Slučajni generator K-mapa

Korman, Zvonimir

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:320207

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-26**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij Elektrotehnika

SLUČAJNI GENERATOR K-MAPA

Završni rad

Zvonimir Korman

Osijek, 2019.

Sadržaj

1. UVOD 2

1.1. Zadatak završnog rada 2

2. MINIMALIZACIJA BOOL IZRAZA 3

2.1. Booleova algebra 3

2.2. Quine-McCluskey algoritam 4

2.2.1. Petrick-ova metoda u kombinaciji sa Quine-McCluskey algoritmom 4

2.2.2. Don’t care stanja u K-mapama 5

2.2.3. Pojednostavljivanje pomoću metode Quine-McCluskey 6

2.2.4. Pojednostavljivanje pomoću Petrick-ove metode 9

3. WEB ALAT ZA MINIMIZACIJU 11

3.1. HTML 11

3.2. JavaScript 11

3.3. Implementacija Quine-McCluskey algoritma za rješavanje K-Mapa 12

3.4. Testiranje 14

4. ZAKLJUČAK 18

LITERATURA 19

SAŽETAK 20

ABSTRACT 21

ŽIVOTOPIS 22

PRILOZI 23

1. UVOD

U mnogim digitalnim sklopovima i praktičnim problemima moramo pronaći izraz s minimalnim brojem varijabli. Minimizirati se mogu Boolovi izrazi od 3, 4 varijable vrlo lako koristeći K-mapu bez uporabe teorema Boolove algebre. S obzirom da postoji potreba za rješavanjem i provjere točnosti takvog problema,nudi se rješenje gdje bi se pomoću web preglednika obavljalo računanje minimalnih izraza pomoću K-mapa. U 2. poglavlju uz teorijsko objašnjenje korištenih metoda i algoritama pokazuje se i njihova upotreba kroz primjer uz korake rješavanja. U 3. poglavlju nabrajaju se i opisuju web alati korišteni za slučajni generator K-mapa te se dodatno objašnjavaju funkcionalnosti i mogućnosti kroz njihovo opisivanje i testiranje.

1.1. Zadatak završnog rada

Zadatak završnog rada je koristeći web tehnologije izraditi slučajni generator K-mapa. Potrebno je moći odabrati veličinu mape te izračunati točno rješenje. Web prikaz treba sadržavati osnovne funkcionalnosti K-mape uz dodatne mogućnosti.

2. MINIMALIZACIJA BOOL IZRAZA

U matematici se izrazi iz više razloga pojednostavljuju. Na primjer, jednostavniji izraz je lakše razumjeti i lakše ga je zapisati, manje je sklon pogreškama tijekom interpretacije, ali najvažnije, pojednostavljeni izrazi su u većini slučajeva učinkovitiji kada se implementiraju u praksi. Boolov izraz se sastoji od varijabli i izraza. Pojednostavljivanje logičkih izraza dovodi do efikasnijih računalnih programa, algoritama i sklopova.

2.1. Booleova algebra

Booleova algebra predstavlja sustav matematičke logike koja služi za prikaz deduktivnog načina zaključivanja. U svojim djelima: *Matematička analiza logike* (1847) [1] i *Istraživanje zakona mišljenja* (1854) [2] George Boole je postavio osnovna pravila Boolove algebre. Booleova algebra koristi se pri analiziranju logičkih sklopova. Osnovne logičke operacije sadrže varijable (logičke izjave), koje mogu imati vrijednost „1“ u slučaju istine i vrijednost „0“ u slučaju laži. One također sadrže jedno od tri logička izraza: logički I, logički ILI i logički NE. Logički NE (-) je izraz koji pretvara izjavu iz istine u laž i obrnuto. Logički I (*) je izraz koji će rezultirati vrijednošću „1“, odnosno istinom samo ako su obje vrijednosti koje prima istinite. Logički ILI (+) rezultirati će istinom ako je barem jedna od dvije primljene vrijednosti istinita. U slučaju da su obje vrijednosti „0“, rezultat logičkog ILI je „0“. Složenije logičke operacije sadrže više od jednog logičkog izraza. Tablice istine za logička stanja NE, ILI i I prikazane su u tablicama 2.1, 2.2 i 2.3 [3].

Tablica 2.1: Prikaz logičkog izraza: Logički NE.

P	**-P**
0	**1**
1	**0**

Tablica 2.2: Prikaz logičkog izraza: Logički ILI.

P	Q	P+Q
0	0	0
1	0	1
0	1	1
1	1	1

Tablica 2.3: Prikaz logičkog izraza: Logički I.

P	Q	P*Q
0	0	0
1	0	0
0	1	0
1	1	1

2.2. Quine-McCluskey algoritam

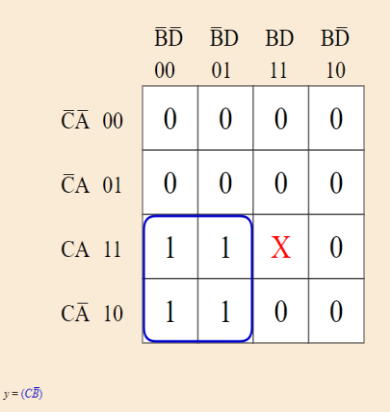
Quine-McCluskey algoritam (ili metoda primarnih implikanata) je metoda koja se upotrebljava za minimiziranje boolovih funkcija. Razvio ju je Willard V. Quine i dodatno proširio Edward J. McCluskey. Po svojoj funkcionalnosti identična je K-mapama, ali zbog svog tabelarnog oblika učinkovitija je za korištenje u računalnim algoritmima, a pruža i način provjere da li je postignut minimalni oblik boolove funkcije. Da bi se precizno koristio Quine-McCluskey, funkciju je potrebno zadati kao zbroj mintermi (minimalnog izraza), jer ako Booleova funkcija nije u minterm obliku, može se pronaći minterm proširenje (engl. *expansion*). Zbroj mintermi se tada koristi za određivanje minimalne sume proizvoda (SOP, engl. Sum of products). Tijekom prvog koraka metode, svi primarni implikati funkcije sustavno se formiraju kombiniranjem mintermi [4].

