ARHITEKTURA BINARNOG SABIRAČA POGODNA ZA IMPLEMENTACIJU NA FPGA

Bojan Jovanović, Milun Jevtić, *Univerzitet u Nišu, Elektronski fakultet, {bojan,milun.jevtic}@elfak.ni.ac.rs*

Sadržaj – *U radu je obavljena teoretska i eksperimentalna komparativna analiza performansi nekoliko arhitektura binarnih sabirača. Takođe, predstavljena je i jedna modifikovana carry-bypass tehnika namenjena poboljšanju karakteristika sabirača. Sve arhitekture implementirane su na FPGA komponentu iz Virtex-6 familije. Rezultati implementacije svedoče da u radu opisana carry-bypass tehnika predstavlja najbolji kompromis u pogledu zauzeća resursa, brzine rada i potrošnje električne energije. Predstavljeni rezultati mogli bi biti od koristi prilikom izbora arhitekture binarnog sabirača koja će se koristiti za implemetnaciju određenog digitalnog dizajna.*

1. UVOD

Gotovo svaki dizajn koji se bavi procesiranjem digitalnih signala (digitalni filtri, analiza slike, neuronske mreže, robotika i automatika itd.) u sebi sadrži neku aritmetičku jedinicu. Šta više, pažljivom analizom celokupne strukture dizajna često se može zaključiti da od izbora aritmetičke jedinice zavise i celokupne performanse sistema. Imajući na umu da je binarno sabiranje najzastupljenija aritmetička operacija, kao i to da se koristi prilikom implementacije mnogih drugih aritmetičkih jedinica (množači, delitelji itd.) u radu je izvršena detaljna analiza performansi nekoliko arhitektura binarnih sabirača. Cilj ovakve analize bio bi da dizajneru pred kojim su postavljena ograničenja u pogledu željenih performansi sistema koji projektuje olakša izbor najpogodnije arhitekture sabirača.

Skup arhitektura predstavljenih u radu sastoji se od klasičnih sabirača sa linearnim kašnjenjem [1], optimizovanih *carry-chain* arhitektura [2,3] kao i sabirača sa logaritamskim kašnjenjem [4,5]. Pored toga, predložena je i jedna arhitekturna modifikacija koja ima za cilj da smanji jaz između hardverski zahtevnih logaritamskih i vremenski zahtevnih linearnih sabirača. *Unit-gate* model predstavljen u [6] korišćen je, zbog svoje neutralnosti, prilikom evaluacije performansi arhitektura sabirača. Prema ovom modelu, svako monotono dvoulazno kolo (npr. AND, OR, NAND itd.) povećava površinu dizajna za jednu jedinicu i unosi kašnjenje ekvivalentno jednoj jedinici. XOR gejt površinu i kašnjenje dizajna povećava za 2, dok se *m*-to ulazna logička ćelija računa kao *m*-1 jedinica površine i $\lceil \log_2 m \rceil$ jedinica kašnjenja. Sa druge strane, sve arhitekture binarnog sabirača opisane su u VHDL-u i implementirane na modernu FPGA komponentu iz Virtex-6 familije [7]. Rezultati implementacije komentarisani su iz perspektive površine, kašnjenja i potrošnje.

2. DOSADAŠNJI RAD

Počev od XIX veka i pojave mehaničkih mašina za sabiranje, prisutna je stalna težnja ka optimizaciji obavljanja ove aritmetičke operacije. Učiniti proces sabiranja što efikasnijim je čak i danas u fokusu istraživanja mnogih autora. Vreme obavljanja operacije, zauzeće resursa (površine) čipa kao i energija potrebna za obavljanje sabiranja uglavnom su mere kvaliteta sabirača.

Dok neki autori istražuju na tranzistorskom nivou primenjujući različite CMOS logičke stilove i tranzistorske strukture [8, 9, 10, 11], drugi predlažu brojne arhitekturne modifikacije i varijacije [12, 13, 14]. Treći, pak, pokušavaju da „podese“ tranzistor (menjajući njegove dimenzije, napone praga i napajanja) tako da zadata arhitektura bude optimalna u pogledu željenih performansi dizajna [15].

