UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA AUTOMATIKU I ELEKTRONIKU

Predmet: Analiza signala i sistema

Završni rad

Pregled FFT algoritama

Mentor: Red. prof. dr Melita Ahić - Đokić, dipl. el. inž.

Student: Adisa Sinanović

Sarajevo, 2009. godina

Zadatak završnog rada

Red. prof. dr Melita Ahić-Đokić, dipl.el.inž.
Asistent Emir Sokić, dipl.el.inž.
Odsjek za automatiku i elektroniku
Sarajevo, 01.02.2009.

Tema za završni rad studenta

na ETF-u u skladu sa principima Bolonjskog procesa

na Odsjeku za automatiku i elektroniku (akademska 2008/09. godina)

Tema: "Pregled FFT algoritama"

Sažetak:

U radu je potrebno definirati pojmove Fourierove transformacije (FT), Diskretne Fourierove transformacije (DFT) i Brze Fourierove transformacije (FFT). Potrebno je obrazložiti razloge za uvođenje brze Fourierove transformacije, i izvršiti kvantitativnu i kvalitativnu analizu najpoznatijih FFT algoritama (tehnika računanja), sa aspekta brzine računanja i potrebnih memorijskih i procesorskih resursa. Rad treba sadržavati i kritički osvrt na tačnost rezultata uzrokovanu računanjem.

Potrebno je softverski realizirati i analizirati neke od najpopularnijih postojećih algoritama za računanje FFT-a, a zatim izvršiti i usporedbu sa rezultatima koji se dobiju korištenjem hardvera specijalne namjene za računanje FFT-a. Posebno analizirati razlog uvođenja prozorskih funkcija u analizi realnih signala, te njihov utjecaj na izobličenje spektra. Pokazati i kako se u praksi pri računanju FFT-a izbjegava mogućnost aliasinga (preklapanja).

Metode rješavanja:

Za izradu softverskog dijela završnog rada i računarsku analizu preporučuje se upotreba MATLAB softverskog paketa, mada je dozvoljeno korištenje i drugih okruženja (LabView, C/C++, Java itd.). Za analizu algoritama FFT-a preporučuje se korištenje specijalne serije osciloskopa sa mogućnošću frekventne analize, i spektralnih analizatora koji se nalaze u vlasništvu ETF Sarajevo, u kombinaciji sa akvizicionim modulima radi jednostavnije računarske obrade.

Potpis mentora:

_____

Red. prof. dr Melita Ahić-Đokić

Sažetak

U ovom radu predstavljene su osnove Fourierove analize, obrada signala i dati FFT algoritmi. Rad je podijeljen na tri poglavlja. U prvom poglavlju objašnjeni su osnovni pojmovi vezani za Fourierov red, Fourierovu transformaciju i diskretnu Fourierovu transformaciju. Pomoću digitalnog osciloskopa pokazana je primjena prozorskih funkcija i neke osobine signala. Drugo i treće poglavlje odnose se na način izvođenja koraka FFT algoritama, u zavisnosti od toga da li se koraci izvode sekvencijalno (u nizu) ili se koraci izvode istovremeno. U drugom poglavlju predstavljena su prvo dva osnovna FFT algoritma koja se razlikuju u načinu defisanja podsekvenci, to su brza Fourierova transformacija sa decimiranjem po vremenu i brza Fourierova transformacija sa decimiranjem po učestanosti. Također, predstavljeni su FFT algoritmi "u - mjestu" po bazi 2 i njihove brzine računanja. Da bi obezbjedili komunikaciju između korisnika i računara u korištenju datih kodova u radu napravljeno je grafičko korisničko okruženje. U trećem poglavlju objašnjeni su principi paralelnih algoritama.

Abstract

This paper presents the basics of Fourier's analysis, signal processing and FFT algorithms. The paper consist of three chapters. The basic concepts related to the Fourier series, Fourier transform and discreet Fourier's transform are explained in the first chapter. The use of window functions and some of the signal's characteristics are demonstrated using a digital oscilloscope. The second chapter describes the FFT algorithms where the steps are performed sequentially. Here are presented the first two basic FFT algorithm, which differ in the way of defining subsequences, that are fast Fourier transform with decimation in time and fast Fourier transform with decimation in frequency. Additionally, FFT algorithms "in - place" in radix 2 are presented and the speed of computation for all algorithms is compared. To better provide communication between users and computers to use the codes given in the paper, Graphical User Interface is made. The principles of parallel algorithms and the FFT algorithms where the steps are performed simultaneously are explaned in the third chapter.

Sadržaj

Lista skraćenica

Skraćenica	Objašnjenje
DFT	Discrete Fourier Transform Diskretna Fourierova transformacija
DIF	Decimation in frequency Decimiranje po učestanosti
DIT	Decimation in time Decimiranje po vremenu
FFT	Fast Fourier Transform Brza Fourierova transformacija
flop	floating - point operation
FT	Fourier Transform Fourierova transformacija
IDFT	Inverse Discrete Fourier Transform Inverzna Fourierova transformacija

Uvod

Predmet istraživanja ovog rad su algoritmi brze Fourierove transformacije. Ovaj rad je odabran zbog mogućnosti eksperimentalnih mjerenja, razvijanja kodova u Matlabu i korištenja grafičkog korisničkog okruženja.

Brza Fourierova transformacija predstavlja jedno od najvažnijih dostignuća u nauci i inženjerstvu. Široka primjena računara uveliko je doprinijela u razvoju FFT algoritama, odnosno sada je potrebno manje vremena za izvršavanja algoritma za isti broj uzoraka. FFT je optimiziran i efikasan algoritam za izračunavanje vrijednosti diskretne Fourierove transformacije neke funkcije. Postoje mnogi različiti FFT algoritmi koji uključuju širok raspon matematičkih metoda, od kompleksnih brojeva, teorije grupa do metoda numeričke matematike. Shvatajući da se Fourierova transformacija sekvence može izračunati iz Fourierove transformacije dvije sekvence upola manje dužine i primjenjujući ovu ideju, Cooley i Tukey (1965.) su pokazali da je potrebno samo $N \log_2 N$ "flopova" za izračunavanje FFT-a, dok je za računanje FFT -a iz direktne formule potrebno N^2 "flopova".

U prvom poglavlju objašnjeni su osnovni pojmovi koji su potrebni da bi razumjeli brzu Fourierovu transformaciju. Da bi lakše shvatili uticaj prozorskih funkcija i osobine signala izvršili smo eksperimentalna mjerenja na digitalnom osciloskopu.

U drugom poglavlju su opisane dvije varijante FFT algoritama, FFT sa decimiranjem po vremenu i FFT sa decimiranjem po učestanosti. U ovom radu prvo smo krenuli od izvođenja FFT algoritama po bazi 2 i dali odgovarajuće kodove koji su napisani u Matlab-u. U nastavku, izvedeni su FFT algoritme po bazi 4 i pokazano je kako se algoritam može primijeniti općenito po bazi 2^s, gdje je s cijeli pozitivan broj. U razmatranja algoritama sa bazama $8, 16, \ldots$ nismo ulazili, zato što je predstavljen optimalan algoritam sa razdvojenom bazom. Za algoritme koji su implementirani u Matlabu data su vremena izvršavanja za različit broj uzoraka za svaki algoritam pojedinačno. Da bi primijenili date kodove na određenu funkciju napravljeno je grafičko korisničko okruženje.

U trećem poglavlju dat je kratak pregled o principima paralelnih algoritama. Zbog preobimnosti problematike nismo ulazili u izvođenje FFT algoritama.

1 Osnovni pojmovi

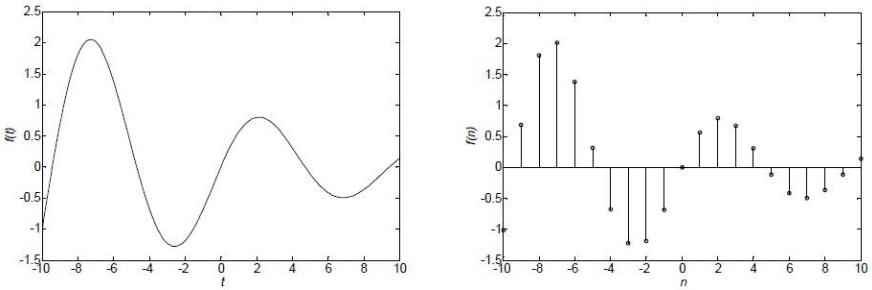
1.1 Jean Baptiste Joseph Fourier

Slika 1.1. *Joseph Fourier (1768 – 1830.)*

Jean Baptiste Joseph Fourier (21.3.1768. – 16.5.1830.) rođen je u Auxerre-u, Francuska. Bio je deveto od dvanaestero djece drugog braka jednog krojača. Sa trinaest godina počeo je pokazivati interes za matematiku, ali sa devetnaest odlučuje učiti za svećenika. Ipak, nakon dvije godine shvaća da svećenička karijera nije za njega te izlazi iz samostana prije polaganja zakletve. Nakon toga nastavlja se baviti matematikom i radi kao učitelj u školi koju je i sam pohađao. 1793. godine počinje se zanimati za politiku i staje na stranu revolucije želeći se boriti za jednakost. Svojim talentom za govorništvo privlači mnoge, ali se ubrzo i razočarao nasiljem revolucije. Izlazak iz politike ipak nije jednostavan, te ostaje zagovornik jedne od frakcija i biva uhapšen. Nakon oslobođenja počinje pohađati École Normale, novoosnovanu školu za učitelje, gdje ga podučavaju Lagrange, Laplace i Monge, a sam Fourier se ističe kao odličan student. Nakon završetka, postaje učitelj na drugim visokim školama, a od 1797. nasljeđuje Lagrange-a na mjestu profesora analize i mehanike na znamenitoj École Polytechnique. Kad 1798. Napoleon kreće u invaziju na Egipat, Fourier se pridružuje ekspediciji kao znanstveni savjetnik. U Egiptu će se zadržati sve do konačnog povratka ekspedicije 1801, a radit će kao tajnik Instituta u Kairu i baviti se skupljanjem znanstvene i književne građe. Pri povratku u Pariz, Fourier se nada nastaviti svoju znanstvenu i nastavnu aktivnost, no Napoleon ga imenuje za prefekta departmana Isère sa sjedištem u Grenoble-u. Tu će Fourier nadzirati razne građevinske radove, ali za to vrijeme će se početi baviti i teorijom topline i tako završiti svoj znameniti memoar o širenju topline u čvrstim tijelima. Taj će memoar izazvati niz kontroverzi u to doba, od kojih je najvažnija bila pitanje smislenosti razvijanja funkcije u red trigonometrijskih funkcija - danas poznati Fourierov red. Nakon konačnog poraza Napoleona, Fourier se vraća u Pariz, nastavlja znanstvenu karijeru, postaje član Akademije znanosti (1817.) i tajnik njenog matematičkog odjela (1822.). U toku zadnjih osam godina života objavio je niz radova iz čiste i primijenjene matematike, ali je nastavio i sukob s Biotom oko prvenstva u teoriji topline. U sukob se protiv Fouriera uključio i Poisson, no kasniji je razvoj pokazao tačnost Fourierovih rezultata i tehnika. Danas se Fourier pamti ponajviše kao pokretača teorije trigonometrijskih redova. Ideja Fourierovih redova je da proizvoljnu periodičnu funkciju (tj. funkciju koja je zadana na nekom osnovnom intervalu, a njen oblik na cijelom skupu realnih brojeva dobijemo "lijepljenjem" kopija te funkcije jednu uz drugu u oba smjera) zapisati kao suma sinusoida različitih amplituda i frekvencija (koje je lakše analizirati jer su svojstva sinusoida dobro poznata).

