

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1196

**DIREKTNA DIGITALNA SINTEZA IZNIMNO
VISOKE TOČNOSTI**

Antonela Šipić

Zagreb, listopad 2009.

*„Sve izgleda iz daleka drukčije...“
Balašević*

Previše je ljudi koji su ostavili trag, svojim riječima, djelima i molitvama učinili život ljepšim, a mene mudrijom da ga isto tako lijepog i doživim!

Hvala svima koji su se ovdje prepoznali i onim skromnijima koji nisu, a trebali bi ☺

Zahvaljujem svom mentoru prof. Davoru Petrinoviću na nesebičnim naporima da podijeli sa mnom bar dio svog znanja.

Diplomski zadatak

Umjesto ove stranice uvezuje se originalni diplomski zadatak!

Sadržaj

Diplomski zadatak.....	2
Sadržaj.....	3
1. Uvod.....	4
2. Tehnike PSAC pretvorbe za realizaciju DDFS prema J.M.P. Langlois i D. Al-Khalili.....	6
2.1 TEMELJNA ARHITEKTURA DDFS.....	6
2.2 METODE IMPLEMENTACIJE PSAC PRETVORNIKA.....	9
2.2.1 Kutna dekompozicija.....	9
2.2.2 Metode temeljene na kutnoj rotaciji.....	10
2.2.3 Amplitudna kompresija.....	11
2.2.4 Polinomijalna aproksimacija.....	13
2.2.5 Kombinacije PSAC-a i digitalno-analognog pretvornika.....	19
2.3 USPOREDBE RAZLIČITIH IZVEDBI PSAC-A.....	19
3. DDFS prema modelu B-Spline interpolacije po dijelovima glatke aproksimacije polinomom ...	21
3.1 MODEL SINUSNE FUNKCIJE B-SPLINE INTERPOLACIJOM.....	22
3.2 SVOJSTVA DDS SA PSAC-OM TEMELJENIM NA KUBIČNOJ B-SPLINE INTERPOLACIJI.....	25
3.3 KVANTIZACIJA POLINOMIJALNIH KOEFICIJENATA I IMPLEMENTACIJA LOGIKOM FIKSNOG ZAREZA.....	26
3.3.1 Korištenje kvadrantne simetrije sinusne funkcije.....	29
3.4 IMPLEMENTACIJA PSAC-a NA FPGA SKLOP.....	30
4. Realizacija <i>DDFS</i> pomoću programskog paketa Xilinx.....	32
4.1 UVOD U PROGRAMSKI PAKET XILINX.....	32
4.2 BLOK SHEMA SUSTAVA I REALIZACIJA PROTOČNE ARHITEKTURE.....	34
4.3 MODULI UNUTAR DDFS-A.....	36
4.4 SPAJANJE MODULA U <i>DDFS</i>	50
4.4.1 Određivanje kašnjenja pojedinih signala u sustavu.....	50
4.4.2 Spajanje modula i provjera ispravnosti izlaza.....	52
4.5 SINTEZA I IMPLEMENTACIJA SKLOPA.....	58
5. Analiza rada sklopa i rasprava o zadovoljenju zahtjeva postavljenih na sklop.....	61
6. Zaključak.....	65
Literatura.....	66
Dodatak.....	67
DODATAK 1 – DIO REFERENTNOG KODA ZA SIMULACIJU <i>DDFS</i> -a KOJI RASPISUJE PROTOČNU STRUKTURU PO STUPNJEVIMA (MATLAB).....	67
DODATAK 2 – IZMJENE U REFERENTNOM KODU.....	69

1. Uvod

Sinusna funkcija odavno već ima značajnu ulogu u području inženjerstva.

U mnogim literaturama se opisuju i proučavaju razni algoritmi i arhitekture čiji je zadatak efikasno izračunati dovoljno preciznu vrijednost amplitude sinusa na osnovu zadanog kuta.

PSAC (*Phase to Sine Amplitude Conversion*) je pretvorba faze u amplitudu sinusne funkcije.

Danas postoje razni algoritmi za izvedbu ove pretvorbe, ali još uvijek je područje od širokog interesa i istraživanja.

Direktna Digitalna Sinteza (DDS) je elektronička metoda za realizaciju digitalnih signala proizvoljnih valnih oblika iz zadane frekvencije.

Direktna digitalna frekvencijska sinteza (DDFS) je poznata tehnika za generiranje sinusoide i kvadratnog signala.

Dozvoljava precizan odabir frekvencije uz jednostavnu implementaciju sintetizatora i mogućnost kontinuiranog podešavanja faze.

Temeljnu arhitekturu DDFS-a dao je Tierney *et al.*¹ 1971. a sastoji se od 2 bloka:

- fazni akumulator
- PSAC

Na isti način izveden je i sklop u okviru ovog rada.

Fazni akumulator generira vrijednost ulazne fazne riječi sinkronizirano sa signalom takta, a izlaz mu predstavlja fazu u intervalu $[0, 2\pi)$.

PSAC obavlja estimaciju sinusa tog kuta.

¹ Tierney, J., Rader, C.M., and Gold, B. "A Digital Frequency Synthesizer," *IEEE Transactions on Audio and Electroacoustics* **AU-19**:1, March 1971, 48-56

Današnje primjene DDS-a su brojne i nalaze se u raznim područjima ljudske djelatnosti: digitalna komunikacija, radari, instrumentacija, primjena elektronike u modernom ratovanju, audio sustavi...

Kako su bitni zahtjevi na sklopove jednostavnost i što veća brzina, najviše se vremena i istraživanja posvećuje upravo poboljšanju performansi različitih arhitektura DDS-a u tom smjeru.

Sklop realiziran u okviru ovog diplomskog rada, uz zadani pribroj faze na ulazu i početnu fazu, na izlazu daje sinusoidu. Korišteni su programski paketi MATLAB 7.5.0 (R2007b), u kojem je dana simulacija izvedbe i ponašanja sklopa, Xilinx ISE 7.1i, u kojem je u VHDL-u napisan kod za realizaciju sustava, napravljena sinteza i implementacija na FPGA sklop i ModelSim XE III/Starter 6.0a simulator za provjeru ispravnosti ponašanja sustava.

2. Tehnike PSAC pretvorbe za realizaciju DDFS prema J.M.P. Langlois i D. Al-Khalili

Ovo poglavlje daje kratak pregled tehnika PSAC pretvorbe za realizaciju DDFS. Ukratko su dani principi realizacije DDFS, a nakon toga predstavljena razmatranja i različiti pristupi za smanjenje složenosti sustava.

Navedene su različite tehnike za realizaciju PSAC pretvorbe i dana je njihova usporedba sa stajališta složenosti sustava i preciznosti rezultata.

2.1 TEMELJNA ARHITEKTURA DDFS

Temeljnu arhitekturu Direktne Digitalne Frekvencijske sinteze predstavili su J. Tierney *et al.* 1971.

Sastoji se od faznog akumulatora i PSAC pretvornika. Pojednostavljena blok shema prikazana je slikom 1.

Sustav ima 2 ulaza: referentni takt i kontrolnu faznu riječ (FCW).

Fazni akumulator daje vrijednost nove fazne riječi u svakom periodu takta. Nakon određenog broja perioda takta, ovisno o duljini fazne riječi (broju bitova), dolazi do preljeva i periodičnog ponavljanja vrijednosti fazne riječi.

Izlaz faznog akumulatora predstavlja kut u intervalu $[0, 2\pi)$.

Frekvencijska rezolucija sintetizatora i izlazna frekvencija dane su izrazom (1) i (2).

$$\Delta f = \frac{f_0}{2^N} \quad (1)$$

$$f_{\text{out}} = \text{FCW} \cdot \Delta f \quad (2)$$

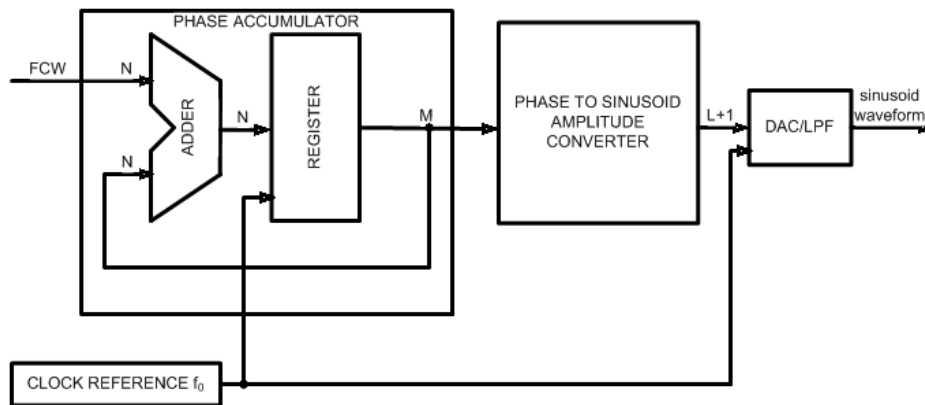
f_0 je frekvencija referentnog takta, a N rezolucija faznog akumulatora (širina fazne riječi).

Očito je da frekvencijska rezolucija sintetizatora može biti proizvoljno mala uz dovoljno velik N .

U praksi se bira kompromis između širine fazne riječi, složenosti sustava, željene frekvencije takta i kašnjenja izlaznog signala.

PSAC ima zadatak da estimira i na izlaz sustava daje vrijednost amplitude sinusne funkcije za svaki novi kut.

Najčešće se implementira digitalno, a ako postoji potreba za analognim izlazom iz sustava, izlaz PSAC-a se dovodi na digitalno-analogni pretvornik i niskopropusni filter.



Slika 1 Pojednostavljeni prikaz DDS arhitekture

Najjednostavnija izvedba PSAC-a je pomoću LUT ROM-a².

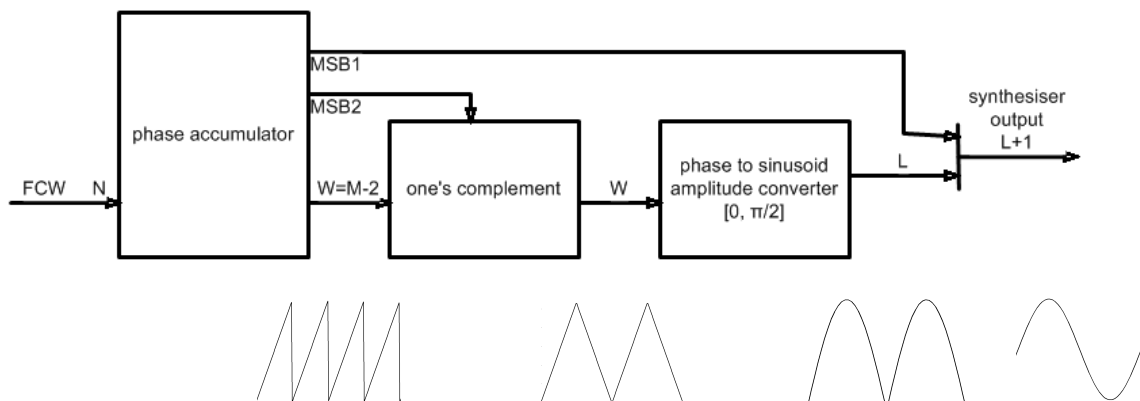
Veza između veličine ROM-a i broja bitova fazne riječi je eksponencijalna i postaje nedozvoljivo velika čak i za sustave umjerenih performansi, pa se širina ulaza u PSAC reducira na M najznačajnijih bitova, slika 1.

² ROM realiziran pomoću „look up“ tabele (LUT).

LUT je osnovna jedinica FPGA logičkih sklopova, predstavlja tablicu istinitosti željene logičke funkcije.

Na taj način se složenost PSAC-a smanjuje bez potrebe za povećanjem frekvencijske rezolucije. Negativna strana ove redukcije je pojavljivanje šuma u izlaznom spektru koji je posljedica periodičke amplitudne pogreške izlazne sinusoide (4).

Složenost PSAC-a se može učinkovito smanjiti i korištenjem svojstava simetrije sinusnog signala. Dva najznačajnija bita fazne riječi određuju u kojem se kvadrantu nalazi određeni kut. Jednostavnim transformacijama se čitava sinusoida može rekonstruirati iz izračunatih vrijednosti amplituda samo za prvi kvadrant. Blok shema koja prikazuje ovakav pristup prikazana je na slici 2.



Slika 2 DDS sa reduciranom fazom i primjenom kvadratne simetrije

Ulaz u PSAC je W -bitna frakcija x , a izlaz je sinusoida $f(x)$ (izrazi (3) i (4)). $f(x)$ je prikazana s N bitova na izlazu i pozitivna je.

$$x \in \frac{\{0,1,2,\dots,2^W-1\}}{2^W}, \quad W = M - 2 \quad (3)$$

$$f(x) = A \cdot \sin\left(\frac{\pi \cdot x}{2}\right) + \varepsilon(x), \quad 0 \leq x < 1 \quad (4)$$

$\varepsilon(x)$ je pogreška izlaza PSAC-a u odnosu na idealnu sinusoidu, predstavljena beskonačnom preciznošću.

Ova pogreška je nastala akumulacijom više pogreški različitog karaktera kao što su pogreška amplitudne kvantizacije i pogreške koje unosi algoritam.

Izlazni frekvencijski spektar ima dvije komponente: čistu sinusoidu (jednu frekvenciju) amplitude A i šum koji je posljedica amplitudne pogreške.

Omjer ovih dviju komponenta izlaznog spektra naziva se SFDR (*spurious free dynamic range*) i kritični je parametar za ocjenu performansi sustava.

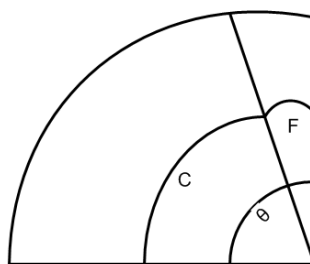
Implementacija funkcije u PSAC-u za prvi kvadrant jedinične kružnice predmet je intenzivnog istraživanja. Razlog tome je jednostavna implementacija LUT ROM-om, koji je pak nepraktičan i skup ukoliko je potreban veliki kapacitet za pohranu podataka.

2.2 METODE IMPLEMENTACIJE PSAC PRETVORNIKA

2.2.1 Kutna dekompozicija

Fazni kut se segmentira tako da susjedni segmenti kuta predstavljaju različite pomake i to od većih („grubljih“) prema manjim („finijim“), slika 3.

Svaki Kut je predstavljen kao zbroj „grubih“ i „finih“ kutova.



Slika 3 Kutna dekompozicija

Koristi se više ROM-ova za spremanje podataka kojima se identificiraju kutovi.

Podaci za izračun najgrubljih segmenata spremljeni su u jedan ROM, zatim podaci za izračun nešto finijih segmenata u drugi i tako redom do podataka za najfinije segmente (najmanje moguće kutne rezolucije) koji su spremljeni u posljednji ROM.

Uz korištenje trigonometrijskih simetrija i aproksimacija, na ovaj način se znatno reducira kapacitet potreban za pohranu podataka za izvedbu PSAC-a.

Primjer implementacije ove metode je Hutchinsonov pristup³:

$$\sin \theta = \sin(C + F) = \sin C \cdot \cos F + \sin F \cdot \cos C \cong \sin c + \sin F \times \cos C \quad (5)$$

Izraz (5) vrijedi ako je F dovoljno mali kut.

Prvi ROM sadrži vrijednosti sin C, a drugi unaprijed izračunate vrijednosti sin F x cos C.

Više je različitih izvedbi, pristupa i poboljšanja obrađeno i objavljeno tokom vremena na temelju ove ideje, ali ovdje se neće detaljno razrađivati ostali pristupi.

2.2.2 Metode temeljene na kutnoj rotaciji

Nekoliko istraživanja temeljeno je na CORDIC (**CO**ordinate **R**otation **D**igital **C**omputer) algoritmu i sličnim algoritmima temeljenim na kutnoj rotaciji, u želji da se eliminira ROM LUT pristup u implementaciji PSAC-a.

CORDIC je jednostavan i efikasan algoritam za računanje hiperbolnih i trigonometrijskih funkcija, najčešće korišten u situacijama kad na raspolaganju nema množila. Temelji se na izvođenju jednostavnih operacija kao što su zbrajanje, oduzimanje, posmak bitova i pregled tablica.

Ostvareni su mnogi uspješni dizajni s visokim SFDR omjerom i prihvatljivom složenošću.

Prednost ovakvog pristupa je izostanak potrebe za kompleksnim množenjem, dok je mana povećano kašnjenje.

³ Hutchinson, B.H. Jr.: 'Contemporary frequency synthesis techniques', in Gorski-Pcpicl, J. (Ed.): 'Frequency synthesis: techniques and applications' (IEEE Press, 1975), pp. 25-45

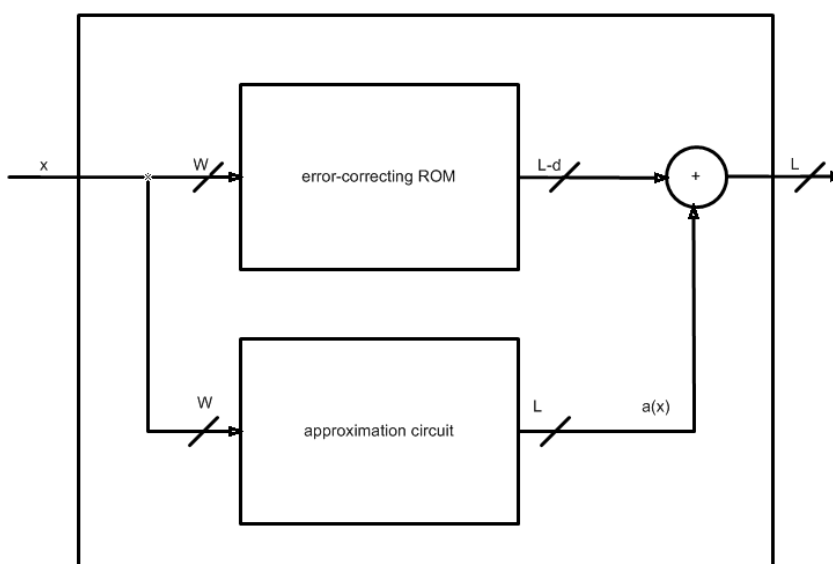
2.2.3 Amplitudna kompresija

Ova metoda temelji se na računanju grubih aproksimacija sinusne funkcije te spremanju rezidualne pogreške u ROM čiji sadržaj će kasnije biti pribrojen aproksimaciji.

Ideja je smanjiti dinamički opseg podataka u ROM-u, čime se smanjuje potreban kapacitet ROM-a za pohranu podataka.

Ovisno o točnosti aproksimacije, podaci u ROM-u su reducirani za broj bitova d , a veličina ROM-a za faktor $(L-d)/L$.

Slika 4 prikazuje arhitekturu PSAC-a temeljenu na amplitudnoj kompresiji.



Slika 4 PSAC arhitektura temeljena na amplitudnoj kompresiji

Ako je fazni kut $x \in [0,1)$, pogreška $\varepsilon(x)$ između amplitude idealnog sinusa i izlaza aproksimacijskog kruga $a(x)$ dana je izrazom (6).

$$\varepsilon(x) = A \sin\left(\frac{\pi x}{2}\right) - a(x) \quad (6)$$

Tipična vrijednost amplitude iznosi $\frac{2^L - 1}{2^L}$, na taj način je maksimalno iskorišten dinamički opseg izlaza sintetizatora.

Radi jednostavnosti, pogreška spremljena u ROM $\varepsilon(x)$ je strogo pozitivna ili negativna. Ako se u ROM sprema 1 bit, maksimalna apsolutna vrijednost $\varepsilon(x)$ mora biti < 0.5 . Za spremanje 2 bita, $|\varepsilon(x)| < 0.25$, za spremanje d bitova, $|\varepsilon(x)| < 2^{-d}$.

Različite primjene ove metode uglavnom se odnose na različito definiranje funkcije $a(x)$ iz izraza (6).

