UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNITVO IN INFORMATIKO

tQCA - Seštevalnik

Seminarska naloga
pri predmetu
Optične in nanotehnologije

Blaž Lampreht, Luka Stepančič,
Igor Vizec, Boštjan Žankar

Povzetek

Za seminarsko nalogo je bilo potrebno realizirati enotritni seštevalnik oz. odštevalnik s pomočjo tQCA struktur. tQCA struktura je podobno kot binarna QCA struktura zgrajena iz QCA celic, le da imajo celice v tQCA strukturi namesto dveh štiri stanja. Želeli smo dokazati, da je v primeru seštevalnika za zgradbo vezja bolje uporabiti večvrednostno logiko kot pa binarno. V okviru seminarske naloge smo analizirali različne možnosti implementacije, tako z uporabo Postovega nabora funkcij, kot tudi z 'ad-hoc' metodo.

Ljubljana, Januar 2008

1 Uvod

Poskusi uporabe večvrednostne logike v integriranih vezjih sega nazaj v leto 1970. Večvrednostna logična vezja so bila sprva implementirana v bipolarni tehnologiji (ECL,CMOS,...), kar ni prineslo nobenih, oziroma malo izboljšav v primerjavi z binarno logiko. Pojavili pa so se novi načini implementacije večvrednostne logike, kot npr. RTT (resonant tunneling transistors), RTD (resonant tunneling diodes), STT (surface tunneling transistors). Negativno diferencialno uporne karakteristike, ki se pojavijo v teh napravah omogočajo čiste pragove ter zanesljivo računanje v večvrednostni logiki.

1.1 Zakaj trojiška logika

Predstavitev triarnih števil lahko razdelimo v dve večji skupini:

- **neporavnana** (unbalanced ali positive ali unsigned) - zahteva, da gredo števila le v »eno smer«. Npr.: 0,1,2 ali 1,1/2,2,...
- **poravnana** (signed) - dovoljuje da gredo števila v »obe smeri«, npr.: -1,0,1

Slednji je zelo uporaben pri realizaciji seštevalnika oziroma odštevalnika, saj lahko za odštevanje in seštevanje uporabimo isto logiko - vse kar moramo storiti, je to, da spremenimo predznak odštevanca, kar je pri uporabi poravnane triarne logike povsem preprosto. Če je odštevanec enak -1, je njegova negativna vrednost enaka $+1$, in obratno.

1.2 Označevanje

Pri realizaciji seštevalnika/odštevalnika smo uporabljali poravnano trojiško logiko. Označevanje z tQCA oznakami { A, B, C, (D) } je neugodno za poravnano logiko, zato bomo v nadaljevanju raje uporabljali oznake $\{-1, 0, +1\}$, oziroma krajšano le $+, 0, -$ in dodatno[1] vrednost D.

$$\begin{array}{ccc} -1 & \leftarrow & \text{A} \\ +1 & \leftarrow & \text{B} \\ 0 & \leftarrow & \text{C} \\ D & \leftarrow & \text{D} \end{array}$$

[1]Ker nam tQCA ponuja tudi prepovedano vrednost D, jo bomo označevali enako, kljub temu, da je v poravnanem trojiškem sistemu ni.

1.3 Predstavitev enotritnega seštevalnika

Enotritni seštevalnik je logično vezje, ki podobno kot binarni seštevalnik potrebuje dva izhoda - vsota s in prenos c. Izhodi so seveda trojiški.

Če bi naprimer želeli testirati seštevalnik pri odštevanju števila B (1) od števila A (-1), bi morali le število B zamenjati z A, ter števili sešteti - torej dobili bi enak rezultat kot pri seštevanju števil x in -y. Na ta način lahko seštevalnik uporabimo tudi kot odštevalnik.

x	y	c	s
$-$	$-$	$-$	$+$
$-$	0	0	$-$
$-$	$+$	0	0
0	$-$	0	$-$
0	0	0	0
0	$+$	0	$+$
$+$	$-$	0	0
$+$	0	0	$+$
$+$	$+$	$+$	$-$

Tabela 1: Pravilnostna tabela seštevalnika

2 Metode

2.1 Postov funkcijsko polni nabor

V Postovi trojiški logiki obstaja funkcijsko poln nabor, ki je zaradi treh vrednosti večji kot so dvojiški nabori (na primer konjunkcija, disjunkcija, negacija). Njen nabor funkcij sestoji iz:

- vseh konstant(A,B,C),
- karakterističnih funkcij konstant(f^A, f^B, f^C),
- ter funkcij minimum($\wedge$) in maksimum($\vee$).

