SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 364

Direktna digitalna sinteza visoke točnosti korištenjem kubične B-spline interpolacije

Marko Brezović

Zagreb, siječanj 2009.

ZAHVALA

Svima koji su mi pomogli savjetima i objašnjenjima prilikom pisanju ovoga rada, zahvaljujem.

Posebna zahvala prof.dr.sc. Davoru Petrinoviću na mentorstvu i usmjeravanju za vrijeme pisanja rada.

Sadržaj

1. Uvod

U današnjem svijetu sve više se izbjegava korištenje analognih signala, preferiraju se digitalni signali. Isti imaju bolja svojstva od analognih, te su pogodnija za korištenje u sklopovima (često se može vidjeti naziv digitalni sklopovi).

Direktna digitalna sinteza sinusa koristi se u određenom eng. software define radio, npr. moderni mobilni telefoni imaju mnogo radiokomunikacijskih funkcionalnosti. Stoga se ti primopredajnici pokušavaju što više digitalizirati. Tako se više ne rade klasični analogni radio primopredajnici, već se signal što prije želi utjerati u digitalnu domenu, te tako koristimo samo neka ulazna i izlazna pojačala. Danas se primljeni signal sa antene samo direktno otipka, uz jednostavnu prilagodbu, eventualno neko pojačanje i odmah se ulazi u AD pretvornik (eng. *Analog Digital Converter*), te se dobiva niz uzoraka koji se na kraju u potpunosti digitalno obrađuju. Da bi se taj digitalni signal mogao modulirati / demodulirati, moraju se koristiti izvori nekih sinusoida koji trebaju biti vrlo precizni, da bi se izvori digitalnih sinusoida, digitalnih uzoraka sinusoide mogli miješati, množiti sa uzorcima signala koji je došao s antene. To je u grubo definirana primjena.

Postoji pet osnovnih tehnika pretvaranja faze u sinus, za direktnu digitalni sintezu:

- Kutna dekompozicija
- Osnovne metode kutne rotacije
- Kompresija sinusne amplitude
- Polinomne aproksimacije
- Kombinacija faze u PSAC-u (eng. *Phase to Sinus Amplitude Conversion*) i DAC-u (*Digital to Analogue Converter*)

2. Osnovna ideja

Zamisao je da se iz faze generira sinus. Fazni akumulator koji generira uzorke faze, može se zamisliti kao neki pilasti napon, zato što fazni akumulator ima zadan korak tj. fazu s kojom se uvećava $\varphi = \varphi + \Delta\varphi$, gdje je φ trenutna faza, a $\Delta\varphi$ korak kojim fazni akumulator uvećava. Fazni akumulator određuje koji uzorak, tj. koji dio sinusoide se obrađuje; npr. fazni akumulator je velik 30 bita, a gornjih 10 bita određuje gdje se uzorak nalazi (na kojem dijelu sinusoide), a donjih 20 bita gdje se uzorak nalazi unutar toga dijela. S obzirom da gornjih 10 bita određuje gdje se uzorak nalazi, znači da je sinusoida, tj. jedan period sinusoide podijeljen na 1024 jednaka dijela. Svaki taj dio opisan je polinomom trećeg stupnja $f(x) = Dx^3 + Cx^2 + Bx + A$ kojim se zapravo aproksimira funkcija sinusa na tom dijelu. Ti su koeficijenti A, B, C i D različiti za svaki dio (njih 1024), te su sami po sebi uzorci sinusoide, neke amplitude i neke faze, iako su oni koeficijenti polinoma koji će dati sinusoidu. Evaluacijom polinoma $f(x)$ dobiva se uzorak sinusoide, gdje je argument x jednak donjih 20 bita faznog akumulatora i govori gdje se nalazim unutra uzorka, 0 na lijevom rubu, 1 gotovo na desnom rubu. Nakon što fazni akumulator odradi za nekoliko koraka i nakon što se za sve te korake evaluiraju određeni polinomi, na izlazu se dobiva sinusoida.

Fazni akumulator može biti i veće širine, veličine. Može se npr. koristit 32 bitni fazni akumulator, znači pribraja se 32 bitni broj u svakom koraku, gdje bi se onda koristilo samo najviših 24 bita koji predstavljaju 24 bitnu vrijednost faze između 0 i 2π. Time se finije odredi korak, a izlazna točnost je koliko treba biti.

Sam izračun koeficijenata A, B, C i D je napravljen u dobivenoj skripti, kodu (naziv skripte je `DDS_sin_A0_simp.m`) za Matlab, te je napravljen na dva načina: numerički, primjenom B-spline transformacijom i dodatno analitički za provjeru. Ti koeficijenti su izračunati za svaki pojedini dio, tj. segment, tako da ta funkcija koja nastaje spajanjem tih segmenata bude maksimalno glatka, tj. da ima kontinuitet vrijednosti prve i druge derivacije tako da na mjestima spoja ne dođe do loma, odnosno da ne postoji mali skok da jedna završi malo niže, a druga skoči gore. Uz to nema loma u prvoj derivaciji, znači da su nagibi na spoju jednaki, tako

da se ne vidi lom u nagibu na tom mjestu. Polinom trećeg stupnja ne može biti glatkiji od toga, nego da mu se izjednače nulta, prva i druga derivacija, tj. vrijednosti prve i druge derivacije su točno poravnate na spoju dva segmenta. Takva funkcija se naziva spline.

2.1. Spline

U matematičkom području brojevne analize, spline je posebna funkcija definirana po dijelovima polinoma. U interpolacijskim problemima, spline interpolacija često se odnosi na polinomnu interpolaciju zato što omogućuje slične rezultate, pa i onda kada se upotrebljavaju polinomi nižeg stupnja.
U informatici, u područjima računarsko potpomognutog dizajna i računarske grafike, pojam spline često se odnosi na po dijelovima polinomne (parametarske) krivulje. Spline su popularne krivulje u tim područjima zbog jednostavnosti konstrukcije, lakoće preciznosti procjene, kao i sposobnosti približavanja kompleksnih oblika kroz okvire krivulja i interaktivne dizajne krivulja.
Termin spline dolazi od fleksibilnih spline uređaja koje upotrebljavaju brodograditelji i grafičari kako bi nacrtali glatke oblike.

2.1.1. Definicija splina

Definicija je limitirana za univerzalni polinomni slučaj, splin je po dijelovima polinomna funkcija.

$$S : [a, b] \to \mathbb{R} \qquad (1)$$

S želim po dijelovima definirati. Kako bi se to postiglo, neka interval (a, b) bude obuhvaćen poretkom k, disjunktnim podintervalima.

$$[t_i, t_{i+1}], 0 \leq i \leq k-1 \qquad (2)$$

$$[a, b] = t_0 \cup t_1 \cup \cdots \cup t_{k-2} \cup t_{k-1} \qquad (3)$$

$$a = t_0 \leq t_1 \leq \cdots \leq t_{k-2} \leq t_{k-1} = b \qquad (4)$$

Svaki taj dio k od $[a, b]$, definira se pomoću polinoma P_i

$$P_i : [t_i, t_{i+1}] \rightarrow \mathbb{R} \tag{5}$$

Na i-tom pod intervalu od $[a, b]$, S je definiran sa P_i

$$S(t) = P_0(t), t_0 \leq t < t_1 \tag{6}$$
$$S(t) = P_1(t), t_1 \leq t < t_2 \tag{7}$$
$$\vdots$$
$$S(t) = P_{k-2}(t), t_{k-2} \leq t < t_{k-1} \tag{8}$$

Dane k točke t_i zovu se čvorovi. Vektor $t = (t_0, \dots, t_{k-1})$ se zove vektorski čvor za splin. Ako su čvorovi jednako udaljeni u intervalu $[a, b]$ kaže se da je splin homogen, a inače je ne homogen.

2.2. B-spline

U matematičkom području numeričke analize, B-spline je spline funkcija koja ima minimalnu potporu s obzirom na stupanj, glatkoću i domenu particije. Glavni teorem kaže da se svaka spline funkcija određenog stupnja, glatkoće i domene particije, može predstaviti kao linearna kombinacija B-spline istoga stupnja, glatkoće i preko iste particije. Izraz B-spline je skovao Isaac Jacob Schoenberg i kratica je od izraza basis spline.

U računarskoj znanosti, područja računarsko potpomognutog dizajna i računarske grafike, pojam B-spline odnosi se na spline krivulju čiji parametri su spline funkcije koje su izražene kao linearne kombinacije B-splina. B-spline je samo generalizacija Bezierove krivulje i može izbjeći Runge fenomen, bez da se poveća stupanj B-spline.

2.2.1. Definicija B-splina

Dane $m+1$ vrijednosti $t_i \in [0,1]$, nazvane su čvorovi, sa

$$t_0 \leq t_1 \leq \cdots \leq t_m \tag{9}$$

B-spline stupnja n je parametarska krivulja

$$S : [t_0, t_m] \rightarrow \mathbb{R}^2 \tag{10}$$

komponirana od osnovnih B-spline stupnja n

$$S(t) = \sum_{i=0}^{m-n-1} P_i b_{j,n}(t), \qquad t \in [t_n, t_{m-n}] \tag{11}$$

P_i su kontrolne točke ili de Boor točke. (Ovdje su $m-n$ kontrolne točke.) Graf se može konstruirati spajanjem de Boorovih točaka sa linijama, počinjući sa P_0 i završavajući sa P_{m-n-1}. Taj graf se zove de Boorov graf.

3. Prilagodba za Spartan-3 FPGA (eng. *Field Programmable Gate Array*) sklop

S obzirom da je ciljani FPGA sklop obitelji Spartan-3, cijela ideja je prilagođena Spartan-3 sklopu i njegovim ograničenjima. U nastavku ću pisati o tome zašto i koliko bitova koristim u određenim dijelovima sklopa.

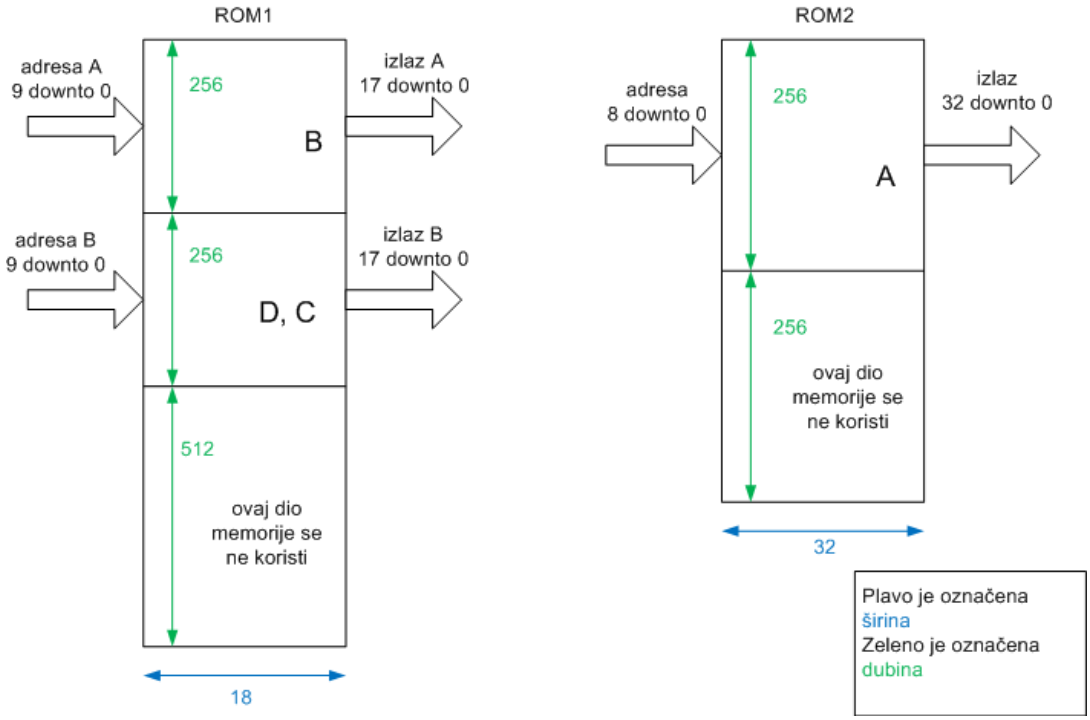
Ulaz u sklop je 26 bitni, gdje je vodeći 26. bit predznak, a ostalih 25 bita je frakcija. Izlaz je 23 bita, s bitom predznaka i 22 bita frakcije. Za prikaz brojeva i kalkulaciju vrijednosti koristi se eng. *fixed point* aritmetika.

Kako je ulaz 26 bitni tada je i fazni akumulator iste širine 26 bita, gdje najviših 8 bita frakcije koristim za određivanje u kojem se intervalu nalazim da bih mogao pročitati potrebne koeficijente A, B, C i D. S obzirom da se koristi 8 bita, znači da je sinusoida podijeljena na 256 jednakih regija, uzoraka te će toliko biti i svakog koeficijenta A, B, C i D. Donjih 17 bita koristim za određivanje gdje se nalazim unutar intervala, kao što je već opisano u cjelini 1. Može se primijetiti da iz izlaza faznog akumulatora koristi samo frakciju 8+17 bita, a vodeći bit, predznak ne koristim zato što je faza uvijek od 0 do 2π. Detaljnija izvedba istog u VHDL-u (eng. *Very-high-speed integrated circuits Hardware Description Language*) biti će objašnjena u daljnjem tekstu.

