SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE,
STROJARSTVA I BRODOGRADNJE
U SPLITU

Digitalni sustavi i strukture
Teorija
Računarstvo 750/3 - šk. god. 2004./2005.

Student: Andrija Vujević

Split, veljača 2005.

1. Zbrajanje po modulu

Zbrajanje po modulu m definiramo kao ostatak (Rez) **cjelobrojnog dijeljenja** zbroja a+b s modulom m:

$$a \oplus b = c = \mathrm{Re}\,z\left(\frac{a+b}{m}\right)$$

Opreracija $\oplus$ se zove zbrajanje po modulu m, dok se $a \oplus b$ čita "**a plus b po modulu m**", odnosno "**a plus b modulo m**". Operacija zbroja po modulu nad konačnim skupom F ima slijedeća svojstva:

- zatvorenost: $\forall a,b \in F:\ a \oplus b = c:\ c \in F$
- komutativnost: $\forall a,b \in F:\ a \oplus b = b \oplus a$
- asocijativnost: $\forall a,b,c \in F:\ (a \oplus b) \oplus c = a \oplus (b \oplus c)$
- neutralni element: $\forall a \in F \quad \exists e \in F:\ a \oplus e = e \oplus a = a$
- inverz: $\forall a \in F \quad \exists a' \in F:\ a \oplus a' = a' \oplus a = e$

Neutralni element:

$$a \oplus e = a \quad \Rightarrow \quad \mathrm{Re}\,z\left(\frac{a+e}{m}\right) = \mathrm{Re}\,z\left(\frac{a}{m}\right) \quad \Rightarrow a + e = a \quad \Rightarrow \quad e = 0$$

Inverzni element: $a \oplus a' = a' \oplus a = 0 \quad \Rightarrow \qquad a' = \begin{cases} 0 & ; \quad a = 0 \\ m - a & ; \quad a > 0 \end{cases}$

Postavljanjem da je $m = 2^n$, povezali smo operaciju zbroja po modulu s **binarnim brojevima** duljine n bita. Skup F sada je istovremeno i skup svih binarnih brojeva koje možemo izraziti kompleksijama od n bita:

$$F \in \{0,\ ...,\ m-1\} = \{0,\ ...,\ 2^n - 1\} = \{0,\ ...,\ a_{max}\}$$

Može se dokazati da je **inverz u binarnom sustavu** (**dvojni komplement**, **drugi komplement**, **komplement po modulu dva**) jednak:

$$a' = \begin{cases} 0 & ; \quad a = 0 \\ \overline{a} + 1 & ; \quad a > 0 \end{cases} \qquad a' = \left\{ \begin{array}{lll} 0 & \text{za} & a = 0 \\ \overline{a} + 1 & \text{za} & a > 0 \end{array} \right\} = \overline{a} \oplus 1$$

što znači da inverz dobijemo povećanjem običnog komplementa za 1 po modulu.

Obični komplement se jednostavno računa pomoću **invertora**.

Obični komplement ili kraće **komplement** broja $a = \sum_{k=0}^{n-1} a_k \cdot 2^k$ u binarnom brojevnom sustavu glasi: $\overline{a} = \sum_{k=0}^{n-1} \overline{a}_k \cdot 2^k$ gdje je $\overline{a}_k$ komplement znamenke a_k.

Obični komplement se jednostavno računa pomoću **invertora**.

Za $a > b$ **oduzimanje po modulu** se definira slijedećim izrazom:

$$a - b|_{MOD} = Rez(\frac{a-b}{m})$$

Vrijedi da je: $Rez(\frac{a-b}{m}) = Rez(\frac{a-b+km}{m})$ gdje je k cijeli broj.

Za $k = 1$ imamo: $Rez\ (\frac{a+m-b}{m}) = Rez(\frac{a+b'}{m})$

ili kraće: $a - b|_{MOD} = a \oplus b'$

Dakle, oduzimanje ostvarimo jednostavno pribrajanjem inverza. Pri tome inverz poprima značenje negativnog broja.

Raspoložive brojeve (kodne riječi) prirodnog binarnog koda možemo smatrati samo pozitivnima (skup $\mathbb{N}_o$) ili pozitivnima i negativnima (skup $\mathbb{Z}$), pri čemu za prikaz negativnih brojeva koristimo inverze po modulu 2.

Kod zbrajanja rezultat točan kad je prijenos (Carry) 0.

Kod oduzimanja rezultat točan kad je prijenos (Carry) 1.

Bit predznaka - najznačajniji bit jednak jedinici $\Rightarrow$ broj negativan.

Bit predznaka - najznačajniji bit jednak nuli $\Rightarrow$ broj pozitivan.

Pozitivni i negativni brojevi se mogu grafički prikazati na kružnici.

2. Booleova algebra

Definicija: Booleova algebra je matematička struktura definirana kao:

$$B.A. = \{G,\ x,\ =,\ S\}$$

gdje je: G - skup operatora algebre;

= - operator jednakosti;

$S = \{0,\ 1\}$ - skup Booleovih konstanti 0 i 1;

$x \in S$ - Booleova varijabla koja uzima vrijednosti iz S.

Algebra logike je Booleova algebra kod koje je skup G:

$$G = \{\&,\ V,\ -\}$$

pa je:

$$A.L. = \{\&,\ V,\ -,\ =,\ x,\ S = \{0,\ 1\}\}.$$

Napomena: Algebarski operatori (konjunkcija, disjunkcija i negacija) preuzeti su iz **logike sudova**, tj. operatori logike sudova su uzeti kao algebarski operatori, dok su istina i neistina iz logike sudova zamijenjeni s 1 i 0: $\top \to 1,\ \ \bot \to 0$.

Koriste se dvije različite oznake za algebarski operator konjunkcije: &, ·

Koriste se dvije različite oznake za algebarski operator disjunkcije: V, +

Postulati (osnovna svojstva) **algebre logike** su:

1. **Zatvorenost**

   a) $\forall x_1, x_2 \in S \ \Rightarrow\ x_1 \vee x_2 \in S$ (disjunkcija je zatvorena u S)

   b) $\forall x_1, x_2 \in S \ \Rightarrow\ x_1 \,\&\, x_2 \in S$ (konjunkcija je zatvorena u S)

   c) $\forall x_1 \in S \ \Rightarrow\ \overline{x}_1 \in S$ (negacija je zatvorena u S)

2. **Neutralni element**

   a) $\forall x_1, 0 \in S \ \Rightarrow\ x_1 \vee 0 = x_1$ (0 je neutralni element za disjunkciju)

   b) $\forall x_1, 1 \in S \ \Rightarrow\ x_1 \,\&\, 1 = x_1$ (1 je neutralni element za konjunkciju)

3. **Komutativnost**

   a) $\forall x_1, x_2 \in S \ \Rightarrow\ x_1 \vee x_2 = x_2 \vee x_1 \in S$ (disjunkcija je komutativna)

   b) $\forall x_1, x_2 \in S \ \Rightarrow\ x_1 \,\&\, x_2 = x_2 \,\&\, x_1 \in S$ (konjunkcija je komutativna)

4. **Distributivnost**

a) $\forall x_1, x_2, x_3 \in S \;\Rightarrow\; x_1 \vee (x_2 \,\&\, x_3) = (x_1 \vee x_2) \,\&\, (x_1 \vee x_3)$

(disjunkcija je distributivna s obzirom na konjunkciju)

b) $\forall x_1, x_2, x_3 \in S \;\Rightarrow\; x_1 \,\&\, (x_2 \vee x_3) = (x_1 \,\&\, x_2) \vee (x_1 \,\&\, x_3)$

(konjunkcija je distributivna s obzirom na disjunkciju)

5. **Komplementiranje**

a) $\forall x_1 \in S \;\Rightarrow\; x_1 \vee \overline{x}_1 = 1$

(disjunkcija nenegirane i negirane varijable jednaka je jedinici)

b) $\forall x_1 \in S \;\Rightarrow\; x_1 \,\&\, \overline{x}_1 = 0$

(konjunkcija nenegirane i negirane varijable jednaka je nuli)

6. **Asocijativnost**

a) $\forall x_1, x_2, x_3 \in S \;\Rightarrow\; x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3$

(disjunkcija je asocijativna)

b) $\forall x_1, x_2, x_3 \in S \;\Rightarrow\; x_1 \,\&\, (x_2 \,\&\, x_3) = (x_1 \,\&\, x_2) \,\&\, x_3$

(konjunkcija je asocijativna)

Teoremi (izvedena svojstva**) algebre logike** su:

T1. Aposrpcija za disjunkciju: $x_1 \vee 1 = 1$

Jedinica disjunktivno vezana s nekim izrazom apsorbira taj izraz.

Dokaz:

$$x_1 \vee 1 \underset{P_{2b}}{\equiv} (x_1 \vee 1) \cdot 1 \underset{P_{5a}}{=} (x_1 \vee 1) \cdot (x_1 \vee \overline{x}_1) \underset{P_{4a}}{=} x_1 \vee (1 \cdot \overline{x}_1) \underset{P_{2b}}{=} x_1 \vee \overline{x}_1 \underset{P_{5a}}{=} 1$$

T2. Idempotentnost za disjunkciju $x_1 \vee x_1 = x_1$

Disjunkcija nekog izraza sa samim sobom jednaka je tom izrazu. Koristi se za sažimanje ili proširenje algebarskog izraza postojećim članom.

Dokaz:

$$x_1 \vee x_1 \underset{P_{2b}}{\equiv} (x_1 \vee x_1) \cdot 1 \underset{P_{5a}}{=} (x_1 \vee x_1) \cdot (x_1 \vee \overline{x}_1) \underset{P_{4a}}{=} x_1 \vee (x_1 \cdot \overline{x}_1) \underset{P_{5b}}{=} x_1 \vee 0 \underset{P_{2a}}{=} x_1$$

T3. Idempotentnost za konjunkciju $x_1 \cdot x_1 = x_1$

Konjunkcija nekog izraza sa samim sobom jednaka je tom izrazu. Koristi se za sažimanje ili proširenje algebarskog izraza postojećim članom.

Dokaz:

$$x_1 \cdot x_1 \underset{P_{2a}}{\equiv} x_1 \cdot x_1 \vee 0 \underset{P_{5b}}{=} x_1 \cdot x_1 \vee x_1 \cdot \overline{x}_1 \underset{P_{4b}}{=} x_1 \cdot (x_1 \vee \overline{x}_1) \underset{P_{5a}}{=} x_1 \cdot 1 \underset{P_{2b}}{=} x_1$$

T4. Dvostruka negacija $\overline{\overline{x}}_1 = x_1$

Dokazuje se tablicom:

x_1	$\overline{x}_1$	$\overline{(\overline{x}_1)} = \overline{\overline{x}}_1$
0	1	0
1	0	1

T5. Apsorpcija za konjunkciju $x_1 \cdot 0 = 0$

Nula konjunktivno vezana s nekim izrazom apsorbira taj izraz.

Dokaz:

$$x_1 \cdot 0 \underset{P_{2a}}{\equiv} x_1 \cdot 0 \vee 0 \underset{P_{5b}}{=} x_1 \cdot 0 \vee x_1 \cdot \overline{x}_1 \underset{P_{4b}}{=} x_1 \cdot (0 \vee \overline{x}_1) \underset{P_{2a}}{=} x_1 \cdot \overline{x}_1 \underset{P_{5b}}{=} 0$$

Teorema s dvije i više varijabli ima nekoliko. Ovdje ćemo navesti samo DeMorganove teoreme.

T6. DeMorganov teorem za disjunkciju $\overline{x_1 \vee x_2} = \overline{x}_1 \cdot \overline{x}_2$

Dokazuju se indukcijom: ako je lijeva strana jednaka desnoj (A = A), mora vrijediti postulat o komplementiranju u oba oblika. Pišemo:

$$A = \overline{x_1 \vee x_2} \quad ; \quad \overline{A} = \overline{\overline{x_1 \vee x_2}} = x_1 \vee x_2 \quad \text{i} \quad A = \overline{x}_1 \cdot \overline{x}_2$$

pa mora vrijediti:

a) $A \vee \overline{A} = 1 \quad \Rightarrow \quad \overline{x}_1 \cdot \overline{x}_2 \vee (x_1 \vee x_2) = 1$

b) $A \cdot \overline{A} = 0 \quad \Rightarrow \quad \overline{x}_1 \cdot \overline{x}_2 \cdot (x_1 \vee x_2) = 0$

Dokaz: a) $\overline{x}_1 \cdot \overline{x}_2 \vee (x_1 \vee x_2) = (\overline{x}_1 \vee x_1 \vee x_2) \cdot (\overline{x}_2 \vee x_1 \vee x_2) = 1 \cdot 1 = 1$

b) $\overline{x}_1 \cdot \overline{x}_2 \cdot (x_1 \vee x_2) = (\overline{x}_1 \cdot \overline{x}_2 \cdot x_1) \vee (\overline{x}_1 \cdot \overline{x}_2 \cdot x_2) = 0 \vee 0 = 0$

čime je teorem dokazan.

T7. DeMorganov teorem za konjunkciju $\overline{x_1 \cdot x_2} = \overline{x}_1 \vee \overline{x}_2$

Dokazuje se preko T6: $\overline{x_1 \cdot x_2} = \overline{x}_1 \vee \overline{x}_2 \quad / \; ^{-}$

$$\overline{\overline{x_1 \cdot x_2}} = \overline{\overline{x}_1 \vee \overline{x}_2}$$

$$x_1 \cdot x_2 = \overline{\overline{x}}_1 \cdot \overline{\overline{x}}_2 = x_1 \cdot x_2$$

čime je teorem dokazan.

3. Booleove funkcije (BF)

Definicija: Za neki skup Booleovih varijabli:

$$X = \{x_1, x_2, ..., x_n\}$$

$P_n(x)$ je nadskup (Power Set) skupa X i sadrži sve kodne riječi varijabli $x_1, ..., x_n$.

Booleova funkcija

$$y = f(x_1, x_2, ..., x_n)$$

je preslikavanje nadskupa $P_n(x)$ u skup $S = \{0, 1\}$.

Booleova funkcija može biti potpuno ili nepotpuno specificirana.

Potpuno specificirana funkcija je funkcija, kod koje je preslikavanje definirano za sve kodne riječi ulaznih varijabli.