Primjeri su dani u nastavku:

*Nicholas et al.*⁴

$$a(x) = x, \quad |\varepsilon(x)| < 0.25, \quad (6.1)$$

podatak u ROM-u sadrži 2 bita

*Yamagishi et al.*⁵

$$a(x) = \begin{cases} \frac{5}{4}x, & \Rightarrow 0 \leq x < \frac{1}{2} \\ \frac{3}{4}x, & \Rightarrow \frac{1}{2} \leq x < 1 \end{cases}, \quad |\varepsilon(x)| < 0.125, \quad (6.2)$$

podatak u ROM-u sadrži 3 bita

*Sodagar i Lahiji*⁶

$$a(x) = x(2-x) \quad 0 \leq x < 2, \quad |\varepsilon(x)| < 0.0625 \quad (6.3)$$

podatak u ROM-u sadrži 4 bita

⁴ Nicholas, H.T., Samueli, H. and Kim, B.: 'The optimization of direct digital frequency synthesizer performance in the presence of finite word length effects'. *Proc. 42nd Annual Symp. on Frequency Control, 1988*, pp. 357-363

⁵ - Yamagishi, A., Ishikawa, M., Tsukahara, T., and Date, S.: 'A 2-V, 2 GHz low-power direct digital synthesizer chip set for wireless communication'. *Proc. IEEE Custom Integrated Circuits Conf., 1995*, pp. 319-322

- Yamagishi, A., Ishikawa, M., Tsukahara, T., and Date, S.: 'A 2-V, 2 GHz low-power direct digital synthesizer chip set for wireless communication'. *Proc. IEEE J. Solid State Circuits, 1998*, **33**, (2), pp. 210-217

⁶ - Sodagar, A.M., and Lahiji, G.R.: 'Parabolic approximation: a new method for phase-to-amplitude conversion in sine-output direct digital frequency synthesizers'. *Proc. Int. Symp. on Circuits and Systems, Switzerland, May 2000*, pp. 515-518

- Sodagar, A.M., and Lahiji, G.R.: 'Mapping from phase to sine-amplitude in direct digital frequency synthesizers using parabolic approximation', *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., 2000*, **47**, (12), pp. 1452-1457

Langlois i Al-Khalili⁷

$$a(x) = \left\{ \begin{array}{l} 0 + \frac{3}{2}x, \Rightarrow 0 \leq x < 2 \\ \frac{5}{32} + x, \Rightarrow \frac{5}{16} \leq x < \frac{3}{4} \\ \frac{1}{2} + \frac{x}{2}, \Rightarrow \frac{3}{4} \leq x < 1 \end{array} \right\}, \quad |\mathcal{E}(x)| < 0.0625 \quad (6.4)$$

podatak u ROM-u sadrži 4 bita

2.2.4 Polinomijalna aproksimacija

Ideja je aproksimirati dijelove sinusnog valnog oblika polinomima određenog stupnja.

Sinusnu funkciju moguće je raspisati prema izrazu (7).

$$\sin\left(\frac{\pi x}{2}\right) \cong \left\{ \begin{array}{ll} \sum_{i=0}^r c_{0i} (x - x_0)^i & x_0 \leq x < x_1 \quad (x_0 = 0) \\ \sum_{i=0}^r c_{1i} (x - x_1)^i & x_1 \leq x < x_2 \\ \vdots & \\ \sum_{i=0}^r c_{ki} (x - x_k)^i & x_k \leq x < x_{k+1} \\ \vdots & \\ \sum_{i=0}^r c_{(s-1)i} (x - x_{s-1})^i & x_{s-1} \leq x < x_s \quad (x_s = 1) \end{array} \right. \quad (7)$$

x je kut predstavljen kao frakcija iz intervala $[0,1)$

r je stupanj polinomijalne aproksimacije

s je broj po dijelovima glatkih polinomijalnih segmenata

c_{ki} su koeficijenti polinoma

x_k je donja granica k -tog segmenta

Izraz (7) iskorišten je za razne implementacije PSAC-a.

⁷ Langlois, J.M.P., and Al-Khalili, D.: 'ROM size reduction with low processing cost for direct digital synthesis'. *Proc. IEEE Pacific Rim Conf. on Communications, Computer and Signal Processing, Victoria, BC, Canada, August 2001*, pp. 287-290

Razlike izvedbi mogu se podijeliti na četiri osnovna kvalifikatora:

- stupanj r , $r \geq 1$, polinomijalne aproksimacije
- broj s , $s \geq 1$, po dijelovima glatkih polinomijalnih segmenata
- granice segmenata x_k
- metoda kojom se računaju koeficijenti polinoma c_{ki}

Ako su segmenti jednake duljine, donja granica segmenata određena je izrazom (8).

$$x_k = \frac{k}{s} \quad k \in \{0,1,2,\dots,s\} \quad (8)$$

Ako je s potencija broja 2, izraz $(x - x_k)$, za $x_k \leq x < x_{k+1}$ izvodi se trivijalno, odbacivanjem $\log_2 s$ najznačajnijih bitova od x .

U nastavku su dane aproksimacije polinomom ovisno o stupnju polinoma.

Aproksimacija polinomom prvog stupnja

Prvi opis izvedbe PSAC-a temeljenog na polinomijalnoj aproksimaciji dao je Freeman 1989⁸.

Freeman je predložio realizaciju PSAC-a dekompozicijom prvog kvadranta sinusne funkcije na 16 ($s=16$) linearnih segmenata jednake duljine.

Koriste se tri ROM-a za spremanje podataka i 6x5 bitno množilo:

Prvi ROM: 16 x 5 bitova, sadrži nagibe linearnih segmenata c_{k1} .

Drugi ROM: 16 x 10 bitova, sadrži početne vrijednosti amplituda pojedinih segmenata c_{k0} .

Treći ROM: sadrži vrijednosti korekcija koje smanjuju maksimalnu amplitudnu pogrešku između idealne i aproksimirane sinusoide.

Freemanova arhitektura koristi jako složen princip amplitudne kompresije.

Koeficijenti pojedinih segmenata računaju se prema izrazu (9):

$$\sin \theta = \sin(C + F) = \sin C \times \cos F + \sin F \times \cos C \cong \sin C + F \times \cos C + \varepsilon(C, F) \quad (9)$$

⁸ Freeman, R.A.: 'Digital sine conversion circuit for use in direct digital synthesizers', U.S. Patent 4,809,205, 28 February 1989

Izraz je sličan Hutchinsonovom izrazu (5), radi se o sinus zbroja 2 kuta od kojih je jedan veći (grublji, C), drugi dosta mali (finiji, F).

C identificira jedan od 16 segmenata, a F je manji pribroj unutar svakog segmenta. $\sin C$ predstavlja početnu vrijednost (amplitudu) segmenta, dok je $\cos C$ nagib pojedinog segmenta. ε je korekcija.

*Bellaouar et al.*⁹

Njihov pristup razlikuje se od Freemanovog u sljedećem:

- broj s linearnih segmenata i širine sabirnica (broj bitova ulaza, izlaza i ostalih signala) u dizajnu mogu se odabrati ovisno o željenoj vrijednosti SFDR
- arhitektura počiva na generiranju kvadratnog signala na izlazu uz korištenje simetrije osmina sinusnog signala što su već ranije predložili Tan i Samueli¹⁰
- odabir koeficijenata polinoma temelji se na razvoju sinusne funkcije u Taylorov red

Za izračun vrijednosti sinusne funkcije, koriste se samo prva dva člana Taylorovog reda:

$$\sin\left(\frac{\pi x}{2}\right) \cong \sin\left(\frac{\pi x_k}{2}\right) + (x - x_k) \times \frac{\pi}{2} \cos\left(\frac{\pi x_k}{2}\right) \quad \text{za} \quad x_k \leq x < x_{k+1} \quad (10)$$

$x \in [0,1)$ i predstavlja kut iz prvog kvadranta

x_k je granica k-tog segmenta

Izraz (10) je identičan izrazu (9) bez korekcijske funkcije.

Početne amplitude i koeficijenti nagiba segmenata spremljeni su u dva manja ROM-a.

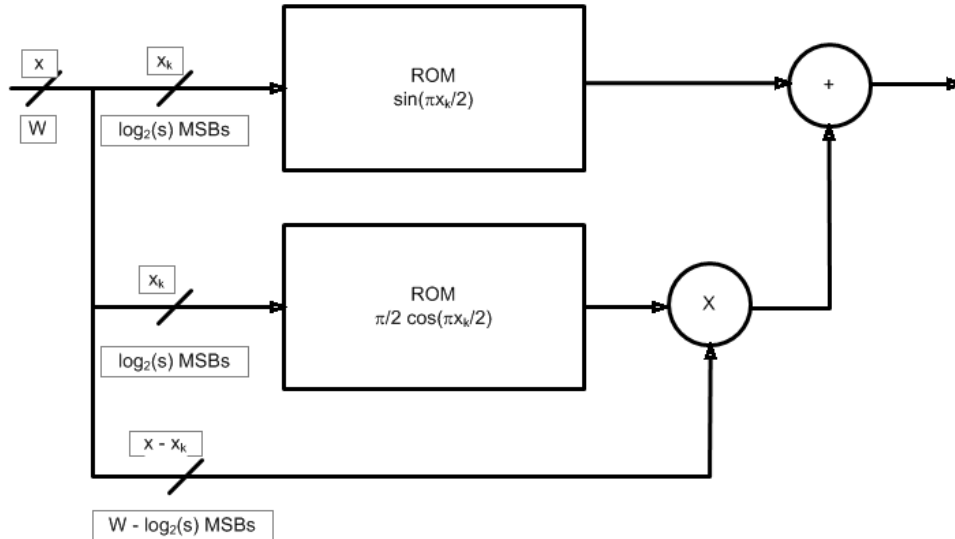
Blok shema arhitekture dana je na slici 5.

⁹ - Bellaouar, A., O'Brecht, M.S., Fahim, A.M., and Elmasry, M.I.: 'Low-power direct digital frequency synthesis for wireless communications'. *Proc. IEEE Custom Integrated Circuits Conf.*, 1999, pp. 593-596

- Bellaouar, A., O'Brecht, M.S., and Elmasry, M.I.: 'Low-power direct digital frequency synthesizer architecture'. U.S. Patent 5 999 581, 7 December 1999

- Bellaouar, A., O'Brecht, M.S., Fahim, A.M., and Elmasry, M.I.: 'Low-power direct digital frequency synthesis for wireless communications', *IEEE J. Solid State Circuits*, 2000, **35**, (3), pp. 385-390

¹⁰ Tan, L.K., and Samueli, H.: 'A 200 MHz quadrature digital synthesizer/mixer in 0.8 μm CMOS', *IEEE J. Solid State Circuits*, 1995, **30**, (3), pp. 193-200



Slika 5 Pojednostavnjena Bellaouar et al. arhitektura

Aproksimacija polinomima drugog i trećeg stupnja

Weaver i Kerr¹¹ su predložili razvoj sinusne funkcije u Taylorov red, uzimajući prva tri člana razvoja (razvoj drugog stupnja).

Aproksimacija za segmente jednake duljine računa se prema izrazu (11).

$$\sin\left(\frac{\pi x}{2}\right) \cong \sin\left(\frac{\pi x_k}{2}\right) + (x - x_k) \times \frac{\pi}{2} \cos\left(\frac{\pi x_k}{2}\right) - (x - x_k)^2 \times \frac{\pi^2}{8} \cos\left(\frac{\pi x_k}{2}\right), \quad x_k \leq x < x_{k+1} \quad (11)$$

Koristi se kombinacija ROM-ova, od kojih jedan ROM sadrži amplitude, prve i druge derivacije sinusne funkcije u točkama donjih granica segmenata.

U drugom ROM-u se nalazi „look-up“ tabela pomoću koje se izvodi razvoj trećeg člana Taylorovog reda (kvadriranje i množenje), izraz (11).

Razne varijacije izvedbe množenja i kvadriranja za dobivanje trećeg člana nastojanja su da se smanji potreban kapacitet trećeg ROM-a.

Fanucci et al.¹² su koristili četiri po dijelovima glatka polinoma drugog stupnja za aproksimaciju prvog kvadranta sinusoide. Na taj način direktno je izvedena implementacija izraza (7), za vrijednosti r=2 i s=4.

¹¹ Weaver, L.A., and Kerr, R.J.: 'High resolution phase to sine amplitude conversion', U.S. Patent 4 905 177, 27 February 1990

¹² Fanucci, L., Roncella, R., and Saletti, R.: 'A sine wave digital synthesizer based on a quadratic approximation'. *Proc. IEEE Int. Frequency Control Symp.*, 2001, pp. 806-810

Koeficijenti polinoma su, prema tvrdnjama autora, odabrani na način da se maksimizira SFDR i spremljeni su u jedan ROM.

Računanje aproksimacije zahtijeva dva sukcesivna množenja.

Koeficijenti su kvantizirani velikim brojem bitova: 12 bitova za c_{k0} , 9 bitova za c_{k1} i 5 bitova za c_{k2} , što zahtijeva kapacitet ROM-a $4 \times 26 = 104$ bita.

Potrebna množila su širina 11×5 i 11×9 bitova.

Postignut SFDR u ovom slučaju je ~ 72 dBc.

Druga izvedba istih autora daje SFDR ~ 117 dBc uz 720-bitni LUT.

*De Caro*¹³ je predstavio dva dizajna za kvadratni signal na izlazu.

Aproksimira se prva osmina sinusoide jednim polinomom drugog odnosno trećeg stupnja. Koeficijenti polinoma u oba slučaja birani su s ciljem da se poveća spektralna čistoća izlaza. Autori su za odabir razmatrali nekoliko vrsta polinoma (Taylorov red, Čebiševljeve i Legendrove polinome). Zaključeno je da se maksimalni SFDR postiže korištenjem nelinearne „Nelder-Mead simplex“ optimizacije¹⁴.

Koeficijenti su kvantizirani velikom preciznošću što je uzrok složenom računanju aproksimacije.

Postoji još mnogo autora, predložaka i različitih dizajna koji se temelje na aproksimaciji polinomima drugog i trećeg stupnja, ovdje se neće svi detaljno navoditi.

Bitno je naglasiti da sklop koji je realiziran i u sljedećim poglavljima opisan, koristi implementaciju PSAC-a na temelju aproksimacije polinomom trećeg stupnja prema radu Davora Petrinovića i Marka Brezovića „Direct Digital Frequency Synthesis Using B-Spline Piecewise-Polynomial Model“ [2].

¹³ - De Caro, D., Napoli, E., and Strollo, A.G.M.: 'Direct digital frequency synthesizers using high order polynomial approximation'. *IEEE Int. Solid State Circuits Conf., Digest of Technical Papers, San Diego, CA, USA, 3-7 February 2002*, pp.134-135

- De Caro, D., Napoli, E., and Strollo, A.G.M.: 'ROM-less direct digital frequency synthesizers exploiting polynomial approximation'. *Proc. IEEE Int. Conf. on Electronics, Circuits and Systems, Croatia, 15-18 September 2002*, pp. 481-484

¹⁴ Nelder, J.A., and Mead, R.: 'A simplex method for function minimization', *Comput. J.*, 1965, 7, pp. 308-313

Aproksimacija polinomima četvrtog i petog stupnja

Sodagar i Lahiji¹⁵ su predložili odabir jednog polinoma četvrtog stupnja, na osnovu kaskade dva parabolna polinoma prvog stupnja i to prema izrazu (6.3).

Parabolna aproksimacija prvog reda dana je izrazom (12).

$$pb_1(x) = 4x(1-x) \quad \text{za} \quad 0 \leq x < 1 \quad (12)$$

Parabolna aproksimacija drugog reda dana je izrazom (13)

$$\begin{aligned} \sin(\pi x) &\cong pb_2(x) = \\ &= pb_1(x) - 4K \times pb_1(x)(1 - pb_1(x)) = \\ &= 64Kx^4 - 128Kx^3 + (80K - 4)x^2 + (4 - 16K)x \end{aligned} \quad (13)$$

gdje je $x \in [0,1)$ i predstavlja kut u rasponu $[0, \pi)$, $K \cong 0.05591$ je konstanta.

Zahtjevi za izračun su: 3 množila, dva sklopa za dvojno komplementiranje, i jedan sklop za oduzimanje.

maksimalna amplitudna pogreška za bilo koji kut iznosi $\cong 9.81 \times 10^{-4}$ ili približno 2^{-10} , uz preciznost sinusne amplitude od 10 bita.

Poznato je da se bilo koja funkcija može transformirati u interval $[-1,1]$, korištenjem Čebiševljevih polinoma.

Palomäki i Niitylahti¹⁶ su uzeli dva Čebiševljeva polinoma četvrtog stupnja (uz $s=1$) i pomoću njih vrlo precizno aproksimirali kosinusnu funkciju u intervalu $[0, \pi/4]$.

Koeficijenti su unaprijed izračunati iz poznatih jednadžbi.

Za rezultat je dobivena poprilično složena PSAC arhitektura koja zahtijeva 2 sklopa za kvadriranje, jedno množilo, tri zbrajala sa dva ulaza i 5 zbrajala sa po pet ulaza, i to vjerojatno nije cijeli popis.

¹⁵ - Sodagar, A.M., and Lahiji, G.R.: 'A novel architecture for ROM-less sine-output direct digital frequency synthesizers by using the 2nd order parabolic approximation'. *Proc. IEEE/IEA Int. Frequency Control Symp., Kansas City, Missouri, 7-9 June 2000*, pp. 284-289

- Sodagar, A.M., and Lahiji, G.R.: 'A pipelined ROM-less architecture for sine-output direct digital synthesizers using the second-order parabolic approximation', *IEEE Trans. Circuit Syst. II, Analog Digit. Signal Process.*, 2001, **48**, (9), pp. 850-857

¹⁶ Palomäki, K.I., and Niitylahti, J.: 'Direct digital frequency synthesizer architecture based on Chebyshev approximation'. *Proc. 34th Asilomar Conf. on Signals, Systems and Computers, Pacific Grove, CA, USA, 29 Oct.-1 Nov. 2000*, pp. 1639-1643

2.2.5 Kombinacije PSAC-a i digitalno-analognog pretvornika

Kada se na izlazu želi dobiti analogni signal, redovito se u kaskadu s PSAC-om spaja digitalno-analogni pretvornik (DAC).

U tom slučaju, SFDR je ograničen izvedbom DAC-a. U nekim istraživanjima uspješno su PSAC i DAC spojeni u jednu cjelinu, gdje se na ulaz DAC-a dovodi direktno izlaz iz faznog akumulatora. Za rezultat se dobije izlaz kod kojeg je SFDR približne vrijednosti kao kod prije spomenutih izvedbi bez DAC-a, ali uz znatno smanjenje potrošnje sklopa.

Ovdje se neće navoditi različite izvedbe i njihovi autori.

2.3 USPOREDBE RAZLIČITIH IZVEDBI PSAC-A

Širok je izbor pristupa i ideja u dizajniranju PSAC-a i konačno realizaciji DDFS-a. Ovisno o potrebama korisnika, primjeni sklopa i mogućnostima tehnologije, iskoristit će se određena izvedba ili kombinacije različitih izvedbi od kojih nisu sve niti nabrojene i opisane u prethodnim poglavljima.

Za okvirnu orijentaciju u tablici 1 dana je usporedba različitih pristupa koji su prije spomenuti u tekstu.

Metoda	Prednosti	Mane
Kutna dekompozicija	<p>Matematički elegantna i intuitivna metoda</p> <p>Neposredna realizacija za jednostavnije slučajeve</p>	<p>Može biti ograničena trigonometrijskim identitetima i aproksimacijama</p> <p>Kutna segmentacija može zahtijevati proces „pokušaja i pogrešaka“ dok se ne dobije zadovoljavajući odabir segmenata</p>
Kutna rotacija	<p>Mogućnost postizanja vrlo preciznih rezultata</p> <p>Mogućnost miješanja frekvencija uz nisku cijenu</p>	<p>Dugo vrijeme kašnjenja izlaza i vrijeme potrebno za točan izlaz nakon podešavanja (promjene) frekvencije</p>
Amplitudna kompresija	<p>Bitno smanjena složenost, uz jednostavno procesuiranje</p>	<p>Obično se koristi u hibridnim metodama gdje zahtijeva dodatni PSAC (korištenje ROM-a ili neki drugi pristup)</p>
Aproksimacija polinomom	<p>Mogućnost postizanja visokog SFDR uz nisku cijenu i nisku potrošnju</p>	<p>Proces odabira parametara može biti vrlo složen i zahtijevati množenja, kvadriranja i potenciranja višeg reda za polinome višeg reda</p>
Kombinacija PSAC i DAC	<p>Sveukupno smanjena potrošnja</p>	<p>Primjenjiva samo kada se na izlazu zahtijeva analogni signal</p> <p>Na višim frekvencijama mali SFDR</p>

Tabela 1 Usporedba različitih metoda izvedbe PSAC-a

3. DDFS prema modelu B-Spline interpolacije po dijelovima glatke aproksimacije polinomom

Detaljan opis ove metode dali su Davor Petrinović i Marko Brezović [2]. Njihova istraživanja koristila su se u realizaciji sklopa u okviru ovog diplomskog rada. U nastavku će biti objašnjen čitav princip, kako matematički, tako i veza s konkretnom realizacijom ove metode.

Metodu ukratko možemo opisati kao aproksimaciju sinusne funkcije po dijelovima polinomom uz B-Spline interpolaciju sinusoide, i to jedne periode signala $[0, 2\pi)$ podijeljene na proizvoljan broj uniformnih polinomijalnih segmenata.

„Spline“ i „B-Spline“ su dobro poznati termini u matematičkoj literaturi¹⁷.

„Spline“ je specijalna funkcija sastavljena od po dijelovima glatkih krivulja (polinoma), koristi se u raznim interpolacijama kad se za rezultat želi dobiti maksimalno glatka krivulja.

„B-Spline“ kovanicu je uveo Isaac Jacob Schoenberg, a znači „basis spline“.

„B-spline“ je „Spline“ funkcija koja daje maksimalno glatku interpolaciju ovisno o stupnju, glatkoći i segmentiranju domene interpolirane funkcije.

Glatkoća krivulje u nekoj točki određena je vrijednošću funkcije u toj točki i vrijednošću derivacija u toj istoj točki. Što je veći stupanj derivacije funkcije različit od nule, to je funkcija više glatka.

