Állapotkódolás

A minimalizált állapottábla elkészítése után a következő lépés az állapotkódolás. Az automatában az állapotokat digitális kódjukkal reprezentáljuk. Minimálisan az állapotok számától (N) függő számú (n) biten kell az állapotokat kódolni, hogy minden állapothoz különböző kódot tudjunk rendelni.

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

Az állapotkód bitejeit a flip-flopok állítják elő, így minimálisan ennyi flip-flop (ekkora méretű állapotregiszter) szükséges.
Az állapokód hozzárendelés jelentősen befolyásolhatja az automata költségét.
Hogy mi jelent költséget (mivel célszerű spórolni), az a megvalósítási környezettől függ. A ma már elavult SSI elemekből (kapuk és flip-flopok) építkezve, mind a kapukkal, mind a flip-flopokkal spórolni kellett. A programozható logikák közül a PLD, CPLD-ben viszonylag bőven van sok bemenetű kombinációs logika, de relatíve kevés a flip-flop. Így ezeknél a flip-flopok számával célszerű spórolni. Az FPGA-k regiszterben gazdagok, viszont a kombinációs logika kevés bemenetű, így ott az utóbbival kell spórolni. Ha minimális regiszterszám megkötéssel kódoljuk az

állapotokat, akkor a flip-flopok vezérlő függvénye általában bonyolultabb, több bemenetű lesz, míg a regiszterekkel "pazarlóan" bánnó kódolások esetén többnyire egyszerűbb, kevesebb bemenetszámú vezérlő függvényt kapunk. Tehát a kódolási algoritmust a megvalósítási környezethez kell megválasztani.

Az állapotkódolásra szisztematikus algoritmusok léteznek. Ezekből mutatunk be néhányat.

Először minimális regiszterszám megkötéssel történő kódolási algoritmusokat tekintünk át.

Valaki mondhatná, hogy felesleges kódolási algoritmusokon törni a fejünket, hiszen ma már a számítógép segítségével megtehetjük, hogy az összes lehetséges módon hozzárendeljük az állapotkódokat az állapotokhoz, minden verzióhoz megtervezzük a hálózatot és a legegyszerűbbet valósítjuk meg.

Sajnos ez nem megy, ugyanis ha n biten M állapotot kódolunk, a lényegesen különböző állapotkódolások száma:

$$\frac{\begin{bmatrix} 2^n \\ M \end{bmatrix} M!}{n! 2^n}$$

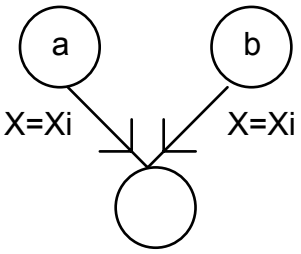
Ez a szám pl. M=6 állapot és n=3 bit esetén 420.

A gyakorlatban előforduló állapotszámok esetén ez kivárhatatlan számítási időt eredményezhet. Ezért szisztematikus kódolási algoritmusokat használunk.

Szomszédos kódolás szinkron hálózatok esetén

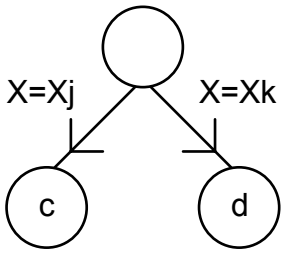
Ez az algoritmus ún. heurisztikus módszer. Nem kimerítő vizsgálat alapján talál egy optimumot, hanem bizonyos szabályokat fogalmaz meg, melyeket ha betartunk, valószínűleg sokkal jobb eredményt kapunk, mintha véletlenszerűen választanánk állapotkódot.

A szabályok:
a. Legyen szomszédos azon állapotok kódja, amelyeknek ugyanaz a következő állapota, valamely azonos Xi bemeneti kombinációra.



(Legyen szomszédos *a* és *b* kódja.)

b. Legyen szomszédos azon állapotok kódja, amelyek ugyanazon állapotnak a következő állapotai.