2.2.1. Petrick-ova metoda u kombinaciji sa Quine-McCluskey algoritmom

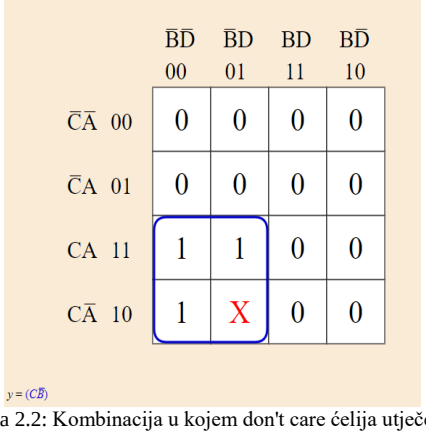
Petrick-ova metoda koristi se za određivanje rješenja minimalne sume proizvoda iz zadane tablice primarnih implikanata. Petrick-ova metoda je još jedna metoda koja se koristi u Booleovoj algebri,odnosno metoda pojednostavljenja koja se koristi za složene sklopove. Ako

se broj varijabli poveća u određenoj funkciji, broj primarnih implikanata i složenost primarne tablice može se drastično povećati. U tim je slučajevima najbolje služiti se metodom pokušaja i pogreške (engl. *trial and error*) pomoću kojih se mogu naći svi minimumi rješenja. Petrick-ova metoda ima puno bolji sustavni pristup od ostalih metoda [5].

2.2.2. Don't care stanja u K-mapama



Slika 2.1: Kombinacija u kojem don't care ćelija ne doprinosi rješenju.



Slika 2.2: Kombinacija u kojem don't care ćelija utječe na rješenje.

Don't care (prevedeno "ne zanima me") su stanja koja mogu biti 1 ili 0, sve dok nije bitno koji je traženi izlaz za stanje ulaza koje se ne očekuje. Ove ćelije se najčešće označavaju znakom „X", uz normalna označavanja polja sa 1 i 0. Prilikom formiranja grupa određena don't care ćelija tretira se na način da se promatra kao 0, 1 ili se zanemaruje. Ovo je korisno ako omogućuje stvaranje veće grupe nego što bi bilo moguće bez don't care ćelije, vidljivo na slici 2.2. Na slici 2.1 prikazano je kako nije nužno potrebno grupiranje svih ili bilo kojeg od don't care ćelija. Koristi se u grupiranju samo kada pojednostavljuje sveukupno logiku rješenja [6].

2.2.3. Pojednostavljivanje pomoću metode Quine-McCluskey

Koristeći navedene metode pojednostavit će se Boolov izraz zadan formulom:

$$f(A,B,C,D) = \sum m(2,6,8,9,10,11,14,15) \tag{2.1}$$

koji predstavlja sumu mintermi. Zadani izraz (2.1) u kanonskom obliku sastoji se od 4 varijable (A, B, C, D). Zadane minterme su: 2, 6, 8, 9, 10, 11, 14, 15.

- Korak 1 - Zadane minterme potrebno je rasporediti uzlaznim redoslijedom i napraviti grupe na temelju broja prisutnih u svojim binarnim prikazima.

Tablica 2.4: Tablica prvog grupiranja pri rješavanju izraza.

Ime grupe	Minerme	A	B	C	D
GRUPA.A1	**8**	**1**	**0**	**0**	**0**
GRUPA.A2	**6**	**0**	**1**	**1**	**0**
	9	**1**	**0**	**0**	**1**
	10	**1**	**0**	**1**	**0**
GRUPA.A3	**11**	**1**	**0**	**1**	**1**
	14	**1**	**1**	**1**	**0**
GRUPA.A4	**15**	**1**	**1**	**1**	**1**

Za zadani primjer se kreiraju 4 grupe, kao što je vidljivo u tablici 2.4. Grupa „A1“ sastoji se od minterme 8, jer njena binarna vrijednost sadrži samo jednu jedinicu („1-0-0-0“). Grupa „A2“ sadrži minterme : 6, 9 i 10, zbog toga što te minterme sadrže dvije jedinice u svojoj binarnoj vrijednosti (binarna vrijednost broja 9 je „1-0-0-1). Postupak se ponavlja za sve zadane minterme.

- Korak 2 – Uspoređuju se minterme koji su prisutne u uzastopnim skupinama. Ako se vrijednost promijeni u samo jednom položaju bita, tada od tih dviju mintermi treba napraviti par. Simbol "-" treba postaviti u promjenjeni položaj bita i zadržati preostale bitove kakvi jesu.

Tablica 2.5: Tablica drugog grupiranja.

Ime grupe	Minerme	A	B	C	D
GRUPA B1	2,6	0	-	1	0
	2,10	-	0	1	0
	8,9	1	0	0	-
	8,10	1	0	-	0
GRUPA B2	6,14	-	1	1	0
	9,11	1	0	-	1
	10,11	1	0	1	-
	10,14	1	-	1	0
GRUPA B3	11,15	1	-	1	1
	14,15	1	1	1	-

Za ovaj korak kreira se tablica 2.5. U ovom slučaju postoje 3 grupe. Svaka od te 3 grupe sadrži kombinacije dvije mintermi (minterma 2, binarne vrijednosti „0-0-1-0“ i minterma 6, binarne vrijednosti „0-1-1-0“ razlikuju se u svom drugom broju, gdje minterma 2 sadrži nulu, a minterma 6 jedinicu. S obzirom da se vrijednosti mjenja u samo jednom položaju bita za njih kreiramo grupu. Grupe se razlikuju po broju jedinica u binarnim vrijednostima svih kombinacija, odnosno u grupi „B2“ sve kombinacije sadrže dvije jedinice uz promjenjeni položaj, a u grupi „B3“ tri jedinice.