Što se tiče FPGA implementacije binarnih sabirača, još u [16] autori su primetili da modeli za kašnjenje i analizu troškova dizajna razvijani za ASIC tehnologiju nisu od koristi prilikom dizajniranja i implementiranja na FPGA. Tako su autori u [16] predstavili mogućnosti optimizacije *carry-bypass* i *carry-select* arhitektura i pokazali da optimizovane verzije ovih sabirača mogu biti brže u odnosu na klasične *ripple-carry* sabirače implementirane na FPGA. Studija u kojoj su predstavljeni kompromisi između veličine, latencije i frekvencije protočnih preciznih sabirača implementiranih na FPGA može se pronaći u [17]. Neke aritmetičke optimizacije za mapiranje *carry-select* sabirača/inkrementera uz pomoć hardverskih *carry* lanaca koji se mogu naći u savremenim FPGA komponentama predstavljene su u [18]. Predložene arhitekture predstavljaju atraktivne alternative visoko-protočnim *ripple-carry* šemama sabiranja.

Na osnovu rezultata dosadašnjih istraživanja nameće se utisak da ne postoji „srebrni metak“ – arhitektura binarnog sabirača koja će pokazati najbolje performanse (najmanje hardverski, vremenski i energetski zahtevna u isto vreme) bez obzira na tehnologiju koja se koristi za implementaciju.

3. ARHITEKTURE SABIRAČA

Neka su $A=a_{n-1},...a_0$, $B=b_{n-1},...b_0$ i c_{in} dva n-to bitna operanda i ulazni prenos ($c_{in}=c_0$), respektivno. Rezultat sabiranja ($A+B+c_{in}$) je n-to bitni signal sume $S=s_{n-1},...s_0$ zajedno sa indikatorom prekoračenja (c_{out}).

3.1. SABIRAČI SA LINEARNIM KAŠNJENJEM

Dve arhitekture koje pripadaju ovoj grupi sabirača razmatrane su u radu: klasični *ripple-carry* (RC) sabirač sastavljen od n redno vezanih ćelija potpunog sabirača i *carry-block* (CB) sabirač baziran na multiplekseru (vidi Sl. 1). U obe arhitekture kritična putanja proteže se od ulaznog (c_{in}) do izlaznog prenosa (c_{out}) i linearno je proporcionalna broju bitova sabiraka (n). Kako se izlazi ćelije potpunog sabirača mogu izraziti kao:

$$s_i = a_i \oplus b_i \oplus c_i \;;\; c_{i+1} = a_i b_i + a_i c_i + b_i c_i, \qquad (1)$$

njena površina i kašnjenje na osnovu *unit-gate* modela su $A_{FA}=9$ i $T_{FA}=4$, respektivno. Prema tome, vreme procesiranja i površina n-to bitnog *ripple-carry* sabirača su:

$$T_{RC} = (n-1)\cdot T_{FA} + T_{XOR} = 3n-1 \qquad (2)$$

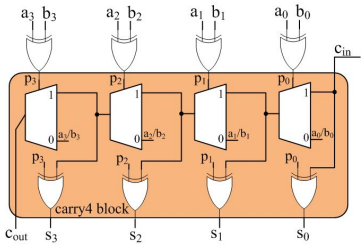
$$A_{RC} = n \cdot A_{FA} = 9n \qquad (3)$$

Što se tiče *carry-block* sabirača sa Sl. 1, kako se multiplekser može okarakterisati sa $A_{MUX}=3$ i $T_{MUX}=2$, vreme procesiranja i površina potrebna za implementaciju ovog sabirača su, respektivno:

$$T_{CB} = (n-1)\cdot T_{MUX} + T_{XOR} = 2n \qquad (4)$$

$$A_{CB} = n \cdot (2A_{XOR} + A_{MUX}) = 7n \qquad (5)$$

Treba napomenuti da je 4-bitni *carry* blok sastavni deo Virtex-6 FPGA slajsa i da je namenjen smanjivanju vremena propagacije carry signala. Smanjivanje vremena propagacije postiže se korišćenjem specijalnih interkonekcija unutar FPGA strukture [7].