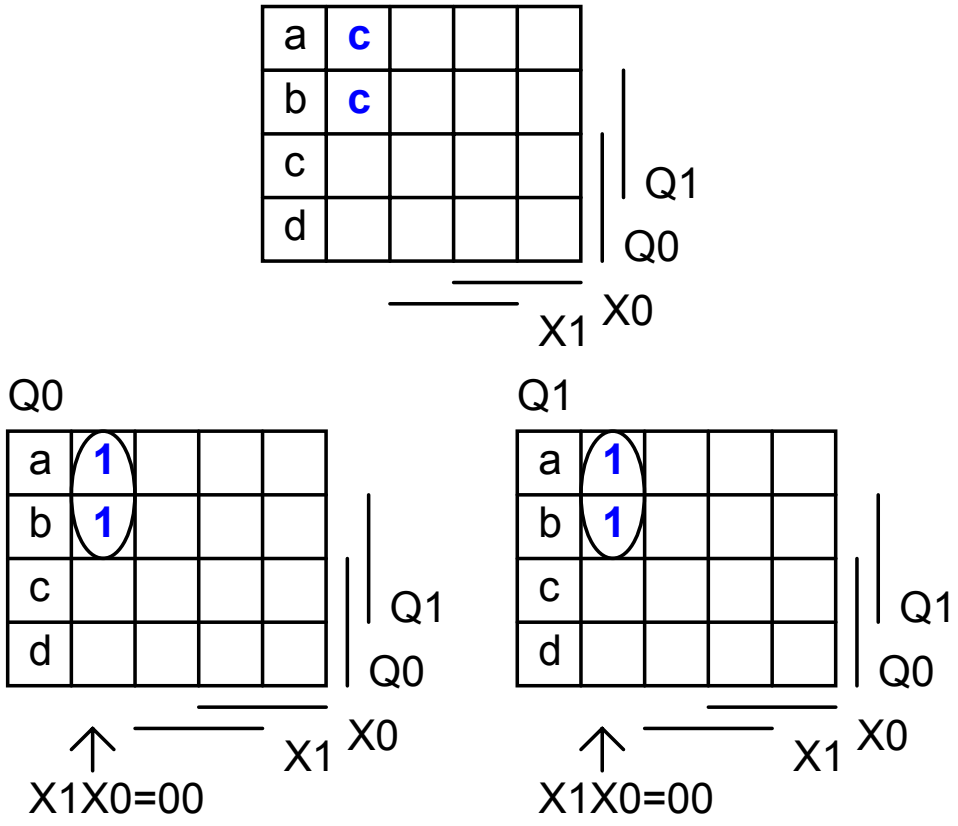


(Legyen szomszédos *c* és *d* kódja.)

c. Az *a* szabály az erősebb.

Miért okozhat egyszerűsítést az "a" szabály?

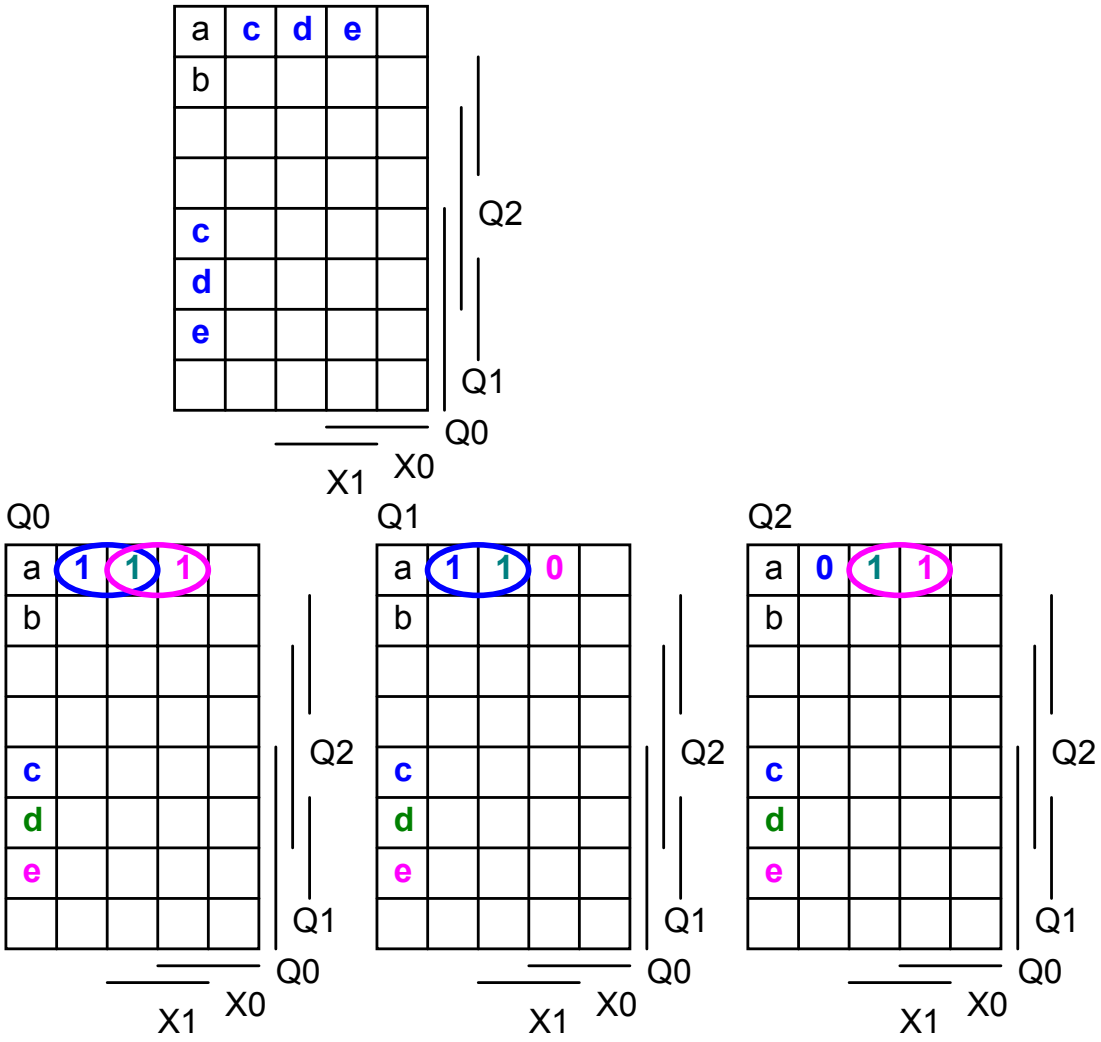
Az alábbi állapottábla részletben az *a* és *b* állapoból az X1X0=11 bemeneti kombinációra a *c* állapot következik. Ezért az *a* és *b* kódját szomszédosra választottuk (*a:*Q0Q1=00) és (*b*:Q0Q1=01). Ezért a c állapot (és a kódbitjei) a kódolt állapottáblán szomszédos rubrikákba kerültek, hiszen a bemenet azonos, az állapotkód pedig 1 bitben különbözik.