Nepotpuno specificirana funkcija je funkcija, kod koje je preslikavanje definirano samo za neke kodne riječi ulaznih varijabli. Nepotpuno specificirana funkcija ima smisla ako pretpostavimo da se neke određene **kompleksije** ulaznih varijabli u praksi neće nikada pojaviti na ulazu u funkciju. To su one kompleksije koje u postupku kodiranja nisu iskorištene i nemaju suvislog značenja. Za njih funkciju nije potrebno definirati, odnosno možemo je definirati proizvoljno. Takve se kompleksije nazivaju redundantnima.

Zapis Booleovih funkcija

Booleovu funkciju važno je formalno zapisati, da bi bila jednoznačno definirana. $P_n(x)$ je konačan, pa su nam na raspolaganju **tablični**, **algebarski** i **grafički** zapisi.

Tablični način zapisa standardiziran je u obliku tablice istine.

Algebarski oblik zapisa Booleove funkcije je zapis konačnim algebarskim izrazom u okviru algebre logike, ili neke druge Booleove algebre. S obzirom na svojstva algebre logike (postulati i teoremi), očito je da se zamjenom može svaki algebarski izraz transformirati na po volji velik broj načina, a da on pri tome ne promjeni svoju vrijednost.

Grafički zapis Booleovih funkcija može se ostvariti pomoću

- Vennovih dijagrama
- Veitchevih dijagrama
- Logičkih dijagrama (blok shemama funkcije)
- Shema sklopa

Potpuni normalni algebarski oblici su najvažniji oblici algebarskog zapisa neke funkcije jer omogućavaju:

- ispis neposredno iz tablice istine (potpuni normalni oblik);
- optimalnu realizaciju sklopa (minimalni normalni oblik);
- minimalno i jednoliko kašnjenje (dvije razine logičkih vrata);
- mogućnost minimizacije egzaktnim postupcima;
- zajamčen prijelaz na NI i NILI vrata bez gubitka navedenih svojstava.

Definicija: **Potpuni disjunktivni normalni oblik** (PDNO) funkcije je disjunkcija onih minterma m_i, za čiji je i-ti redak u tablici istine vrijednost funkcije T_i jednaka jedinici. Opći oblik PDNO je:

$$f(x_1, x_2, ..., x_n) = \bigvee_{i=0}^{2^n-1} m_i \cdot T_i$$

Definicija: **Minterm** i-tog retka tablice istine, m_i, je konjunkcija svih varijabli funkcije uzetih tako da su one koje u kodnoj riječi imaju vrijednost 0 negirane, a one koje imaju vrijednost 1 nenegirane. Npr. minterm trećeg retka za n=3 glasi:

$$m_3(x_1, x_2, x_3)_{011} = \overline{x}_1 \cdot x_2 \cdot x_3$$

Definicija: **Potpuni konjunktivni normalni oblik** (PKNO) funkcije je konjunkcija onih maksterma M_i, za čiji je i-ti redak u tablici istine vrijednost funkcije T_i jednaka nuli. Opći oblik PKNO je:

$$f(x_1, x_2, ..., x_n) = \bigwedge_{i=0}^{2^n-1} (M_i \vee T_i)$$

Definicija: **Maksterm** i-tog retka tablice istine, M_i, je disjunkcija svih varijabli funkcije uzetih tako da su one koje u kodnoj riječi imaju vrijednost 0 nenegirane, a one koje imaju vrijednost 1 negirane. Npr. maksterm trećeg retka za n=3 glasi:

$$M_3(x_1, x_2, x_3)_{011} = x_1 \vee \overline{x}_2 \vee \overline{x}_3$$

4. Potpuni skupovi funkcija algebre logike

Elementarne funkcije su funkcije jedne i dvije varijable (operanda), a to su npr.

Sve funkcije jedne varijable (unarne funkcije) se mogu definirati tablicom istine:

x_1	f_o	f_1	f_2	f_3
0	0	1	0	1
1	0	0	1	1

Unarna funkcija f_1 je negacija. Binarnih funkcija imamo ukupno 16, a među njih spadaju i konjunkcija i disjunkcija i njihove negacije. Neke funkcije dviju varijabli se mogu interpretirati kao funkcije jedne varijable.

Tablica istine za n varijabli ima 2^n redaka. Funkciju definiramo stupcem koji možemo smatrati kodnom riječi od $N = 2^n$ bita. Takvih kodnih riječi ima:

$$2^N = 2^{2^n}$$

što znači da imamo upravo toliko različitih Booleovih funkcija n varijabli.

Definicija: Potpuni skup funkcija algebre logike je takav skup operatora pomoću kojeg se konačnim algebarskim izrazom može zapisati proizvoljna Booleova funkcija.

Definicija: Skup konjunkcije, disjunkcije i negacije je potpuni skup funkcija algebre logike. **Potpunost** bilo kojeg drugog skupa dokazujemo tako, da pokušamo izraziti konjunkciju, disjunkciju i negaciju. Za skup operatora kojim uspijemo izraziti konjunkciju, disjunkciju i negaciju zaključujemo da je potpun.

Dakle, skup $\{\&, \vee, -\} \equiv \{\cdot, \vee, -\}$ je potpuni skup fja algebre logike.

Skup $\{\&\}$ nije potpun.

Skup $\{\vee\}$ nije potpun.

Skup $\{-\}$ nije potpun.

Skup $\{\&, -\}$ je potpun. Dokaz: $x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{\overline{x}_1 \cdot \overline{x}_2}$

Skup $\{V, -\}$ je potpun. Dokaz: $x_1 \cdot x_2 = \overline{\overline{x_1 \cdot x_2}} = \overline{\overline{x}_1 \vee \overline{x}_2}$

Shaefferov operator $\left(\{|\}, \text{Shaeffer, NI, NAND, } \overline{x_1 \cdot x_2} \right)$ je sam za sebe potpuni skup funkcija algebre logike.

Dokaz: $x_1 | x_1 = \overline{x_1 \cdot x_1} = \overline{x}_1$ ili $x_1 | 1 = \overline{x_1 \cdot 1} = \overline{x}_1$

$$(x_1 | x_1) | (x_2 | x_2) = \overline{x}_1 | \overline{x}_2 = \overline{\overline{x}_1 \cdot \overline{x}_2} = x_1 \vee x_2$$

$$(x_1 | x_2) | (x_1 | x_2) = \overline{\overline{x}_1 | \overline{x}_2} = \overline{\overline{\overline{x}_1 \cdot \overline{x}_2}} = x_1 \cdot x_2$$

To znači da jednom vrstom logičkih vrata možemo realizirati proizvoljnu Booleovu funkciju. Budući da elektroničkom tehnologijom lako ostvarujemo NI vrata (štoviše to su elementarna vrata za TTL tehnologiju), njihova upotreba je optimalna i omogućuje optimizaciju broja integriranih krugova sklopa.

Shaefer (NI) operator sam za sebe nije asocijativan, tj. $x_1 | x_2 | x_3 \neq (x_1 | x_2) | x_3$.

Da bi se izbjegla nesigurnost u stvarno značenje algebarskih izraza, u praksi se ne koristi simbol $|$ već se za zapis ni operatora koristi sustav $\{\&, -\}$. U takvom zapisu prepoznat ćemo negiranu konjunkciju dviju ili više varijabli kao NI operator.

Pierce operator $\left(\{\downarrow\}, \text{Pierce, NILI, NOR, } \overline{x_1 \vee x_2} \right)$ je sam za sebe potpuni skup funkcija algebre logike.

Dokaz: $x_1 \downarrow x_1 = \overline{x_1 \vee x_1} = \overline{x}_1 \cdot \overline{x}_1 = \overline{x}_1$ ili $x_1 \downarrow 0 = \overline{x_1 \vee 0} = \overline{x}_1$

$$(x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2) = \overline{\overline{x}_1 \vee \overline{x}_2} = \overline{\overline{x}}_1 \cdot \overline{\overline{x}}_2 = x_1 \cdot x_2$$

$$(x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2) = \overline{(x_1 \downarrow x_2)} = \overline{\overline{x_1 \vee x_2}} = x_1 \vee x_2$$

Budući da je Pierce operator također sam za sebe potpuni skup funkcija algebre logike. jednom vrstom logičkih vrata možemo realizirati proizvoljnu Booleovu funkciju i lakše optimizirati broj integriranih krugova sklopa.

Pierce (NILI) operator sam za sebe nije asocijativan, tj. $x_1 \downarrow x_2 \downarrow x_3 = (x_1 \downarrow x_2) \downarrow x_3$.

Da bi se izbjegla nesigurnost u stvarno značenje algebarskih izraza, u praksi se ne koristi simbol $\downarrow$ već se za zapis NILI operatora koristi sustav $\{\vee, -\}$. U takvom zapisu prepoznat ćemo negiranu disjunkciju dviju ili više varijabli kao NILI operator.

5. Minimizacija Booleovih funkcija

Složeni digitalni sklop ima više ulaza i izlaza, a on se sastoji od više jednostavnih digitalnih sklopova. Jednostavni sklop ima samo jedan izlaz i njegov rad je opisan jednom Booleovom funkcijom.

Među svim algebarskim oblicima zapisa neke Booleove funkcije želimo izabrati optimalan oblik, tj. onaj oblik koji zadovoljava postavljene kriterije minimalnosti.

Cilj minimizacije je da konačan sklop bude:

- **ekonomičan** u proizvodnji i primjeni (dakle minimalan),
- **pouzdan**,
- **brz**,
- takav da se njegova konstrukcija može izvesti **egzaktnim postupcima** da bi se mogućnost pogreške svela na najmanju moguću mjeru.

Minimalnost možemo definirati kao:

- minimalan broj (diskretnih) komponenti;
- minimalan broj integriranih krugova;
- **minimalan broj logičkih vrata;**
- minimalna površina štampane pločice;
- minimalna potrošnja energije.

Koristit ćemo kriterij minimalnog broja logičkih vrata i NI i NILI operatore. Uključimo li ostale zahtjeve, algebarski oblik funkcije mora biti napisan tako da:

- bude minimalan;
- osigura minimalno kašnjenje i jednolikost kašnjenja;
- omogući primjenu postupaka minimizacije;
- omogući primjenu NI i NILI vrata.

Sve ove kriteriji zadovoljavaju minimalni normalni oblici.

Algebarske osnove minimizacije

Primjenom postulata i teorema algebre logike, iz potpunog disjunktivnog normalnog oblika (PDNO) dobije se minimalni disjunktivni normalni oblik (MDNO), a iz potpunog konjunktivnog normalnog oblika (PKNO) može se dobiti minimalni konjunktivni normalni oblik (MKNO). U praksi, MKNO se dobije transformacijom negirane funkcije.

Definicija: **Minimalni disjunktivni normalni oblik (MDNO)** je disjunkcija nužnih elementarnih članova tipa minterma.

Definicija: **Minimalni konjunktivni normalni oblik (MKNO)** je konjunkcija nužnih elementarnih članova tipa maksterma.

Razlikujemo osnovni algebarski postupak i pomoćne algebarske postupke minimizacije normalnih oblika.

Kod osnovnog postupka minimizacije tražimo članove čije su pripadne kodne riječi susjedne (susjedni članovi).

Osnovni postupak minimizacije za PDNO:

$$x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x}_3 \vee \ldots = x_1 \cdot x_2 \vee \ldots$$

Osnovni postupak minimizacije za PKNO:

$$\left(x_1 \vee \overline{x}_2 \vee x_3\right) \cdot \left(\overline{x}_1 \vee \overline{x}_2 \vee x_3\right) \cdot \ldots = \left(\overline{x}_2 \vee x_3\right) \cdot \ldots$$

Pomoćni postupci minimizacije normalnih oblika zasnivaju se na mogućnosti proširenja algebarskih oblika. To se radi na taj način da se postojeći član dopisuje na osnovi svojstva idempotentnosti: $\left(x \vee x = x,\ x \,\&\, x = x\right)$ te dopisivanjem redundantnog člana za nepotpuno specificirane funkcije.

Za PDNO kad izraz proširujemo postojećim članom vrijedi:

$$\begin{aligned} & x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x}_3 \vee x_1 \cdot \overline{x}_2 \cdot x_3 \vee \ldots \\ & = x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x}_3 \vee x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x}_2 \cdot x_3 \vee \ldots \\ & = x_1 \cdot x_2 \vee x_1 \cdot x_3 \vee \ldots \end{aligned}$$

Za PDNO kad izraz proširujemo redundantnim članom vrijedi:

$$\begin{aligned} & x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x}_3 \vee x_1 \cdot \overline{x}_2 \cdot x_3 \vee x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \cdot R^{=1} \ldots \\ & = x_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2 \vee \ldots = x_1 \vee \ldots \end{aligned}$$

Slično vrijedi za PKNO.

Postupci minimizacije normalnih oblika:

- Veitchevim dijagramom
- Quinn-McClusky postupkom
- Harvardskim postupkom

Minimizacija Veitchevim dijagramom

Definicija: Osnovni postupak minimizacije u postupku Veitchevog dijagrama odgovara objedinjavanju susjednih površina.

Definicija: Kod minimizacije PDNO u MDNO u Veitchev dijagram upišemo sve jedinice i redundantne članove (1 i R), sve jedinice zaokružimo sa što manjim brojem što većih površina, te ispišemo MDNO.

Definicija: Kod minimizacije PKNO u MKNO u Veitchev dijagram upišemo sve nule i redundantne članove (0 i R), sve nule zaokružimo sa što manjim brojem što većih površina, te ispišemo MKNO.

Minimizacija se može provoditi i pomoću inverzne (negirane) funkcije.

Definicija: U Veitchev dijagram upišemo PDNO negirane funkcije i minimiziramo. Ispišemo MDNO negirane funkcije i izračunamo MKNO izvorne funkcije negacijom jednadžbe te primjenom DeMorganovih teorema.

Minimizacija Quinn-McClusky postupkom

To je tablično-algebarski postupak koji se sastoji od slijedećih koraka:

- sve minterme PDNO ispišemo jedan ispod drugog,
- provjeravamo susjednost svih parova od vrha prema dnu,
- ako su dva člana susjedna, desno ispisujemo reducirani član,
- na isti način provjeravamo susjednost novih članova te ispisujemo eventualne reducirane članove.

Postupak je gotov kad više nema susjednih članova.

Postupak je definiran za PDNO, a MKNO dobijemo preko negirane funkcije.

Minimizacija Harvardskim postupkom

Ovaj postupak je nastao iz Quinn-McClusky postupka tako da su za određeni broj varijabli unaprijed izračunati svi reducirani članovi. To omogućuje formiranje standardnih tablica pomoću kojih se lako minimizira funkcija.

