II. modul: Sorrendi hálózatok tervezése

4. lecke: Aszinkron sorrendi hálózatok tervezése mintapéldákon keresztül: kritikus versenyhelyzet és lényeges hazárdok aszinkron sorrendi hálózatokban.

Cél:

A lecke célja, hogy a hallgató elsajátítsa az aszinkron sorrendi hálózatok tervezésének fő lépéseit, felismerje és kiküszöbölje a kritikus versenyhelyzeteket, ismerje a lényeges hazárdok fogalmát. Szimulációs program 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 az aszinkron sorrendi hálózatok tervezési eljárásainak főbb lépéseit
- felismeri a kritikus versenyhelyzeteket és képes kiküszöbölni azokat
- meg tudja határozni és be tudja állítani az előírt kezdeti állapotot (RESET logika) egy aszinkron sorrendi hálózatban
- ismeri a lényeges hazárd fogalmát
- tervezőprogram segítségével áramköri realizációt tud építeni, és szimulátor segítségével képes értelmezni a különböző tervezési eljárásokkal megépített áramkörök működési sajátosságait

Időszükséglet:

A tananyag elsajátításához kb. 180 percre van szükség.

Kulcsfogalmak:

- aszinkron sorrendi hálózat
- kritikus versenyhelyzet
- RESET logika, lényeges hazárd

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

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

4.1 Aszinkron sorrendi hálózat tervezése: 1.mintafeladat - a kritikus versenyhelyzet felismerése és kiküszöbölése aszinkron sorrendi hálózatokban

Példa:

közvetlenül visszacsatolt kombinációs hálózattal tervezzünk olyan egybemenetű (X) és egykimenetű (Z) aszinkron sorrendi hálózatot, amelynek kimenetén a szint mindannyiszor ellenkezőjére vált, ahányszor a bemenet magas szintről alacsony szintre vált. Bekapcsolás után a hálózat az X=0 bemenetnél Z = 0 kimenetet szolgáltasson!

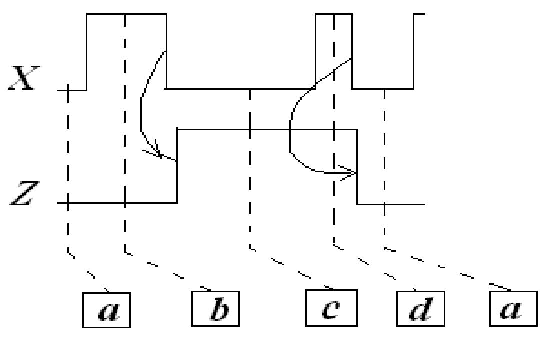
Megoldás:

Aszinkron sorrendi hálózatokban nincs szinkronizáló jel, azaz a következő állapot (*n+1*) értelmezését nem egy órajel ciklus lezajlása után determináljuk. Ezekben a hálózatokban általánosságban azt feltételezhetjük, hogy a bemenet minden egyes változása egy új állapotot generál. Természetesen ez nem feltétlenül van így minden esetben, hiszen egy következő (*n+1*) állapot gyakran megegyezik az aktuális állapottal (*n*), amire már az előző feladatok során láttunk példákat. Amikor arról döntünk, hogy egy adott változásra új állapotot vegyünk-e fel, vagy megteszi egy már felhasznált szimbolikus állapot, gondoljunk arra, hogy új állapot felvétele sohasem vezet logikai hibához, és a feltétlenül szükséges állapotokat később úgyis szisztematikus módszerrel határozzuk meg (állapot összevonás). Ezzel szemben az, ha egy régi állapotot használunk fel kellő óvatosság és meggondolás nélkül, abból könnyen lehet funkcionális hiba. Ezért az a legfontosabb, hogy ismerjük fel azokat az eseteket, amikor nem szabad régi állapotot felhasználni. (Ezek listáját majd később, némi példa megoldási tapasztalat birtokában fogjuk megadni.)

A feladat megoldásának menete – a szinkron hálózatoknál már megismert szisztematikus eljárási módszer alapján - a következő:

1. *Idődiagram (állapot-átmeneti gráf) felrajzolása.*
2. *Előzetes szimbolikus állapottábla felvétele.*
3. *Összevont szimbolikus állapottábla megszerkesztése.*
4. *Kódolt állapottábla elkészítése a kritikus versenyhelyzet elemzésével*
5. *- közvetlen visszacsatolt kombinációs hálózat esetében a szekunder változó(k) és a kimenet(ek) függvényeinek Karnaugh tábla segítségével történő statikus hazárdmentesített megadása algebrai alakban;*
   - SR tárolók alkalmazása esetén a vezérlési tábla, valamint a szükséges Karnaugh táblák alapján az SR tároló(k) és a kimenet(ek) statikusan hazárdmentesített algebrai függvényalakjainak megadása.
6. *Kezdeti állapotról történő gondoskodás.*
7. *Lényeges hazárdok vizsgálata (szükség esetén késleltetések beiktatása).*
8. *Realizáció logikai kapukkal (és tárolókkal).*