- Korak 3 – Ponavlja se korak 2 s novoformiranim izrazima dok se ne dobiju svi mogući primarni implikanti.

Tablica 2.6: Tablica sa 2 integracije.

Ime grupe	Minerme	A	B	C	D
GRUPA B1	2,6,10,14	-	-	1	0
	2,10,6,14	-	-	1	0
	8,9,10,11	1	0	-	-
	8,10,9,11	1	0	-	-
GRUPA B2	10,11,14,15	1	-	1	-
	10,14,11,15	1	-	1	-

Tablica 2.7: skraćena verzija tablice 2.6.

Ime grupe	Minerme	A	B	C	D
GRUPA C1	2,6,10,14	-	-	1	0
	8,9,10,11	1	0	-	-
GRUPA C2	10,11,14,15	1	-	1	-

Ponavljanjem koraka sa novim izrazima nastaje tablica 2.6. Ova tablica sadrži 2 grupe, koje se razlikuju u broju jedinica u svojim binarnim vrijednostima. U grupi „B1“ nalazi se kombinacija mintermi 2,6,10,14, odnosno iz prošle tablice kombiniraju se 2 postojeće kombinacije. Uspoređujući grupu 2,6 i 10,14 vidljivo je kako se grupe razlikuju u prvom broju binarne vrijednosti, dok u oba slučaja postoji simbol promjenjog bita na drugom broju. Ukoliko se u ovom koraku pojavljuju kombinacije bitova koje sadrže sve iste minterme u drugačijem redosljedu (2,6,10,14 i 2,10,6,14) jednu od tih kombinacija se može ukloniti. Uklanjanjem nastaje skraćena verzija tablice pod brojem 2.7.

- Korak 4 – Potrebno je formulirati tablicu primarnih implikanata.Tablica 2.8 sastoji se od niza redaka i stupaca. Primarni se implikati se upisuju u redove, a minimalni izrazi u stupce. Vrijednost "1" postavlja u ćelije koje odgovaraju minimalnim izrazima koji su obuhvaćeni u svim primarnim implikacijama.

Tablica 2.8: Tablica za određivanje primarnih implikanata.

Minterme/Primarni implikanti	**2**	**6**	**8**	**9**	**10**	**11**	**14**	**15**
CD'	**1**	**1**			**1**		**1**	
AB'			**1**	**1**	**1**	**1**		
AC					**1**	**1**	**1**	**1**

Promatrajući prošli korak, kombinacija mintermi 2,6,10,14 sadrži vrijednost „1“ za varijablu C, i vrijednost „0“ za varijablu D. Na temelju toga se primarni implikant za tu kombinaciju označava sa „CD'“. Bitni (engl. Essential) primarni implikanti određuju se na način da se promatraju stupci, odnosno minterme, i koliko puta se pojavljuju s obzirom na zadane grupe. Minterme 2 i 6 pojavljuju se samo unutar CD', dakle CD' je bitni primarni implikant. CD' će zbog toga biti dio pojednostavljenog Boolovog izraza.

- Korak 5 – Reducirati tablicu primarnih implikanata uklanjanjem retka svakog osnovnog implikata i stupaca koji odgovaraju mintermama koje su obuhvaćene tim bitnim primarnim implikantom. Korak se ponavlja dok se ne završe svi minimalni uvjeti date Boolove funkcije.

Tablica 2.9: Skraćivanje tablice primarnih implikanata

Minterme/Primarni implikanti	8	9	11	15
AB'	1	1	1	
AC			1	1

Skraćivanjem tablice 2.8 nastaje tablica 2.9. Isto kao i u prošlom koraku, minterme 8 i 9 su obuhvaćene samo primarnim implikantom AB', dakle AB' je bitni primarni implikant i dio je pojednostavljenog Boolovog izraza.

Tablica 2.10:Posljednje skraćivanje bitnog primarnog implikanta.

Minterme/Primarni implikanti	15
AC	1

U sljedećoj tablici 2.10 preostaje samo minterma 15, koja je obuhvaćena zadnjim primarnim implikantom AC stvarajući zadnji bitni primarni implikant. Izraz pod oznakom (2.2) predstavlja pojednostavljen Boolov izraz i rješenje početnog izraza [7].

$$f(A,B,C,D) = CD' + AB' + AC \tag{2.2}$$

2.2.4. Pojednostavljivanje pomoću Petrick-ove metode

Koristeći navedene metode pojednostavit će se Boolov izraz zadan formulom:

$$f(A,B,C,D) = \sum m(5,7,9,11,13,15) \tag{2.3}$$

Zadani izraz (2.3) prvo se rješava Quine-McCluskey metodom , odnosno do 4. koraka rješavanja. Kako bi se primjenila Petrick-ova metoda potrebna je tablica primarnih implikanata. Pridržavanjem koraka rješavanja Quine McCluskey metode nastaje tablica 2.11.

Tablica 2.11:Tablica za određivanje primarnih implikanata za Petrick-ovu metodu.

Minterme/Primarni implikanti	5	7	9	11	13	15
BD	1	1			1	1
AD			1	1	1	1

Petrick-ova metoda koristi teoreme Boolove algebre kako bi izraz minimizirao koliko je moguće.