¹⁷ - I.J. Schoenberg, „Contribution to the problem of approximation of equidistant data by analytic functions“, *Quart. Appl. Math.*, vol. 4, pp. 45-99, 112-141, 1946.

- I.J. Schoenberg, „Cardinal interpolation and spline functions“, *J. Approximation theory*, vol. 2, no. 2, pp. 167-206, 1969.

- C. de Boor, „On calculating with B-Splines“, *J. Approximation theory*, vol. 6, no. 1, pp. 50-62, 1972.

- C. de Boor, „A practical guide to splines“, *in Applied Mathematical Sciences*, New York; Springer-Verlag, vol. 27, 1978.

Realizacija DDS-a mora osigurati proizvoljan odabir točaka unaprijed konstruirane po dijelovima kubične funkcije. To se rješava korištenjem Farrow strukture¹⁸ koja se temelji na Hornerovom algoritmu računanja vrijednosti polinoma u točki.

Dizajn PSAC-a u sklopu ovog rada optimiziran je za FPGA arhitekturu i zahtijeva tri 18x18 bitna množila i jedan ROM kapaciteta 18 kBy.

VHDL implementacija protočne strukture ima 28-bitni ulaz i 32-bitni izlaz.

3.1 MODEL SINUSNE FUNKCIJE B-SPLINE INTERPOLACIJOM

Idealna sinusna funkcija:

$y_s(\phi) = \sin(\phi)$, gdje je ϕ zadana faza, aproksimira se po dijelovima polinomijalnom funkcijom $y_r(\phi)$ koja ima N uniformnih segmenata unutar sva četiri kvadranta $y_s(\phi)$.

Ako se radi o kubičnom polinomu, segment polinoma između faznih granica $\frac{n \cdot 2\pi}{N}$

i $\frac{(n+1) \cdot 2\pi}{N}$ računa se prema Hornerovom pravilu:

$$w_n(\Delta x) = ((d[n] \cdot \Delta x + c[n]) \cdot \Delta x + b[n]) \cdot \Delta x + a[n] \quad (14)$$

Argument Δx kubičnog polinoma je jednak nuli u desnoj točki segmenta, a jedinici u lijevoj točki. $0 \leq \Delta x \leq 1$. Kontinuiranost faze aproksimirane sinusne funkcije dobiva se spajanjem pojedinih kubičnih segmenata prema izrazu (15).

$$y_r(\phi) = \left\{ w_n(\Delta x) \middle| \Delta x = N \frac{\phi}{2\pi} - n, \quad \phi \in \left[\frac{n \cdot 2\pi}{N}, \frac{(n+1) \cdot 2\pi}{N} \right), \quad \forall n \in [0, N-1] \right\} \quad (15)$$

Zapis koeficijenata vektorskom notacijom:

$$cub[n] = [d[n] \quad c[n] \quad b[n] \quad a[n]]^T, \quad n \in [0, N-1] \quad (16)$$

¹⁸ - C.W. Farrow, „A continuously variable digital delay element“, *Proc. Circuits and Systems 1988., IEEE Int. Symp.* vol. 3., pp. 2641-2645, 1988.

- S.R. Dooley, R.W. Stewarts, T.S. Durrani, „Fast on-line B-spline interpolation“, *IEEE Electronics Lett.* vol. 35., no. 14, pp. 1130-1131, 1999.

- T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, „Splitting the Unit Delay-Tools for Fractional Delay Filter Design“ *IEEE Signal Processing Magazine*, vol. 13, pp. 30-60, Jan. 1996.

Koeficijenti se biraju tako da se minimizira maksimalna pogreška aproksimacije (17):

$$e(\phi) = y_s(\phi) - y_r(\phi) \quad , \quad \phi \in [0, 2\pi] \quad (17)$$

Razmatrano je nekoliko različitih kriterija u svrhu smanjenja $e(\phi)$ ¹⁹.

Interpolacija temeljena na B-spline-u, koja se ovdje primjenjuje, maksimizira glatkoću (regularnost) po dijelovima polinomijalnog modela.

Koeficijenti $cub[n]$ mogu se izračunati filtriranjem diskretnog sinusnog signala

$y_d[n] = y_s\left(\frac{n \cdot 2\pi}{N}\right)$ nekauzalnim sustavom definiranim transfer-matricom, izraz (18):

$$T(z) = \frac{1}{z^{-1} + 4 + z} \begin{bmatrix} -z^{-1} + 3 - 3z + z^2 \\ 3z^{-1} - 6 + 3z \\ -3z^{-1} + 3z \\ z^{-1} + 4 + z \end{bmatrix} \quad (18)$$

Z-transformacijom $cub[n]$, $y_d[n]$ uz filtriranje sa $T(z)$ dobijemo:

$$Cub(z) = [D(z) \quad C(z) \quad B(z) \quad A(z)]^T = T(z) \cdot Y_d(z) \quad (19)$$

Zahvaljujući linearnosti $T(z)$, koeficijenti $cub[n]$ su također diskretne sinusoide čije su magnitude i faze određene frekvencijskim odzivom $T(z)$ na frekvenciji

$$\omega_0 = \frac{2\pi}{N}, \quad \text{za } z = e^{j\omega_0}.$$

$$cub[n] = \begin{bmatrix} d[n] \\ c[n] \\ b[n] \\ a[n] \end{bmatrix} = \begin{bmatrix} D_m \sin(n\omega_0 + \theta_d) \\ C_m \sin(n\omega_0 + \theta_c) \\ B_m \sin(n\omega_0 + \theta_b) \\ A_m \sin(n\omega_0 + \theta_a) \end{bmatrix} \quad (20)$$

Amplituda i faza korištenog filtra $T(z)$ dani su izrazima (21) i (22).

$$|T(e^{j\omega_0})| = \begin{bmatrix} D_m \\ C_m \\ B_m \\ A_m \end{bmatrix} = \frac{1}{2 + \cos \omega_0} \begin{bmatrix} 3 \sin\left(\frac{\omega_0}{2}\right) - \sin\left(3\frac{\omega_0}{2}\right) \\ 3(1 - \cos \omega_0) \\ 3 \sin \omega_0 \\ 2 + \cos \omega_0 \end{bmatrix} \quad (21)$$

¹⁹ De Caro, D., Strollo, A.G.M., „High-performance direct digital frequency synthesizers using piecewise-polynomial approximation“, *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 2, pp. 324-337, Feb. 2005.

$$\angle T(e^{j\omega_0}) = \begin{bmatrix} \theta_d \\ \theta_c \\ \theta_b \\ \theta_a \end{bmatrix} = \begin{bmatrix} (\omega_0 - \pi)/2 \\ \pi \\ \pi/2 \\ 0 \end{bmatrix} \quad (22)$$

Opisani B-spline interpolator rekonstruira signal $y_s(\phi)$ s kontinuiranom fazom iz uzoraka $y_d[n]$ oponašajući pritom idealni „sinc“ interpolator.

Rekonstruirani signal $y_r(\phi)$ analitički se može izvesti iz izraza (15):

$$y_r(\phi) = G \cdot \sum_{i=-\infty}^{\infty} \frac{\sin(\phi(1-iN))}{(1-iN)^4} \quad (23)$$

G je odziv spektra $H(\omega)$ kardinalnog spline-a na normaliziranoj frekvenciji ω_0 .

$$G = H(\omega_0) = \frac{\sin(\omega_0/2)^4}{(\omega_0/2)^4} \cdot \frac{3}{2 + \cos \omega_0}, \quad \text{za } \omega_0 = \frac{2\pi}{N} \quad (24)$$

Rekonstruirani signal $y_r(\phi)$ sadrži skalirani željeni signal $G \cdot y_s(\phi)$ uz beskonačan broj neželjenih harmonika čije magnitude opadaju s faktorom $\approx iN^{-4}$.

RMS pogreška opisane aproksimacije može se analitički izvesti:

$$\bar{e} = \sqrt{\frac{1}{2\pi} \int_0^{2\pi} e^2(\phi) d\phi} = \sqrt{\frac{1}{2} E(\omega_0)} \quad (25)$$

$E(\omega_0)$ naziva se „kernel pogreška“:

$$E(\omega_0) = \frac{2}{2835} x^8 + \frac{32}{18711} x^{10} + \frac{226}{154791} x^{12} + O(x^{14}), \quad x = \frac{\omega_0}{2} = \frac{\pi}{N} \quad (26)$$

Za $N \gg$, ova pogreška se može aproksimirati samo prvim članom $E(\omega_0)$:

$$\bar{e} \cong \frac{\pi^4}{9\sqrt{35}} N^{-4} \cong 1.83 N^{-4} \quad (27)$$

Analiza pokazuje da je interpolacija kubičnim spline-om vrlo učinkovita, jer je pogreška reducirana faktorom $\approx N^{-4}$, gdje je N broj interpoliranih segmenata.

Za primjer 8-bitne tablice gdje je $N=256$, RMS pogreška približno iznosi $\bar{e} = 2^{-31}$.

Ovo je aproksimacija najvećeg mogućeg reda izvediva po dijelovima kubičnom interpolacijom. Dobiveni aproksimirani signal $y_r(\phi)$ ima kontinuirane derivacije prvog i drugog reda na svim prijelazima (spojnim točkama) segmenata.

3.2 SVOJSTVA DDS SA PSAC-OM TEMELJENIM NA KUBIČNOJ B-SPLINE INTERPOLACIJI

Za konkretnu primjenu, aproksimirani signal $y_r(\phi)$ može se proizvoljno podijeliti na željeni broj segmenata, što znači da je moguće proizvoljno odabrati diskretne fazne vrijednosti koje određuju granice segmenata:

$$\phi[m] = \omega_s \cdot m + \phi_0 \quad (28)$$

Da bi se olakšala izvedba DDS-a, željena frekvencija diskretne sinusoide ω_s i njena početna faza ϕ_0 , prikazuju se frakcijama:

$$\omega_s = \frac{k}{M \cdot N} \cdot 2\pi, \quad \phi_0 = \frac{k_0}{M \cdot N} \cdot 2\pi, \quad 0 < k < \frac{M \cdot N}{2} \quad (29)$$

M određuje frakcionalnu rezoluciju faze.

Konkretna vrijednost trenutne faze je:

$$\phi[m] = \frac{2\pi}{M \cdot N} (k \cdot m + k_0)_{\text{mod}(M \cdot N)} \quad (30)$$

i može se rastaviti na cjelobrojni i frakcionalni dio:

$$\phi[m] = \frac{2\pi}{N} (n[m] + \Delta x[m]) \quad (31)$$

$$n[m] = \left\lfloor \frac{(k \cdot m + k_0)_{\text{mod}(M \cdot N)}}{M} \right\rfloor, \quad n[m] \in [0, N-1] \quad (32)$$

$$\Delta x[m] = \frac{(k \cdot m + k_0)_{\text{mod}(M \cdot N)}}{M}, \quad \Delta x[m] \in [0, 1) \quad (33)$$

Željeni uzorak $y_r(\phi[m])$ dobiva se zamjenom $\Delta x[m]$ u izrazima (14) i (15) sa kubičnim koeficijentima (izrazi (20), (21) i (22)) indeksiranim prema $n[m]$.

Ako su N i M odabrani kao potencije broja 2, $N = 2^{b_N}$ i $M = 2^{b_M}$, izrazi (31), (32) i (33) jednostavno se implementiraju pomoću akumulatora širine $(b_N + b_M)$ bitova.

Gornjih b_N bitova akumulatora predstavljaju vrijednost indeksa $n[m]$, a donjih b_M bitova predstavljaju poziciju frakcionalnog uzorkovanja (unutar segmenta N), $M\Delta x[m]$.

3.3 KVANTIZACIJA POLINOMIJALNIH KOEFICIJENATA I IMPLEMENTACIJA LOGIKOM FIKSNOG ZAREZA

Radi jednostavnosti, izvodi se samo zaokruživanje idealnih realnih koeficijenata, bez dodatne optimizacije u diskretnoj domeni.

Minimiziranje RMS i maksimalne apsolutne pogreške može se postići optimizacijom pojedinačnih segmenata²⁰.

n-ti kubični segment s kvantiziranim koeficijentima $\hat{d}[n]$, $\hat{c}[n]$, $\hat{b}[n]$, $\hat{a}[n]$ i kvantiziranim međuproduktima Hornerove evaluacije polinoma dan je izrazom (34):

$$\hat{w}_n(\Delta x) = w_n(\Delta x) + e_q[n, \Delta x] = ((\hat{d}[n] \cdot \Delta x + \Delta q_1 + \hat{c}[n]) \cdot \Delta x + \Delta q_2 + \hat{b}[n]) \cdot \Delta x + \Delta q_3 + \hat{a}[n] \quad (34)$$

$e_q[n, \Delta x]$ označava izlaznu kvantizacijsku pogrešku frakcionalne pozicije Δx unutar n-tog segmenta.

Produkt $\hat{d}[n] \cdot \Delta x$ se pribraja koeficijentu $\hat{c}[n]$ koji ima konačnu rezoluciju b_c , pa je logično je da se taj produkt također kvantizira na b_c bitova. Analogno se čini i sa ostalim međuproduktima koji se pribrajaju koeficijentima $\hat{b}[n]$ odnosno $\hat{a}[n]$.

Pogreška uslijed zaokruživanja zapisana je kao Δq_i ($i=1, 2, 3$).

Pogreške kvantizacije koeficijenata dane su izrazima:

$$\begin{aligned} \Delta d[n] &= \hat{d}[n] - d[n] \\ \Delta c[n] &= \hat{c}[n] - c[n] \\ \Delta b[n] &= \hat{b}[n] - b[n] \\ \Delta a[n] &= \hat{a}[n] - a[n] \end{aligned} \quad (35)$$

Ukupna kvantizacijska pogreška dana je izrazom (36):

$$e_q[n, \Delta x] = \Delta d[n] \cdot \Delta x^3 + (\Delta c[n] + \Delta q_1) \Delta x^2 + (\Delta b[n] + \Delta q_2) \Delta x + (\Delta a[n] + \Delta q_3) \quad (36)$$

²⁰ De Caro, D., Strollo, A.G.M., „High-performance direct digital frequency synthesizers using piecewise-polynomial approximation“, *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 2, pp. 324-337, Feb. 2005.

Na početku se pretpostavlja jednaka rezolucija za sva četiri koeficijenta $b_d=b_c=b_b=b_a=b_w$, gdje je b_w izlazna rezolucija $\hat{w}_n(\Delta x)$.

Da se pojednostavni analiza pogreške, sve kvantizacijske pogreške su modelirane kao set statistički nezavisnih i uniformno distribuiranih varijabli u rasponu

$[-0.5, 0.5)$ LSB_w . Na taj način sva zaokruženja odgovaraju vrijednosti $LSB_w=2^{-b_w}$.

Uz pretpostavku jednake vjerojatnosti pojavljivanja svih indeksa n , očekivana kvadratna kvantizacijska pogreška na izlazu iznosi:

$$E[e_q^2[n, \Delta x]] = \frac{1}{6}(1 + \Delta x^2 + \Delta x^4 + \frac{1}{2}\Delta x^6)LSB_w^2 \quad (37)$$

Maksimalna apsolutna pogreška ograničena je sa

$$|e_q[n, \Delta x]| \leq (1 + |\Delta x| + |\Delta x|^2 + \frac{1}{2}|\Delta x|^3)LSB_w \quad (38)$$

Obje funkcije pogreške su monotono rastuće sa $|\Delta x|$, uz maksimalnu vrijednost na

desnom kraju interpolacijskog segmenta za $\Delta x = 1$: $E[e_q^2[n, 1]] = \frac{7}{12}LSB_w^2$ i

$$|e_q[n, 1]| \leq \frac{7}{2}LSB_w.$$

Ako se argument Δx kubičnog polinoma transformira u interval $[-0.5, 0.5)$, tada se obje pogreške znatno smanjuju jer vrijedi:

$$\frac{E[e_q^2[n, 1]]}{E[e_q^2[n, 0.5]]} = \frac{448}{169} \cong 2.6 \quad \text{i} \quad \frac{\max|e_q[n, 1]|}{\max|e_q[n, 0.5]|} = \frac{56}{29} \cong 1.9 \quad (39)$$

Potrebne rezolucije koeficijenata b , c i d i prva dva međuprodukta mogu se reducirati tako da se postigne jednaka izlazna pogreška. Ova činjenica omogućuje bitno smanjenje složenosti realiziranog sklopa.

Potrebno je provesti jednostavne transformacije dane izrazima (40), (41) i (42).

$$\Delta x' = \begin{cases} \Delta x, & \Delta x < 0.5 \\ \Delta x - 1 & \Delta x \geq 0.5 \end{cases} \quad (40)$$

$$w_n(\Delta x) = w'_n(\Delta x') = d'[n] \cdot \Delta x'^3 + c'[n] \cdot \Delta x'^2 + b'[n] \cdot \Delta x' + a'[n] \quad (41)$$

$$\begin{cases} d'[n] = d[n] \\ c'[n] = c[n] \\ b'[n] = b[n] \\ a'[n] = a[n] \end{cases}, \quad \Delta x' \geq 0 \quad \begin{cases} d'[n] = d[n] \\ c'[n] = c[(n+1)_{\text{mod } N}] \\ b'[n] = b[(n+1)_{\text{mod } N}] \\ a'[n] = a[(n+1)_{\text{mod } N}] \end{cases}, \quad \Delta x' < 0 \quad (42)$$

U konkretnoj implementaciji u VHDL-u, Δx će biti transformiran u interval $[-1, 1)$, na taj način će se binarni zapis lako interpretirati kao dvojni komplement uz još jednostavniju transformaciju koja zahtijeva samo eventualno komplementiranje Δx . Također se ovakvom transformacijom postiže direktno čitanje koeficijenata $\hat{d}[n]$, $\hat{c}[n]$, $\hat{b}[n]$, $\hat{a}[n]$ bez potrebe za dohvatom koeficijenta koji pripada sljedećem segmentu $[n+1]$ kao što je bio slučaj u izrazu (42).

Nakon transformacije izlazna pogreška je $e_q[n, \Delta x] = \hat{w}_n'(\Delta x') - w_n'(\Delta x')$.

Da bi se kvantizacijom koeficijenata postigli ujednačeni doprinosi izlaznoj pogrešci, rezolucije koeficijenata je potrebno izabrati na način:

$$b_d' = b_c' - 1 = b_b' - 2 = b_a' - 3 = b_w - 3 \quad (43)$$

Ogovarajuće kvantizacijske pogreške u tom slučaju iznose:

$$\Delta d' \in [-4, 4), \quad \Delta c', \Delta q_1' \in [-2, 2), \quad \Delta b', \Delta q_2' \in [-1, 1), \quad \Delta a', \Delta q_3' \in [-0.5, 0.5) \text{ LSB}_w.$$

Nakon ovakve dodjele rezolucije pojedinim koeficijentima i međuproduktima, očekivana i maksimalna pogreška iznose:

$$E[e_q'^2[n, \Delta x']] = \frac{1}{6}(1 + (2\Delta x')^2 + (2\Delta x')^4 + \frac{1}{2}(2\Delta x')^6) \text{LSB}_w^2 \quad (44)$$

$$|e_q'[n, \Delta x']| \leq (1 + |2\Delta x'| + |2\Delta x'|^2 + \frac{1}{2}|2\Delta x'|^3) \text{LSB}_w \quad (45)$$

Pogreška je jednaka pogrešci kod izvedbe se maksimalnom rezolucijom sva četiri koeficijenata i tri međuprodukta.

Ovom transformacijom se štedi 6 bitova po jednom redu tablice.

Dodatna ušteda na veličini potrebnog ROM-a dobiva se i normalizacijom kubičnih koeficijenata, bez negativnog utjecaja na točnost PSAC-a.

Uobičajeni faktor normalizacije $2^{p_{xx}}$ za koeficijente d, c i b ($xx=d, c, b$) određen je izrazom (46).

$$\begin{bmatrix} p_d \\ p_c \\ p_b \end{bmatrix} = - \left\lceil \log_2 \begin{bmatrix} D_m \\ C_m \\ B_m \end{bmatrix} \right\rceil = \begin{bmatrix} 3b_N - 6 \\ 2b_N - 5 \\ b_N - 3 \end{bmatrix}, \text{ gdje je } b_N = \log_2 N \quad (46)$$

Pojedinačne rezolucije koeficijenata konačno su:

$$\begin{bmatrix} \overline{b_d} \\ \overline{b_c} \\ \overline{b_b} \\ \overline{b_a} \end{bmatrix} = \begin{bmatrix} b_d' \\ b_c' \\ b_b' \\ b_a' \end{bmatrix} - \begin{bmatrix} p_d \\ p_c \\ p_b \\ 0 \end{bmatrix} = b_w - \begin{bmatrix} 3b_N - 3 \\ 2b_N - 3 \\ b_N - 2 \\ 0 \end{bmatrix} \quad (47)$$

3.3.1 Korištenje kvadrantne simetrije sinusne funkcije

Dodatno pojednostavnjenje izvedbe PSAC-a koje se redovito primjenjuje je korištenje svojstava simetrije i antisimetrije sinusne funkcije. Na taj način moguće je realizirati PSAC samo za izračun aproksimacije amplituda za kutove prvog kvadranta i iz tih vrijednosti rekonstruirati cijelu periodu funkcije.