Ahogyan a szinkron hálózattervezés kezdeti lépéseként az állapotgráf felvétele ajánlott, úgy ajánlható aszinkron hálózat tervezésének első lépéséül egy ún. *idődiagram* felvétele. Az idődiagram (vagy más néven ütemdiagram) felvételekor figyelembe kell venni, hogy az aszinkron hálózat minden új stabil állapotba való elindulása egy bemeneti jel változására indul meg, és működtetési szabály, hogy egyidejűleg csak egyetlen bemeneti jel változhat. Az idődiagram jól mutatja a bemeneti jelváltozások és a kimeneti kombináció-változások közötti ok-okozati összefüggéseket, és segítséget nyújt az előzetes, szimbolikus állapottábla felvételéhez(4.1.a ábra):



4.1.a ábra[1]

Az 1.mintafeladat idődiagramja

Az idődiagramban az *X=0* bemenethez tartozó kezdeti állapotot $\boxed{a}$-val jelöltük, és ehhez felvettük a *Z=0* kezdeti kimeneti értéket. Az *X* első felfutása hatástalan, de fontos hogy az első felfutás tényét a hálózat regisztrálja egy új, $\boxed{b}$ állapotba menetellel. Az ezután bekövetkező lefutás nemcsak újabb állapotváltást ($\boxed{c}$), de a kimenet felfutását is kiváltja. A $\boxed{c}$ állapot egyértelműen jelzi, hogy az *X* első lefutása bekövetkezett. *X* második felfutás ismét új állapot bevezetését igényli ($\boxed{d}$)a kimenet változása nélkül. Az is érthető, hogy az újabb lefutás a kezdeti állapotot ($\boxed{a}$)állítja be, mind a belső, mind a kimeneti állapot szempontjából.

Következő lépésként meg kell adni a feladathoz tartozó előzetes szimbolikus állapottáblát. Az aszinkron hálózat szimbolikus előzetes állapottáblájának felvétele ugyanúgy intuitív módon oldandó meg, mint a szinkron hálózatok esetében, de általában nehezebb feladat annál. Fontos tervezői döntés, hogy Mealy-, vagy Moore-típusú hálózatot akarunk-e tervezni, hiszen az előzetes állapottábla felépítése ettől jelentősen függ. A specifikáció alapján itt is meg kell határoznunk egy kezdeti állapotot, amelybe a realizált hálózatnak a bekapcsolás után kerülnie kell, és szükséges, hogy ehhez egy bemeneti kombináció tartozzék. A táblázatban az idődiagram alapján, a bemenetekre előírt jelváltozási szabály szem előtt tartásával előírjuk a következő szimbolikus állapotokat. Úgy képzeljük, hogy egy új állapot kezdetben tranziens állapotként jelentkezik az f_y kimenetén, majd a bemenetre visszajutva stabilizálódik.

Példánkban a felrajzolt idődiagram alapján szerkesztett előzetes szimbolikus állapottábla a 4.1.b ábrán látható:

akt.áll. \ bem.	*X=0*	*X=1*
a	*a*/0	*b* /0
b	*c* /1	*b*/0
c	*c*/1	*d* /1
d	*a* /0	*d*/1

4.1.b ábra[1]

Az 1.mintafeladat előzetes szimbolikus állapottáblája

Az állapottáblán körbe-foglalással jelöljük a stabil állapotokat. Tudjuk, hogy aszinkron állapottáblán stabil következő állapot az, amelynek szimbóluma azonos az aktuális állapot szimbólumával. A működést az állapottáblán is követhetjük. A kezdeti *a* aktuális állapotban az *X=0* bemenet az *a* állapotot stabilizálja. Az állapot mellett „/" jellel elválasztva látjuk a kezdeti kimeneti értéket.

Induljunk most el ebből a stabil *a* állapotból az *X=1* változás hatására a következő *b* stabil állapotba! Ez az ún. *stabil-stabil állapot átmenet* a következőképpen értelmezhető (4.1.c ábra): a bekarikázott *a* állapot *X=0* oszlopából átlépünk ugyanezen sor *X=1* oszlopába, ahol a *b,* mint *tranziens* (nem stabil, azaz átmeneti) következő állapotkódot találjuk, változatlan *Z* értékkel. Ha ez visszajut a bemenetre, akkor átlépünk a *b* aktuális állapot sorára, ahol ebben az oszlopban *(X=1*) azt látjuk, hogy a *b* állapot stabilizálódik. A táblázatban szereplő többi stabil-stabil állapot átmenet is (*b→c*, *c→d*, *d→a*) hasonlóképpen követhető.

bem. / akt.áll.	X=0	X=1
a	*a* /0	*b* /0
b	*c* /1	*b* /0
c	*c* /1	*d* /1
d	*a* /0	*d* /1

4.1.c ábra[1]

Stabil-stabil állapot átmenet (a→b) szemléltetése az 1.mintafeladat előzetes szimbolikus állapottáblájában

Miután megszerkesztettük az előzetes szimbolikus állapottáblát, következő lépésként vizsgáljuk meg, lehetséges-e az állapotok számának csökkentése. Az összevonhatóság feltételei alapján sajnos ebben az esetben nem találunk nem megkülönböztethető állapotokat, így a kódolt állapottábla felvételénél ebből a táblázatból kell kiindulnunk.

