



M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Irányítástechnika és Informatika Tanszék

Dr. Pilászy György

# Digitális technika 1

## 10. előadás

(Sorrendi hálózatok tervezése)

Lektorálta: Dr. Horváth Tamás

**Minden jog fenntartva. Jelen könyvet, illetve annak részleteit a szerzők írásbeli engedélye nélkül tilos reprodukálni, adatrögzítő rendszerben tárolni, bármilyen formában vagy eszközzel elektronikus vagy más módon közölni.**

## Szinkron sorrendi hálózat tervezésének lépései

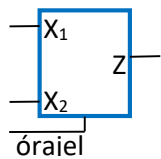
Hasonlóan a kombinációs hálózatokhoz, a szinkron sorrendi hálózatok tervezése is a feladat megfogalmazásával kezdődik. A szöveges leírás alapján általában egy előzetes állapottáblát veszünk fel, mely a szükségesnél több állapotot is tartalmazhat. Ezt követően az állapottábla feleslegesen megkülönböztetett sorait összevonjuk, majd az állapotok számának ismeretében meghatározzuk az állapotok megkülönböztetéséhez szükséges szekunder változók számát és értékét. Ez alapján az összevont állapottáblát állapotkódokkal ellátva létrehozuk a kódolt állapottáblát. A megvalósításhoz szükséges, hogy a szekunder változók előállítására megfelelő flip-flopokat válasszunk. A kiválasztott flip-flopok vezérléséhez a kódolt állapottábla alapján elkészítjük a vezérlési táblát. Ez alapján a már tanult kombinációs függvény minimalizálás felhasználásával elkészítjük az egyes flip-flopok vezérlő hálózatait és a kimeneteket előállító logikai hálózatokat. A jobb áttekinthetőség érdekében pontokba szedve az alábbi lépéseket kel elvégeznünk:

1. Feladat megfogalmazása
2. Előzetes állapottábla felvétele
3. Állapotösszevonás
4. Állapotkódolás
5. Flip-flop választás
6. Vezérlési tábla elkészítése
7. Vezérlési egyenletek előállítása
8. Elvi logikai rajz elkészítése

A következőkben egy konkrét tervezési feladaton keresztül mutatjuk be ezeket a lépéseket, majd külön fejezetekben részletesen foglalkozunk néhány lépéssel.

### Mealy-modell szerint működő soros összeadó tervezése

Tervezzünk Mealy-modell szerint működő szinkron sorrendi hálózatot, amely tetszőlegesen hosszú bináris számok összeadását végzi el az alábbiak szerint. A hálózat  $X_1$  és  $X_2$  bemenetére a legkisebb helyi értéktől kezdődően bináris számok érkeznek az órajellel ütemezetten, a korábban definiált szinkronizációs feltételnek megfelelően. Az összeadás – adott helyi értékhez tartozó -eredményét a hálózat a  $Z$  kimenetén jelenítse meg.



A bináris összeadás során a pillanatnyi helyi értékekhez tartozó  $X_1$  és  $X_2$  számjegyeket össze kell adni. Az összeghez hozzá kell adni a korábbi helyi értéken végzett összeadás során keletkezett átvitelt. Ez lesz a sorrendi hálózatunk állapotváltozója, hiszen ugyanahhoz a bemeneti kombinációhoz a korábbi átviteltől függően különböző kimenetérték tartozhat. Az állapottábla első sorában legyen az átvitel ( $C_y$ ) 0 értékű, a második sorban pedig 1 értékű. Ha tehát az összeadás során átvitel keletkezik, akkor a hálózat a B állapotba kerül. Ha az átvitel ismét 0 lesz, akkor visszakerülünk az A állapotba. Az, hogy éppen melyik oszlopban vagyunk, az adott ütemben érkezett összeadandó számjegyeiktől függ.