Ako su a, b, c i d koeficijenti polinoma za neki segment prvog kvadranta sinusne funkcije, koeficijenti ostalih kvadranta dobiju se iz izraza danih u tablici 2.

1. KVADRANT	2. KVADRANT	3. KVADRANT	4. KVADRANT
a	a	-a	-a
b	-b	-b	b
c	c	-c	-c
d	-d	-d	d

Tabela 2 Međusobna veza koeficijenata po kvadrantima

Zapis u tablici 2 ne predstavlja stvarne predznake koeficijenata, već odnos koeficijenata 2., 3. i 4. kvadranta u odnosu na već izračunate koeficijente prvog kvadranta.

O kojem kvadrantu se ustvari radi lako je vidjeti iz MSB i NSB fazne riječi koju daje akumulator.

Tablica 3 daje prikaz stvarnih predznaka koeficijenata i sadržaj MSB i NSB bitova za svaki kvadrant.

1. KVADRANT		2. KVADRANT		3. KVADRANT		4. KVADRANT	
MSB=0	NSB=0	MSB=0	NSB=1	MSB=1	NSB=0	MSB=1	NSB=1
a > 0		a > 0		a < 0		a < 0	
b > 0		b < 0		b < 0		b > 0	
c < 0		c < 0		c > 0		c > 0	
d < 0		d > 0		d > 0		d < 0	

Tabela 3 Stvarni predznak koeficijenata i sadržaj MSB i NSB bitova po kvadrantima

Svi koeficijenti pohranjeni u ROM tablici su pozitivni, dodjeljivanje predznaka izvedeno je jednostavnim operacijama komplementiranja ovisno o kvadrantu za koji se koeficijenti računaju.

Preciznije rečeno, u ROM su pohranjene frakcije koeficijenata, jer se oni ionako nalaze u rasponu [0, 1).

Nakon dohvata koeficijenata iz ROM-a dodaje se MSB=0 svakom koeficijentu, i ovisno o MSB i NSB fazne riječi generirane akumulatorom, se provodi dvojno komplementiranje određenih koeficijenata, prema tablici 3.

3.4 IMPLEMENTACIJA PSAC-a NA FPGA SKLOP

Cilj ovog rada je primijeniti metodu dosad opisivanu u teoriji na stvarnu realizaciju PSAC arhitekture za DDS visoke točnosti i provjeriti odgovaraju li rezultati zahtjevima postavljenim i razrađenim u prethodnim poglavljima.

Sklop je realiziran VHDL-om za sveprisutnu Xilinx Spartan 3 FPGA arhitekturu²¹.

Cilj je bio smjestiti sklop u jedan blok-RAM i tri 1.17x1.17-bitna množila uz postizanje najveće moguće točnosti.

²¹ *Spartan-3 Generation FPGA User Guide*, UG331, Xilinx Inc., San Jose, CA, USA, 2009.

Proračunati parametri PSAC-a su:

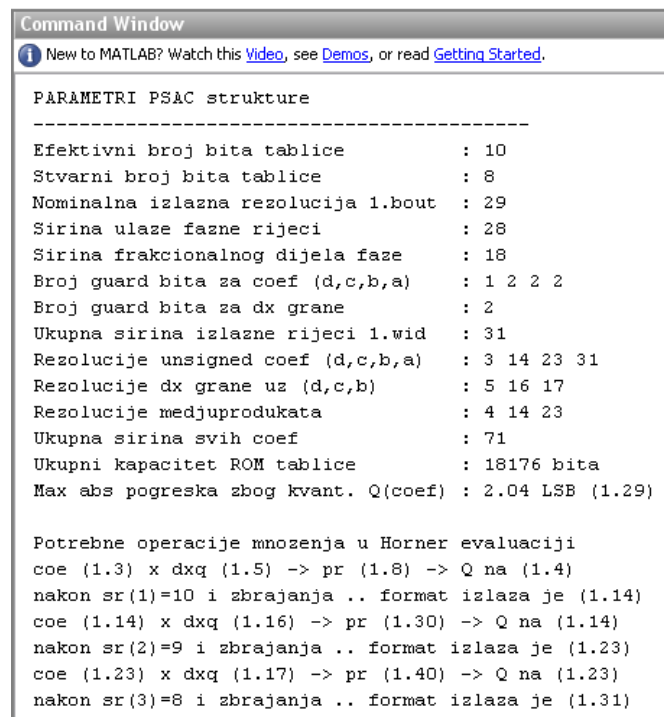
- ulazna rezolucija: $b_N + b_M = 28$ bitova
- rezolucija tablice $b_N = 10$ bitova
- izlazna rezolucija sinusnog signala: $1.b_w = 1.31$ bit ($LSB_w = 2^{-31}$)
- rezolucije koeficijenata: $b_d = 3$, $b_c = 14$, $b_b = 23$ i $b_a = 31$ bit
- ukupna širina tablice ROM-a: 71 bit
- rezolucije Δx grana: 1.5, 1.16 i 1.17 bitova
- rezolucije međuprodukata: 1.4, 1.14 i 1.23 bita

ROM je realiziran kao „dual-ported“²² ROM s dva 36-bitna podatkovna priključka i 8-bitnom adresom.

Ponašanje sklopa kroz sve faze protočne strukture, kao i proračun optimalnih parametara i svih potrebnih rezolucija signala simulirani su u MATLAB-u.

Koeficijenti koje je potrebno pohraniti u ROM, također su izračunati u MATLAB-u.

Rezultati se vide na slici 6.



```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

PARAMETRI PSAC strukture
-----
Efektivni broj bita tablice           : 10
Stvarni broj bita tablice             : 8
Nominalna izlazna rezolucija 1.bout  : 29
Sirina ulaze faze riječi              : 28
Sirina frakcionalnog dijela faze      : 18
Broj guard bita za coef (d,c,b,a)    : 1 2 2 2
Broj guard bita za dx grane           : 2
Ukupna sirina izlazne riječi 1.wid   : 31
Rezolucije unsigned coef (d,c,b,a)  : 3 14 23 31
Rezolucije dx grane uz (d,c,b)       : 5 16 17
Rezolucije međuprodukata              : 4 14 23
Ukupna sirina svih coef               : 71
Ukupni kapacitet ROM tablice          : 18176 bita
Max abs pogreska zbog kvant. Q(coef) : 2.04 LSB (1.29)

Potrebne operacije množenja u Horner evaluaciji
coe (1.3) x dxq (1.5) -> pr (1.8) -> Q na (1.4)
nakon sr(1)=10 i zbrajanja .. format izlaza je (1.14)
coe (1.14) x dxq (1.16) -> pr (1.30) -> Q na (1.14)
nakon sr(2)=9 i zbrajanja .. format izlaza je (1.23)
coe (1.23) x dxq (1.17) -> pr (1.40) -> Q na (1.23)
nakon sr(3)=8 i zbrajanja .. format izlaza je (1.31)
```

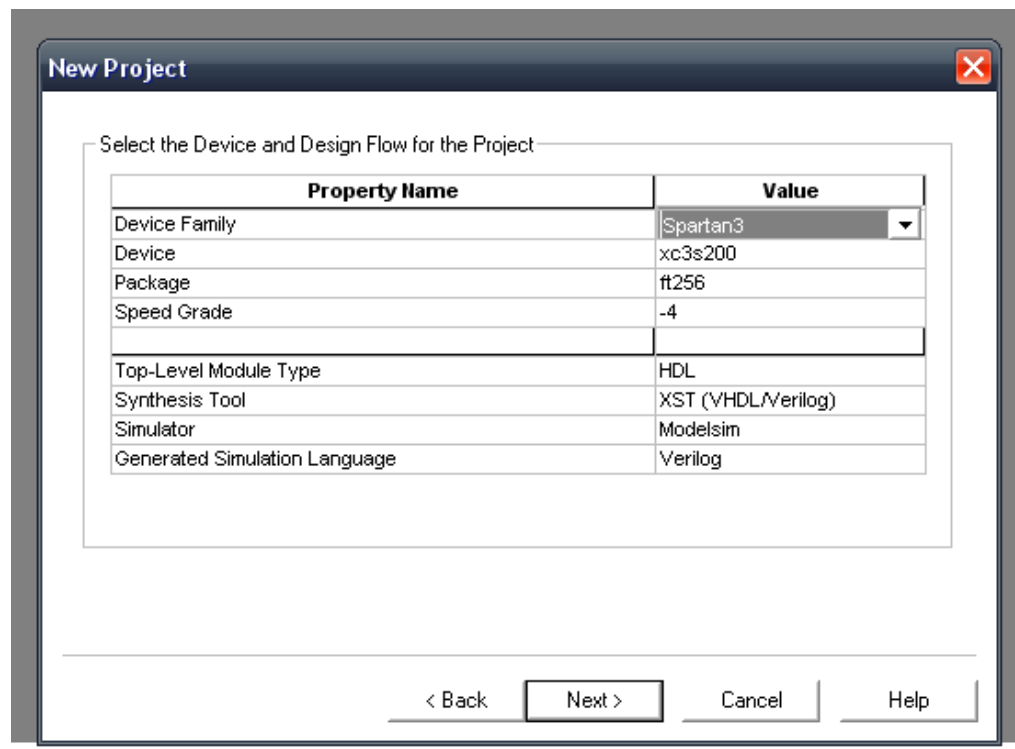
Slika 6 Rezultati optimizacije parametara sklopa dobiveni pomoću programskog paketa **MATLAB**

²² ROM s dva ulaza i dva izlaza

4. Realizacija *DDFS* pomoću programskog paketa Xilinx

4.1 UVOD U PROGRAMSKI PAKET XILINX

Implementacija se vrši na Xilinx FPGA sklopu. Sklop se odabire pri kreiranju novog projekta u izborniku prikazanim na slici 7.



Slika 7 Xilinx - Izbornik za odabir sklopa pri kreiranju novog projekta

Odabrana svojstva sklopa:

- Familija kojoj uređaj pripada: Spartan 3
- Oznaka sklopa: xc3s200
- Kućište: ft256 (256-ball Fine-Pitch Thin Ball Grid Array (FTBGA))
- Brzina: -4 (Standard Performance)

Nakon kreiranja projekta, kreće se sa programiranjem pojedinih komponenti ili modula.

Modul je dio sklopa koji obavlja određenu funkciju za vrijeme jedne periode referentnog takta. Neki moduli su napravljeni kao generički, što znači da pojedini parametri nisu fiksno definirani, već se pri korištenju takvog modula u pojedinoj fazi rada sklopa naknadno definiraju ti parametri. Bit će jasnije iz kasnijih primjera na koji način se to izvodi. Parametri koji se ne definiraju fiksno su najčešće rezolucije ulaza i izlaza.

Nakon kreiranja referentnog modela (*Behavioral model*) koji opisuje funkciju pojedine komponente kreira se i ispitno okruženje (*Test Bench model*) za svaku komponentu u kojem se definira stanje na ulaznim priključcima u određenim vremenskim trenucima.

Pomoću ispitnog modela moguće je pokrenuti *Model-Sim* simulator koji simulira zadano stanje na ulaznim priključcima i stanje izlaznih priključaka ovisno o zadanim parametrima radnog okruženja.

Moguće je pomoću ovog programskog paketa postaviti željena vremenska i prostorna ograničenja na komponentu (*User Constraints*).

Potrebno je razviti fizički ostvariv model (RTL model odnosno *Register Transfer Level Model*), da bi se sklop mogao implementirati.

Konačno se vrši sinteza i implementacija sustava a korisnik može kao rezultat vidjeti preglednik (*Floor Planner*) koji pokazuje unutrašnji raspored ćelija FPGA sklopa.

4.2 BLOK SHEMA SUSTAVA I REALIZACIJA PROTOČNE ARHITEKTURE

Svi parametri za implementaciju PSAC-a odnosno DDS-a, dobiveni su simulacijom ponašanja sklopa pomoću programskog paketa MATLAB i dani su u poglavlju 4.4.

Svi signali u sklopu su interpretirani kao dvojni komplement.

Sklop treba obavljati aritmetičke operacije dane izrazom

$$w_n(\Delta x) = ((d[n] \cdot \Delta x d + c[n]) \cdot \Delta x c + b[n]) \cdot \Delta x b + a[n] \quad (48)$$

Radi lakšeg razumijevanja pojedinog dijela blok sheme sklopa uvode se oznake međurezultata:

$$d[n] \cdot \Delta x d = m_2 \quad (49)$$

$$d[n] \cdot \Delta x d + c[n] = m_3 \quad (50)$$

$$(d[n] \cdot \Delta x d + c[n]) \cdot \Delta x c = m_4 \quad (51)$$

$$(d[n] \cdot \Delta x d + c[n]) \cdot \Delta x c + b[n] = m_5 \quad (52)$$

$$((d[n] \cdot \Delta x d + c[n]) \cdot \Delta x c + b[n]) \cdot \Delta x b = m_6 \quad (53)$$

Blok shema DDFS-a dana je slikom 8.

Sve komponente su sinkronizirane tako da daju izlaz na rastući brid takta.

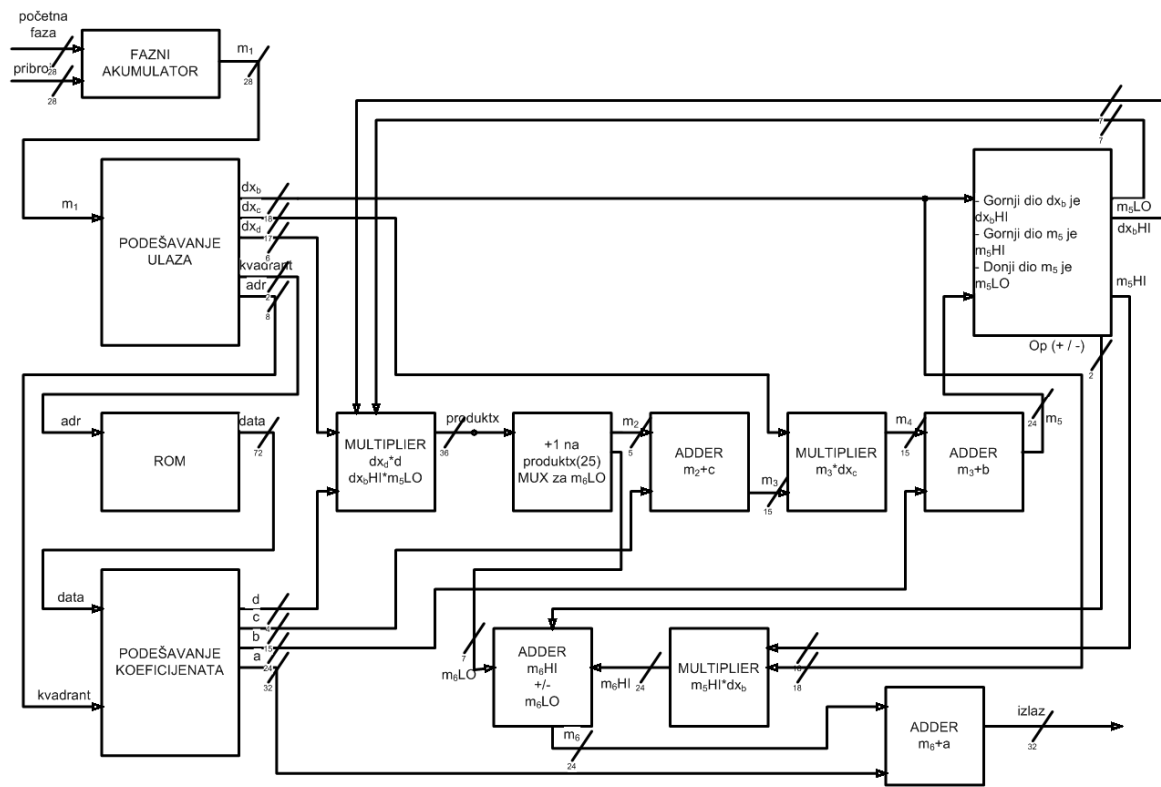
Realizirana je protočna arhitektura, što znači da svi moduli vidljivi na blok shemi istovremeno obavljaju svoje funkcije, ali ulazi i izlazi ne odgovaraju obradi istog uzorka, već svaki sljedeći modul obrađuje izlaz prethodnog modula dobiven jedan period ranije. Da bi se to postiglo, potrebno je neke signale zakasniti određeni broj perioda prije nego dođu na ulaz modula koji ih obrađuje.

Modul koji je zadužen za kašnjenje signala (*Register*) nije prikazan na blok shemi, a iskorišten je čak 8 puta.

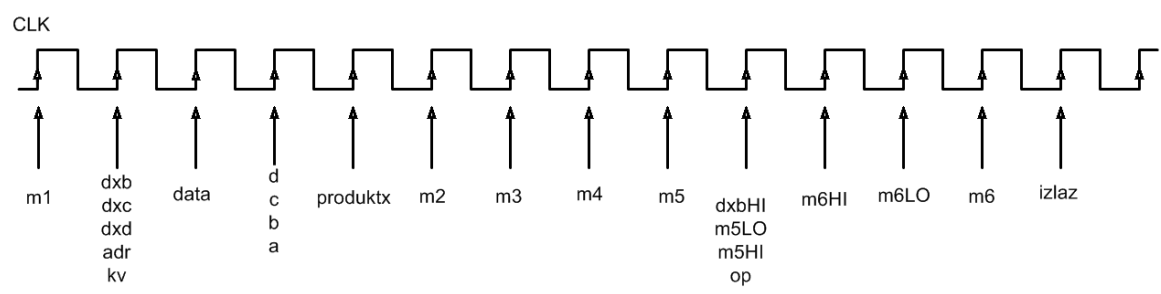
Na slici 9 dan je slijed od nekoliko perioda signala takta (CLK) uz praćenje propagacije signala kroz sklop.

Prvi izlaz pojavljuje se nakon 13 perioda takta, na rastućem bridu CLK i potom na svakom sljedećem.

U sljedećim poglavljima detaljno će se objasniti funkcija svih modula.



Slika 8 blok shema DDFS-a



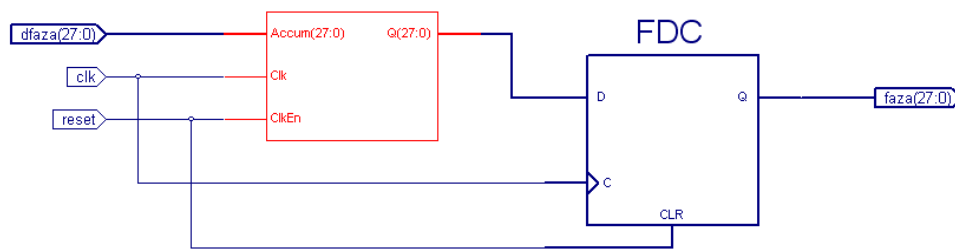
Slika 9 Propagacija signala kroz sustav

4.3 MODULI UNUTAR DDFS-A

Fazni akumulator (phase_accumulator)

Fazni akumulator generira novu faznu riječ na svaki rastući brid signala takta. Na ulaz se dovodi 28-bitna fazna riječ (*pocetna_faza*) i 28-bitni pribroj (*dfaza*). Izlaz je također 28-bitni (*faza*).

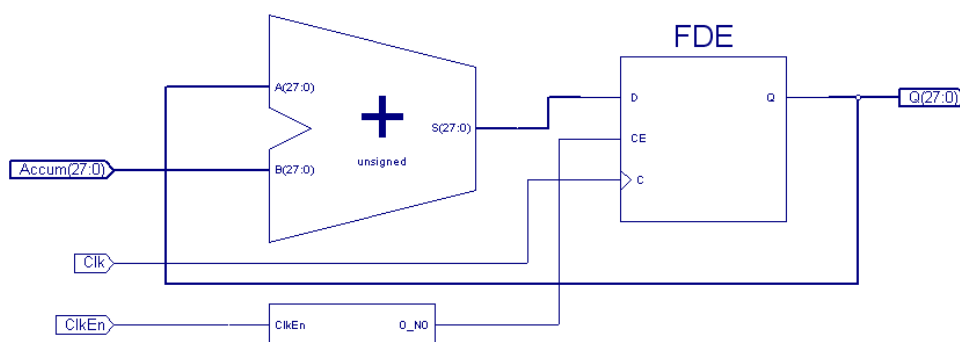
Blok shema dana je slikom 10.



Slika 10 Blok shema faznog akumulatora

Sastavni dijelovi su funkcijski blok i D-bistabil koji drži trenutni signal i omogućava sinkronizaciju na rastući brid takta.

Detaljniji prikaz bloka za generiranje nove fazne riječi (označen crvenom bojom na slici 10) dan je slikom 11.



Slika 11 Detaljniji prikaz funkcijskog bloka faznog akumulatora

Riječ je o običnom zbrajalu na čije ulaze se dovode prethodno izračunata fazna riječ i fazni pribroj koji je konstantan sve dok korisnik ne zada novi pribroj.

Definiranje entiteta u VHDL-u prikazano je slikom 12. Vide se ulazni i izlazni priključci i njihove rezolucije.