Mivel négy állapotot kell bináris kóddal megkülönböztetni, ezért ehhez két szekunder változót (Y_1; Y_2) vezetünk be. Az egyes állapotokhoz rendeljük hozzá az alábbi (szabadon választott) kódolást:

	Y_1	Y_2
a :	0	0
b :	0	1
c :	1	0
d:	1	1

A kódolt állapotokat a 4.1.d ábrán látható kódolt állapottábla tartalmazza:

kódolt aktuális állapot $Y_1^v Y_2^v$	**kódolt következő állapot/kimenet** X=0 $Y_1^v Y_2^v$/Z	X=1 $Y_1^v Y_2^v$/Z
0 0	*0 0/0*	*0 1/0*
0 1	*1 0/1*	*0 1/0*
1 0	*1 0/1*	*1 1/1*
1 1	*0 0/0*	*1 1/1*

4.1.d ábra

Az 1.mintafeladat kódolt állapottáblája egy szabadon választott kódolás alapján

A feladat megoldásában ennél a pontnál egy nagyon fontos vizsgálatot kell elvégeznünk!

Ennek oka pedig a következő: eddigi tanulmányaink során a szinkron hálózatok állapotkódolásánál szabad kezünk volt abban, hogy a megfelelő hosszúságú szavakból álló kódkészlet szavait hogyan rendeljük hozzá a szimbolikus állapotokhoz, ugyanis minden választás a specifikációnak megfelelő megoldáshoz vezetett. Aszinkron hálózatok állapotkódolásakor nem ilyen jó a helyzet!

<u>*Amennyiben egy tranziens állapot kódja egynél több szekunder változó értékében különbözik a kiinduló stabil állapot kódjától, a reális hálózaton az eltérő jelkésleltetési utak miatt átmenetileg olyan más tranziens állapotok is jelentkezhetnek az f_y hálózat kimenetén, amelyek stabilizálódhatnak. Ezzel más, a specifikációnak ellentmondó pályára áll az aszinkron hálózat. Az ilyen hibalehetőségeket **kritikus versenyhelyzeteknek** nevezzük.*</u> Kiküszöbölésükre számos módszert dolgoztak ki, amelyek a kód megfelelő megválasztását eredményezik. Ezek közül most egy egyszerű, intuitív módszert mutatunk be a feladat kapcsán. (Egy későbbi fejezetben egy szisztematikus állapotkódolási módszert is megismerünk majd.)

Vizsgáljuk meg a 4.1.e ábrán látható táblázat segítségével a ’*01*’→’*10*’ stabil-stabil állapot átmenetet!

kódolt aktuális állapot	**kódolt következő állapot/kimenet**	
$Y_1^v Y_2^v$	X=0 $Y_1^v Y_2^v$/Z	X=1 $Y_1^v Y_2^v$/Z
0 0	*0 0/0*	*0 1/0*
0 1	*1 0/1*	*0 1/0*
1 0	*1 0/1*	*1 1/1*
1 1	*0 0/0*	*1 1/1*

4.1.e ábra

Egy ideális, ’01’→’10’ stabil-stabil állapot átmenetet

Tegyük fel, hogy az *X=1*-hez tartozó ’*01*’ kódú *b* állapotban vagyunk. Ha most *X* bemenetet ’*0*’-ra kapcsoljuk, akkor tranziens állapotként az *’10*’ (*c*) állapot beállását várjuk, amely a bemenetre visszajutva stabilizálódik, és ezzel megtörténik az elvárt ’*01*’→ ’*10*’ átmenet. *Azonban ha a hálózatot ezzel az állapotkóddal megvalósítjuk, hibás lehet a valóságos működés!* A 4.1.f ábrán szemléltetjük, mi lesz annak a következménye, ha a késleltetési idők különbözősége miatt egy másik tranziens állapot, a ’*00*’ (*a*) áll be.

Ebben az esetben az átmenet kétféleképpen is megvalósulhat:

1. először Y_2^v vált ’*1*’-ről ’*0*’-ra ($Y_1^v Y_2^v$ =’*01*’→’*00*’*)* az *X=1→0* hatására*.* Ekkor a ’*00*’ kódú sorban az *X=0* oszlopában következő állapotként a ’*00*’ kódot látjuk, ami azt

jelenti, hogy ebben a kódú állapotban stabilizálódik a hálózatunk. Ez helytelen, nem előírásszerű, vagyis HIBÁS működést eredményez, hiszen a kiindulásbeli, *'01'→'10'* vezérlés (4.1.e ábra) az *'10'* állapotkódba jutást irányozza elő!

2. először Y_1^v vált '0'ról '1'-re ($Y_1^v Y_2^v$ *='01'→'11')* az *X=1→0* hatására. Ez azt jelenti, hogy az '11' kódú állapot sorába ugrunk, az *X=0* oszlopban. Itt megint csak egy tranziens kódot látunk: *'00'*. Vagyis következő lépésként a *'00'* állapotkód sorába ugrunk, maradva természetesen ebben az oszlopban. Itt viszont most a következő állapot „önmaga", vagyis stabil *'00'*. Tehát ez esetben is helytelen a működés, és nem teljesül az eredetileg *'10'* állapotba jutás és stabilizálódás.

kódolt aktuális állapot	**kódolt következő állapot/kimenet**	
$Y_1^v Y_2^v$	X=0 $Y_1^v Y_2^v$/Z	X=1 $Y_1^v Y_2^v$/Z
0 0	*0 0/0*	*0 1/0*
0 1	*1 0/1*	1. *0 1/0*
1 0	*1 0/1*	2. *1 1/1*
1 1	*0 0/0*	*1 1/1*