Az előzetes állapototábla a következő:

$X_1X_2$ $y$	00	01	11	10
A	A,0	A,1	B,0	A,1
B	A,1	B,0	B,1	B,0

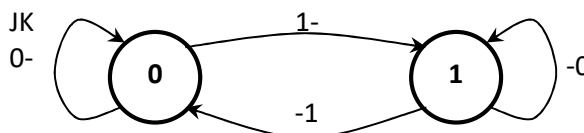
Ez az állapototábla a lehető legkevesebb sort tartalmazza, így további összevonás nem lehetséges. Válasszunk állapotkódot az átvitel értékének megfelelően. Mivel két sorunk van, ehhez egyetlen szekunder változó ( $y$ ) elegendő. A választott állapotkód és a kódolt állapototábla:

Állapot	$y$	leírás
A	0	Átvitel=0
B	1	Átvitel=1

$X_1X_2$ $y$	00	01	11	10
0	Y,Z 0,0	0,1	1,0	0,1
1	0,1	1,0	1,1	1,0

Állapotkód és a hozzá tartozó kódolt állapototábla

A megvalósításhoz válasszunk J-K flip-flopot. A vezérlési tábla előállításához vegyük elő újra a J-K flip-flop állapotgráfját.



A vezérlési tábla kitöltésének menete a következő. A kódolt állapototábla első sorában a szekunder változó ( $y$ ) értéke „0”. Ez megegyezik a megvalósításhoz használni kívánt J-K flip-flop kimenetének pillanatnyi értékével. Ennek megfelelően az állapotgráf első (0) állapotában vagyunk. Ha  $y=0$  mellett a kódolt állapototábla  $Y=0$  értéket ír elő, akkor a vezérlési táblába „0-”, bejegyzést teszünk, ugyanis ez a vezérlés fogja biztosítani, hogy az órajel hatására a korábban „0” kimenetű flip-flop értéke változatlan maradjon. Ha  $y=0$  értéke mellett a kódolt állapototábla  $Y=1$  értéket ír elő, akkor a J-K bemenetekre „1-”, vezérlést kell adnunk, ezért ezt a bejegyzést írjuk a vezérlési tábla megfelelő cellájába. Hasonlóan járunk el a vezérlési tábla második sorának kitöltésekor, azzal a különbséggel, hogy ebben a sorban  $y=1$ , ezért  $Y=0$  előállításához „-1”;  $Y=1$  előállításához „-0” vezérlés szükséges.

A flip-flop használatához szükséges vezérlési tábla:

$X_1X_2$ $y$	00	01	11	10
0	JK 0-	0-	1-	0-
1	-1	-0	-0	-0

A kódolt állapototábla és a vezérlési tábla alapján meghatározhatjuk a hálózatot leíró három logikai függvényt:

J

	X <sub>1</sub>			
	0	0	1	0
y	-	-	-	-
	X <sub>2</sub>			

J = X<sub>1</sub> · X<sub>2</sub>

K

	X <sub>1</sub>			
	-	-	-	-
y	1	0	0	0
	X <sub>2</sub>			

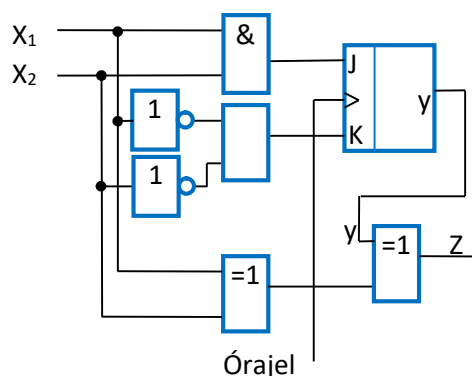
K =  $\overline{X_1} \cdot \overline{X_2}$

Z

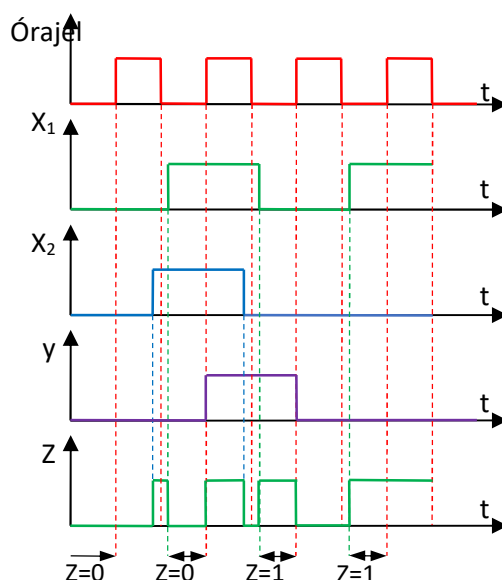
	X <sub>1</sub>			
	0	1	0	1
y	1	0	1	0
	X <sub>2</sub>			

