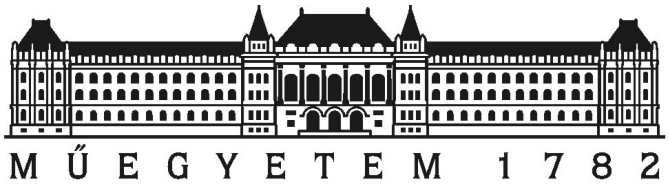


Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Dr. Pilászy György

Digitális technika 1

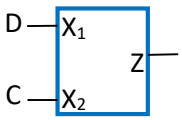
11. előadás

(Aszinkron sorrendi hálózat tervezése)

Lektorálta: Dr. Horváth Tamás

Aszinkron sorrendi hálózat tervezésének bemutatása

Aszinkron sorrendi hálózatok esetén a visszacsatoló ágakba nincs semmilyen órajellel vezérelt elem, amely lassítaná az Y kimenetek visszahatását a hálózat y bemeneteire. Az alábbiakban egy két bemenetű egy kimenetű aszinkron sorrendi hálózat tervezési lépéseit mutatjuk be egy konkrét példán keresztül.



A hálózat két bemenettel (D és C) és egy kimenettel (Z) rendelkezik. A Z kimenet értéke megegyezik a C bemenet 0→1 váltásának pillanatában tapasztalható D bemenet értékével. Más bemeneti változások a kimenet értékét nem befolyásolják.

Ha végig gondoljuk a fenti specifikációt, akkor feltűnhet, hogy ez éppen ráillik az előző pontokban megismert szinkron működésű D flip-flop leírására. Valóban, az elkövetkezőkben a felfutóél-vezérelt D-flip-flop belsejét tervezzük meg visszacsatolt aszinkron sorrendi hálózattal.

Első lépésben a hálózat aszinkron előzetes állapottábláját határozzuk meg. Az aszinkron működésről tanultak miatt pár sajátosságra hívjuk fel a figyelmet. Az előzetes állapottáblát úgy szerkesztjük meg, hogy minden sorban pontosan egy stabil állapot legyen. Figyelembe véve, hogy kizárólag szomszédos bemeneti változást engedünk meg a bemeneteken, így minden sorban lesz legalább egy közömbös bejegyzéssel kitöltött cella. Ahol az instabil állapoton keresztül haladva változik a kimenet, ott az instabil állapothoz tartozó kimenet értéket közömbössel jelöljük. Az előzetes állapottábla állapotait az abc kisbetűivel jelöljük. Majd az összevont állapottábla új állapotait fogjuk nagy betűkkel jelölni. A stabil állapotokat a könnyebb felismerhetőség érdekében félkövér betűtípussal írjuk.

Aszinkron hálózatot tervezünk. Aszinkron hálózat csak akkor vált állapotot, ha a bemenet megváltozik és a hálózat kimenete is csak ekkor változhat. Ebben a feladatban a hálózatnak akkor kell a kimenetet állítania, ha a C bemeneten 0→1 változás érkezik. Ekkor a kimenetnek fel kell vennie a D bemenet pillanatnyi értékét. Vagyis a kimenet állítás szempontjából két bemeneti kombináció-változás lesz lényeges: 00→01 váltás esetén a kimenetet 0-ra kell állítani, amennyiben az 1 értékű volt, 10→11 váltás esetén pedig 1-re, amennyiben az 0 értékű volt. A többi bemeneti kombináció váltás esetén a hálózatot olyan stabil állapotba kell vezetni, amely megtartja az eredeti kimeneti kombinációt.

A tábla kitöltésénél a következők szerint jártunk el. (a/00) sor első celláját önkényesen Z=0 értékre és stabil állapotra rögzítettük, mint induló állapot. Ezen stabil állapothoz tartozik egy nem szomszédos bemeneti kombináció, amelynek a cellájába közömbös bejegyzés kerül. Ha (a/00)-ban vagyunk és a hálózat bemenetére 00→01 változás érkezik, akkor a feladat szerint őrizzük a Z=0 kimenet értéket, tehát egy másik sorba, olyan stabil állapotba kell mennünk, ahol Z=0; ezért létrehoztuk és kitöltöttük a (b/01) cellát „b,0” bejegyzéssel. Ha (a/00)-ban vagyunk és a hálózat bemenetére 00→10 változás érkezik, akkor a feladat szerint őrizzük a Z=0 kimenet értéket, tehát egy másik sorba, olyan stabil állapotba kell mennünk, ahol Z=0; ezért létrehoztuk és kitöltöttük a (c/10) cellát „c,0” bejegyzéssel.