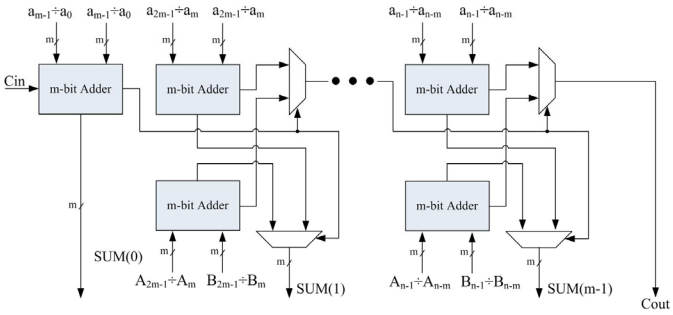


Sl.1. *4-bitni carry-block binarni sabirač.*

3.2. OPTIMIZOVANE CARRY-CHAIN ARHITEKTURE

Glavne prednosti sabirača sa linearnim kašnjenjem su jednostavnost njihove konstrukcije i mala površina čipa potrebna za njihovu implementaciju. Ipak, činjenica da se *carry* signal prostire kroz svaku 1-bitnu ćeliju čini ove sabirače manje pogodnim za implementacije kod kojih je potrebna velika brzina obrade podataka. Zbog toga je skraćivanje kritične putanje ovih sabirača (postiže se uglavnom na račun povećanja površine) cilj mnogih arhitekturnih modifikacija [5]. U radu ćemo analizirati *carry-select* arhitekturu i predložiti jednu *carry-bypass* tehniku.

Carry-select (CSL) sabirači ubrzavaju proces sabiranja time što dupliraju hardver imajući u vidu činjenicu da ulazni prenos može imati samo dve vrednosti (logička 0 ili logička 1). Tako je *n*-to bitni sabirač podeljen na *n/m* blokova od po *m* bitova. U svakom bloku, hardver se replicira kako bi se odredili signali sume i prekoračenja za oba moguća slučaja ulaznog prenosa. Koncept ovakvog sabiranja ilustrovan je na Sl. 2. Multiplekser na kraju bloka propušta jedan od dva dobijena lokalna signala prekoračenja na osnovu signala prekoračenja iz prethodnog bloka.



Sl.2. *Arhitektura carry-select sabirača.*

Kritična putanja u ovoj implementaciji prostire se kroz prvi blok *m*-to bitnog sabirača i mrežu multipleksera sledećih blokova (budući da se sabiranja svih blokova obavljaju paralelno). Imajući to u vidu može se zaključiti da je vreme potrebno za obavljanje sabiranja kod ovakvih arhitektura takođe linearno zavisno od broja ulaznih bitova ali sa nagibom koji je znatno manji nego u slučaju prethodno opisanih sabirača sa linearnim kašnjenjem. Površina kola je od prilike duplo veća.

$$T_{CSL} = T_m + \lceil (n-m)/m \rceil \cdot T_{MUX} \quad (6)$$

$$A_{CSL} = A_m + \lfloor (n-m)/m \rfloor \cdot (2A_m + (m+1)A_{MUX}), \quad (7)$$

gde su *Tm* i *Am* vreme procesiranja i površina potrebna za implementaciju *m*-to bitnog blok sabirača (vidi Sl. 2). Blokovi različitih veličina (*m*) takođe mogu da poboljšaju performanse sabirača [19]. Veličina konkretnog bloka bira se tako da ulazi u multipleksere iz tog bloka stižu u istom vremenskom trenutku. Na primer, ako pretpostavimo da je kašnjenje multipleksera približno isto kašnjenju kroz ćeliju potpunog sabirača, minimalno kašnjenje *carry-select* sabirača ostvaruje se onda kada je veličina prvog bloka 1, drugog bloka 2 itd.

3.3. SABIRAČI SA LOGARITAMSKIM KAŠNJENJEM

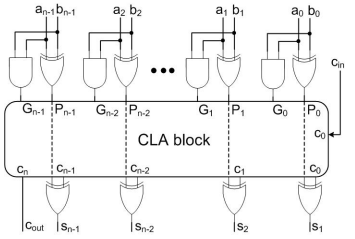
Sabirači ovog tipa su generalno najbrži ali i hardverski najzahtevniji. Propagaciono kašnjenje ovakvih arhitektura logaritamski je proporcionalno broju bitova sabiraka *n*. Arhitektura im se generalno sastoji od tri nivoa logike: (1) nivo pred-procesiranja u kome se računaju *carry-generate* (*G*) i *carry-propagate* (*P*) signali, (2) glavni nivo koji sadrži digitalnu logiku koja računa sve *carry* signale u isto vreme i (3) logika za post-procesiranje pomoću koje se generišu bitovi sume i prekoračenja. U radu će biti razmotrene dve arhitekture ovog tipa: *carry-look-ahead* sabirač i *Sklansky-prefix* sabirač.