Mivel a rákövetkező állapot kódjában levő 1-esek (0-ák) szomszédos rubrikákba esenek ezért ***összevonási lehetőség adódik***, mely során ***szekunder változó esik*** ki. (A példában Q1esik ki.)

Ha a rákövetkező állapot kódjában több 1-es (0) van (a példában C kódja Q0Q1=11), azok több Karnaugh táblában azonos helyre eshetnek, vagyis ***közös prímimplikánsok alakulhatnak ki***, amelyeket csak egyszer kell megvalósítani. (Az előbbi példában Q1 és Q0 Karnaugh táblájában /X1./X0./Q0 közös prímimplikáns). Az *a* szabály tehát egyszerre kétféle egyszerűsödéshez vezethet.

Milyen egyszerűsítést hozhat a "b" szabály?



A fenti állapottáblán az *a* állapotra rákövetkező X1X0=00-ra *c*, 10-ra *d,* 11-re *e*. Ezért a kódjukat a következő módon szomszédosra választottuk:
c:Q0Q1Q2=110, d:Q0Q1Q2=111 e:Q0Q1Q2=101

A szekunder változók szoszédossága miatt a különböző szekunder változók Karnaugh táblái azonos szekunder változó kombinációkhoz tartozó (Q2Q1Q0) rubrikáiba sok azonos 1 (0) érték kerül. Ezek között jó eséllyel lesznek szomszédos bemeneti kombinációkhoz tartozó értékek is. Így, ha az azonos szekunder változó kombinációhoz és szomszédos bemeneti kombinációhoz tartozó bejegyzés 1 (0), akkor ***összevonási lehetőség adódhat, és bemeneti változó eshet ki***. (A példa Karnaugh tábláján kékkel /Q0/Q1/Q2X0 ill. pirossal /Q0/Q1/Q2X1 jelölt összevonások. Az előbbiből X1, utóbbiból X0 esett ki.)
Ráadásul esély van arra is, hogy az összevonások során keletkezett prímimplikánsok közösek legyenek a különbözö szekunder változók Karnaugh tábláin. A ***közös prímimplikánsok***at csak egyszer kell megvalósítani. (A példában Q0 és Q1-nek /Q0/Q1/Q2X0 ill. Q1 és Q2-nek /Q0/Q1/Q2X1 lett a közös prímimplikánsa.)

Mivel az *a* szabály biztosabban vezet előnyhöz, ezért elsősorban ezt célszerű kielégíteni.

Mintapéldaképpen kódoljuk most szomszédosan a szinkron hálózatok bevezetőjében véletlenszerűen kódolt állapottáblát.

	x=0	x=1
A	B/0	C/0
B	D/0	E/0
C	E/0	F/0
D	**A**/0	**A**/0
E	**A**/0	**A**/1
F	**A**/1	**A**/0

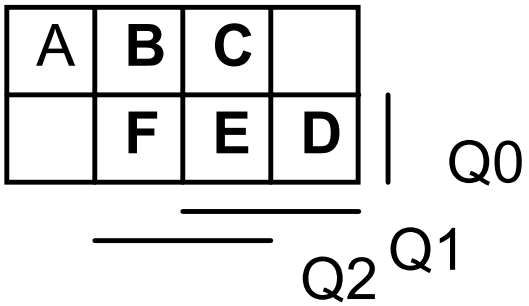
Az *a* szabály alapján kielégítendő szomszédossági feltételek:
Itt az azonos bemeneti kombinációkhoz tartozó (egy oszlopban levő) azonos következő állapotok soraiban levő kiinduló állpotokat (legelső oszlopban levő állapotok) kell összegyűjteni. Pl. a 0 bemeneti kombináció oszlopában a A következő állapot kiinduló állapotai **D,E,F**. Itt ugyanez adódik az 1 bementre is. Többet nem találunk, ezért az a szabály alapján az összes szomszédossági igény: **DEF** vagyis **DE, DF, EF**.