Z = X<sub>1</sub> ⊕ X<sub>2</sub> ⊕ y

Érdekes a kapott eredményt szemügyre venni. A J-K flip-flop vezérlési egyenletei mindössze két ÉS kaput igényelnek két inverterrel. A Z kimenet a korábban bemutatott szimmetrikus függvény  $S^3_{1,3}$ . (A Karnaugh tábla alapján könnyen meggyőződhetünk róla, hogy a Z csak akkor „1”, ha pontosan egy vagy pontosan három bemeneten található „1” logikai érték). A vezérlési egyenletek alapján felrajzolhatjuk a Mealy-modell szerint működő soros összeadó áramkör elvi logikai rajzát.



A kapott hálózat működését az alábbi idődiagramon szemléltetjük. Érdekes megfigyelni, hogy mi történik az egyes órajel éleknél. A diagram megalkotásakor feltételeztük, hogy a késleltetési viszonyok miatt az X1 bemenet az órajel lefutó éle után egy „kicsivel”, míg az X2 bemenet egy „kicsivel” az órajel lefutó éle előtt változik. Példában a 1010 és a 0010 számokat adjuk össze bitsorosan, elsőként a legkisebb helyi értéktől kezdődően.



Az ábra alján bejelöltük a kimenet azon időszávjait, ahol az eredmény érvényes. Érdekes megfigyelni, hogy a hálózat bemenetein fellépő késleltetési problémák miatt a kimeneten rövid pulzusok keletkeznek. A Z kimenet érvényessége emiatt csak az X bemeneti változó állandósult értékétől kezdődően az órajel felfutó éléig tartanak. Az idő többi részében a kimenet értékét nem szabad

felhasználni. Ezt a szinkronizációs feladatot a felhasználónak kell megoldania például pótlólagos flip-flop beépítésével.

## Moore-modell szerint működő soros összeadó tervezése


A Moore-modell szerinti tervezés lényege, hogy úgy veszünk fel állapotokat, hogy azokból a Z kimenet értéke közvetlenül előállítható legyen ( $x_1, x_2$  nem szerepelhet az algebrai alakban). Az előző pontban láttuk, hogy ha egy adott pillanatban megszületik az eredmény a Mealy-modell szerint működő hálózatban, akkor az azonnal megjelenik a Z kimeneten. Moore modell esetén ez a változás csak a soron következő felfutó él hatására fog megjelenni a hálózat kimenetén. Ez azt jelenti, hogy az előzetes állapottábla megalkotásakor nem elég az átvitel alapján megkülönböztetni az állapotokat, hanem minden átvitel értékhez kell, hogy tartozzon  $Z=0$  és  $Z=1$  kimeneti értéket szolgáltató állapot is. Azaz a hálózatnak arra kell "emlékeznie", hogy az előző helyi érték milyen átvitelt eredményezett, és ezen átvitel mellett aktuálisan milyen kimenetet kell szolgáltatni. Vagyis négy állapotunk lesz, A és B állapot 0 értékű átvitelre, C és D állapot 1 értékű átvitelre "emlékezik", különböző kimeneti érték adása mellett.

Az előzetes állapottábla tehát a következő:

$X_1X_2$ y	00	01	11	10
A	A,0	B,0	C,0	B,0
B	A,1	B,1	C,1	B,1
C	B,0	C,0	D,0	C,0
D	B,1	C,1	D,1	C,1

Mivel négy sorunk van, ehhez két szekunder változó ( $y_1$  és  $y_2$ ) elegendő. A választott állapotkód és a kódolt állapottábla:

Állapot	$y_1y_2$	leírás
A	00	Átvitel=0, Z=0
B	01	Átvitel=0, Z=1
C	10	Átvitel=1, Z=0
D	11	Átvitel=1, Z=1



$X_1X_2$ $y_1y_2$	00	01	11	10
00	$Y_1Y_2, Z$ 00,0	01,0	10,0	01,0
01	00,1	01,0	10,1	01,1
11	01,1	10,1	11,1	10,1
10	01,0	10,0	11,0	10,0

Állapotkódok és a hozzá tartozó kódolt állapottábla

A kódolt állapottábla felírásakor az állapotkódokat is szomszédos sorrendben tüntettük fel, így a C és D sorok felcserélődtek. Ennek a sorcserének a tervezés későbbi lépéseinél lesz előnye, könnyebb lesz átírni a vezérlő függvényeket Karnaugh táblába. Az  $y_1$  és  $y_2$  szekunder változók előállításához két flip-flopra lesz szükségünk. Válasszunk mindkét szekunder változóhoz J-K flip-flopot. A továbbiakban jelöljük az  $y_1$  szekunder változóhoz tartozó flip-flop bemeneteit  $J_1, K_1$ -el, az  $y_2$  szekunder változóhoz tartozó flip-flop bemeneteit  $J_2, K_2$ -vel.