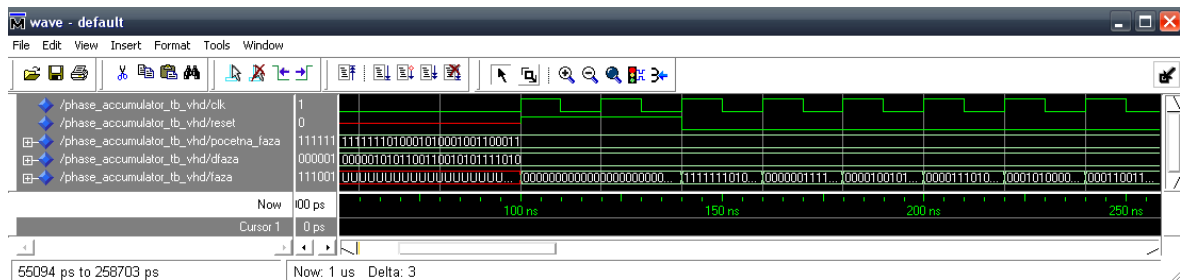
```
entity phase_accumulator is
    port (
        signal clk: in std_logic;
        signal reset: in std_logic;
        signal pocetna_faza: in std_logic_vector (27 downto 0);
        signal dfaza: in std_logic_vector (27 downto 0);
        signal faza: out std_logic_vector (27 downto 0)
    );
end;
```

Slika 12 Definiranje entiteta faznog akumulatora

Za provjeru ispravnosti rada potrebno je definirati i ispitno okruženje (phase_accumulator_tb). Vizualni rezultat simulacije sklopa pomoću ModelSim simulatora, u zadanom ispitnom okruženju prikazan je slikom 13.

Na prvi pogled važno je uočiti trenutke u kojima sklop daje izlaz (rastući brid takta), nakon što je postavljen valjan ulaz i ponašanje u odnosu na aktivan i neaktivan signal *reset*.

Ovakav prikaz simulacije nije pogodan za provjeru ispravnosti izlaza, već se rezultati simulacije spremaju u tekstualnu datoteku kao niz brojeva (*izlaz* ∈ N) i kasnije čitanjem i grafičkim prikazom u MATLAB-u provjerava podudarnost sa referentnim rezultatima.



Slika 13 Simulacija rada faznog akumulatora ModelSim simulatorom

Modul za podešavanje ulaza (podesenja)

Fazna riječ koja se dovodi na ulaz ovog sklopa (signal m_1) ima rezoluciju 28 bitova, od kojih gornjih 10 govori o kojem se segmentu (od ukupno 1024) sinusoide radi.

Kako su u ROM-u spremljeni koeficijenti za samo 256 segmenata prve četvrtine perioda sinusnog signala, MSB i NSB (signal kv) ulaza određuju o kojem se kvadrantu radi, pa se na osnovu te informacije pročitani koeficijenti kasnije transformiraju prema tablici 2. u poglavlju 4.3.1.

Adresu predstavlja sljedećih 8 bitova (od 25. do 18. bita).

Signal kv služi i za transformaciju adrese (dio koda na slici). Ova činjenica također proizlazi iz simetrije i antisimetrije sinusnog signala ali na osnovu segmenata.

Ako je perioda podijeljena na 1024 segmenta, tada je 1. segment osnosimetričan 512. segmentu, 2. segment osnosimetričan 511. itd. s obzirom na $\pi/2$.

Isto vrijedi i za segmente od 513. do 1024. koji su simetrični s obzirom na os $3\pi/2$.

Postoji također i centralna simetrija oko π pa je 1. segment simetričan 1024., 2. je simetričan 1023., 511. je simetričan 512. itd.

```
if (ulaz(27 downto 26)="01" or ulaz(27 downto 26)="11") then
  adr<=not(ulaz(25 downto 18));
else
  adr<=ulaz(25 downto 18);
end if;
```

Slika 14 Dio koda za transformaciju signala adr

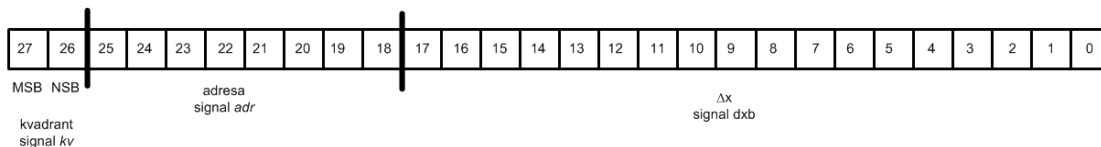
Najnižih 18 bitova predstavljaju signal $\Delta x = \Delta xb$ iz izraza 48.

Kako je $\Delta x \in [0,1)$ i predstavlja točno mjesto uzorkovanja unutar određenog segmenta, ovaj sklop radi i transformaciju tog signala u interval $[-1,1)$, pa se kasnije signal samo interpretira kao dvojni komplement.

Transformacija je krajnje jednostavna i izvodi se komplementiranjem signala Δx .

Δxc i Δxd su istovjetni signalu $\Delta x = \Delta xb$, ali su odrezani na gornjih 17 odnosno 6 bitova tog signala.

Signal m_1 prikazan je slikom 15.



Slika 15 Signal m1, ulaz u sklop *podesenja*

Ulazno izlazni priključci i njihove rezolucije vide se na slici 16, koja pokazuje definiranje entiteta u VHDL-u.

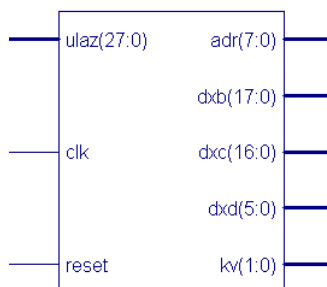
```

30  entity podesenja is
31
32      port (
33          signal clk: in std_logic;
34          signal reset: in std_logic;
35          signal ulaz: in std_logic_vector(27 downto 0);
36          signal adr: out std_logic_vector(7 downto 0);
37          signal kv: out std_logic_vector(1 downto 0);
38          signal dxb: out std_logic_vector(17 downto 0);
39          signal dxc: out std_logic_vector(16 downto 0);
40          signal dxd: out std_logic_vector(5 downto 0)
41      );
42  end podesenja;

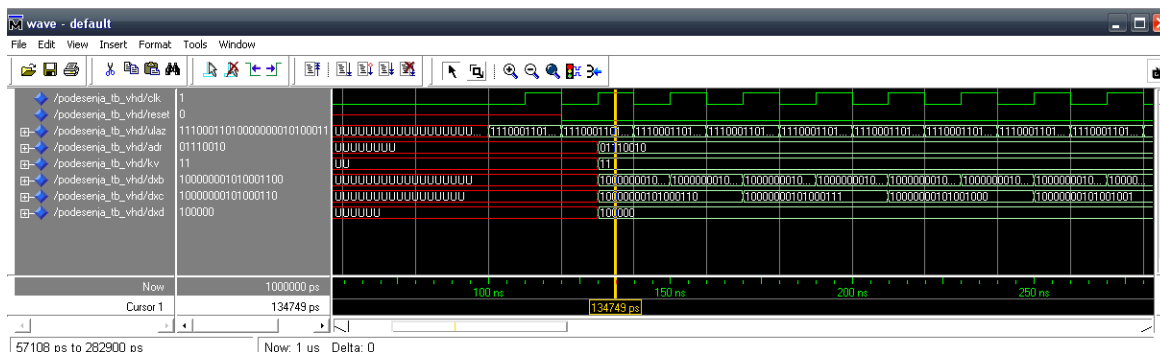
```

Slika 16 Entitet sklopa *podesenja*

Blok shema ove komponente i rezultat simulacije ModelSim simulatorom vide se na slikama 17 i 18.



Slika 17 Blok shema modula *podesenja*



Slika 18 Simulacija modula *podesenja* ModelSim simulatorom

Spremanje koeficijenata u ROM

Koeficijenti d, c, b i a za pojedini segment spremljeni su u gotovu komponentu *RAMB16_S36_S36*: *Virtex-II/II-Pro, Spartan-3/3E 512 x 32 + 4 Parity bits Dual-Port RAM*, iz biblioteke *UNISIM.VComponents*.

Radi se o *Block RAM*-u s dva ulaza i izlaza, ali ovdje će se koristiti kao RAM s jednim ulazom i jednim izlazom.

Odabrani *Block RAM* u ovom slučaju „oponaša“ ROM jer su koeficijenti fiksno upisani i sustav radi tako da iz RAM-a samo čita i ništa ne upisuje.

Kapacitet odabranog *Block RAM*-a je:

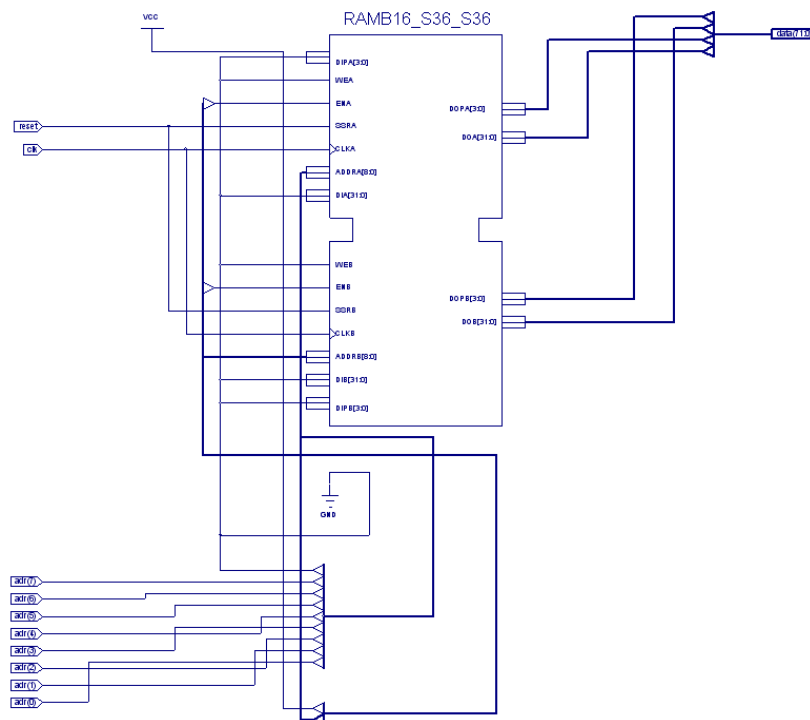
$$512 * (32 \text{ bita} + 4 \text{ bita za paritet}) = 18432 \text{ bita}$$

Kapacitet koji je potreban za spremanje koeficijenata iznosi:

$$256 * (3 + 14 + 23 + 31) = 18176 \text{ bitova}$$

Na izlazu je signal *data* rezolucije 72 bit koji predstavlja slijedni zapis d, c, b i a koeficijenata s tim da MSB nije iskorišten. U sljedećem modulu se koeficijenti iščitavaju iz signala *data*.

Na slici 19 dana je blok shema modula *ROM* dok se na slici 20 vidi dio koda u kojem se definira entitet modula *ROM* i instancira gotova komponenta *Block RAM* u modul.



Slika 19 Blok shema modula *ROM*

```

28 library UNISIM;
29 use UNISIM.VComponents.all;
30
31 entity ROM is
32     Port (
33         signal clk : in std_logic;
34         signal reset : in std_logic;
35         signal adr : in std_logic_vector (7 downto 0);
36         signal data : out std_logic_vector (71 downto 0)
37     );
38 end ROM;
39
40 architecture Behavioral of ROM is
41
42 begin
43
44
45     -- RAMB16_S36_S36: Virtex-II/II-Pro, Spartan-3/3E 512 x 32 + 4 Parity bits Dual-Port RAM
46     -- Xilinx HDL Language Template, version 10.1
47
48     RAMB16_S36_S36_inst : RAMB16_S36_S36
49     generic map (
50         INIT_A => X"000000000", -- Value of output RAM registers on Port A at startup
51         INIT_B => X"000000000", -- Value of output RAM registers on Port B at startup
52         SRVAL_A => X"000000000", -- Port A output value upon SSR assertion
53         SRVAL_B => X"000000000", -- Port B output value upon SSR assertion
54         WRITE_MODE_A => "NO_CHANGE", -- WRITE_FIRST, READ_FIRST or NO_CHANGE
55         WRITE_MODE_B => "NO_CHANGE", -- WRITE_FIRST, READ_FIRST or NO_CHANGE
56         SIM_COLLISION_CHECK => "ALL", -- "NONE", "WARNING", "GENERATE_X_ONLY", "ALL"
57         -- The following INIT_xx declarations specify the initial contents of the RAM
58         -- Address 0 to 127
59         INIT_00 => X"05E36EA9051A8E5C0451A1770388A9EA02BFA9A401F6A297012D96B1006487E3",
60         INIT_01 => X"8C27C3890B5F8D9F8A973BA509CECF8909064B3A083DB0A7877501BE86AC406F",
61         INIT_02 => X"9264994E919D894110D64DBD100EE8AD8F475BFF8E7FA99E8DB7D3760CEFDB76",
62         INIT_03 => X"1896172897D0A7BC970AFD8D16451A83157F008614B8B17F13F22F58932B7BF9",
63         INIT_04 => X"9EB86B461DF5163F9D31774D1C6D90531BA963359AE4F1D61A203E1B995B49EA",
64         INIT_05 => X"A4C7CD3324070B082345EFF8A2847DE0A1C2B69C21009C0CA03E300D1F7B7481",
65         INIT_06 => X"AAC080262A02C7B8A944A7A2A88621B927C737D3A707EBC726483F6C2588349D",
66         INIT_07 => X"309ED556AFE49BA72F29EBC2E6EC792ADB330C72CF72939AC3AB2B92B7DCF17",
67         INIT_08 => X"B65F2E3B35A8E625B4F219A8343ACA873382FA88B2CAAB6F3211DF043158970E",

```

Slika 20 Entitet modula ROM i instanciranje gotovog Block RAM-a

Modul za podešavanje koeficijenata (Podesavanje_coef)

Ulaz u ovaj modul je izlaz iz modula *ROM* koji ima rezoluciju 72 bita. U ovom modulu se taj podatak razlaže na koeficijente, potom se oni još i transformiraju prema tablici 2 iz poglavlja 4.3.1.

Da bi se znalo o kojem kvadrantu se radi, jedan od ulaza je i signal *kv*, koji je izlaz modula *podesenja*. Kako izlaz iz *ROM-a* kasni za signalom *kv* jedan period takta, potrebno je *kv* to vrijeme zakasniti. Svi zakašnjeni signali imaju nastavak „_t“, tako da je stvarni ulaz u ovaj modul *kv_t*.

Ulazni i izlazni priključci dani su slikom 21 koja pokazuje kod za definiranje entiteta modula i slikom 22 koja pokazuje blok shemu.

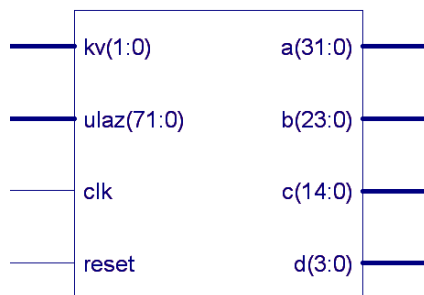
Simulacija je dana slikom 23.

```

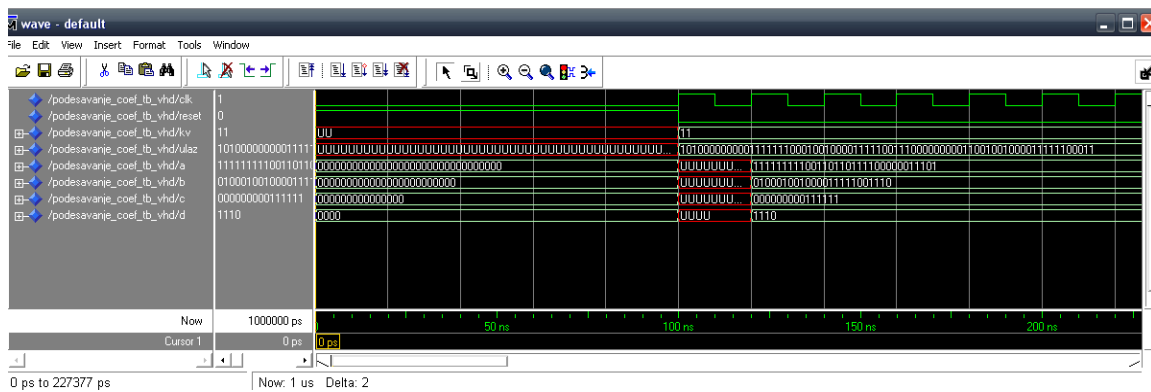
31 entity Podesavanje_coef is
32
33     port(
34         signal clk: in std_logic;
35         signal reset: in std_logic;
36         signal kv: in std_logic_vector(1 downto 0);
37         signal ulaz: in std_logic_vector(71 downto 0);
38         signal a: out std_logic_vector(31 downto 0);
39         signal b: out std_logic_vector(23 downto 0);
40         signal c: out std_logic_vector(14 downto 0);
41         signal d: out std_logic_vector(3 downto 0)
42     );
43
44 end Podesavanje_coef;
45

```

Slika 21 Entitet modula *Podesavanje_coef*



Slika 22 Blok shema modula *Podesavanje_coef*



Slika 23 Simulacija modula *Podesavanje_coef* ModelSim simulatorom

Modul za množenje (*multiplier1*)

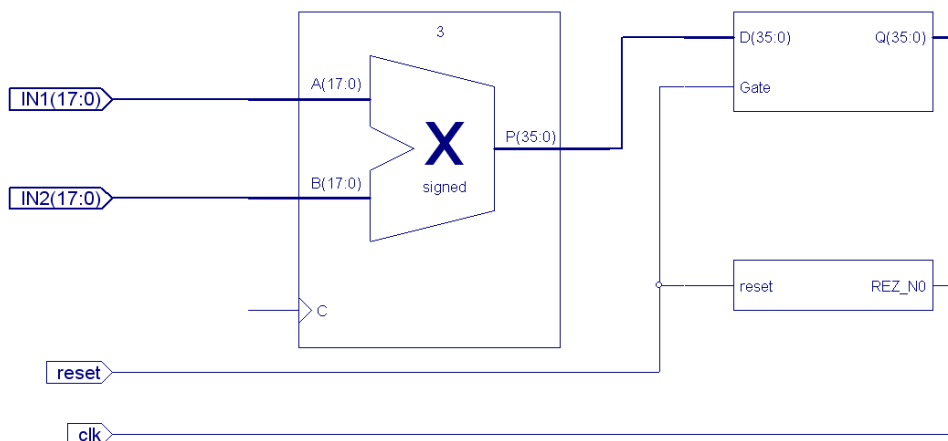
Moguće je koristiti i gotovu komponentu za sklop za množenje koja se gotovo ne razlikuje od ispogramirane komponente *Multiplier1*. Jedini razlog zašto gotova komponenta nije bila odabrana u realizaciji ovog sustava je činjenica da se kod nje ulaz „hvata“ na rastući brid takta. Kako svaki izlaz iz prethodnog sklopa postaje valjan na rastući brid takta, ali uz kašnjenje jer se radi o signalima, događa se kašnjenje izlaza od dva perioda takta u odnosu na izlaz. Svi ostali moduli su realizirani tako da daju valjan izlaz jedan period takta nakon što je pristigao valjan ulaz.

Ulazi su 17-bitni, dok je izlaz 35-bitni što se vidi na slici 24. Kako u je u sklopu potrebno množiti i signale manjih rezolucija, namještanje signala (posmak ulaza i odsijecanje izlaza na traženu rezoluciju) se vrši prije odnosno poslije faze u kojoj se odvija množenje.

```
34 entity multiplier1 is
35     port (
36         signal clk: in std_logic;
37         signal reset: in std_logic;
38         signal IN1: in std_logic_vector(17 downto 0);
39         signal IN2: in std_logic_vector(17 downto 0);
40         signal REZ: out std_logic_vector(35 downto 0)
41     );
42 end multiplier1;
```

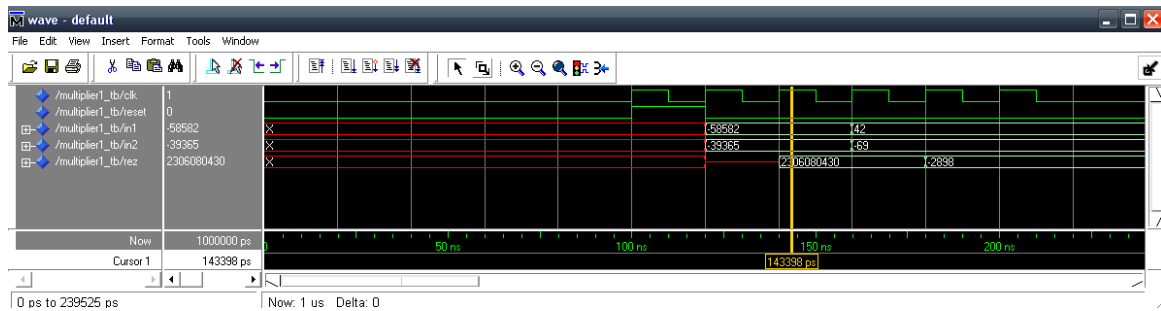
Slika 24 Entitet modula *multiplier1*

Blok shema modula dana je slikom 25. Prikazan je samo funkcijski dio, zbog preglednosti su izostavljeni bistabili koji služe za sinkronizaciju modula.