Sedaj ko smo si izbrali trojiško logiko, in vzeli funkcijsko polni nabor, lahko zapišemo enačbi za vsoto dveh števil nad tritom, ter enačbo za prenos.

Tu smo si pomagali z minimizacijo z Vietchovim diagramom. Iz tabele 1 pravilnostna tabela seštevalnika, izhajata naslednja diagrama za minimizacijo.

y \ x	A	C	B
A	B	A	C
C	A	C	B
B	C	B	A

(a) Vsota

y \ x	A	C	B
A	A	C	C
C	C	C	C
B	C	C	B

(b) Prenos

Tabela 2: Veitchevi diagrami za vsoto in prenos

Iz diagramov dobimo minimizirani enačbi, ki ustrezata seštevalniku odštevalniku.

$$s = x^A y^A B \vee x^A y^B C \vee x^C y^C C \vee x^C y^B B \vee x^B y^A C \vee x^B y^C B$$

$$c = x^C \vee y^C \vee x^A y^B C \vee x^B y^A C \vee x^B y^B B$$

2.2 'Ad-hoc' metoda

Do t. i. 'ad-hoc' postopka smo prišli po ugotovitvi, da z uporabo Postove trojiške logike praktično ni mogoče testirati implementacije, kajti že strukture s 100 in več celicami so za trenutno uporabljeni simulirator preveliko breme. Da bi prišli do idealne rešitve, bi bilo potrebno poiskati vse možne primitive v tQCA. Primeri najosnovnejših primitivov v 'tQCA':

1. majoritetna vrata:

$$maj(x, y, z)$$

2. negacija po vertikali ($x \rightarrow \neg x$):

$$A \rightarrow B, \quad B \rightarrow A, \quad C \rightarrow C, \quad D \rightarrow D$$

3. negacija po diagonali ($x \rightarrow \sim x$):

$$A \rightarrow A, \quad B \rightarrow B, \quad C \rightarrow D, \quad D \rightarrow C$$

2.2.1 Postopek iskanja rešitve 'ad-hoc'

Funkcija signala *'carry'* (prenos) smo takorekoč dobili že iz majoritetne funkcije - poleg vhodov operandov x in y je bilo potrebno napeljati samo konstanto $'0'$ na majoritetna vrata,

$$c = maj(x, y, 0).$$

Med preiskovanjem majoritetne funkcije smo opazili tudi precejšnjo podobnost med funkcijo vsote $s(x, y)$ in funkcijo $f_1 = maj(x, y, D)$. Na spodnji tabeli so prikazani izhodi funkcije f_1 in želeni izhodi za vsoto (s), s sivo so označene razlike.

x	y	c	s	f_1
$-$	$-$	$-$	$+$	$-$
$-$	0	0	$-$	$-$
$-$	$+$	0	0	0
0	$-$	0	$-$	$-$
0	0	0	0	D
0	$+$	0	$+$	$+$
$+$	$-$	0	0	0
$+$	0	0	$+$	$+$
$+$	$+$	$+$	$-$	$+$

Funkcija f_1 se v treh od devetih vrednosti razlikuje, vendar hitro lahko opazimo nekaj zanimivih lastnosti:

1. funkcija je polarno simetrična[2] - ravno tako kot vsota in prenos
2. prva in zadnja vrednost f_1 sta enaki negiranim vrednostim funkcije prenosa ($\neg c$)
3. precej moteča je prepovedana vrednost D pri vhodih $x = 0$, $y = 0$, le-te bi se znebili, če bi imeli element, ki pretvori $D \rightarrow C$, ki pa obenem ohrani vse ostale vrednosti.