Za razliku od sabirača sa linearnim kašnjenjem kod kojih se interni signali prekoračenja generišu sekvencijalno, čineći signal prekoračenja iz ćelije zavisnim od signala ulaznog prenosa, kod *carry-look-ahead* (CLA) implementacija to nije slučaj. Lokalni signali prekoračenja (c_i) računaju se paralelno, uz pomoć *G* i *P* signala koji se dobijaju kao:

$$G_i = a_i \cdot b_i \ ; \ Pi = a_i \oplus b_i \quad (8)$$

Ovo je ilustrovano na Sl. 3. Na osnovu jednačine (8) i činjenice da se signal prekoračenja *i*-te ćelije generiše kada su oba sabirka (a_i i b_i) jednaka 1, ili kad je jedan od sabiraka jednak 1 i postoji ulazni prenos iz prethodne ćelije, signali prekoračenja (*carry*) se mogu paralelno izračunati kao:

$$\begin{aligned} c_0 &= G_0 + P_0 c_{in} \\ c_1 &= G_1 + P_1 c_0 = G_1 + P_1 G_0 + P_1 P_0 c_{in} \\ &\ldots \\ c_j &= G_j + \sum_{i=1}^{j} G_{j-i} \prod_{k=1}^{i} P_{j-k+1} + c_{in} \prod_{p=0}^{j} P_p \end{aligned} \quad (9)$$



Sl.3. *Carry-look-ahead sabirač.*

Bitovi sume su u tom slučaju jednaki:

$$s_i = P_i \oplus c_{i-1} \quad (10)$$

Kritična putanja kod ovih sabirača proteže se od msb bita ulaza (a_{n-1}/b_{n-1}) do msb bita sume (s_{n-1}) i sadrži dva XOR kola i dva AND kola sa $n+1$ ulaza:

$$T_{CLA} = 4 + 2 \cdot \log_2(n+1) \quad (11)$$

Na osnovu jednačina (8), (9) i (10) dobija se da je površina potrebna za implementaciju ovakvog sabirača jednaka:

$$A_{CLA} = \frac{1}{6}n^3 + n^2 + \frac{35}{6}n \quad (12)$$

Loša strana CLA sabirača je ta što izrazi za prekoračenje (a samim tim i hardverska implementacija) postaju vrlo kompleksni za sabirke koji imaju više od 4 bita. Zbog toga se višebitni CLA sabirači obično realizuju modularno, kaskadnim vezivanjem 4-bitnih CLA ćelija.

Prefix sabirači [4] se generalno karakterišu činjenicom da su izlazi ($y_{n-1},y_{n-2},\ldots,y_0$) računaju u rekurzivnoj formi uz pomoć *n*-bitova ulaza ($x_{n-1},x_{n-2},\ldots,x_0$) i asocijativnog binarnog operatora $*$:

$$y_0 = x_0 \ ; \ y_i = x_i * y_{i-1} \ ; \ i = 1,2,\ldots,n-1 \quad (13)$$

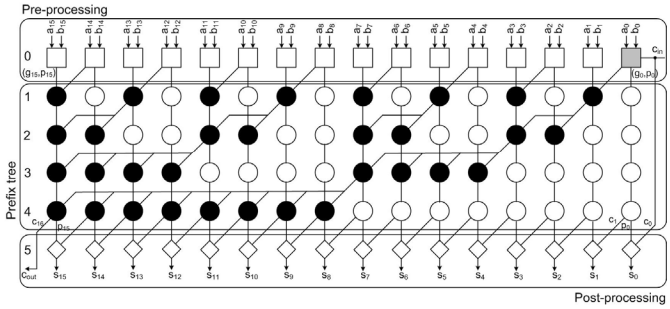
Prema tome, svaki izlazni signal zavisi od ulaznih signala iste ili manje težine. Usled asocijativne osobine ovog generičkog $*$ operatora, operacije se mogu obavljati proizvoljnim redosledom. Takođe, pod-setovi operanada mogu se zajedno grupisati (uvođenjem grupnih varijabli $Y_{i,k}$) kako bi se do

parcijalnih rešenja došlo paralelno. Varijabla $Y^{l}_{i,k}$ biće rezultat operacije nad bitovima $(x_k,x_{k-1},...,x_i)$ na nivou l. Grupne varijable na poslednjem nivou m sadržaće ceo opseg ulaznih bitova, od 0 di i ($Y^{m}_{0,i}$):