DC / y	00 Y,Z	01	11	10
a	**a,0**	b,0	-	c,0
b		**b,0**		
c				**c,0**
…				

Ha a hálózat a (b/01)-ben van és a bemenetre 01→00 változás kerül, akkor visszamehetünk az „a,0” stabil állapotba; 01→11 változás hatására új stabil állapotra van szükség, ezért létrehoztuk és kitöltöttük a (d/11) cellát „d,0” bejegyzéssel.

DC / y	00 Y,Z	01	11	10
a	**a,0**	b,0	-	c,0
b	a,0	**b,0**	d,0	-
c				**c,0**
d			**d,0**	

Ha a hálózat a (c/10)-ban van és a bemenetre 10→11 változás kerül, akkor meg kell változtatnunk a kimenet értékét. Ehhez létrehoztuk és kitöltöttük a (e/11) cellát „e,1” bejegyzéssel. A (c/11) cellájába „e,-„ bejegyzés kerül, mert ezen az instabil állapoton keresztül haladva megváltozik a kimenet.

DC / y	00 Y,Z	01	11	10
a	**a,0**	b,0	-	c,0
b	a,0	**b,0**	d,0	-
c	a,0	-	e,-	**c,0**
d	-	b,0	**d,0**	c,0
e			**e,1**	

Hasonló gondolatmenettel tölthetjük ki a többi cellát is. A kitöltésnek akkor van vége, ha egyetlen további új állapotot sem kellett felvenni és minden korábban felvett sort kitöltöttünk.

Az elmondottak alapján a specifikációnak megfelelő aszinkron előzetes állapottábla:

DC / y	00 Y,Z	01	11	10
a	**a,0**	b,0	-	c,0
b	a,0	**b,0**	d,0	-
c	a,0	-	e,-	**c,0**
d	-	b,0	**d,0**	c,0
e	-	g,1	**e,1**	f,1
f	h,1	-	e,1	**f,1**
g	h,1	**g,1**	e,1	-
h	**h,1**	b,-	-	f,1

Érdemes megfigyelni, hogy a feladat specifikációja miatt minden oszlopban keletkezett két stabil állapot, az egyikben Z=0, a másikban Z=1 értékkel. Az instabil állapotok nagy része ezek között vezeti át a hálózatot. Az első négy sorban Z=0 értéket „őrzünk”, az utolsó négy sorban pedig Z=1 értéket. A két csoport között a (c/11) és a (h/01) cellák teremtenek átjárást.

A tervezés következő lépésében összevonjuk a feleslegesen megkülönböztetett állapotokat. Az állapotösszevonást most intuitívan tesszük, de később mutatunk egy szisztematikus lejárást is erre.

Figyelmesen megnézve az első négy sort, megállapíthatjuk, hogy az a,b,d állapotok összevonhatóak, hiszen ahol specifikáltak, ott azonos bejegyzéseket tartalmaznak. Hasonlóképpen az utolsó négy

állapotból e,f,g állapotok is összevonhatók. Ezen összevonások alapján négy állapottal megvalósíthatjuk a feladatban specifikált működést. Vezessük be az A(abd); B(c), C(efg) és D(h) új állapotokat. Az új táblázat ellentmondás mentes kitöltéséhez nézzük meg az új A állapotba összevonandó bejegyzéseket. Az összevont táblában csak a nagybetűs állapotnevek szerepelhetnek. Az első három oszlopban stabil állapot van előírva azonos osztályon belül, ezért az új tábla (A) sorába „A,0” bejegyzések kerülnek. Az utolsó oszlopban pedig az eredeti c állapot új néven B,0 bejegyzésként szerepel.

a	**a,0**	b,0	-	c,0
b	a,0	**b,0**	d,0	-
d	-	b,0	**d,0**	c,0
A(abd)	**A,0**	**A,0**	**A,0**	B,0

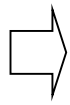
részlet az összevonás magyarázatához

Ezek felhasználásával felírhatjuk az összevont állapottáblát.