Kako bi rješavanje bilo jednostavnije primarni implikanti se označavaju sa jednim slovom umjesto da se piše svaki pojedini primarni implikant.Za ovaj primjer primarni implikant BD će se označavati sa A, a primarni implikant AD će se označavati sa B. Izraz se zapisuje na način da se promatra tablica primarnih implikanata stupac po stupac, gdje svaka jedinica predstavlja umnožak. Ukoliko se u jednom stupcu pojavljuje više od jedne jedinice, taj izraz se zbraja. Izraz za zadani primjer glasi:

$$Y = A * (A) * (B) * (B) * (A + B) * (A + B) \tag{2.4}$$

Zadani izraz(2.4) potrebno je pojednostaviti koristeći teoreme Boolove algebre. Koristeći teorem koji glasi : „ X*X=X “, sljedeći izraz (2.5) glasi:

$$Y = A * B * (A + B) \tag{2.5}$$

Izlučivanjem primarnog implikanta A i uz dodatno izlučivanje primarnog implikanta B unutar zagrade prošli izraz poprima novi oblik, kao što se vidi u izrazu (2.6).

$$Y = A\big((A * B) + B\big) = A(B(1 + A) \tag{2.6}$$

Koristeći teorem koji glasi: „ X+1=1 “, konačni izraz (2.7) predstavlja zadnje pojednostavljenje izraza.

$$Y = AB \tag{2.7}$$

Kada se dobije konačni izraz, potrebno je zamijeniti slova prvobitnim primarnim implikantima i primarne implikante zbrojiti. Rješenje je pojednostavljeni Boolov izraz (2.8) [8].

$$Y = BD + AD = D(B + A) \tag{2.8}$$

3. WEB ALAT ZA MINIMIZACIJU

Koristeći dostupne web alate poput prezentacijskog jezika za izradu web stranica HTML (engl. *HyperText Markup Language*), koji služi za vizualni prikaz te ne obavlja nikakvu zadaću te JavaScript, skriptni programski jezik koji izvršava kompleksna računanja te koristi i implementira prethodno spomenute metode postiže se interaktivna web stranica koja rješava korisnički zadanu K-mapu.

3.1. HTML

Hypertext Markup Language (HTML) prezentira dokumente namjenjene za prikaz u web-pregledniku. Koristi tehnologije poput CSS (engl.*Cascading Style Sheets,*stilski jezik za opis prezentacije dokumenata napisanog u HTML jeziku) i skriptnih jezika poput JavaScript-a. Web-preglednici primaju HTML lokalno pohranjene dokumente, dokumente sa web servera te se ti dokumenti pretvaraju u multimedijske web stranice. HTML je izvorno sadržavao naznake za izgled dokumenta. HTML elementi su sastavni dijelovi svake HTML stranice. S HTML konstrukcijama objekti poput slika, tablica i videa mogu biti ugrađeni u prikazanu stranicu. HTML omogućuje stvaranje strukturiranih dokumenata označavanjem tekstualne semantike kao što su naslovi, odlomci, poveznice, citati, popisi i druge stavke. HTML elementi su napisani unutar uglatih zagrada. Preglednici ne prikazuju HTML oznake, već ih upotrebljavaju za tumačenje sadržaja određene web stranice. HTML ima mogućnost ugrađivanja programa napisanih u jednom od skriptnih jezika što utječe na prikaz sadržaja na web stranicama. Uključivanje CSS-a definira izgled i pregled sadržaja.[9]

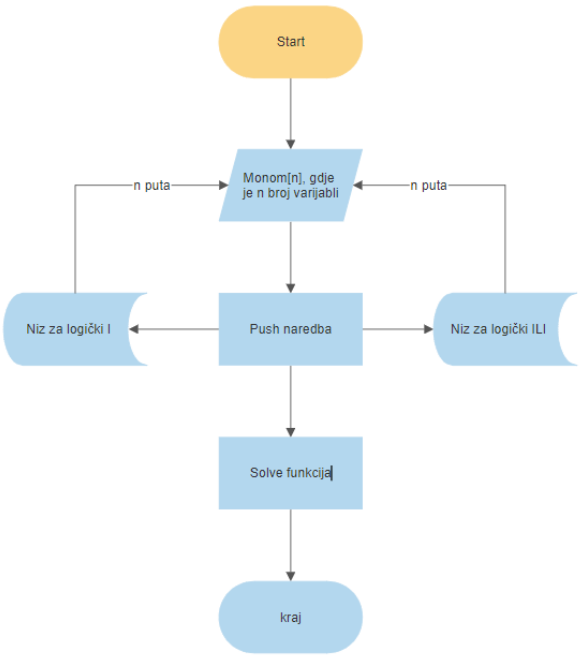
3.2. JavaScript

JavaScript, poznat i kao JS, vrlo je popularan i široko korišten skriptni jezik. Sintaksa pisanja u JavaScriptu sastoji se od vitičastih zagrada,funkcija i dinamičnog pisanja. Uz HTML i CSS, JavaScript je jedna od glavnih tehnologija svjetskog weba(World Wide Web,skraćeno WWW). JavaScript omogućava interaktivnost web stranice i bitan je dio gotovo svih web aplikacija. Korišten je u većini web stranica, a glavni web preglednici sadrže JavaScript mehanizam za njegovo izvršavanje. Sadrži API potreban za rad s tekstom, datumima, regularnim izrazima i DOM-om (engl. *Document Object Model*), ali jezik sam po sebi nije zadužen za uključivanje I/O (engl. *Input/Output*, komunikacija između npr. Kompjutera i

vanjskog svijeta kao što je čovjek ili sustav za procesiranje informacija), kao što su umrežavanje ili pohrana. Izrazi poput Vanilla JavaScript i Vanilla JS se odnose na JavaScript koji nije dodatno proširen raznim dodatnim bibliotekama. Osim toga što Java I javaScript dijele sličnosti poput samog imena , sintakse i standardnih biblioteka, dva su jezika bitno različita i vrlo se razlikuju po dizajnu.[9]