1.2 Pojam signala

Signali se susreću u gotovo svim oblastima nauke i tehnike, na primjer, u fizici, seizmologiji, biomedicinskim istraživanjima, telekomunikacijama, sistemima upravljanja itd. Razlikuju se dvije opšte klase signala: analogni i diskretni. Ako je signal u dijelovima kontinualna funkcija nezavisne promjenljive (koja je, po pravilu, vrijeme), radi se o analognom (kontinualnom, neprekidnom) signalu. Tipičnim predstavnikom analognih signala može se smatrati vremenski signal napona ili struje običnog RLC kola. Analogni signal se može predstaviti funkcijom $f(t)$, definisanom na intervalu (t_1, t_2) realne ose, gdje je $-\infty \leq t_1 \leq t_2 \leq \infty$. S druge strane, diskretni signal je definisan samo u diskretnim trenucima vremena, na primjer, svake sekunde ili svakog dana. Neki primjeri diskretnih signala su: mjesečni broj nezaposlenih, osnovni godišnji nacionalni dodatak, mjesečna proizvodnja automobila itd. Diskretni signal se predstavlja funkcijom $f(n)$, gdje je n - cio broj iz intervala (n_1, n_2) cjelobrojne ose $(-\infty \leq n_1 \leq n_2 \leq \infty)$. Na slici 1.2 je dat grafički primjer jednog analognog $f(t)$ i jednog diskretnog signala $f(n)$.



Slika 1.2. *Grafički prikaz analognog signala $f(t)$ i diskretnog signala $f(n)$*

1.3 Fourierov red

Iako je Fourierov red izvorno namijenjen za rješavanje jednačine širenja topline, nakon nekog vremena postalo je jasno da se ista metoda može primijeniti na velik broj matematičkih i fizikalnih problema koji su dotad bili nerješivi. Fourierov red najčešće se primjenjuje u elektrotehnici, optici i obradi signala i slika.

Da bi razvili periodičnu funkciju u Fourierov red (koji je također poznat kao i trigonometrijski red) potrebno je da funkcija zadovoljava Dirichletovu teoremu, koja nam daje dovoljne uslove za postojanje Fourierovog reda.

Teorema 1. (Dirichletova teorema) Ako su zadovoljeni uvjeti:

1. funkcija $f(t)$ na $[a, b]$ je neprekidna ili na (a, b) ima samo konačan broj prekida prve vrste,

2. funkcija $f(t)$ ima na $[a, b]$ samo konačan broj maksimuma i minimuma, tj. segment $[a, b]$ se može razložiti na konačan broj dijelova na kojima je $f(t)$ monotona,

tada Fourierov red te funkcije konvergira za sve vrijednosti $t \in [a, b]$ i jednak je:

a) $f(t)$ kad je t tačka neprekidnosti iz (a, b),

b) $\frac{f(t-0)+f(t+0)}{2}$ kada je t tačka prekida iz (a, b).

c) $\frac{f(a_+)+f(b-)}{2}$ na krajevima segmenta $[a, b]$.

Ako funkcija $f(t)$ nije periodična, onda se njena restrikcija ili sama ta funkcija može periodički produžiti do funkcije definirane na skupu realnih brojeva.

Razvoj funkcije perioda T u Fourierov red na intervalu $[a, b]$:

$$f(t) \sim \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos \frac{2\pi kt}{T} + b_k \sin \frac{2\pi kt}{T} \right),$$

gdje je $T = b - a$, a Fourierovi koeficijenti se izračunavaju po formuli

$$a_k = \frac{2}{T}\int_a^b f(t)\cos\frac{2\pi kt}{T}dt, \quad k = 0, 1, 2, \ldots, \tag{1.1}$$

$$b_k = \frac{2}{T}\int_a^b f(t)\sin\frac{2\pi kt}{T}dt, \quad k = 1, 2, \ldots \tag{1.2}$$

Fourierov red periodične funkcije sastavljen je od niza harmonika čije su frekvencije cjelobrojni djelitelji osnovne frekvencije $\omega_0 = 2\pi/T$, pa se razvoj u Fourierov red može zapisati i na sljedeći način:

$$\frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k\cos k\omega_0 t + b_k\sin k\omega_0 t) = \pm\frac{c_0}{2} + \sum_{k=1}^{\infty} c_n\cos(k\omega_0 t + \varphi_k),$$

pri čemu je

$$c_0 = |a_0|, \qquad c_k = \sqrt{a_k^2 + b_k^2}, \qquad tg\varphi_k = \frac{a_k}{b_k}.$$

Često je od praktičnog interesa zapisati Fourierov red funkcije $f(t)$ u ekvivalentnom kompleksnom obliku. Ako su a_k, b_k Fourierovi koeficijenti djelomično neprekidne funkcije $f(t)$ na segmentu $[a, b]$ u trigonometrijskom sistemu, onda se lahko pokaže da vrijedi:

$$f(t) \sim \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k\cos\frac{2\pi kt}{T} + a_k\sin\frac{2\pi kt}{T} = \sum_{k=-\infty}^{\infty} C_k e^{j\frac{2\pi kt}{T}}, \tag{1.3}$$

gdje je

$$C_0 = \frac{a_0}{2}, \qquad C_{\pm k} = \frac{a_k \mp jb_k}{2} \text{ za } k \geq 1.$$

Koristeći jednačine (1.1) i (1.2) iz Dirichletove teoreme da bi izračunali a_k i b_k za $k \geq 1$ imamo

$$\begin{aligned} C_{\pm k} &= \frac{a_k \mp jb_k}{2} \\ &= \frac{1}{2}\left[\frac{2}{T}\int_a^b f(t)\left(\cos\frac{2\pi kt}{T} \mp j\sin\frac{2\pi kt}{T}\right)dt\right] \\ &= \frac{1}{T}\int_a^b f(t)e^{\mp j\frac{2\pi kt}{T}}dt. \end{aligned}$$

Primjećujemo da je

$$C_0 = \frac{a_0}{2} = \frac{1}{2}\left[\frac{2}{T}\int_a^b f(t)\cos 0dt\right] = \frac{1}{T}\int_a^b f(t)e^0 dt.$$

Koeficijente Fourierovog reda signala $f(t)$, koje nazivamo spektralni koeficijenti, određujemo iz relacije:

$$C_k = \frac{1}{T}\int_a^b f(t)e^{-j\frac{2\pi kt}{T}}dt \tag{1.4}$$

koja vrijedi za svako k. Signal $f(t)$ i koeficijenti C_k predstavljaju transformacioni par Fourierovog reda, jer iz C_k možemo odrediti $f(t)$, dok iz $f(t)$ možemo odrediti C_k. Tu vezu možemo zapisati u obliku:

$$f(t) \overset{FR}{\longleftrightarrow} C_k.$$

1.4 Fourierov integral

Uvođenjem kraćih oznaka $\omega_0 = 2\pi/T = \Delta\omega$ i $k\omega_0 = \omega_k$ i uvrštavanjem izraza iz C_k (1.4) u (1.3) imamo:

$$\begin{aligned} f(t) &= \sum_{k=-\infty}^{k=\infty} \frac{1}{T} \int_a^b f(x) e^{-jk\omega_0 x} e^{jk\omega_0 t} dx = \sum_{k=-\infty}^{k=\infty} \frac{1}{T} \int_a^b f(x) e^{jk\omega_0 (t-x)} dx \\ &= \sum_{k=-\infty}^{k=\infty} \frac{\omega_0}{2\pi} \int_a^b f(x) e^{j\omega_k (t-x)} dx \end{aligned}$$

Ako posmatramo graničnu vrijednost posljednjeg izraza kada $l \to +\infty$ (slijedi da $\Delta\omega \to 0$) dobijemo da je:

$$\begin{aligned} f(t) &= \frac{1}{2\pi} \lim_{T\to\infty} \sum_{k=-\infty}^{k=\infty} \Delta\omega \int_a^b f(x) e^{j\omega_k (t-x)} dx \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} d\omega \int_{-\infty}^{\infty} f(x) e^{j\omega_k (t-x)} dx. \end{aligned} \tag{1.5}$$

Posljednja relacija se u stručnoj literaturi često naziva *dvojni Fourier integral u kompleksnom obliku*.

Uslovi konvergencije Fourierovog integrala mogu se formulirati na sljedeći način:

1. funkcija $f : \mathbf{R} \to \mathbf{R}$ je apsolutno integrabilna funkcija, tj. $\int_{-\infty}^{\infty} |f(t)| < \infty$,
2. funkcija f zadovoljava Dirichletove uslove.

