Sistem za direktnu digitalnu sintezu

Kristijan Mitrović
Odsek za elektroniku
Elektrotehnički fakultet
Beograd, Srbija
mk150214d@student.etf.bg.ac.rs

Dragan Božinović
Odsek za elektroniku
Elektrotehnički fakultet
Beograd, Srbija
bd150211d@student.etf.bg.ac.rs

***Sažetak*—Ovaj rad predstavlja opis rešenja projekta iz predmeta *Hardversko-softverska obrada signala*. Dat je pregled realizovanih funkcionalnosti uz potrebne teorijske temelje nužne za razumevanje odabrane implementacije.**

I. Razvojno okruženje

Programski kod koji obavlja zahtevanu funkcionalnost sistema napisan je u programskom jeziku *C++*, dok je razvojno okruženje korišćeno za izradu ovog projekta *Visual Studio Code*.

Za potrebe fiksne aritmetike korišćene su biblioteke `ac_fixed.h` i `ac_int.h`, pri čemu prva omogućava rad sa brojevima sa fiksnim zarezom dok druga biblioteka služi za korišćenje celobrojnih promenljivih ograničenih na određen broj bita.

Funkcionalnost je ostvarena u okviru fajla "*main.cpp*". Vrednosti koje se po uslovu zadatka vode na DAC konvertor i određeni međurezultati bivaju smešteni u fajl "*DDSlog.csv*", dok fajl "*logProcess.m*" predstavlja skriptu koja se pokreće iz *MATLAB*-a i služi za prikaz rezultata smeštenih u okviru *log* fajla.

II. Funkcionalni blokovi sistema

A. Fazni akumulator

Fazni akumulator realizovan je u vidu promenljive `phaceAcc`. Ova promenljiva, budući da je ograničena širinom od $W = 40$ bita, predstavlja broj u opsegu $[0, 2^{40} - 1]$. Frekvencija generisanog signala određena je inkrementom faznog akumulatora, označenog sa `dteta`, pri čemu minimalna vrednost inkrementa predstavlja frekvencijsku rezoluciju. Ipak, budući da su obe promenljive bezdimenzioni brojevi, omogućena je intuitivna manipulacija ovim vrednostima. Imajući u vidu da je najmanji inkrement faznog akumulatora jednak $\Delta\theta_{min} = 2^0 = 1$, `dtheta` se računa po formuli

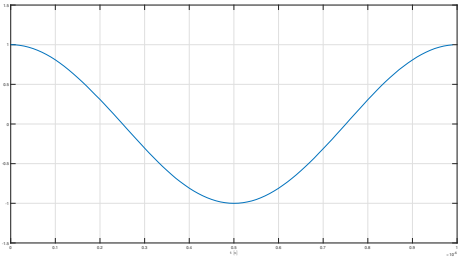
$$\Delta\theta = M_{inc}\,\Delta\theta_{min} = \frac{f_0}{\Delta f} = \frac{f_0}{f_{clk}}\,2^W, \tag{1}$$

pri čemu je sa f_o označena učestanost željenog signala a sa f_{clk} učestanost takta. Ovakvim pristupom, korisniku je omogućeno da unese željenu frekvenciju u vidu promenljive `f0` koja nadalje biva skalirana u odgovarajući inkrement faznog akumulatora, čime u potpunosti biva lišen unošenja neintuitivnih brojeva. Ujedno je i za najmanju ostvarivu frekvenciju, Δf, inkrement faznog akumulatora jednak bitu najmanje težine, čime je postignuta uniformna kvantizacija faznog akumulatora, odnosno, pri ovoj učestanosti kvantizacioni šum ne postoji. Maksimalna ostvariva učestanost ovakvog sistema, $f_s/2$, biva preslikana u inkrementalni pomeraj od 2^{W-1}, odnosno, u pomeraj koji je jednak π.

Početna faza, označena sa `phi`, predstavlja takođe ceo broj dužine $W = 40$ bita. Korisnik unosi željenu početnu fazu preko promenljive `phi0`, pri čemu njena vrednost pripada intervalu $[0, 2\pi]$. Tako unesena vrednost se dalje skalira: kako maksimalni pomeraj od 2^W odgovaraja inkrementu faze za 2π, to znači da za $\phi_0 = 2\pi$ inkrement akumulatora mora biti jednak 2^W. Stoga, početna faza koja se pridodaje faznom akumulatoru računa se po formuli

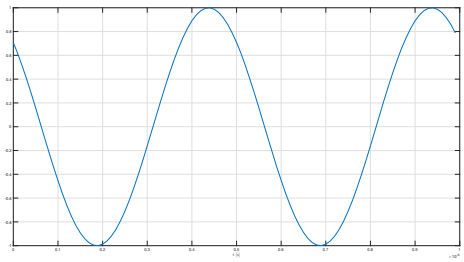
$$\phi = \frac{\phi_0}{2\pi}2^W \tag{2}$$

Na slici 1 prikazan je kosinusoidalni signal čija je učestanost $f_0 = 1$ MHz.