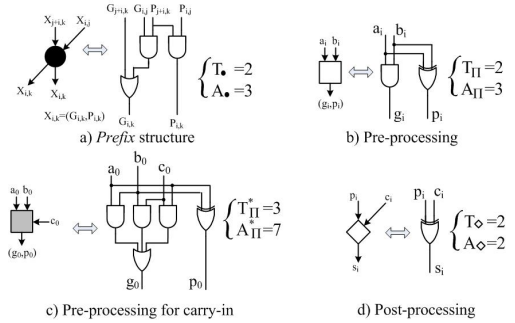
$$Y^{l}_{i,k} = Y^{l-1}_{i,j} * Y^{l-1}_{j+1,k}\ , i < j < k\ ; l = 1,2,...,m \quad (14a)$$

$$Y^{0}_{i,i} = x_i\ ;\ y_i = Y^{m}_{0,i}\ , i = 0,1,...,(n-1) \quad (14b)$$

Iz grupe nekoliko različitih struktura *prefix* sabirača *Sklansky* topologija [20] prikazana na Sl. 4 jedna je od najefikasnijih. Svi međusignali računaju se paralelno uz pomoć struktura sa minimalnim stablom i prosleđuju svim višim nivoima kojima su potrebni.



Sl.4. *Sklansky-prefix struktura sabirača.*



Sl.5. *Unutrašnja struktura blokova Sklansky sabirača.*

Nedostak ove arhitekture je veliki *fan-out* izlaznih čvorova (povećava se sa povećanjem broja bitova sabiraka). Svaki bit (c_i) *carry* vektora računa se uz pomoć signala iz poslednjeg nivoa ($G^{m}_{0,i}, P^{m}_{0,i}$). Bitovi sume dobijaju se nakon faze post-procesiranja. Varijable $G^{l}_{i,k}$ i $P^{l}_{i,k}$ u međunivou l računaju se na osnovu blokova predstavljenih na Sl. 5. Vreme procesiranja i površina *Sklansky* sabirača jednaki su:

$$T_{SA} = T^{*}_{\Pi} + \log_2(n)\cdot T_{\bullet} + T_{\Diamond} \quad (15a)$$

$$= 3 + 2\log_2 n + 2 = 2\log_2 n + 5 \quad (15b)$$

$$A_{SA} = (n-1)\cdot A_{\Pi} + A^{*}_{\Pi} + (1/2\, n\log_2 n)\cdot A_{\bullet} + n\cdot A_{\Diamond} \quad (16a)$$

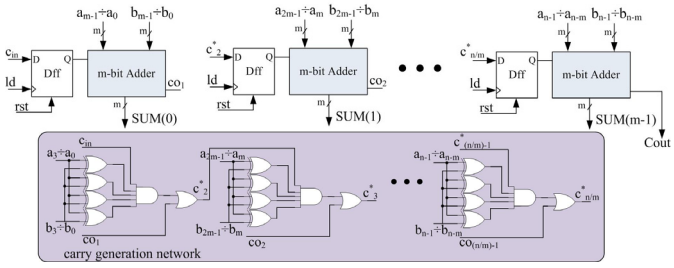
$$= 3(n-1) + 7 + 3(1/2\, n\log_2 n) + 2n \quad (16b)$$

$$= 3/2\, n\log_2 n + 5n + 4 \quad (16c)$$

4. MODIFIKOVANA CARRY-BYPASS TEHNIKA

Kako logaritamski sabirači zahtevaju veliku površinu čipa potrebnu za njihovu implementaciju (naročito sa povećanjem broja bitova sabiraka), a linearni sabirači mogu biti previše spori za neke aplikacije, u radu je predložena jedna *carry-bypass* kompromisna arhitektura koja ima za cilj da smanji jaz između ove dve krajnosti.

Glavna ideja kod *carry-bypass* (bp) pristupa je da se kritična putanja premesti sa *carry* lanca na mrežu koja će generisati *carry* signale (cg) tako da je dodatno povećanje hardvera plaćeno velikim smanjivanjem kašnjenja. Koncept je ilustrovan na Sl. 6. Slično kao kod *carry-select* tehnike, n-to bitni sabirač podeljen je na n/m blokova od po m bitova. Svaki blok sadrži po jedan *D flip-flop* koji blok sabirača obezbeđuje ulazni prenos. Ulazi u *flip-flopove* dobijaju se iz mreže za generisanje *carry* signala.