Ako su zadovoljeni uslovi 1. i 2., onda vrijedi:

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(x) e^{-j\omega_k x} dx \right] e^{j\omega_k t} d\omega = \begin{cases} f(t), & f \text{ neprekidna u } t, \\ \frac{1}{2}\left[f(t+0) + f(t-0)\right], & f \text{ ima prekid prve vrste u } t. \end{cases} \tag{1.6}$$

1.5 Fourierova transformacija

Posmatrajmo relaciju (1.5). Integral $\int_{-\infty}^{\infty} f(t) e^{j\omega(t-x)} dx$ možemo smatrati funkcijom realne promjenljive ω i tada imamo:

$$F(j\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt,$$

pa Fourierov integral možemo pisati kao:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(j\omega) e^{j\omega t} d\omega.$$

Definicija 1.1. Neka je funkcija $f(t)$ apsolutno integrabilna i neka zadovoljava Dirichletove uslove na bilo kom konačnom intervalu u $\mathbf{R}$. Tada funkciju

$$F(j\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

zovemo *kompleksnim Fourierovim likom funkcije* $f(t)$ ili *Fourierovom transformacijom*, pri čemu funkciju $f(t)$ zovemo originalom i to kraće pišemo u obliku

$$\mathcal{F}\left[f(t)\right] = F(j\omega).$$

Definicija 1.2. Ukoliko postoji Fourierov integral dat relacijom (1.6), onda funkciju

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(j\omega) e^{j\omega t} d\omega$$

nazivamo *inverznom Fourierovom transformacijom funkcije* $F(j\omega)$ i možemo kraće pisati:

$$\mathcal{F}^{-1}\left[F(j\omega)\right] = f(t).$$

1.6 Aliasing

Aliasing je pojam koji opisuje posebnu pojavu pri analogno - digitalnoj konverziji, a posljedica je nedovoljne učestalosti uzorkovanja nekog analognog signala.

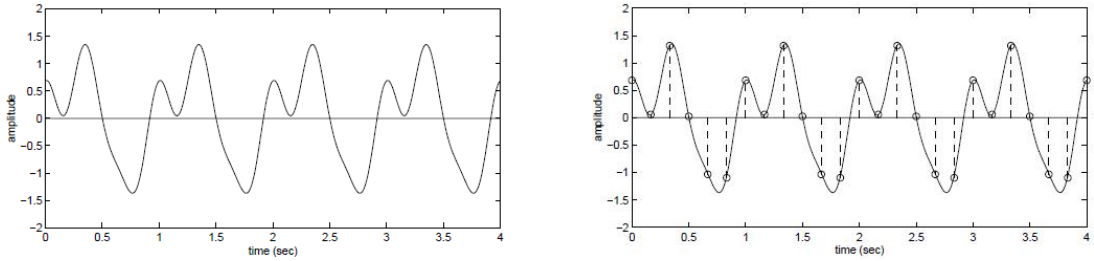
Nyquistova teorema uzorkovanja

Nyquistova teorema uzorkovanja definiše nominalno uzorkovanje da bi se izbjegao aliasing i glasi:

frekvencija uzorkovanja trebalo bi da bude najmanje dva puta veća od najveće frekvencije signala.

Ili matematički napisano, $f_s \geq 2f_c$, gdje je f_s frekvencija uzorkovanja (sampliranja), a f_c predstavlja najveću frekvenciju sadržanu u signalu.

Iz Fourierove teoreme znamo da se kontinualni signal može razložiti na sumu sinusoida različitih frekvencija i faza. Na primjer, sljedeći valni oblik je sastavljen od sume sinusoida frekvencija 1Hz, 2 Hz i 3 Hz, slika 1.3a. Prema Nyguistovoj teoremi uzorkovanja signal je potrebno uzorkovati najmanje dva puta većom frekvencijom koja je sadržana u signalu. U ovom slučaju, $f_c = 3Hz$, odakle slijedi da frekvencija uzorkovanja iznosi najmanje $f_s = 6Hz$, slika 1.3b.



(a) Kontinualni signal (b) Diskretizovani signal

Slika 1.3. *Nyquistova teorema*

Možemo primijetiti da ako ne izaberemo odgovarajuću frekvenciju uzorkovanja nećemo samo izgubiti informaciju, nego ćemo također dobiti pogrešnu informaciju o signalu. Iz ovoga vidimo odakle potiće naziv *aliasing*. Odnosno, kada signal poprima drugčiji oblik zbog nedovoljne frekvencije uzorkovanja, [3].

1.7 Diskretna Fourierova transformacija

Ulazna funkcija diskretne Fourierove transformacije (engl. **D**iscrete **F**ourier **T**ransform, DFT) mora biti definirana diskretnim vrijednostima, $t = kT_1$, i analizira se na ograničenom intervalu. S obzirom da je ulazna funkcija konačna sekvenca realnih i kompleksnih brojeva, DFT je idealna za procesiranje informacija pohranjenih u računaru. DFT ima vrlo veliko područje primjene, posebno u obradi signala i otklanjanju

šumova. Upravo zato je brza realizacija DFT jedan od najvažnijih i najčešće korištenih algoritama uopće. Na njemu su bazirani i mnogi drugi brzi algoritmi u aritmetici i algebri.

$$\bar{X}(m) = \sum_{k=0}^{N-1} \tilde{x}(k) W_N^{mk} \tag{1.7}$$

Relacija (1.7) predstavlja *direktnu diskretnu Fourierovu transformaciju* koja se dobija iz spektra zvjezdaste funkcije $F^*(j\omega)$.

$$\tilde{x}(k) = \frac{1}{N} \sum_{m=0}^{N-1} \bar{X}(m) W_N^{-mk} \tag{1.8}$$

Realcija (1.8) definiše *inverznu diskretnu Fourierovu transformaciju* koja se dobija iz izvedene periodičke funkcije $f_T(t)$.

Relacije (1.7) i (1.8) nazivaju se *diskretna Fourierova sekvenca DFS* ili *diskretni Fourierov red.* Spektar zvjezdaste funkcije i izvedena periodička funkcija izvedeni su i objašnjeni u [1].

1.7.1 Računanje težinskih faktora W_N

Težinske faktore (engl. twiddle) W_N računamo preko sljedeće relacije

$$W_N^r = e^{jr\theta} = \cos(r\theta) + j\sin(r\theta), \qquad \theta = \frac{2\pi}{N}, \qquad j = \sqrt{-1},$$

gdje je N eksponent broja 2. Svaki put kada budemo rješavali sekvencu dijelit ćemo je na dva dijela, pa će nam biti potrebno samo da izračunamo $N/2$ težinskih faktora W_N. Ispod je dat algoritam u Matlab-u za računanje težinskih faktora W_N.

Algoritam 1.1. Računanje težinskih faktora W_N

```
w = zeros(1,N/2);                    % alociramo memoriju za smještanje
    Theta = 2*pi/N;                  % težinskih faktora u niz
    for twiddle = 1:(N/2)
        w(twiddle) = exp(-1i*Theta*(twiddle-1)); % popunjavanje niza
    end                                          % težinskim faktorima
```

1.7.2 Kompleksni brojevi

Podsjetimo se da jedno kompleksno sabiranje sadrži dva realna sabiranja, kao što je prikazano u jednačini (1.9), a jedno komplesno množenje sadrži četiri realna množenja i dva realna sabiranja, kao što je prikazano u jednačini (1.10). Neka kompleksni brojevi z_1 i z_2 imaju oblik $z_1 = a + jb$ i $z_2 = c + jd$. Tada je:

$$z_1 + z_2 = (a + jb) + (c + jd) = (a + c) + j(b + d), \tag{1.9}$$

$$z_1 \times z_2 = (a + jb) \times (c + jd) = (a \times c - b \times d) + j(a \times d + b \times c). \tag{1.10}$$

Operaciju sabiranja ili množenja brojeva sa pomičnim zarezom nazvat ćemo flopom (engl. **f**loating - **p**oint **o**peration, flop). Jedan od dva faktora koja učestvuju u kompleksnom množenju u toku računanja FFT-a je težinski faktor W_N. Na primjer, ako je $W_N^l = c + jd$, možemo prvo izračunati i spasiti u memoriju $\delta = d + c$ i $\gamma = d - c$, koji predstavljaju međurezultate. Sa δ, γ i realnim dijelom c sačuvanim svako kompleksno množenje u FFT koje uključuje $x_l = a + jb$ i $W_N^l = c + jd$ može se izračunati koristeći tri realna množenja i tri realna sabiranja/oduzimanja, kao što je pokazano u jednačini (1.11).

$$\begin{aligned} m_1 &= (a+b)\times c \\ m_2 &= \delta \times b \\ m_3 &= \gamma \times a \\ Re\,(\lambda) &= m_1 - m_2 \\ Im\,(\lambda) &= m_1 + m_3 \end{aligned} \tag{1.11}$$

Ako uporedimo jednačinu (1.11) sa jednačinom (1.10), ukupno broj flopova u jednačini (1.11) je šest, odnosno jedno množenje je zamjenjeno sa jednim sabiranjem/oduzimanjem.

1.7.3 Računanje kompleksnosti algoritma

Kompleksnost algoritma (engl. arithmetic cost) definisat ćemo sa redom rasta vremena izvršavanja u funkciji od veličine ulaza.

Algoritmi za računanje FFT-a su rekurzivni, odnosno rješenje sekvence dužine N predstavlja rezultat rješenja dvije podsekvence dužine $N/2$, zajedno sa kombinovanjem njihovih rješenja da bi dobili rješenje originalne sekvence dužine N. Da bi odredili kompleksnost algoritma potrebno je da odredimo funkciju $T(N)$, gdje je

$$T(N) = \begin{cases} 2T\left(\frac{N}{2}\right) + \beta N, & N = 2^k > 1, \\ \gamma, & N = 1. \end{cases} \tag{1.12}$$

U ovoj jednačini βN predstavlja kompleksnost kombinacije rješenja dvije podsekvence. Neki algoritmi za dobijanje krajnje originalne sekvence dužine N rješavajući α sekvenci dužine N/α i kombinujući njihove rezultate da bi dobili rješenje originalne sekvence. U ovom slučaju βN predstavlja kompleksnost kombinovanja rješenje α sekvenci dužine N/α.

$$T(N) = \begin{cases} \alpha T\left(\frac{N}{\alpha}\right) + \beta N, & N = \alpha^k > 1, \\ \gamma, & N = 1. \end{cases} \tag{1.13}$$

- **Općenito rješavanje T(N)**

Ako je:

$$T(N) = \begin{cases} aT\left(\frac{N}{c}\right) + bN, & N = c^k > 1, \\ b, & N = 1, \end{cases}$$

gdje su $a \geq 1$, $c \geq 2$ i $b > 0$ konstante, tada je rješenje kompleksnosti algoritma jednako

$$T(N) = \begin{cases} \frac{bc}{c-a}N, & a < c, \\ bN\log_c N + b(N), & a = c, \\ \frac{ab}{a-c}N^{\log_c a}, & a > c. \end{cases}$$