Pri tem se vprašamo, *kako spremeniti vrednosti pri vhodih* $(-1, -1)$ *in* $(+1, +1)$*, da bodo ustrezale funkciji* $\neg c$. Najlažji način, da to storimo, je tako, da na majoritetna vrata napeljemo f_1 in nekako »obtežen« vhod $\neg c$. Trik je v tem, da se obteži samo vrednosti $+1$ in -1, medtem ko mora biti

[2] vrednosti na spodnji polovici so enake negiranim vrednostim na zgornji polovici

x_1	x_2	x_3	f_1	ρ
A	D	A	A	0.998592
A	D	C	A	0.998592
A	D	B	D	0.998685
C	D	A	A	0.998592
C	D	C	C	0.998516
C	D	B	B	0.998592
B	D	A	D	0.998685
B	D	C	B	0.998593
B	D	B	B	0.998592

Tabela 3: Rezultati simulacije za f_1

vpliv vrednosti 0 nekako nevtraliziran - tudi to je razmeroma enostavno, ker lahko s pomočjo negacije po diagonali dobimo 'nasprotne' vrednosti za 0 (C) in D, medtem ko -1 (A) in $+1$ (B) ostanejo nespremenjene. Če zgoraj omenjene signale napeljemo na majoritetna vrata, takorekoč izničimo vpliv polarizacije C oz. 0.

$$\begin{aligned} f_2 &= maj(\sim(maj(x,y,D)), \neg c, \sim(\neg c)) \\ &= maj(\sim(maj(x,y,D)), \neg(maj(x,y,0)), \neg(maj(x,y,0))) \end{aligned}$$

x	y	c	f_1	$\neg c$	$\sim(\neg c)$	f_2
$-$	$-$	$-$	$-$	$+$	$+$	$+$
$-$	0	0	$-$	0	D	$-$
$-$	$+$	0	0	0	D	0
0	$-$	0	$-$	0	D	$-$
0	0	0	D	0	D	D
0	$+$	0	$+$	0	D	$+$
$+$	$-$	0	0	0	D	0
$+$	0	0	$+$	0	D	$+$
$+$	$+$	$+$	$+$	$-$	$-$	$-$

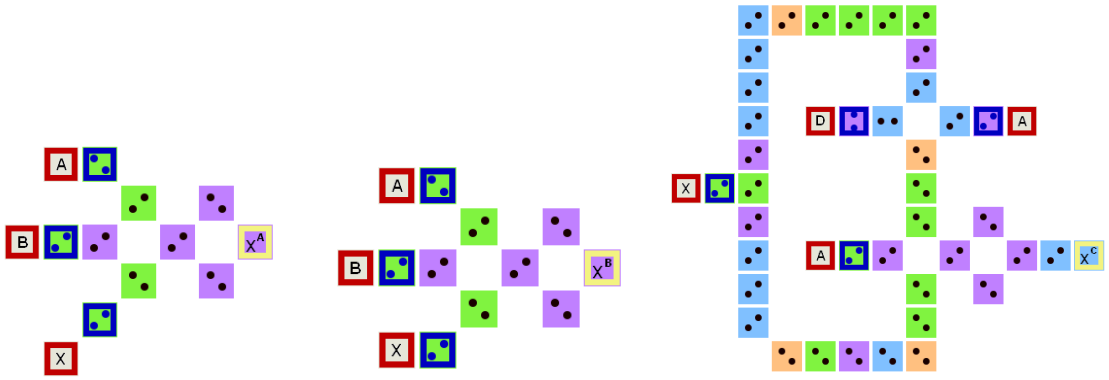
Tabela 4: Vrednosti funkcije/signala f_2 in vmesne vrednosti

Rezultat je sedaj precej bolj podoben vsoti, vendar na žalost v tej enačbi nastopa funkcija f_1, ki vrača pri vh. kombinaciji $(0,0)$ vrednost D. Nujno

je bilo treba poiskati funkcijo, ki pretvori vrednost D v vrednost 0 oz. C, sicer bi bila metoda neuporabna. Odločili smo se sistematično preiskati vse 'primitive' na polju celic velikosti 3×3, in sicer, za vse možne vhodne kombinacije - ta postopek bi lahko trajal celo večnost, vendar se je, na srečo, iskana operacija pojavila že v tretjem poizkusu.

