**RTRK**

**Normalizacija FP broja**

**Nebojsa Markovic**

Normalizacija Floating Point broja

# Problem i predlozeni algoritam

Format ulaznog broja je |1|8|31| , gdje su ovim brojevima, respektivno, predstavljeni znak, eksponent i razlomak.

Takodje, razlomljeni dio dobijen je u formatu |5|23|3| gdje, opet respektivno, imamo cijeli dio broja pretstavljen sa 5 bita, razlomljeni dio (predstavljen sa 23 bita) i dodatna 3 bita koja mogu posluziti pri odredjivanju izlaznog rezultata.  
  
Zadatak jeste da se projektuje sklop koji daje normalizovan broj na izlazu.

Predlozeno rijesenje ili algoritam jeste da:

* Odredimo poziciju na kojoj se nalazi prva jedinica u razlomku
* U zavisnosti od toga da li je prva detektovana jedinica lijevo ili desno od pozicije koja razdvaja cijeli i razlomljeni dio broja, dodati ili oduzeti broj pomjeraja do jedinice eksponentu broja
* U dio za razlomak upisati bite poslije prve jedinice(gdje se ta jedinica podrazumijeva)
* Izvrsiti zaokrugljivanje (ako 3 sledeca bita, koja ne pripadaju rezultatu, predstavljaju broj koji je veci od decimalnog 4 (b'100), tada ako je LSB jednak 1, sabrati razlomljeni dio sa binarnom jedinicom, dok ako je 0 ne raditi nista)
* Nakon ovog koraka, ako je potrebno, ponovo izvrsiti pomijeranje i dodavanje na eksponent

Na kraju na izlaz proslijediti normalizovan broj u jednom od zahtijevanih formata:

* |1|8|7| ,
* |1|8|15| ili
* |1|8|23|

Radi jednostavnije prezentacije idejnog rijesenja, drzacemo se samo zadnjeg navedenog formata, ali principijelno rijesenje je isto za bilo koji od njih.

# Ideja

Problem se moze svesti na nekoliko koraka:

* pronalazenje pozicije na kojoj je MSB jedinica
* uzeti bite od date pozicije do nulte pozicije i ako je potrebno dopuniti nulama
* izvrsiti zaokrugljivanje i dodijeliti razlomljenom dijelu izlaznog broja
* ako je bilo sabiranja razlomljenog dijela sa 1 mantisu cemo siftovati u desno (na MSB se upisjue 0), dok ako nije bilo sabiranja sa 1, vec imamo izracunat razlomljeni dio... Navedimo primjer zast dodajemo nulu: 1.11111111

+0.00000001

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10.00000000

Odakle se jasno vidi da ako pomjerimo zarez za jedno mjesto u lijevo usiftavamo nulu na MSB razlomljenog dijela

* kada racunamo eksponent(da ne bismo komplikovali sa tim da li je jedinica lijevo ili desno od tacke koja razdvaja cijelobrojni i razlomljeni dio) racunacemo ga kao:

exp = exp + CB – poz

gdje je CB = 4 ako nije bilo pomjeranja nakon zaokrugljivanja ili 5 ako jeste bilo

a poz je pozicija prve jedinice koju smo vec izracunali

# Odredjivanje pozicije prve jedinice

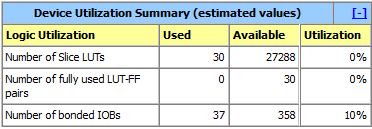
Poziciju prve jedinice mozemo odrediti prioritetnim koderom, ali za 32 ulazna signala to bi bio veliki sklop, sa razlicitim duzinama propagacija(mada ovo nije problem ako smo spremni koristiti nn-ulazna logicka kola, sto nije slucaj, pa dolazimo na ovaj problem ), pa predlazem sledeci sklop za odredjivanje pozicije MSB jedinice:



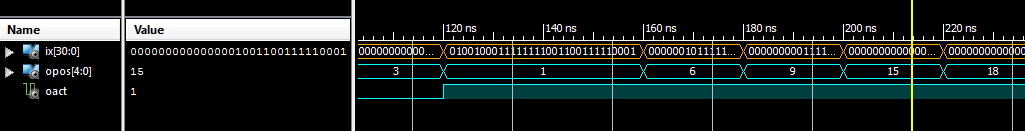
sl. Racunanje pozicije MSB jedinice

Blok MUL radi mnozenje sa 4 i ovdje je prikazan samo da bi ideja bila bolje prikazana. Naime, poziciju jedinice racunamo kao ***4\*blok+offset.*** Za mnozenje sa 4 n treba nam nikakav poseban modul jer je rezultat mnozenja uvijek taj broj sa dvije nule sa desne strane, tj b'X\*b'4 = b'X00

Prioritetni koderi na ulazu, sem kodovane pozicije jedinice daju I izlazni signal koji je ‘1’ u slucaju da uopste postoji jedinica na nekom od njegovih ulaza.



sl. Resursi utroseni za realizaciju modula na Spartan6 cipu



sl. Rezultati simulacije

# Blok za racunanje razlomljenog dijela i zaokrugljivanje

Ako pretpostavimo, za sad, da smo na sabirac doveli bite poslije prve jedinice i u slucaju potrebe dopunili nulama(logiku za ovaj dio imam osmisljenu, ali dodacu je sutra jer mi treba konsultovanje), te da smo odredili bite koji uticu na zaokrugljivanje(R2,R1,R0), Blok za racunanje razlomljenog dijela broja ima sledeci izgled:

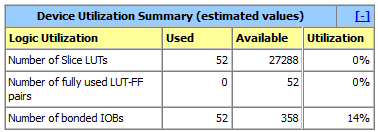


sl. Racunanje razlomljenog dijela broja

Na slici je prikazana ranije iznijeta ideja da se u slucaju sabiranja broja sa 1 pri zaokrugljivanju razlomljenom dijelu broja dodaje nula na MSB poziciju.

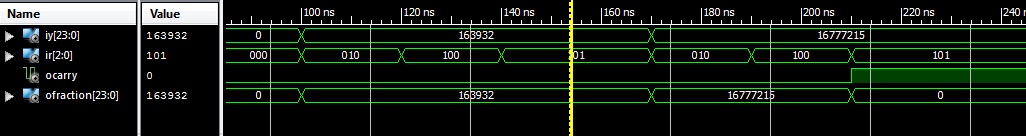
Na slici je prikazan nBitni sabirac koji sabira dobijeni broj sa nulom, dok je na zastavicu za prenos doveden bit koji je rezultat zaokrugljivanja. Ovo se naravno moze pojednostaviti – sabirac koji uvijek sabire sa nulama je ovdje da bi princip bio ocigledniji, dok se naravno moze projektovati kolo koje po potrebi uvecava ulaz za 1.

Rezultat Cout cemo koristiti i kod racunanja eksponenta.



sl. Resursi utroseni za realizaciju modula

Zbog brze realizacije u modulu koji radi sabiranje sa b’1 koristen tip unsigned I sabiranje koje radi sa podacima tog tipa, te koncverzije u std\_logic\_vector. Ovi rezultati se mogu mnogo popraviti projektovanjem modula koji radi sabiranje sa b’1.



sl. Rezultati simulacije modula za odredjivanje razlomljenog dijela i zaokrugljivanje

Vidimo da se sabiranje sa 1 vrsi samo kada je R2R1R0 veci od “100”, sto je dobar rezultat.

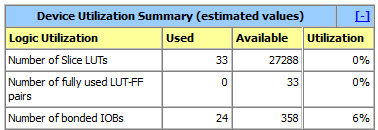
# Racunanje eksponenta

Rezultujuci eksponent jednak je primljenom eksponentu uvecanom za 4(u slucaju da pri zaokrugljivanju Cout jednak nuli) ili za 5(u slucaju da je Cout jednak 1), te umanjen za poziciju na kojoj se nalazi prva jedinica, a tu informaciju nosi nam signal POZICIJA[4:0] iz prvog opisanog bloka. Odavde dolazimo do principijelne seme bloka za racunanje eksponenta normalizovanog broja:

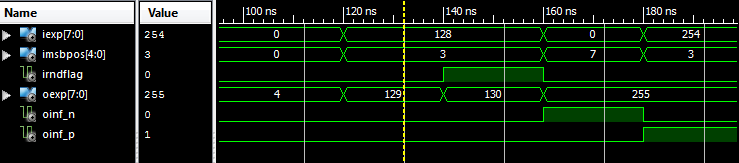


sl. Racunanje eksponenta normalizovanog broja

S obzirom da se radi o principijelnom rijesenju na slikama nisam previse paznje pridavao tome sto POZICIJA i rezultat prvog sabiranja, koje dovodimo na drugi sabirac nemaj isti broj bita – ovo se lako rijesava, a na slikama su ostale prave velicine tih signala da bi se imao bolji uvid u neke druge stvari(kao sto su resursi koji su neophodni) dok je princip jasan i ovako.



sl. Resursi utroseni za realizaciju modula na Spartan6 cipu



sl. Rezultati simulacije

Sa slike gdje su prikayani reyultati simulacije modula ya racunanje eksponenta vidimo da za npr. Eksponent nenormalizovanog broja 128, ako je pozicija MSB jedinice 3, dobijamo izlazni eksponent cija je vrijednost 129, sto je ocekivan rezultat I dobijen je po formuli:

oExp = iExp +4 + iRoundFlag – MSBPossition

Takodje, modul posjeduje signale za signalizaciju izlaska iz opsega (+/- Inf).

# Odredjivanje znacajnih n bita

Znacajni biti su biti iza prve jedinice.

U slucaju hardverskog rijesenja(gdje se koriste poznati moduli), za realizaciju ovog dijela osmislio sam rijesenje gdje mi trebaju tri sloja multipleksera, dok ako rijesenje ne mora da bude uz pomoc gotovih modula, nego mogu da se koriste konverzije u VHDL-u, jednostavno dolazimo do ovih bita na sledeci nacin:

Signal sYRes : std\_logic\_vector(n downto 0) := (others => '0');

sRes(n downto (n- to\_integer(unsigned(POZICIJA)))) <= iFraction(to\_integer(unsigned(POZICIJA))) downto 0);

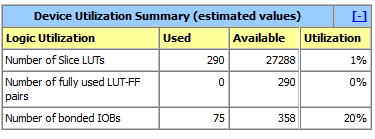
-dodijelili smo signalu sYRes sve nule, a zatim samo bitne bite izmjenili, sto je isti rezultat kao da smo pronasli bitne bite, a zatim do broja n dopunili nulama

Kao rezultat svega ovoga imamo normalizovan ulaznoi broj...

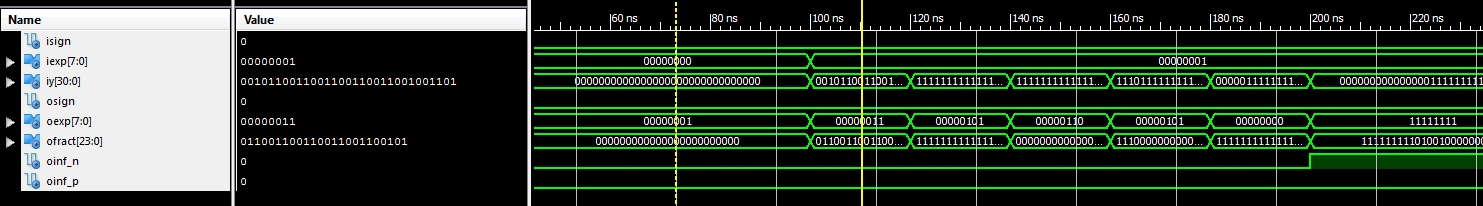
Ako je idejno rijesenje dobro, lako se mpoze dopuniti da radi za bilo koji od izabranih izlaznih formata.



sl. Sema kompletnog modula



sl. Resursi utroseni za realizaciju kompletnog sklopa na Spartan6 cipu



sl. Rezultati simulacije