Dokaz 1. Pretpostavimo da je $N = c^k > 1$ za $c \geq 2$:

$$\begin{aligned}
T(N) &= aT\left(\frac{N}{c}\right) + bN \\
&= a\left(aT\left(\frac{N}{c^2}\right) + b\left(\frac{N}{c}\right)\right) + bN \\
&= a^2T\left(\frac{N}{c^2}\right) + ab\left(\frac{N}{c}\right) + bN \\
&= a^2\left(aT\left(\frac{N}{c^3}\right) + b\left(\frac{N}{c^2}\right)\right) + ab\left(\frac{N}{c}\right) + bN \\
&= a^3T\left(\frac{N}{c^3}\right) + a^2b\left(\frac{N}{c^2}\right) + ab\left(\frac{N}{c}\right) + bN \qquad (1.14) \\
&\vdots \\
&= a^kT\left(\frac{N}{c^k}\right) + a^{k-1}b\left(\frac{N}{c^{k-1}}\right) + a^{k-2}b\left(\frac{N}{c^{k-2}}\right) + \ldots + ab\left(\frac{N}{c}\right) + bN \\
&= a^kT(1) + a^{k-1}bc + a^{k-2}bc^2 + \ldots abc^{k-1} + bc^k \\
&= a^kb + a^kb\left(\frac{c}{a}\right) + a^kb\left(\frac{c}{a}\right)^2 + \ldots + a^kb\left(\frac{c}{a}\right)^{k-1} + a^kb\left(\frac{c}{a}\right)^k \\
&= a^kb\sum_{i=0}^{k}\left(\frac{c}{a}\right)^i.
\end{aligned}$$

Ako je $a = c$, tada je $\frac{c}{a} = 1$ i $N = c^k = a^k$, odnosno vrijedi da je $k = \log_c N$. Jednačina (1.14) postaje

$$T(N) = a^kb\sum_{i=0}^{k}\left(\frac{c}{a}\right)^i = Nb\sum_{i=0}^{k}1 = Nb(k+1) = bN\log_c N + bN.$$

Ako je $a > c$, tada je $\frac{c}{a} < 1$ i $\lim_{i\to\infty}\left(\frac{c}{a}\right)^i = 0$. $N = c^k$ i vrijedi da je $k = \log_c N$. Jednačina (1.12) postaje

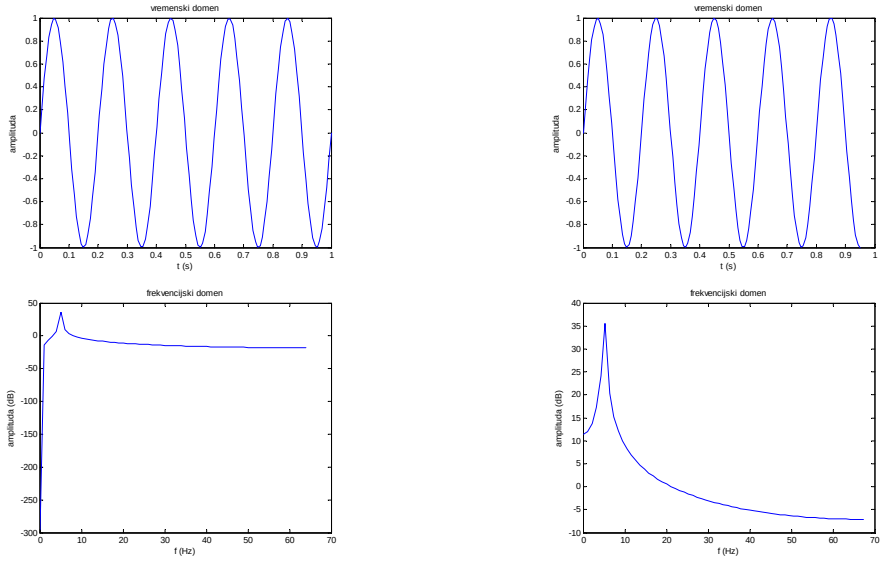
$$\begin{aligned}
T(N) &= a^kb\sum_{i=0}^{k}\left(\frac{c}{a}\right)^i = a^kb\left(\frac{1-(c/a)^{k+1}}{1-(c/a)}\right) \\
&\approx a^kb\left(\frac{a}{a-c}\right) = a^{\log_c N}\left(\frac{ab}{a-c}\right) \\
&= \frac{ab}{a-c}N^{\log_c a}.
\end{aligned}$$

Ako je $a < c$, tada je $\frac{c}{a} > 1$ i dobijemo:

$$
\begin{aligned}
T(N) &= a^k b \sum_{i=0}^{k} \left(\frac{c}{a}\right)^i \\
&= a^k b + a^{k-1}bc + a^{k-2}bc^2 + \ldots + abc^{k-1} + bc^k \\
&= c^k b \left(\frac{a}{c}\right)^k + c^k b \left(\frac{a}{c}\right)^{k-1} + \ldots + c^k b \left(\frac{a}{c}\right) + c^k b \\
&= c^k b \sum_{i=0}^{k} \left(\frac{a}{c}\right)^i \\
&= Nb \left(\frac{1 - (a/c)^{k+1}}{1 - (a/c)}\right) \\
&\approx bN \left(\frac{c}{c-a}\right) = \frac{bc}{c-a} N.
\end{aligned}
$$

1.8 Prozorske funkcije

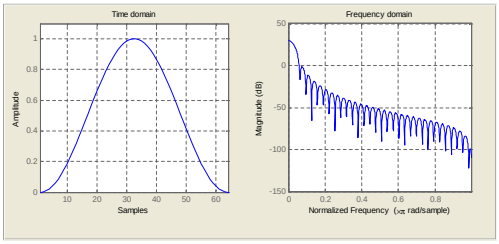
Prozorske funkcije (engl. window) je tehnika koja se koristi da bi minimizirali efekat curenja (engl. leakage). Ovaj efekat se dešava kada se računa FFT neperiodičnog diskretiziranog signala. Curenje rezultuje u energiji signala tako što se "razlije" u širokom opsegu frekvencija umjesto da se nalazi u ograničenom frekventnom opsegu. Na slici 1.4 je prikazan razlika između periodičnog sinusnog signala i odgovarajućeg FFT-a (lijevo) i neperiodičnog signala i odgovarajućeg FFT-a sa efektom curenja (desno).



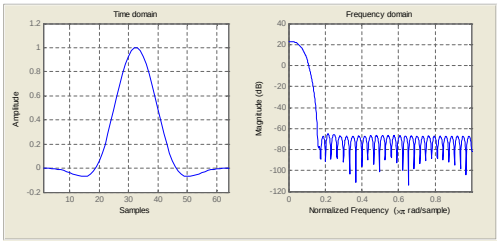
Slika 1.4. *Efekat curenja*

Da bi ispravili ovaj problem primjenjujemo odgovarajuću funkciju prozora. Kada se ne primjeni odgovarajući prozor greške možemo uočiti u amplitudi, frekvenciji ili u obliku spektra. Svaka vrsta prozora utiče na spektar na različit način. Tokom vremena, predloženi su mnogi prozori svaki sa svojim prednostima i nedostacima. Neke vrste tih prozora su: pravougaoni prozor, "Hammingov" prozor, "Hanningov" prozor, "Bartlettov" prozor, "Gaussov" prozor, "Blackmanov" prozor kao i mnogi drugi. Neki poboljšavaju frekventnu rezoluciju, to jest olakšavaju nam da otkrijemo tačnu centralnu frekvenciju, dok neki poboljšavaju tačnost amplitude, odnosno dobijamo tačnije podatke pri mjerenje amplitude na centralnoj frekvenciji. Za svaku specifičnu primjenu

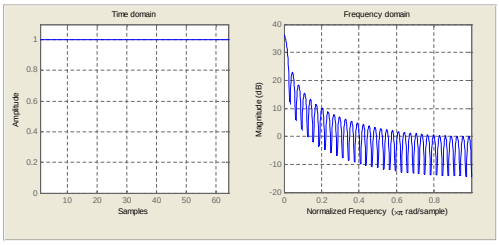
treba odabrati odgovarajući prozor. Prozori koje smo korstili pri analizi FFT-a predstavljeni su na sljedećoj slici.



(a) "Hanningov" prozor



(b) "Flat Top" prozor



(c) Pravougaoni prozor

Slika 1.5. *Vremenski domeni i spektar prozorskih funkcija za $N = 64$ za a) "Hanningov" prozor, b) "Flat Top" prozor i c) pravougaoni prozor*

"Hanningov" prozor daje glatki prelaz do nule kako se približavaju krajevi signala. Obezbjeđuje dobru frekventnu rezoluciju po cijenu nešto manje tačnosti amplituda. Slabljenje prvog bočnog luka (engl. lobe) iznosi -31 dB. Karakteristika "Flat Top" prozora jeste da ima ravniji vrh u frekventnom domenu i obezbjeđuje bolju amplitudnu tačnost, ali zato lošiju frekventnu rezoluciju. "Flat Top" prozor ima slabljenje prvog bočnog luka dosta niže i iznosi -44 dB. Za pravougaoni prozor slabljenje prvog bočnog luka iznosi -13 dB. Pravougaoni prozor se koristi tamo gdje znamo da neće doći do efekta curenja. Većina mjerenja zahtijeva korištenje "Hanningovog" prozora ili "Flat Top" prozora. Izborom između ova dva prozora pravimo kompromis između rezolucije frekvencije i tačnosti amplitude. Ukoliko je frekventni domen filtra uži, analizator može bolje da razluči dvije spektralne linije na bliskom rastojanju. Ako je frekventni domen širi i ravniji, mjerenje amplitude će biti tačnije i naravno smanjuje se frekventna rezolucija.

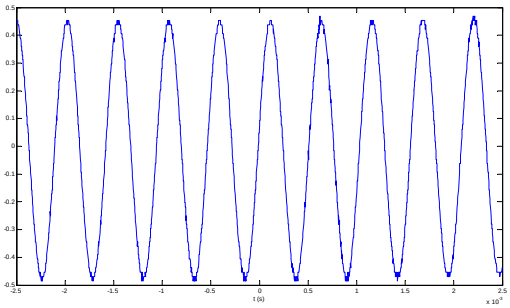
Na slici 1.6 je prikazan digitalni osciloskop (spektralni analizator), koji smo koristili da bi pokazali primjenu različitih prozora, kao i neke osnovne osobine signala. Prvo smo dodali modul za mjerenje i pohranjivanje (HP 54657A) i time smo dobili dodatne matematske mogučnosti obrade signala, uključujući i FFT. Nakon što smo povezali osciloskop sa generatorom signala pritiskom na "Autoscale" priključeni kanali se prikazuju na ekranu spektralnog analizatora. Povezivanje osciloskopa sa računarom može se obaviti na dva načina: serijskom vezom (RS232) i GPIB vezom (IEEE 488.2). Koristili smo komunikaciju preko GPIB veze. Osciloskop smo preko GPIB BUS-a povezali na USB port računara. Na računaru smo koristili programski paket *HP Benchlink Scope* da bi ostvarili prenos podataka sa osciloskopa.