Postupak je također definiran za PDNO, a MKNO dobijemo preko negirane funkcije.

Značaj Harvardskog postupka je u mogućnosti programiranja na računalu.

i	x_1	x_2	x_3	$x_1 \cdot x_2$	$x_1 \cdot x_3$	$x_2 \cdot x_3$	$x_1 \cdot x_2 \cdot x_3$	y
0	0	0	0	0	0	0	0	1
1	0	0	1	0	1	1	1	R
2	0	1	0	1	0	2	2	0
3	0	1	1	1	1	3	3	0
4	1	0	0	2	2	0	4	1
5	1	0	1	2	3	1	5	1
6	1	1	0	3	2	2	6	1
7	1	1	1	3	3	3	7	1

Posupak minimizacije:

1. upisivanje funkcije u tablicu,
2. precrtamo retke za koje je fja jednaka nuli (ne redundantne članove),
3. prcrtamo (horizontalnim crtama) brojeve po stupcima koji su precrtani u prvom koraku,
4. provjeravajući s desna na lijevo, precrtamo po recima (kosim crtama) one brojeve koji s lijeve strane imaju neprecrtan kraći član,
5. neprecrtane brojeve proglasimo elementarnim članovima,
6. izaberemo nužne elementarne članove tako da u njima budu sadržani svi mintermi za koje je fja jednaka 1, a po potrebi i redundantni članovi.

i	x_1	x_2	x_3	$x_1 \cdot x_2$	$x_1 \cdot x_3$	$x_2 \cdot x_3$	$x_1 \cdot x_2 \cdot x_3$	y
0	0	0	0	0	0	0	0	1
1	0	0	1	0	1	1	1	R
2	0	1	0	1	0	2	2	0
3	0	1	1	1	1	3	3	0
4	1	0	0	2	2	0	4	1
5	1	0	1	2	3	1	5	1
6	1	1	0	3	2	2	6	1
7	1	1	1	3	3	3	7	1

Na kraju se dobije: $y = x_1 \vee \overline{x}_2$.

Transformacije za NI i NILI vrata

Booleova funkcija realizira se NI vratima u slijedećim koracima:

1. Polazimo od PDNO,
2. Izračunamo MDNO,
3. Dvostruko negiramo algebarski izraz,
4. Primjenom DeMorganovih teorema transformiramo izraz,
5. Dobijemo izraz za NI vrata zapisan sustavom $\{\&, -\}$.

Transformacije za opći oblik funkcije su:

$$f(x) = \bigvee_{i=0}^{2^n-1} m_i \cdot T_i = \overline{\overline{\bigvee_{i=0}^{2^n-1} m_i \cdot T_i}} = \overline{\bigwedge_{i=0}^{2^n-1} m_i \cdot T_i}$$

Booleova funkcija realizira se NILI vratima u slijedećim koracima:

1. Polazimo od PDNO negirane funkcije,
2. Izračunamo MDNO negirane funkcije,
3. Negiramo lijevu i desnu stranu jednadžbe,
4. U negaciji s disjunkcijama prepoznamo NILI operator,
5. Dvostruko negiramo svaki član tipa minterma,
6. Primjenom DeMorganovih teorema transformiramo članove tipa minterma u članove tipa maksterma,
7. Uočimo da dobivene disjunkcije s gornjom negacijom čine NILI vrata,
8. Dobijemo izraz za NILI vrata zapisan sustavom $\{\vee, -\}$.

Transformacije za opći oblik funkcije su:

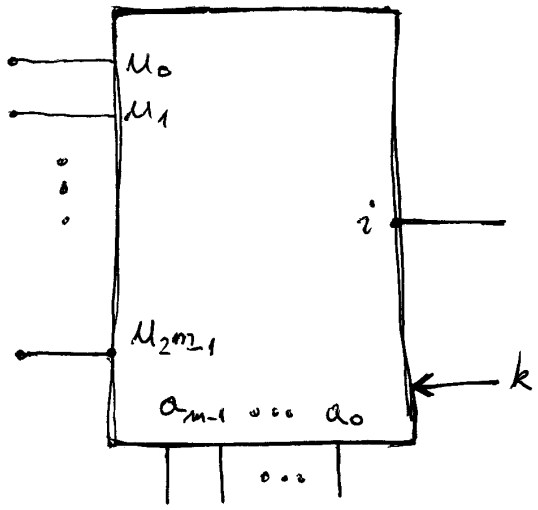
$$\overline{f}(x) = \bigvee_{i=0}^{2^n-1} m_i \cdot \overline{T}_i \quad / \ ^-$$

$$\overline{\overline{f}}(x) = f(x) = \overline{\bigvee_{i=0}^{2^n-1} m_i \cdot \overline{T}_i} = \overline{\overline{\bigvee_{i=0}^{2^n-1} m_i \cdot \overline{T}_i}} = \overline{\bigvee_{i=0}^{2^n-1} M_i \vee T_i}$$

6. Realizacija Booleovih funkcija multiplekserom

Multiplekser je funkcionalni blok (sklop) koji ima:

- m adresnih ulaza: $a_{m-1}, a_{m-2}, ..., a_1, a_0$
- 2^m informacijskih ulaza: $u_{2^m-1}, u_{2^m-2}, ..., u_1, u_0$
- 1 informacijski izlaz: i
- kontrolne ulaze: k



Digitalni multiplekser

Definicija: Multiplekser na informacijski izlaz **i** dovodi vrijednost sa onog informacijskog ulaza u_j, kojemu je redni broj j kao kodna riječ prisutan na adresnim ulazima $a_{m-1}, a_{m-2}, ..., a_1, a_0$, pod uvjetom da je funkcija sklopa omogućena aktivnim signalima na kontrolnim ulazima k. Ako je sklop isključen s kontrolnih ulaza, izlaz je prisilno postavljen na neutralnu vrijednost 0 ili u stanje visoke impedancije.

Kodna riječ se algebarski može prepoznati kao minterm, odnosno kao konjunkcija svih adresnih varijabli. Algebarski možemo zapisati:

$$i = \bigvee_{j=0}^{2^m-1} m_j(a) \cdot u_j = \overline{\overline{\bigvee_{j=0}^{2^m-1} m_j(a) \cdot u_j}} = \overline{\overline{\bigwedge_{i=0}^{2^m-1} m_j(a) \cdot u_j}}$$

Na osnovi ovog izraza, digitalni multiplekser se može ostvariti korištenjem NI vrata.

Stvarni multiplekser redovito ima jedan ili više kontrolnih ulaza. U razmatranju se može koristiti multiplekser bez kontrolnih ulaza kojeg nazivamo **školskim modelom multipleksera.**

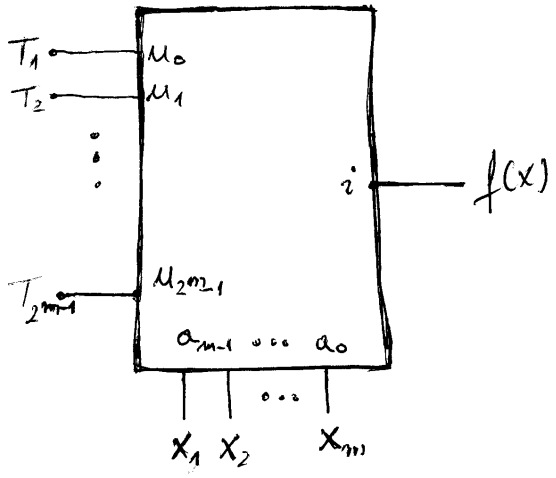
Booleove funkcije multiplekserom realiziramo tako da na m adresnih i 2^m informacijskih ulaza multipleksera spojimo n varijabli Booleove funkcije:

$$y = f(x) = f(x_1, x_2, \ldots, x_n).$$

U posebnom slučaju kad je m = n, Booleova funkcija se realizira na osnovi PDNO izraza:

$$f(x) = \bigvee_{j=0}^{2^m-1} m_j(x) \cdot T_j$$

Definicija: Booleovu funkciju m varijabli realiziramo multiplekserom s m adresnih ulaza tako da na adresne ulaze multipleksera dovedemo redom varijable funkcije (MSB na MSB, LSB na LSB), a na informacijske ulaze multipleksera redom vrijednost funkcije (0 ili 1; 0 spajanjem na masu, a 1 spajanjem na V_{cc} preko otpornika). Redoslijed varijabli je važan, da bi redoslijed ulaza multipleksera odgovarao redoslijedu redaka tablice istine, dakle redoslijedu vrijednosti funkcije.



Za opći slučaj vrijedi n > m, pa se gubi strukturni identitet:

$$\bigvee_{j=0}^{2^m-1} m_j(a) u_j = \bigvee_{i=0}^{2^n-1} m_i(x) T_i$$

Potrebno je transformirati desni, veći izraz za PDNO funkcije, da bismo ponovo postigli strukturni identitet. Na raspolaganju nam stoji mogućnost razbijanja PDNO na **preostale funkcije**. Zbog svojstva asocijativnosti konjunkcije, svaki miniterm možemo napisati kao konjunkciju minterma prvih m i minterma preostalih n-m varijabli:

$$m_i(x_1, \ldots, x_n) = m_j(x_1, \ldots, x_m) \cdot m_k(x_{m+1}, \ldots, x_n)$$

U PDNO funkcije grupiramo minterme sa zajedničkim prvim dijelom, te korištenjem svojstva distributivnosti izlučimo prvi, zajednički dio:

$$f(x) = \bigvee_{j=0}^{2^m-1} m_j(x_1, ..., x_m) \cdot \left[\bigvee_{k=0}^{2^{n-m}-1} m_k(x_{m+1}, ..., x_n) \cdot T_{j\cdot 2^{n-m}+k} \right]$$

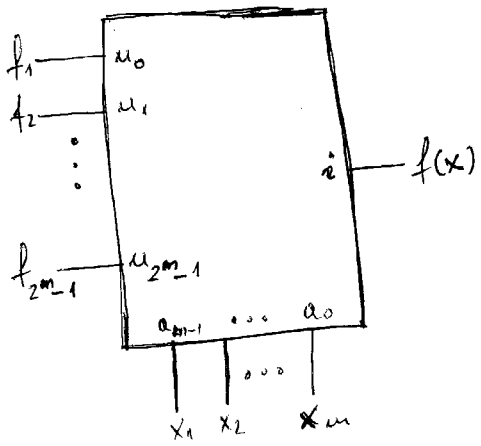
Izraz u zagradi je PDNO preostale funkcije, tj. vrijedi da je:

$$\bigvee_{k=0}^{2^{n-m}-1} m_k(x_{m+1}, ..., x_n) \cdot T_{j\cdot 2^{n-m}+k} = f_j(x_{m+1}, ..., x_n) = f_j(x)$$

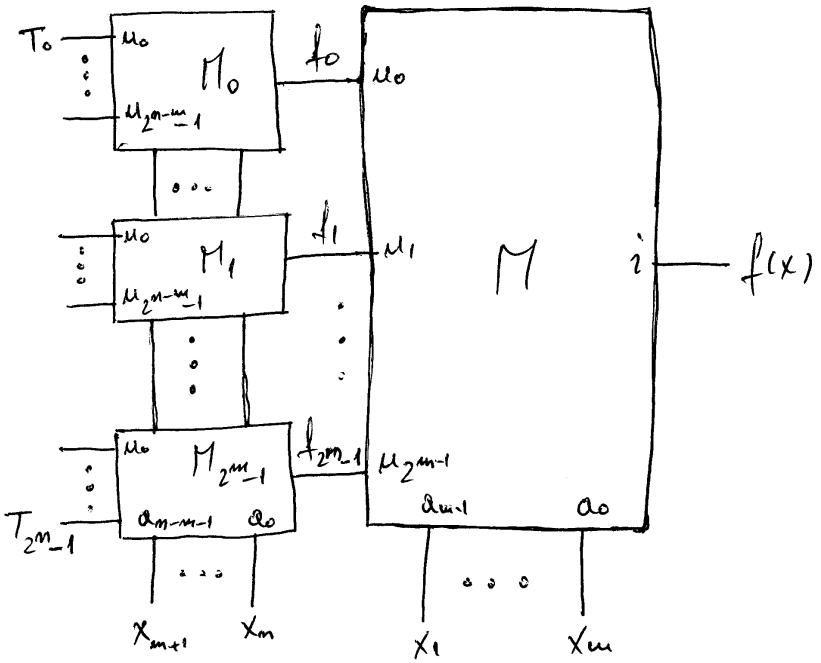
pa Booleovu funkciju možemo pisati kao:

$$f(x) = \bigvee_{j=0}^{2^m-1} m_j(x_1, ..., x_m) \cdot f_j(x_{m+1}, ..., x_n) = \bigvee_{j=0}^{2^m-1} m_j(x) \cdot f_j(x)$$

Definicija: **Booleovu funkciju n varijabli realiziramo multiplekserom s m adresnih ulaza** tako da na adresne ulaze multipleksera dovedmo m varijabli funkcije (to su adresne varijable), a na informacijske ulaze multipleksera preostale funkcije preostalih n-m varijabli funkcije (to su preostale varijable). Redoslijed adresnih varijabli određuje redoslijed preostalih funkcija.



Preostale funkcije treba realizirati posebnim sklopovljem. To mogu biti logička vrata, ali isto tako i multiplekseri. Ako preostale funkcije realiziramo multiplekserima, dobit ćemo strukturu koju nazivamo **multipleksersko stablo**. Ako je stablo s ukupno n adresnih ulaza potpuno, rezultirajući sklop je funkcionalno identičan multiplekseru s n adresnih ulaza.



U primjeni se adresne varijable biraju prema kriteriju minimalnosti sklopa.

Za slučaj realizacije preostalih funkcija **logičkim vratima** adresne varijable ćemo birati tako da ukupni broj logičkih vrata bude minimalan. Funkcije koje realiziramo bez sklopovlja su funkcije jedne varijable.

Za slučaj realizacije preostalih funkcija **multiplekserima** adresne varijable ćemo birati tako da multipleksersko stablo bude minimalno. Pritom nastojimo da što više preostalih funkcija bude funkcija jedne varijable. Sve preostale funkcije su funkcije jedne varijable u **posebnom slučaju n = m +1**.