3.3. Implementacija Quine-McCluskey algoritma za rješavanje K-Mapa

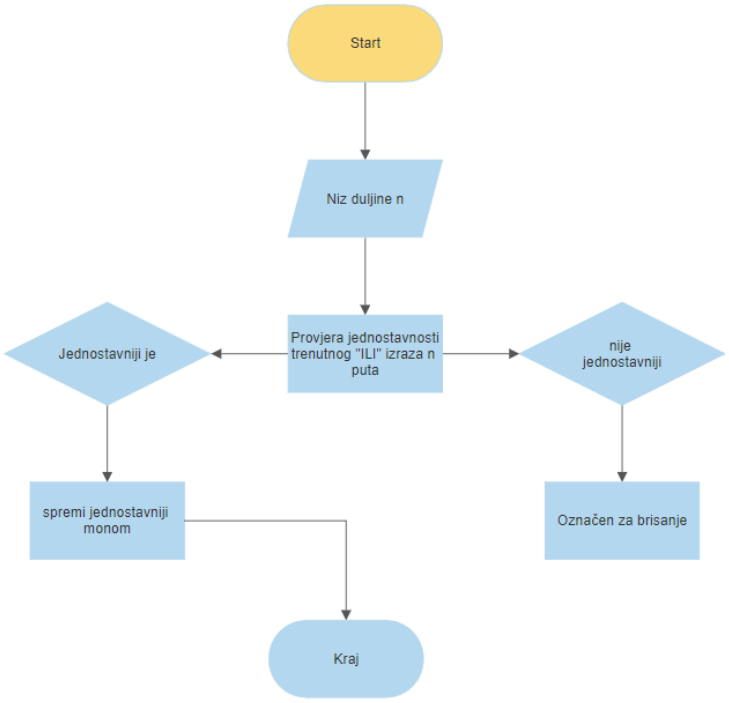
Quine-McCluskey metoda predstavlja atraktivno rješenje potrebno za minimizaciju složenih logičkih izraza neovisno o količini, tj. duljini varijabli. Ova inačica omogućuje minimizirani izraz zajedno sa njegovim prikazom na K mapi. Da bi se algoritam mogao pravilno koristiti potrebno je programski implementirati logička stanja pomoću funkcija. Funkcija „test“ koristi dva niza (engl.*Array*) koja predstavljaju logičko stanje I (AND) i ILI (OR). Pseudokôd u obliku dijagrama stanja prikazan je na slici 3.1.



Slika 3.1: Dijagram stanja za popunjavanje logičkih stanja unutar funkcije „test“

Navedena funkcija ponavlja popunjavanje oba niza za sve varijable kako bi sama implementacija metode bila moguća za 4 varijable s kojima ovaj kod radi. Funkcija također pomaže u minimiziranju Boolovog izraza jer ju koristi Petrick-ova metoda u svrhu pojednostavljivanja izraza.[9]

Pojednostavljivanje je vrlo bitno kako bi rješenje uistinu bilo minimizirano koliko je moguće. U primjeru pseudokôda na slici 3.2 doći će do promjene ukoliko se u nizu pronađe isti



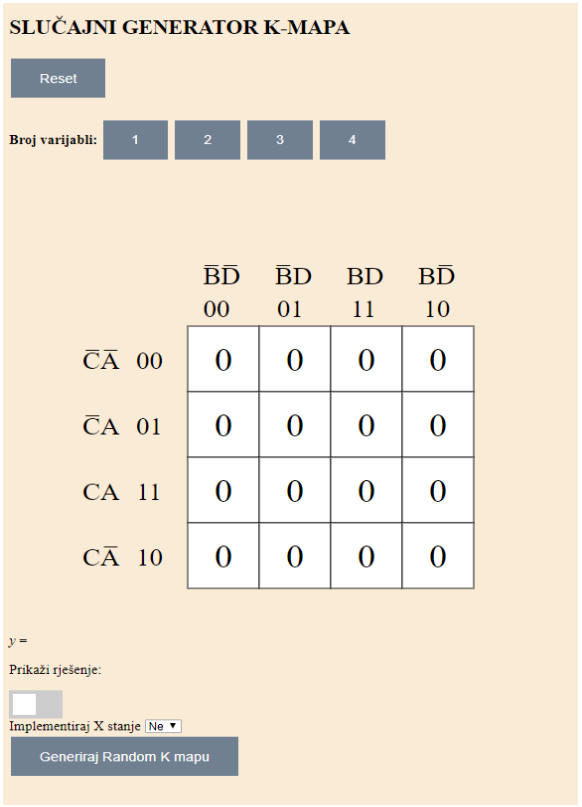
Slika 3.2: primjer pseudokoda za funkciju zaduženu za pronalazak najjednostavnijeg rješenja.

ili jednostavniji „ILI“ izraz. Ukoliko je pronađen takav izraz, povratna vrijednost vraća novi ILI izraz [10].