Slika 1.6. *HP 54645 osciloskop*

Eksperiment 1: *Objašnjenje pojmova brzine sampliranja, frekventne rezolucije i curenja spektra za ulazni sinusni signal*

Ovo je jednostavan eksperiment koji pokazuje vezu između efektivne brzine sampliranja i dobivene frekventne rezolucije za spektralnu analizu koristeći FFT. Također je prikazano curenje spektra za odgovarajuće prozore.

Potrebno je sinusni signal amplitude 1 V i frekvencije 2 kHz spojiti na jedan kanal osciloskopa. Pritiskom na dugme "Autoscale" na osciloskopu dobivamo sljedeći signal.



Slika 1.7. *Sinusni signal*

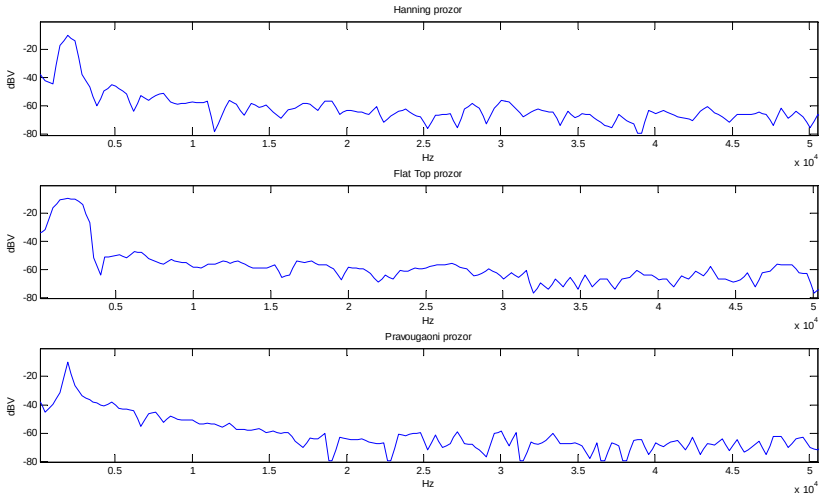
Kao što je poznato iz teorije FFT-a bolja frekventna rezolucija se dobije uzimanje većeg broja uzoraka u samom spektru signala. Kako je period uzorkovanja u spektru jednak frekvenciji sampliranja u vremenskom domenu, tj. efektivnoj brzini sampliranja to je za bolju rezoluciju spektralnog sadržaja signala bolje uzeti nižu frekvenciju sampliranja. Međutim, smanjujući brzinu sampliranja možemo narušiti Nyquist-ov uslov sa sampliranje signala čime bi došlo do pojave aliasinga. Efektivna brzina uzorkovanja je recipročna vrijednost vremena između uzoraka i zavisi od podešenja "time/div" na osciloskopu. Za HP 54600 seriju osciloskopa, efektivna brzina uzorkovanja je data sa :

$$f_{eff} = \frac{100}{time/div} \; [Sa/s] \, .$$

Nakon dobivanja signala ulazimo u FFT meni i podešavamo parametre FFT transformacije za sljedeće tri vrste prozora: "Hanning", "Flat Top" i pravougaoni prozor i dobivamo spektre signala koji su dati na slici 1.8. Općenito, prozori sa manjim curenjem spektra imaju i manju frekventnu rezoluciju, što se i najbolje vidi mjerenjem širine glavnog luka (engl. lob).

Efektivna brzina sampliranja	Referentni level	Frekvencija centra	Frekvencija mjernog područja
500 kSa/s	0 dBV	122,1 kHz	244,1 kHz

Tablica 1. *FFT meni*



Slika 1.8. *Spektri signala*

Vertikalna osa je logaritamska, prikazana je u dBV. Simbol decibel se uobičajno predstavlja sa sufiksom, koji označava koju referentnu veličinu koristimo.

Sa slike se vidi da je korištenjem pravougaonog prozora dobijena bolja rezolucija, ali i veće curenje spektra. Sada možemo izmjeriti širinu glavnog luka. Ova mjerenja ćemo izvršiti pritiskom na dugme "Cursor" i izborom komande "Find Peaks" pomoću kojeg se mjeri osnovna frekvencija sinusoide. Širina glavnog luka se mjeri pomoću pokazivača nivoa v i frekvencije f. Pokazivač $v_1$se postavi na vrh luka, a v_2 31 dB niže za "Hanningov" prozor, 44 dB niže za "Flat Top" prozor i 13 dB niže za pravougaoni prozor. Pokazivači f_1 i f_2 se postavljaju u tačke gdje v_2 sijeće linije glavnog luka. U tabeli 2 date su vrijednosti širine glavnog luka za odabranu prozorsku funkciju.

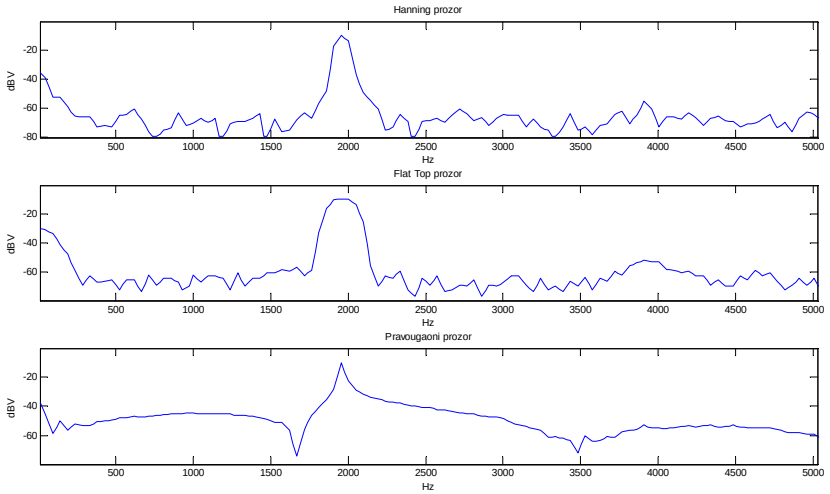
Vrsta prozora:	Hanning	Flat Top	Pravougaoni
Širina glavnog luka (kHz):	1,146	1,623	669,5

Tablica 2. *Širina glavnog luka*

Koristeći dugme "Time/Div" smanjit ćemo efektivnu brzinu sampliranja na 50 kSa/s i 10 kSa/s i ponoviti proceduru.

Efektivna brzina sampliranja	Referentni level	Frekvencija centra	Frekvencija mjernog područja
50 kSa/s	0 dBV	12,21 kHz	24,41 kHz

Tablica 3. *FFT meni*



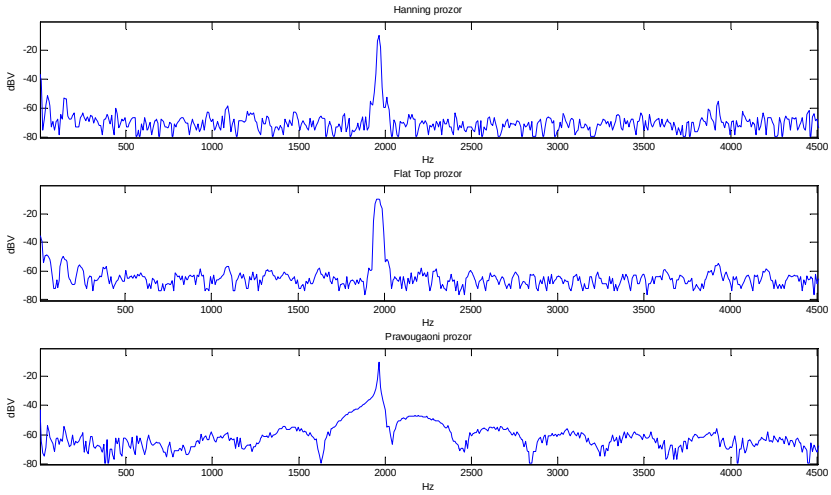
Slika 1.9. *Spektri signala*

Vrsta prozora:	Hanning	Flat Top	Pravougaoni
Širina glavnog luka (Hz):	121,1	240,3	121,1

Tablica 4. *Širina glavnog luka*

Efektivna brzina sampliranja	Referentni level	Frekvencija centra	Frekvencija mjernog područja
10 kSa/s	0 dBV	2,441 kHz	4,883 kHz

Tablica 5. *FFT meni*



Slika 1.10. *Spektri signala*

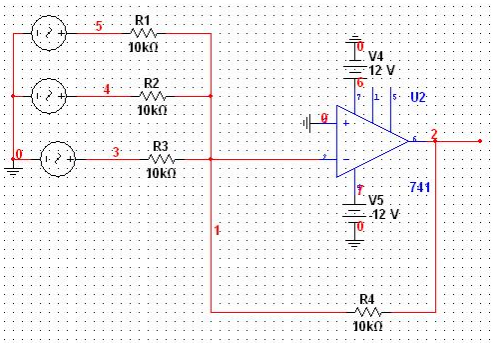
Vrsta prozora:	Hanning	Flat Top	Pravougaoni
Širina glavnog luka (Hz):	68,86	78,2	54,36

Tablica 6. *Širina glavnog luka*

Vidi se da je smanjivanjem efektivne brzine sampliranja poboljšana FFT analiza, odnosno dobili smo bolju frekventnu rezoluciju, ali također da bi izbjegli aliasing potrebno je paziti da je efektivna brzina sampliranja veća od Nyguistove frekvencije. Zavisno od izbora prozora smanjuje se i spektar curenja, odnosno najmanje je vidljiv kod Hanningovog prozora, a najviše kod pravougaonog prozora. Osobina curenja spektra kod pravougaonog prozora zanemaruje njegovu dobru osobinu frekventne rezolucije.

Eksperiment 2: *Zbir sinusoida*

U ovom eksperimentu pokazat ćemo kako se FFT može koristiti u analizi spektralnog sadržaja signala koji se sastoji od zbira tri sinusoide. Potrebno je spojiti kolo dato na slici 1.11.