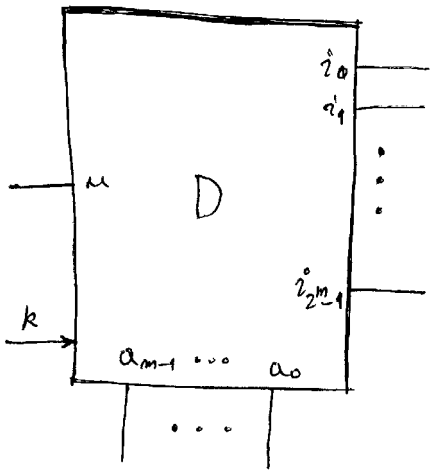
Preostale funkcije se računaju pomoću Veitchevog dijagrama. Izborom jedne adresne varijable, PDNO funkcije i Veitchev dijagram se raspadaju na dva dijela. Svaki dio predstavlja po jednu preostalu funkciju. Izborom slijedećih adresnih varijabli, PDNO funkcije i Veitchev dijagram se raspadaju na 4, 8, itd. dijelova.

Napomena: multiplekser obavlja funkciju izbornika (**selektora**), odnosno obavlja funkciju **paralelno-serijskog pretvornika**.

7. Realizacija Booleovih funkcija demultiplekserom

Demultiplekser je funkcionalni blok (sklop) koji ima:

- m adresnih ulaza: a_{m-1}, a_{m-2}, ..., a_1, a_o
- 1 informacijski ulaz: u
- 2^m informacijskih izlaza i_{2^m-1}, i_{2^m-2}, ...,i_1, i_o
- kontrolne ulaze: k



Definicija: Demultiplekser vrijednost s informacijskog ulaza **u** dovodi na onaj informacijski izlaz **i** kojemu je redni broj j kao kodna riječ prisutan na adresnim ulazima a_{m-1}, a_{m-2}, ..., a_1, a_o pod uvjetom da je funkcija sklopa omogućena aktivnim signalima na kontrolnim ulazima k. Ako je sklop isključen s kontrolnih ulaza, izlazi su prisilno postavljeni na neutralnu vrijednost 0 ili u stanje visoke impedancije.

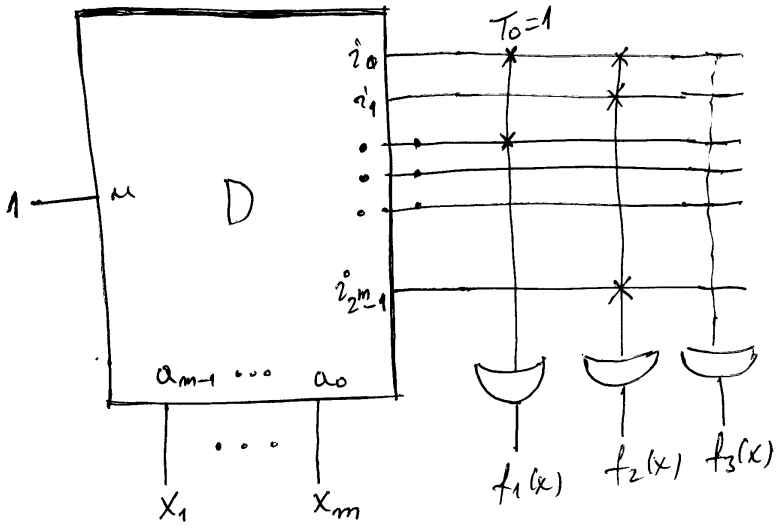
Demultiplekser obavlja funkciju **dekodera**, odnosno obavlja funkciju **serijsko-paralelnog pretvornika.**

Osnova za realizaciju **Booleove funkcije**

$$y = f(x) = f(x_1, x_2, \ldots, x_n)$$

demultiplekserom je ta što demultiplekser ima 2^m informacijskih izlaza, od kojih svaki ostvaruje jedan minterm. Dovoljno je povezati potrebne izlaze demultipleksera na ILI vrata i tako realizirati PDNO.

Definicija: Booleovu funkciju m varijabli realiziramo demultiplekserom s m adresnih ulaza tako da na informacijski ulaz dovedemo 1 (dekoder), na adresne ulaze demultipleksera dovedemo redom varijable funkcije (MSB na MSB, LSB na LSB), a informacijske izlaze demultipleksera za koje vrijednost funkcije jednaka 1 spajamo na izlazna ILI vrata. Redoslijed varijabli je važan, da bi redoslijed izlaza demultipleksera odgovarao redoslijedu redaka tablice istine, dakle redoslijedu vrijednosti funkcije.



$$f(x) = \bigvee_{j=0}^{2^m-1} i_j \cdot T_j$$

Mana realizacije Booleove funkcije demultiplekserom je ta što nema mogućnosti minimizacije funkcije. **Prednost** je što se može realizirati proizvoljan broj funkcija istih varijabli.

Za n = m Booleovu funkciju možemo realizirati pomoću demultipleksera i NI vrata na osnovi izraza:

$$f(x) = \bigvee_{j=0}^{2^m-1} i_j \cdot T_j = \overline{\overline{\bigvee_{j=0}^{2^m-1} i_j \cdot T_j}} = \overline{\overline{\&_{j=0}^{2^m-1} i_j \cdot T_j}}$$

Umjesto ILI ili NI vrata u integriranoj tehnologiji koristi se diodna realizacija ILI vrata. Prednost diodne tehnike je u laganom postizanju potrebnog broja ulaza jednostavnim dodavanjem dioda. Simbolički diodu predstavljamo križićem. Dioda se postavlja za one retke tablice istine za koje je funkcija jednaka jedinici. Za više funkcija istih varijabli imamo polje dioda koje se zove **diodna matrica**.

Za opći slučaj vrijedi n > m i tada možemo pisati:

$$f(x_1, \ldots, x_n) = \bigvee_{j=0}^{2^m-1} i_j \cdot f_j(x_{m+1}, \ldots, x_n)$$

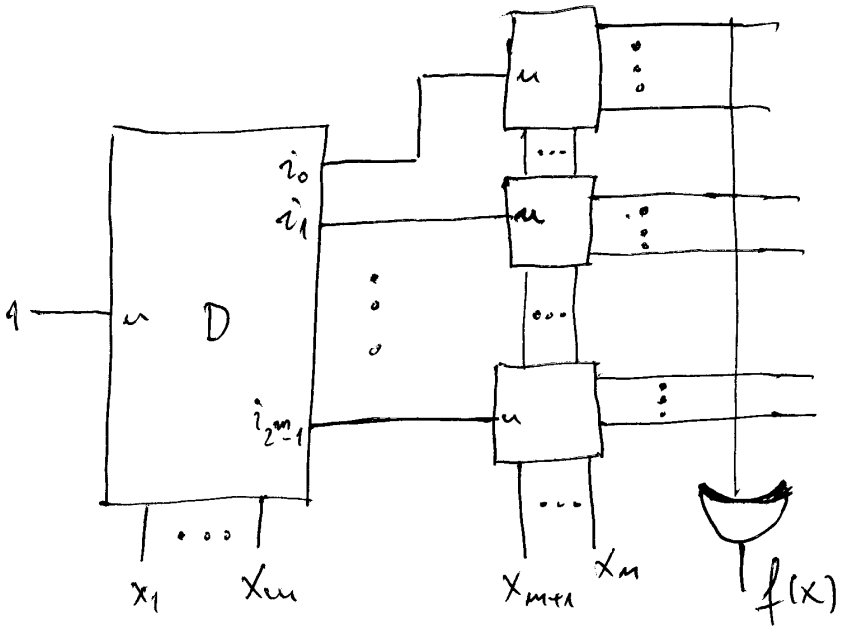
gdje su preostale funkcije opisane izrazom:

$$f_j(x_{m+1}, \ldots, x_n) = \bigvee_{k=0}^{2^{n-m}-1} m_k(x_{m+1}, \ldots, x_n) \cdot T_{j \cdot 2^{n-m}+k}$$

Definicija: **Booleovu funkciju n varijabli realiziramo demultiplekserom s m adresnih ulaza** tako da na adresne ulaze demultipleksera dovedmo m varijabli funkcije (to su adresne varijable), a na informacijskim izlazima demultipleksera aktiviramo sklopove preostalih funkcija preostalih n-m varijabli funkcije (to su preostale varijable). Redoslijed adresnih varijabli određuje redoslijed preostalih funkcija.

Preostale funkcije se realiziraju pomoću **demultipleksorskog stabla.** Iz praktičnih razloga se ne koriste logički sklopovi (nezgrapan sklop, potrebna upotreba I vrata).

Izlaz glavnog demultipleksera se dovodi na informacijski ulaz demultipleksera preostale funkcije čime ga aktiviramo.



Vrijedi da je: $$f(x) = \bigvee_{j=0}^{2^m-1} i_j(x) \cdot \bigvee_{k=0}^{2^{n-m}-1} u \cdot i_k(x) \cdot T_{j \cdot 2^{n-m}+k}$$

Preostale funkcije se računaju pomoću Veitchevog dijagrama. Izborom jedne adresne varijable, PDNO funkcije i Veitchev dijagram se raspadaju na dva dijela. Svaki dio predstavlja po jednu preostalu funkciju. Izborom slijedećih adresnih varijabli, PDNO funkcije i Veitchev dijagram se raspadaju na 4, 8, itd. dijelova.

Minimizacija demultiplekserskog stabla se provodi eliminacijom pojedine grane stabla. To je moguće postići potpunom eliminacijom preostale funkcije, tj. kad je preostala funkcija jednaka konstanti 0 ili 1. Adresne varijable ćemo birati tako da što veći broj preostalih funkcija bude konstanta.

I kod demultipleksera poseban je slučaj za **n = m+1.** Na adresne ulaze dva demultipleksera dovodimo m varijabli funkcije, a na njihove informacijske ulaze dovodimo preostalu varijablu u izvornoj i negiranoj vrijednosti.

Osnovni demultiplekser **m=1** nije potreban jer realizira minterme jedne varijable.

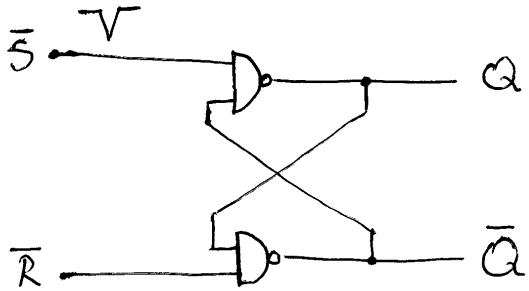
8. Bistabili

Definicija bistabila

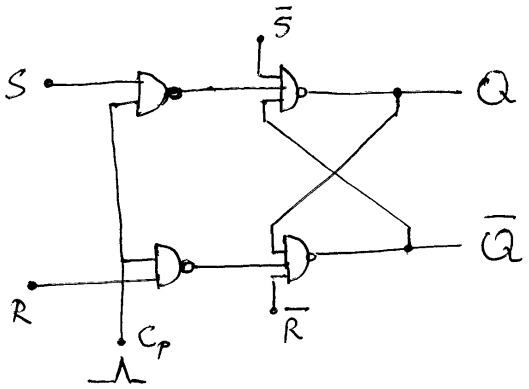
Bistabil je elementarni digitalni sklop za pamćenje jednog od dva stanja: 0 ili 1.

Bistabil spada u skupinu digitalnih sklopova koji ostvaruju povratnu vezu, tj. kod kojih se vrijednost izlaza dovodi na ulaz. To je skop sastavljen od dvaju pojačala (tranzistora) sa snažnom pozitivnom povratnom vezom, takvom da drži izlaze u zasićenju.

Osnovni bistabil je **RS bistabil**. Svi drugi bistabili u svojoj osnovici koriste RS bistabil. RS bistabil možemo konstruirati korištenjem NI vrata.



RS bistabilu se dodaje taktni ulaz.



Ulazi R i S su sinkroni i djeluju na sklop samo u trenutku pozitivnog taktnog implulsa.

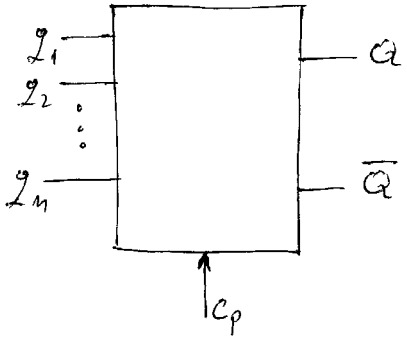
Ulazi $\overline{R}$ i $\overline{S}$ su dodatni asinkroni ulazi izvedeni s osnovnog bistabila - prilikom uključenja napajanja pomoću njih se bistabil može postaviti u poznato početno stanje.

C_p (Clock Pulse) - taktni signal.

Zapisivanje bistabila

Opći bistabil ima:

- informacijske ulaze $q_1, q_2, \ldots, q_n$
- taktni ulaz C_p
- informacijske izlaze Q i $\overline{Q}$



Opći bistabil u trenutku nastupa taktnog signala mijenja svoje stanje na osnovi:

- vrijednosti na ulazu i
- sadašnjeg vlastitog stanja.

Bistabil se može zapisati tablicom prijelaza, potpunom i skraćenom. Tablica prijelaza ima vremenski odnos, pa se vrijednosti s lijeve strane odnose na sadašnji trenutak, a vrijednosti s desne strane tablice na slijedeći trenutak. Oznaka Q^n ovdje znači stanje bistabila u sadašnjem trenutku, a oznaka Q^{n+1} znači stanje bistabila u slijedećem trenutku.

$(q_1$	q_2	....	q_n	$Q)^n$	Q^{n+1}	Q^{n+1}
0	0	....	0	0	0	Q^n
0	0	....	0	1	1	
		.		0	1	$\overline{Q}^n$
		.		1	0	
		.		0	0	0
		.		1	0	
1	1	....	1	0	1	1
1	1	....	1	1	1	

Potpuna tablica prijelaza općeg bistabila

$(q_1 \quad q_2 \quad .... \quad q_n)^n$	Q^{n+1}
0 0 0	Q^n
0 0 1	$\overline{Q}^n$
⋮	
1 1 1	0
1 1 1	1

Skraćena tablica prijelaza općeg bistabila

Bistabil možemo zapisati i **funkcijom prijelaza**:

$$Q^{n+1} = f(Q, q_1, q_2, \ldots, q_n)^n$$

Funkcija prijelaza je slična Booleovoj funkciji, ali za razliku od nje ima vremenskimodnos lijeve i desne strane, pa se s obje strane može pojaviti ista varijabla.