A flip-flopok használatához szükséges vezérlési tábla:

$X_1X_2$ $y_1y_2$	00		01		11		10	
	$J_1K_1$	$J_2K_2$						
00	0-	0-	0-	1-	1-	0-	0-	1-
01	0-	-1	0-	-0	1-	-1	0-	-0
11	-1	-0	-0	-1	-0	-0	-0	-1
10	-1	1-	-0	0-	-0	1-	-0	0-

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

The figure displays six 4x4 matrices arranged in a 3x2 grid. Each matrix has a horizontal axis labeled  $X_1$  and a vertical axis labeled  $Y_2$ . The matrices are labeled  $J_1$ ,  $K_1$ ,  $Z$  in the top row, and  $J_2$ ,  $K_2$  in the bottom row. The matrices  $X_1$  and  $X_2$  are also labeled, but their content is not shown. The matrices  $J_1$ ,  $K_1$ ,  $Z$ ,  $J_2$ , and  $K_2$  contain numerical values (0, 1, -) and have blue paths highlighted. The paths in  $J_1$ ,  $K_1$ , and  $Z$  are vertical, while the paths in  $J_2$  and  $K_2$  are more complex, involving both horizontal and vertical steps.

$J_1$	$X_1$	$K_1$	$X_1$	$Z$	$X_1$				
0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	1	1	1	1	1
-	-	-	-	1	1	1	1	1	1
-	-	-	-	0	0	0	0	0	0

$J_2$	$X_1$	$K_2$	$X_1$				
0	1	0	1	-	-	-	-
-	-	-	-	1	0	1	0
-	-	-	-	0	1	0	1
1	0	1	0	-	-	-	-

$$J_1 = X_1 \cdot X_2$$

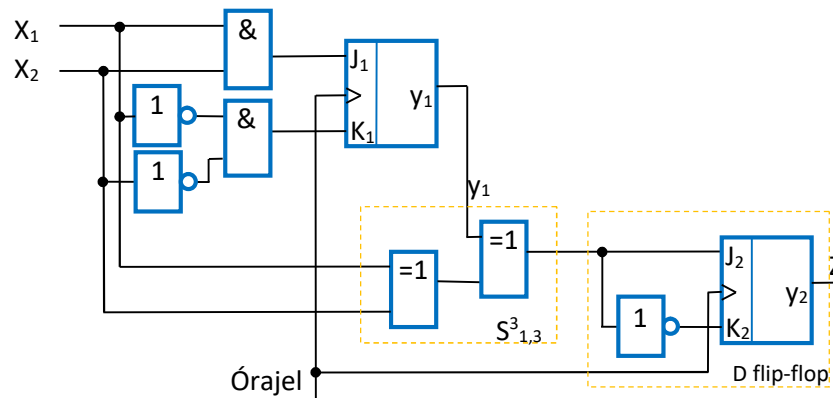
$$K_1 = \overline{X_1} \cdot \overline{X_2}$$

$$J_2 = \overline{X_1} \cdot \overline{X_2} \cdot y_1 + \overline{X_1} \cdot X_2 \cdot \overline{y_1} + X_1 \cdot X_2 \cdot y_1 + X_1 \cdot \overline{X_2} \cdot \overline{y_1} = X_1 \oplus X_2 \oplus y_1$$