Koeficijenti A, B, C i D, koji su izračunati pomoću Matlabove skripte, kao što je već ranije napisano, su neki mali realni brojevi. Na tim brojevima je izvršena kvantizacija koeficijenata na izlaznu rezoluciju, širinu koja je redom za koeficijente A, B, C, D - 23, 18, 12, 5 bita te je naravno vodeći bit predznak, a ostali bitovi frakcija. Objašnjenje zašto su baš te veličine koeficijenti, napisano je u nastavku u cjelini 4.

S obzirom da koeficijenti imaju jedan bit za predznak, pa ostali bitovi čine frakciju, što znači da su u rasponu od -1 do 1. Fizički, ti brojevi nisu u rasponu od -1 do 1, koeficijenti A, B, C i D, oni su puno manji realni brojevi. Te brojeve se poveća, po modulu, tako da ih se pomnoži sa 2^N tako da ne prijeđu ±1 i da postoje koeficijenti koji koriste sve bitove. Znači da sigurno postoje koeficijent A,

B, C i D, najveći od istih A, B, C i D, koji su nakon normalizacije u rasponu od ±0,5 do ±1. Stoga su normalizirani množenjem 2^{22}, zato što je izlaz 22 bita, da bi se dobili potrebni brojevi, tj. koeficijenti za pohranu u ROM (eng. *Read-only Memory*) memoriju. Koeficijenti su također dodatno skalirani s 0,99999 tj. amplituda se smanjila na 0,99999. Kada je bila 1 dolazilo je do preljeva, a da bi se to spriječilo koeficijenti su skalirani gdje je onda amplituda izlaza iz sklopa ±0,99999.
U samom modelu za pohranu koeficijenata koristi se blok (eng. *block) RAM* (eng. *Random Access Memory*), ali kao ROM, jer se nikada ne piše, već samo čita iz memorije. Slika 3.-1 prikazuje izgled i smještaj koeficijenata u memoriju.
Za koeficijente B, C i D koristi se jedna dvo-portna memorija (eng. *dual-port memory*) veličine 1Kx18(x16 + 2 paritet), gdje se uvijek sa jednog porta čita koeficijent B koji je 18 bita, a sa drugog porta uvijek koeficijenti C i D koji su ukupno 17 bita. Paritet memorije koristi se za podatke, a ne kao paritet, stoga pola memorije ostaje neiskorišteno, jer se sa jednog porta pristupa prvih 256 lokacija, a sa drugog drugih 256 lokacija.
Za koeficijent A koristi se jedna jedno-portna memorija (eng. *single-port memory*) veličine 512x36(32 + 4 paritet), ovdje se ne koristi paritet već samo 23 bita od 32. Također i ovdje pola memorije ostaje neiskorišteno. Detaljnije izvedbe memorija u VHDL-u biti će opisane u nastavku teksta.



Slika 3.-1 Izgled i prikaz koeficijenata u memoriji

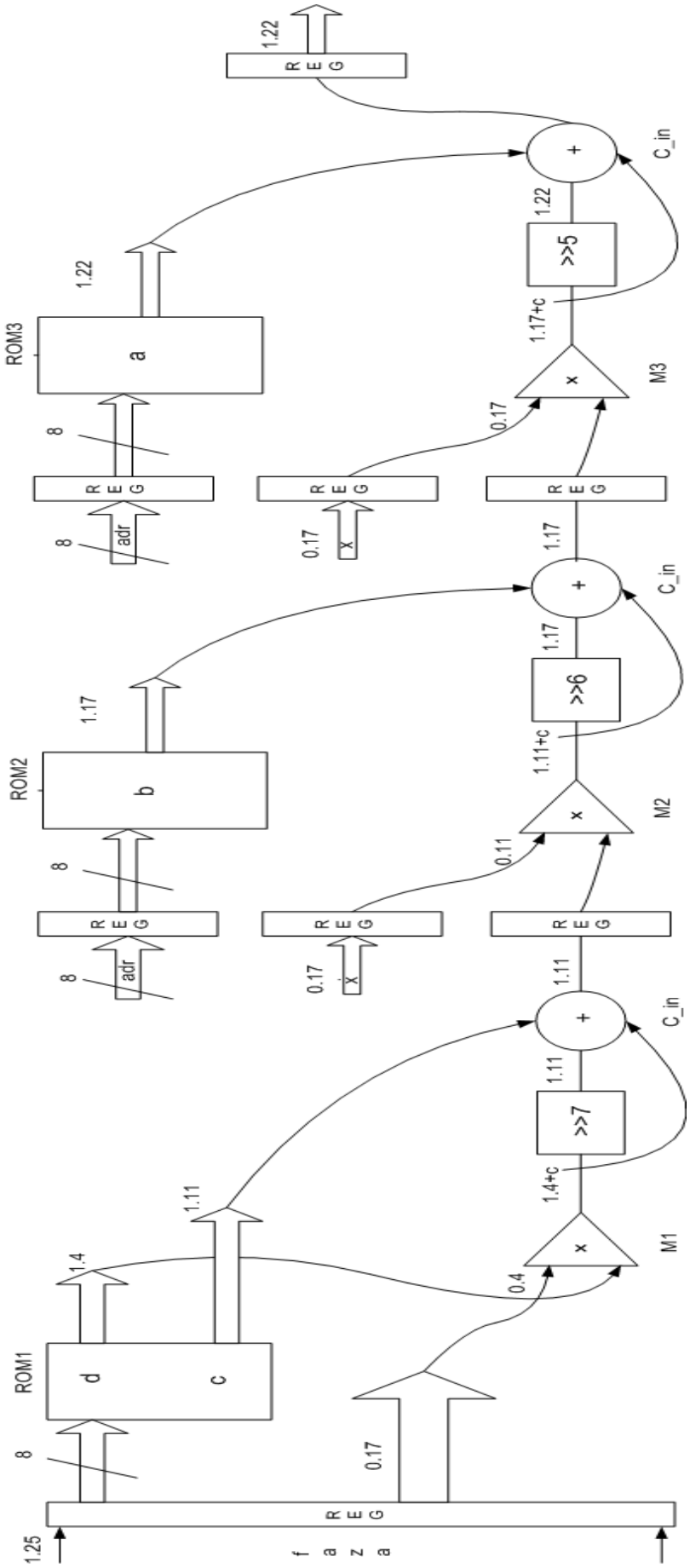
Kako je već rečeno svaki uzorak ima svoj polinom trećeg stupnja,
$f(x) = Dx^3 + Cx^2 + Bx + A$.
Evaluacija tako zapisanog polinoma u Spartanu je moguća, ali bi trošila previše resursa, sama izvedba bila bi kompliciranija i najvažnije, sporije bi radila tj. evaluacija polinoma bi bila spora. Da bi se izbjegli svi mogući problemi, polinom se zapiše drugačije, primjenom Hornerovog pravila.

Polinom trećeg stupnja primjenom Hornerovog pravila tada se zapiše ovako:
$f(x) = \big((Dx + C)x + B\big)x + A$.
Ovako zapisan polinom puno je jednostavnije prikazati u sklopovlju Spartana. Sada se gotovo uvijek radi jedno množenje i jedno zbrajanje, a za evaluaciju polinoma trećeg stupnja potrebna su tri množenja i zbrajanja.

U modelu se koriste još po tri množila (eng. *multiplier*), posmak u desno (eng. *right shift*) i zbrajala (eng*. adder*).
Sva tri množila koja se koriste su jednaka, s obzirom da u Spartan-3 postoje gotova množila 18x18 bita. Jedina razlika između njih je u širini bitova množitelja i u širini umnoška. Na jedan ulaz množila će dolaziti x iz izlaza faznog akumulatora, dakle donjih 17 bita, ali u prvo najviših 4 bita od 17, u drugo najviših 11 i u treće svih 17 bita. Ti ulazi su uvijek bez predznaka (eng. *unsigned*), znači da je najviši bit uvijek u nuli iza kojega slijede ostali bitovi. Na drugi ulaz prvog množila prvo se dovodi koeficijent D, a nakon posmaka i zbrajanja dovodi se na drugo množilo te isto tako i na treće, razlog tome je Hornerov zapis polinoma. Izlazi su jednaki ulazima sa predznakom plus jedan bit, koji će biti ulazni bit prijenosa u zbrajalo. Razlog odabranih širina ulaza i izlaza objašnjen je u nastavku teksta.
Tri posmaka u desno zapravo nisu pravi sklopovi za posmak, već se radi „višestruko spajanje iste žice", jednostavno se vodeći bit proširi N puta. S obzirom da su koeficijenti normalizirani te su u memoriji jednako zapisani od -1 do 1, ali se zna da će umnožak biti 2^N puta manje važan od koeficijenta s kojim će se zbrajati. Zato što je umnožak 2^N puta manje težine, manje važan od broja s kojim će se zbrajati, stoga se radi posmak za N mjesta da bi se brojeve pravilno zbrojilo, da se bitovi „poravnaju" na odgovarajuće težine. Npr. ako je umnožak oblika 1.4 on

može prikazati brojeve od $-\frac{16}{16}$ do $\frac{15}{16}$, 32 razine, a zbraja se sa koeficijentom 1.11 koji ima 2048 razina. Tada se umnožak mora posmaknuti u desno za 7 mjesta da bi ispravno zbrojio umnožak sa koeficijentom. Posmake u desno radi se za 7, 6 i 5 mjesta.
Tri zbrajala su obična zbrajala sa ulazom za prijenos koji je, kao što je gore napisano, jednak „posljednjem" bitu umnoška. Prvo zbrajalo je 12 bitno, drugo 18 bitno i treće 23 bitno, stoga zbrajaju brojeve s predznakom (eng. *signed*).
Izvedbe množila, posmak i zbrajala u VHDL-u će biti opisane u nastavku teksta.

Blok shema koja slijedi prikazuje moguću izvedbu sustava i ujedno prikazuje neke gore opisane sklopove.



Slika 3.-2 Blok shema moguće izvedbe modela

4. Pogreška u modelu

Kao i svi modeli tako i ovaj nije savršen, tj. postoji određena greška u izračunu modeliranja sinusoide.

Na izlazu ima 22 bita i ti svi bitovi nisu točni. Polinomna aproksimacija nije savršena, ona je dobra i s njom se može vrlo dobro modelirati, ali ima grešku. Greška je već to što se koristi polinom, a ne sinus. Znači kada bi se imali idealni realni koeficijenti A, B, C i D polinoma, već to što se koriste ti koeficijenti, a ne sinus, uzrokuje neku pogrešku. Ta pogreška je najveća negdje oko sredine intervala, između dva uzorka, zato što je polinom interpolacijski polinom i on mora savršeno točno proći kroz uzorke. Znači da je na rubovima intervala to savršeno točni uzorak, jer je x na rubu intervala nula pa imamo polinom:

$$f(0) = \big((D \times 0 + C) \times 0 + B\big) \times 0 + A = A$$

gdje ostaje A i ako je A točan onda je i uzorak točan. Ista stvar je ako je x gotovo 1 (0,99999...) gdje je onda taj polinom gotovo jednak zbroju svih koeficijenata. Najveća greška je, kao što je napisano, kada je $x = 0{,}5$ zato što se tada nalazim najdalje od rubova, stoga je tu najveće odstupanje od pravoga sinusa.
Nije nebitno kakva je ta greška, kako se ona raspoređuje, distribuira u spektru. Želi se postići to da, ako se uzme ta modelirana sinusoida, njezini uzorci i izračuna se Fourierova transformacija, da se ima samo šiljak na toj željenoj frekvenciji, a da je sve ostalo kao ujednačeni šum kroz cijeli spektar. Znači da ne postoje neki šiljci, zato što su lažni harmonici neželjeni. Bilo bi jako loše kada bi se na izlazu dobivala sinusoida kojoj bi greška bila koncentrirana u nekim šiljcima, zato što bi onda bile relativno velike amplitude u odnosu na glavni šiljak, stoga se mora imati greška koja je ujednačena kroz cijeli spektar da bi se imalo male amplitude bez velikih izraženih šiljaka.
Ta greška koja nastaje, zato što sam sinus zamijenio polinomom trećeg stupnja po odsječcima definiranog, označava koliko je taj model kao takav loš, eng. *baseline error*.

4.1. Greška kod kvantizacije koeficijenata

S obzirom da postoji neka osnovna pogreška modela, tada postoji i jaka limitiranost s točnošću modela. Tada nema potrebe da koeficijenti budu zapisani točnije nego je to ograničenje. Npr. ako je utjecaj kvantizacije na koeficijent D toliko mali, tj. da je unutar granica greške samog modela, tada nema potrebe da se koristi više bita od 4 (plus 1 bit predznaka). Zato su koeficijenti A, B, C i D kvantizirani redom 1.4, 1.11, 1.17 i 1.22, s obzirom da je taj doprinos kvantizacije koeficijenata manji ili jednak nego što je greška samog modela i kada se zbroje sve te pogreške i dalje smo u nekim granicama greške modela.