A *b* szabály alapján kielégítendő szomszédossági igények:
Itt a kiinduló állapot sorában levő állapotokat kell összegyűjteni.

	x=0	x=1
A	**B**/0	**C**/0
B	**D**/0	**E**/0
C	**E**/0	**F**/0
D	A/0	A/0
E	A/0	A/1
F	A/1	A/0

Legyen szomszédos: **BC**, **DE**, **EF**

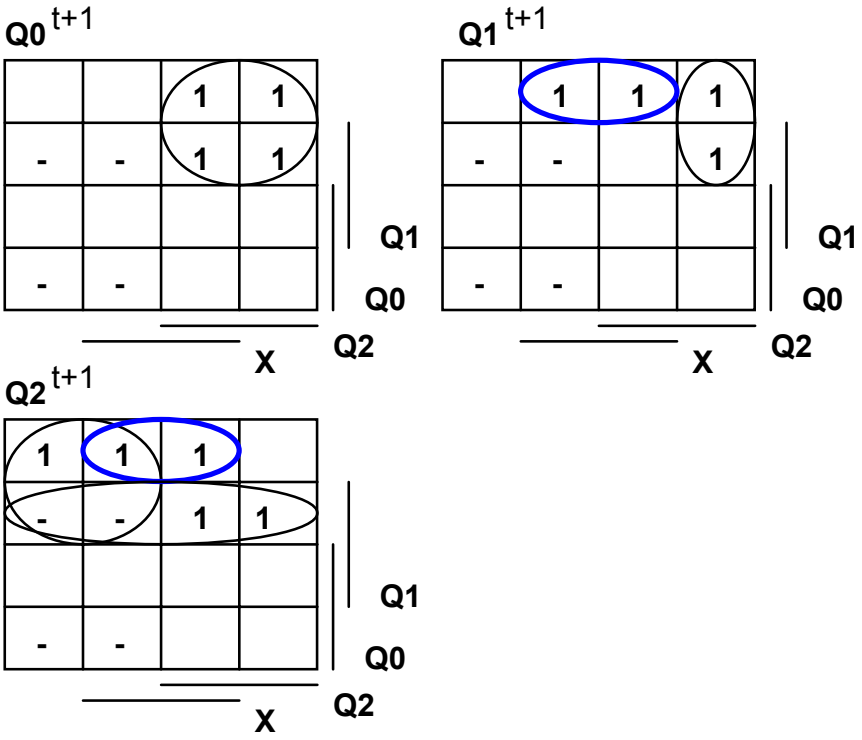
A szomszédossági feltételek kielégítéséhez kézi módszerként a Karnaugh táblát használjuk fel, mivel abban a szomszédosság “szemre” jól látható.
Ebbe a táblába próbálunk úgy szomszédosan elhelyezni az állapotokat, hogy mennél több *a* szabálybeli DE, DF, EF, majd mennél több *b* szabálybeli BC, (DE), (EF) követelmény teljesüljön.



Az *a* szabály szerint szomszédos kellene hogy legyen:
<u>de</u>, df, <u>ef</u>
Ebből az aláhúzottak teljesültek. Mindhármat egyszerre 3 szekunder változó esetén nem lehet teljesíteni.
A *b* szabály szerint szomszédos kellene hogy legyen:
<u>bc</u>, <u>de</u>, <u>ef</u>
Ebből az aláhúzottak teljesültek, vagyis mind.
Az A állapot nem szerepel a szomszédossági követelményekben, ezért tetszőlegesen kódolható.
Sok azonosan jó megoldás lehetséges.
Az egyes állapotok kódja a táblából Q2Q1Q0 sorrendben kilovasva:
A 000, B 100, C 110, D 011, E 111, F 101

A kódolt állapottábla:

Q2Q1Q0	x=0	x=1
A 000	100/0	110/0
B 100	011/0	111/0
C 110	111/0	101/0
D 011	000/0	000/0
E 111	000/0	000/1
F 101	000/1	000/0
001	---/-	---/-
010	---/-	---/-



Q0= /Q0.Q2

Q1=**X./Q0./Q1** + /X./Q0.Q2

Q2= **X./Q0./Q1** + /Q0./Q2 + /Q0.Q1

A Q2 Karnaugh tábláján a kékkel jelölt hurok nem a legegyszerűbb, vagyis nem prímimplikáns. Így azonban közös lesz a Q1 egyik termével, s az így adódó hálózat kevesebb bemenetből áll.
A közös prímimplikánsokat is figyelembe véve összesen 17 kapu bemenet, szemben a véletlenszerű kódolásnál adódó 26-al.

Kódolás Helyettesítési Tulajdonságú (HT) partíciók alapján (Önfüggő szekunder változó csoportok kialakítása)

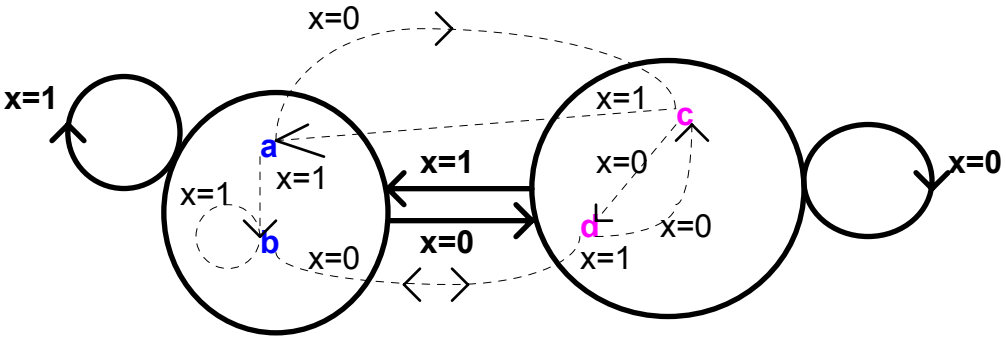
Az állapotok egy Π partíciója, olyan B diszjunkt részhalmazokra osztja az A állapothalmazt, melyek uniója az összes állapotot tartalmazza.