Funkcija prijelaza se često piše u **kanonskom obliku**:

$$Q^{n+1} = \left(G_1 \cdot Q \vee G_2 \cdot \overline{Q} \right)^n$$

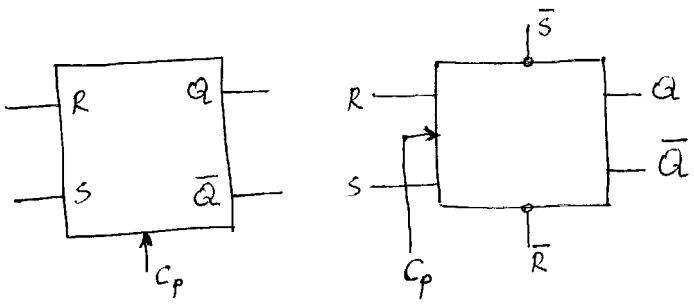
Standardni bistabili se proizvode kao integrirani sklopovi i često se koriste u realizaciji složenih sklopova. To su:

- RS bistabili
- JK bistabili
- T bistabili
- D bistabili

RS bistabil je osnovni bistabil. Može biti u dvije varijante:

- s asinkronim ulazima
- bez asinkronih ulaza

Simboli RS bistabila su:

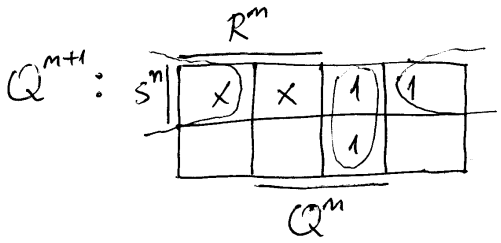


RS bistabil je zadan skraćenom tablicom:

$(R$	$S)^n$	Q^{n+1}
0	0	Q^n
0	1	1
1	0	0
1	1	X

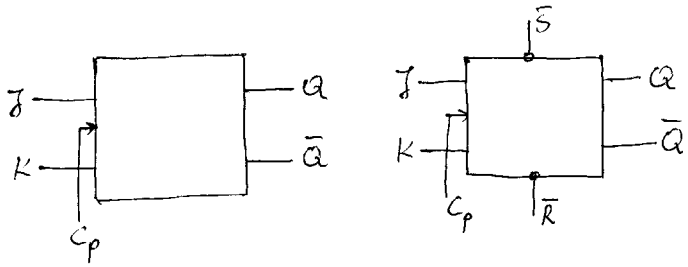
Oznaka X znači da je određeno stanje neodređeno. Stoga je uvjet učinkovitosti korištenja RS bistabila $R \cdot S = 0$, tj. konstruktor sklopa mora osigurati da R i S nikada ne budu istovremeno u jedinici. Tada možemo smatrati da je X jednakovrijedan oznaci redundantnog člana.

Veitchev dijagram RS bistabila glasi:



$$\Rightarrow \qquad G_1 = \overline{R}, \qquad G_2 = S, \qquad Q^{n+1} = \left(G_1 \cdot Q \vee G_2 \cdot \overline{Q} \right)^n = \left(\overline{R} \cdot Q \vee S \cdot \overline{Q} \right)^n$$

JK bistabil je univerzalni bistabil. Simboli su mu:

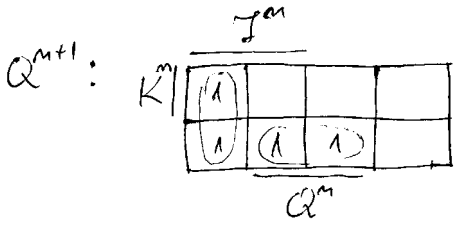


Skraćena tablica JK bistabila glasi:

$(J$	$K)^n$	Q^{n+1}
0	0	Q^n
0	1	0
1	0	1
1	1	$\overline{Q}^n$

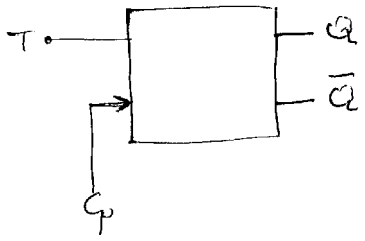
Za JK bistabil se kaže da je **upravljiv s ulaza**, jer u skraćenoj tablici ima sve četiri moguće vrijednosti funkcije.

Veitchev dijagram JK bistabila glasi:



$$\Rightarrow \qquad G_1 = \overline{K}, \qquad G_2 = J, \qquad Q^{n+1} = \left(G_1 \cdot Q \vee G_2 \cdot \overline{Q} \right)^n = \left(\overline{K} \cdot Q \vee J \cdot \overline{Q} \right)^n$$

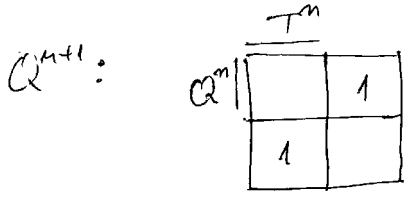
T bistabil zadržava ili mijenja stanje. Simbol mu je:



Skraćena tablica T bistabila je:

T^n	Q^{n+1}
0	Q^n
1	$\overline{Q}^n$

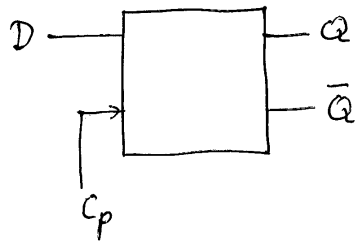
Veitchev dijagram T bistabila glasi:



$$\Rightarrow \qquad G_1 = \overline{T}, \qquad G_2 = T, \qquad Q^{n+1} = \left(G_1 \cdot Q \vee G_2 \cdot \overline{Q} \right)^n = \left(\overline{T} \cdot Q \vee T \cdot \overline{Q} \right)^n$$

G_1 i G_2 su komplementarni što ograničava mogućnosti rada s funkcijom prijelaza.

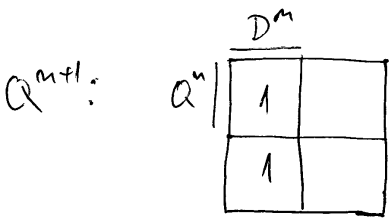
D bistabil pamti 0 ili 1 izravno s ulaza. Za D bistabil kažemo da pamti podatak i da realizira kašnjenje. Simbol mu je:



Skraćena tablica D bistabila je:

D^n	Q^{n+1}
0	0
1	1

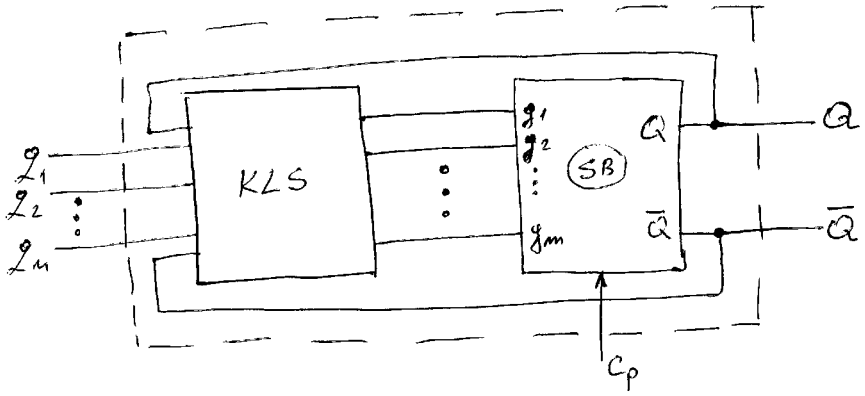
Veitchev dijagram D bistabila glasi:



$$\Rightarrow \qquad G_1 = G_2 = D, \qquad Q^{n+1} = \left(G_1 \cdot Q \vee G_2 \cdot \bar{Q} \right)^n = \left(D \cdot Q \vee D \cdot \bar{Q} \right)^n = D^n$$

Model realizacije općeg bistabila

Opći bistabil realiziramo korištenjem standardnog bistabila i **kombibacijske logičke strukture** (KLS).



Izlazi standardnog bistabila su ujedno i izlazi općeg bistabila:

$$Q_{OB}^{n+1} = Q_{SB}^{n+1} \quad , \quad Q_{OB}^{n} = Q_{SB}^{n}$$

Nakon izbora standardnog bistabila, **sinteza** općeg bistabila se svodi na sintezu njegove KLS.

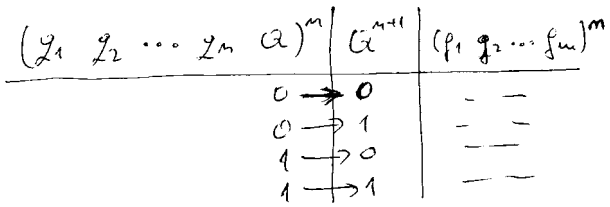
Postupci realizacije općeg bistabila su:

- postupak rekonstrukcije,
- postupak izjednačavanja,
- postupak za D bistabil.

Postupak rekonstrukcije pogodan je za sve bistabile, a jedini je upotrebljiv za RS bistabile (zbog uvjeta RS=0) i T bistabile (zbog komplementarnosti G_1 i G_2).

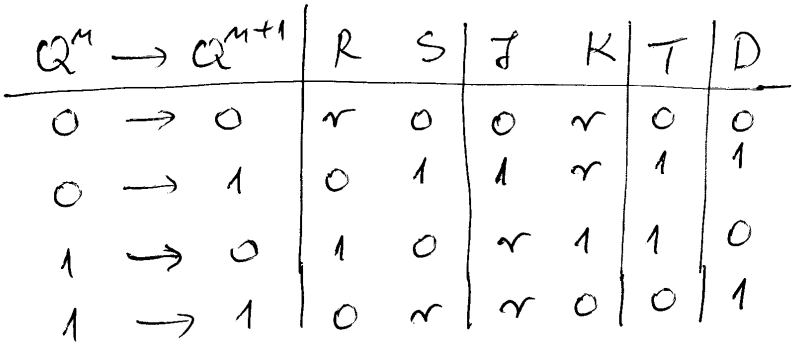
Postupak se provodi tako da:

- potpunoj tablici prijelaza općeg bistabila s desne strane dopišemo rekonstruirane ulaze u standarsdni bistabil da bi on obavljao upravo te prijelaze,



- lijeva i dopisana desna strana tablice čine tablicu istine za KLS,
- minimiziramo i realiziramo Booleovu funkciju čime je završena sinteza općeg bistabila.

Rekonstruirane vrijednosti ulaza standardnih bistabila su dane tablicom:



Primjer: konstrukcija JK bistabila korištenjem RS bistabila i NI vrata postupkom rekonstrukcije.

Postupak izjednačavanja pogodan je za JK bistabile.

Izjednačimo funkcije prijelaza standardnog (JK) i općeg bistabila:

$$Q_{SB}^{n+1} = Q_{OB}^{n+1} \quad \Rightarrow \quad \left(\overline{K} \cdot Q \vee J \cdot \overline{Q} \right)^n = \left(G_1 \cdot Q \vee G_2 \cdot \overline{Q} \right)^n$$

$$\Rightarrow \quad G_1 = \overline{K}, \quad G_2 = J \quad \Rightarrow \quad K = \overline{G}_1, \quad J = G_2$$

Dovoljno je u Veitchev dijagram upisati vrijednosti Q^{n+1} općeg bistabila te minimizirati funkcije $\overline{G}_1$ i G_2.

Postupak za D bistabil je istovremeno postupak rekonstrukcije i postupak izjednačavanja. Zasniva se na funkciji prijelaza:

$$Q^{n+1} = D^n$$

Potpuna tablica općeg bsitabila numerički je identična tablici istine za KLS. Postupak se provodi tako da po potrebi minimiziramo funkciju D^n ili negiranu funkciju $\overline{D}^n$.

Naputak: Radi jasnoće ovog teorijskog pitanja proučiti primjere u knjizi.

Mali dodatak

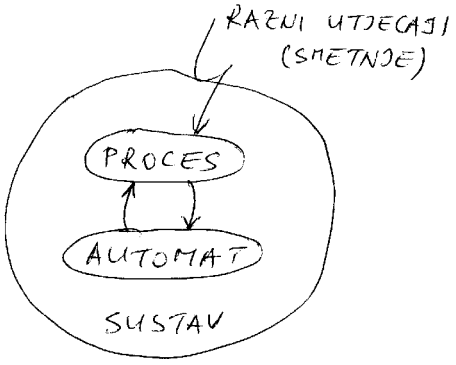
Složene strukture s bistabilima su:

- **registar** - više D bistabila sa zajedničkim taktnim ulazom
- **pomačni registar** - više D bistabila sa zajedničkim taktnim ulazom, izlaz jednog ide na ulaz idućeg
- **brojilo** (counter) - više T ili JK bistabila - izlaz jednog taktni ulaz idućeg

9. Zadavanje digitalnih automata (DA)

Digitalni automat je univerzalni sekvencijalni sklop čije ponašanje ovisi o sadašnjem i prethodnim ulaznim stanjima - događajima.

Automat i **proces** čine **sustav s upravljanjem**. Na proces djeluje **okolina** remeteći njegov rad, a automat pokušava održati proces u optimalnim uvjetima rada kako bi mogao ostvariti postavljenu **funkciju cilja**.



Da bi upravljanje moglo funkcionirati proces mora biti **mjerljiv** i **upravljiv**.

Govorimo o digitalnim automati koji su:

- **konačni** (konačan broj stanja, konačna memorija)
- **digitalni** (imaju digitalni ulaz i izlaz)
- **diskretni** (rade u diskretnom vremenu)
- **determinirani** (jednoznačno obavljaju svoju funkciju)
- **specificirani** - *potpuno* (mogući svi nizovi ulaznih događaja)
  - *nepotpuno* (mogući samo neki nizovi ulaznih događaja)
- **sinkroni** (diskretno vrijeme definirano taktnim signalom)

Apstraktni automat je zadan petorkom:

$$A = \langle U,\ I,\ S,\ \delta, \lambda \rangle$$

gdje je: U - skup **ulaznih simbola** ili **ulazni alfabet**,

I - skup **izlaznih simbola** ili **izlazni alfabet**,

S - skup **unutarnjih stanja** automata,

δ - **funkcija prijelaza**,

λ - **funkcija izlaza**.

Dva ulazna simbola se ne mogu pojaviti istovremeno kao ni dva izlazna simbola.