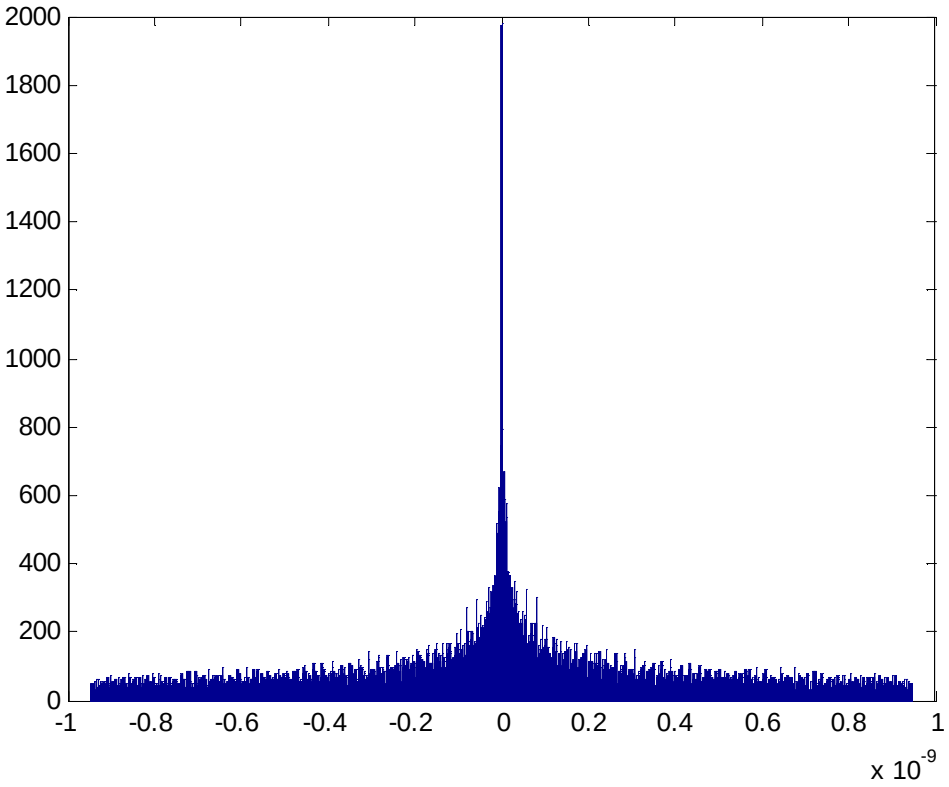
4.2. Greška kod umnoška

Jednako kao i u odjeljku 4.1. može se gledati i za množila. Kada se pomnože dva broja dobije se umnožak dvostruke širine npr. 0.4×1.4=1.8. Sada se od tih 1.8 odbace donjih 4 bita i vidi se koliko je s time greške uneseno, ako je manje od same greške modela, onda ta donja 4 bita nisu značajna i slobodno se odbace. Zato su izlazi množila kvantizirani na 1.4, 1.11 i 1.17 i svugdje plus bit za prijenos, eng. *carry* s kojim se zaokružuju vrijednosti.
Drugi ulaz koji je x, 17 bitni broj ne mora biti točniji od umnoška, izlaza množila i tada se slobodno može skratiti, pa se koriste rezolucije 0.4, 0.11 i 0.17.

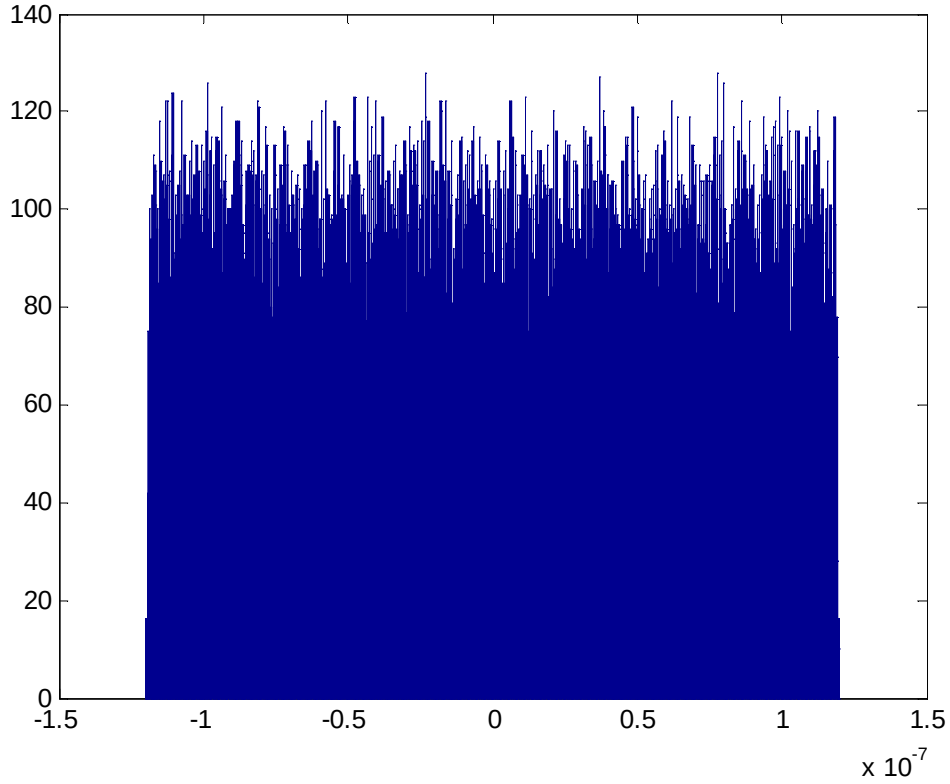
4.3. Prikaz grafova iz Matlaba

Dobiveni kod za Matlab može raditi generički za bilo koju rezoluciju pojedinih koeficijenata A, B, C i D i za bilo koji broj intervala, uzoraka sinusoide. Kako je već napisano, ciljana arhitektura je Spartan-3 pa je tako i Matlab kod prilagođen, gdje su širine 8 bita za tablice koeficijenata, 25 bita za fazni akumulator i 22 bita je izlazna rezolucija sinusa.
Slike (histogrami) koje slijede prikazuju pogreške interpolacija kao razliku idealne

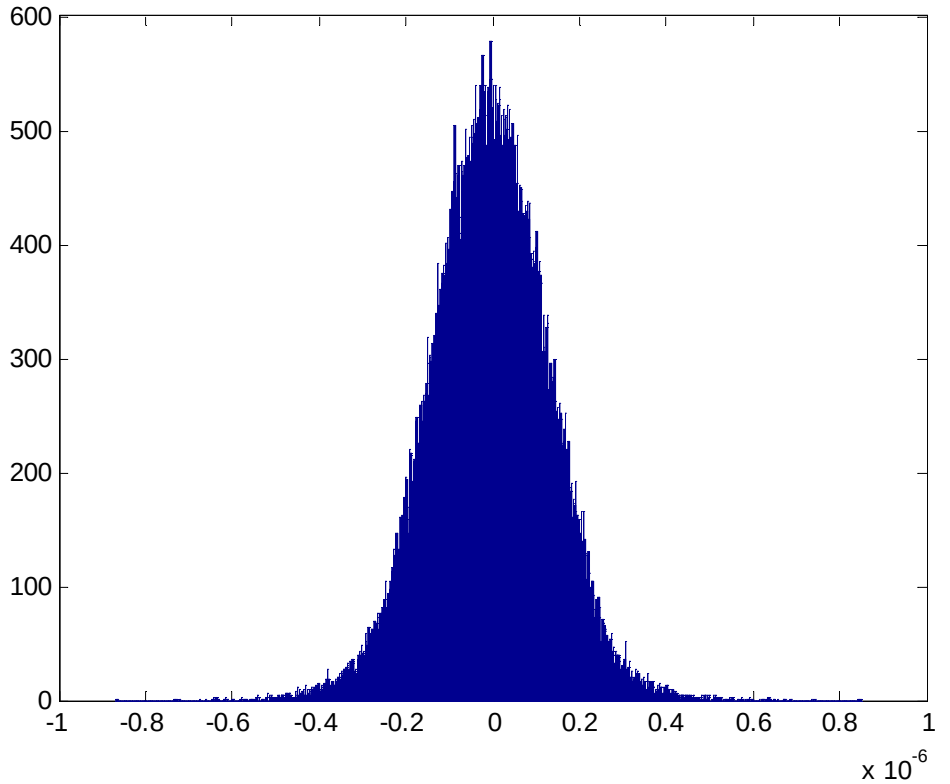
vrijednosti sinusa za zadanu ulaznu fazu i interpolacije izračunate kao eng. *base-line* sustav na osnovu idealnih eng. *floating-point* koeficijenata, a ulazna faza je 1024 . Prva slika je za idealnu eng. *floating-point* verziju, druga za eng. *floating-point* verziju kod koje je izlazno rješenje skraćeno na izlaznu rezoluciju od 22 bita i treća za eng. *fixed point* verziju.



Slika 4.3.-1 Greška za idealnu eng. *floating-point* verziju



Slika 4.3.-2 Greška za eng. *floating-point* verziju sa skraćenim izlazom



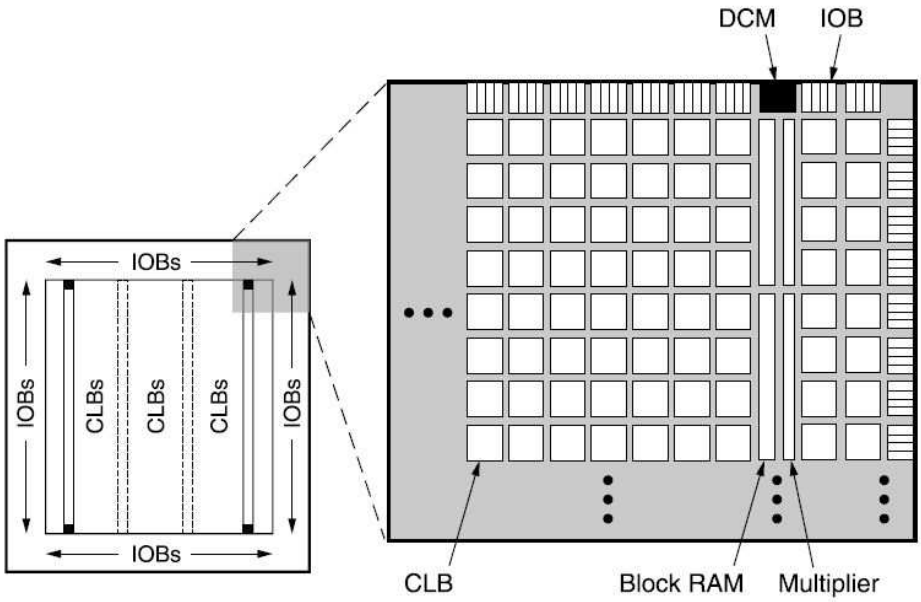
Slika 4.3.-1 Greška za eng. *fixed point* verziju

5. Xilinx Spartan-3 FPGA sklop

Arhitektura Spartan-3 FPGA sklopova sastoji se od pet osnovnih programibilnih funkcijskih jedinica na integriranom krugu (eng. *chip*).

- CLB blokovi (eng. *Configurable Logic Blocks*) sadrže RAM zasnovane funkcijske tablice za izvedbu logičkih i memorijskih elemenata. CLB blokovi mogu biti programirani da izvode različite logičke funkcije ili da služe za pohranu podataka.
- Ulazno izlazi blokovi (eng. *Input/Output Blocks, IOB*) kontroliraju protok podataka između ulazno izlaznih pinova i unutarnje logike. Svaki IOB podržava dvosmjerni protok podataka i stanje visoke impedancije. Dostupno je 26 različitih naponskih razina, uključujući 8 diferencijalnih standarda. Digitalno kontrolirana impedancija omogućava automatsko terminiranje signala čime uvelike pojednostavnjuje izradu sklopa.
- Memorijski blokovi (eng. *Block RAM*) omogućuju pohranu podataka u obliku od 18 Kbit dvo-portnih blokova (eng. *dual-port blocks*).
- Blokovi za množenje imaju 18 bitne ulaze i računaju 36 bitni umnožak.
- Blokovi za digitalnu kontrolu perioda signala vremenskog vođenja (eng. *Digital Clock Manager, DCM*) sinkroniziraju signal vremenskog vođenja po cijelom sklopu, omogućavaju množenje, dijeljenje i fazni pomak perioda signala vremenskog vođenja.

U daljnjem tekstu neću opisivati sve prethodno navedene osnovne elemente, već samo memorijske blokove i blokove za množenje, s obzirom da samo njih koristim u svojemu modelu.