DC / y	00 Y,Z	01	11	10
A	**A,0**	**A,0**	**A,0**	B,0
B	A,0	**-**	C,-	**B,0**
C	D,1	**C,1**	**C,1**	**C,1**
D	**D,1**	A,-	**-**	C,1

A tervezés következő lépése az állapotkódolás. Először próbáljuk meg a következő állapotkódokat: A:00, B:01, C:10, D:11. Ennek megfelelően felírhatjuk a kódolt állapottáblát.

	y_1	y_2
A	0	0
B	0	1
C	1	0
D	1	1



DC / y_1y_2	**00**	**01**	**11**	**10**
00	Y_1Y_2,Z **00,0**	**00,0**	**00,0**	01,0
01	00,0	-	10,-	**01,0**
11	**11,1**	00,-	-	10,1
10	11,1	**10,1**	**10,1**	**10,1**

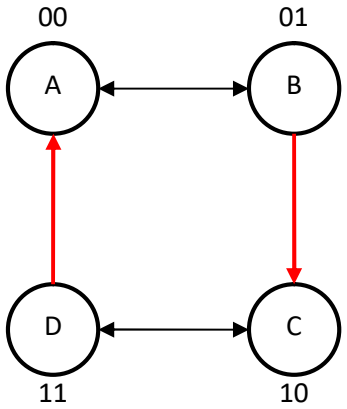
Kódválasztás és a kódhoz tartozó kódolt állapottábla

Kritikus versenyhelyzet

Ahogyan előírtuk a felhasználó számára a szomszédos bemeneti változások használatát, úgy nem árt meggyőződnünk, hogy a hálózaton belül nem alakulhat-e ki olyan versenyhelyzet, amely hibás állapotba viheti a hálózatot. Sajnos a fenti kódválasztásnál két helyen is találunk ilyen pontot, ezeket bekarikázással jelöltük. A 01 sor 10 oszlopából indulva 10→11 szomszédos bemeneti változás hatására mindkét szekunder változónak meg kell változnia (versenyhelyzet). Ha ez a nem szomszédos változás közben kialakul az y1y2=00 kombináció, akkor a hálózat az előírt 10 sor helyett a 00-sor stabil állapotába kerül. Ezt a jelenséget **kritikus versenyhelyzetnek** nevezzük. Fontos megjegyezni, hogy itt nem a versenyhelyzettel van elsősorban problémánk, hanem azzal, hogy emiatt bekerülhet a hálózat egy, az előírástól eltérő állapotba.

A versenyhelyzet megszüntetéséhez rajzoljuk fel a fenti kódolt állapottábla egyszerűsített állapotátmeneti gráfját. Az „egyszerűsített” jelző itt két egyszerűsítést jelent. Egyrészt nem tüntetjük

fel a gráf nyilain sem a bemenetek értékeit, sem a kimenetek értékeit, csak az irányított vonalakkal jelezzük az átmenetet. Másrészt csak azon állapotok közé húzunk nyilakat, amelyek között az átmenet valóban létre is jöhet az aszinkron működés szabályainak betartása során (pl.: szomszédos változásra, stabil állapotból).



Figyeljük meg az egyszerűsített állapotátmeneti gráfon, hogy az élek mentén létrejövő állapotkód változások hol nem szomszédosak. A fenti gráfon a B→C és a D→A átmenet ilyen. Ha felcseréljük a C és D állapot kódját, akkor az élek mentén haladva mindenütt szomszédos átmeneteket kapunk. Ennek megfelelően az új állapotkód és a hozzá tartozó (kritikus versenyhelyzet mentes) kódolt állapottábla az alábbi ábrán látható.