$$\Pi_i = B_1,..B_i,..B_n \qquad \bigcup_i B_i = A$$

$$B_i \cap B_j = 0 \qquad \forall i \neq j - re$$

Az állapotok egy HT partíciója olyan tulajdonságú blokkokból áll, hogy ismerve, hogy egy állapot a partíció mely blokkjában van és ismerve a bemenet értékét, megmodható, hogy a következő állapot a partíció mely blokkjában lesz . (Azt viszont nem tudjuk megmondani, hogy a blokkon belül mely állapotban.)

$$B^{t+1} = f(B^t, X)$$



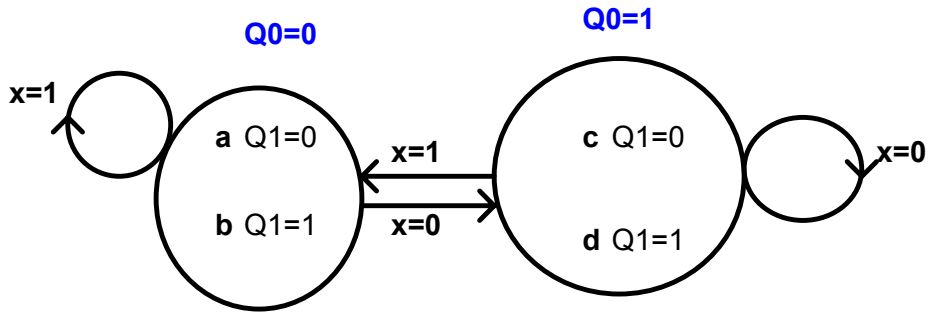
A fenti gráf egy HT partíciója Π_1= (ab)(cd)

A makrográf vastag vonallal, az eredeti szaggatottal.

Triviális, HTP szerinti kódolásra nem alkalmas partíciók:
Minden állapot külön blokkban szerepel: Π_O
Minden állapot azonos blokkban szerepel: Π_I

A triviálisaktól eltérő partíciók alapján önfüggő szekunder változó csoportokat lehet kialakítani.

Kódoljunk úgy, hogy a szükséges számú Q_{Ai} szekunder változóval különböztessük meg a partíció blokkjait, és a legtöbb állapotot tartalmazó blokk által megszabott számúval Q_{Bj} a blokkon belüli állapotokat.



A partíció blokkjait megkülönböztető szekunder változók aktuális értéke és a bemenet alapján megmondható azok következő értéke:

$$Q_{AI}^{t+1} = f(Q_{AI}^{t}, \underline{X})$$

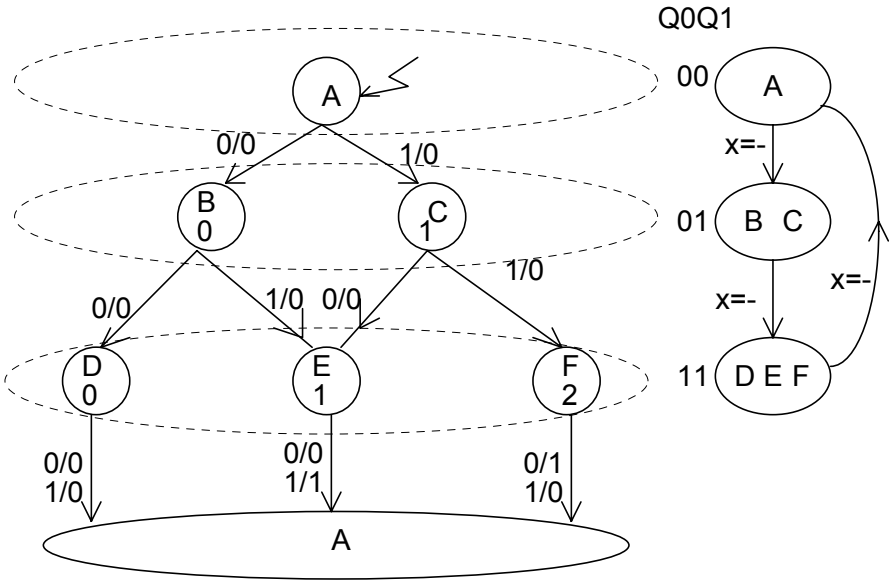
Tehát a blokkokat kódoló szekunder változók önfüggőek. Itt $Q0^{t+1} = f(Q0^{t},x)$

A HT partíció szerinti kódolás előnye, hogy a partíció blokkjait kódoló szekunder változók függvényei kevesebb változótól függnek, mint a többi, vagyis egyszerűbbek, olcsóbban megvalósíthatók.