Slika 5.-1 Arhitektura Spartan-3 sklopova [1]

5.1. Blok RAM

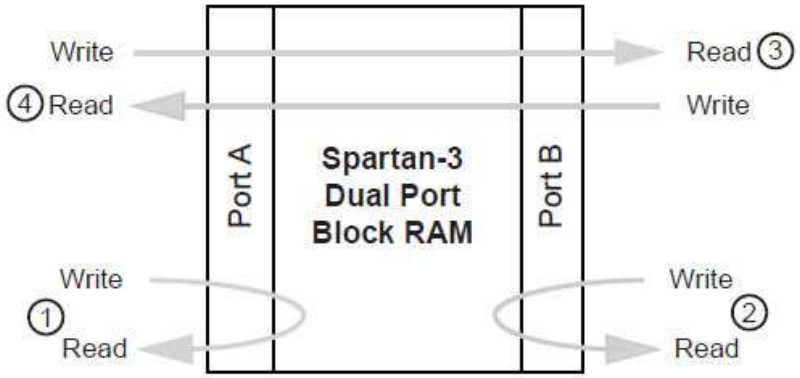
Omjer širine i visine svakog bloka RAM-a je konfigurabilan. Nadalje, više blokova može se poredati stepeničasto kako bi se stvorile još šire i više memorije.

Izbor između osnovnih funkcija određuje kako funkcionira RAM blok, kao dvo-portna ili jedo-portna memorija. Ime forme RAMB16_S[WA]_S[WB] poziva osnovnu dvo-portnu memoriju, gdje cijeli brojevi WA i WB određuju širinu staze ukupnih podataka na ulazima WA i WB. Stoga je RAMB16_S9_S18, dvo-portni RAM sa 9 bitnom širinom ulaza A i širinom ulaza B od 18 bita. Ime oblika RAMB16-S[w] definira jedno-portnu osnovnu memoriju, gdje cijeli broj w određuje ukupnu širinu prolaska podataka samog ulaza. RAMB16_S18 je jedno-portni RAM sa ulazom širine 18-bita.

Blok RAM i množila su interno povezani, stoga je dopušten simultani rad; ali s obzirom da množila dijele ulaze s bitovima viših podataka blok RAM-a, najveća širina protoka podataka bloka RAM-a u ovom slučaju je 18 bita.

Blok RAM ima dvo-portnu strukturu. Dva identična ulazna porta, A i B dopuštaju neovisan pristup zajedničkom blok RAM-u, koji ima maksimalni kapacitet od 18,432 bita- ili 16,384 bita kada se ne upotrebljavaju paritetne linije. Svaki port ima svoj nezavisan set podataka, kontrole i takt (eng. *clock*) za sinkrone

opcije čitanja i pisanja. Postoje četiri osnovna puta podataka, kao što je prikazano na slici 5.1.-1: (1) piši i čitaj s Port A, (2) piši i čitaj s Port B, (3) prijenos podataka s Port A u Port B i (4) prijenos podataka iz Port B u Port A.

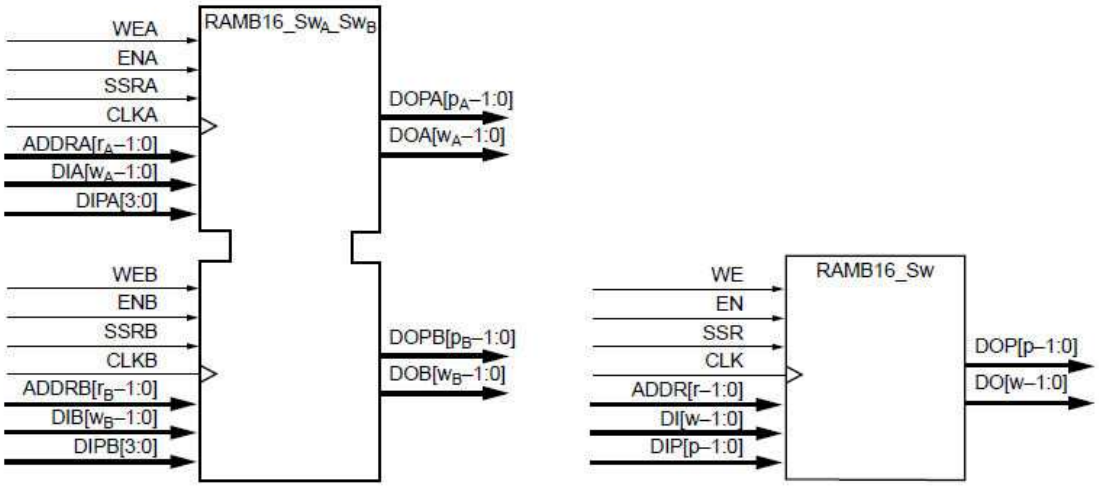


Slika 5.1.-1 Mogući načini rada memorije [1]

U načinima rada (1) i (2), portovi imaju jednaku širinu podatkovne sabirnice, a u načinima rada (3) i (4) portovi imaju različitu širinu podataka. U tablici 5.1.-1 navedeni su svi mogući kapaciteti blok RAM-a koje se može konfigurirati.

Tablica 5.1.-1 Konfiguracije blok memorije [2]

Ukupni bitovi RAM-a, uključujući paritet	18 432 (16K podaci + 2K paritet)
Organizacija memorije	16Kx1 8Kx2 4Kx4 2Kx8 (bez pariteta) 2Kx9 (x8 + 1 za paritet) 1Kx16 (bez pariteta) 1Kx18 (x16 + 2 za paritet) 512x32 (bez pariteta) 512x36 (x32 + 4 za paritet) 256x72 (samo jedno-portna memorija)



Slika 5.1.-2 Dvo-portna i jedno-portna memorija [1]

Na slici 5.1.-2 su prikazane memorije kao dvo-portna i jedno-portna memorija gdje su w_A i w_B cijeli brojevi koji predstavljaju ukupnu širinu podataka, p_A i p_B su cijeli brojevi koji označavaju broj paritetnih bitova, r_A i r_B su cijeli brojevi koji predstavljaju širinu adresne sabirnice i signali CLK, WE, EN i SSR koji imaju mogućnost obrnutog polariteta. Na slici je također jasno označeno koji signali su ulazni, a koji izlazni.

Jedina razlika između signala dvo-portne i jedno-portne memorije je u tome što signali dvo-portne memorije imaju uz oznaku signala A i B.

Opis signala memorije sa prethodne slike 5.1.-2:

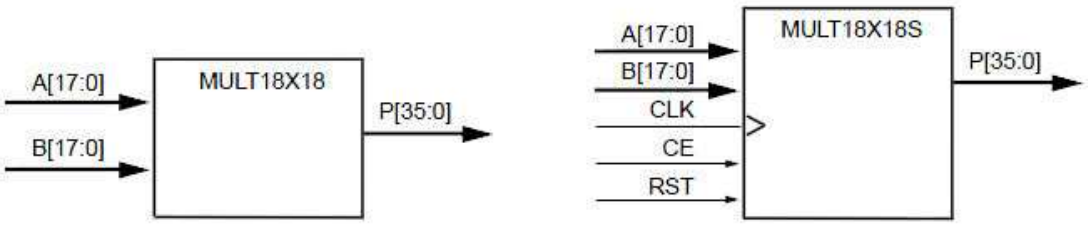
- ADDR – adresa podatka, eng. *Address Bus*
  Adresna sabirnica odabire lokaciju memorije na koju će se pisati ili čitati. Širina (w) podataka određuje broj dostupnih adresnih linija (r). Kada god je ulaz omogućen (EN = visoko, eng. *High*), adresni prijelazi moraju ispuniti listu ulaza i održavati vremena s obzirom na ulazni takt (CLK). Ovaj zahtjev mora biti ispunjen, iako izlaz čitanja RAM-a nije od važnosti.
- DI – podatkovni ulaz, eng. *Data Input Bus*
  Podatak sa ulazne sabirnice DI je zapisan na adresiranu memorijsku lokaciju, koja je adresirana na omogućeni aktivni CLK brid.

- DIP – ulaz pariteta, eng. *Parity Data Input(s)*
  Paritetni ulazi predstavljaju dodatne bitove uključene u ulazni put podataka kako bi podržali detekciju pogreške. Broj paritetnih bitova „p" uključenih u DI (kao i u DO sabirnicu) ovisi o ukupnoj širini podataka.
- DO – podatkovni izlaz, eng. *Data Output Bus*
  Osnovni pristup podacima događa se kada god je WE neaktivan. DO izlazi kopiraju podatke pohranjene na adresiranoj memorijskoj lokaciji.
  Dostupnost podacima sa konstatiranim WE također je moguć, ako se odabere jedan od sljedeća dva atributa: WRITE_FIRST i READ_FIRST. WRITE_FIRST simultano predstavlja novi ulazni podatak na DO izlaz i piše podatak na adresiranu lokaciju RAM-a. READ_FIRST predstavlja stari podatak spremljen u RAM na DO izlaz, istodobno pišući novi podatak u RAM. Treći atribut, NO_CHANGE, ne radi promjene na DO izlazu za vrijeme zapisivanja.
- DOP – izlaz pariteta, eng. *Parity Data Output(s)*
  Isto kao i za DIP (pogledati gore).
- WE – omogući pisanje, eng. *Write Enable*
  Kada je potvrđen sa EN, ovaj ulaz omogućava pisanje podatka u RAM. U tom slučaju, podatak pristupa atributima WRITE_FIRST, READ_FIRST ili NO_CHANGE određuju jesu li i kako je podatak aktualiziran (eng. *updated*) na DO izlaz.
  Kada je WE neaktivan sa konstatiranim EN, operacije čitanja su ipak moguće. U tom slučaju, transparentni zaporni sklop prosljeđuje podatak iz adresirane memorijske lokacije na DO izlaz.
- EN – omogući takt, eng. *Clock Enable*
  Kada je potvrđen, ovaj ulaz omogućuje CLK signal da po redu sinkronizira blok RAM funkcije: pisanje podatka na DI ulaz (kada je WE potvrđen), aktualizaciju podatka na DO izlaz, kao i postavljanje/resetiranje zapornim sklopom DO izlaza.
  Kada nije potvrđen, gore navedene funkcije su onemogućene.

- SSRA – postavi/resetiraj, eng. *Set/Reset*
  Kada je potvrđen, ovaj izvod prisiljava DO izlaz zapornim sklopom na vrijednost na koju je podešen SRVAL atribut. Postavljanje/resetiranje na jednom portu nema učinka na rad ostalih, niti smeta sadržaju podataka u memoriji. Sinkroniziran je na CLK signal.
- CLK – takt, eng. *Clock*
  Ovaj ulaz prihvaća signal takta prema kojem su sinkronizirane operacije čitanja i pisanja. Svi povezani portovi moraju uskladiti postavljanje vremena (eng. *setup times*) u skladu s aktivnim bridom signala takta. Sabirnica izlaza podatka odgovara nakon što se izlazni takt kašnjenja poveže prema aktivnom bridu signala takta.

5.2. Blokovi za množenje

Spartan-3 sklop ima ugrađena množila koja prihvaćaju dvije 18 bitne riječi na ulazu, a na izlazu imaju 36 bitni umnožak. Ulazne sabirnice za množitelja prihvaćaju podatke zapisane u dvojnom komplementu, svih 18 bita s predznakom (eng. *signed*) ili 17 bita bez predznaka (eng. *unsigned*). Jedno takvo množilo je pridruženo svakom blok RAM-u na ploči.
Fizička blizina istih osigurava efektivno korištenje podataka. Kaskadni množitelji dopuštaju množenicima više od tri u broju i širinu veću od 18 bita.



Slika 4.2.-1 Osnovna množila [1]

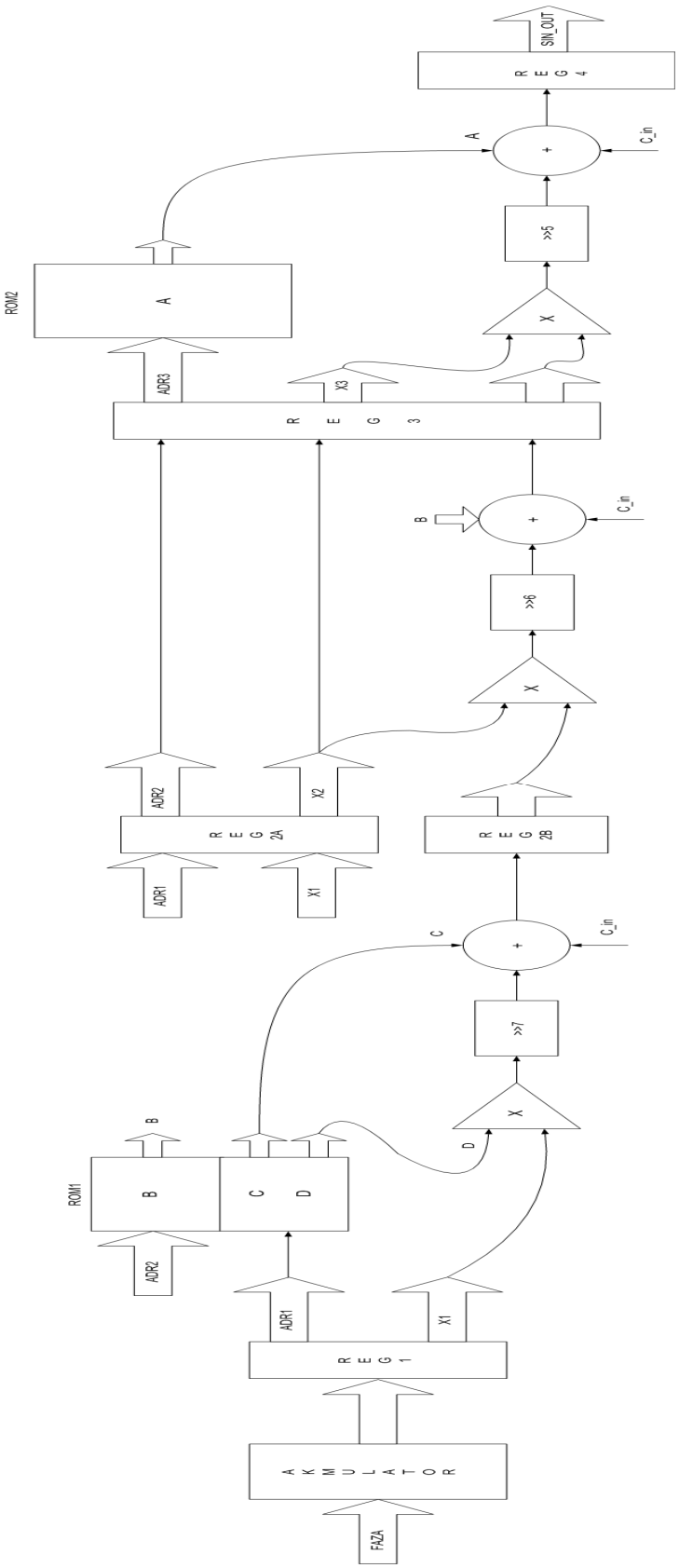
Opis signala sa gornje slike:

- A[17:0]
  Primjeni jedan 18 bitni množitelj na ove ulaze. Osnovni MULT18X18S

zahtijeva pristupno vrijeme (eng. *setup time*) prije omogućenog rastućeg brida CLK.