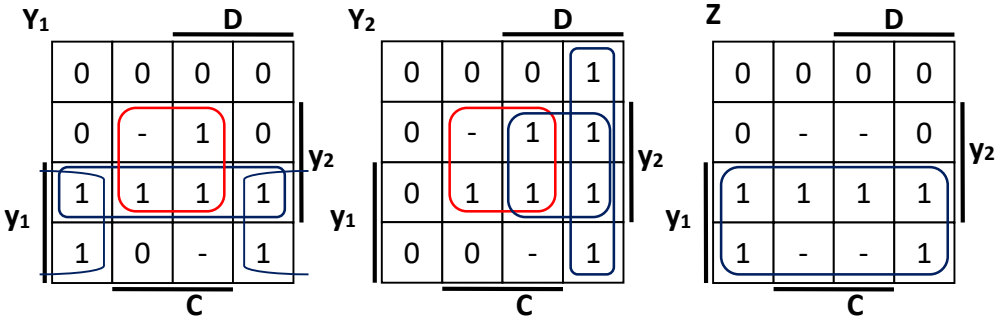
	y_1	y_2
A	0	0
B	0	1
C	1	1
D	1	0

DC / **y_1y_2**	**00**	**01**	**11**	**10**
	Y_1Y_2,Z			
00	00,0	**00,0**	**00,0**	01,0
01	00,0	-	11,-	**01,0**
11	10,1	**11,1**	**11,1**	**11,1**
10	**10,1**	00,-	-	11,1

Kritikus versenyhelyzet mentes állapotkód és a hozzá tartozó kódolt állapottábla

Aszinkron megvalósítás közvetlen visszacsatolással

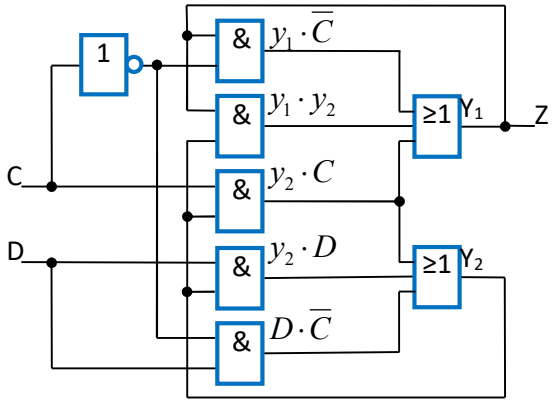
Mivel aszinkron hálózatot tervezünk, azt megvalósíthatjuk közvetlenül visszacsatolt kombinációs hálózattal is. Valósítsuk meg a fenti kódolt állapottáblát visszacsatolt kombinációs hálózattal. Ehhez a kódolt állapottáblát felhasználva három darab négyváltozós függvény (Y_1, Y_2 és Z) legegyszerűbb kétszintű **hazárdmentes** alakját kell előállítanunk. Mivel a kódolt állapottábla peremezése már szomszédos, így nincs más dolgunk, mint az egyes összetartozó oszlopokat átmásolni a megfelelő Karnaugh táblákba.



A grafikus minimalizálás után az alábbi algebrai alakokat kaptuk:

$$Y_1 = y_2 \cdot C + y_1 \cdot y_2 + y_1 \cdot \overline{C}$$
$$Y_2 = y_2 \cdot C + y_2 \cdot D + D \cdot \overline{C}$$
$$Z = y_1$$

Az elvi logikai rajz elkészítése előtt érdemes észrevenni, hogy az Y1 és Y2 függvényekben található egy közös prímimplikáns, amit elegendő egyszer megvalósítani és mindkét függvénynél felhasználni.