A 4.1.f ábra

Kritikus versenyhelyzet az $Y_1^v Y_2^v$ ='01'→'10' állapot átmenet esetében

A kritikus versenyhelyzetek sok esetben egyszerű kód átrendezéssel, vagy a nem használt kódszavak bevonásával kiküszöbölhetők. A lényeg, hogy <u>*a kiindulási és a cél stabil állapotok kódjai között csak egy szekunder változó értékében legyen különbség.*</u>

Ezen elvet követve kódoljuk újra a megkülönböztetendő állapotokat a következők szerint:

	Y_1	Y_2
a :	0	0
b :	0	1
c :	1	1
d:	1	0

Készítsük el ez alapján újból a kódolt állapottáblát, és vizsgáljuk meg, valóban kritikus versenyhelyzet mentes-e a kódolásunk (4.1.g ábra)!

kódolt aktuális állapot $Y_1^v Y_2^v$	**kódolt következő állapot/kimenet**	
	X=0 $Y_1^v Y_2^v$/Z	X=1 $Y_1^v Y_2^v$/Z
0 0	*0 0/0*	*0 1/0*
0 1	*1 1/1*	*0 1/0*
1 1	*1 1/1*	*1 0/1*
1 0	*0 0/0*	*1 0/1*

4.1.g ábra

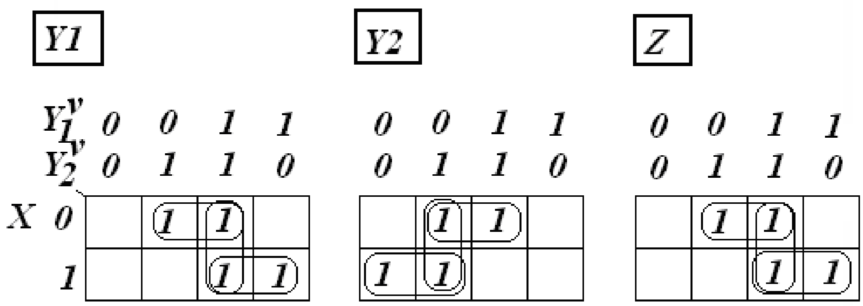
Az 1.mintafeladat kritikus versenyhelyzetektől mentes állapotkódolása

Láthatjuk, hogy egyetlen stabil-stabil állapot átmenet esetében sem merül fel kritikus versenyhelyzet, tehát a hálózatunk a specifikáció szerint fog működni.

A megfelelően kódolt állapottábla megszerkesztése után adjuk meg Karnaugh táblás egyszerűsítés segítségével a szükséges algebrai függvényalakokat! Ehhez készítsük el a szekunder változók és a kimenet kódolt állapottáblájából származtatott Karnaugh táblák kitöltését, és írjuk fel a minimalizáláshoz a lehetséges prímimplikánsokat.

Figyelem! Valamennyi szekunder változóhoz tartozó függvényt statikus hazárdoktól mentesen kell lefedni: az f_y hálózat kimenetein jelentkező statikus hazárd ugyancsak a specifikációtól eltérő, hibás működéshez vezethet, és általában követelmény a kimenet hazárdmentessége is!

Mindezek figyelembe vételével a 4.1.h ábra szerinti grafikus függvényalakokat kapjuk:



4.1.h ábra[1]

Az 1.mintafeladat megoldásához tartozó Karnaugh táblák

A Karnaugh táblákon felrajzolt prímimplikánsok algebrai alakjai:

$$
\begin{aligned}
Y_1 &= Y_2^v\,\overline{X} + Y_1^v X + Y_1^v Y_2^v \\
Y_2 &= \overline{Y_1^v}\,X + \overline{Y_1^v}\,Y_2^v + Y_2^v\,\overline{X} \\
Z &= Y_1
\end{aligned}
$$

<u>Megjegyzés:</u> *az eredményként kapott függvényeket vizsgálva a következő megállapítást tehetjük: Z nem függ a bemenettől, csakis egyetlen szekunder változótól (Y_1). Mealy-típusú hálózat tervezésébe fogtunk, mégis annak speciális eseteként, Moore-típusú hálózatot kaptunk.*

A feladat megoldásának utolsó lépéseként el kell készíteni a kapuszintű realizációt. Ezúttal azonban még egy fontos vizsgálat meg kell, hogy előzze az áramkör felrajzolását: *a helyes kezdeti állapot beállásának ellenőrzése*.

Ennek meghatározásához először is meg kell állapítani a kezdeti állapothoz tartozó elsődleges és másodlagos változók, valamint a kimenet(ek) értékét. Példánkban az *X* bemenő változó értékét a kezdeti állapotban ’0’-nak tekintettük. Ehhez a kezdeti *X=0* értékhez az *a* (kezdeti stabil állapotnak tekintett) állapotban a *Z=0* kimeneti értéket rendeltük. Az *a* állapothoz pedig kódoláskor az $Y_1^v Y_2^v$*=00* szekunder változó kódot rendeltük. Ezekből kiindulva pedig egyértelműen meg lehet és kell tudni állapítani, hogy szükség van-e külön kezdeti állapot beállító segédbemenetre.