- B[17:0]
  Primjeni drugi 18 bitni množitelj na ove ulaze. Osnovni MULT18X18S zahtijeva pristupno vrijeme prije omogućenog rastućeg brida CLK.
- P[35:0]
  Izlaz na P sabirnici je 36 bitni umnožak množitelja A i B. Ako je osnovni MULT18X18S, omogućeni rastući brid CLK ažurira P sabirnicu.
- CLK
  CLK je samo ulaz u osnovni MULT18X18S. Taktni signal primijenjen na ovaj ulaz, kada je CE omogućen, ažurira izlazni registar koji upravlja P sabirnicom.
- CE
  CE je samo ulaz u osnovni MULT18X18S. On omogućava CLK signal. Potvrda tog ulaza omogućava CLK signalu da ažurira P sabirnicu.
- RST
  RST je samo ulaz u osnovni MULT18X18S. Potvrda tog ulaza resetira izlazni registar na omogućeni, rastući CLK brid, što prisiljava P sabirnicu na nulu.

6. Blok shema modela

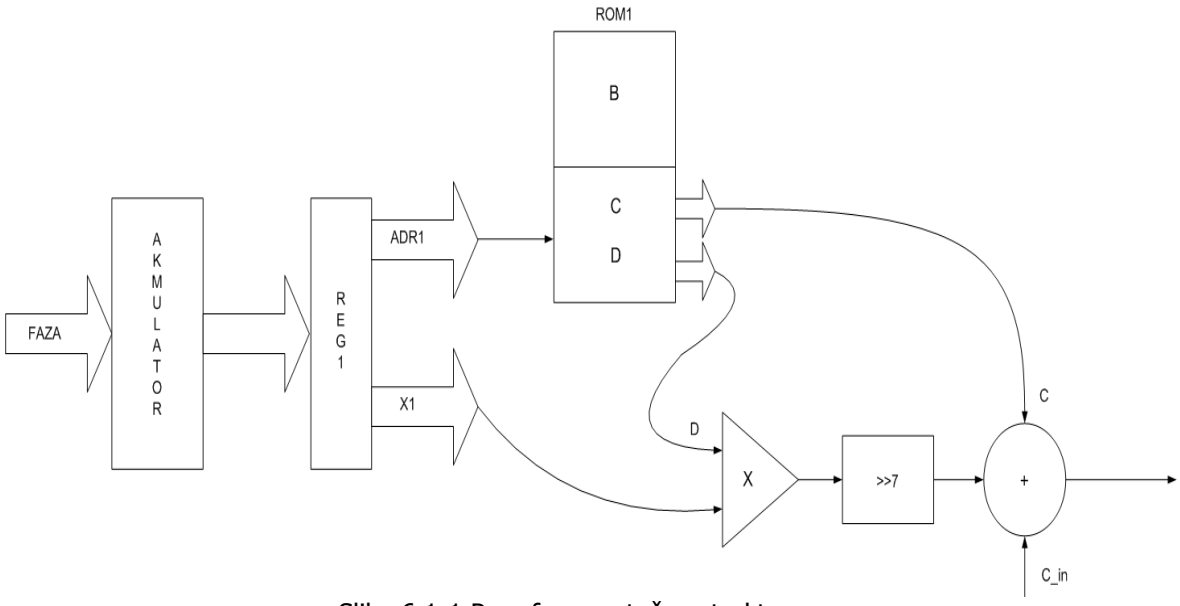


Slika 6.-1 Blok shema modela

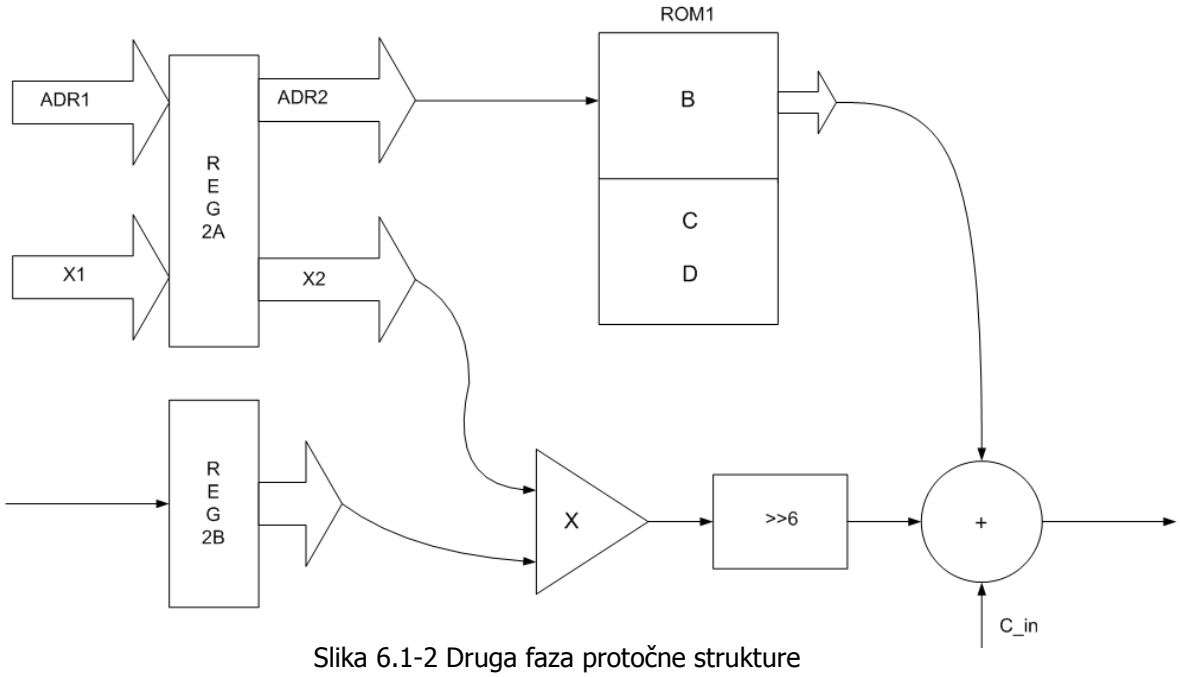
Slika 6.-1 je jednaka realiziranom modelu u VHDL-u. Ova shema je ekvivalentna kodu u cjelini 7.7. Na slici nisu navedeni svi nazivi signala i širine kao u cjelini 7.7., ali je na slici jasno označeno koji signal gdje ide.

6.1. Protočna struktura (eng. *pipeline*)

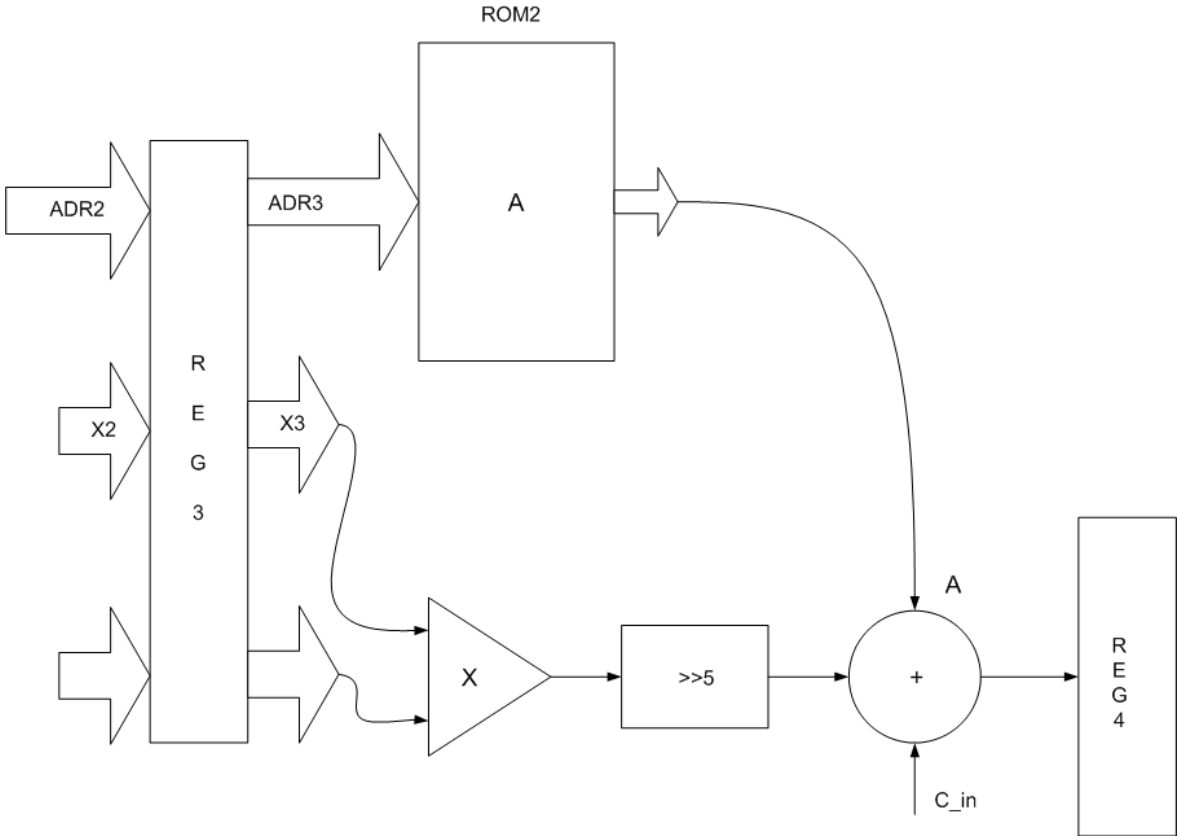
Sljedeće slike predstavljaju protočnu strukturu izvedenog modela. U svakom periodu signala takta izvedu se sve operacije prikazane slikama. Npr. za sliku 6.1.-1. Fazni akumulator prebacuje vrijednost u registar koji adresira koeficijente D i C, množi D sa x1 posmiče u desno i zbraja sa koeficijentom C. Na slijedeće dvije slike se radi gotovo isti princip. Važno je da se „stara" vrijednost faznog akumulatora u svakom periodu signala takta prenese u sljedeću fazu da ne izgubimo adresu i x. Nakon što se protočna struktura napuni u svakom taktu dobiva se novi uzorak sinusa.