Slika 25 Blok shema funkcijskog dijela modula *multiplier1*

Kod modula koji obavljaju aritmetičke operacije korisno je provjeriti, osim sinkronizacije, i rezultat koji se dobiva na izlazu (slika 26).



Slika 26 Simulacija modula *multiplier1* ModelSim simulatorom

Modul za zbrajanje i oduzimanje (*shifter_adder*)

Modul za zbrajanje ili oduzimanje koji prije zadane operacije obavi još i posmak ulaznih signala da bi se mogli interpretirati kao dvojni komplement, a nakon operacije vrši rezanje rezultata na zadanu rezoluciju.

Kada se radi o signalima različitih rezolucija ($b1$ i $b2$ gdje je $b1 < b2$) potrebno je proširiti signal s manjom rezolucijom ($b1$) popunjavajući mjesta do MSB proširenog signala MSB-om originalnog signala. Na taj način je izvorni signal jednak originalnom i bit će pri zbrajanju interpretiran kao dvojni komplement.

Osim ulaza i izlaza, ovaj modul ima ulaze *carry_in* (kao kod potpunog zbrajala) i signal *op* koji određuje da li je potrebno pribrojiti ili oduzeti ulazne argumente.

Osim rezultata na izlaz daje i *carry_out* koji u ovom slučaju nije klasični bit preljeva kao kod potpunog zbrajala, već predstavlja prvi desni bit od najmanjeg bita rezultata nakon rezanja na željenu rezoluciju. Pomoću ovog bita se kasnije, gdje je to potrebno vrše zaokruženja rezultata.

Modul je realiziran kao generička komponenta s promjenjivim rezolucijama ulaznih i izlaznog signala.

Definicija promjenjivih parametara kao i ulaznih i izlaznih priključaka vidi se na slici 27.

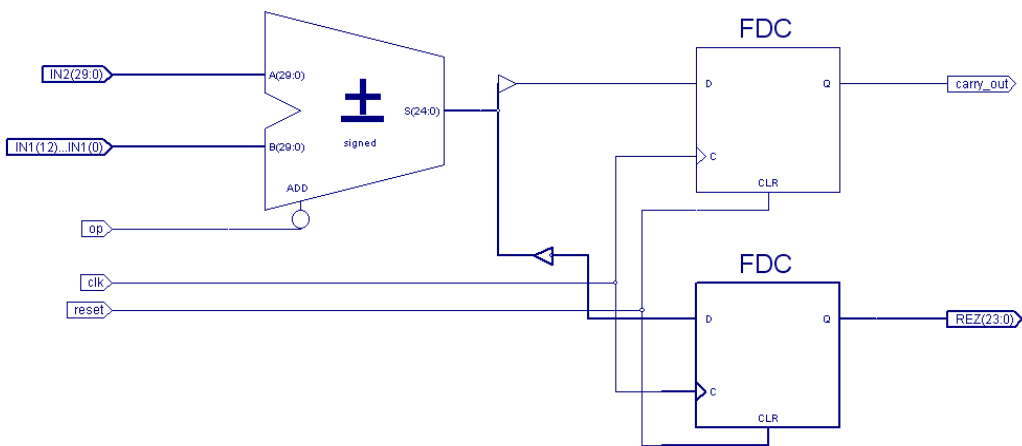
```

203 component shifter_adder is
204 generic(
205     b1, b2, bout: integer
206 );
207 port (
208     signal clk: in std_logic;
209     signal reset: in std_logic;
210     signal IN1: in std_logic_vector(b1 downto 0);
211     signal IN2: in std_logic_vector(b2 downto 0);
212     signal carry_in: in std_logic;
213     signal op: in std_logic;
214     signal carry_out: out std_logic;
215     signal REZ: out std_logic_vector(bout downto 0)
216 );
217 end component shifter_adder;
218

```

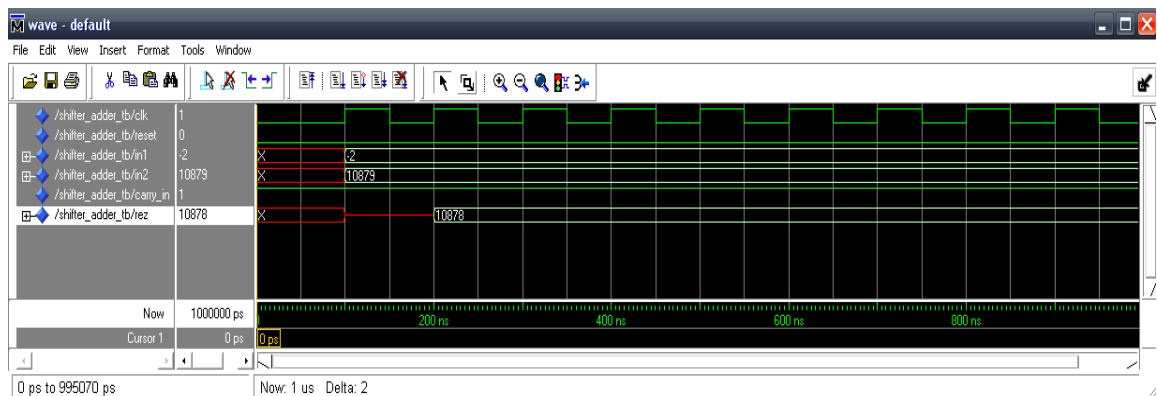
Slika 27 Entitet modula *shifter_adder*

Na slici 28 dana je blok shema modula u slučaju kada se obavlja zbrajanje parcijalni produkata m6HI i m6LO.



Slika 28 Blok shema modula *shifter_adder*

Kako se radi o jednostavnim aritmetičkim operacijama, lako je provjeriti funkcijsku ispravnost rada modula ModelSim simulatorom (slika 29).



Slika 29 Simulacija modula *shifter_adder* ModelSim simulatorom

Modul za pripremu signala prije proširenog množenja (unsig_dxb_m5)

Množenje $((d[n] \cdot \Delta xd + c[n]) \cdot \Delta xc + b[n]) \cdot \Delta xb = m_5 \cdot \Delta xb = m_6$ je kritično za realizaciju jer se radi o faktorima rezolucija 24 i 18 bitova.

Množilo koje je moguće izvesti na odabranom FPGA sklopu je maksimalno 18x18 bitova.

U sklopu je potrebno je realizirati tri množenja:

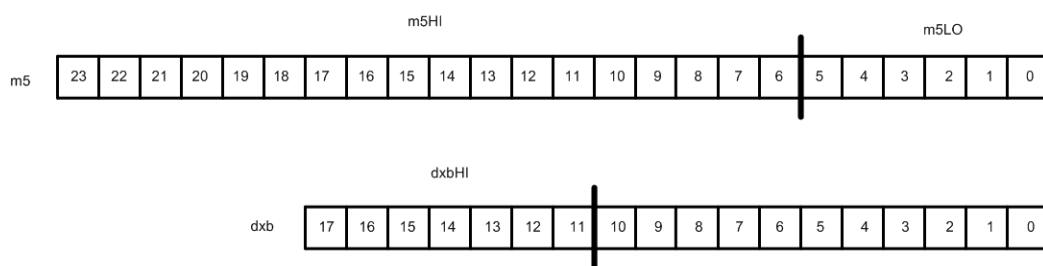
1. $d[n] \cdot \Delta xd$
2. $(d[n] \cdot \Delta xd + c[n]) \cdot \Delta xc = m_3 \cdot \Delta xc = m_4$
3. $((d[n] \cdot \Delta xd + c[n]) \cdot \Delta xc + b[n]) \cdot \Delta xb = m_5 \cdot \Delta xb = m_6$

gdje su rezolucije faktora:

1. 4 x 6 bitova
2. 15 x 17 bitova
3. 24 x 18 bitova

Množenje pod 3. realizira se rastavljanjem m_5 na dva dijela, gornjih 18 i donjih 6 bitova od kojih se oba množe sa Δxb a potom se parcijalni produkti zbroje.

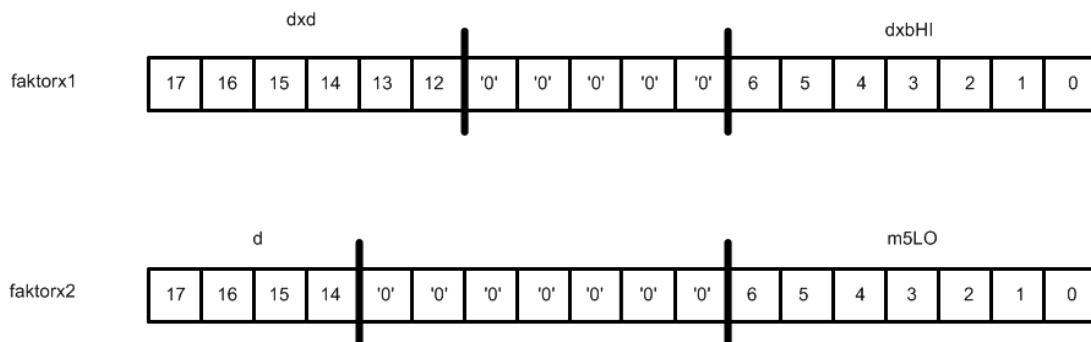
Kako se u konačnici rezultat odsijeca na gornja 24 bita, nije potrebno uzeti svih 18 bitova za drugi parcijalni produkt, jer ionako neće punom širinom utjecati na rezultat zbrajanja. Dijelovi signala koji ulaze u parcijalna množenja dani su slikom 30.



Slika 30 Razdvajanje signala prije proširenog množenja

Oba signala se odsijeku se prema slici 30 na $m5HI$, $m5LO$ i $dxbHI$.

U jednom množilu se množe $m5HI$ i dxb (rezultat je $m6HI$), dok se množenje $m5LO$ i $dxbHI$ odvija u istom množilu koje množi signale d i dxd (rezultat je $m6LO$), slika 31.



Slika 31 Raspored bitova ulaza u prvo množilo

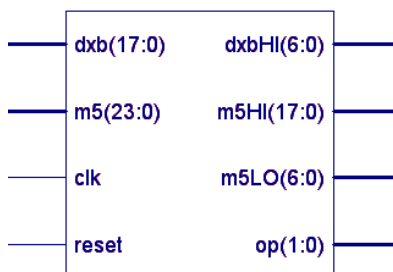
Signali *dxd* i *d* se smještaju na gornje pozicije ulaza u množilo, a *dxbHI* i *m5LO* na donje.

Rezultat ovakvog množenja će na gornjih 10 bitova (MSB se zanemari jer predstavlja „dupli predznak“) dati umnožak $dxd \cdot d$, a na donjih 12 umnožak $dxbHI \cdot m5LO$. Preostali bitovi rezultata su mješoviti produkti ova 4 faktora i nisu bitni. Potrebno je još „namjestiti“ gornji dio rezultata dodavanjem '1' na 25. bit rezultata, to se obavlja u modulu *namjesti_m2_MUXm6LO*.

Nakon obavljenih množenja parcijalni produkti (*m6HI* i *m6LO*) se zbrajaju ili oduzimaju ovisno o signalu *op*.

U ovoj fazi protočne arhitekture javlja se pogreška u odnosu na referentni kod u MATLAB-u koji množenje $((d[n] \cdot \Delta xd + c[n]) \cdot \Delta xc + b[n]) \cdot \Delta xb = m_5 \cdot \Delta xb = m_6$ pretpostavlja kao obično množenje s tim da uzima jedan bit više pri odsijecanju rezultata koji će kasnije služiti za zaokruženje.

Na slici 32 vide se ulazni i izlazni priključci modula *unsig_m5_dxb*.



Slika 32 Blok shema modula *unsig_m5_dxb*

Modul za multipleksiranje signala *m6LO* i „namještanje“ produkta *m2* (*namjesti_m2_MUXm6LO*)

Namještanje produkta *m2* je dodavanje '1' na 25. bit signala *produktx*, na taj način se poništava eventualno djelovanje mješovitih parcijalnih produkata *dxbHI*d* i *dxd*m5LO* koji su smješteni na preostale bitove signala *produktx* koje ne koristimo.

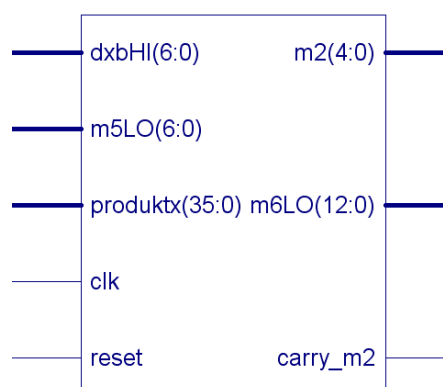
Multipleksiranje signala *m6LO* radi se da se riješi slučaj kada je *dxbHI* niz bitova „1000000“. U tom slučaju bi se pri uzimanju apsolutne vrijednosti, dogodilo da signal ostane isti (negacija i dodavanje '1') jer je signal reprezentiran kao dvojni komplement. Potrebno je dobiti broj 1 da bi s njim pomnožili *m5LO* a kako se u dvojnem komplementu s logikom fiksnog zareza ne može zapisati broj 1, rješenje je jednostavno propuštanje signala *m5LO* kao rezultat množenja.

Multipleksor dakle služi za odabir *m6LO*, što se i vidi na slici 33.

```
if dxbHI="1000000" then
  m6LO_pom(12 downto 6) :=m5LO;
  m6LO_pom(5 downto 0) :=(others=>'0');
else
  m6LO_pom:='0' &produktx(11 downto 0);
end if;
```

Slika 33 Dio koda modula *namjesti_m2_MUXm6LO* koji obavlja multipleksiranje signala *m6LO*

Na slici 34 vide se ulazni i izlazni priključci modula i njihove rezolucije.



Slika 34 Blok shema modula *namjesti_m2_MUXm6LO*

Moduli za kašnjenje (*registar1* i *registar*)

Neke signale u sklopu potrebno je zakasniti da bi se realizirala protočna arhitektura.

Za ovaj sustav realizirana su 2 modula za kašnjenje, *registar1* služi da bi se signal zakasnio za jedan period takta, a *registar* ako je potrebno signal zakasniti za 2 ili više perioda takta.

Kako se radi o signalima različitih rezolucija i različitog vremena kašnjenja, moduli su realizirani kao generički, s promjenjivim parametrima b , odnosno N i b , gdje je $N+1$ broj perioda signala takta za koje signal mora biti zakašnjen, a b rezolucija signala. Kod modula *registar1* kašnjenje je fiksno jedan period takta, pa ne postoji parametar N .

Definicija promjenjivih parametara i priključaka prikazana je na slikama 36 i 37.

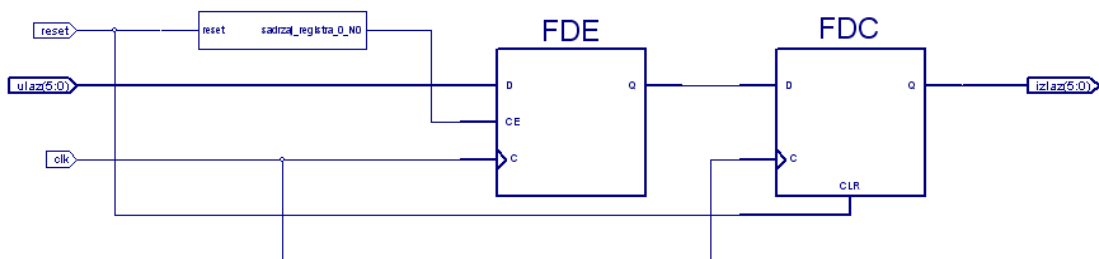
```
31 entity registar1 is
32   generic(
33     b: integer
34   );
35   port (
36     signal clk: in std_logic;
37     signal reset: in std_logic;
38     signal ulaz: in std_logic_vector(b downto 0);
39     signal izlaz: out std_logic_vector(b downto 0)
40   );
41
42 end registar1;
```

Slika 35 Entitet modula *registar1*

```
31 entity registar is
32   generic(
33     N : integer;
34     b: integer
35   );
36   port (
37     signal clk: in std_logic;
38     signal reset: in std_logic;
39     signal ulaz: in std_logic_vector(b downto 0);
40     signal izlaz: out std_logic_vector(b downto 0)
41   );
```

Slika 36 Entitet modula *registar*

Blok shema modula za signal *dxd* koji je potrebno zakasniti 2 perioda takta prikazan je slikom 37.



Slika 37 Blok shema modula *register* za signal *dxd*

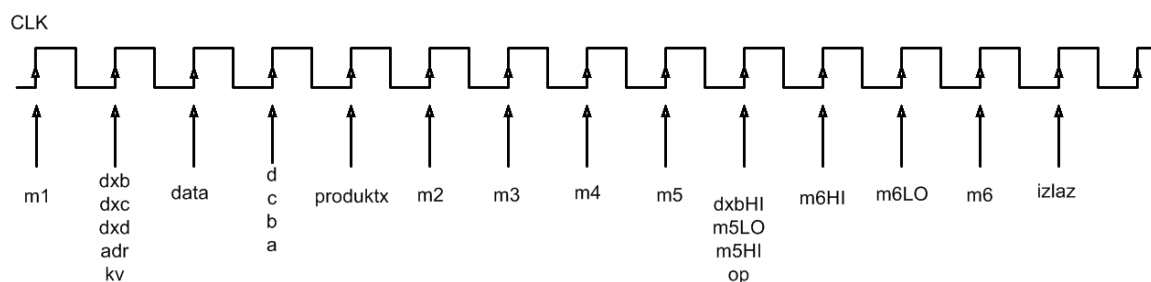
4.4 SPAJANJE MODULA U DDFS

4.4.1 Određivanje kašnjenja pojedinih signala u sustavu

Moduli se spajaju prema blok shemi na slici 8 iz poglavlja 5.2.

Na blok shemi nije prikazano spajanje modula za kašnjenje.

Za određivanje vremena kašnjenja pojedinih signala poslužit će slika 9 iz poglavlja 5.2 (ovdje slika 39).



Slika 38 Propagacija signala kroz sustav

Da bi se u prvom množilu obavilo množenje signala $dxd*d$, signal dxd potrebno je zakasniti za 2 perioda takta, tako da na ulazu u množilo istovremeno dođu $dxd[n]$ i $d[n]$. U tablici dana su kašnjenja svih signala u sustavu i njihove rezolucije (stvarna rezolucija je $b+1$, indeksi bitova počinju od 0).

SIGNAL	N+1	b
dxd	2	5
kv	1	1
c	2	5
dxc	5	16
b	4	23
dxb	7	17
predznak	1	1
a	8	31

Tabela 4 Tablica potrebnih kašnjenja svih signala u sustavu

U nastavku je dan prikaz spajanja registara u DDFS u VHDL-u.

```

296 -----elementi za kašnjenje pojedinih signala-----
297 Reg_kvadrant_1:      component registar1
298                     generic map (b=>1)
299                     port map (clk=>clk, reset=>reset, ulaz=>kv, izlaz=>kv_t);
300
301 Reg_dxd_2:          component registar
302                     generic map (N=>1, b=>5)
303                     port map (clk=>clk, reset=>reset, ulaz=>dxd, izlaz=>dxd_t);
304
305 Reg_c_2:            component registar
306                     generic map (N=>1, b=>14)
307                     port map (clk=>clk, reset=>reset, ulaz=>c, izlaz=>c_t);
308
309 Reg_dxc_5:          component registar
310                     generic map (N=>4, b=>16)
311                     port map (clk=>clk, reset=>reset, ulaz=>dxc, izlaz=>dxc_t);
312
313 Reg_b_4:            component registar
314                     generic map (N=>3, b=>23)
315                     port map (clk=>clk, reset=>reset, ulaz=>b, izlaz=>b_t);
316
317 Reg_dxb_7:          component registar
318                     generic map (N=>6, b=>17)
319                     port map (clk=>clk, reset=>reset, ulaz=>dxb, izlaz=>dxb_t);
320
321 Reg_dxb_8:          component registar
322                     generic map (N=>7, b=>17)
323                     port map (clk=>clk, reset=>reset, ulaz=>dxb, izlaz=>dxb_t2);
324
325 Reg_dxbHI_1:        component registar1
326                     generic map (b=>6)
327                     port map (clk=>clk, reset=>reset, ulaz=>dxbHI, izlaz=>dxbHI_t);
328
329 Reg_m5LO_1:         component registar1
330                     generic map (b=>6)
331                     port map (clk=>clk, reset=>reset, ulaz=>m5LO, izlaz=>m5LO_t);
332
333 Reg_m6HI_1:         component registar1
334                     generic map (b=>29)
335                     port map (clk=>clk, reset=>reset, ulaz=>m6HI, izlaz=>m6HI_t);
336
337 Reg_a_9:            component registar
338                     generic map (N=>8, b=>31)
339                     port map (clk=>clk, reset=>reset, ulaz=>a, izlaz=>a_t);
340
341 Reg_op_2:           component registar
342                     generic map (N=>1, b=>1)
343                     port map (clk=>clk, reset=>reset, ulaz=>op, izlaz=>op_t);

```

Slika 39 Dio koda DDFS-a u VHDL-u koji opisuje spajanje signala module za kašnjenje
Zakašnjeni signali imaju nastavak „_t“, tako da se razlikuju od originalnih signala.