Az ellenőrzéshez nem kell mást tennünk, mint a megoldásként kapott függvényekbe, mint egyenletekbe be kell helyettesíteni a kezdeti állapothoz tartozó ismert és előírt értékét. Ha egyértelműek az eredmények, és teljesül az egyenlőség, akkor nem kell külön kezdő állapot beállító jelről gondoskodni.

Példánkban az egyenletekbe történő behelyettesítés nem hoz egyértelmű eredményt, hiszen pl. Y_1 és Y_2 esetében sem látjuk biztosítva az egyenletek jobboldalán történő behelyettesítések eredményeképpen a baloldalon meghatározott ’0’ értékek megjelenését, mivel a bekapcsolás pillanatában a visszacsatolások miatt nem specifikált az Y_1^v *és* Y_2^v értéke, így a szorzatok eredménye is bizonytalan:

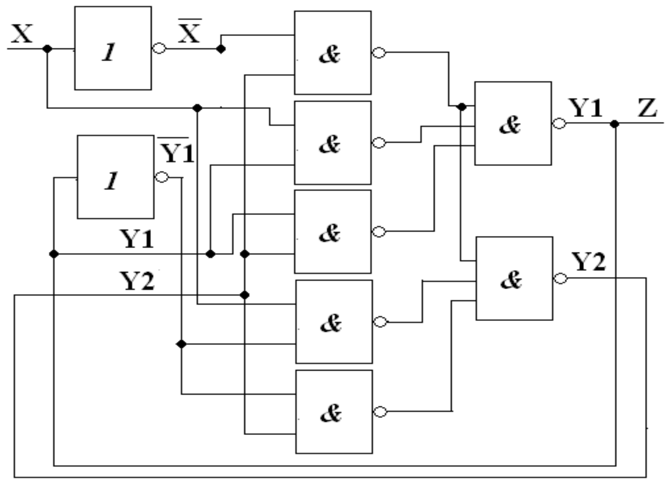
$$0 = Y_2^v \cdot 1 + Y_1^v \cdot 0 + Y_1^v Y_2^v$$
$$0 = \overline{Y_1^v} \cdot 0 + \overline{Y_1^v} Y_2^v + Y_2^v \cdot 1$$
$$0 = Y_1$$

Ebben az esetben mindenképpen célszerű a kezdeti állapot beállító jel (’*R*’=reset) alkalmazása. Az állapottáblából világosan megállapítható, hogy a tábla szerinti kezdeti állapot beállításának érdekében három feltételt kell teljesíteni:

1. a kezdeti állapot kódját rá kell kényszeríteni az f_y hálózatra, a visszacsatolástól függetlenítve ezeket a bemeneteket. Ezt a helyzetet legalább addig kell fenntartani, amíg az f_y kimenetein kialakul a kezdeti állapot kódja (illetve, ha SR tárolókkal történik a visszacsatolás, akkor azok kimenetén alakul ki ez a kód),
2. rá kell kapcsolni a hálózatra azt a bemeneti kombinációt, amely a kezdeti állapothoz tartozik
3. végül meg kell szüntetni a kényszerített visszacsatoló ágat (állapotot), és helyre kell állítani az eredeti visszacsatolást, ezzel a hálózat a kezdeti állapotban stabilizálódik.

(Megjegyzés: az aszinkron sorrendi hálózatok kezdeti állapotba kényszerítésének egyéb módszerei és lehetőségei egy későbbi modulban külön kerülnek megtárgyalásra.)

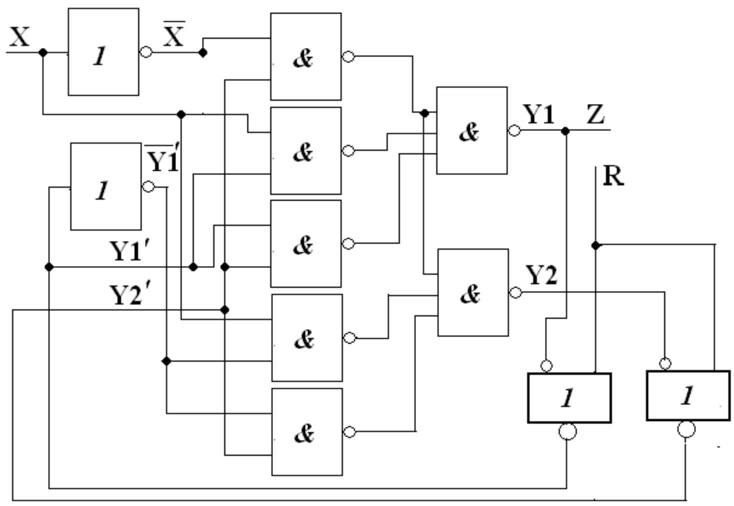
A kezdeti állapot beállító jel (’*R*’) nélküli realizáció a 4.1.i ábrán látható:



4.1.i ábra[1]

Az 1.mintafeladat realizációja kezdő állapot beállítás nélkül

A 4.1.j ábrán látható a már felhasított visszacsatoló ágba beillesztett *'R'* kezdő állapotot beállító jel:



4.1.j ábra[1]

Az 1. mintafeladat realizációja a kiegészítő 'reset'(R) logikával

A ’R’ jel áramköri beiktatásának elve a következő: hasítsuk fel a visszacsatolásokat, és illesszünk be a visszacsatoló körbe 2 darab két-bemenetű logikát. Az egyik bemenetük közös, a kezdeti állapotba kényszerítő ’R’ (reset) jel, a másik bemenetük a visszacsatolandó szekunder változókra ($Y_1^v Y_2^v$) kapcsolandó. A kimeneteket kapcsoljuk az f_y hálózat bemeneteire. Mindkét R-logika az R = ’1’ esetben ’0’-át ad tovább, ez pedig a kezdeti állapot kódja. Ha *R=0*, a logikák kimenetére a megfelelő szekunder változó kerül, tehát él a visszacsatolás.