2.3 Implementacija

Teoretična podlaga za to rešitev je zelo preprosta. Imamo dve enačbi, ki ju je le potrebno predelati v tQCA vezje. Prvi problem je nastal takoj, saj nismo imeli vseh struktur funkcijsko polnega nabora. Iskanje karakterističnih funkcij, kot strukture je nivo nižje od sestavljanja vezja preko že znanih primitivov. Primitive dobimo s preiskovanjem prostora celic in upanja da bomo našli iskano. Primitive karakterističnih funkcij[3], katere so našli v laboratoriju za računalniške strukture in sisteme.



(a) f^A (b) f^B (c) f^C

Slika 1: Karakteristične funkcije

Do naslednjega problema pa smo naleteli, ko smo glede na enačbe predvideli celotno strukturo. Enačba za prenos je še kar lepa, saj se jo da minimizirati. Enačba seštevalnika pa se žal ne čisto ne minimizirana, in njen diagram predstavlja analogijo »šahovnice« Vietchovega diagrama v dvojiški logiki (problem, ki se ga ne da minimizirati). Preden smo se odločili strukturo sestaviti, smo naredili oceno končne strukture in smiselnost rešitve.

[3]Te funkcije smo testirali samostojno, potrebno bi bilo tudi testirati kako se obnašajo v večjem vezju.

struktura	št. potrebnih elem.	št. celic strukture	št. celic
f^A	6	∼10	60
f^B	8	∼10	80
f^C	6	∼40	240
min(maj. vrata)	18	∼10	180
max(maj. vrata)	9	∼10	90
Vseh celic (brez povezovalnih linij)			650

Tabela 5: Poraba celic

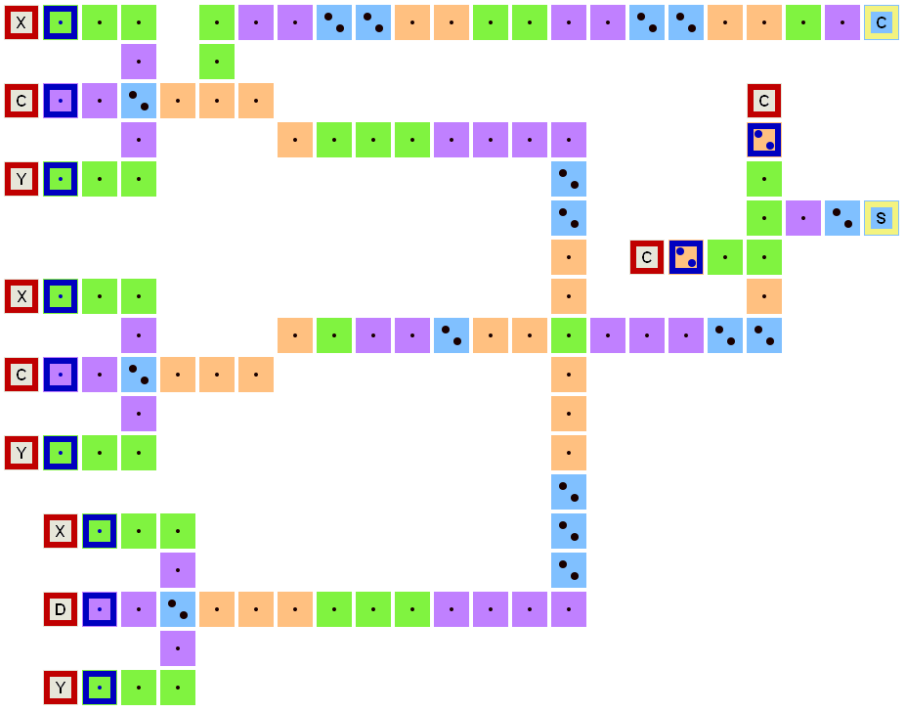
Iz zgornje tabele je razvidno, da potrebujemo za seštevalnik oz. odštevalnik, ki deluje nad dvema tritoma potrebno vsaj 650 celic. V tej točki smo prenehali iskati rešitev s Postovim funkcijsko polnim naborom. Dokazali bi si edino, da enačbe delujejo. Nebi pa našli strukture, ki bi bila majhna, ki bi se jo dalo uporabiti za gradnjo npr. kompleksnejšega kaskadnega seštevalnika.