Ulazna sekvenca ili **ulazna riječ** je vremenski niz ulaznih simbola koji se pojavljuju u uzastopnim diskretnim vremenima.

Izlazna sekvenca ili **izlazna riječ** je vremenski niz izlaznih simbola koji se pojavljuju u uzastopnim diskretnim vremenima.

Preslikavanja funkcija prijelaza i izlaza

Funkcija prijelaza δ određuje slijedeće stanje automata na osnovi sadašnjeg stanja i sadašnjeg ulaza:

$$\delta: \quad s^{n+1} = \delta(s,\ u)^n \qquad S \times U \to S$$

Funkcija izlaza λ određuje sadašnji izlaz automata. Razlikujemo Mealy i Moore model automata.

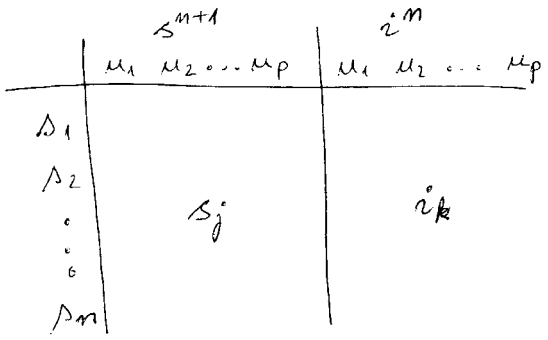
$$\lambda: \quad i^n = \begin{cases} \lambda(s,\ u)^n & \text{Mealy} \quad S \times U \to I \\ \lambda(s)^n & \text{Moore} \quad S \to I \end{cases}$$

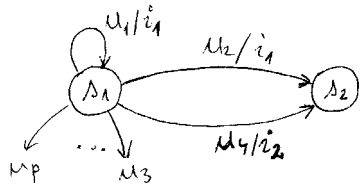
Mealy automat već u diskretnom periodu u kojem primi ulazni simbol generira izlazni simbol. **Moore automat** u diskretnom periodu u kojem primi ulazni simbol najprije prijeđe u stanje pa tek u slijedećem diskretnom periodu generira izlazni simbol. Stoga Moore automatu izlaz za jedan diskretni period kasni za Mealy automatom, te redovito za istu funkciju treba veći broj stanja.

Zapisivanje automata

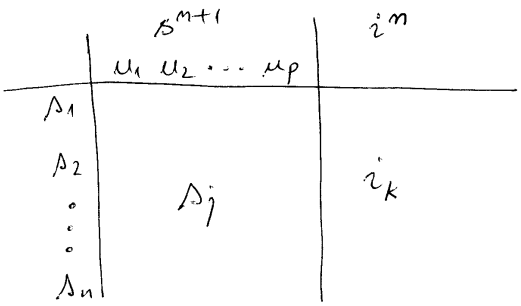
Automat se zapisuje **tablicom prijelaza i izlaza** te (usmjerenim) **grafom**.

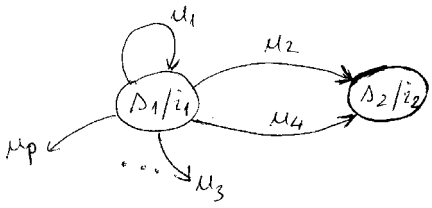
Tablica prijelaza i izlaza te graf Mealy automata:





Tablica prijelaza i izlaza te graf Moore automata:





Pristup početnom zadavanju DA

Automat početno možemo zadati na jedan od tri moguća načina:

- kao transformator sekvence,
- kao akceptora sekvence,
- korak po korak.

Sva tri pristupa za neki problem rezultiraju istim automatom.

Zadavanje automata kao **transformatora sekvence** provodi se kroz postavljanje pravila o transformaciji ulazne na izlaznu sekvencu. Ova pravila spadaju u skupinu **matematičkih gramatika.** Govorimo o preslikavanju ulazne sekvence na izlaznu.

Zadavanje automata kao **akceptora sekvence** provodi se kroz prepoznavanje sekvenci koje izazivaju pojavu nekog simbola na izlazu. Koriste se postupci zadavanja kao što je npr. **jezik regularnih izraza.** Govorimo o transformaciji ulazne sekvence na izlazni simbol.

Zadavanje automata postupkom **korak po korak** provodi se kroz analizu svakog mogućeg para stanje - ulazni simbol. Izravno se crta graf automata, po mogućnosti u obliku potpunog stabla. Za velike automate stablo je preveliko pa se koriste reducirani grafovi. Govorimo o transformaciji ulaznog simbola, uz stanje, na izlazni simbol.

Rad automata znatno ovisi o karakteru ulazne sekvence. Imamo dvije vrste ulaznih sekvenci:

- sekvence bez strukture
- sekvence sa strukturom

Sekvenca bez strukture je beskonačna sekvenca ulaznih simbola kod koje se tražena sekvenca može pojaviti u proizvoljnom trenutku.

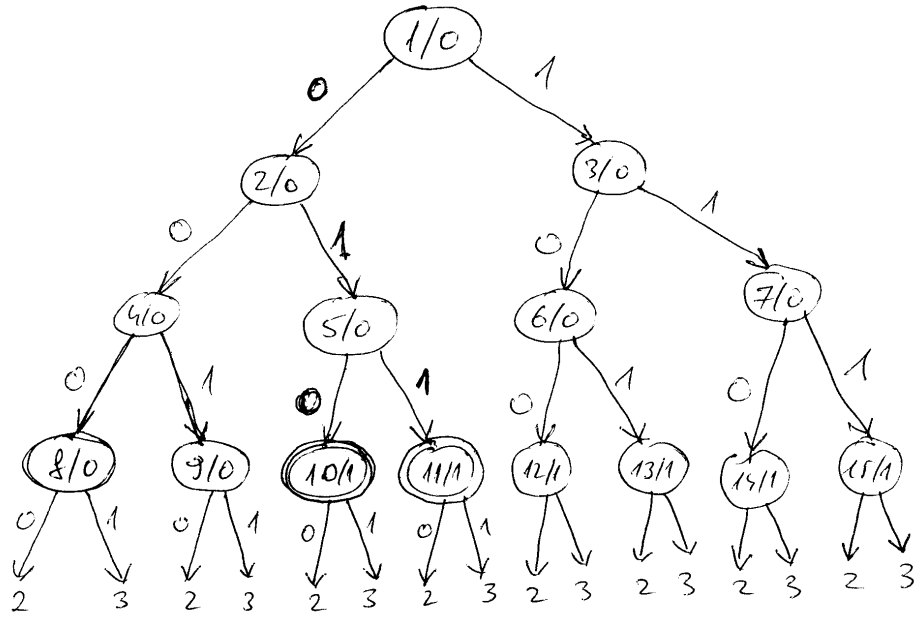
Sekvenca sa strukturom je beskonačna sekvenca ulaznih simbola koja se sastoji od beskonačnog niza konačnih sekvenci. Tražena sekvenca se ovdje može dogoditi nakon što se dogodi prethodna. Konačne sekvence mogu biti iste duljine ali ne moraju.

Metoda potpunog stabla

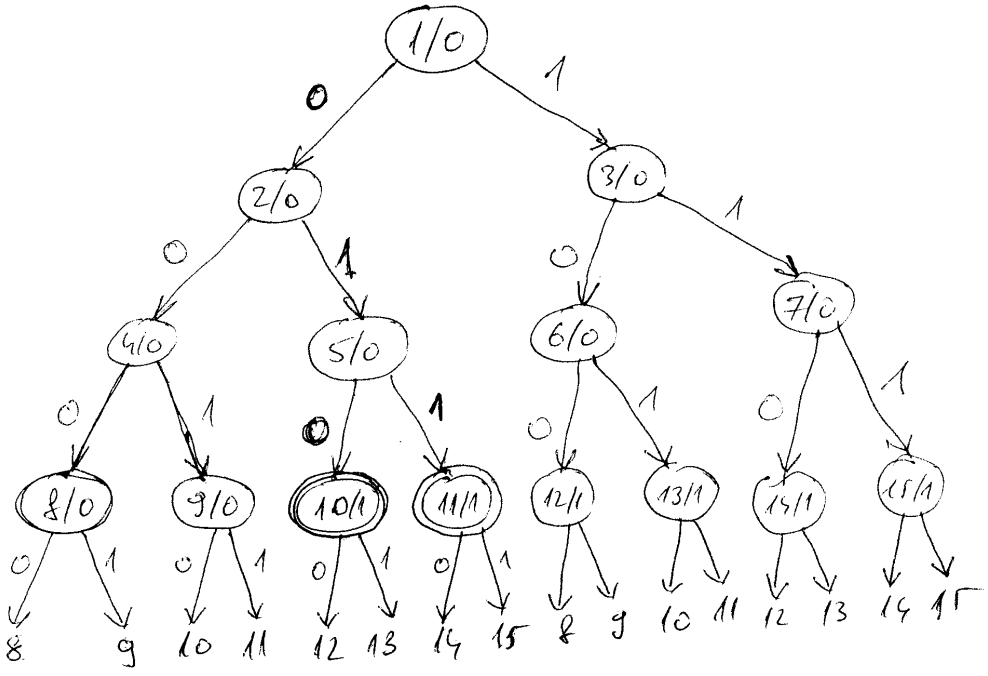
Kod postupka korak po korak crta se graf u obliku potpunog stabla. Ovisno o vrsti sekvence i automata imamo četiri mogućnosti.

Neka su ualzni i izlazni skup $U = I = \{0,1\}$ te neka automat među sekvencama od tri bita jedinicom detektira 010 i 011 te ako prepozna neku od njih na izlazu treba dati 1, a u svim ostalim slučajevima 0. (Napomena: *Ovo su kodirani ulazi i izlazi u odnosu na primjer u knjizi - radi preglednosti.*)

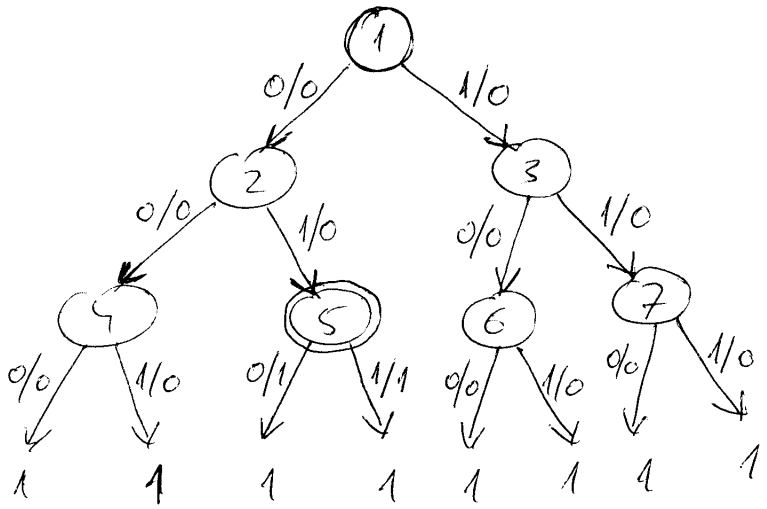
Potpuno stablo za **Moore** automat i sekvencu **sa strukturom**:



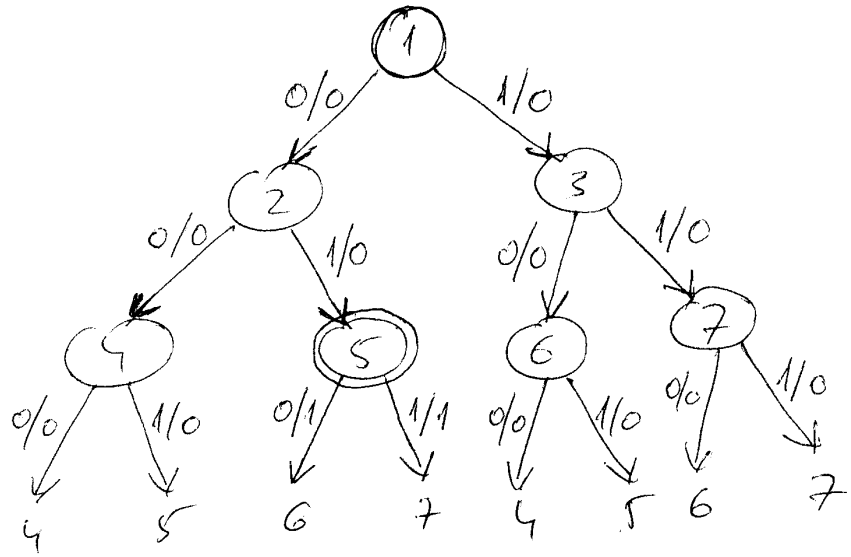
Potpuno stablo za **Moore** automat i sekvencu **bez strukture**:



Potpuno stablo za **Mealy** automat i sekvencu sa strukturom:



Potpuno stablo za **Mealy** automat i sekvencu bez strukture:



Važni pojmovi:

Akceptorska stanja - stanja u kojima je sekvenca prepoznata i aktivni izlazni simbol generiran.

Primitivna tablica (početna tablica) automata - tablica prijelaza i izlaza koja izravno slijedi iz potpunog stabla.

Tehnika pisanja regularnih izraza

Promatramo automat kao akceptor sekvence. Ukaznu sekvencu opisujemo tako da izdvojimo: - dio sekvence koji nije tražena sekvenca,

- dio sekvence koji je tražena sekvenca.

Koristimo različite tehnike pisanja regularnih izraza za sekvence sa strukturom i sekvence bez strukture.

Ako imamo više akceptorskih izlaznih simbola, pišemo zasebni regularni izraz za svaki od tih simbol. Zajednički dio ne uključuje tražene sekvence.

Pisanje regularnog izraza za sekvencu sa strukturom:

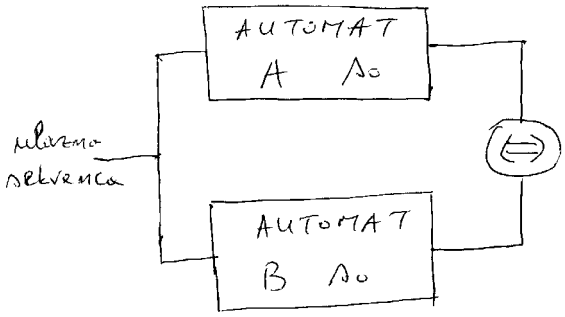
- iteracijskom zagradom obuhvatimo sekvence koje **nisu** tražene,
- običnom zagradom obuhvatimo sekvence koje su tražene za promatrani akceptorski (aktivni) izlazni simbol,
- pišemo više izraza ako je više akceptorskih izlaznih simbola.