4.2 Aszinkron sorrendi hálózatok tervezése: 2.mintafeladat

Példa:

*tervezzünk kétbemenetű (${\boldsymbol{X_1}}$,$\boldsymbol{X_2}$) ún. „sorrendi ÉS” áramkört, amelynek **Z** kimenete akkor és csakis akkor ad magas szintet, ha az $\boldsymbol{X_1}$ bemenet előbb áll ’**1**’-re, mint az $\boldsymbol{X_2}$. Kezdeti állapotban az $\boldsymbol{X_1 X_2}$**=00** bemeneti bitkombináció esetén **Z=0**. A tervezést végezzük el a következő állapotot előállító hálózat közvetlen visszacsatolásával, és SR tárolókkal történő visszacsatolással is !*

Megoldás:

Korábbi gyakorlatunktól eltérően – kellő rutin birtokában - a megoldást ezúttal rögtön az előzetes szimbolikus állapottábla felvételével kezdhetjük (ebben az esetben az idődiagram nem feltétlenül olyan szemléletes kiinduló alap, és az állapottábla anélkül is megszerkeszthető.) A tábla (4.2.a ábra) kitöltést célszerű a stabil kezdőállapot felvételével indítani: esetünkben ez a specifikáció szerint az $\boldsymbol{X_1 X_2}$***=00***-nál a ***Z=0,*** és rendeljük hozzá az ***a*** szimbólumot. (A kimenet egyes állapotokhoz meghatározandó értékénél a feladat specifikációjából egyértelmű, hogy ***Z*** magas szintjei csak az ’***11***’ oszlopban lesznek, de csak azoknál az állapotoknál, amelyek az ***’10’***-ban levő állapotokat követik.)

Induljuk el tehát a bekapcsolás utáni kezdeti ***a*** állapotunkból, és következő lépésként vegyük számba a lehetséges bemeneti változásokat. Az ***a*** állapotban az ’***11***’ bemeneti kombinációra való áttérés nem megengedett, hiszen ebben az esetben két bemeneti változó is értéket váltana, ezt pedig megtiltjuk. Így az ’***11***’-hez tartozó következő állapot és a kimenet értéke közömbös(’**-/-**’). A ’***01***’ és az ’***10***’ azonban megengedettek, mindkettőre új állapotot vettünk fel, és a hozzájuk tartozó kimenetek természetesen ’***0’***-k. A ***b*** és ***c*** állapot sorainak kitöltésekor ismét célszerű először a tiltott bemeneti kombinációkhoz tartozó bejegyzésekről gondoskodni. Ha a ***b*** állapotban ’***00***’ jelentkezik, az ***a*** állapotba mehetünk vissza. Ha ’***11***’ jön, akkor egy új, ***d*** állapotot vesszük fel, és ***Z***-t továbbra is alacsonyan tartjuk. A ***c*** állapotból ’***00’***-ra ***a***-ba mehetünk a ***Z=0***-val, ’***11***’-re viszont az új ***e*** állapot beálltához a ***Z=1*** tartozik, hiszen teljesült a speciális ’ÉS’ feltétel,

X_2 az X_1-t követően emelkedett magasra. Most a két legutóbb felvett állapotról, $\boldsymbol{d}$-ről és $\boldsymbol{e}$-ről kell gondoskodni. Egyikből sem kapcsolhatunk ’***00***’-ra, de a ’***01***’-re a $\boldsymbol{b}$ ***/ 0***, ’***10***’-ra a $\boldsymbol{c}$ ***/ 0*** jó választás, hiszen az előbbi esetben X_1 lefut, így a speciális ’ÉS’ feltétel teljesülésének lehetősége távolabbra kerül, a második esetben viszont fennmarad.

Az előzetes szimbolikus állapottábla a 4.2.a ábrán látható.

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

4.2.a ábra

A 2.mintafeladat előzetes szimbolikus állapottáblája

A következő lépésként az állapotok számának csökkentésével próbálkozunk, az állapot összevonás kettős feltételének figyelembe vételével. Ezt a szabályt most némi kiegészítéssel élve alkalmazhatjuk, ugyanis a közömbös bejegyzések lehetőséget adnak erre. Ez esetünkben azt jelenti, hogy *két állapot összevonható, ha bemenő kombinációnként megegyeznek a specifikált kimeneti kombinációk, és a specifikált következő állapotok.* Ennek alapján a következő párok vonhatók össze: ***ab , ad , bd , ce***. Az összevont állapotok tehát : (***abd***), (***ce***). Jelöljük az (***abd***) összevont állapotot s_1-gyel, a (***ce***)-t s_2-vel. Az összevont állapottábla sorainak kitöltésénél az eredeti tábla közömbös bejegyzései okoznak gondot. Könnyen belátjuk azonban, hogy mindig azt az összevont állapotot kell beírnunk, amelyhez tartozó állapot az adott oszlopban, az összevont állapot eredeti állapotainak sorában szerepelt. Például: s_1 sorában az ’***11***’ oszlopban s_1-et írunk, mivel az s_1-hez tartozó állapotok specifikált következő állapota a $\boldsymbol{d}$, ami az s_1-ben szerepel.

