REALIZACIJA BRZIH N-BITNIH SABIRAČA KORIŠĆENJEM DISJUNKTIVNE NORMALNE FORME

Vladan Vučković, Elektrofinski Fakultet u Nišu
Dragan Mlišeć, Elektrofinski Fakultet u Nišu
E-mail: vld@elfak.ni.ac.yu, dragan@elfak.ni.ac.yu

Sažetak - U radu je prikazana realizacija brzih n-bitnih sabirača korišćenjem disjunktivne normalne forme (DNF). Korišćenje DNF reprezentacije omogućava realizaciju sabirača pomoću dvo-nivovskih predlagačkih memorije što je efikasnije nego klasična realizacija preko kaskadno spajanja potensialera. Kao osnovni alat za minimizaciju DNF-a korišćena je aplikacija Hyperplane razvojena od strane autora, čije se funkcionalnost biti objašnjena u radu i uporedeno sa klasičnim Espresso programom slične namene. Kao jedna od važnih metoda za redukciju broja minternova u DNF-u tačno razmatra XOR transformaciju ulaznih promenljivih i njen učinak na izgubu broj logičkih elementa u finalnoj realizaciji.

1. UVOD

Sabirač predstavlja osnovnu aritmetičku jedinicu mikroprocesora i njegova implementacija se može naći u veleustručnim nijerama. Sabirač je neophodna komponenta sada za izvršavanje matematičkih ADD naredbi veće i u adresnim i indirektnim dekodiranju, proračunu računarskih sklopova, kao sastavni element drugih, složenijih elemenata aritmetičko-logičke jedinice kao što su množiće, delitelj, aritmetika u pokretom zarez... Zbog toga je vecna važna da se sabirač što je moguće efikasnije realizuje tako da njegova latencija u radu bude minimalna. Postoje nekoliko osnovnih metoda za realizaciju sabirača [1]. Najčešće korišćenje metoda je sa kaskadnom povezivanjem polisabirača [2] kako je to detaljnije objašnjeno u poglavlju 3. Međutim, ova realizacija sabirača implicira linearno povezivanje vremena koja je nekcornica od broja kaskada koje se uporedjuju u realizaciji sabirača. Da bi se smanjila latencija sabirača u radu je korišćen drugačiji pristup - direktna realizacija preko disjunktivne normalne forme. Pomoću DNF može se realizovati funkcija dvo-nivovskom memorijom u čim se osnovnom impulzovima, nulo zrastaju konzertivnih mintermeta dok je sekundarni impulz proizvoljan samo jednim logičkim OR korom. Na taj način maksimalno kaštjenje logičkog DNF sklopa je 2^n, dok je kaštjenje pojedinačnog AND odnosno OR elemenata. Kašćeno je zavisno od broja ulaznih promenljivih, što često se smatra da je DNF dobra metoda za realizaciju.brzih sabirača. Kao negativna usitnica javlja se problem eksponečnog rasta broja logičkih kola koja se moraju upotrebiti u realizaciji sabirača [1]. Minimalizaciju DNF (poglavlje 4) drastično smanjuje broj logičkih kola. Korisnici XOR transformaciju ulaznih promenljivih (poglavlje 5), uz nekadašnju minimalizaciju logičke funkcije, moguće je dodatno redukovati broj logičkih kola [1][4]. Cena transformacije je dodavanje trećeg (XOR) nivoa time se smanjuje latencija poredača na 3At, što se obično na vecima nivoa 3At kod modernih CMOS transistora koji se upotrebjavaju u realizaciji procesora nije značajno.

2. DISJUNKTIVNA NORMALNA FORMA KAO METODA PREDSVEZANJA LOGIČKE FUNKCIJE

Potpuna disjunktivna normalna forma predstavlja logičku sumu kombinacije ulaznih promenljivih ili njihovih negacija. Za svaku varijablu logičke jedinice u tabeli istinosti u disjunktivnoj normalnoj formi pojavljuje se odgovarajući minterm tako da ukupan broj minternova u DNF zavisit će od broja tačnih vrednosti. Kao primer, neka je data tabela istinosti za neku logičku funkciju koja ima tri ulazne promenljive x1, x2 i x3.