4.4.2 Spajanje modula i provjera ispravnosti izlaza

Nakon spajanja pojedinog modula u sustav provjerava se podudarnost izlaza modula s izlazima generiranim pomoću referentnog koda u MATLAB-u.

Na taj način u hodu se izbjegavaju pogreške u spajanju modula, koje bi kasnije bilo puno teže pronaći.

Početne vrijednosti signala u MATLAB-u i odgovarajući signali u *DDFS-u*:

poc = 266904163 → „1111111010001010001001100011“ → *pocetna_faza*

kor = 5662074 → „0000010101100110010101111010“ → *dfaza*

Simulira se izlaz u 4096 uzoraka.

U nastavku su dani dijelovi koda koji pokazuju definiranje entiteta *DDFS-a* i spajanje modula *phase_accumulator*, *podesenja*, *ROM* i *Podesavanje_coef*.

```
30 ▶entity DDFS is
31
32     port (
33         signal clk: in std_logic;
34         signal reset: in std_logic;
35         signal pocetna_faza: in std_logic_vector (27 downto 0);
36         signal dfaza: in std_logic_vector (27 downto 0);
37         signal sinus: out std_logic_vector (31 downto 0)
38     );
39
40 end DDFS;
```

Slika 40 Definiranje entiteta *DDFS-a*

```

227 begin
228
229
230 Akumulator:          component phase_accumulator
231                      port map(clk => clk,
232                                reset => reset,
233                                pocetna_faza => pocetna_faza,
234                                dfaza => dfaza,
235                                faza => m1);
236
237 Podesavanje_ulaza:   component podesenja
238                      port map(clk => clk,
239                                reset => reset,
240                                ulaz => m1,
241                                adr => adr,
242                                kv => kv,
243                                dxk => dxk,
244                                dxc => dxc,
245                                dxd => dxd);
246
247 ROM_data:           component ROM
248                      port map(clk => clk,
249                                reset => reset,
250                                adr => adr,
251                                data => data);
252
253 Podesavanje_koeficijenata: component Podesavanje_coef
254                      port map(clk => clk,
255                                reset => reset,
256                                kv => kv_t,
257                                ulaz => data,
258                                a => a,
259                                b => b,
260                                c => c,
261                                d => d);
262

```

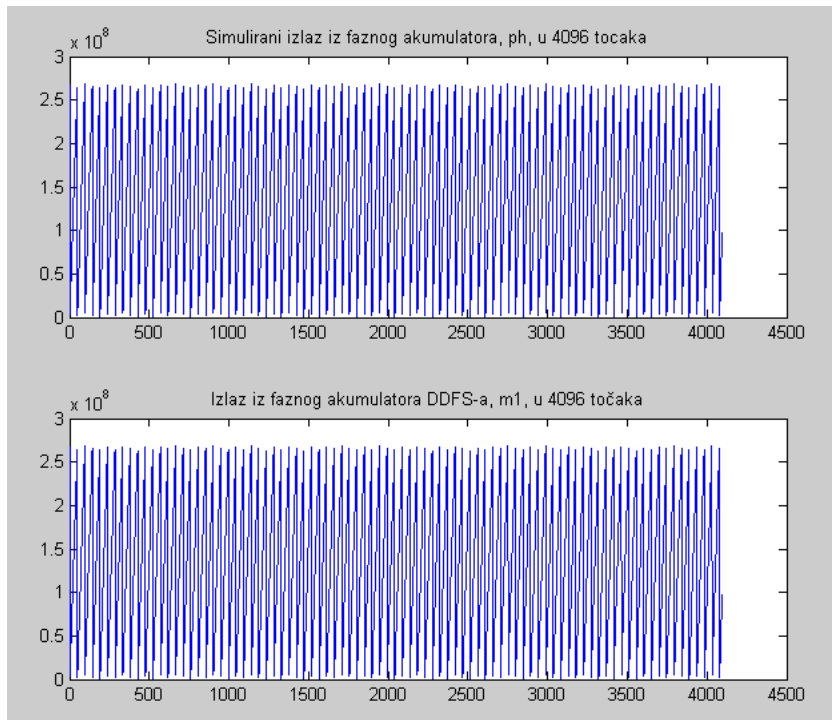
Slika 41 Spajanje pojedinih modula u *DDFS*

Na slikama 43 – 45 dani su grafički prikazi signala *m1* na izlazu modula *phase_accumulator*, signala *ph* koji je izgeneriran pomoću MATLAB-a i njihova razlika. Prije usporedbe ovih dvaju signala, potrebno ih je „namjestiti“ tako da se uspoređuju odgovarajući uzorci. Kod koda u MATLAB-u protočna arhitektura je izvedene u manje stupnjeva, pa na to treba pripaziti pri usporedbi.

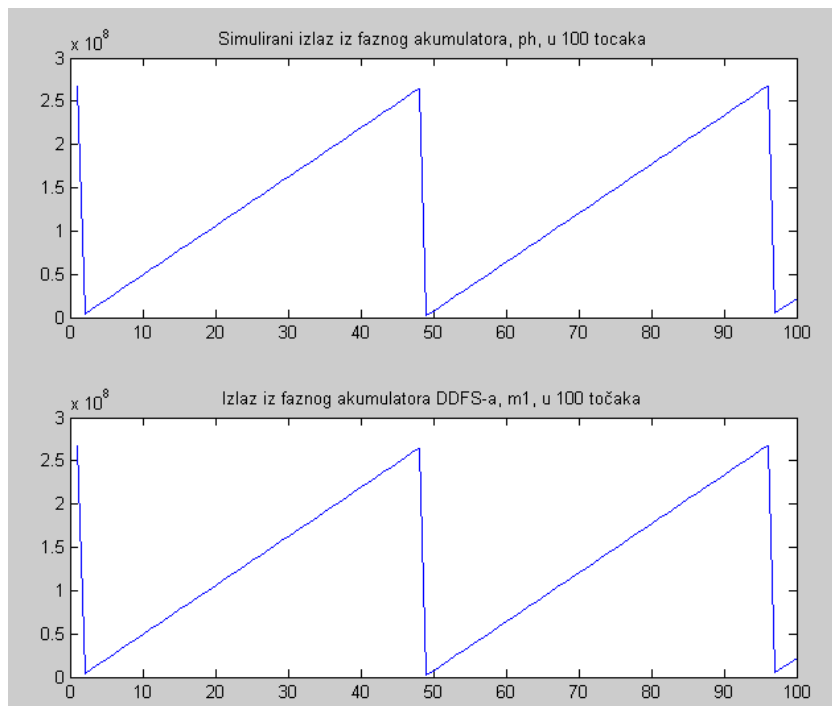
Na isti način provjerava se podudarnost ostalih signala u *DDFS-u* s odgovarajućim signalima dobivenim pomoću MATLAB-a.

Nema potrebe za ovako detaljnim prikazom preostalih signala iako je za svakog na ovaj način provjerena ispravnost.

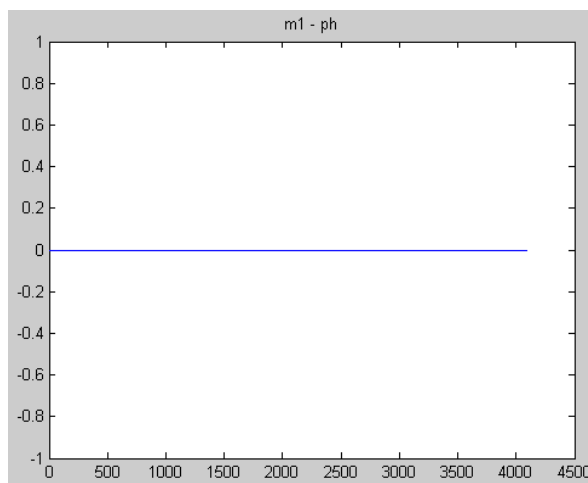
Bitno je da se svaki sljedeći modul neće spojiti u *DDFS* dok prethodni nije dao potpuno podudaran izlaz sa signalom iz MATLAB-a.



Slika 42 Signali ph , $m1$ prikazani pomoću MATLAB-a u 4096 točaka



Slika 43 Signali ph i $m1$ prikazani pomoću MATLAB-a u 100 točaka



Slika 44 Razlika signala *m1* i *ph* prikazana pomoću MATLAB-a u 4096 točaka

Izlazi pojedinih modula ne ulaze u sljedeći modul u punoj rezoluciji, već ih je potrebno „odrezati“, pa se to učini prije spajanja modula u *DDFS*.

Na slikama 45 – 53 vidi se spajanje preostalih modula u *DDFS* i priprema pojedinih signala prije dovođenja na ulaz modula.

```

300
301 -----prošireno množenje-----
302 faktorx1(17 downto 12) <= dxd_t;
303 faktorx1(6 downto 0) <= dxbHI;
304 faktorx2(17 downto 14) <= d;
305 faktorx2(6 downto 0) <= m5LO;
306
307 Množenje_dxd_d_i_dxb_m5_lo: component multiplier1
308                               port map(clk => clk,
309                                         reset => reset,
310                                         IN1 => faktorx1,
311                                         IN2 => faktorx2,
312                                         REZ => produktx);
313 -----
314
315

```

Slika 45 Priprema signala *faktorx1* i *faktorx2* prije ulaza u prvi *multiplier1* i njegovo spajanje u *DDFS*

```

366
367 -----namjestanje proširenog produkta i multipleksiarnje m6LO-----
368 Namjestanje_produktxHI: component namjesti_m2_MUXm6LO
369                               port map (clk => clk,
370                                         reset => reset,
371                                         produktx => produktx,
372                                         m5LO => m5LO_t,
373                                         dxbHI => dxbHI_t,
374                                         m2 => m2,
375                                         carry_m2 => carry_m2,
376                                         m6LO => m6LO);
377 -----
378

```

Slika 46 Spajanje modula *namjesti_m2_MUXm6LO* u *DDFS*


```

380
381 -----zbrajanje m2 i c-----
382 Zbrajanje_m2_c:      component shifter_adder
383                     generic map (b1=>4, b2=>14, bout=>14)
384                     port map (clk => clk,
385                               reset => reset,
386                               IN1 => m2,
387                               IN2 => c_t,
388                               carry_in => carry_m2,
389                               op => '0',
390                               carry_out => carry_out,
391                               REZ => m3);
392 -----
393

```

Slika 47 Zbrajanje signala *m2* i *c_t*

```

346
347 -----mnozenje m3 i dxc-----
348 IN1_dxc_t(16 downto 0) <= dxc_t;
349 IN1_dxc_t(17) <= dxc_t(16);
350 IN2_m3(14 downto 0) <= m3;
351 IN2_m3(17 downto 15) <= (others => m3(14));
352
353 Mnozenje_dxc_m3:      component multiplier1
354                     port map (clk => clk,
355                               reset => reset,
356                               IN1 => IN1_dxc_t,
357                               IN2 => IN2_m3,
358                               REZ => m4_pr);
359 -----
360

```

Slika 48 Priprema signala *m3* i *dxc_t* i njihovo množenje

```

411
412 -----zbrajanje m4 i b-----
413 m4 <= m4_pr(30 downto 16);
414 carry_m4 <= m4_pr(15);
415 Zbrajanje_m4_b:      component shifter_adder
416                     generic map (b1=>14, b2=>23, bout=>23)
417                     port map (clk => clk,
418                               reset => reset,
419                               IN1 => m4,
420                               IN2 => b_t,
421                               carry_in => carry_m4,
422                               op => '0',
423                               carry_out => carry_out1,
424                               REZ => m5);
425 -----

```

Slika 49 Priprema signala *m4* i zbrajanje signala *m4* i *b_t*

```

428
429 -----priprema faktora za parcijalno mnozenje-----
430 Unsig_dxbHI_i_m5_lo:  component unsig_m5_dxb
431                     port map (clk => clk,
432                               reset => reset,
433                               dxb => dxb_t,
434                               m5 => m5,
435                               dxbHI => dxbHI,
436                               m5LO => m5LO,
437                               m5HI => m5HI,
438                               op => op);
439 -----
440

```

Slika 50 Spajanje modula *unsig_m5_dxb* u *DDFS*

```

392
393 -----mnozenje gornjeg dijela m5 i dxb-----
394 Mnozenje_dxb_m5HI:      component multiplier1
395                          port map(clk => clk,
396                                  reset => reset,
397                                  IN1 => dxb_unsig,
398                                  IN2 => m5_unsig,
399                                  REZ => m6HI_pr);
400 -----
401

```

Slika 51 Dobivanje gornjeg parcijalnog produkta *m6HI*

```

453
454 -----zbrajanje m6 parcijalnih produkata-----
455 m6HI <= m6HI_pr(34 downto 5);
456 Zbrajanje_m6HI_m6LO:   component shifter_adder
457                          generic map(b1=>12, b2=>29, bout=>23)
458                          port map(clk => clk,
459                                  reset => reset,
460                                  IN1 => m6LO,
461                                  IN2 => m6HI_t,
462                                  carry_in => '0',
463                                  op => op_t(0),
464                                  carry_out => carry_m6,
465                                  REZ => m6);
466 -----
467

```

Slika 52 Zbrajanje parcijalnih produkata *m6HI* i *m6LO*

```

469
470 -----zbrajanje m6 i a-----
471 Zbrajanje_m6_a:        component shifter_adder
472                          generic map(b1=>23, b2=>31, bout=>31)
473                          port map(clk => clk,
474                                  reset => reset,
475                                  IN1 => m6,
476                                  IN2 => a_t,
477                                  carry_in => carry_m6,
478                                  op => '0',
479                                  carry_out => carry_out2,
480                                  REZ => sinus);
481 -----

```

Slika 53 Zbrajanje signala *m6* i *a_t*

Ovime je završeno definiranje referentnog modela *DDFS* u VHDL-u. Sinteza i implementacija sustava bit će objašnjenje u sljedećem poglavlju.

4.5 SINTEZA I IMPLEMENTACIJA SKLOPA

Nakon realizacije referentnih i RTL modela svih modula i njihovog spajanja potrebno je još sintetizirati i implementirati *DDFS*.

Xilinx iz VHDL koda prepoznaje korištene komponente, obavlja optimizacije pri fizičkom razmještanju pojedinih komponenti na odabrano FPGA sklopovlje, povlači vodove pojedinih signala i računa vremenske odnose signala.

The screenshot displays the Xilinx Project Navigator interface for a design named 'ddfs'. The 'Sources in Project' pane on the left shows a hierarchical tree of VHDL files, including behavioral models for multipliers, registers, and ROMs. The 'Processes for Source' pane shows the compilation and implementation steps, with 'Place & Route' and 'Generate Programming File' completed. The main area shows the 'Design Overview for ddfs' with the following data:

Property	Value
Project Name:	d:\fer\diplomski\ddfs_final
Target Device:	xc3s200
Report Generated:	Tuesday 10/06/09 at 20:12
Printable Summary (View as HTML):	ddfs_summary.html

The 'Device Utilization Summary' table provides a detailed breakdown of resource usage:

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops:	558	3,840	14%	
Number of 4 input LUTs:	327	3,840	8%	
Logic Distribution:				
Number of occupied Slices:	393	1,920	20%	
Number of Slices containing only related logic:	393	393	100%	
Number of Slices containing unrelated logic:	0	393	0%	
Total Number 4 input LUTs:	437	3,840	11%	
Number used as logic:	327			
Number used as a route-thru:	12			
Number used as Shift registers:	98			
Number of bonded IOBs:	90	173	52%	
Number of Block RAMs:	1	12	8%	
Number of MULT18X18s:	3	12	25%	
Number of GCLKs:	1	8	12%	

The 'Performance Summary' indicates that all signals are completely routed and no constraints are failing. The 'Detailed Reports' table lists the following reports:

Report Name	Status	Last Date Modified
Synthesis Report	Current	Tuesday 10/06/09 at 20:11
Translation Report	Current	Tuesday 10/06/09 at 20:11
Map Report	Current	Tuesday 10/06/09 at 20:11
Pad Report	Current	Tuesday 10/06/09 at 20:12
Place and Route Report	Current	Tuesday 10/06/09 at 20:12
Post Place and Route Static Timing Report	Current	Tuesday 10/06/09 at 20:12

The bottom status bar shows 'Speed Grade: -4' and a console area with 'Find in Files', 'Errors', and 'Warnings' buttons.

Slika 54 Sažete informacije o implementiranom sklopu generirane *Xilinx*-om

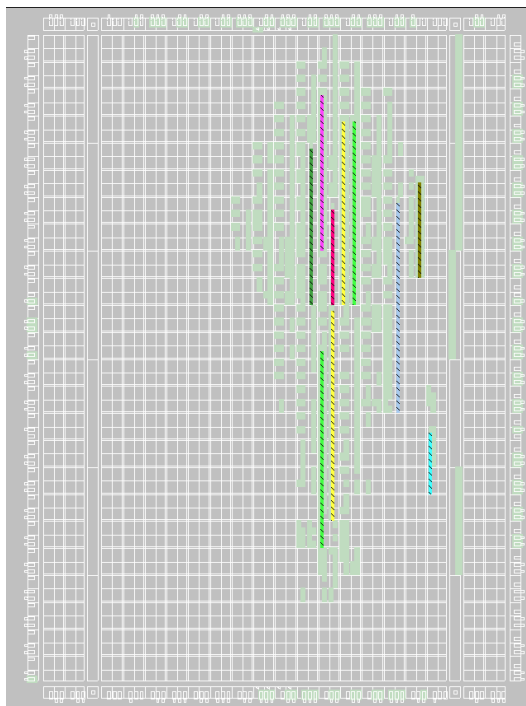
Potpune informacije o realiziranom sklopu vidljive su iz izvješća koja *Xilinx* generira tokom sinteze i implementacije sklopa:

- *Synthesis Report* – Popis svih prepoznatih logičkih primitiva (registri, zbrajala, množila, bistabili...), uklanjanje ponovljenih signala/bitova, optimizacija modula, vremenska analiza (maksimalna frekvencija), greške sustava, upozorenja, informacije
- *Translation Report*
- *Map Report* – Razmještaj ulaznih i izlaznih signala u IOB blokove
- *Pad Report* – Popis pinova koji će se koristiti i njihovo pridjeljivanje ulaznim i izlaznim signalima
- *Place and Route Report* – informacije o razmještanju komponenti i povlačenju vodova signala
- *Post Place and Route Report* – vremena držanja i postavljanja (t_{hold} t_{setup}) ulaznih i izlaznih signala u odnosu na aktivni brid takta

Sažetak informacija o implementiranom dizajnu može se vidjeti na slici 54.

Može se vidjeti stvarni razmještaj komponenti na FPGA sklopu pomoću opcije *View/Edit Placed design (Floorplanner)*.

Na slici 55 vide se područja na koje su razmještene komponente, i iskorišteni IOB blokovi.



Slika 55 *Floorplanner* realiziranog *DDFS-a*

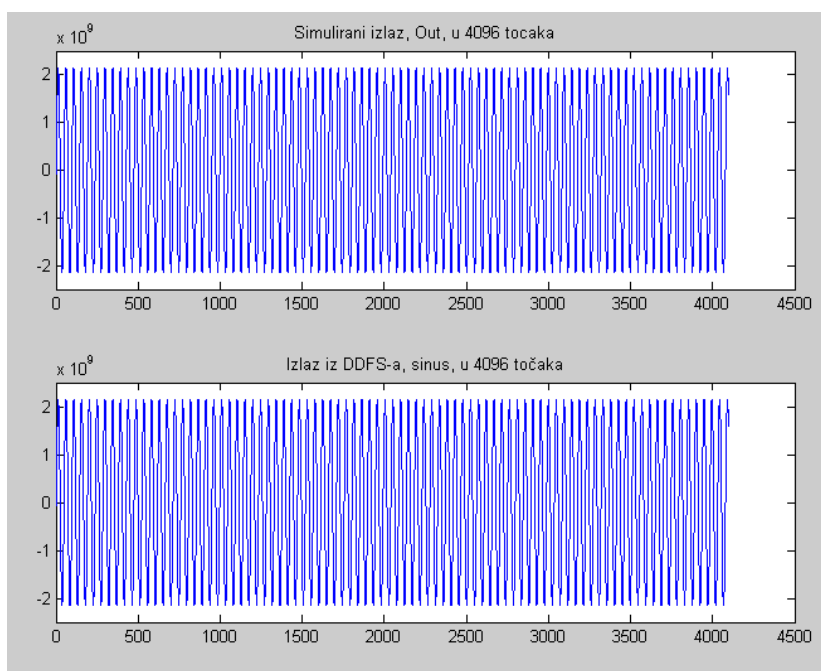
Vidljivo je iz prikaza 55 da je odabrani FPGA sklop relativno malo iskorišten ako se uzmu u obzir raspoloživi logički resursi. Kombinajska logika, koja se pohranjuje u LUT tablice zauzima samo 14% raspoloživih tablica, iskorištena su 3 množila od mogućih 12 i 1 blok RAM (od 12 raspoloživih).