$$K_2 = \overline{J_2}$$

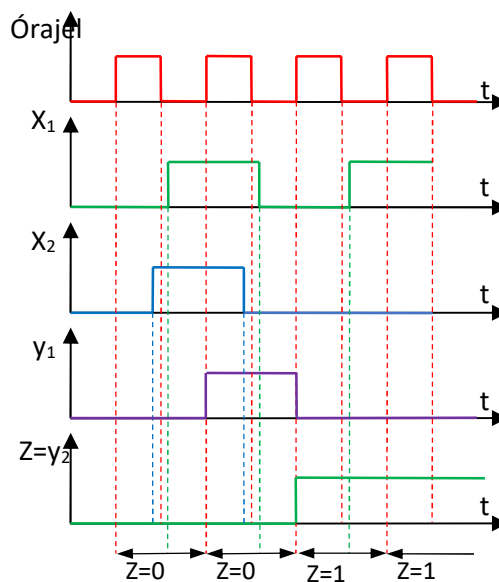
$$Z = y_2$$

A kapott eredményeket érdemes összehasonlítani a Mealy-modell megoldásával. Az első flip-flop vezérlése pontosan ugyanaz. A második flip-flop tulajdonképpen D flip-flopként működik, és mintavételezi az  $S^3_{1,3}$  szimmetrikus függvény kimenetét.



Moore-modell szerinti bitsoros összeadó egység

Az időbeli működést szemlélteti az alábbi ábra.

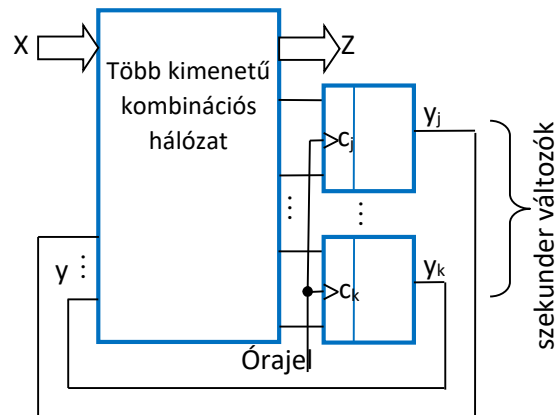


Megállapíthatjuk, hogy ezzel tulajdonképpen megoldja a kimenet értelmezésének korábban említett problémáját, hiszen a flip-flop kimenetére nincs hatással a bemeneten fellépő „tüske”. A Moore modell adott X bemenetre adott kimeneti értéke a soron következő felfutó éltől kezdve érvényes.



## Szinkron sorrendi hálózat általános struktúrája

Az eddigi példák alapján felrajzolhatjuk a szinkron sorrendi hálózatok általános struktúráját. Ennek lényege, hogy a sorrendi hálózat „lelke” egy több bemenetű, több kimenetű kombinációs hálózat. A kimenetek egy része szolgáltatja a Z értékeket, a többiek pedig vezérlik a szekunder változókat megvalósító flip-flopokat. Ezek a flip-flopok nem feltétlenül azonos típusúak. A szinkron működéshez szükséges órajelet mindegyik flip-flop megkapja. D flip-flop alkalmazása esetén az Y értéket kell a kombinációs hálózatnak előállítania, D-től eltérő flip-flopok esetén pedig olyan vezérlést kell megvalósítani, ami az előírt Y szerint működteti a kiválasztott flip-flopot.

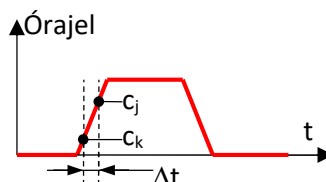


## Szinkronizációs feltétel biztosítása szinkron sorrendi hálózatokban

Az előző pontban bemutatott modellben bármilyen szinkron flip-flopot használhatunk a szekunder változók előállítására. A flip-flopok ismertetése során definiált mintavevő tartó áramkör órajel szerinti működését úgy definiáltuk, hogy az órajel  $0 \rightarrow 1$  váltásának pillanatában fennálló bemeneti érték szerint állítsa be a kimenetét. Ezt a működési módot egyszerű **élvezérelt, felfutóél-vezérelt** működésnek hívjuk. Amennyiben szükséges, definiálhatjuk a **lefutóél-vezérelt** működést is, amely az órajel  $1 \rightarrow 0$  átmeneteinek pillanatában teszi mindezt.

## Rendszerhazárd

Most vizsgáljuk meg azt, hogy mi történik, ha egynél több flip-flop órajel bemenetére „ugyanazt” az órajelet vezetjük. Egy valódi áramkörben az órajel nem tud „végtelen” meredekséggel változni. Az egyes kapuáramkörök eltérő feszültségszinten komparálnak (valahol a „tiltott” sávon belül). Az órajelet továbbító vezetékezés is lehet eltérő jelkésleltetési idejű (pl.: különböző vezeték hossz, eltérő szórt kapacitások, induktivitások, stb.). Az alábbi ábra szemlélteti az eltérő feszültség szintek érzékeléséből eredő késleltetést.

