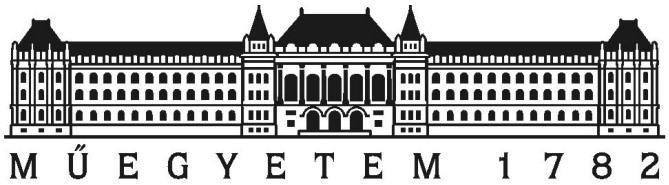


Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Dr. Pilászy György

Digitális technika 1

13. előadás

(Állapotkódolás)

Lektorálta: Dr. Horváth Tamás

Állapotkódolás

A sorrendi hálózatok tervezésének következő lépése az állapotkódolás. Az állapotkódolás során az összevont állapottábla egyes állapotaihoz konkrét bináris kombinációkat rendelünk. Szintén az állapotkódolás során kell meghatároznunk a szükséges állapotváltozók darabszámát is. A legkevesebb szekunder változó számát (k) az alábbi összefüggéssel határozhatjuk meg az összevont állapottábla sorainak száma alapján (n):

$$k = \lceil \log_2(n) \rceil$$

A képletben szereplő $\lceil \ \rceil$ operátor a logaritmus eredményének felfele kerekítését jelöli.

Bizonyos programozható logikai eszközökben találkozhatunk olyan igénnyel, hogy az állapottábla minden állapotához külön szekunder változó tartozzon. Ehhez n darab szekunder változó szükséges és minden pillanatban legfeljebb egy változó lehet „1” értékű.

Moore modell szerint működő hálózatoknál lehet az is elvárás, hogy a flip-flopok kimenete közvetlenül (további kapuáramkörök nélkül) szolgáltassa a kimeneti kombinációt. Gondoljunk például a később ismertetésre kerülő számláló/léptető funkciót megvalósító sorrendi hálózatokra.

Szinkron hálózatok esetében tetszőlegesen választhatunk különböző kódokat, a szinkronizációs feltétel betartása esetén működő hálózatot kapunk. Nem mindegy azonban, hogy milyen költségű lesz az adott kódoláshoz tartozó vezérlő hálózat. A hálózat egyszerűsége szempontjából legkedvezőbb kódolást csak akkor tudnánk megkapni, ha az összes lehetséges kódválasztással elvégeznénk a megvalósítást, majd a kapott megoldásokból kiválasztanánk a legkedvezőbbet [3]. Jelölje n az összevont állapottábla állapotainak számát, p a felhasználni kívánt szekunder változók számát ($p \geq k$). Az összes különböző szekunder változó kombinációk száma: 2^p. Ennyi lehetséges kódból kell n darabot kiválasztanunk. A kiválasztott n darab kódot tetszőleges sorrendben rendelhetjük az n darab állapothoz. A kiválasztási lehetőségek száma: $\binom{2^p}{n}$, ezek közül minden kiválasztás n! sorrendben szerepelhet, így a lehetséges kódolások száma: $N = \binom{2^p}{n} \cdot n! = \frac{2^p!}{(2^p - n)! \cdot n!} \cdot n! = \frac{2^p!}{(2^p - n)!}$

A hálózat bonyolultsága szempontjából nem jelentenek különböző kódolást azok, melyek a szekunder változók átcsoportosításával (felcserélésével) vagy invertálásával egymásból előállíthatók. Ezek csak az elnevezéseken változtatnak, a vezérlési függvények struktúráján nem, ezért érdemes a fenti összefüggést tovább finomítani az egyszerűsítő hatás szempontjából megkülönböztetendő kódolások számának meghatározásával (N_k). Invertálással 2^p számú különböző kombináció képezhető, felcseréléssel p! variációs lehetőségünk van. Ezeket figyelembe véve:

$$N_k = \frac{N}{2^p \cdot p!} = \frac{\frac{2^p!}{(2^p - n)!}}{2^p \cdot p!} = \frac{(2^p - 1)!}{(2^p - n)! \cdot p!}$$

N_k értékét néhány jellegzetes esetre az alábbi táblázatban foglaltuk össze.

n	p	N_k
2	1	1
4	2	3
5	3	140
8	3	840
9	4	10 810 800
16	4	54 486 432 000

A következőkben kifejezetten szinkron működésű sorrendi hálózatok állapotkódolásához mutatunk be szisztematikus eljárást. Az eljárás szisztematikus volta sajnos nem garantálja, hogy a végeredményként kapott megoldás optimális, de azt reméljük, hogy valamilyen szempontból optimális megoldást kapunk.