Slika 1. *Kosinus učestanosti 1 MHz*

Promenom korisnički definisanih parametara na $f_0 = 2$ MHz i $\phi_0 = \pi/4$ dobija se kosinusoida prikazana na slici 2. Kao što se da videti, učestanost kosinusoide je duplo veća, pri čemu je izvršen fazni pomeraj koji je unet od strane korisnika.



Slika 2. *Kosinus učestanosti 2 MHz i početne faze π/4*

B. Generator odbiraka cos x

Izlaz iz faznog akumulatora, promenljiva `phaseAccOut`, dovodi se na ulaz u *CORDIC* algoritam, gde je nadalje potrebno izračunati vrednost amplitude na osnovu dovedene vrednosti. S obzirom na to da se vrednosti iz *CORDIC* algoritma prosleđuju DA konvertoru, odnosno, da je broj različitih vrednosti koje DAC može prihvatiti jednak 2^{14}, proizilazi da je broj bita koji treba proslediti ovoj tabeli jednak $M = N = 14$.

Kako je maksimalna vrednost ugla koja se može proslediti *CORDIC* algoritmu jednaka $99.88°$, potrebno je na neki način ograničiti vrednost ugla koja se prosleđuje za izračunavanje amplitude. To je učinjeno tako što se najviših dva bita dovodi na ulaz multipleksera u vidu promeljive `muxSel`, dok se ostalih 12 bita koristi za mapiranje vrednosti u opsegu od $[0, \frac{\pi}{2}]$ u vidu `phaseAccOut`.

CORDIC algoritam izvršava se u rotacionom modu u *M* iteracija, a po sledećoj formuli:

$$x_{i+1} = x_i - \sigma_i y_i \gg i \quad (3)$$

$$y_{i+1} = y_i + \sigma_i x_i \gg i \quad (4)$$

$$z_{i+1} = z_i - \sigma_i \arctan 2^{-i} \quad (5)$$

$$\sigma_i = \mathrm{sgn}(z_i) \quad (6)$$

Radi optimizacije, pre samog početka rada napravljena je svojevrsna tabela vrednosti za $\arctan 2^{-i}$. Budući da se taj niz vrednosti pojavljuje u svakoj iteraciji faznog akumulatora, ovim je postignuta manja redudansa kao i brže izvršavanje samog programa.

Ipak, kako kontrolna promenljiva na ulazu u *CORDIC* tabelu, `phaseAccOut`, nema nikakvu realnu dimenziju, potrebno je izvršiti skaliranje. Budući da je maksimalna vrednost `phaseAccOut` jednaka $2^{M-2} - 1$ i da ona odgovara fazi od $\pi/2$, promenljiva `z` se računa po formuli:

$$z = \frac{phaseAccOut}{2^{M-2} - 1} \frac{\pi}{2} \quad (7)$$

Izlazne vrednosti, x i y, smeštaju se zatim u promenljive `sampleSin` i `sampleCos`, na osnovu vrednosti promenljive `muxSel`.

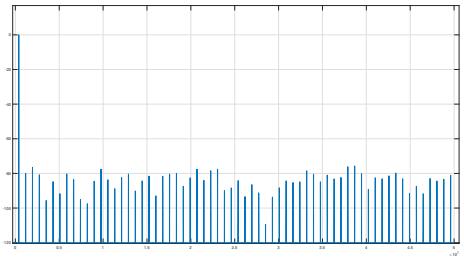
ZAŠTITNI BITI?

C. Diter

Spurovi se javljaju kao neželjene periodične smetnje u spektru izlaznog signala i posledica su neuniformne kvantizacije faze. Pri koherentnom odabiranju uslovljenom jednačinom

$$\frac{f_0}{f_{clk}} = \frac{P}{M} \quad (8)$$

gde je f_0 željena učestanost signala, f_{clk} učestanost odabiranja, P paran broj, a $M = 2^m$ broj odbiraka, spurovi su jasno vidljivi u spektru signala, kao na slici 3.