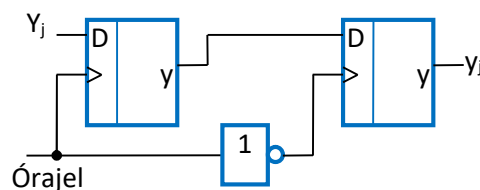


Sajnos ezekből a késleltetésekből eredően előfordulhat, hogy az órajel felfutó élének pillanatában az egyik szekunder változó megváltozik, majd ez az új érték visszahat a sorrendi hálózat bemenetére, ahol új Y kombinációt hoz létre úgy, hogy egy másik flip-flop már ezt az új értéket mintavételezi. Ez sajnos hazárdjelenségként viselkedik, hiszen a mintavételezés pontos időpontja a változó késleltetési viszonyoktól függ. Ezt a hazárdjelenséget **rendszerhazárddnak** nevezzük.

A rendszerhazárd kiküszöbölésére módosítanunk kell a visszacsatolásban alkalmazott mintavevő és tartó áramkör órajel szerinti viselkedését. Elsőként próbálunk meg olyan viselkedést előírni, amely az órajel „1” értéke alatt mintavételezi a bemenetét, majd a lefutó élre változtatja meg a kimenetét. A mintavétel alatt a bemenet nem változhat. Az ilyen működésű flip-flopokat főként a belső felépítésük miatt **master-slave** működésű flip-flopoknak nevezik. Az ilyen felépítésű flip-flopokban két tároló fokozat működik egymás után, az első folyamatosan működik az órajel „1” értéke alatt, a második pedig a lefutóél hatására átmásolja az első fokozat kimenetét. Ezzel a felfutóél pillanatában bekövetkező változások látszólag nem okoznak zavart. Sajnos a megoldás még nem teljesen jó, mert az órajel „1” értéke alatt bekövetkező változások befolyásolhatják a kimenet működését, így mégis zavart okozhatnak. (Részletesen nem foglalkozunk a master-slave építő elemek belsejével).

Ha a mintavétel az órajel felfutó élére történik, de a minta megjelenítése az  $y$  kimeneteken csak a lefutóélre, akkor egészen biztosak lehetünk abban, hogy a mintavételezéskor nem fognak változni a hálózat vezérlő kimenetei. Az ilyen órajel szerinti viselkedést **reteszelt** működésű vagy **data-lock-out** működésnek nevezzük.

Ezt a működést könnyen megvalósíthatjuk a korábban megtervezett felfutóél-vezérelt flip-flopok felhasználásával. Az alábbi ábrán egy data-lock-out működésű D flip-flop kialakítása látható. A reteszelt működést úgy érjük el, hogy beépítünk egy második flip-flopot, amely az invertált órajel miatt lefutóélre mintavételezi az előtte lévő flip-flop kimenetét. Így teljesül az előírt működés, felfutóélre az első flip-flop vesz mintát, lefutóra pedig ezt a mintát mintavételezzük tovább és jelenítjük meg a kimeneten.



Reteszelt működés megvalósítása két felfutóél-vezérelt flip-flopból

## A flip-flopok órajel szerinti viselkedésének összefoglalása

Röviden összefoglaljuk az eddig megismert flip-flopok órajel szerinti viselkedése szempontjából fontos elnevezéseket és működési módokat. Az egyes működési módok abban különböznek, hogy mikor mintavételezik a bemenetüket, illetve mikor változtatják meg a kimenetüket a flip-flopra jellemző működésnek megfelelően.

Az alábbi táblázatban eszerint tüntetjük fel az órajel szerinti viselkedéseket.

Működés	Bemenet mintavételezése	Kimenet beállítása	Szemléltető ábra
<b>felfutóél-vezérelt</b>	Órajel 0→1	Órajel 0→1	
<b>lefutóél-vezérelt</b>	Órajel 1→0	Órajel 1→0	
<b>Master-slave</b>	Órajel = "1"	Órajel 1→0	
<b>Data-lock-out</b>	Órajel 0→1	Órajel 1→0	