Slika 6.1-1 Prva faza protočne strukture



Slika 6.1-2 Druga faza protočne strukture



Slika 6.1-3 Treća faza protočne strukture

7. Izvedba modela u VHDL-u

U ovoj cjelini ću opisati detaljniju izvedbu u VHDL-u, tj. opisati ću koje su postavke značajne, s obzirom da sam koristio komponente koje sam kasnije spojio u glavnom programu. Najprije ću opisivati komponente pojedinačno, te ću konačno opisati glavni program u kojemu su korištene prethodno navedene komponente. Osnovni oblik datoteke koja sadrži VHDL model je:

- navođenje korištenih paketa
- deklaracija entiteta
- arhitektura entiteta

U svaku komponentu su uključeni sljedeći paketi:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

s time da je paket

```
library UNISIM;
use UNISIM.VComponents.all;
```

uključen kada koristim neku od primarnih komponenti Xilinixa Spartan-3 sklopa, u mojem slučaju blok RAM i množila. U modelu sam koristio jezične predloške (eng. *Language Templates*). Također svaka VHDL datoteka ima i svoje ispitno okruženje za VHDL eng. *test bench VHDL*, a glavni program ima dva ispitna okruženja, za referentni model i za implementacijski model.

ddss Project Status (01/14/2009 - 02:31:16)			
Project File:	ddss.ise	**Current State:**	Placed and Routed
Module Name:	ddss	**• Errors:**	No Errors
Target Device:	xc3s400-4pq208	**• Warnings:**	22 Warnings
Product Version:	ISE 10.1 - WebPACK	**• Routing Results:**	All Signals Completely Routed
Design Goal:	Balanced	**• Timing Constraints:**	All Constraints Met
Design Strategy:	Xilinx Default (unlocked)	**• Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	134	7,168	1%	
Number of 4 input LUTs	91	7,168	1%	
Logic Distribution				
Number of occupied Slices	83	3,584	2%	
Number of Slices containing only related logic	83	83	100%	
Number of Slices containing unrelated logic	0	83	0%	
Total Number of 4 input LUTs	91	7,168	1%	
Number used as logic	78			
Number used as Shift registers	13			
Number of bonded IOBs	49	141	34%	
Number of RAMB16s	2	16	12%	
Number of MULT18X18s	3	16	18%	
Number of BUFGMUXs	1	8	12%	

Slika 7.-1 Broj potrošeni resursa Spartan-3 sklopa

7.1. Fazni akumulator

Deklaracija entiteta:

```
entity Acc is
      Port ( acc_in  : in  STD_LOGIC_VECTOR (25 downto 0);
             clk     : in  STD_LOGIC;
             acc_out : out  STD_LOGIC_VECTOR (25 downto 0));
end Acc;
```

U arhitekturi sam definirao proces (što komponenta radi):

```
process (clk)
      begin
            if clk='1' and clk'event then
                  temp <= temp + acc_in;
            end if;
end process;
```

Komponenta reagira na svaku promjenu takta `clk` i to kada prelazi iz logičke 0 u logičku 1, te se zatim zbraja prijašnja vrijednost s ulazom akumulatora `acc_in`. Morao sam definirati i jedan pomoćni signal `temp`

```
signal temp : STD_LOGIC_VECTOR (25 downto 0) := (others => '0');
```

koji se nakon kraja procesa proslijedi na izlaz akumulatora `acc_out`.

```
acc_out <= temp;
```

7.2. Generički registri

Deklaracija entiteta:

```
entity RegN is
      generic (N : positive);
      Port ( a_in : in  STD_LOGIC_VECTOR (N-1 downto 0);
             clk  : in  STD_LOGIC;
             b_out : out  STD_LOGIC_VECTOR (N-1 downto 0));
end RegN;
```

Generički znači da se ovaj registar može koristiti za bilo koju potrebnu širinu registra koja se mora zadati, `N` je cijeli pozitivni broj. U mojem modelu se koristi pet registara.

U arhitekturi je definiran proces:

```
process (clk) is
      begin
            if(rising_edge(clk)) then
                  b_out <= a_in;
            end if;
end process;
```

Na svaki rastući brid podatak sa ulaza `a_in` registra prebaci se na izlaz `b_out` registra

7.3. Blok RAM kao ROM

Kao što je već rečeno koriste se dvije vrste memorije, dvo-portna `RAMB16_S18_S18` i jedno-portna `RAMB16_S36`.

Deklaracija entiteta za dvo-portnu memoriju:

```
entity ROM1 is
```

```
Port ( adr_a      : in  STD_LOGIC_VECTOR (9 downto 0);
       adr_b      : in  STD_LOGIC_VECTOR (9 downto 0);
       clka       : in  STD_LOGIC;
       clkb       : in  STD_LOGIC;
       da_out     : out  STD_LOGIC_VECTOR (17 downto 0);
       db_out     : out  STD_LOGIC_VECTOR (17 downto 0));

end ROM1;
```

Deklaracija entiteta za jedno-portnu memoriju:

```
entity ROM2 is
Port ( adr : in  STD_LOGIC_VECTOR (8 downto 0);
       clk : in  STD_LOGIC;
       d_out : out  STD_LOGIC_VECTOR (31 downto 0));

end ROM2;
```

Kao što se može vidjeti dvo-portna memorija ima dvostruko više priključaka i može se primijetiti da ne postoje priključci za unos podataka zato što se memorija koristi kao ROM memorija.

Obje memorije su uzete iz predloška koji se samo kopira u arhitekturu pripadajućeg entiteta i podesi po želji, s obzirom da u predlošku ne postoji mogućnost podešavanja na koji brid takta `clk` da se pročita podatak. Definirao sam dodatne signale `NotClka` i `NotClkb`, tj. dodatni signal `NotClk` za jedno-portnu memoriju, koji su jednaki invertiranom taktu `clk`.

```
NotClka <= clka;
NotClkb <= clkb;
```

Konfiguracija dvo-portne memorije:

```
INIT_A => X"00000",
INIT_B => X"00000",
SRVAL_A => X"00000",
SRVAL_B => X"00000",
WRITE_MODE_A => "NO_CHANGE",
WRITE_MODE_B => "NO_CHANGE",
SIM_COLLISION_CHECK => "ALL",
```

U odjeljku 4.1. je objašnjeno značenje pojedinih atributa, a za jedno-portnu memoriju je jednako tako definirano uz eventualno veće širine podataka i ne postoji `SIM_COLLISION_CHECK`.

Konfiguracija priključaka dvo-portne memorije:

```
DOA => da_out (15 downto 0),
DOB => db_out (15 downto 0),
```

```
DOPA => da_out (17 downto 16),
DOPB => db_out (17 downto 16),
ADDRA => adr_a,
ADDRB => adr_b,
CLKA => NotClka,
CLKB => NotClkb,
DIA => X"0000",
DIB => X"0000",
DIPA => B"00",
DIPB => B"00",
ENA => '1',
ENB => '1',
SSRA => '0',
SSRB => '0',
WEA => '0',
WEB => '0'
```

U odjeljku 5.1. je objašnjeno značenje pojedinih priključaka, a razlika između jedno-portne memorije je što ona ima dvostruko manje priključaka. ENA i ENB su uvijek u logičkoj 1 tako da je blok RAM uvijek omogućen, a SSRA, SSRB, WEA i WEB su uvijek u logičkoj 0 tako da se izlazi nikada ne resetiraju na SRVAL vrijednost i da je zabranjeno pisanje u RAM, tj. RAM se koristi kao ROM. S obzirom da se koristi ROM, podaci moraju biti zapisani u njega, a ROM mora biti inicijaliziran. Inicijalizacijsku datoteku u tekst formatu radi Matlab skripta imena inicijalizacija, te se taj sadržaj samo kopira na zadano mjesto. Izgled jednog inicijalizacijskog reda podataka i jednog inicijalizacijskog reda pariteta je sljedeći:

```
INIT_00 => X"772D7A9D7DD380CE8. . .C48F19902F910891A39200921F",
INITP_00 => X"AAAAAAAAAAABFFF. . .FFFFFC000000000555555555555",
```

U jednom redu se nalaze 64 heksadecimalne znamenke, a prva adresa počinje od desne strane, pa ako je širina podataka 18 bita (16 bita podaci + 2 bita pariteta) tada je u ovom slučaju podatak sa adresom 0 1921F, a podatak sa adresom 15 2772D.

7.4. Množila

Koriste se tri množila MULT18X18 koja se razlikuju jedino u širini podataka, kao što je objašnjeno u cjelini 2. Stoga je ovdje prikazan kod samo od prvog množila.

Deklaracija entiteta:

```
entity Mull1 is
      Port ( m1_in : in  STD_LOGIC_VECTOR (3 downto 0);
             m2_in : in  STD_LOGIC_VECTOR (4 downto 0);
             p_out : out  STD_LOGIC_VECTOR (5 downto 0));
end Mull1;
```

Da bi množilo ispravno množilo, vodeći bit, predznak treba proširiti do vrha množila ili podatak pomaknuti do vrha. Stoga se moraju definirati tri pomoćna signala:

```
signal TEMP1 : STD_LOGIC_VECTOR (17 downto 0) := O"000000";
signal TEMP1 : STD_LOGIC_VECTOR (17 downto 0) := O"000000";
signal TEMP3 : STD_LOGIC_VECTOR (35 downto 0);
```

Pomoćni signali TEMP1 i TEMP2 su početno postavljeni u nulu tako da kada se podatak postavi na vrh, da niži bitovi od podatka budu u nuli.

```
TEMP1 (16 downto 13) <= m1_in;
TEMP2 (17 downto 13) <= m2_in;
```

Konfiguracija priključaka množila:

```
P => TEMP3,
A => TEMP1,
B => TEMP2
```

I na kraju se podatak samo proslijedi na izlaz množila:

```
p_out <= TEMP3 (34 downto 29);
```

7.5. Posmak u desno

Posmak u desno nije „pravi" posmak u desno već samo prespajanje signala kao što je rečeno u cjelini 2. U modelu se koriste tri posmaka koja se jedino razlikuju u broju posmaka u desno, stoga je ovdje prikazan kod samo od prve komponente za posmak u desno.

Deklaracija entiteta:

```
entity ShfR1 is
      Port ( shf_in  : in  STD_LOGIC_VECTOR (4 downto 0);
             shf_out : out  STD_LOGIC_VECTOR (11 downto 0));
end ShfR1;
```

Arhitektura, definiranje prespajanja:

```
shf_out(11) <= shf_in(4);
shf_out(10) <= shf_in(4);
shf_out(9) <= shf_in(4);
shf_out(8) <= shf_in(4);
shf_out(7) <= shf_in(4);
shf_out(6) <= shf_in(4);
shf_out(5) <= shf_in(4);
shf_out(4 downto 0) <= shf_in(4 downto 0);
```

Jednostavno se najviši bit ulaznog podatka proširi do vrha.

7.6. Generička zbrajala

Zbrajala su izvedena generički poput registara. Znači da ulazni i izlazni podaci mogu biti proizvoljne širine `N` (`N` je cijeli pozitivni broj) koja mora biti zadana.

Deklaracija entiteta:

```
entity AddN is
      generic (N : positive);
Port ( o1_in : in  STD_LOGIC_VECTOR (N-1 downto 0);
       o2_in : in  STD_LOGIC_VECTOR (N-1 downto 0);
       c_in : in  STD_LOGIC;
       s_out : out  STD_LOGIC_VECTOR (N-1 downto 0));
end AddN;
```

U modelu postoje tri zbrajala koja se razlikuju po širini podataka što je opisano u cjelini 2.

Arhitektura zbrajala je vrlo jednostavna:

```
s_out <= o1_in + o2_in + c_in;
```

7.7. Izvedba modela

Deklaracija entiteta:

```
entity ddss is
      Port ( clk : in  STD_LOGIC;
             phase : in  STD_LOGIC_VECTOR (25 downto 0);
             sin_out : out  STD_LOGIC_VECTOR (22 downto 0));
      end ddss;
```

`clk` je takt, `phase` je ulazna faza, a `sin_out` su izlazni uzorci sinusa.

U arhitekturi trebaju biti definirane sve ove gore navedene komponente, npr. za fazni akumulator:

```
component Acc is
      port (acc_in  : in STD_LOGIC_VECTOR (25 downto 0);
            clk     : in STD_LOGIC;
            acc_out : out STD_LOGIC_VECTOR (25 downto 0));
end component Acc;
```

Sve ostale komponente su jednako definirane.

Također u arhitekturi su definirani potrebni signali:

```
signal NC1, NC2         :     STD_LOGIC;
signal NC3              :     STD_LOGIC_VECTOR (8 downto 0);
signal reg1_in          :     STD_LOGIC_VECTOR (24 downto 0);
signal dx1, dx2, dx3          :     STD_LOGIC_VECTOR (16 downto 0);
signal addr1, addr2, addr3    :     STD_LOGIC_VECTOR (7 downto 0);
signal D                      :     STD_LOGIC_VECTOR (4 downto 0);
signal C, shf1, reg2_in, reg2_out : STD_LOGIC_VECTOR (11 downto 0);
signal B, shf2, reg3_in, reg3_out : STD_LOGIC_VECTOR (17 downto 0);
signal A, shf3, reg4_in       :     STD_LOGIC_VECTOR (22 downto 0);
signal P1                     :     STD_LOGIC_VECTOR (5 downto 0);
signal P2                     :     STD_LOGIC_VECTOR (12 downto 0);
signal P3                     :     STD_LOGIC_VECTOR (18 downto 0);
```

Signali `NC1`, `NC2` i `NC3` se spajaju na one priključke komponenti koje se ne koriste, npr. najviši bit sa izlaza faznog akumulatora. Signali `reg1_in`, `reg2_in`, `reg3_in` i `reg4_in` služe za spajanje izlaza neke komponente na ulaz registara, npr. `reg2_in` ide iz zbrajala u sljedeći registar. `reg2_out` i `reg3_out` su izlazi iz registara u drugo i treće množilo. Sa signalima `A`, `B`, `C` i `D` se prenose, tj. spajaju koeficijenti na prvo množilo, koeficijent `D` i na zbrajala koeficijente `C`, `B` i `A`. Signali `dx1`, `dx2` i `dx3` služe za prijenos x-a koji označava gdje se nalazim između dva uzorka (cjelina 1. i cjelina 2.), a signali `addr1`, `addr2` i `addr3` služe za prijenos adresa da se ne izgubi stara adresa, a zajedno su ti signali jednaki sadržaju izlaza iz faznog akumulatora bez predznaka. Signali `P1`, `P2` i `P3` služe za spajanje izlaza iz množila na komponentu za posmak bez najnižeg bita koji je prijenos (eng. *carry in*) i koji je priključen na zbrajalo, signali `shf1`, `shf2` i `shf3` su izlazi iz komponente za posmak koji se priključe na ulaz zbrajala.