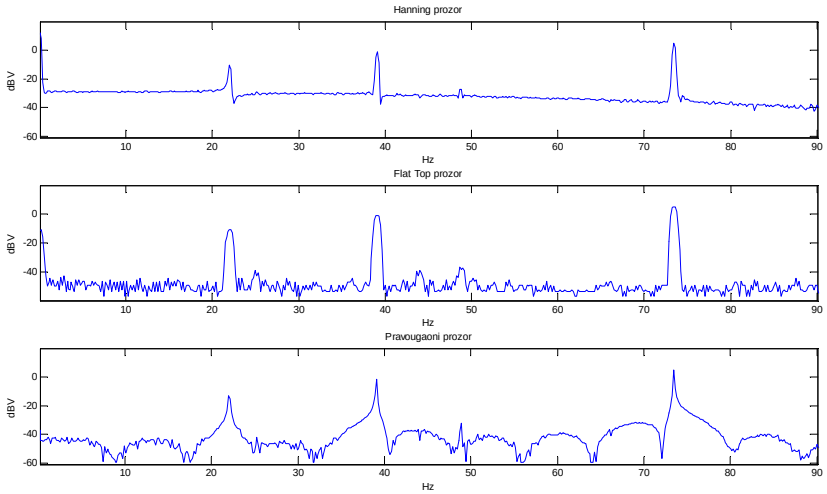
Slika 1.11. *Šema za spajanje kola*

Generatore funkcija smo namjestili da generišu sinusoide frekvencija: 75,21 Hz, 40 Hz i 23 Hz. Parametre za FFT smo podesili prema tabeli 7.

Efektivna brzina sampliranja	Referentni level	Frekvencija centra	Frekvencija mjernog područja
200 kSa/s	0 dBV	48,83 kHz	97,66 kHz

Tablica 7. *FFT meni*

Nakon što smo dobili spektre signala, slika 1.12, odredili smo dobijene frekvencije i dobili sljedeće rezultate: 75,2 Hz, 40,4 Hz i 22,41 Hz. Dobijene frekvencije se podudaraju sa frekvencijama na generatorima funkcija.



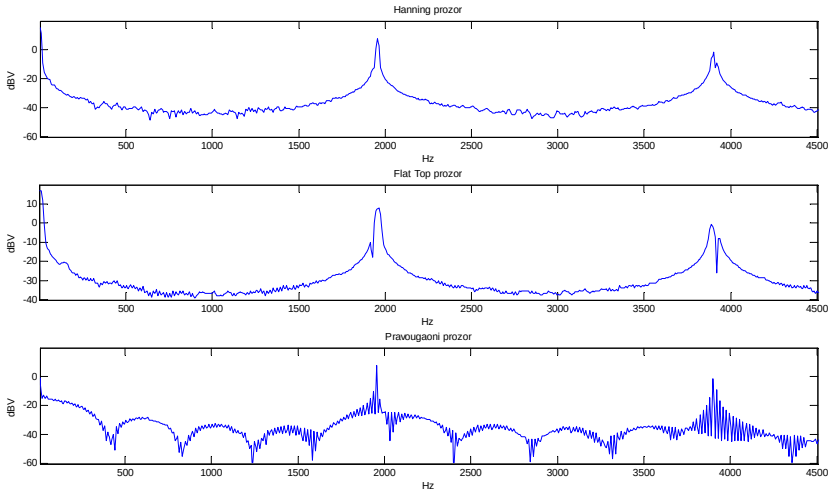
Slika 1.12. *Spektri signala*

Eksperiment 3: *Aliasing*

Na osciloskop smo doveli četvrtke frekvencije 2 kHz i amplitude 5,6 V. Parametre za FFT meni smo podesili prema tabeli 8 i dobili odgovarajuće spektre, slika 1.13.

Efektivna brzina sampliranja	Referentni level	Frekvencija centra	Frekvencija mjernog područja
10 kSa/s	20 dBV	2,441 kHz	4,883 kHz

Tablica 8. *FFT meni*

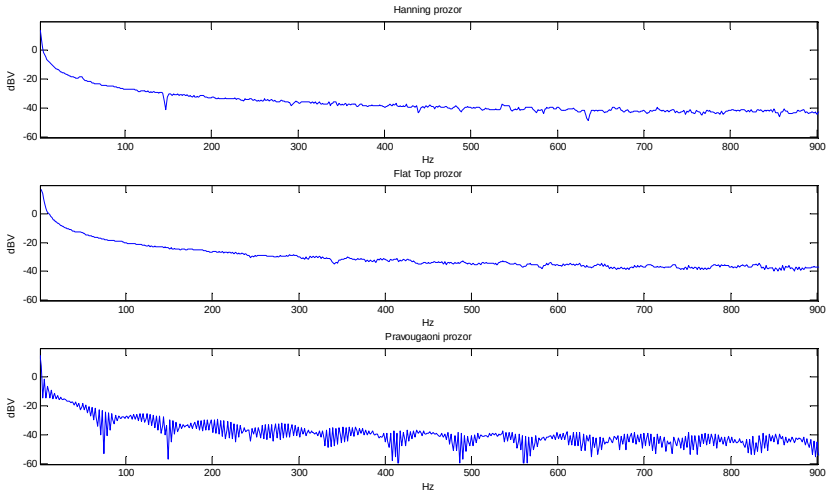


Slika 1.13. *Spektri signala*

Kao što smo i očekivali, nakon korištenja opcije "Find Peaks" dobili smo da se svaki sljedeći vrh nalazi na udaljenosti od 2 kHz. Da bi pokazali djelovanje aliasinga odlučili smo smanjiti efektivnu brzinu sampliranja i podesili smo parametre FFT-a prema tabeli 9.

Efektivna brzina sampliranja	Referentni level	Frekvencija centra	Frekvencija mjernog područja
2 kSa/s	20 dBV	488,3 kHz	976.6 kHz

Tablica 9. *FFT meni*



Slika 1.14. *Spektri signala*

Nakon što smo koristili opciju "Find Peaks" dobili smo rezultate koje se ne podudaraju sa traženim, odnosno dokazali smo djelovanje aliasinga.

U ovom poglavlju korištena je literatura [1] ,[3],[4],[10] i [11].

2 Sekvencijalni FFT algoritam

2.1 Algoritam podijeli i riješi za dva osnovna FFT algoritma

Svrha ovog poglavlja je objasniti osnovne ideje brze Fourierove transformacije (engl. **F**ast **F**ourier **T**ransform, FFT). Ovo će nam biti osnova za sljedeća podpoglavlja, gdje ćemo razmotriti mnoge teme vezane za efikasnost i implementaciju različith algoritama.

Tri glavna koraka principa podijeli i riješi (engl. the Divide and Conquer Paradigm) su:

1. podijeliti sekvencu u dvije ili više podsekvenci,
2. riješiti svaku sekvencu rekurzivno pomoću istog algoritma i postaviti granične uslove da bi završili sa rekurzijom kada dužina podsekvence postane dovoljno mala i
3. sastaviti rješenje originalne sekvence kombinirajući rješenja podsekvence.

FFT algoritam baze 2 je rekurzivni algoritam koji se sastoji od dijeljenja dobivene sekvence (i svake podsekvence) u dvije podsekvence upola manje dužine. Postoje dva uobičajno korištena FFT algoritma koja se razlikuju u načinu kako su podsekvence definisane. To su brza Fourierova transformacija sa decimiranjem po vremenu (engl. **D**ecimation **I**n **T**ime, DIT) i brza Fourierova transformacija sa decimiranjem po učestanosti (engl. **D**ecimation **I**n **F**requency, DIF).

Intuitivno je jasno da će algoritam podijeli i riješi imati najbolje rezultate ako je dužina sekvence N, eksponent broja 2, jer podjela sekvenci u sukcesivno manje grupe može se nastaviti dok ne ostane samo jedan par u grupi. Također, postoji mnogo slučajeva kada N nije eksponent broja 2 i tada je potrebno modifikovati algoritam. Ove modifikacije će biti razmotrene detaljno u kasnijim podpoglavljima. U ovom podpoglavlju polazi se od pretpostavke da je $N = 2^n$.

Da bi pojednostavili izvođenja zanemarit ćemo član $\frac{1}{N}$ u formuli za DFT. Formula koju ćemo koristiti u izvođenju je

$$X_r = \sum_{k=0}^{N-1} x_k W_N^{rk}, \quad r = 0, 1, \ldots, N-1. \tag{2.1}$$

2.1.1 DIT FFT po bazi 2

Relaciju (2.1) ćemo napisati u obliku:

$$X_r = \sum_{k=0}^{\frac{N}{2}-1} x_{2k} W_N^{r(2k)} + W_N^r \sum_{k=0}^{\frac{N}{2}-1} x_{2k+1}\, W_N^{r(2k)}, \quad r = 0, 1, \ldots, N-1. \tag{2.2}$$

Uzimajući u obzir da je $\omega_{\frac{N}{2}} = \omega_N^2$ dobijemo:

$$X_r = \sum_{k=0}^{\frac{N}{2}-1} x_{2k} W_{\frac{N}{2}}^{rk} + W_N^r \sum_{k=0}^{\frac{N}{2}-1} x_{2k+1}\, W_{\frac{N}{2}}^{rk}, \quad r = 0, 1, \ldots, N-1. \tag{2.3}$$

Pošto vrijedi da je $W_{\frac{N}{2}}^{rk} = W_{\frac{N}{2}}^{(r+\frac{N}{2})k}$ neophodno je samo izračunati sume za $r = 0, 1, \ldots, N/2 - 1$. Također, svaka suma u relaciji (2.3) može se protumačiti kao DFT dužine $N/2$. Prvi dio uključuje sume sa parnim indeksima $\{x_{2k} | k = 0, 1, \ldots, N/2 - 1\}$, a drugi dio sume sa neparnim indeksima $\{x_{2k+1} | k = 0, 1, \ldots, N/2 - 1\}$. Odatle i potiče sam naziv učestanost po vremenu. Ako definiramo $y_k = x_{2k}$ i $z_k = x_{2k+1}$ u jednačini (2.3) i

podijelimo na dvije podsekvence od kojih svaka ima identičnu formu kao u jednačini (2.1) sa N zamjenjenim sa $N/2$ dobijamo :

$$Y_r = \sum_{k=0}^{\frac{N}{2}-1} y_k W_{\frac{N}{2}}^{rk}, \quad r = 0, 1, \ldots, N/2 - 1 \tag{2.4}$$

i

$$Z_r = \sum_{k=0}^{\frac{N}{2}-1} z_k W_{\frac{N}{2}}^{rk}, \quad r = 0, 1, \ldots, N/2 - 1. \tag{2.5}$$

Nakon rekurzivnog rješavanja ove dvije podsekvence, rješenje originalne sekvence dužine N se dobije koristeći (2.3). Prvih $N/2$ članova se dobije pomoću:

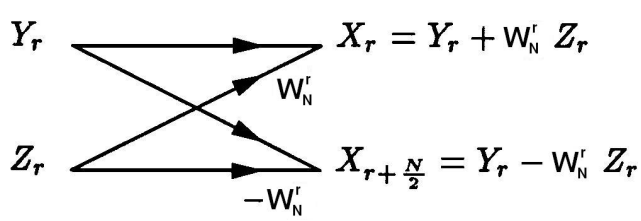
$$X_r = Y_r + W_N^r Z_r, \quad r = 0, 1, \ldots, N/2 - 1, \tag{2.6}$$

i koristeći činjenicu da su $W_N^{\frac{N}{2}+r} = -W_N^r$ i $W_{\frac{N}{2}}^{\frac{N}{2}} = 1$ drugi dio sume možemo izračunati na sljedeći način:

$$\begin{aligned} X_{r+\frac{N}{2}} &= \sum_{k=0}^{\frac{N}{2}-1} y_k W_{\frac{N}{2}}^{(r+\frac{N}{2})k} + W_N^{\frac{N}{2}+r} \sum_{k=0}^{\frac{N}{2}-1} z_k W_{\frac{N}{2}}^{(r+\frac{N}{2})k} \\ &= \sum_{k=0}^{\frac{N}{2}-1} y_k W_{\frac{N}{2}}^{rk} - W_N^r \sum_{k=0}^{\frac{N}{2}-1} z_k W_{\frac{N}{2}}^{rk} \\ &= Y_r - W_N^r Z_r, \quad r = 0, 1, \ldots, N/2 - 1. \end{aligned} \tag{2.7}$$

Primjećujemo da se jednačine (2.6) i (2.7) mogu primjeniti na bilo koju sekvencu gdje je N eksponent broja 2. Kada je sekvenca dužine eksponenta broja 2, dvije podsekvence definisane su jednačinama (2.4) i (2.5) (kao i sljedeće podsekvence ≥ 2) mogu se riješiti po istom principu.

Računanje predstavljeno jednačinama (2.6) i (2.7) se u literaturi često odnosi na "Cooley - Tukey butterfly" algoritam, kao što je prikazano na slici 2.1.



Slika 2.1. *"Cooley - Tukey butterfly" algoritam*

- **Analiza kompleksnosti algoritma**

Računanje kompleksnosti algoritma DIT FFT po bazi 2 dužine N označit ćemo sa $T(N)$. Prema jednačinama (2.6) i (2.7) potrebno je N kompleksnih sabiranja i $N/2$ kompleksnih množenja da bi izvršili transformaciju. Jedno kompleksno sabiranje se sastoji od dva realna sabiranje, a jedno kompleksno množenje se sastoji od tri realna množenja i tri realna sabiranja.

Potrebno je $2N$ flopa za N kompleksnih sabiranja i $3N$ flopa za $N/2$ kompleksnih množenja. Ukupno nam je potrebno $5N$ flopova da bi završili transformaciju poslije dvije podsekvence koje su upola manje dužine,

odnosno svaka podsekvenca ima kompleksnost algoritma od $T\left({}^{N}/_{2}\right)$. Prema tome kompleksnost algoritma predstavljeno sa $T(N)$ na osnovu relacije (1.12) ima sljedeće vrijednosti:

$$T(N) = \begin{cases} 2T(\frac{N}{2}) + 5N, & N = 2^n \geq 2, \\ 0, & N = 1, \end{cases}$$

odnosno, možemo napisati

$$T(N) = 5N \log_2 N.$$

2.1.2 DIF FFT po bazi 2

Kao što samo ime implicira, DIF FFT po bazi 2 je decimiranje članova izlazne frekventne sekvence u skup sa parnim indeksima $\{x_{2k} | k = 0, 1, \ldots, {}^{N}/_{2} - 1\}$ i u skup sa neparnim indeksima $\{x_{2k+1} | k = 0, 1, \ldots, {}^{N}/_{2} - 1\}$. Da bi definisali dvije podsekvence, jednačinu (2.1) ćemo napisati u obliku

$$\begin{aligned} X_r &= \sum_{l=0}^{\frac{N}{2}-1} x_l W_N^{rl} + \sum_{l=\frac{N}{2}}^{N-1} x_l W_N^{rl} = \sum_{l=0}^{\frac{N}{2}-1} x_l W_N^{rl} + \sum_{l=0}^{\frac{N}{2}-1} x_{l+\frac{N}{2}} \, W_N^{r\left(l+\frac{N}{2}\right)} \\ &= \sum_{l=0}^{\frac{N}{2}-1} \left(x_l + x_{l+\frac{N}{2}} \, W_N^{r\frac{N}{2}} \right) W_N^{rl}, \quad r = 0, 1, \ldots, N-1. \end{aligned} \tag{2.8}$$

Sume sa parnim indeksima dobijemo zamjenom $r = 2k$ u jednačinu (2.8)

$$\begin{aligned} X_{2k} &= \sum_{l=0}^{\frac{N}{2}-1} \left(x_l + x_{l+\frac{N}{2}} \, W_N^{kN} \right) W_N^{2kl} \\ &= \sum_{l=0}^{\frac{N}{2}-1} \left(x_l + x_{l+\frac{N}{2}} \right) W_{\frac{N}{2}}^{kl}, \quad k = 0, 1, \ldots, {}^{N}/_{2} - 1. \end{aligned} \tag{2.9}$$

Ako zamjenimo $Y_k = X_{2k}$ i $y_l = x_l + x_{l+\frac{N}{2}}$ u jednačini (2.9) dobijemo jednačinu prve podsekvence:

$$Y_k = \sum_{l=0}^{\frac{N}{2}-1} y_l W_{\frac{N}{2}}^{kl}, \quad k = 0, 1, \ldots, {}^{N}/_{2} - 1. \tag{2.10}$$

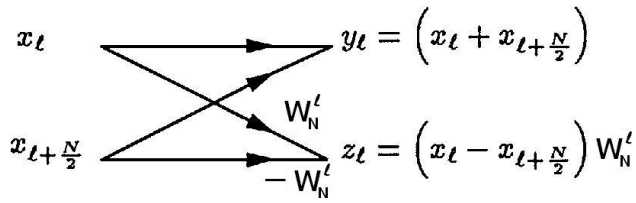
Slično, za sume sa neparnim indeksima zamjenom $r = 2k + 1$ u jednačinu (2.8) dobijamo:

$$\begin{aligned} X_{2k+1} &= \sum_{l=0}^{\frac{N}{2}-1} \left(x_l + x_{l+\frac{N}{2}} \, W_N^{(2k+1)\frac{N}{2}} \right) W_N^{(2k+1)l} \\ &= \sum_{l=0}^{\frac{N}{2}-1} \left(\left(x_l - x_{l+\frac{N}{2}} \right) W_N^{l} \right) W_{\frac{N}{2}}^{kl}, \quad k = 0, 1, \ldots, {}^{N}/_{2} - 1. \end{aligned} \tag{2.11}$$

Ako zamjenimo $Z_k = X_{2k+1}$ i $z_l = \left(x_l - x_{l+\frac{N}{2}} \right) W_N^{l}$ u jednačini (2.11) dobijemo jednačinu druge podsekvence:

$$Z_k = \sum_{l=0}^{\frac{N}{2}-1} z_l W_{\frac{N}{2}}^{kl}, \quad k = 0, 1, \ldots, N/2 - 1. \tag{2.12}$$

Primjećujemo da je u jednačinama (2.10) i (2.12) $Y_k = X_{2k}$ i $Z_k = X_{2k+1}$, odnosno nije potrebno više računanja da bi se dobilo originalno rješenje nakon što se riješe ove dvije podsekvence. Računanje y_l i z_l se u literatiri naziva "Gentleman - Sande butterfly" i prikazano je na slici 2.2.



Slika 2.2. *"Gentleman-Sande butterfly"*

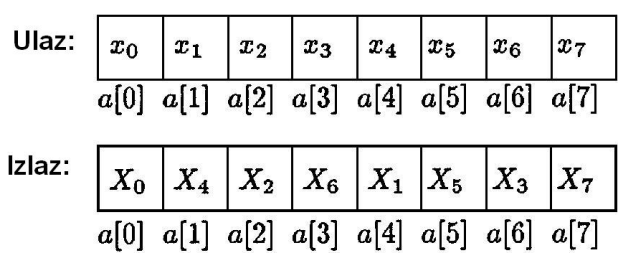
- **Analiza kompleksnosti algoritma**

Za računanje y_l i z_l potrebno je N kompleksnih sabiranja i $N/2$ kompleksnih množenja, koliko i iznosi kompleksnost algoritma i u DIT FFT po bazi 2 što smo diskutovali u ranijem podpoglavlju. Prema tome, kompleksnost algoritma iznosi:

$$T(N) = 5N \log_2 N.$$

2.2 DIF FFT po bazi 2 sa izlaznim podacima u bit - inverznom redosljedu

U praksi, računanje FFT-a se obavlja u - mjestu u jednodimenzionalnom nizu, gdje se nove vrijednosti upisuju preko starih vrijednosti. U - mjestu (engl. in place) znači da se u svakom koraku algoritma koriste rezultati samo iz prethodnog koraka. Na primjer, slika 2.2, implicira da se y_l upisuje preko x_l i da se z_l upisuje preko $x_{l+\frac{N}{2}}$. Posljedica ovoga je da je izlaz ispremješten; redosljed elemenata vektora X u nizu neće općenito odgovarati ulazima x. Na primjer, ako primjenjujemo DIF FFT na podatke x koji su smješteni u memoriji niza a kao rezultat imamo ispremještene podatke X u a kada se računanje završi, kao što je pokazano na slici 2.3.



Slika 2.3. *Ulaz x u nizu a je prepisan sa ispremještenim izlazom X*

Razmotrimo prvi korak DIT FFT sa bazom 2, koji je opisan sa algoritmom "Gentlemen - Sande butterfly" na slici 2.2. Podsjetimo se da definišući $y_l = x_l + x_{l+\frac{N}{2}}$ i $z_l = \left(x_l - x_{l+\frac{N}{2}}\right) W_N^l$ dobijene su jednačine (2.10) i (2.12).

Definicija 2.1. Ulazni podaci $x_0, x_1, \ldots, x_{N-1}$ se nalaze u prirodnom redosljedu ako su x_i i x_{i+1} smješteni u rastućem redosljedu u nizu a za sve $0 \leq i \leq N-2$. Slično, izlazni podaci $X_0, X_1, \ldots, X_{N-1}$ se nalaze u prirodnom redosljedu ako su X_i i X_{i+1} smješteni u rastućem redosljedu u nizu a za sve $0 \leq i \leq N-2$.