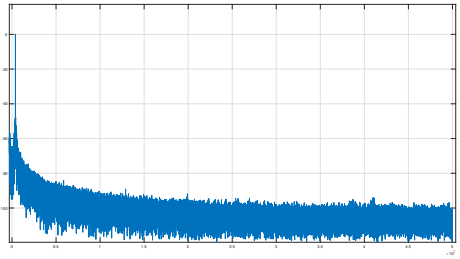


Slika 3. *Spektar izlaznog signala u kom su vidljivi spurovi*

Taktovanje celog sistema u *C++*-u implementirano je `for` petljom koja se, na osnovu jednačine (8), izvršava 2^n puta. U implementaciji je uzeto da je $n = 16$, koje obezbeđuje izuzetno brzo izvršavanje programa u *C++*-u, dok, s druge strane, *MATLAB*-u pruža dovoljan broj odbiraka za detaljno iscrtavanje signala i njegovog spektra. Naravno, ovo uzrokuje da je frekvencijski bin u *MATLAB*-u širine $\Delta f = f_{clk}/2^{16}$, a ne $\Delta f = f_{clk}/2^{40}$, ali je dovoljno dobar kompromis između oprečnih zahteva za preciznošću i kratkim vremenom izvršavanja.

Tehnika koja je iskorišćena za razbijanje spurova je tzv. *ditering*. Ova tehnika sastoji se od dodavanja nekorelisanog ili pseudoslučajnog signala na "spurovit" signal, čime se razbijaju periodične komponente. Pseudoslučajan signal u *C++*-u implementiran je instanciranjem objekata `default_random_engine` i `normal_distribution` iz `<random>` biblioteke, koji omogućavaju generisanje slučajnih brojeva sa normalnom raspodelom $\mathcal{N}{\sim}(\mu, \sigma^2)$ sa proizvoljnim μ i σ. Generisani pseudonasumični brojevi sa $\mu = 0$ i $\sigma = 1$ pomnoženi su sa 2^{26} kako bi se šiftovali do bita koji se prosleđuju *CORDIC* algoritmu, kao i sa `ditherAmp` čime se podešava nivo ditera koji se dodaje na fazni akumulator u svakoj iteraciji pomenute `for` petlje.

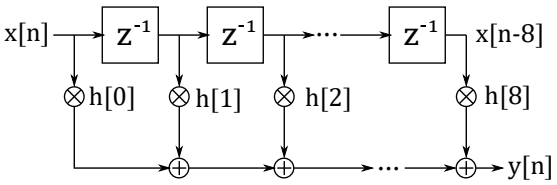
Na slici 4 prikazan je spektar izlaznog signala popravljen nesupstraktivnim diterom, na kom se više ne uočavaju spurovi.



Slika 4. *Spektar izlaznog signala bez spurova*

D. FIR filtar

FIR filtar za korekciju $\sin x/x$ frekvencijske karakteristike kola zadrške nultog reda implementiran je direktnom realizacijom sa slike 5.



Slika 5. *Direktna realizacija FIR filtra*

Prethodni sistem može se matematički zapisati kao

$$y[n] = \sum_{i=0}^{8} h[i]\, x[n-i] \tag{9}$$

što predstavlja konvoluciju koeficijenata filtra $h[0] \ldots h[8]$ sa trenutnim $x[n]$ i prethodnim $x[n-1] \ldots x[n-8]$ izlazima *CORDIC*-a.

Na identičan način je u nizove `cordicSin` i `cordicCos` smeštano poslednjih devet izlaza generatora odbiraka, i ti nizovi su po jednačini (9) konvoluirani sa koeficijentima filtra prethodno izračunatim u *Python*-u tako da filter zadovoljava zahtevane uslove, i smeštenim u `filter` niz. Rezultati konvolucije, tj. filtriranja `DACinputCos` i `DACinputSin` potom su upisani u izlazni fajl za logovanje.

LITERATURA

[1] Dr. Dušan Grujić, Predavanja iz predmeta *Hardversko-softverska obrada signala*, Elektrotehnički fakultet, Beograd

[2] Bruce Land, *Direct Digital Synthesis*, Cornell University, New York, https://www.youtube.com/watch?v=YDC5zaEZGhM

[3] Analog Devices, *Fundamentals of Direct Digital Synthesis*, https://www.analog.com/media/en/training-seminars/tutorials/MT-085.pdf

[4] HLS LIBS, *AC DATATYPES*, https://hlslibs.org/#collapseACDatatypes

[5] C++, *std::normal_distribution*, http://www.cplusplus.com/reference/random/normal_distribution/