Pisanje regularnog izraza za sekvencu bez strukture:

- iteracijskom zagradom obuhvatimo sekvence do duljine tražene sekvence koje **nisu** početak tražene sekvence,
- upišemo prvi simbol tražene sekvence,
- iteracijskom zagradom obuhvatimo ostatke sekvenci nakon prvog simbola koje **nisu** početak tražene sekvence, ali čiji je zadnji simbol prvi simbol tražene sekvence (ako takvih ima),
- upišemo drugi simbol tražene sekvence,
- iteracijskom zagradom obuhvatimo ostatke sekvenci nakon drugog simbola koje **nisu** početak tražene sekvence, ali čija su zadnja dva simbola prva dva simbola tražene sekvence (ako takvih ima),
- upišemo treći simbol tražene sekvence,
- itd. do ispisa tražene sekvence.

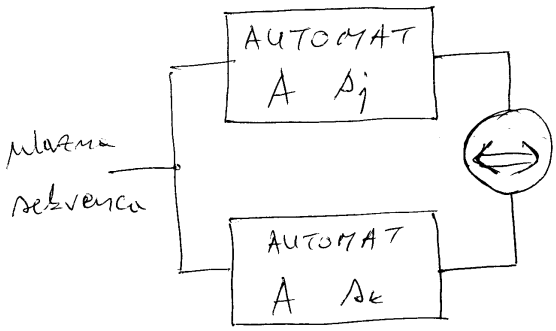
10. Minimizacija digitalnih automata

Definicija: Dva automata (A i B) su **ekvivalentna** ako počnu raditi iz početnog stanja s_o te za istu **proizvoljnu** sekvencu na ulazu dadu **istu** sekvencu na izlazu.



Ekvivalentni automati se mogu **razlikovati** u **skupu stanja**, a minimalan je onaj s najmanjim brojem stanja.

Definicija: Dva stanja automata A $\left(s_j \text{ i } s_k\right)$ su **ekvivalentna** ako automat počne raditi iz jednog pa iz drugog stanja, te za istu **proizvoljnu** sekvencu na ulazu dade u oba testa **istu** sekvencu na izlazu.



Definicija: Nužan uvjet ekvivalencije kaže da su dva stanja potencijalno ekvivalentna ako su im reci u tablici izlaza automata identični.

Definicija: Dovoljan uvjet ekvivalencije kaže da su dva stanja ekvivalentna ako je zadovoljen nužan uvjet te ako su im reci u tablici prijelaza automata isti.

Da bi se dobio minimalan automat, dovoljno je otkriti podskupove ekvivalentnih stanja te sva stanja nekog podskupa zamijeniti s jednim stanjem.

Postupci minimizacije automata su:

- postupak primitivne tablice,
- Hufmann-Mealy algoritam,
- Paul-Unger algoritam (tablica implikanata).

Postupak primitivne tablice provodi se u koracima:

1. Traže se stanja s istim recima u tablici prijelaza i izlaza,
2. Precrtaju se svi reci ekvivalentnih stanja osim jednog,
3. Sve oznake precrtanih stanja zamijenimo oznakom onog neprecrtanog,
4. Nastavi se postupak dok ima ekvivalentnih stanja.

Kod **Hufmann-Mealy algoritma** polazimo od pretpostavke da su sva stanja sa zadovoljenim nužnim uvjetom ekvivalentna. Koristeći kriterij istih izlaza formiramo primarne klase (skupove) ekvivalentnosti za koje ispitujemo zatvorenost. Kao rezultat dobijemo skup zatvorenih klasa stanja koje zamjenjujemo s po jednim stanjem. Dobiveni automat je minimalan.

Klasa je zatvorena ako sva stanja klase imaju iste prijelaze u klase ili ako klasa sadrži samo jedno stanje. U suprotnom klasa je otvorena.

Kod **Paul-Unger algoritma** koristimo metodu **tablice implikanata** kod koje skupove S_p formiramo sistematski od po dva stanja iz skupa S, tj. ispisujemo sve parove stanja. Koristi se trokutasta matrica sa (n-1) redaka i stupaca.

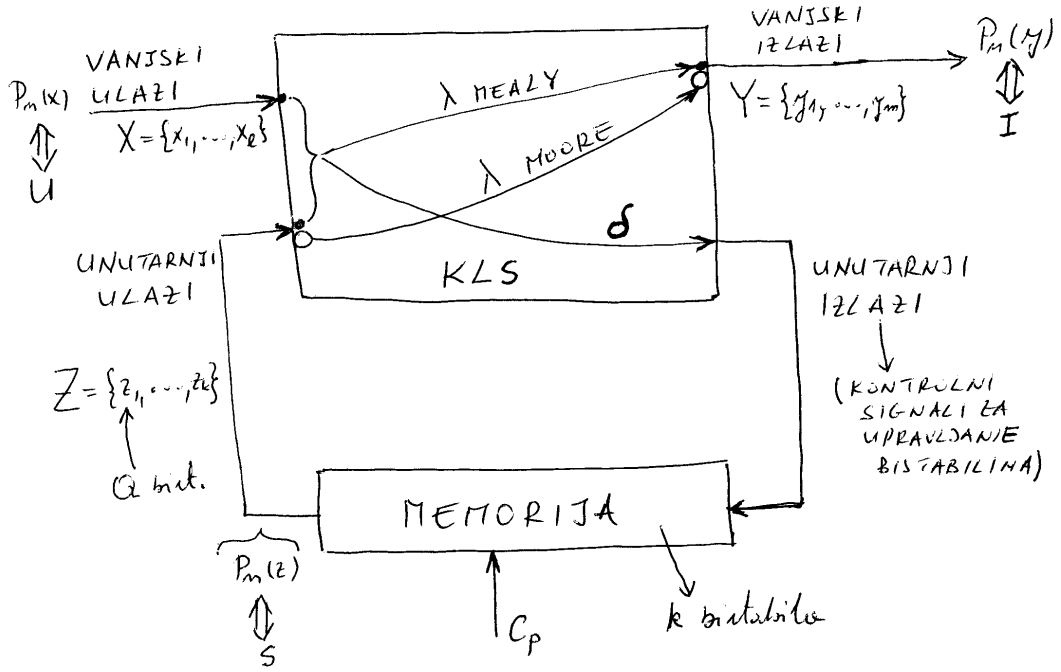
11. Strukturna sinteza digitalnih automata

Strukturna sinteza automata polazi od minimalnog apstraktnog automata te ima slijedeće korake:

1. Kodiranje stanja, ulaza i izlaza,
2. Uvrštavanje kodova, prepoznavanje:
   - tablica prijelaza za pojedine bistabile,
   - tablica istine za izlazne varijable,
3. Realizacija automata: - bistabilima i logičkim vratima,
   - MD strukturom i D bistabilima (MDD)

Model realizacije automata

Model realizacije automata se sastoji od kombinacijske logičke strukture (KLS) i memorije.



Memorija automata sastoji se od k bistabila. Ona ima k ili 2k ulaza, ovisno o vrsti bistabila. Stanje memorije se mijenja u trenutku nastupa aktivnog dijela taktnog signala, koji je zajednički za sve bistabile. Memorija automata je zapravo **registar**.

Znanje o radu automata sadržano je u **KLS**. KLS na osnovi vanjskih i unutarnjih ulaza generira: - kontrolne signale na ulazu bistabila i time upravlja prijelazom memorije,

- kodne riječi varijabli y na vanjskim izlazima.

Veza modela s apstraktnim automatom ostvaruje se kroz kodiranje ulaznih i izlaznih simbola te kroz kodiranje stanja automata.

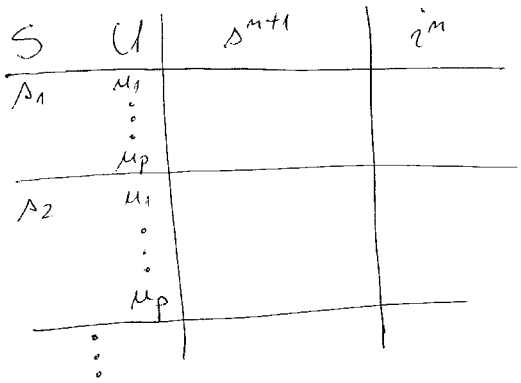
Kodiranje DA

Kodiranje **ulaznih i izlaznih simbola** ovisi o okolini automata. Preko njih automat komunicira s procesom pa kodovi moraju biti usklađeni. Stoga su često ulazni i izlazni kod zadani, tj. često nemamo slobodu kodiranja.

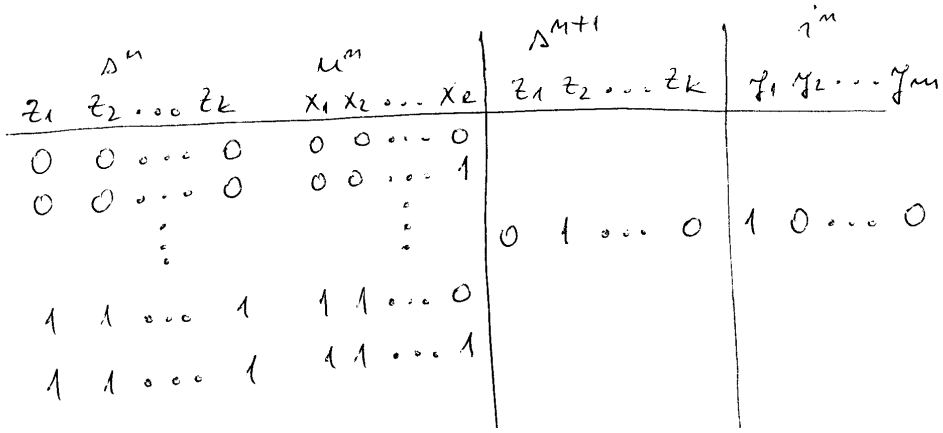
Kodiranje **stanja** je unutarnja stvar automata. Stanje automata predstavlja znanje o prethodnim događajima na ulazu. Stanja se kodiraju minimalnim brojem bistabila, odnosno **koncentriranim kodom**. Ne postoji egzaktan postupak kodiranja koji bi rezultirao minimalnim sklopom. Koristi se kodiranje po kriteriju susjednosti. To znači da stanjima između kojih postoji mogućnost prijelaza nastojimo dodijeliti susjedne kompleksije, ili kompleksije sa što manjom distancom. Kodiranje se vrši primjenom Veitchevog dijagrama.

Kodovi dobiveni postupkom kodiranja uvrštavaju se u tablicu prijelaza i izlaza automata. Koristi se jednodimenzionalna tablica. U toj tablici redom nabrajamo parove stanje-ulazni simbol, dakle članove Kartezijevog produkta $S \times U$.

Jednodimenzionalna tablica apstraktnog automata ima oblik:



Jednodimenzionalna tablica nakon upisa kodova ima oblik:

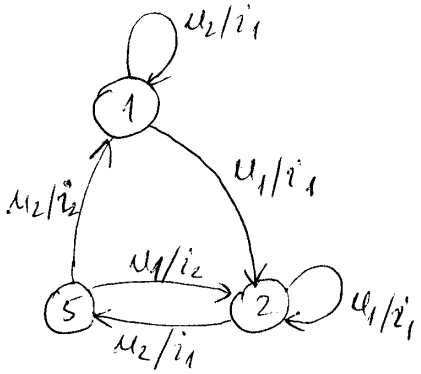


Svaki redak opet pripada paru stanje - ulazni simbol, ali su sada oni poredani po kriteriju prirodnog binarnog niza kodnih riječi.

S desne strane imao najprije dio koji se odnosi na prijelaze. To su kodne riječi stanja bistabila, ali u slijedećem trenutku. Sasvim desno su kodne riječi izlaza u sadašnjem trenutku.

Primjer: Dobivanje kodirane tablice prijelaza i izlaza minimalnog Mealy automata

S^m	S^{m+1}		y^m	
	u_1	u_2	u_1	u_2
1	2	1	1	1
2	2	5	1	1
5	2	1	2	2



U	x_1
u_1	0
u_2	1

I	z_1
i_1	0
i_2	1

z_2 \ z_1		
	5	2
		1

1 = 0 0
2 = 0 1
5 = 1 1

	$(z_1$	z_2	$x_1)^m$	$(z_1$	$z_2)^{m+1}$	y^m
S_1	0	0	0	0	1	0
	0	0	1	0	0	0
S_2	0	1	0	0	1	0
	0	1	1	1	1	0
R	1	0	0	R	R	R
	1	0	1	R	R	R
S_5	1	1	0	0	1	1
	1	1	1	0	0	1

Realizacija DA

Realizacija automata provodi se kroz sintezu sklopovlja prema osnovnom modelu.

Sinteza memorije se provodi izborom vrste i izračunavanjem broja bistabila (po kriteriju jednoznačnosti kodiranja stanja automata).

Sinteza KLS-a provodi se pomoću potpunih tablica prijelaza bistabila i tablica istine za izlazne varijable.

KLS automata se sastoji od:

- KLS-a pojedinih (općih) bistabila,
- KLS-a kojima realiziramo Booleove funkcije izlaznih varijabli.