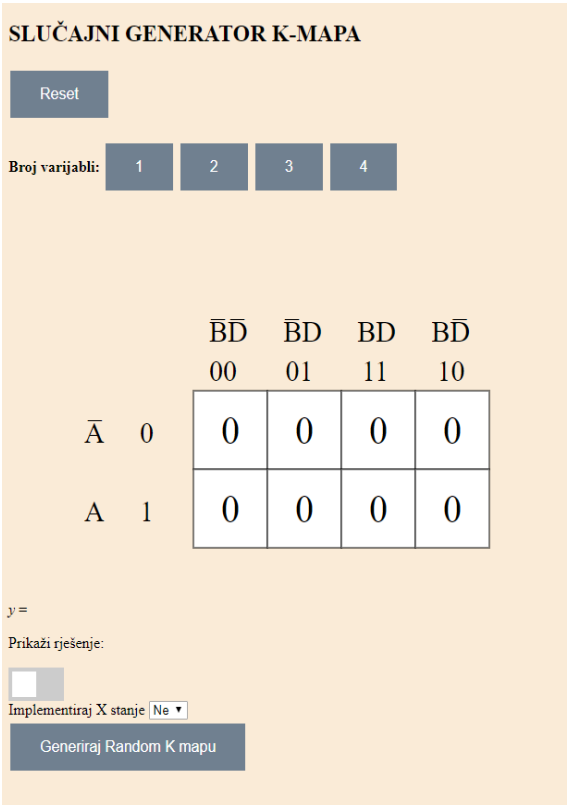
3.4. Testiranje

Web stranica, odnosno zadatak ovog završnog rada sastoji se od sljedećeg:

- Tipka „RESET“
- Tipke koje predtavljaju broj varijabli
- Interaktivnu K-mapu zajedno sa ispisom rezultata
- Klizač(engl. *Slider*) koji određuje hoće li se prikazati rješenje
- Mogućnost odabira hoće li se implementirati X(don't care) stanje
- Tipka „Generiraj Random K mapu“



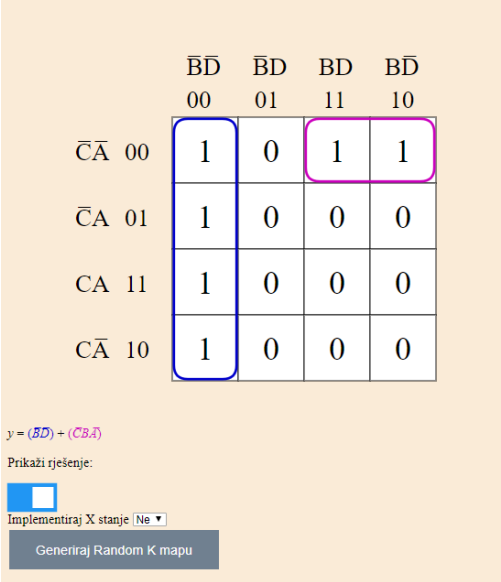
Slika 3.3: : Početna stranica slučajnog generatora K-mapa, odnosno prikaz za 4 varijable



Slika 3.4: Prikaz K mape za 3 varijable.

Tipka „RESET“ koja je prikazana na slikama 3.3 i 3.4 u gornjem lijevom kutu mjenja sve vrijednosti, koje su u trenutku pritiska tipke različite od nule, u nulu. Tipka neće utjecati na postavljeni odabir klizača za X stanje i prikaz rješenja. Pritiskom na neku od ponuđenih tipki za broj varijabli izgled K mape se mjenja u skladu sa odabirom.

Dok je klizač za prikaz rješenja „ugašen“(na lijevoj strani), korisniku su vidljive promjene na samoj K mapu poput mjenjanja vrijednosti u 1 ili X, ali rješenje se neće pokazati. Pomicanjem klizača u desnu stranu rješenja se pokazuju na način da se specifične kombinacije jedinica po pravilima zaokružuju određenom bojom te se pojednostavljeni Boolov izraz pokazuje u obliku „y=“, i boja primarnog implikanta je u skladu sa svojim prikazom na samoj K mapi, kao što je prikazano na slici 3.5.