(Megjegyzés: általában 's_x' szimbólumokkal szokták jelölni az egyes állapotokat az ún. állapotgépes (state-machine) tervezéseknél, melyeket gyakran használnak elsősorban az ipari gyártási folyamatok szekvenciális működésének leírására. Az 's' szimbólum a 'state' szóból származik.)

Az ezek alapján elkészített összevont szimbolikus állapottábla a 4.2.b ábrán látható:

aktuális állapot,	**következő állapot/kimenet** X_1X_2			
	00	*01*	*10*	*11*
s_1	$s_1/0$	$s_1/0$	$s_2/0$	$s_1/0$
s_2	$s_1/0$	$s_1/0$	$s_2/0$	$s_2/1$

4.2.b ábra

A 2.mintafeladat összevont szimbolikus állapottáblája

A két állapot kódolásához egyetlen szekunder változó (***Y***) bevezetése elegendő: rendeljük az $\boldsymbol{s_1}$ állapothoz ennek '0' értékét, az $\boldsymbol{s_2}$-höz pedig az '***1***'-et. Nyilvánvaló, hogy ebben az esetben az egy szekunder változó miatt a kritikus versenyhelyzet problémája fel sem merül.

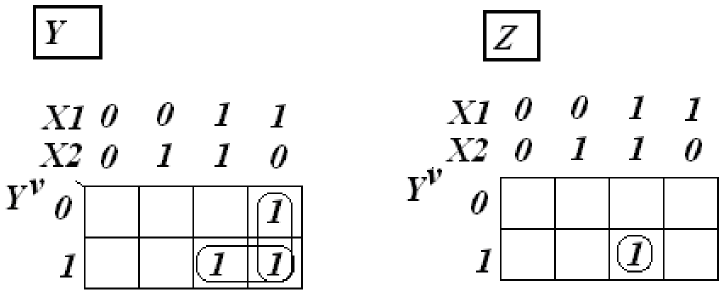
Az így kialakult kódolt állapottábla a 4.2.c ábrán látható:

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

4.2.c ábra

A 2.mintafeladat kódolt állapottáblája

A kódolt állapottábla szerint vegyük fel és töltsük ki a Karnaugh táblákat. A lefedések alapján felírhatjuk a realizáció alapjaként szolgáló függvényalakokat (4.2.d ábra):



4.2.d ábra[1]

A 2.mintafeladat Karnaugh táblái

A szekunder változóra és a kimentre kapott algebrai kifejezések a következők:

$$Y = X_1 \overline{X_2} + X_1 Y^v$$
$$Z = X_1 X_2 Y^v$$

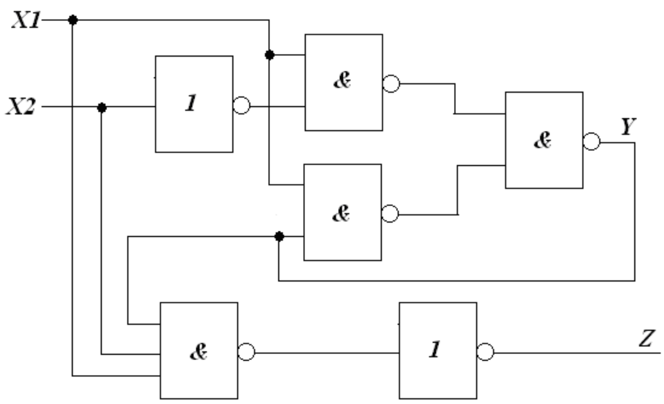
A realizáció elkészítése előtt meg kell még vizsgálnunk a kezdeti állapot beállását. Ezúttal könnyű dolgunk van ennek megállapításában, hiszen a függvényekbe behelyettesítve a megfelelő értékeket egyértelműen látszik, hogy a kezdeti állapotban (a) mind a másodlagos változó (Y), mind pedig a kimenet (Z) az előírt logikai értéket veszi fel:

$$0(Y) = 0 \cdot 1 + 0 \cdot Y^v$$

$$0(Z) = 0 \cdot 0 \cdot Y^v$$

vagyis nem kell külön 'R' jel beiktatásáról gondoskodnunk.