S druge strane, ako se pogledaju IOB blokovi, čak više od 50% ih je iskorišteno za implementaciju dizajna. Različite realizacije dale bi, naravno različitu statistiku vezano za logičke dijelove sklopa, ali broj iskorištenih IOB blokova ne može se smanjiti jer su zadane fiksne rezolucije izlaznih i ulaznih signala.

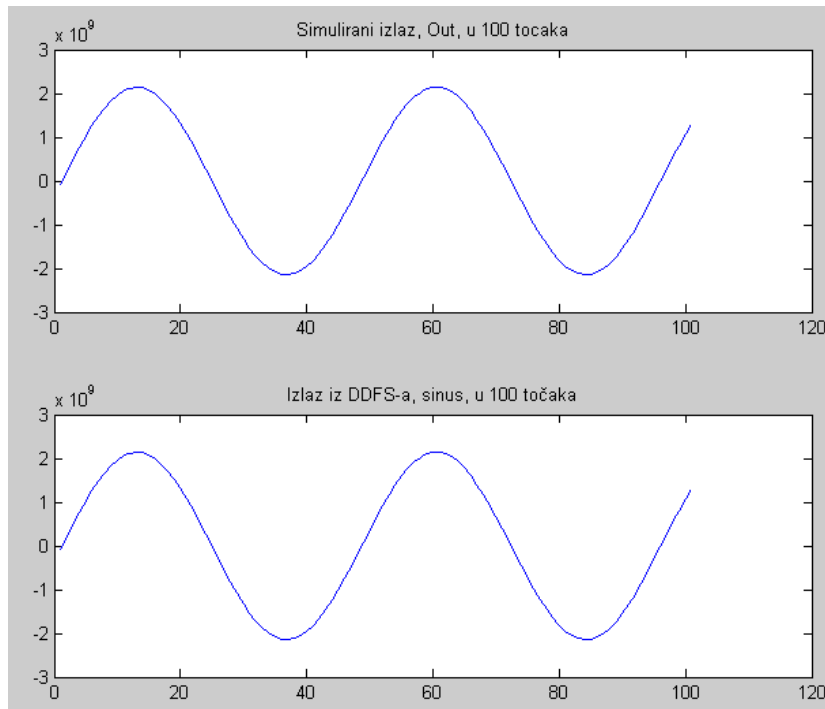
5. Analiza rada sklopa i rasprava o zadovoljenju zahtjeva postavljenih na sklop

Usporedbom izlaza realiziranog sklopa sa izlazom dobivenim u MATLAB-u vide se eventualna odstupanja od referentnog koda koji simulira ponašanje sklopa prema ovom modelu PSAC-a, dok se usporedbom sa idealnom sinusoidom također izgeneriranom u MATLAB-u vide stvarni performansi realiziranog *DDFS-a*.

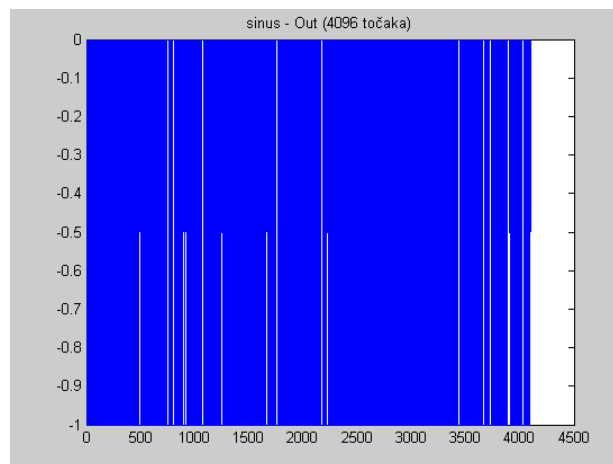
Slike 57-59 prikazuju signale *sinus* (32-bitni izlaz *DDFS-a*) i *Out* („32-bitni“ signal dobiven MATLAB-om, odgovara signalu *sinus* u protočnoj arhitekturi).



Slika 56 Signali *Out* i *sinus* u 4096 točaka, prikazani pomoću MATLAB-a



Slika 57 Signali Out i sinus u 100 točaka, prikazani pomoću MATLAB-a



Slika 58 Prikaz razlike signala Out i sinus, dobiven pomoću MATLAB-a

Razlika je 1 LSB izlaznog signala.

Do ove razlike došlo je isključivo zbog realizacije proširenog množenja signala dx_b i m_5 . Simulacija sklopa MATLAB-om ovu fazu protočne arhitekture pretpostavlja kao obično množenje s jednim bitom proširenja i kasnije zaokruženje rezultata pri zbrajanju m_6 s koeficijentom a .

Zbog odstupanja od referentnog koda učinjene su izmjene u kodu koje točno simuliraju fazu protočne arhitekture u kojoj se množe signali m_5 i dx_b kako je to objašnjeno u poglavlju 5.2, uzimajući u obzir razdvajanje m_5 na dva dijela

rezolucija 18 (viših) i 6 (nižih) bitova i parcijalno množenje s cijelim odnosno samo višim dijelom (gornjih 6 bitova) dx_b .

Rezultat izmijenjenog koda u potpunosti se podudara s izlazom *sinus* realiziranog sklopa.

Tablica 5 daje usporedbu referentnog i izmijenjenog koda u odnosu na pogreške naspram idealne sinusoide.

	Maksimalna apsolutna pogreška (u LSB)	Standardna devijacija pogreške (u LSB)	Maksimalna spektralna pogreška (u dB)	Maksimalna pogreška testnog signala (u LSB)
REFERENTNI KOD	0.7571	0.1351	-209.3116	0.6864
IZMIJENJEN KOD	0.8260	0.1459	-194.5611	0.6864

Tabela 5 Usporedba pogreški referentnog i izmijenjenog koda u odnosu na idealnu sinusoidu

Dio koda u MATLAB-u koji simulira realizaciju protočne arhitekture dan je u Dodatku 1, a izmjene koda u Dodatku 2.

Može se već sad pretpostaviti da realizirani sklop, iako je u granicama tolerancije zadanim u diplomskom zadatku (točnost od barem 30 bita), neće dati rezultat jednako precizan kao i simulirani sklop.

Uspoređuje se vrijednost $rez = \sinus * 2^{-31}$ sa idealnom sinusoidom y_s nacrtanom u MATLAB-u. Pogreška izlaza *DDFS-a* vidi se na slikama 60 i 61.

Maksimalne pogreške testnog sinusnog signala dobivenog *DDFS-om* su:

- Maksimalna apsolutna pogreška *rez*: $1.2785 * 10^{-9}$
- Maksimalna pogreška u LSB pri rezoluciji 1.29 bitova: 0.6864

Zadovoljena je željena preciznost realiziranog sinusnog signala u odnosu na testni signal.

Osim preciznosti, bitan pokazatelj performansi realiziranog *DDFS-a* je maksimalna brzina koja se može postići.

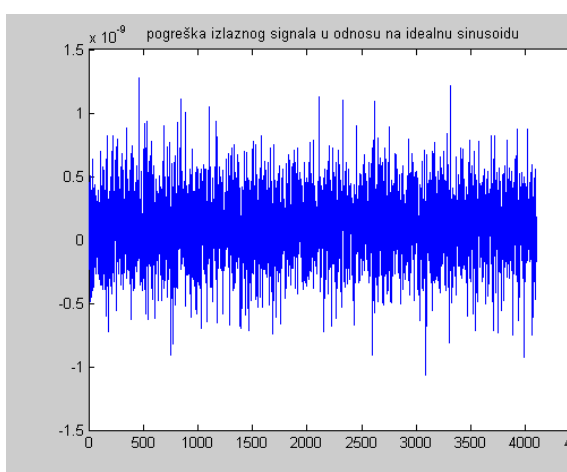
Rezultati su izračunati u *Xilinxu* i vide se na slici 60.

```

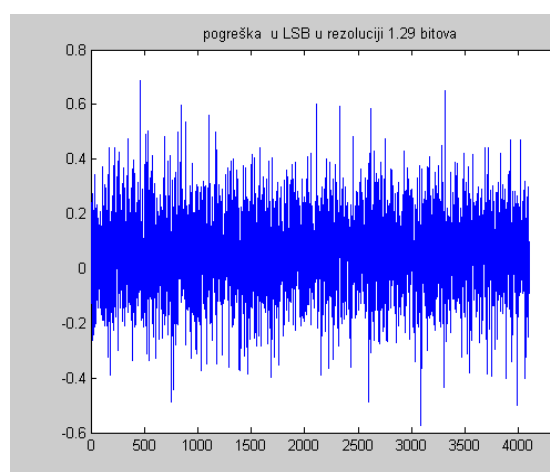
Timing Summary:
-----
Speed Grade: -4

Minimum period: 9.235ns (Maximum Frequency: 108.284MHz)
Minimum input arrival time before clock: 8.957ns
Maximum output required time after clock: 7.165ns
Maximum combinational path delay: No path found
    
```

Slika 59 Vremenske karakteristike realiziranog *DDFS-a*



Slika 60 Razlika idealne sinusoide i izlaza *DDFS-a*



Slika 61 Pogreška *DDFS-a* u LSB kod rezolucije 1.29 bitova

Maksimalna frekvencija koja se može postići je 108.284 MHz, što je relativno zadovoljavajuće. Danas je zahtjev kod većine sustava što veća frekvencija rada sklopovlja.

Ako bi *DDFS* sadržavao samo fazni akumulator maksimalna frekvencija rada sklopa je ≈ 180 MHz, što znači da pojedini dijelovi protočne arhitekture rade sporije od ostalih, pa ima smisla pokušati ih drugačije realizirati u svrhu eventualnog povećanja brzine rada sklopa.

```

Timing Summary:
-----
Speed Grade: -4

Minimum period: 5.614ns (Maximum Frequency: 178.126MHz)
Minimum input arrival time before clock: 5.783ns
Maximum output required time after clock: 7.165ns
Maximum combinational path delay: No path found
    
```

Slika 62 Vremenske karakteristike *DDFS-a* koji se sastoji samo od faznog akumulatora

Teoretski je maksimalna frekvencija sinusnog signala koji se može dobiti na izlazu *DDFS-a* određene ulaznim signalom *dfaza*, koji određuje udaljenost između dva izgenerirana uzorka signala.

Puna širina rezolucije ulaznog signala iznosi 1.27 bitova, a oni predstavljaju fazu u rasponu $[0, 2\pi)$. Minimalni pribroj faze odnosno razlika između dva susjedna kuta za koji se računa vrijednosti sinusa je:

$$\Delta\varphi_{\min} = \frac{2\pi}{2^{28}} \text{ rad} = \frac{2\pi}{268435456} \text{ rad} = 2.34 \cdot 10^{-8} \text{ rad}$$

Poznati teorem o otipkavanju i rekonstruiranju otipkanog signala je Nyquist – Shannon-ov teorem koji određuje minimalnu frekvenciju otipkavanja signala, koja omogućuje interpolaciju i rekonstrukciju signala bez aliasinga:

$$f_{s \min} > \frac{f_{MAX}}{2}, \text{ ovo daje teoretski maksimalnu frekvenciju sinusnog signala koji se}$$

može dobiti na izlazu sklopa.

Maksimalna frekvencija sinusne funkcije koja se može rekonstruirati iz izlaznih uzoraka *DDFS-a* bez aliasinga određena je brojem izgeneriranih uzoraka po periodu signala.

Za sinusnu funkciju taj broj teoretski iznosi 2 uzorka / periodu signala, ukoliko ne postoji DC komponenta signala.

Ako se sad uzme u obzir i maksimalna brzina generiranja uzoraka od oko 108 MHz, dobije se maksimalna frekvencija signala koji se može generirati:

$$2 \text{ uzorka} \rightarrow 2 \text{ perioda signala takta} \rightarrow 2 * 9.235 \text{ ns} \rightarrow 1 \text{ period sinusnog signala}$$
$$f_{\text{sinusMAX}} \approx 54 \text{ MHz.}$$

Jasno je da će mali broj uzoraka po periodu sinusnog signala najobičnijom linearnom interpolacijom dati izlomljen signal (trokutasti, trapezasti...), ali ako se uzorci naknadno propuštaju kroz *sinc* ili neki drugi odgovarajući interpolator, dobit će se jednoznačna sinusna funkcija.

6. Zaključak

Idealni sinusni signal neizostavan je u mnogim primjenama kao referentni signal s kojim se vrše usporedbe testiranog signala s aspekta frekvencijskih ili vremenskih karakteristika signala (medicinska mjerenja, osciloskopi). Služi i kao nosilac kod moduliranih signala, a takvim principom se danas realizira gotovo svaki prijenos signala s kojima se susrećemo svaki dan (radio, TV, digitalne komunikacija).

Potpuno idealan signal nemoguće je realizirati, ali ulažu se veliki naponi da se dobije što preciznija sinusoida pomoću što jednostavnijih sklopovlja.

U okviru ovog diplomskog rada realiziran je generator sinusnog signala na odabranom FPGA sklopovlju, s naglaskom na preciznost izlaznog signala, što jednostavnije sklopovlje i maksimalnu brzinu rada.

Odabrana je jedna od mnoštva matematičkih ideja i razmatranja dobivanja sinusnog signala na osnovu zadane promjene faze.

Rezultat je zadovoljavajući s aspekta preciznosti dobivenog signala.

Zadovoljavajući, ali ne i optimalan u odnosu na referentni kod, stoga zahtjeva daljnji rad na poboljšanju i optimizaciji ovakve realizacije.

Fizička realizacija sklopa je jednostavna, što je jedan od bitnih zahtjeva, iako ni tu nema razloga za potpuno zadovoljstvo. Optimalnijim razmještajem komponenti na odabrano sklopovlje i pojednostavljivanjem pojedinih dijelova sklopa i tu se da učiniti uštede i ubrzanje rada sklopa.

Brzina je jedna od bitnih karakteristika opisane izvedbe sklopa. Iako je maksimalna frekvencija zadovoljavajućih 108.284 MHz ipak bi se moglo još ubrzati rad sklopa i to na više načina: odabirom što više gotovih komponenti koje su ionako raspoložive na odabranom FPGA sklopovlju, pažljivim razmještajem komponenti na FPGA sklopovlje, pažljivim projektiranjem pojedinih modula i razdvajanjem sklopovlja na još više faza protočne arhitekture, što bi u ovom slučaju bilo potpuno primjenjivo rješenje uz uvjetno prebrodivu manu povećanog kašnjenja izlaza (ovisno o primjeni sklopa).

Literatura

[1] J.M.P. Langlois and D. Al-Khalili: "Phase to sinusoid amplitude conversion techniques for direct digital frequency synthesis"

[2] Davor Petrinović, Member IEEE and Marko Brezović: „Direct Digital Frequency Synthesis Using B-Spline Piecewise-Polynomial Model“, unpublished paper

[3]

http://www.dsprelated.com/dspbooks/pasp/Farrow_Structure_Variable_Delay.html

(3. listopad, 2009)

[4] <http://www.bobpowell.net/cardinalspline.htm>

(3. listopad, 2009)

[5] Mladen Vučić i Goran Molnar: „Napredni alati za razvoj digitalnih sustava – Materijali za predavanja“; „Napredni alati za razvoj digitalnih sustava – Upute za laboratorijske vježbe“, Zagreb, 2004.

[6] <http://www.xilinx.com/>

(3. listopad, 2009)

[7] <http://www.mathworks.com/>

(3. listopad, 2009)

[8] <http://www.ieindia.org/pdf/88/88ET104.pdf>

(3. listopad, 2009)

Dodatak

DODATAK 1 – DIO REFERENTNOG KODA ZA SIMULACIJU DDFS-a KOJI RASPISUJE PROTOČNU STRUKTURU PO STUPNJEVIMA (MATLAB)

```
:
% Raspis protodne strukture po stupnjevima (STAGE)
% STAGE 1
ph=mod([0:Nt-1]*kor+poc,NM);

% STAGE 2
INd=[NaN floor(ph/M)]; % lokalno
MSB_INd=floor(INd/(N/2)); % lokalno
NSB_INd=floor((INd-MSB_INd*(N/2))/(N/4)); % lokalno
DXd=[NaN mod(ph,M)]; % lokalno
MSB_DXd=floor(DXd/(M/2)); % lokalno
ZDX=[DXd-M/2]; % ovo je samo
% komplementiranje
% najviseg bita i
% interpretacija podatka kao
% dvojnog komplement

SIGd=[NaN ~xor(MSB_INd(2:end),NSB_INd(2:end))];
SIGc=[NaN ~MSB_INd(2:end)];
SIGb=[NaN xor(MSB_INd(2:end),NSB_INd(2:end))];
SIGa=[NaN MSB_INd(2:end)];
INQ=[NaN (~NSB_INd(2:end).*mod(INd(2:end),N/4)+NSB_INd(2:end).*(N/4-1-
mod(INd(2:end),N/4)))]];

% STAGE 3 % Citanje ROMa
DU=[NaN2 DUm(INQ(2:end)+1)];
CU=[NaN2 CUm(INQ(2:end)+1)];
BU=[NaN2 BUm(INQ(2:end)+1)];
AU=[NaN2 AUm(INQ(2:end)+1)];
ZDXd=[NaN ZDX];
SIGd_d=[NaN SIGd];
SIGc_d=[NaN SIGc];
SIGb_d=[NaN SIGb];
SIGa_d=[NaN SIGa];

% STAGE 4 % Restauracija predznaka svih koeficijenata
D=[NaN3 (~SIGd_d(3:end).*DU(3:end) + SIGd_d(3:end).*(-DU(3:end)))];
C=[NaN3 (~SIGc_d(3:end).*CU(3:end) + SIGc_d(3:end).*(-CU(3:end)))];
B=[NaN3 (~SIGb_d(3:end).*BU(3:end) + SIGb_d(3:end).*(-BU(3:end)))];
A=[NaN3 (~SIGa_d(3:end).*AU(3:end) + SIGa_d(3:end).*(-AU(3:end)))];
ZDXdd=[NaN ZDXd];

figure(1);
plot((diag(2.^-rez_coef)*[D' C' B' A']')')

% STAGE 5
Xg=floor(ZDXdd*2^(bM-rez_DX(1)-1)); % Odrezana X-grana (vrh) ... lokalno
Pr=floor(Xg.*D*2^(rez_coef(1)+rez_DX(1)-(rez_prod(1)+1)));
% Umnozак s jednim bitom viska
Cg=[NaN C+floor(Pr/2)+mod(Pr,2)]; % Zaokruženje
```

```

ZDXddd=[NaN ZDXdd];
Bd=[NaN B];
Ad=[NaN A];

% STAGE 6
Xg=floor (ZDXddd*2^-(bM-rez_DX(2)-1));           % Odrezana X-grana (vrh)
Pr=floor (Xg.*Cg*2^-(rez_coef(2)+rez_DX(2)-(rez_prod(2)+1)));
                                                    % Umnozak s jednim bitom viska
Cgd=[NaN Bd+floor (Pr/2)+mod (Pr,2)];           % Zaokruzenje
ZDXdddd=[NaN ZDXddd];
Add=[NaN Ad];

% STAGE 7
Xg=floor (ZDXdddd*2^-(bM-rez_DX(3)-1));           % (u pravilu cijeli DX)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ovaj dio koda će biti izmijenjen %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Pr=floor (Xg.*Cgd*2^-(rez_coef(3)+rez_DX(3)-(rez_prod(3)+1)));
                                                    % Umnozak s jednim bitom viska

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Out=[NaN Add+floor (Pr/2)+mod (Pr,2)];           % Zaokruzenje

% Provjera rezultata
ys=sin (2*pi*ph/NM);                               % idealna sinusoida
y_fix=Out (7:end)*2^-rez_coef (end);               % izlaz PSAC-a
figure (2);
plot ([ys' y_fix'])

gr=(ys-y_fix)*2^b_out;                             % pogreska u LSB od 1.b_out
m_gr (k)=max (abs (gr));

std_gr (k)=std (gr);                               % max. abs. pogreska u odnosu na float sinus
                                                    % std pogreske
:

```

DODATAK 2 – IZMJENE U REFERENTNOM KODU

Za razliku od referentnog koda, izmjene su pisane pomoću fiksnih rezolucija za lakše povezivanje s realizacijom sklopa u *Xilinx-u*.

```
:
% STAGE 7
Xg=floor (ZDXdddd*2^-(bM-rez_DX(3)-1)); % (u pravilu cijeli DX)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% izmijenjeni dio koda %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Cgd1=floor (Cgd*(2^-6)); % m5HI
Cgd2=abs (Cgd-(Cgd1*2^6)); % m5LO
XgHI_pom=abs (floor (Xg*2^-11)); % dxbHI
XgHI=XgHI_pom;
Pr1=floor (Xg.*Cgd1*2^-5); % m6HI
Pr2=floor (XgHI.*Cgd2); % m6LO
for i=1:length (Xg)
    if Xg(i)>0
        Pr(i)=floor ((Pr1(i)+Pr2(i))*2^-5);
    else
        Pr(i)=floor ((Pr1(i)-Pr2(i))*2^-5);
    end; % zbrajanje/oduzimanje parcijalnih
        % produkata s jednim bitom viška
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Out=[NaN Add+floor (Pr/2)+mod (Pr,2)]; % Zaokruzenje
:
```