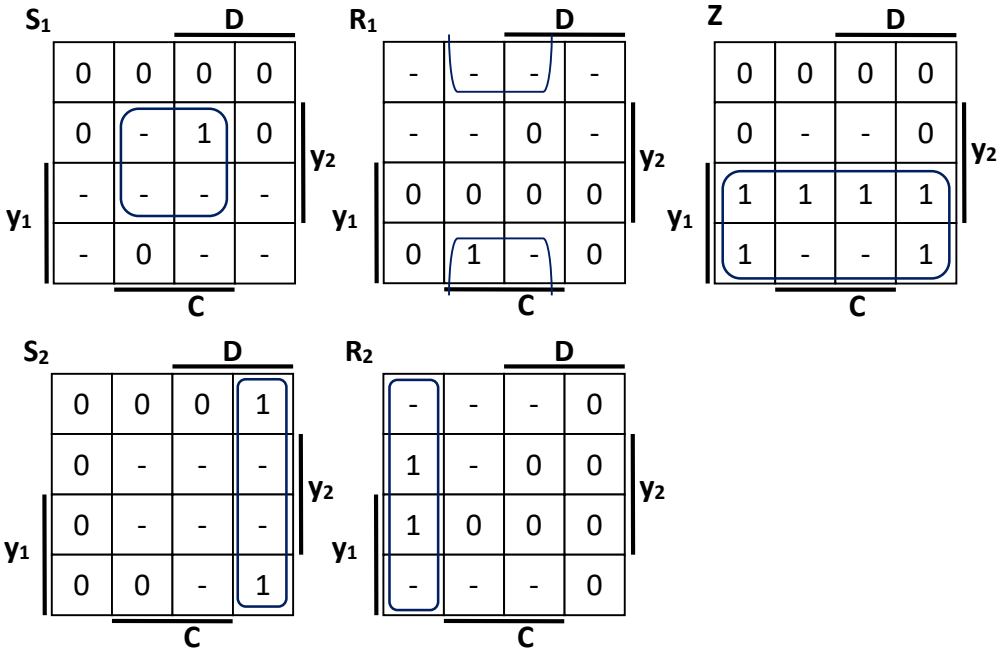


Aszinkron megvalósítás S-R flip-flopok felhasználásával

Hasonlóan a szinkron hálózatokhoz, az aszinkron sorrendi hálózatoknál is alkalmazhatunk flip-flopokat a szekunder változók előállításához. Egyetlen megkötés, hogy aszinkron működésű (órajelet nem igénylő) flip-flopot (két ilyen flip-flopról tanultunk, S-R és D-G) kell alkalmaznunk. Az előző pontban megalkotott kódolt állapottáblához kapcsolódóan készítsük el a vezérlési táblát S-R flip-flop használatához. A könnyebb áttekinthetőség miatt az egyes cellákat függőlegesen egy szaggatott vonallal megfeleztük, a baloldali rész az első-, a jobb oldali rész a második flip-flopra vonatkozó vezérlési előírásokat tartalmazza.

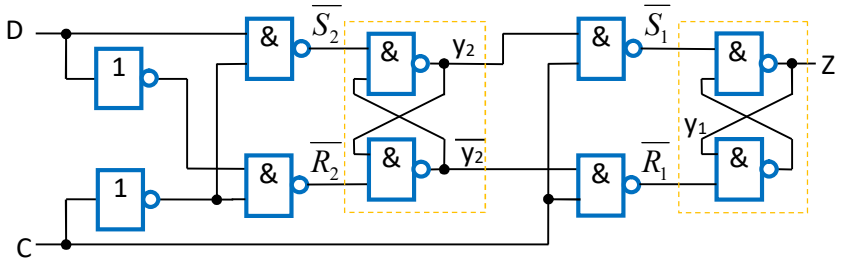
DC / **y_1y_2**	**00**		**01**		**11**		**10**	
	S_1R_1	S_2R_2						
00	0-	0-	0-	0-	0-	0-	0-	10
01	0-	01	--	--	10	-0	0-	-0
11	-0	01	-0	-0	-0	-0	-0	-0
10	-0	0-	01	0-	--	--	-0	10

A kódolt állapottábla és a vezérlési tábla alapján meghatározhatjuk a hálózatot leíró öt logikai függvényt:



$$S_1 = y_2 \cdot C$$
$$R_1 = \overline{y_2} \cdot C$$
$$S_2 = \overline{C} \cdot D$$
$$R_2 = \overline{C} \cdot \overline{D}$$
$$Z = y_1$$

Ábrázoljuk a kapott függvényalakot kizárólag NAND kapuk és inverterek felhasználásával. Az S-R flip-flopokat is a tanult módon NAND kapukkal valósítsuk meg.