U samoj realizaciji možemo koristiti:

- integrirane krugove niske razine integracije (**bistabile i logička vrata**)
- integrirane krugove srednje razine integracije (**multipleksere, demultipleksere i registre - MDD**)
- programibilne logičke strukture koje u izlaznoj makro-ćeliji imaju D bistabil (GAL)

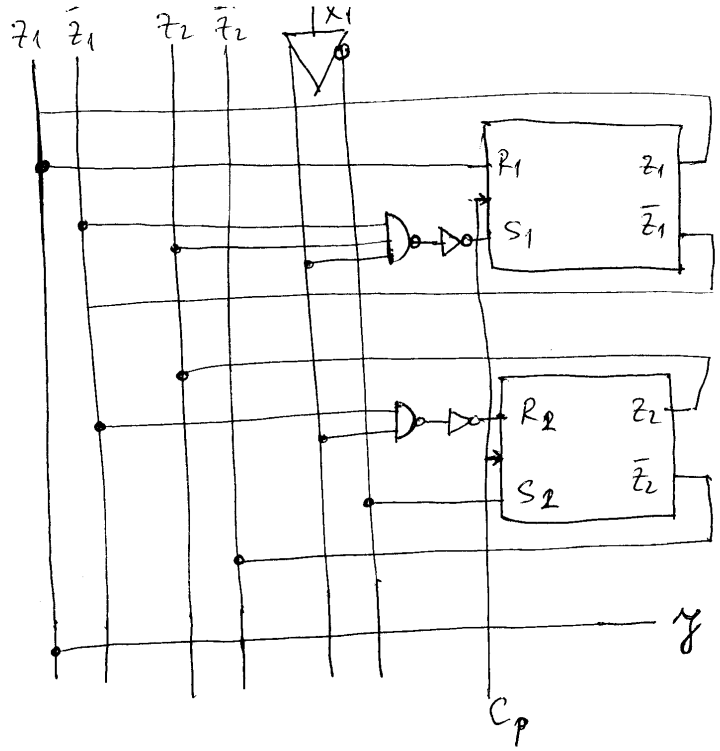
Kod realizacije automata **bistabilima i logičkim vratima** biramo NI ili NILI vrata neki od standardnih bistabila. Prihvatljivi su svi standardni bistabili iako RS i JK imaju dva ulaza. Kompleksnost sklopa najviše ovisi o mogućnosti minimizacije, tj. o načinu kodiranja stanja, ulaznih simbola i izlaznih simbola.

Primjer: Realizirajmo prethodno zadani automat **NI vratima i RS bistabilima**. Zbog NI vrata minimiziramo izvorne funkcije, a zbog RS bistabila koristimo postupak rekonstrukcije.

	$(z_1\ z_2\ x_1)^n$	$(z_1\ z_2)^{n+1}$	y^n	$(R_1\ S_1)^n$	$(R_2\ S_2)^n$
S_1	0 0 0	0 1	0	r 0	0 1
	0 0 1	0 0	0	r 0	r 0
S_2	0 1 0	0 1	0	r 0	0 r
	0 1 1	1 1	0	0 1	0 r
R	1 0 0	R R	R	r r	r r
	1 0 1	R R	R	r r	r r
S_5	1 1 0	0 1	1	1 0	0 r
	1 1 1	0 0	1	1 0	1 0

Nakon minimizacije pomoću Veitchevog dijaframa dobije se:

$$R_1 = z_1,\ S_1 = \overline{\overline{z_1 \cdot z_2 \cdot x_1}},\ R_2 = \overline{\overline{z_1 \cdot x_1}},\ S_2 = \overline{x}_1,\ y = z_1$$



Ovo je shema automata realiziranog NI vratima i RS bistabilima.

Multipleksersko-demultiplekserska struktura s D bistabilima (MDD)

Upotreba D bistabila u MDD strukturi važna je iz tri razloga:

1. D bistabili su registri srednje razine integracije,
2. D bistabil ima samo jedan ulaz, a MD struktura (ROM) ostvaruje minterme funkcije, tj. ne omogućuje minimizaciju,
3. Izravno se popunjava sadržaj ILI matrice MD strukture.

Veličina multipleksera i demultipleksera određuje se prema kriteriju kvadratičnosti (diodne) matrice.

Ukupni broj multipleksera M je:

$$M = k + p$$

gdje je: k - ukupni broj bistabila,

p - ukupni broj izlaznih varijabli.

Ukupni broj adresnih ulaza u strukturu **n** je:

$$n = d + m = k + \ell$$

gdje je: d - ukupni broj adresnih ulaza demultipleksera,

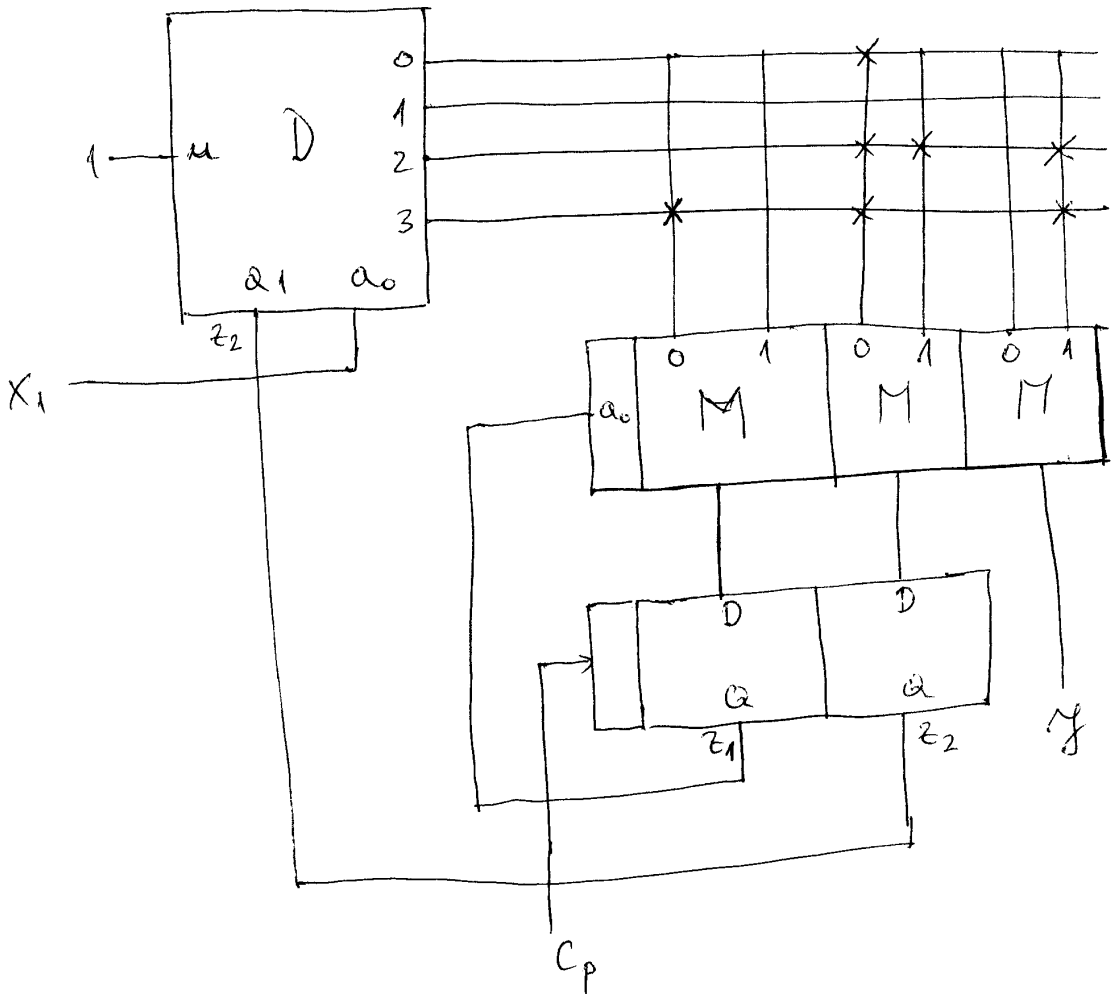
m - ukupni broj adresnih ulaza multipleksera,

ℓ - ukupni broj ulaznih varijabli.

Uvjet kvadratičnosti je isti kao za MD strukturu: $2^d \approx M \cdot 2^m$.

Radi jednostavnosti postupka, prvih m varijabli dovodimo na adresne ulaze multipleksera, a preostalih n-m varijabli na ulaze demultipleksera. Tada će se stupci MDD strukture poklapati sa stupcima kodirane tablice prijelaza i izlaza.

Primjer:



12. Regularni izrazi

Regularni izrazi su **konačni** algebarski izrazi u okvirima **algebre događaja**.

Algebra događaja

Ulazne sekvence u automat nazivamo **nizovima događaja**.

Događaj može biti elementarni i složeni (niz elementarnih).

Algebra događaja uzima u obzir vremenske odnose te uvodi **algebarske operacije** među događajima.

Algebarske operacije među događajima su:

- **disjunkcija** $R = R_1 \vee R_2$

  (vrijedi svojstvo komutativnosti: $R_1 \vee R_2 = R_2 \vee R_1$),

  (traje koliko traje disjunktivni član koji se desio)

- **konjunkcija** $R = R_1 \,\&\, R_2$

  (ne vrijedi svojstvo komutativnosti: $R_1 \cdot R_2 \neq R_2 \cdot R_1$),

  (traje koliko traju svi članovi)

- **negacija** $R = \overline{R}_1$

  (potpuni skup događaja može biti beskonačan - ne koristi se u praksi)

- **iteracija** (iteracijska zagrada) $R = R_1^* = (R_1)^*$

  (događaj koji se desi kad se događaj pod iteracijom desi proizvoljan broj puta, čak i ni jednom)

  $\left(R_1\right)^* = \left(\wedge \vee R_1 \vee R_1R_1 \vee R_1R_1R_1 \vee \ldots\right)$

  $\wedge$ - oznaka za praznu sekvencu

Veza automata i algebre događaja:

- Ulazni simbol automata je elementarni događaj,
- Sekvenca ulaznih simbola je složeni događaj.

Pristup zadavanju automata RI

- promatramo automat kao akceptor sekvence,
- trebamo opisati traženu sekvencu,
- automat mora odbaciti sekvence koje nisu tražene,
- treba opisati i sekvence koje nisu tražene.

Tehnika pisanja regularnih izraza

Promatramo automat kao akceptor sekvence. Ukaznu sekvencu opisujemo tako da izdvojimo: - dio sekvence koji nije tražena sekvenca,

- dio sekvence koji je tražena sekvenca.

Koristimo različite tehnike pisanja regularnih izraza za sekvence sa strukturom i sekvence bez strukture.

Ako imamo više akceptorskih izlaznih simbola, pišemo zasebni regularni izraz za svaki od tih simbola. Zajednički dio ne uključuje tražene sekvence.

Pisanje regularnog izraza za sekvencu sa strukturom:

- iteracijskom zagradom obuhvatimo sekvence koje **nisu** tražene,
- običnom zagradom obuhvatimo sekvence koje su tražene za promatrani akceptorski (aktivni) izlazni simbol,
- pišemo više izraza ako je više akceptorskih izlaznih simbola.

Pisanje regularnog izraza za sekvencu bez strukture:

- iteracijskom zagradom obuhvatimo sekvence do duljine tražene sekvence koje **nisu** početak tražene sekvence,
- upišemo prvi simbol tražene sekvence,
- iteracijskom zagradom obuhvatimo ostatke sekvenci nakon prvog simbola koje **nisu** početak tražene sekvence, ali čiji je zadnji simbol prvi simbol tražene sekvence (ako takvih ima),
- upišemo drugi simbol tražene sekvence,
- iteracijskom zagradom obuhvatimo ostatke sekvenci nakon drugog simbola koje **nisu** početak tražene sekvence, ali čija su zadnja dva simbola prva dva simbola tražene sekvence (ako takvih ima),
- upišemo treći simbol tražene sekvence,
- itd. do ispisa tražene sekvence.

Definicija mjesta u RI

Osnovna mjesta su ona koja se nalaze s lijeve strane obične i iteracijske zagrade i s desne strane slova.

Predosnovna mjesta su ona koja se nalaze s desne strane obične i iteracijske zagrade i s lijeve strane slova.

Između dva slova mjesto je istovremeno osnovno i predosnovno te se smatra **osnovnim**.

Između dvije zagrade mjesto je istovremeno osnovno i predosnovno te se smatra **predosnovnim**.

Osnovna mjesta odgovaraju **stanjima** automata, a predosnovna **prijelazima**.

Pravila o rasprostiranju indeksa

Osnovna mjesta označimo **kratkim** okomitim crtama i indeksiramo u **prvom** redu ispod izraza.

Predosnovna mjesta označimo **dugim** okomitim crtama i indeksiramo u **drugom** redu ispod izraza.

Osnovnim mjestima dodijelimo vlastite indekse: 1, 2, ...

Na predosnovna mjesta rasprostiremo indekse osnovnih mjesta koristeći 5 pravila:

1. Indeks mjesta ispred obične i iteracijske zagrade rasprostiremo na **početna predosnovna** mjesta disjunktivno vezanih članova unutar zagrade
   (zato jer se disjunkcija desi kad se desi jedan od disjunktivno vezanih članova)
2. Indeks mjesta ispred iteracijske zagrade rasprostiremo na predosnovno mjesto **iza** zagrade
3. Indeks **završnih** mjesta disjunktivno vezanih članova unutar obične i iteracijske zagrade rasprostiremo na predosnovno mjesto **iza** zagrade
   (zato jer se disjunkcija desi kad se desi jedan od disjunktivno vezanih članova)
4. Indekse svih mjesta koja smo rasprostirali na predosnovno mjesto **iza iteracijske** zagrade rasprostiremo na početna predosnovna mjesta disjunktivno vezanih članova unutar zagrade
   (zato jer iteracija dozvoljava proizvoljno ponavljanje)
5. Indeks **završnog** mjesta regularnog izraza rasprostiremo na ona predosnovna mjesta na koja se je rasprostirao indeks početnog mjesta regularnog izraza. Iznimno, ako za sekvencu bez strukture dozvoljavamo preklapanje, indeks završnog mjesta rasprostiremo na mjesta na koja se rasprostirao indeks mjesta u koje dolazimo nakon dijela sekvence koja se preklapa.

Dodijeljivanje izlaznih simbola

- Za **Moore** automat akceptorski izlazni simbol dodijelimo **završnom osnovnom** mjestu regularnog izraza
- Za **Mealy** automat akceptorski izlazni simbol dodijelimo posljednjem predosnovnom mjestu RI, a to je uvijek mjesto ispred posljednjeg simbola tražene sekvence
- Za sva ostala mjesta podrazumijevamo **neutralni** izlazni simbol

Pravila za redukciji indeksa

- Indekse **svih** mjesta koji su se rasprostirali na **isto** predosnovno stanje mjesto zamijenimo jednim indeksom, pod uvjetom da su im dodijeljeni isti izlazni simboli
- Indekse svih mjesta u koja dolazimo iz istog predosnovnog mjesta s istim ulaznim simbolom zamijenimo jednim indeksom

Ispis primitivne tablice automata

Primitivnu tablicu automata ispišemo tako da preostale indekse osnovnih mjesta uzmemo za stanja automata, a njihove prijelaze odredimo preko ulaznih simbola

13. Automati i jezici - zanačaj analize jezika

Preuzeti s Interneta - predmet Digitalna elektronika

Napomena: Treba naučiti prvih 12 pitanja, a dobro je pogledati i 13.