3 Rezultati

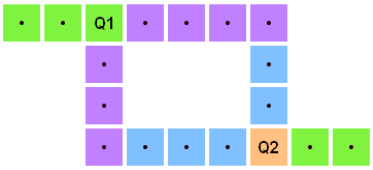
Pri implementaciji smo težili k rešitvi, ki bi bila karseda enostavna (najmanjše število celic, vhodov in konstant). V 2.3 smo predstavili dve možni smeri iskanja rešitve tQCA seštevalnika, ter odnehali z iskanjem rešitve preko funkcijsko polnega nabora. Mogoče obstajajo manjše strukture funkcij nabora, vseeno pa enačbi zahtevata 47 operacij, česar pa ne moremo zmanjšati.

Zato je edino vprašanje, ali je naša 'ad-hoc' rešitev **optimalna**. Zaseda 120 celic, kar je manj kot 20% prostora, ki bi ga zasedala rešitev iz Postovega nabora funkcij - kar ni slaba izboljšava. Kljub temu, pa ne moremo reči, da je vezje idealno, saj je v vezju **dvakrat izračunana** vrednost negiranega prenosa. Redundanci celic bi se izognili, če nebi potrebovali negiranega prenosa, še negirati z sodo linijo ($D \rightarrow C$). Oba rezultata pa se morata obvezno združiti v eni celici. Na sliki je to celica Q2, kjer se kraka združita na levi in zgornji stranici. Kraka pa bi morala nastati iz ene celice Q2, po principu razvejitve. Tu nastane problem - oba kraka, ne glede kako postavimo celici Q1 in Q2, sta vedno le sode ali lihe dolžine, ne pa različnih dolžin. Zato se, na žalost, s tem pristopom ne da zmanjšati redundance.

Ko smo v iskanju rešitve dopustili konstanto D in uporabo sode dolžine linije, kot posebna negacija, smo se približali rešitvi. Problem je nastal, ko so se v rezultatu pojavili obe vrednosti C in D. Če bi imeli le vrerdnosti C



Slika 2: Celotno vezje seštevalnika

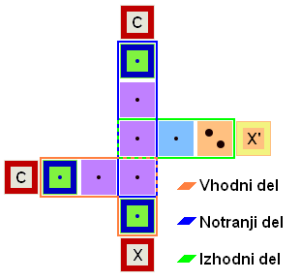


Slika 3: Problem kvadrata

ni problema, če pa imamo le vrednosti D pa lahko dodamo celico, oziroma jo le odstranimo in tudi rešimo problem. Ker pa sta nastopali obe, je bilo potrebno najti novo strukturo.

Ne glede, da je v vezju struktura, kot nekaj kar bi se mogoče dalo minimizirati, je pa pomemben stranski rezultat. Zelo kmalu pri razvoju strukture smo opazili, da je vrednost C velik vzrok težav. Je vrednost, ki bi jo lahko opisali kot absorbirana vrednost. Njena lega se izkaže, da je velikokrat stanje minimalne energije ter tudi le stežka dosežemo kaj drugega. Opazili smo, da če združimo dve liniji pravokotno z vrednostjo C in D dobimo vrednost B.

Tako ta struktura olajša delo, saj se sedaj ni potrebno »bati« in izogibati vrednosti D stanja C.



Slika 4: Vezje elementa, ki filtrira prepovedane vrednosti

Vseeno pa je potrebno povedati, da je struktura mogoče manj stabilna od majoritetnih vrat, negacije, in še kakšnega podobnega primitiva. Izkaže se, da nanjo močno vplivajo sosednje strukture. Zato jo je pomembno postaviti raje kakšno celico bolj stran od vezja. To pa ravno ni velika omejitev, saj je struktura, ki jo ponavadi potrebujemo v stanju izhoda. Deluje pa ravno na principu združevanja stanj C in D v pravokotni združitvi linij.