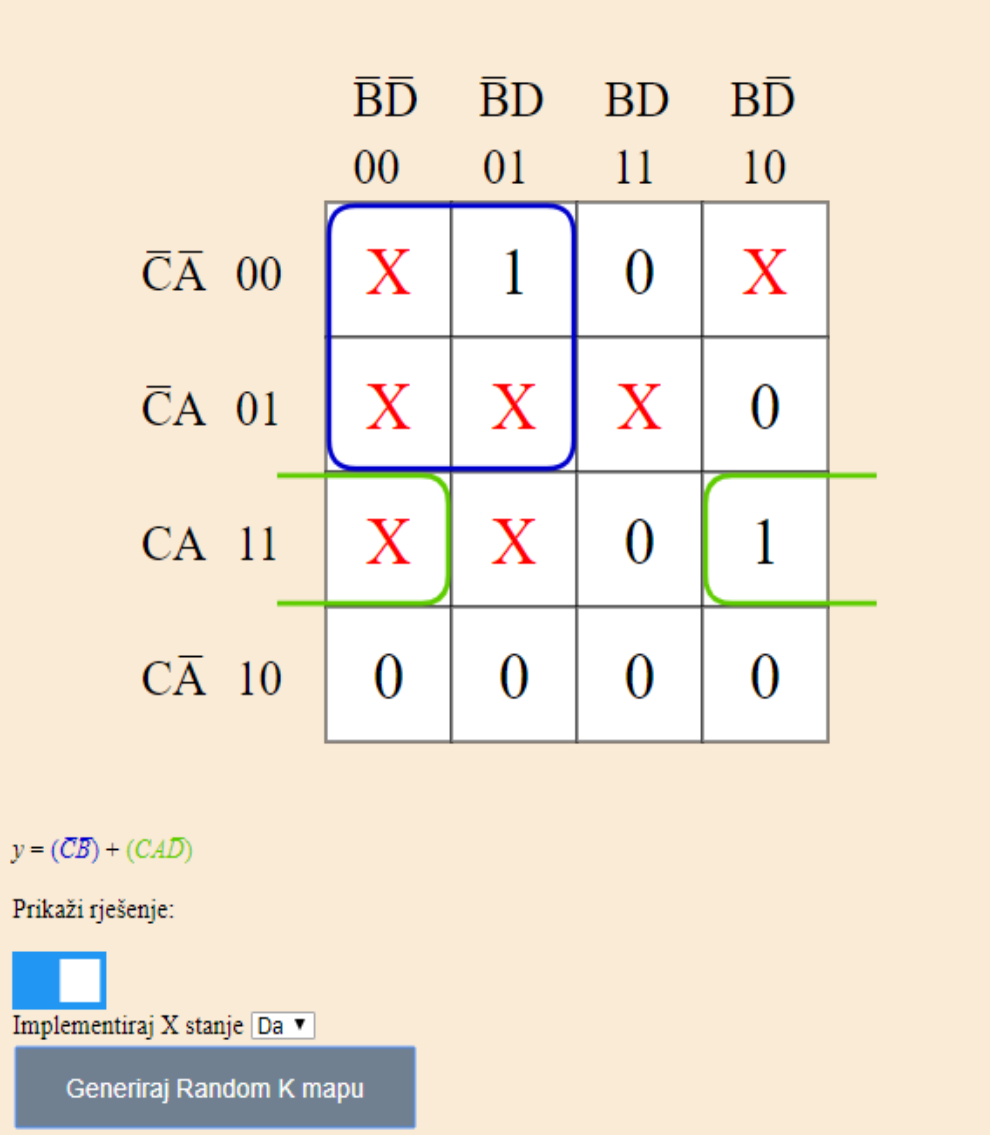
Slika 3.5:prikaz rješenja sa upaljenim klizačem.

Uključivanjem implementacije X stanja u K mapi se po korisnikovom odabiru može pojaviti i X. Program će automatski iskoristiti X varijable samo ako one pozitivno djeluju na rješenje odnosno dodatno ga pojednostavljuju.

```
Linija   Kod
1:       this.implementXs = function (type) {
2:       if (type > 0) {
3:       data.implementX = true;
4:       } else {
5:       data.implementX = false;
6:       }
7:       data.clear();
8:       this.update();
9:       };
```

Slika 3.6: primjer koda za implementiranje X stanja

Primjer implementacije X stanja u JavaScriptu nalazi se na slici 3.6, gdje će se implementirati X stanje ovisno o predanom tipu toj funkciji. Tip koji se predaje ovisi o tome kako je podešena tipka sa njeno implementiranje na samoj web stranici. Rješenje na K mapi zajedno sa klizačem za prikazanje rješenja i implementiranjem X stanja prikazano je na slici 3.7.



Slika 3.7: Prikaz rješenja uz uključenu implementaciju X stanja

Tipka „Generiraj Random K mapu“ generirat će za svako polje vrijednost 0 ili 1. Ukoliko je uključena opcija za „X“ stanje tada će se generirati i X [11].

```
Linija  Kod
1:      this.random = function() {
2:      for(var i in this.fields){
3:      if(this.implementX){
4:      this.fields[i].value=Math.floor(Math.random()*3);
5:      }else {
6:      this.fields[i].value=Math.floor(Math.random()*2);
7:              }
8:      this.quinemccluskey.data.setFuncData(this.fields[i].mapID, this.fields[i].value);
9:      }
10:     this.quinemccluskey.data.compute();
11:     this.quinemccluskey.update();
12:     this.compute();
13:     };
```

Slika 3.8: primjer koda za implementiranje X stanja

Funkcija sa slike 3.8 koristi dodatne funkcije kao što su Math.floor i Math.random. Svakoj vrijednosti polja se daje nasumično generirani broj.Math floor omogućuje zaokruživanje vrijednosti (3,71 zaokružilo bi se na 3) [9].

4. ZAKLJUČAK

Tijekom izrade završnog rada, za implementiranje funkcionalnosti K mape uz pravilni algoritam rješavanja, bilo je potrebno napraviti web stranicu u HTML-u, dok je sama funkcionalnost napisana u JavaScript jeziku. Ovakav način implementacije omogućuje jednostavnije korištenje nego konkretni program, zbog toga što je dovoljno ući u web stranicu umjesto instalacije programa.

Kada korisnik pokrene web sučelje, za svaki klik na neke od dodatnih funkcionalnosti ili same K mape, JavaScript funkcije i metode primaju parametre i vraćaju i/ili mjenjaju nove. Tako će se promjenom broja varijabli promjeniti i sam izgled K-mape da odgovara zadanom broju varijabli. Za svaki klik na samu K-mapu mjenja se parametar, i ta promjena predaje se JavaScriptu čije metode ponovno prolaze kroz postupak dobivanja rješenja i kroz metode zadužene za zaokruživanje grupa mintermi raznim bojama za svaku skupinu koja je ujedno i dio rješenja. Primarni implikanti u ispisu rješenja prikazani su bojama koje su prikladne samom postupku grupiranja. Ovakav pristup omogućuje da korisnik sa upaljenim klizačem za prikaz rješenja svakim novim klikom vidi rješenje za trenutni poredak varijabli, bez da ponovno mora pritisnuti tipku za rješavanje svaki puta kada promjeni vrijednost nekog parametra.

Korisnik može koristiti kreirano web sučelje za vježbanje i provjeravanje rješenja koje je dodatno olakšano zbog načina grupiranja vidljivo na K-mapi i u samom rješenju.