<table>
<thead>
<tr>
<th>Tabela 1: Tablica istinosti (primer).</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Korišćenjem potpune DNF bi se ova funkcija mogla napisati na sledeći način: $f(x_1, x_2, x_3, x_{1\overline{1}}, x_{1\overline{2}}, x_{1\overline{3}}, x_{2\overline{1}}, x_{2\overline{2}}, x_{2\overline{3}}, x_{3\overline{1}}, x_{3\overline{2}}, x_{3\overline{3}})$ odnosno u minimalizovanoj formi kao $f(x_1, x_2, x_3, x_{1\overline{1}}, x_{1\overline{2}}, x_{2\overline{1}}, x_{3\overline{2}})$.

Na sledenoj slici prikazana je shema realizacije u dva nivoa proizvoljne funkcije korišćenjem DNF:

![Slika 1. Dvo-nivovski nivač za realizaciju DNF.](image-url)

Što se tiče latencije u izvršavanju kod ovakvih logičkih uređaja ona je konstantna i nevaruje 2At, dok je 3At vreme latencije pojedinačnog logičkog sklopa (AND/OR kola).
Mreže ovoga tipa predstavljaju najefikasniji način realizacije logičkih funkcija što se tiče vremena kadaženja kroz logički sklapi. Ovo postižete pre svega iz činjenice da na prvim nivojima svi logički AND (i) i elementi radu u paraleli tako da je ukupno vreme kadaženja na ovom nivou samo $\Delta t$.

Sa druge strane, kako broj elemenata u tabeli istinitosti raste eksponentijalno kao $2^n$ (n je broj ulaznih promenljivih), očigledno je da ovakav način realizacije logičkih funkcija sa aspekta praktične implementacije zahteva veliki broj elementarnih logičkih kola. U cilju smanjivanja ovih zahteva pristupa se minimizaciji DNF prema njemu što se preda na konkretnu hardvernu realizaciju.

3. METODE ZA REALIZACIJU SABIRACA

U ovom poglavlju uporedimo dve metode za realizaciju sabirača.

3.1 STANDARDNA METODA - KASKADNO POVEZANI POLUSAIBRAČI

Standardna metoda za realizaciju sabirača osnažuje se na kaskadnom povezivanju osnovnih gradivnih elemenata - polusaibrača [2], pri čemu se sami polusaibrači realizuju logičkim funkcijama. Na sljedećoj slici dat je primer realizacije 4-bitog sabirača korišćenjem polusaibrača:

![Slika 2. Realizacija sabirača kaskadnim povezivanjem polusaibrača.](image)

Binarni brojevi koji se sabiraju dati su kao linearna kombinacija ulaznih promenljivih $x_i$ i $y_i$ tako da je $f = x_0 \cdot x_1 \cdot y_0 + y_1 + x_1 \cdot y_0 + y_1$. Svaki polusaibrač generiše prenos $(c_{i+1})$ za susjedni polusaibrač. Jasnje je da se porastom broja ulaznih promenljivih podesnost sklopa raste linearno što je dobra osobina ovakvog rešenja. Međutim, vreme latencijske sklopa takođe raste sa brojem ulaznih promenljivih što je moguća propagacija prenosa od prvog do zadnjeg polusaibrača (1.3). Ako pretpostavimo da je vreme latencijske pojedinačnog polusaibrača $\Delta t$ (pod pretpostavkom da se polusaibrač realizuje kao DNF), maksimalno vreme latencijske iznosi $3\Delta t$.

3.2 REALIZACIJA KORIŠĆENJEM DNF

Osnovna ideja kod ovog pristupa je istrebjavanje sabirača kao logičke funkcije i njegovu direktnu realizaciju pomoću DNF. Postupak za realizaciju sabirača direktnom metodom omogućava da se dobiju sklopi koji će obavljati funkciju sabiranja za maksimalno vreme $\Delta t$ bez obzira na broj ulaznih promenljivih. To znači da će, što se vremena istrebjavanja tiče, ovaj pristup uvek imati prednost o odnosu na klasičan metod, pri čemu ta prednost još više došlo do izražaj na povećavanju broja ulaznih promenljivih.