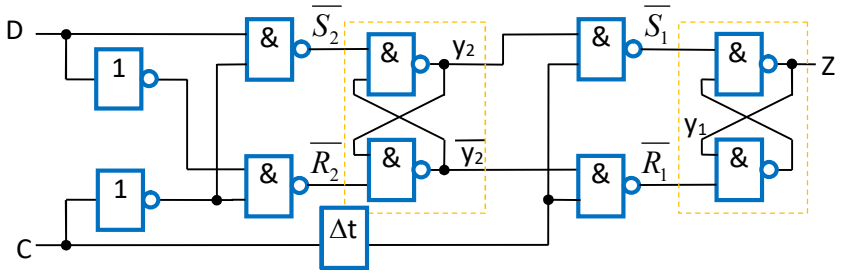


Lényeges hazárd

Az aszinkron sorrendi hálózatokról eddig tanultak alapján kétféle hazárdlehetőséggel találkoztunk. Az egyik a hálózat X bemeneteinek nem szomszédos változásából eredő funkcionális hazárd, melyet a kombinációs hálózatokról szóló részben ismertettünk. Ennek elkerülése érdekében kellett előírnunk, hogy az aszinkron sorrendi hálózat bemenetein kizárólag szomszédos bemeneti változásokat várunk. A másik megismert hazárdot kritikus versenyhelyzetnek neveztük, ekkor a bemeneten fellépő szomszédos változás hatására egynél több szekunderváltozó megváltozása okozta a hibás stabil állapotba kerülést. Ennek kiküszöbölésére megfelelő állapotkód alkalmazását láttuk, de szükség esetén további instabil állapotok beiktatásával is elérhető az előírt működés biztosítása.

Ha egy aszinkron sorrendi hálózatban egynél több szekunder változó van, akkor gondolnunk kell egy olyan versenyhelyzet kialakulására is, melyben a bemeneti változó (X) és valamely szekunder változó (y) versenyez egymással. Helyesen működő esetben a hálózat minden része érzékeli a bemenet megváltozását, majd ezt követően kezdődik a szekunder változó megváltozása. Sajnos a késleltetési viszonyok kedvezőtlen alakulása esetén előfordulhat, hogy az egyik szekunder változót előállító részhálózat már az új y értéket szolgáltatja, miközben a másik részhálózat még nem érzékelte a bemeneti változást (de az új szekunder változót már igen!). Ha emiatt hibás állapotba kerülhet a hálózat, akkor a hálózatban lényeges hazárd van. A lényeges hazárd nem az állapotkódolástól függ, hanem a feladatban „struktúrálisan” található meg, ezért, ha sikerül kimutatni a jelenlétét a hálózat állapottáblájában, akkor a versenyhelyzetben érintett szekunder változókat késleltetni kell olyan mértékben, hogy a hálózat minden részén biztosan előbb kerüljön érzékelésre a bemeneti változók új értéke, mint az (ezen új érték hatására kialakuló) új y értékek.

A leírtak szemléltetésére vegyük elő az előző pontban megtervezett aszinkron D flip-flop elvi logikai rajzát. A lényeges hazárd modellezésére beépítettünk egy késleltetést (Δt) is. A hálózat „bal fele” felel az y_2 változó előállításáért, míg a „jobb fele” felel az y_1 szekunder változóért.



A hazárd fellépését kövessük a hálózat kódolt állapottábláján. Tegyük fel, hogy a hálózat az y_1y_2=00 sor DC=11 oszlopában lévő stabil állapotban van. Adjunk a hálózat bemenetére DC=11→10 szomszédos változást. Előírás szerint ennek hatására az y_1y_2=01 állapotba kell kerülnünk. Sajnos a Δt késleltetés miatt először az y2 szekunder változó vált 1-be, emiatt a hálózat „jobb fele” még úgy érzékeli, hogy a DC=11 oszlopban van. A DC=11 oszlopban viszont az y_1y_2=01 sorban Y_1Y_2=11 van előírva ezért a visszacsatolás gyors működése révén a hálózat átkerül az y_1y_2=11 állapotba, mire észreveszi a DC=10 megváltozott értéket.