Lista spajanja komponenata u arhitekturi:

```
Accumulator : component Acc port map (
            acc_in => phase, clk => clk,
            acc_out (25) => NC1,
            acc_out (24 downto 0) => reg1_in);
Register1 : component RegN generic map(N => 25) port map (
            a_in => reg1_in, clk => clk,
            b_out (24 downto 17) => addr1,
            b_out (16 downto 0) => dx1);
Register2_A : component RegN generic map (N => 25) port map (
            a_in (24 downto 17) => addr1,
            a_in (16 downto 0) => dx1, clk => clk,
            b_out (24 downto 17) => addr2,
            b_out (16 downto 0) => dx2);
Memory1 : component ROM1 port map (
            adr_a (9 downto 8) => "00",
            adr_a (7 downto 0) => addr2,
            adr_b (9 downto 8) => "01",
            adr_b (7 downto 0) => addr1,
            clka => clk, clkb => clk,
            da_out => B, db_out(17) => NC2,
            db_out (16 downto 12) => D,
            db_out (11 downto 0) =>  C);
Multipler1 : component Mull1 port map (
            m1_in => dx1(16 downto 13),
            m2_in => D, p_out => P1);
Shifter1 : component ShfR1 port map (
            shf_in => P1 (5 downto 1), shf_out => shf1);
Adder1 : component AddN generic map (N => 12) port map (
            o1_in => C, o2_in => shf1,
            c_in => P1(0), s_out => reg2_in);
Register2_B : component RegN generic map (N => 12) port map (
            a_in => reg2_in, clk => clk, b_out => reg2_out);
Multipler2 : component Mull2 port map (
            m1_in => dx2 (16 downto 6),
            m2_in => reg2_out, p_out => P2);
Shifter2 : component ShfR2 port map (
            shf_in => P2 (12 downto 1), shf_out => shf2);
Adder2 : component AddN generic map (N => 18) port map (
            o1_in => B, o2_in => shf2,
            c_in => P2(0), s_out => reg3_in);
Register3 : component RegN generic map (N => 43) port map (
            a_in (42 downto 35) => addr2,
            a_in (34 downto 18) => dx2,
            a_in (17 downto 0) => reg3_in, clk => clk,
            b_out (42 downto 35) => addr3,
            b_out (34 downto 18) => dx3,
            b_out (17 downto 0) => reg3_out);
Memory2 : component ROM2 port map (
            adr(8) => '0', adr (7 downto 0) => addr3,
            clk => clk, d_out (31 downto 23) => NC3,
            d_out (22 downto 0) => A);
Multipler3 : component Mull3 port map (
            m1_in => dx3, m2_in => reg3_out, p_out => P3);
```

```
Shifter3 : component ShfR3 port map (
            shf_in => P3 (18 downto 1), shf_out => shf3);
Adder3 : component AddN generic map (N => 23) port map (
            o1_in => A, o2_in => shf3,
            c_in => P3(0), s_out => reg4_in);
Register4 : component RegN generic map(N => 23) port map (
            a_in => reg4_in, clk => clk, b_out => sin_out);
```

Blok shema u cjelini 6. prikazuje kako su ove komponente spojene jedna s drugom. Može se primijetiti kako je u prvoj memoriji `adr_a(9 downto 8) =>“00“`, a `adr_b(9 downto 8) => “01“`, zato što se s prvog porta čitaju adrese od 0 do 255, a s drugog porta adrese od 256 do 511. Jednako tako je i u drugoj memoriji `adr(8)=>'0'`, zato što se koriste samo adrese od 0 do 255. Također se može primijetiti način spajanja komponente za posmak koji je priključen, npr. za prvi posmak `shf_in => P1 (5 downto 1)`, najviših pet se posmiče, a zadnji bit se šalje na zbrajalo `c_in => P1(0)`.

8. Simulacija modela

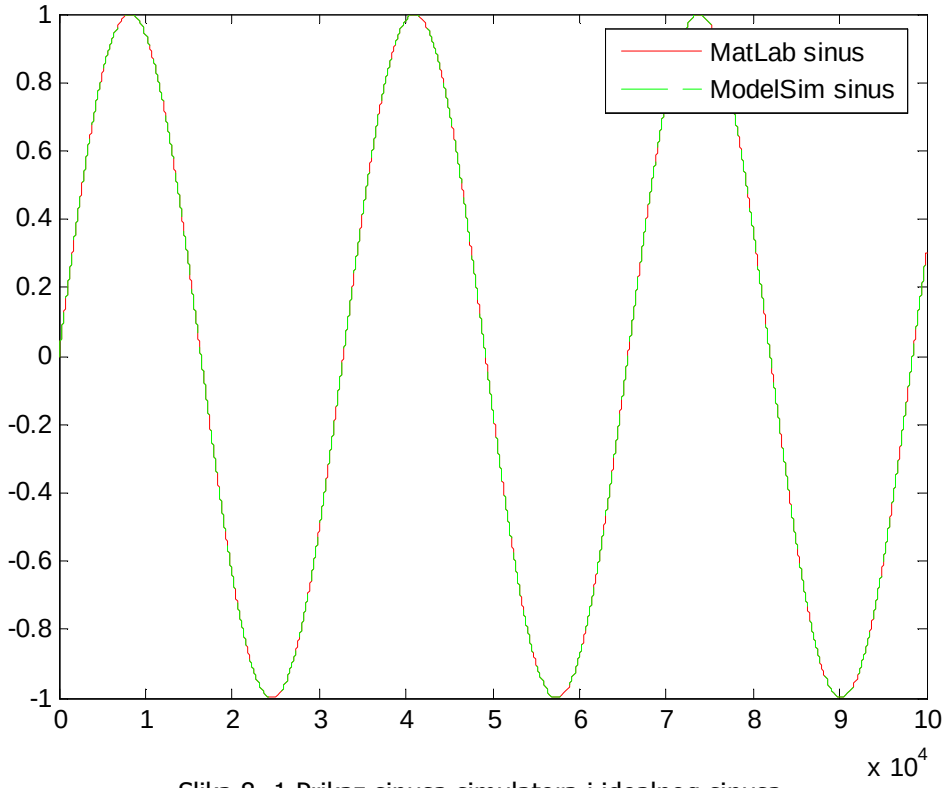
Slika 8.-1 prikazuje idealni sinus dobiven u Matlabu i sinus dobiven simulacijom u Modelsimu. Može se primijetiti kako se sinusi poklapaju, tj. greška nije vidljiva okom iako postoji, što je prikazano slikom 8.-2. Pogreška je izražena kao razlika idealnog i modeliranog sinusa u LSB-u (eng. *least significant bit)*, što se dobije množenjem razlike sa 22^{22} (22 zato što je to širina izlaza).

Simulacija u Modelsimu se pokreće `do ddss_tb_beh.do` za eng. *Behavioral* model, a za eng. *Post-Place & Route* model naredbom `do ddss_tb_beh.do`. Slika 8.-3 prikazuju simulaciju u Modelsimu za *Behavioral* model, a slike 8.-4 za simulaciju *Post-Place & Route* modela.

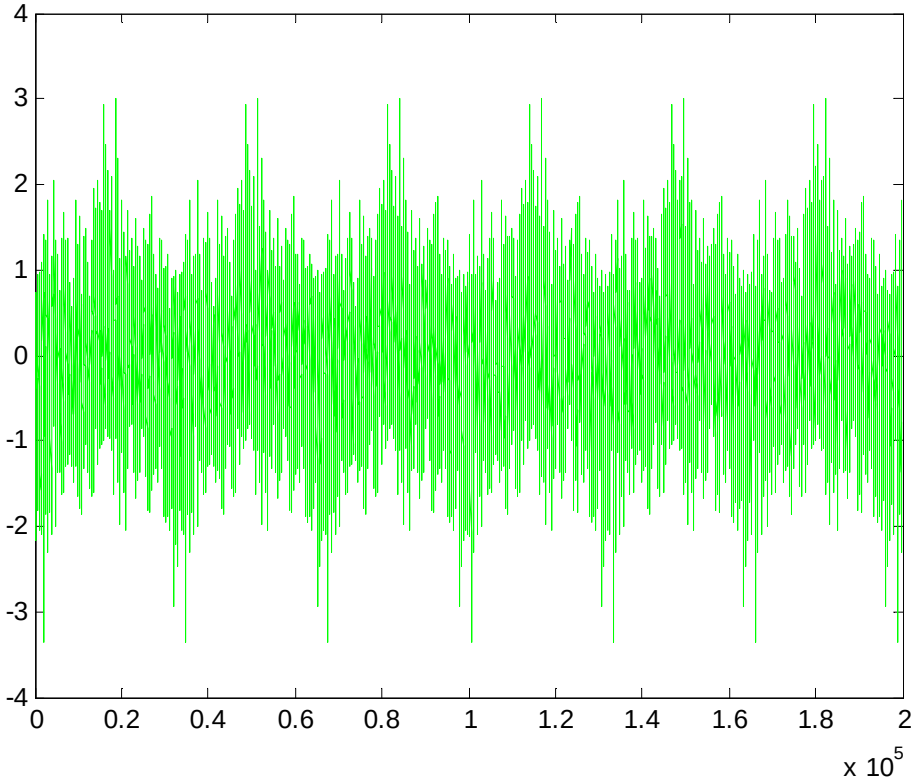
Na slici 8.-3 crvenim krugom je označen dio gdje se vrlo dobro vidi protočna struktura. U dx3 ide stari podatak iz dx2, a u dx2 ide stari podatak iz dx1 i u dx1 ide novi podatak iz akumulatora. Kada se protočna struktura napuni tada se u svakom periodu signala takta dobiva na izlazu novi signal.

Na slici 8.-4 crvenim krugovima je označen do sada ne razriješen problem. U simulaciji *Post-Place & Route* modela treba više od tri perioda signala takta da bi se protočna struktura popunila i počela davati točne uzorke na izlazu.

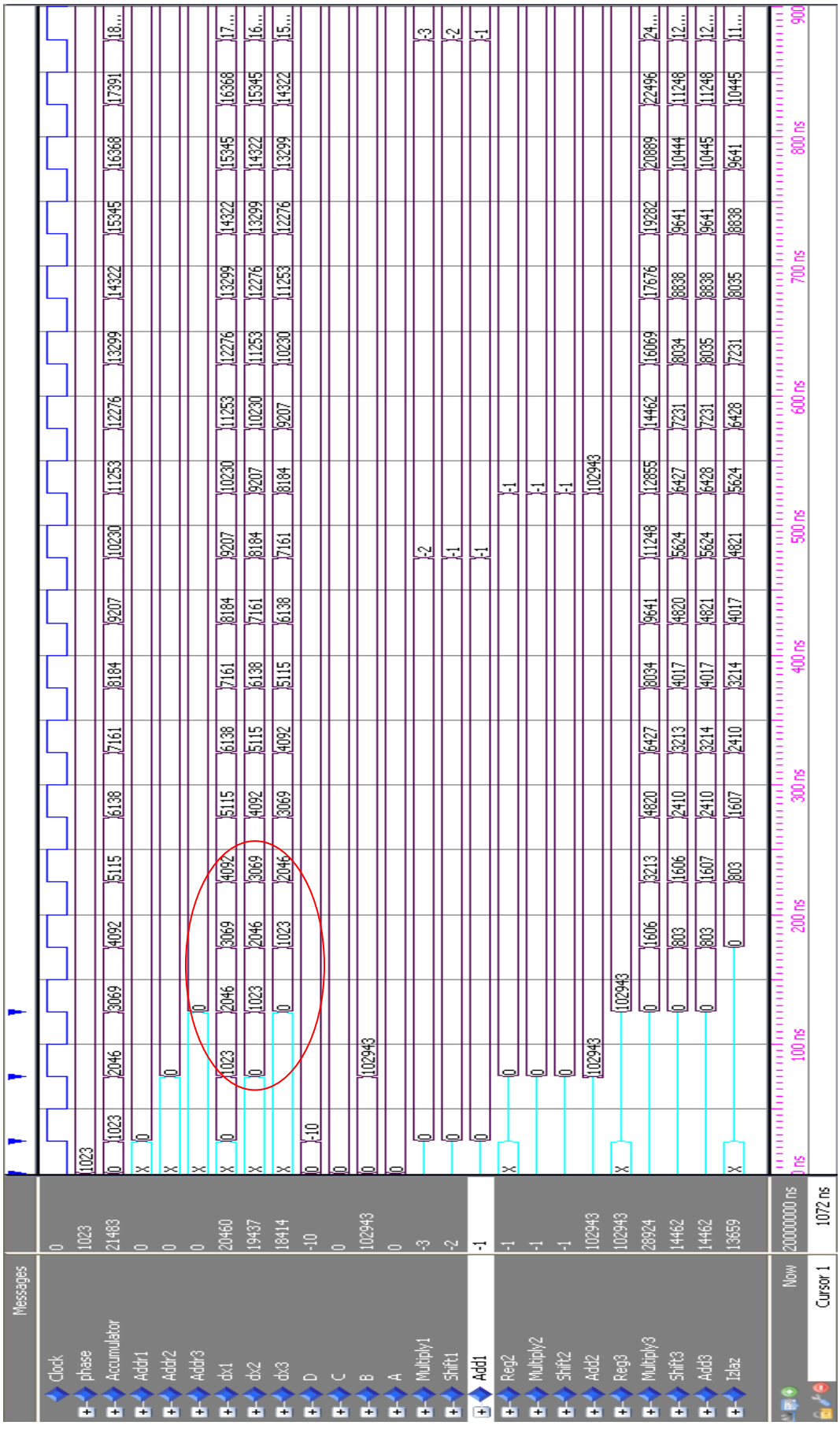
Maksimalna brzina modela je 77,5 MHz (period signala takta je 12,897 ns). Ukoliko bi se brzina modela još povećala, tako da je period signala takta manji od 12,897 ns nastala bi greška na izlazu modela. Što zapravo nije greška, već model jednostavno ne bi znao što treba doći na izlazu, kao što je prikazano slikom 8.-5 (označeno crvenim krugovima).