Na slici 3. prikazana je blok-sHEMA realizacije sabirača korišćenjem DNF pristupa.

![Slika 3. Direktna realizacija sabirača korišćenjem DNF.](image)

Uzun promenljive su podijeljene u dve simetrične grupe $x_i$ i $y_i$, koje predstavljaju ulaze u sabirac. Na izlazu dobijamo $f_{i+1} = x_i + y_i$, jer se generiše i skupni prenos iz sabirača. Postupak direktnih realizacija već je prikazan u poglavlju 2. Uzun broj ulaznih promenljivih je za što znači u tabeli istinitosti ima $2^n$ elementa. Sa glediska direktnih realizacija ovo je jako nepovoljno jer broj minimova za sabiranje stoji ovu jako eksponentijalnu zavisnost. Da bi se smanjio broj minimova pristupa se postupkom minimizacije.

4. MINIMIZACIJA DNF SABIRACA

 Za PC računare postoji niz algoritama za minimizaciju. U ovom poglavlju prikazujemo standardni program Expresso kao i Hypermine eksperimentalni program za minimizaciju razvijen od strane autora. Svi eksperimentalni rezultati generirani su ovom radu dobijeni su korišćenjem Hypermine aplikacije i verifikovani su rezultatima dobijenim programom Expresso.

4.1 ESPRESSO APLIKACIJA

Expresso je MS DOS aplikacija koja je pre svega namenjena optimizaciji DNF logičkih funkcija koja se realizuju PLA ili PLD logičkim kolima [5]. Aplikacija ima mogućnost minimizacije DNF-a što često iskorišća za realizaciju optimalne disjunktivne forme za sabirac. Program obezbeđuje da se na disku nalazi inicijalni fajl i kome se nalaze informacije o broju ulaznih i izlaznih promenljivih kao i tablica istinitosti za sve kombinacije bivaca na ulaz. Za primer, izvrsiti smo minimizaciju 4-bitog sabirača. Broj izlaznih promenljivih je 5 (računaci i prenos). Fragment ulazne datoteke .add pl. dat je u nastavku:

```
1 8
0.5
20 500
00000000 0000
00000001 0000
00000010 0010
00000011 0011
00000100 0100
00000101 0101
00000110 0110
00000111 0111
```

71
4.2 HYPERMINE APLIKACIJA

Za razliku od Expresso aplikacije čiji se optimizacijski algoritam bazira na radi sa implikantama, Hypermine aplikacija pristupa problemu optimizacije preko hipertabalova. Pokazalo se da je ova pristup znatno efikasniji tako da je minimalizacija korišćenjem ove aplikacije nekoliko puta brža nego ona koja je izvršena korišćenjem Expresso programa. Aplikacija je opremljena svim standardnim funkcijama i može da radi sa unutarnjim generisanjem tabele isinosti ili sa unesenim informacijama iz spoljašnjih ASCII datoteka čiji je format kompatibilan sa Expresso programom. Aplikacija je pisana u Delphi ja, u MS Windows okruženju.

Vizualni izgled aplikacije, koja je podešena na rešavanje istog problema - minimalizacija 4-bitnog sabirača, dat je na slici 4. U ovom tekst korišćeno je unutarnje generisanje tabele isinosti podno sledeće generatorke procedure:

```
function GENERATOR(inputs:longint; longint; var x,y:longint; real;
begin
 x:=inputs div divider;
y:=inputs mod divider;
generator:=x+y; [x...y]
end;
```

Slika 4. Vizualni izgled Hypermine aplikacije.
6. XOR TRANSFORMACIJA ULAZNIH PROMENJIVIH SABAIRA

Veoma značajna ideja je transformacija ulaznih promenljivih tako da se nad višim delom ($x_n$,...) obve ne operacija ekskluzivnog sabiranja po modulu 2 (logička XOR operacija) su nižim delom ($y_n$,...) [3],[4]. Blok shema realizacije sabaira za DNF čiji je viši dio ulaznih promenljivih transformisan XOR operacijom prikazana je na slici 5.