Szomszédos kódolás

A módszer alapgondolata, hogy megnézzük, milyen állapotátmenetek fordulnak elő gyakran az összevont állapottáblában. A leggyakrabban előforduló átmeneteket szomszédos kódokkal ellátva a megvalósítás során szomszédos bejegyzések keletkeznek a Y_i logikai függvényekben, ami segíti a vezérlő függvények minimalizálását. A kétféle vizsgálandó szomszédossági előírás:

a) legyenek páronként szomszédosak azon állapotok kódjai, amelyeknek azonos következő állapotuk van valamilyen azonos bemeneti kombináció mellett.

b) Legyenek páronként szomszédosak azon állapotok kódjai, amelyek ugyanannak a pillanatnyi állapotnak a következő állapotai (szomszédos bemeneti változásra).

Ha egy adott állapottáblában a két követelmény egyidejű teljesítése ellentmondáshoz vezetne, akkor ajánlatos az a) pontot előnyben részesíteni, mert ilyenkor nagyobb egyszerűsítő hatást várhatunk. A kétféle előírás szerint készítsünk összesítést, hogy melyik állapotpárra hány előírás adódott. Az előfordulások nyilvántartására használhatjuk az állapotösszevonás során megismert lépcsős táblát is. Elsőként a legtöbb előírást kapott állapotokhoz választunk szomszédos kódokat. A kódválasztást a még szabad kódokkal folytatjuk mindaddig, míg valamennyi állapothoz nem sikerült kódot rendelni.

A módszer szemléltetésére válasszunk állapotkódot az alábbi összevont állapottáblához.

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

Az a) szabály szerinti szomszédossági előírások:

Következő állapot	Kivel legyen szomszédos
A	ABCDE
B	ABDE
C	ABC
D	BD,DE
E	ACE

A b) szabály szerinti szomszédossági előírások:

Pillanatnyi állapot	Kivel legyen szomszédos
A	AB, BE, CE, AC
B	AB,BD,CD,AC
C	AD,DE,CE,AC
D	AB,BD,AD
E	AB,BE,DE,AD

A különböző állapotpárokra vonatkozó előírások összesítése:

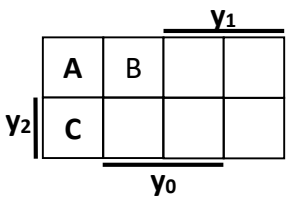
	a)	b)						
B	3	4						
C	3	3	2	0				
D	2	3	3	2	1	1		
E	3	0	2	2	2	2	3	2
	A		B		C		D	

Mivel öt állapotunk van, ezért legalább három szekunderváltozóra lesz szükségünk. A szomszédos kódok keresésére kitűnően alkalmazhatjuk a logikai függvények minimalizálása során megismert Karnaugh táblázatot, ha a szekunder változók szerint peremezzük és az állapotok betűjeleit a cellákba írjuk, majd a peremezésről leolvassuk az állapotokhoz tartozó kódokat.

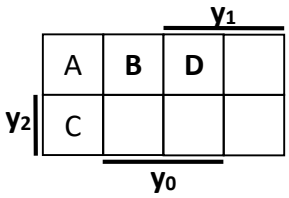
A táblázat alapján a legtöbbször előforduló állapotpár az AB, őket fogjuk először kódolni. Mivel még a teljes tábla üres, bárhová helyezhetjük az AB párt, csak egymás mellé kerüljenek.

	y_1	
y_2	**A** **B**	
		y_0

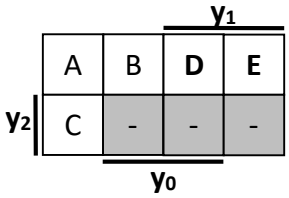
A következő „népszerű” állapotpár az AC. Mivel az A kódja már rögzítésre került, már csak két szabad szomszédja van, azok közül bármelyikbe beírhatjuk C-t. Ezzel AC kódválasztásával elkészültünk.



A „harmadik” helyen két párunk van, a BD és ED, mindkettő ugyanannyiszor fordul elő a feltételekben. Mivel a B kódja már rögzített, ezért célszerű a D állapotot melléje helyezni.