Sl.6. *Carry-bypass arhitektura sabirača.*

Neka su c^{*}_{i} i co_i ulazni i izlazni prenos i-tog bloka, respektivno (c^{*}_{i} je ulaz u *flip-flop*). Ulazni prenos sledećeg bloka (c^{*}_{i+1}) biće 1 ako su bitovi sume i-tog bloka i ulazni prenos jednaki 1 ili ako je izlazni prenos i-tog bloka jednak 1

$$c^{*}_{i+1} = (A_i \oplus B_i)\cdot c^{*}_{i} + co_i, \quad (17)$$

gde su A_i i B_i m-to bitni sabirci bloka i. *Flip-flopovi* se na početku resetuju pa se nakon toga, pošto se izračunaju u mreži za generisanje prenosa, u njih upisuju c^{*}_{i} bitovi ($i \in [1, n/m]$). Kritična putanja sadrži mrežu za generisanje prenosa, *D flip-flop* i m-to bitni blok sabirač:

$$T_{bp} = T_{cg} + T_{Dff} + T_m \quad (18a)$$

$$= \lceil (n-m)/m \rceil \cdot (3 + \log_2 5) + 3 + T_m, \quad (18b)$$

gde je $T_{Dff}=3$. Površina *carry-bypass* arhitekture neznatno je povećana u poređenju sa klasičnim sabiračima sa linearnim kašnjenjem:

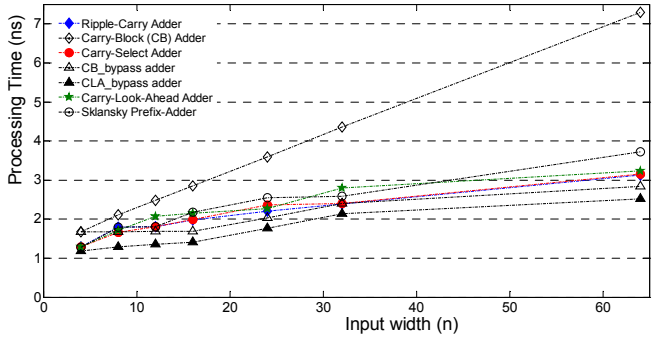
$$A_{bp} = A_{cg} + \lceil n/m \rceil \cdot (A_m + A_{Dff}) \quad (19a)$$

$$= \lceil (n-m)/m \rceil \cdot (3m+1) + \lceil n/m \rceil \cdot (A_m + 4), \quad (19b)$$

gde je $A_{Dff}=4$. Menjajući vrednost m kao i tip blok-sabirača može se uticati na performanse *carry-bypass* sabirača.

5. FPGA IMPLEMENTACIJA

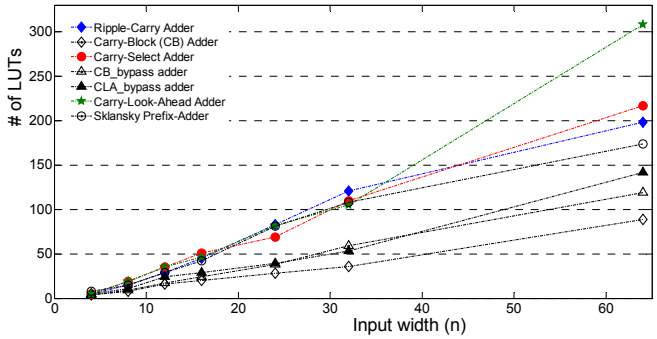
Sve prethodno predstavljene arhitekture binarnih sabirača opisane su u VHDL-u implementirane na FPGA komponentu iz Virtex-6 familije. Za sve modularne arhitekture sabirača (koje se sastoje od n/m blokova) usvojeno je da je parametar m jednak 4. U slučaju implementacije CSL sabirača za m-to bitni blok-sabirač uzet je 4-bitni CLA. Takođe, opisane su dve *carry-bypass* arhitekture: jedna sa CLA sabiračem kao blok-sabiračem (CLA_bp) i druga koja sadrži 4-bitni *carry-block* sabirač (CB_bp) na mesto m-to bitnog sabirača. *Carry-block* sabirač implementiran je na Virtex-6 kao kaskadna realizacija 4-bitnih *carry4* blokova (vidi Sl. 1) koji su sastavni deo njenog slajsa [7]. XPower alat [21] u okviru ISE 12.4 programskog paketa korišćen je za procenu potrošnje sabirača. Dobijena vremena procesiranja i potrebni Virtex-6 hardverski resursi (izraženi brojem LUT-ova) prikazani su na Sl. 7 i 8.