HT partíciók keresése

Egyes állapotgráf tipusok esetén “szemre” is található HT partíció.

A szinkron hálózatok bevezető feladata állapotgráfjának egy HT partíciója:



Tehát a partíció blokkjai: **(A) (BC) (DEF)**

Ez alapján a kódoláshoz 4 szekunder változó kell:

Q2Q3

Q0Q1	00	01	11	10
00	**A**			
01		**B**	**C**	
11		**D**	**E**	**F**
10				

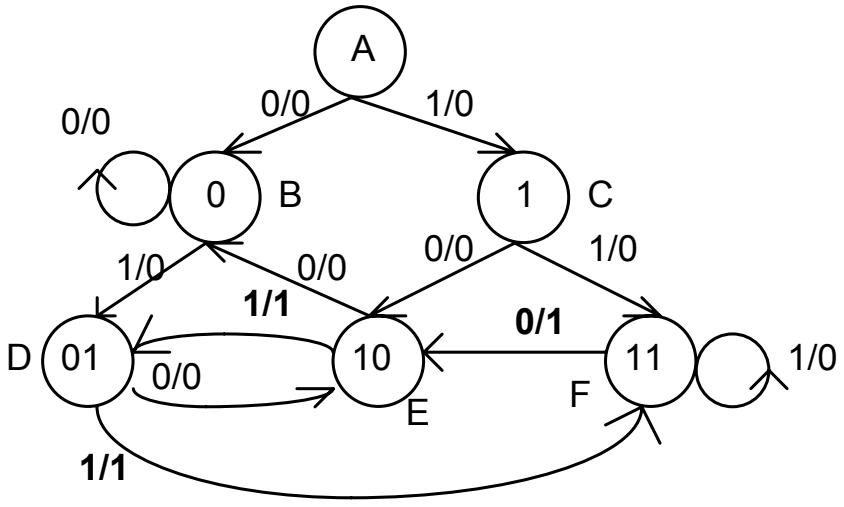
Önfüggő szekunder változók: Q0,Q1

Mivel a fenti feladat 3 változóval is kódolható lenne, itt nem célszerű a HT partíció szerinti kódolás.

Szisztematikus módszer HT partíciók keresésére

A módszer lényege, hogy minden állapotpárra felírjuk, az egy blokkba partozásuk feltételét:
Két állapot egy blokkba tartozik, ha tetszőleges, de azonos bemenetre a következő állapotok is egy blokkba tartoznak. A feltételeket lépcsős táblába írjuk be.
Ez hasonló az TSH állapotminimalizálásnál a lépcsős tábla kitöltéséhez, csak itt a kimenetet nem kell figyelembe venni.

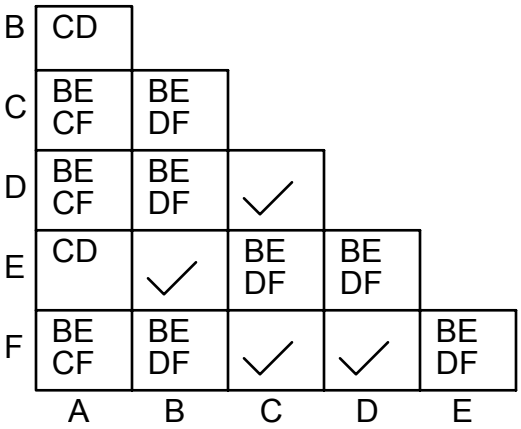
Egy példán mutatjuk be. Egy olyan automata állapotgráfján mutatjuk be, amelynek feladata az egyetlen bementére sorosan az órajellel szinkronban jövő bitfolyamban jelezni, ha a legutolsó 3 bitben pontosan 2 db 1-es van. Az állapotgráf az alábbi.
x:0110111001011…
z:0011110100011…



Az állapottábla:

	X=0	X=1
A	B/0	C/0
B	B/0	D/0
C	E/0	F/0
D	E/0	F/1
E	B/0	D/1
F	E/1	F/0

A lépcsős tábla:



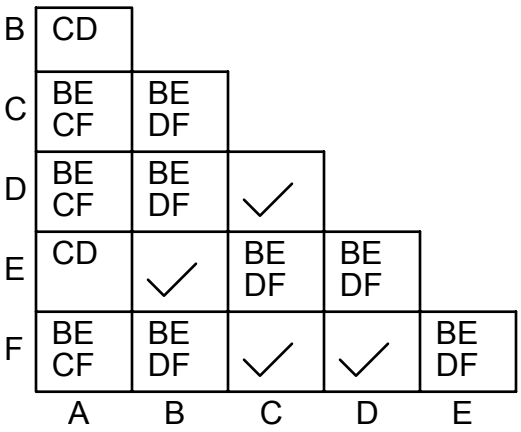
Minden állapotpárból kiindulva meg lehet próbálni, a feltételláncot felgöngyölíteni. A feltételláncban nem szereplő állapotok önmagukban a HTP egy blokkját alkotják. A közös állapotokat tartalmazó feltételeket egyetlen blokká kell olvasztani (tranzitivitás):

(AB): (CD → Π1= (AB) (CD) (E) (F)