![Slika 5. Realizacija sabaira dvo-nivosek prekidačke mrežom sa transformacijom ulaznih promenljivih.](image_url)

Postupak realizacije bi bio sličan kao i za slučaj da nema transformacije ulaznih promenljivih. Uvođenje tretog nivoa povećava latenciju bloka za d, tako da je kašnjenje za ovu mrežu takođe konstantno i bez obzira na broj ulaznih promenljivih iznosi 3d. Da bi se izvršila minimalizacija ovako transformisane mreže pomoću programa Hypermin, potrebno je promeniti generatorsku funkciju uvođenjem XOR instrukcije:

```
def generator(x, y):
    return x ^ y
```

Ukupan broj generisanih minternova je 20 što iznosi 27% od broja koji je dobijen kada nije bilo ulazne transformacije.

7. EKSPERIMENTALNI REZULTATI

Da bi provrdili uticaj ulazne transformacije na broj minternova izvršen je eksperiment u kome su generisani i minimizovani svi sabairi od $adder_1$ do $adder_8$ (16 ulaznih promenljivih). Rezultati su prikazani u tabeli 2. Može se proveriti da broj minimizovanih minterova bez ulazne transformacije raste eksponencijalno sa porastom broja ulaznih promenljivih kao funkcija $2^n$ (kolona "Orig."). Sa druge strane, kada se upotребi ulazna transformacija porast je potrebnih logičkih elemenata za realizaciju je linearen (kolona "Transfer").

![Tabela 2. Rezultati minimizacije DNF-a sabaira.](image_url)

<table>
<thead>
<tr>
<th>IME</th>
<th>Uloga</th>
<th>Iznos</th>
<th>Orig.</th>
<th>Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$adder_1$</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>$adder_2$</td>
<td>4</td>
<td>3</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>$adder_3$</td>
<td>6</td>
<td>4</td>
<td>31</td>
<td>12</td>
</tr>
<tr>
<td>$adder_4$</td>
<td>8</td>
<td>5</td>
<td>73</td>
<td>20</td>
</tr>
<tr>
<td>$adder_5$</td>
<td>10</td>
<td>6</td>
<td>167</td>
<td>30</td>
</tr>
<tr>
<td>$adder_6$</td>
<td>12</td>
<td>7</td>
<td>355</td>
<td>42</td>
</tr>
<tr>
<td>$adder_7$</td>
<td>14</td>
<td>8</td>
<td>735</td>
<td>56</td>
</tr>
<tr>
<td>$adder_8$</td>
<td>16</td>
<td>9</td>
<td>1499</td>
<td>72</td>
</tr>
</tbody>
</table>

8. ZAKLJUČAK

Realizacija n-bitnog sabaira pomoću DNF predstavlja način da se ova aritmetička funkcija veoma efikasno realizuje logičkim kolima sa maksimalnom ukupnom latencijom 2d bez obziра na broj ulaznih promenljivih. To je značajno brže nego kod klasičnih shema sa polusabiracima kod kojih je latencija u linearnoj zavisnosti od broja ulaznih promenljivih. Da bi ovaj prijist bio opravdavan što se toče hardverske realizacije, uvodi se transformacija ulaznih promenljivih pomoću XOR koja tamo se eksplozibiljna ekspansija broja minternova sviđa na linearnu zavisnost. Učestalo se pri tome postiže raste sa povećanjem broja ulaza i veća je od 5:2:2:2 za osnovni sabair. Latencija za transformisan sabair je 3d zbog uvođenja tretog XOR nivoa. Na ovaj način postignut je optimalni u pogledu latentnog vremena radi sabaira kao i u pogledu broja minternova potrebnih za njegova fizičku realizaciju.

LITERATURA


Abstract - This paper is concerned with fast n-bit adder realization using successive normal forms (DNF). DNF enables realization of adders using two-level representation that is more efficient than classical cascade semi-adders realization. The application Hypermin, developed by the author, will be described and used to perform minimization procedures onto DNFs. Also its functionality will be compared with Zimpoco - standard minimization application. One of the very important methods for reduction of the final number of minterms is XOR transform of the input variables. Its influence to the optimzing of the final realization will also be considered.

REALIZATION OF THE FAST N-BIT ADDERS USING DISJUNCTIVE NORMAL FORMS
Vladan Vucković, Dраган Мацићевић