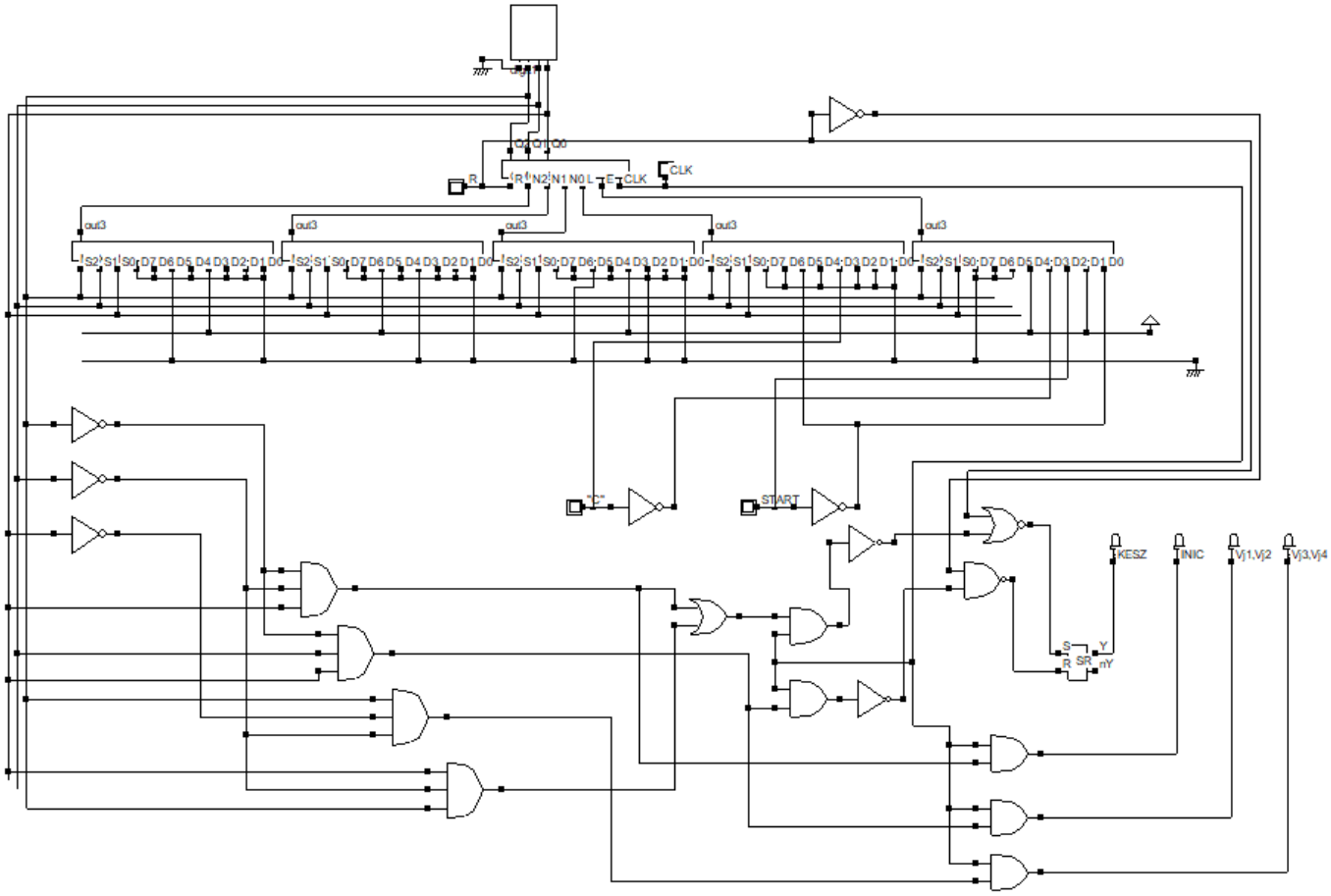
ábrán látható (az egyes ’számlálóállások’ dekódolt formában, egyetlen bittel kerültek feltüntetésre az ábrán):



3.1.4.2.c ábra

A 3.mintafeladat vezérlőjel-generátorainak kapuszintű sematikus vázlata

A feladat komplex kapuszintű realizációja (DSCH) a 3.1.4.2.d ábrán látható:



3.1.4.2.d ábra

A 3.mintafeladat realizációja (DSCH)

Ellenőrző kérdések:

Mit jelent a JK-MS flip-flop „kettes-osztó” funkciója?

Milyen be- és kimenetei vannak egy mod-m számlálónak?

Milyen átalakításokat lehet elvégezni egy adott modulusú szinkron számlálón?

Milyen tervezési lépésekkel lehet megvalósítani egy számláló alapú szinkron sorrendi hálózatot?

Ismertesse (sematikus ábra) a számláló bázisú időzítő-vezérlő egység strukturális felépítését! Mit jelent a hazárdmentes vezérlés feltétele (ábra!)?