(AC): (BE) (CF) → Π2= (ACF) (BE) (D)

(AD): (BE) (CF) → Π3= (AD) (BE) (CF)

(AF): (BE) (CF) →(ACF) (BE) (D) már van

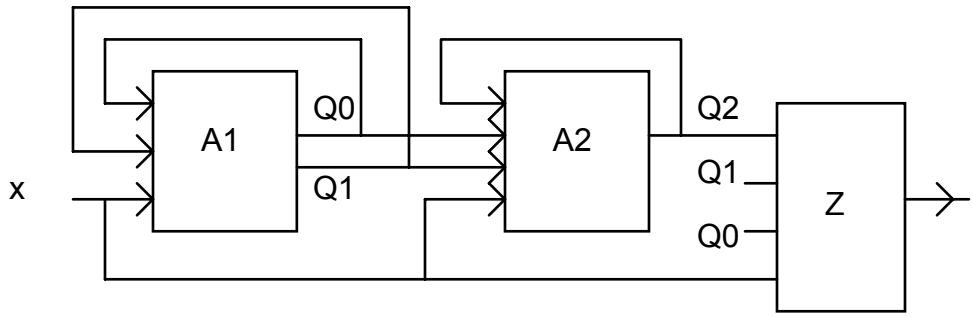


(BC): (BE) (DF) → Π4= (BCE) (DF) (A)
(BD): (BE) (DF) → Π5= (BDEF) (A) (C)
(BE): → Π6= (BE) (A) (C) (D) (F)
(BF): (BE) (DF) → (BDEF) (A) (C) már van
(CD): → Π7= (CD) (A) (B) (E) (F)
(DE): (BE) (DF) → (BDEF) (A) (C) már van
(FD): → Π8= (FD) (A) (B) (C) (E)
(FE): (BE) (DF) → (BDEF) (A) (C) már van

A legkevesebb (3) szekunder változóval kódolható:
Q0Q1: 00, 01, 11 (önfüggő szekunder változók)
Π3 = (AD) (BE) (CF)
Q2: 0,1 0,1 0,1

A Π3 az optimális, mivel abban több az önfüggő szekunder változó, így egyszerűbb a hálózat.

Az így adódó hálózat struktúrája:



Az A1 automatával, mely az önfüggő szekunder változókat valósítja meg, sorba kapcsolódik az A2 automata.

Egy HT partíció alapján kódolva soros dekompozíció alakul ki.

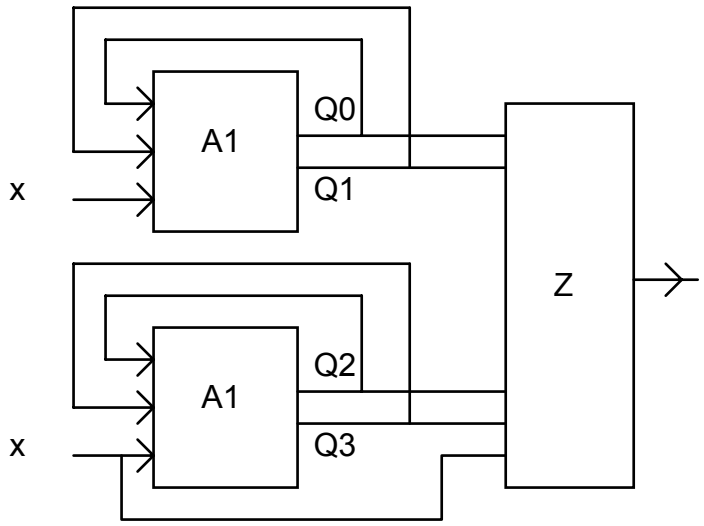
Bizonyos esetekben akár egyszerre két HT partíció alapján kódolhatunk, így 2 önfüggő szekunder változó csoport alakul ki.

Q0Q1	00	01	11	10
Q2Q3		ACF	BE	D
00	AB	**A(0001)**	**B(0011)**	
01	CD	**C(0101)**		**D(0110)**
11	E		**E(0111)**	
10	F	**F(1010)**		

Π1= (AB) (CD) (E) (F) és Π2= (ACF) (BE) (D) esetén a partíciók blokkjainak metszete maximum 1 állapotot tartalmaz és az összes állapot szerepel benne, vagyis **ortogonálisak**.

Az előbbi példa esetén azonban ez bonyolultabb hálózatot eredményez, mivel a minimális 3 helyett 4 flip-flopra van szükség.