DC / y_1y_2	00	01	11	10
00	Y_1Y_2,Z 00,0	**00,0**	**00,0**	01,0
01	00,0	-	11,-	**01,0**
11	10,1	**11,1**	**11,1**	**11,1**
10	**10,1**	00,-	-	11,1

Helyes átmenet

DC / y_1y_2	00	01	11	10
00	Y_1Y_2,Z 00,0	**00,0**	**00,0**	01,0
01	00,0	-	11,-	**01,0**
11	10,1	**11,1**	**11,1**	**11,1**
10	**10,1**	00,-	-	11,1

Hibás átmenet

A probléma kiküszöbölhető, ha az y_2 változót megfelelő mértékben (>Δt) késleltetjük, hiszen ennek túl gyors megváltozása okozta az idő előtti „sor váltást” az állapottáblában. Ilyen késleltetést kialakíthatunk például páros számú (általában 2-nél nagyobb páros számú) inverter beépítésével.

A most bemutatott példában még egy ilyen lényeges hazárdot tartalmazó hely van az y_1y_2=11 sor DC=01→00 váltása során. Ennek kiküszöböléséhez is az y_2 változó késleltetése szükséges.

A lényeges hazárd vizsgálatát magunk is elvégezhetjük az alábbi módon. Kiválasztunk egy stabil induló állapotot, majd végrehajtunk egy szomszédos bemeneti változást a kiválasztott állapot egyik szomszédjának irányába (oda). Megjegyezzük ezt az állapotot, ahová érkeztünk. Ebből az új állapotból visszaváltunk az induló állapothoz tartozó bemeneti kombinációra, melynek hatására ismét új stabil állapotba kerülhetünk (vissza). Ebből az állapotból ismét hajtsuk végre a legelső szomszédos bemeneti változást (oda). Ha ez a stabil állapot, ahová most érkeztünk, és a „megjegyzett” stabil állapot nem azonos, akkor a kiindulási helyen lényeges hazárd van. A most leírt lépéseket az alábbi ábra szemlélteti.

DC / y_1y_2	00	01	11	10
00	00,0	**00,0**	**00,0**	01,0
01	00,0	-	11,-	**01,0**
11	10,1	**11,1**	**11,1**	**11,1**
10	**10,1**	00,-	-	11,1

11→10 „oda”

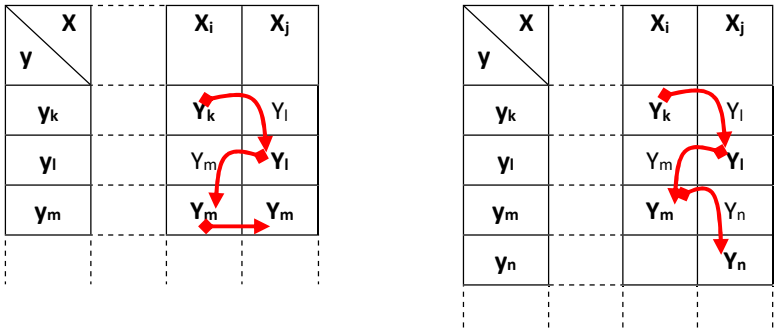
DC / y_1y_2	00	01	11	10
00	00,0	**00,0**	**00,0**	01,0
01	00,0	-	11,-	**01,0**
11	10,1	**11,1**	**11,1**	**11,1**
10	**10,1**	00,-	-	11,1

10→11 „vissza”

DC / y_1y_2	00	01	11	10
00	00,0	**00,0**	**00,0**	01,0
01	00,0	-	11,-	**01,0**
11	10,1	**11,1**	**11,1**	**11,1**
10	**10,1**	00,-	-	11,1

11→10 „oda”

Általánosságban az alábbi állapot elrendeződések ismerhetők fel az állapottáblában lényeges hazárd esetén:



Az állapottáblát tehát lényeges hazárd szempontjából úgy vizsgálhatjuk, hogy minden stabil állapotból kiindulva szomszédos bemeneti változásokat feltételezve ellenőrizzük, hogy nem mutatható-e ki a fenti elrendeződések valamelyike (Unger-tétel). Ha nem találunk ilyen elrendeződést, akkor megfelelő kódválasztással biztosítható a helyes működés. Ha kimutatható a fenti elrendeződések valamelyike, akkor pótlólagos késleltetés beépítése szükséges a visszacsatoló ágakba. Megjegyezzük, hogy a fenti elrendeződések csak normál aszinkron táblák esetén vizsgálhatók. Nem normál működés esetén az instabil állapotokból kiinduló változásokat is vizsgálni kell [3].