Zaradi boljšega razumevanja samega elementa smo na sliki 4 razdelili celice v tri področja delovanja. Ta območja se prekrivajo in celice so v istem ciklu ure. Oranžen oziroma vhodni del, ter tudi sredinski - modri del, imata kot vhod konstanto C. Z svojo oddaljenostjo smo poskrbeli, da vpliv na vhoda A in B ne vplivata in ju ne spremenita pri prehodu do izhoda. Pri vhodni vrednosti C ne pridemo do nobenega problema saj ga konstanti le ojačata. Pri vrednosti D pa se zgodi to, da se po celotnem notranjem delu celice postavijo na vrednost C. Z liho dolžino linije na izhodu pa pskrbimo da se pravilno prenese preko izhodnega dela.

Kljub vsem problemom, pa smo glede na izbiro primitivov, vezje optimizirali na najmanjše možno. Število celic je v območju želenega rezultata. Vseeno pa se nam je kar poznalo pri sami simulaciji z qdSim. Struktura z 12 vhodi, 2 izhodoma in 120 celicami, je za simulacijo, porabila v povprečju dobro uro na vhodno kombinacijo. Spodaj je še prikaz izhoda simulacije našega vezja.

Vhodi v vezje											Izhodi			
x		y	x		y	x		y			c	$\rho(c)$	s	$\rho(s)$
A	C	A	A	C	A	A	D	A	C	C	A	0.999967	B	0.999967
A	C	C	A	C	C	A	D	C	C	C	C	0.999964	A	0.999967
A	C	B	A	C	B	A	D	B	C	C	C	0.999964	C	0.999964
C	C	A	C	C	A	C	D	A	C	C	C	0.999964	A	0.999967
C	C	C	C	C	C	C	D	C	C	C	C	0.999964	C	0.999964
C	C	B	C	C	B	C	D	B	C	C	C	0.999964	B	0.999967
B	C	A	B	C	A	B	D	A	C	C	C	0.999964	C	0.999964
B	C	C	B	C	C	B	D	C	C	C	C	0.999964	B	0.999967
B	C	B	B	C	B	B	D	B	C	C	B	0.999967	A	0.999967

Tabela 6: Rezultati končne simulacije, $\epsilon = 1 \times 10^{-5}$

4 Zaključek

Pot do naše rešitve ni bila niti približno lahka. Vsakič ko smo bili blizu rešitve se je le-ta spretno izmuznila in oddaljila.

Prva ovira nas je doletela že na samem začetku iskanje rešitve. Ker ni bilo nobenega grafičnega vmesnika za sestavljanje struktur, je bilo potrebno vsako strukturo najprej natančno narisati na papir in jo nato prepisati v tekstovno datoteko. Popravki so bili izjemno zamudni. Zato se je Luka pogumno lotil velikega projekta izdelave grafičnega vmesnika in ga tudi (skoraj) uspešno dokončal. S tem smo pridobili marsikatero uro pri gradnji naših struktur.

Takoj za tem smo naleteli na drugo oviro. Izjemno **dolgi časi** testiranja struktur. To smo lahko nekoliko omilili in smo žrtvovali nekaj natančnosti za hitrejše izvajanje. To nas je pripeljalo do nekaterih na videz pravilnih rešitev, ki so kasneje ob večji natančnosti odpovedale.

Če bi imeli **primitive** primernih velikosti (znotraj ranga 3x3 polj), bi nam to že na začetku iskanja rešitve zelo olajšalo delo. Ker do danes primitivov v tako majhni obliki še ni, smo lahko nekatere rešitve našli le v teoriji, a so se hitro izkazale prevelike za realizacijo.

Naslednjo veliko težavo nam je predstavljal **problem križanja** linij, ki jih trenutni simulator ne podpira. To je privedlo do strukture z večimi vhodi kot bi bilo potrebno in posledično do večje strukture, kar je še dodatno podaljšalo čas testiranja.

Za gradnjo in testiranje struktur bi vsekakor potrebovali veliko zmogljivejše računalnike ali pa kar računalniško polje. Navkljub našim težavam pri realizaciji, pa smo prepričani, da bodo QCA avtomati vsekakor del prihodnosti računalništva.