LITERATURA

[1] G. BOOLE, *MATHEMATICAL ANALYSIS OF LOGIC: being an essay towards a calculus of deductive reasoning ... (classic reprint)*. Place of publication not identified: FORGOTTEN Books, 2015.

[2] *An Investigation of the Laws of Thought*. Erscheinungsort nicht ermittelbar: Watchmaker Publishing, 2010.

[3] "Boolean Algebra - Operation, Laws, Operators & Much More." [Online]. Available: https://www.toppr.com/bytes/boolean-algebra/. [Pristupljeno: 02-08-2019].

[4] "Everything About the Quine-McCluskey Method." [Online]. Available: https://www.allaboutcircuits.com/technical-articles/everything-about-the-quine-mccluskey-method/. [Pristupljeno: 01-09-2019].

[5] "quine-mccluskey-handout.pdf." .

[6] M. K. Gooroochurn, *Introduction to digital logic & Boolean algebra*. Place of publication not identified: Publisher not identified, 2018.

[7] W. V. Quine, *Philosophy of logic*, 2nd ed. Cambridge, Mass: Harvard University Press, 1986.

[8] "Prime Implicant Simplification Using Petrick's Method." [Online]. Available: https://www.allaboutcircuits.com/technical-articles/prime-implicant-simplification-using-petricks-method/. [Pristupljeno: 30-08-2019].

[9] J. Duckett, *JavaScript & jQuery ; HTML & CSS*. 2014.

[10] "Use of Flowcharts in Computer Programming - Yonyx." [Online]. Available: https://corp.yonyx.com/customer-service/use-flowcharts-computer-programming/. [Pristupljeno: 12-08-2019].

[11] "Function Simplification and Don't Care Conditions - Digital System Design: Basics and Vulnerabilities | Coursera." [Online]. Available: https://www.coursera.org/lecture/hardware-security/function-simplification-and-don-t-care-conditions-btLxr. [Pristupljeno: 13-07-2019].

SAŽETAK

Cilj ovog završnog rada bila je izrada interaktivne web stranice za rješavanje K-mape koja ima mogućnost klikanja po samoj mapi za formiranje pojednostavljenog Boolovog izraza. Kako bih K-mapa bila ispravna bilo je potrebno koristiti skriptni jezik, u ovom slučaju JavaScript, za dobivanje željene točnosti prilikom rješavanja. Za web sučelje korišten je HTML napisan u Visual Studio Code-u, dok je izgled, odnosno stil web sučelja napisan u CSS stilskom jeziku. Kako bi se ostvario cilj ovog završnog rada bilo je potrebno implementirati metode i algoritme koje će omogućiti željenu funkcionalnost. Metoda rješavanja same funkcionalnosti K-mape je Quine-McCluskey zbog svoje mogućnosti pojednostavljivanja Bool izraza, dok je Petrick-ova metoda korisna za pojednostavljivanje primarnih implikanata uz korištenje Quine-McCluskey metode.

Korisnik može stranicu prilagoditi svojim potrebama, na način da mjenja broj varijabli, koji može biti bilo koji broj od 1 do 4, da koristi ili ne koristi „X“ stanja i najbitnije, može pokušati rješiti primjer random generirane K-mape dok je klizač za prikazanje rješenja isključen i nakon rješavanja provjeriti točnost tako da uključi klizač. Dobiveni rezultat korisnik tada može usporediti sa svojim, što omogućuje lagano i interaktivno vježbanje u području K-mapa. Korisnik također za vlastito rješenje može koristiti bilo koju metodu prikladnu za rješavanje K-mapa, s obzirom da će rješenje K mape uvijek biti isto neovisno o metodi, samo što su neke metode lakše za implementiranje u programu a teže u praksi, i obrnuto.

Ključne riječi : K-mapa, Bool, JavaScript, HTML, Minimizacija

ABSTRACT

The aim of this thesis was to make a interactive K-map solver website that has the ability to click on the map itself to form a simplified Boolean expression. In order for the K-map to be correct, it was necessary to use a scripting language, in this case JavaScript, to obtain the desired accuracy when solving. The web interface uses HTML written in Visual Studio Code, while the layout or the style of the web interface is written in CSS style language. In order to achieve the goal of this thesis, it was necessary to implement methods and algorithms that will provide the desired functionality. The method of solving the K-map functionality itself is Quine-McCluskey because of its ability to simplify Bool expressions, while the Petrick method is useful for simplifying the primary implications when used with the Quine-McCluskey method.

The user can customize the page to their needs by changing the number of variables, which can be any number from 1 to 4, by using or not using "X" states and most importantly, can try to solve an example of a randomly generated K-map while the slider designed for showing the solution is off and after solving be able to check if they are correct by turning on the slider. The result can then be compared by the user to his own solution, which allows for easy and interactive studying in the K-map area. The user can also use any method suitable for solving K-maps for his own solution, since the solution will always be the same regardless of the method, except that some methods are easier to implement in the program and harder in practice, and vice versa.

Key words: K-map, Boolean, JavaScript, HTML, Minimisation

ŽIVOTOPIS

Zvonimir Korman rođen je 15. siječnja 1997 godine u Osijeku. Završio je Osnovnu školu Višnjevac i I. Gimnaziju Osijek. Tijekom svog srednješkolskog školovanja zainteresirao se za tehnologiju i informatiku. Trenutno je redovan student 3. godine preddiplomskog studija Elektrotehnika,smjer Komunikacije i Informatika na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Tečno govori engleski jezik. Trenutno završava 3. godinu i želi upisati diplomski sveučilišni studij.

PRILOZI

Na priloženom CD-u nalaze se:

P1 Kompletan kôd

P2 Rad u .docx i .pdf formatu