A kapuszintű realizáció NEM-ÉS logikával a 4.2.e ábrán látható:



4.2.e ábra[1]

A 2.mintafeladat NAND-NAND realizációval megvalósított áramköre

A realizációt a feladat kiírása szerint el kell végezni *SR* tárolók felhasználásával is. Ehhez az előzőekben megszerkesztett kódolt állapottábla alapján fel kell venni a feladatra specifikált *SR* vezérlési táblát is. Mivel egyetlen szekunder változónk van, ezért elegendő egyetlen tároló vezérlését megadni. Az *SR* tároló saját vezérlési táblája –a már ismert módon – az összetett igazságtáblájából származtatható a 4.2.f ábrán látható táblázat szerint:

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	–

Y^V →	Y	S	R
0	0	0	–
0	1	1	0
1	0	0	1
1	1	–	0

4.2.f ábra[1]

Az SR tároló összetett igazságtáblájából származtatott vezérlési táblája

A feladatra specifikált *SR* vezérlés táblája a 4.2.g ábrán látható:

kódolt aktuális állapot	kódolt következő állapot			
	X_1X_2			
	00	01	10	11
Q	S R	S R	S R	S R
0	0 -	0 -	1 0	0 -
1	0 1	0 1	- 0	- 0

4.2.g ábra

A 2.mintafeladat szekunder változóját reprezentáló SR tároló vezérlése

A származtatott Karnaugh táblák az *S, R* és *Z* függvényekkel a 4.2.h ábrán látható:

S

	X1	0	0	1	1
	X2	0	1	1	0
Y^v	0				1
	1			–	–

R

	X1	0	0	1	1
	X2	0	1	1	0
Y^v	0	–	–	–	
	1	1	1		

Z

	X1	0	0	1	1
	X2	0	1	1	0
Y^v	0				
	1			1	

$$S = X_1\overline{X_2}$$

$$R = \overline{X_1}$$

$$Z = X_1X_2Y^v$$

4.2.h ábra[1]

A 2.mintafeladat Karnaugh táblái és az azokból felírt S, R, Z függvényalakok

A realizáció megkezdése előtt ellenőrizzük a kezdeti állapot beállását! Ezúttal arra kell figyelnünk, hogy kialakul-e az *SR* tárolónk kimenetén a szükséges alacsony szint. Ehhez az *S* bemenetre ’0’-át, az *R* bemenetre mindenképpen ’1’-et kell biztosítani. Végezzük el a kapott függvényekbe a behelyettesítést! Ennek eredménye a következő:

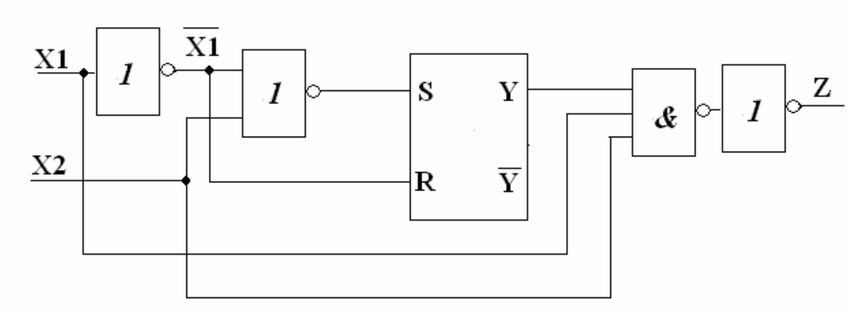
$$0(S) = 0 \cdot 1$$

$$1(R) = 1$$

$$0(Z) = 0 \cdot 0 \cdot Y^{v}$$

tehát nem kell külön ’*R*’ kezdeti állapot beállító jel beiktatásáról gondoskodni.

A kapuszintű realizáció NEM-ÉS logikával a 4.2.i ábrán látható:



4.2.i ábra[1]

A 2.mintafeladat SR tárolós realizációval megvalósított áramköre(NAND-NAND)

4.3 Lényeges hazárdok aszinkron sorrendi hálózatokban.

A hazárdjelenségek tárgyalása során osztályoztuk a hazárdokat viselkedésük szerint. Akkor a következő felsorolással éltünk:

- *statikus hazárdok*
- *dinamikus hazárdok*
- *funkcionális hazárdok.*

Az első három hazárdtípust már korábban specifikáltuk, létezik azonban egy negyedik típus is, a *lényeges hazárdok* családja. Mivel ezek a hazárdok aszinkron sorrendi hálózatokban fordulnak elő, ezért ebben a fejezetben értelmezzük és határozzuk meg pontosan, mit is takar ez a meghatározás.

Az eddigi aszinkron hálózat tervezési példáink megoldása során csak a szekunder változók versengése miatt kialakuló hibákkal, és azok kiküszöbölésével foglalkoztunk. Ez csak akkor tekinthető korrekt eljárásnak, ha garantálni tudjuk azt, hogy a bemeneti jelek változása okozta események a szekunder változók értékeinek megváltozása kezdete előtt már lezajlanak. Ez a feltételezésünk abban is megnyilvánul, hogy amikor az állapottáblán követjük az aszinkron hálózat működését, egyik oszlopról a másikra térünk át, és csak ezután vizsgáljuk a tranzienseket. A valóságban ez a feltételezés nem mindig jogos. *A szekunder változók és egyik bemeneti változó kritikus versenyhelyzete úgynevezett lényeges hazárd veszélyével jár. Ennek kiküszöbölése időkésleltetési manipulációkat igényel.*

Megjegyzés: a fent említett időkésleltetésekkel a feladatok megoldása során nem foglalkoztunk, mert feltételeztük a bemeneti értékek/kombinációk időben konstans jellegét a stabil-stabil állapot átmenetek vizsgálata alatt.

Felhasznált irodalom:

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