Az így adódó hálózat struktúrája:



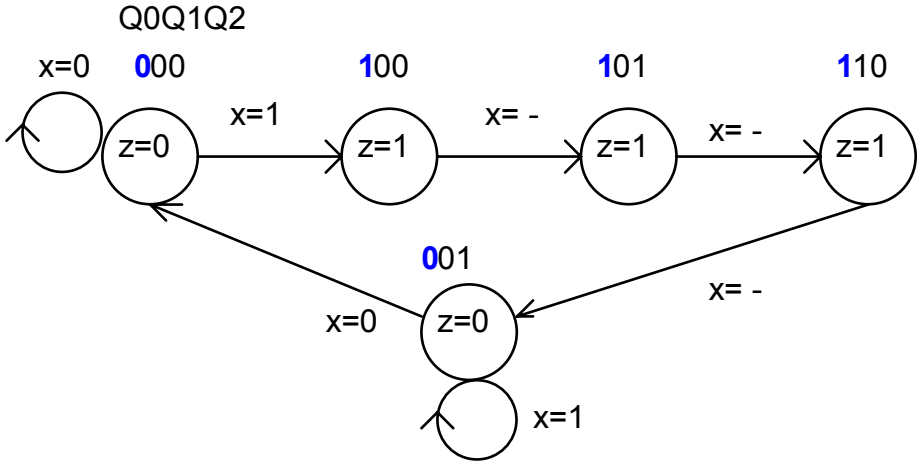
A két önfüggő szekunder változó csoportot megvalósító automata egymástól függetlenül, párhuzamosan működik.

Két ortogonális HT partíció alapján kódolva, párhuzamos dekompozíció alakul ki.

Kódolás kimenet alapján

Ha az állapotkódot úgy választjuk meg, hogy valamely szekunder változók közvetlenül a kimenetet adják, akkor elmarad a kimenetet előállító kombinációs hálózat és a kimenet hazárdmentes lesz, mivel flip-flop kimenetek állítják elő.

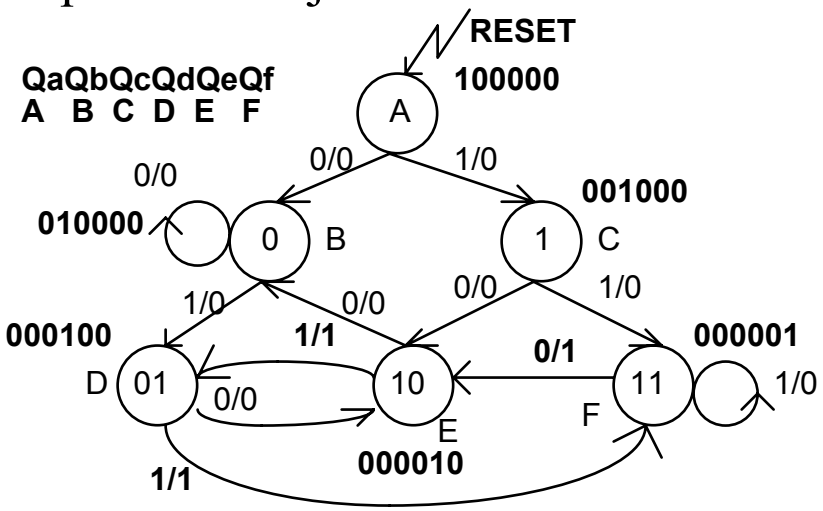
A kimenet ilyen előállítása sokszor olyan kényszer az állapotkód előállításánál, ami csak többlet szekunder változók felvételével elégíthető ki. Így konkrét esetben mindig mérlegelni kell, hogy megéri-e. Vannak feladatok, amikor előírás a kimenet szerinti kódolás (pl. számlálók esetén).
Példa: Készítsünk olyan áramkört, amely az x bemenetére adott legalább egy órajel hosszú magas impulzust követően 3 órajel hosszú hazárdmentes kimeneti impulzust produkál (digitális monostabil) a z kimenetén. Az állapotgráf:



Itt a kódolás olyan, hogy a Q0 adja a Z kimenetet.

Állapotonként egy bit kódolás (Bit per state, one hot encoding)

Az FPGA-esetén sok a regiszter, de viszonylag kevés bemenetszámú a hozzá tartozó kombinációs hálózat. Ezért itt sokkal fontosabb, hogy egyszerű legyen a vezérlő függvény, mint hogy kevés flip-floppot használjunk fel. Ilyen esetben n-ből 1 kódolást alkalmaznak, azaz minden állapothoz egy flip-flopot rendelnek és az n db flip-flopból mindig csak az 1 értékű, amely az aktuális állapothoz tartozik, ahogy az alábbi példa mutatja.



A szekunder változók függvényei a gráfból rögtön felírhatók és láthatóan egyszerűek:

Qa = RESET

Qb= (Qa./X + Qb./X + Qe./X)./RESET

Qc = Qa.X./RESET

Qd= (Qb.X + Qe.X). /RESET

Qe = (Qc./X + Qd./X + Qf./X)./RESET

Qf = (Qc.X + Qd.X + Qf.X)./RESET

A flip-flopok alaphelyzetét a RESET jel állítja be, a törlő (Cl) bemenetüket aktivizálva. Ugyanez a hatása, ha a RESET jellel ÉS kapcsolatba hozzuk a Qb-Qf szekunder változók függvényeit, ahogyan a példában tettük.

Ez a kódolás érzékeny a zavarokra. A lehetséges állapotkódok közül csak keveset használ ki, így nagyon sok az illegális állapot. Ha az automata zavar hatására illegális állapotba kerül, onanntól kezdve hibásan működik. Ha viszont az összes illegális állapotból vissza akarjuk vezetni valamely legálisba, akkor elbonyolódik a kapcsolás.

Szinkron hálózatok tervezési lépései