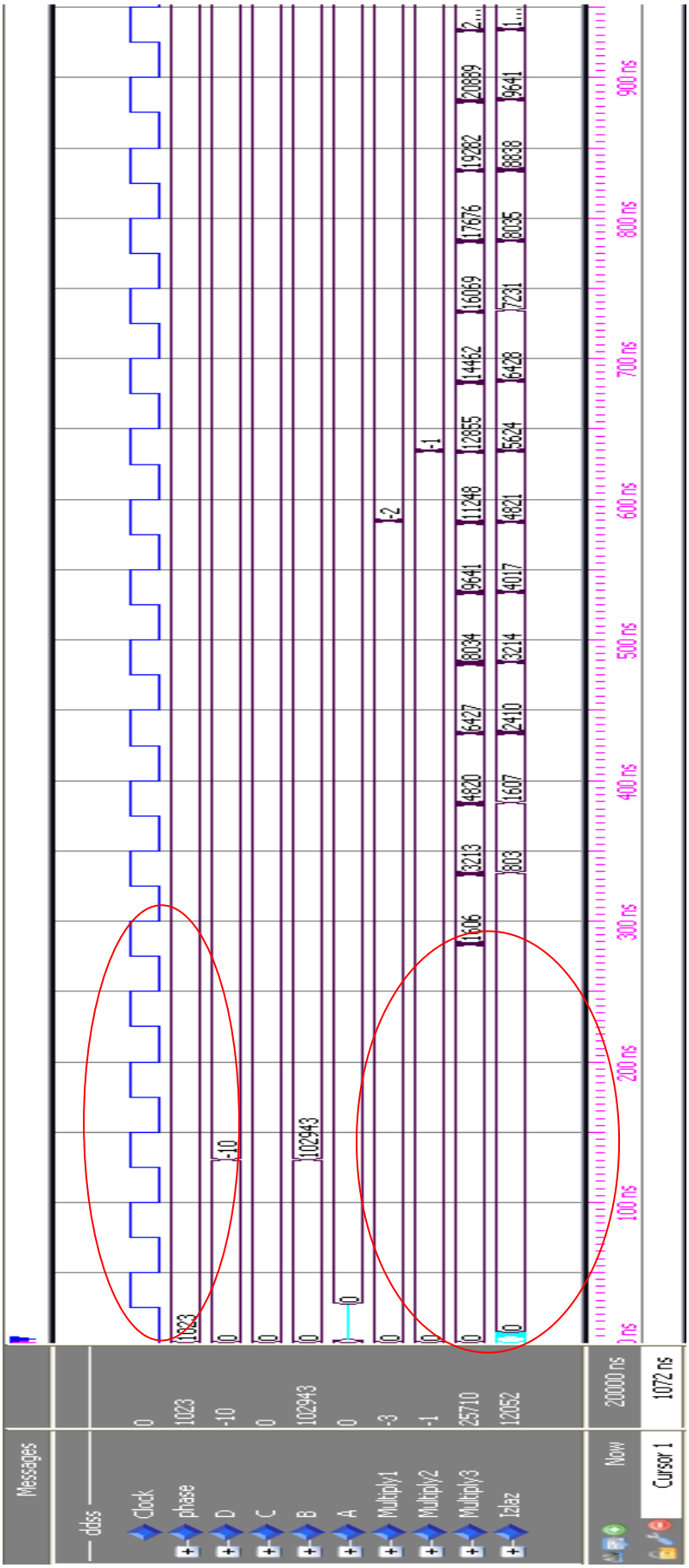
Slika 8.-1 Prikaz sinusa simulatora i idealnog sinusa



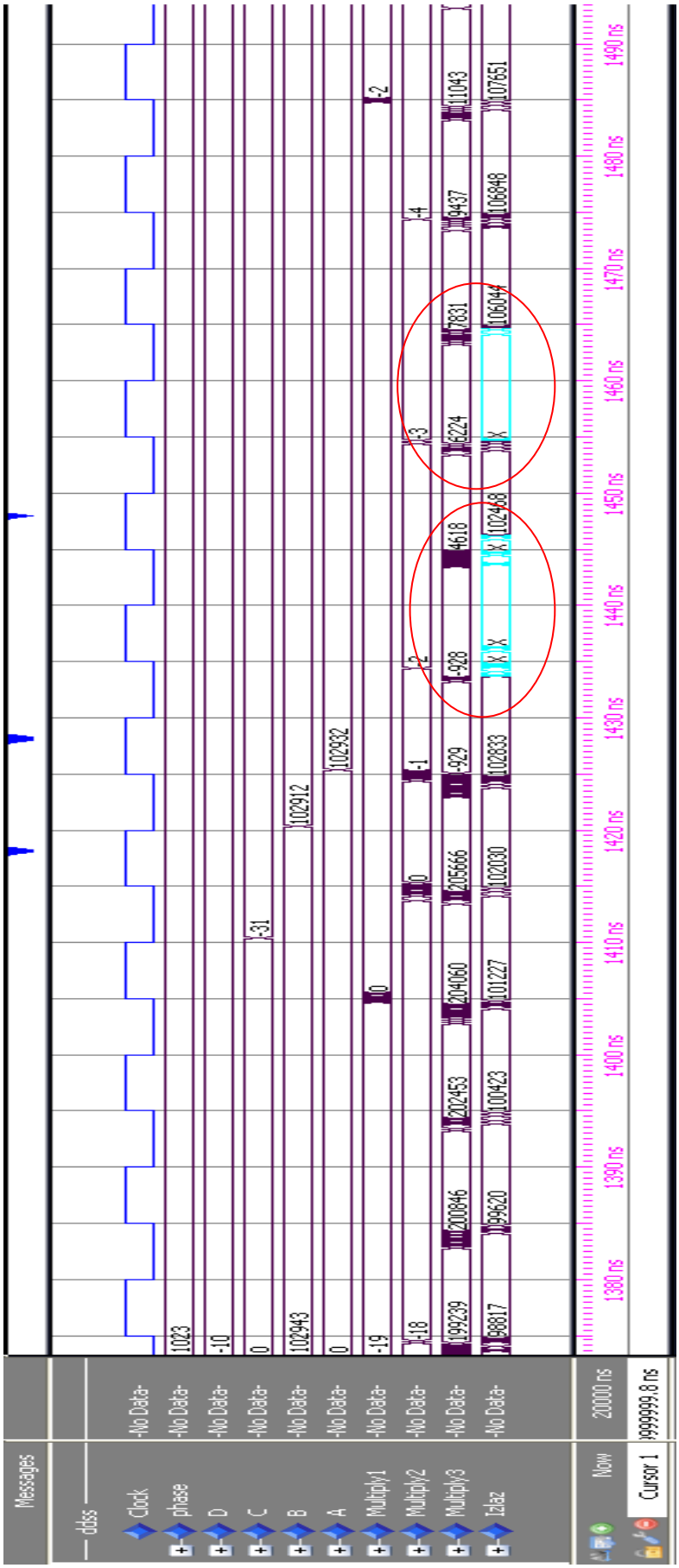
Slika 8.-2 Pogreška izražena u LSB



Slika 8.-3 Simulacija u Modelsimu za *Behavioral* model



Slika 8.-4 Simulacija *Post-Place & Route* modela



Slika 8.-5 Simulacija *Post-Place & Route* modela sa malim periodom takta

9. Moguća poboljšanje modela

S obzirom da za sada još nije otkriven uzrok problema koji je prikazan na slici 8.-4 i kritični odsječak, koja komponenta je najsporija, samo ću napomenuti moguća poboljšanja.

U model bi bilo dobro dodati signal za globali reset koji bi cijeli model postavio u početno stanje, a koji bi mogao pomoći u razrješavanju već spomenutog problema.

Moguća je uporaba i tri blok RAM-a ukoliko bi se pokazalo da tako implementirani model ima bolja svojstva.

Moguće je dodavanje još jednog koraka u protočnoj strukturi, ako jedna od operacija koje se izvršavaju u jednom koraku protočne strukture pravi probleme, tada tu operaciju stavimo u novi korak i time ubrzamo model.

Moguće je staviti sve koeficijente u jedan dvo-portni blok RAM veličine 512x36 bita, gdje bi sa jednog porta čitali koeficijente A, a sa drugoga koeficijente B, C i D. Poravnavanje u vremenu, zbog protočne strukture, bi se radilo pomoću registara.

10. Zaključak

Tokom rada na ovome modelu pokazalo se da isti ima veliku preciznost, gotovo 22 bita; što se može vidjeti po slici 7.-2 gdje je prikazana pogreška modela; dok se na slici 7.-1 može vidjeti, tj. ne može vidjeti razlika između simulirane sinusoide i idealne sinusoide.

Behavioral model u Modelsimu radi vrlo dobro, dok Post-Place & Route model ima „neku" pogrešku, tj. trenutno se ne zna koji je uzrok te duge inicijalizacije. Izvještaj nakon Post-Place & Route simulacije kaže da sklop može raditi na brzini od 77,5 MHz, te i radi na istoj, ali uz gore navedenu pogrešku. Može se reći da je to jedna dosta dobra brzina rada.

Dakako moguća su i poboljšanja modela, ali nakon što se pogreška analizira, da bi se moglo vidjeti koji dijelovi protočne strukture su najsporiji.

Izvedba u VHDL-u izvedena je pomoću komponenata koje se kasnije instanciraju iz glavnog programa, što pojednostavljuje pisanje samoga koda, pregledniji je i neupućena osoba vrlo lako se može snaći u njemu.

Kako se danas sve više koriste digitalni signali u pretvornicima, važno je razvijati što preciznije i kvalitetnije direktne digitalne pretvarače, kako bi bio moguć brži i bolji razvoj digitalnih sklopova koji koriste AD pretvarače.

Marko Brezović

Literatura

[1] Xilinx, DS099, 25.lipnja 2008.
Spartan-3 FPGA Family Data Sheet,
http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf
(29.11.2008.)

[2] Xilinx, ug331 ,25. lipnja 2008.
Spartan-3 Generation FPGA User Guide,
http://www.xilinx.com/support/documentation/user_guides/ug331.pdf
(29.11.2008.)

[3] Prof. dr. sc. Davor Petrinovic, Matlab skripta, 20. rujan 2008.
DDS_sin_A0_simp.m

[4] Robert Gotal, PSAC, 3. prosinca 2008.
Amplitudni konverter faze u sinus

Dodatak A

Naslov:

Direktna digitalna sinteza visoke točnosti korištenjem kubične B-spline interpolacije

Ključne riječi:

Spline, B-spline, Hornerovo pravilo, Xilinx Spartan-3, blok RAM, množila, VHDL, Matlab, protočna struktura

Sažetak:

Cilj rada je iz faze dobiti sinusoidu te je implementirati u VHDL-u. Zadana faza je korak faznog akumulatora pomoću kojeg se određuje gdje sam, na kojem uzorku i gdje sam između dva uzorka. Svaki taj uzorak je prikazan polinomom trećeg stupnja čijom evaluacijom se dobiva uzorak sinusoide.
Arhitektura za razvoj modela je Spartan-3 te je sve prilagođeno za rad na njemu. Najvažnije komponente Spartan-3 sklopa u modelu, su blok RAM (koristi se kao ROM) i množila.
Dobivenu sinusoidu provjeriti sa dobivenim Matlab kodom.

Dodatak B

Title:

Direct digital sintesis of high accuracy, using cubical interpolation of B-spline.

Key words:

Spline, B-spline, Hoorners rule, Xilinx Spartan-3, Block RAM, multipliers, VHDL, Matlab, pipeline

Summary:

The main purpose of this paper is to get a sinusoid and implement it into VHDL. Default phase is a step of a phase accumulator, by which can be easily determined where am I, on which sample, and where between two samples. Each sample is presented by the third degree polynomial, whom evaluation is given a sample of a sinusoid.
The given architecture is used for a Spartan-3 model expantion, so all is set to work with it.
The most important components of a Spartan-3 model circuit, are block RAM (used as ROM) and multipliers.
The given sinusoid should be tested with a gained Matlab code.