Sl.7. *Komparativna analiza brzine binarnih FPGA sabirača.*

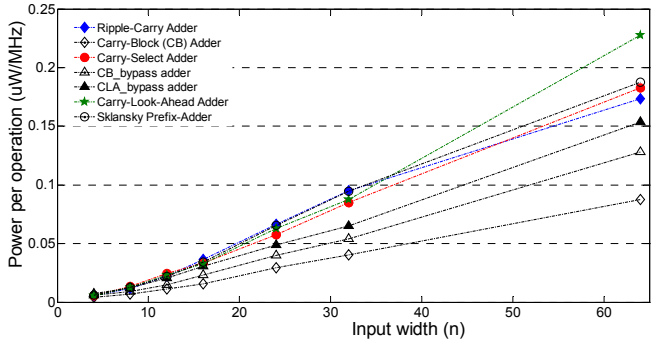
Sa stanovišta brzine i površine može se zaključiti da je *carry-bypass* tehnika predložena u ovom radu pogodna za FPGA implementacije. Mali broj LUT-ova potreban za

implementaciju *carry-block* sabirača posledica je činjenice da ovaj tip sabirača uglavnom koristi *carry4* blokove umesto LUT-ova. Takođe, iako su namenjeni za brzu propagaciju signala prenosa, *carry4* blokovi ne ubrzavaju proces sabiranja kada se koriste u kaskadi. Ovo je, kao što se može videti sa Sl. 7, naročito tačno kada se povećava broj redno vezanih blokova.

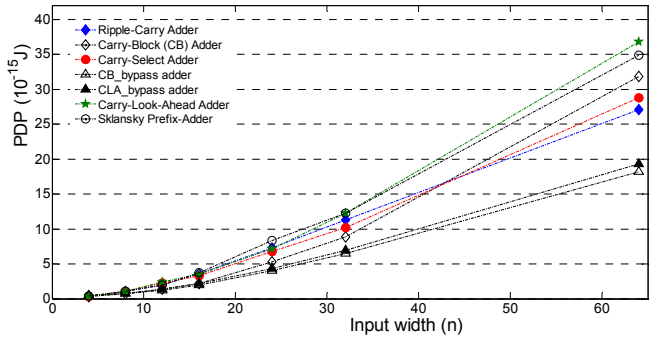


Sl.8. *Potrebni Virtex-6 hardverski resursi.*

Prosečna dinamička potrošnja potrebna za obavljanje jedne operacije sabiranja kao i proizvod između potrošnje i kašnjenja (PDP) kao mere energetske efikasnosti arhitektura prikazani su na Sl. 9 i 10. Sa stanovišta potrošnje takođe je očigledno da su *carry-bypass* arhitekture najpogodinije za implementaciju na FPGA. Potrošnja ovih sabirača najbolje se „prevodi“ u brzinu rada.



Sl.9. *Dinamička potrošnja po jednoj operaciji sabiranja.*



Sl.10. *PDP proizvod skupa arhitektura binarnih sabirača.*

6. ZAKLJUČAK

Prilikom određivanja arhitekture digitalnog dizajna koja bi trebalo da obezbedi najbolje performanse, dizajner mora uzeti u obzir i tehnologiju koja će se koristiti za implementaciju. U radu je, sa stanovišta implementacije na FPGA izvršena analiza skupa arhitektura binarnih sabirača. Takođe je predložena i jedna *carry-bypass* arhitekturna modifikacija za koju se pokazalo da predstavlja najbolji kompromis sa stanovišta potrošnje, brzine rada i zauzeća FPGA resursa.

Predstavljena teoretska i eksperimentalna analiza mogla bi biti od koristi dizajnerima digitalnih sistema prilikom izbora topologije sabirača i implementacione tehnologije.

ZAHVALNOST

Rezultati prikazani u ovom radu ostvareni su u okviru projekta III44004 čiju realizaciju finansira Ministarstvo prosvete i nauke Rebublike Srbije.