Végül az ED feltételt teljesítjük.



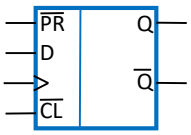
Mivel minden állapothoz sikerült kódot rendelni, készen vagyunk, felírhatjuk a kódolt állapottáblát. Látható, hogy a lépcsőstáblában szereplő összes szomszédossági igényt nem sikerült teljesíteni, de a leggyakoribbakat igen. A három szekunder változó felhasználása miatt maradt még három fel nem használt állapotkód. A kódolt állapottáblában ezeket a sorokat közömbös bejegyzéssel jelölhetjük, ha a feladat nem tartalmaz más előírást.

X_1X_2 / $y_2y_1y_0$	**00**	**01**	**11**	**10**
A 000	000,0	001,1	010,0	100,1
B 001	000,0	001,1	011,0	100,1
D 011	000,0	001,1	011,0	011,0
E 010	000,0	001,0	010,0	011,0
C 100	000,0	011,0	010,0	100,1
101	-	-	-	-
111	-	-	-	-
110	-	-	-	-

Sorrendi hálózatok alaphelyzetbe állítása

Az eddigiek alapján láttuk, hogy szinkron sorrendi hálózatok visszacsatoló ágaiban bármilyen szinkron működésű flip-flop használható. Azzal viszont nem foglalkoztunk, hogy egy berendezés bekapcsolását követően hogyan biztosíthatjuk az előírt működéshez szükséges kezdőállapotot. A legtöbb esetben a tervezendő eszközt ellátjuk egy plusz külső (reset) bemenettel, amely a teljes rendszer alaphelyzetbe állításáért felel. A logikai feladatban meghatározott kezdő állapot kódja – az állapotkódolás során választott kódtól függően - tetszőleges értékű lehet, ezért minden egyes flip-flip esetében biztosítanunk kell, hogy a külső reset jel hatására az előírt 0- vagy 1 értékre álljanak.

Az alaphelyzetbe állítás működhet aszinkron és működhet szinkron módon is a specifikációtól és a választott flip-flop tulajdonságaitól függően. **Aszinkron** működés esetén a reset jel aktív állapotba állítása során a lehető leggyorsabban, más jelektől függetlenül megtörténik a flip-flopok beállítása. Ehhez a flip-flop áramköröket külön beállító bemenetekkel látják el. pl.: SN7474 D-flip-flop esetében a korábban megismert D és órajel bemeneteken kívül a flip-flop rendelkezik még egy alacsony aktív clear és egy alacsony aktív preset bemenetel is [6]. Az „alacsony aktív” elnevezés azt jelenti, hogy logikai alacsony szintre kell kapcsolni a bemenetet a nevében adott funkció végrehajtásához. Az alábbi ábrán az SN7474 típusú D-flip-flop katalógusa alapján készített rajzszimbólumot látjuk. Az áramkör rendelkezik egy $\overline{PR}$ és egy $\overline{CL}$ aszinkron beállító bemenettel, valamint ponált (Q) és negált ($\overline{Q}$) kimenetel is. Amennyiben „alacsony” szintre kapcsoljuk a $\overline{PR}$ bemenetét, akkor azonnal Q=1-be áll és ott marad. Hasonlóan a $\overline{CL}$ bemenet „alacsony” szintre kapcsolása Q=0 értéket eredményez. A normál szinkron működés során mindkét beállító bemeneten „magas” logikai szintet kell biztosítani, különben felülbírálja a flip-flop állapotát.



Szinkron alaphelyzetbe állítás a flip-flopok bemenetére épített vezérlő hálózat segítségével biztosítható. A megfelelő alaphelyzetbe állításhoz először aktiválni kell a reset jelet, majd biztosítani kell, hogy legalább egy órajel ideig fennálljon. A vezérlő hálózatot a flip-flop típusához és az alaphelyzetnek megfelelő beállítandó értékhez kell igazítani.

Az alábbiakban D és J-K flip-flopok „0” és „1” kezdőértékének szinkron beállítását láthatjuk Reset=1 / Preset=1 vezérlés hatására. Reset=0 érték mellett a pótlólagosan beépített logikai kapuk nem változtatnak a vezérlés eredeti működésén.