LITERATURA

[1] C. Fang, C. Huang, J. Wang and C. Yeh, “Fast and compact dynamic ripple carry adder design,” *IEEE Asia-Pacific Conf. on ASIC*, pp. 25-28, November 2002.

[2] O. Bedrij, “Carry-select adder,” *IRE Trans. Electron. Comput.*, vol. 11, pp. 340-346, June 1962.

[3] M. Alioto, and G. Palumbo, “Optimized design of carry-bypass adders,” *European Conf. on Circuit Theory and Design (ECCTD’01)*, pp. 245-248, August 2001.

[4] J. Chen, *Parallel-prefix structures for binary and modulo {2^n-1,2^n,2^n+1} adders*, PhD thesis, Oklahoma State University, 2008.

[5] R. Zimmermann, *Binary adder architectures for cell-based VLSI and their synthesis*, PhD thesis, Swiss Federal Institute of Technology, 1997.

[6] A. Tyagi, “A reduced area scheme for carry-select adders,” *IEEE Trans. Comput.*, vol. 42, pp. 1162-1170, October 1993.

[7] Xilinx Inc., Virtex-6 FPGA data sheet: DC and Switching Characteristics, v3.4 edition, January 2012.

[8] B. Jose, and D. Radhakrishnan, “Delay optimized redundant binary adders,” *Int. Conf. on Electronics, Circuits and Systems*, pp. 514-517, December 2006.

[9] A. Shams, T. Darwish, and M. Bayoumi, “Performance analysis of low-power 1-bit CMOS full adder cells," *IEEE Trans. on VLSI Systems*, vol. 10, no. 1, pp. 20-29, February 2002.

[10] C. Chang, J. Gu, and M. Zhang, “A Review of 0.18 /spl mu/ m full adder performances for tree structured arithmetic circuits," IEEE Trans. on VLSI Systems, vol. 13, no. 6, pp. 686-695, June 2005.

[11] M. Alioto, G. D. Cataldo, and G. Palumbo, “Mixed full adder topologies for high-performance low-power arithmetic circuits," Microelectronics Journal, vol. 38, no. 1, pp. 130-139, January 2007.

[12] B. Amelifard, F. Fallah, and M. Pedram, “Closing the gap between carry select adder and ripple carry adder: a new class of low-power high performance adders,” *Int. Symp. on Quality of Electronic Design*, pp. 148-152, March 2005.

[13] S. Knowles, “A family of adders,” *15th IEEE Symp. on Computer Arithmetic*, pp. 277-281, August 2001.

[14] F. Kharbash, *Redundant adder architectures*, PhD Thesis, University of Missouri, 2011.

[15] D. Patil, O. Azizi, M. Horowitz, and R. Ho, “Robust energy-efficient adder Topologies,” *18th IEEE Symp. on Computer Arithmetic*, pp. 16-28, June 2007.

[16] X. Shanzen, and W. Yu, “FPGA adders: performance evaluation and optimal design,” *IEEE Design&Test of Computers*, vol. 15, no. 1, pp. 24-29, March 1998.

[17] F. Dinechin, H. Nguyen, and B. Pasca, “Pipelined FPGA adders,” *Int. Conf. on Field Programmable Logic*, pp. 422-427, September 2010.

[18] H. Nguyen, B. Pasca, and T. Preusser, “FPGA-specific arithmetic optimizations of short-latency adders,” *Int. Conf. on Field Programmable Logic*, pp. 232-237, September 2011.

[19] V. Oklobdzija, and E. Barnes., “Some optimal schemes for ALU implementation in VLSI technology,” Proc. of the 7th Symposium on Comp. Arith, pp. 2-8, June 1985.

[20] J. Sklansky, “Conditional sum addition logic,” *IRE Trans. on Electronic Computers,* vol. 9, no. 6, pp. 226-231, June 1960.

[21] XPower Analyzer tutorial, ftp://ftp.xilinx.com/pub/documentation/tutorials/xpowerfpgatutorial.pd

Abstract – In this paper both theoretical and experimental comparative performance analysis of several binary adder architectures are performed. Also, one modified *carry-bypass* technique for adder performance improvement is presented. All architectures are implemented in Virtex-6 FPGA device. Implementation results evidence that here presented *carry-bypass* technique is the best tradeoff for such devices in terms of area, speed and power consumption.

FPGA-SUITED BINARY ADDER ARCHITECTURE

Bojan Jovanović, Milun Jevtić