Mogućnost izvršenja FFT algoritma u - mjestu postoji zahvaljujući prvobitnom memorisanju uzoraka u bit - inverznom redosljedu. Ovaj redosljed se dobije tako što se indeksi uzoraka prvo napišu u binarnom zapisu, pa se zatim redosljed brojeva okrene za 180°, kao što je i pokazano u tabeli 10.

Prirodni redosljed	Binarni zapis	Binarno-inverzni zapis	Bit - inverzni redosljed
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Tablica 10. *Formiranje bit - inverznog redosljeda*

Na primjer, osam ulaznih podataka $x_0, x_1, \ldots, x_7$ na slici 2.3 se nalaze u prirodnom redosljedu, dok se izlazni podaci $X_0, X_1, \ldots, X_7$ nalaze u bit - inverznom redosljedu. U ovom popoglavlju je pretpostavljeno da su ulazni podaci x_i, $0 \leq i < N$ smješteni u jednodimenzionalnom nizu a u prirodnom redosljedu.

Podsjetimo se da je svaki podpodijeljeni korak definisan rekurzivno sa "Gentleman - Sande butterfly" algoritmom. Pošto nema koraka kombinacije iterativni algoritam ćemo izvesti dijeljenjem svake podsekvence uzastopno dok ne dostignemo dužinu sekvence dva.

Algoritam 2.1. Iterativni FFT algoritam sa decimiranjem po učestanosti po bazi 2 u Matlab-u

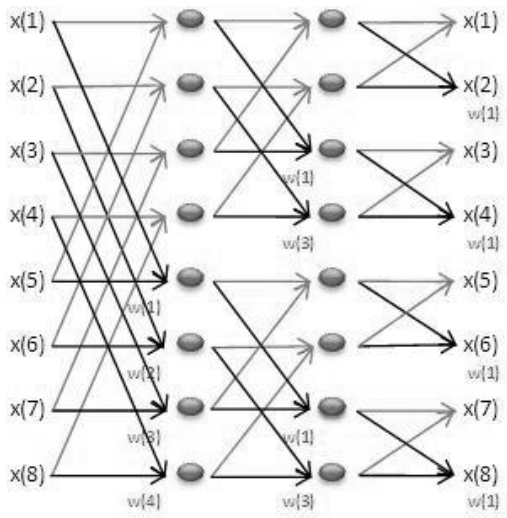
```
function [X] = dftuciter (x)
  NumOfProblems = 1;
  N = length(x);
  ProblemSize = N;

  w = zeros(1,ProblemSize/2);              % alociramo memoriju za smještanje
  Theta = 2*pi/ProblemSize;                % "težinskih" faktora u niz
  for twiddle = 1:(ProblemSize/2)
    w(twiddle)= exp(-1i*Theta*(twiddle-1));% popunjavanje niza "težinskim"
  end                                      % faktorima

  while (ProblemSize > 1)
    HalfSize = ProblemSize / 2;
      for K = 0:(NumOfProblems-1)          % primjenjujemo "Gentleman - Sande
        JFirst = K * ProblemSize + 1;      % butterfly" algoritam da bi
        JLast = JFirst + HalfSize-1;       % podijelili K-ti problema na
        Jtwiddle = 1;                      % dva dijela, postupak ovog
        for J = JFirst:JLast               % algoritma je objašnjen
          W = w(Jtwiddle);                 % u podpoglavlju 2.1.
          Temp = x(J);
          x(J) = Temp + x(J + HalfSize);
          x(J + HalfSize) = W * (Temp - x(J + HalfSize));
          Jtwiddle = Jtwiddle + NumOfProblems;
        end
      end
      NumOfProblems = 2 * NumOfProblems;    % sada imamo duplo vise, ali
      ProblemSize = HalfSize;               % duplo manjih problema
```

```
    end
                                        % izlazni podaci se nalaze u bit -
                                        % inverznom redosljedu
    X = bitrevorder(x);                 % podatke ispremjštamo da ih
                                        % dobijemo u prirodnom redosljedu
end
```

2.2.1 Primjer iterativnog DIF FFT po bazi 2

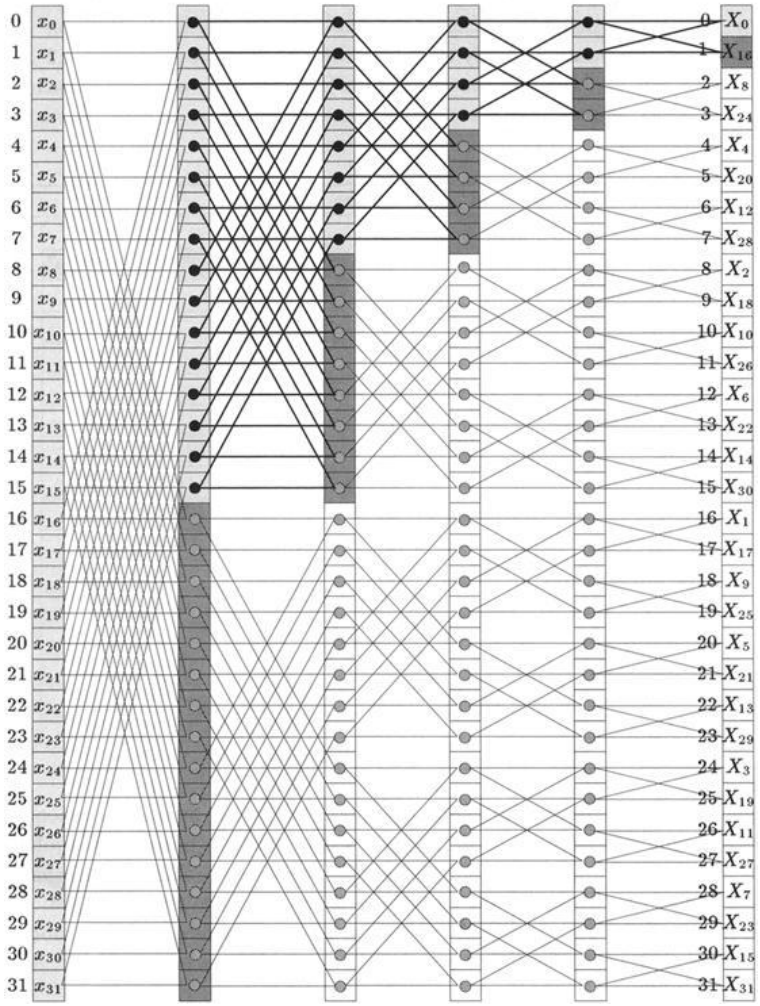
Općenito, za ulaznu sekvencu od $N = 2^n$ elemenata postoji tačno $N/2$ "butterfly" računanja u svakoj $\log_2 N$ fazi.

Za $N = 2^3 = 8$ DIF FFT se sastoji od tri faze "butterfly" računanja. Transformacija će se završiti kada se izvedu sva "butterfly" računanja u svakoj fazi. Prvo ćemo izračunati četiri odgovarajuće težinske faktore, a nakon toga ulazimo u "while" petlju. U "while" petlji ćemo primijeniti "Gentleman - Sande butterfly" algoritam. Kao i što smo objasnili u podpoglavlju 2.1 prvo ćemo rastaviti sekvencu na dva dijela. Nakon prve "butterfly" faze u prvoj polovini sekvence će se nalaziti prvi član sekvence, koji ćemo označiti sa x_l i koji ćemo sabrati sa x_{l+k} za $k = 1, 2, 3$ i 4, a u drugoj polovini sekvence ćemo oduzeti x_l i x_{l+k} i pomnožiti sa odgovarajućim težinskim faktorom za $k = 1, 2, 3$ i 4. Da bi izračunali drugu "butterfly" fazu, potrebno je prvo da podijelimo sekvencu na dva dijela i onda da ponovimo proceduru. Za računanje treće "butterfly" faze, nakon dijeljenja sekvence, dobit ćemo da imamo četiri sekvence od kojih se svaka sastoji od jednog para, odnosno od dva člana i primijeniti ćemo odgovarajuću proceduru. Nakon što smo završili sa "butterfly" računanjem dobijena sekvenca se nalazi u bit - inverznom redosljedu i potrebno je primijeniti funkciju *bitrevorder* da bi dobili rezultate u prirodnom redosljedu.



Slika 2.4. *"Butterfly" računanje za* $N = 8$

Za $N = 2^5 = 32$ DIF FFT se sastoji od pet faze "butterfly" računanja.



Slika 2.5. *"Butterfly" računanje za* $N = 32$

2.3 DIF FFT po bazi 2 sa bit - inverznim ulazom

Na ulaz je potrebno da dovedemo sekvencu zapisanu u bit - inverznom redosljedu prije nego što počnemo sa računanjem, kao što je to pokazano na slici 2.6.

Ulaz:	x_0	x_4	x_2	x_6	x_1	x_5	x_3	x_7
	$a[0]$	$a[1]$	$a[2]$	$a[3]$	$a[4]$	$a[5]$	$a[6]$	$a[7]$
Izlaz:	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
	$a[0]$	$a[1]$	$a[2]$	$a[3]$	$a[4]$	$a[5]$	$a[6]$	$a[7]$

Slika 2.6. *Bit - inverzni ulaz i prirodni redosljed podataka na izlazu*

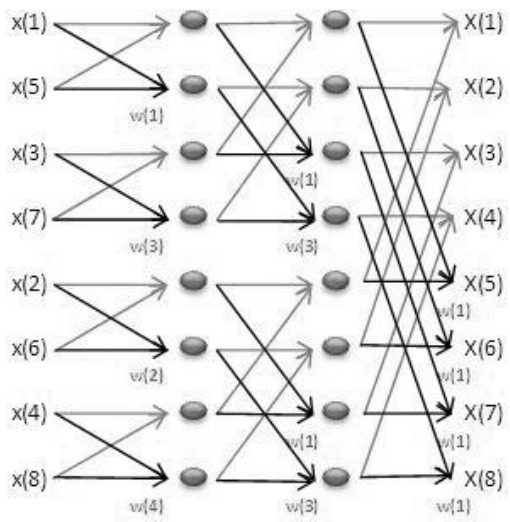
Također, i težinski faktori W_N moraju se nalaziti u bit - reverznom redosljedu. Iterativni FFT algoritam sa decimiranjem po učestanosti po bazi 2 prikazan je ispod.

Algoritam 2.2. Iterativni FFT algoritam sa decimiranjem po učestanosti po bazi 2 sa bit - inverznim ulazom

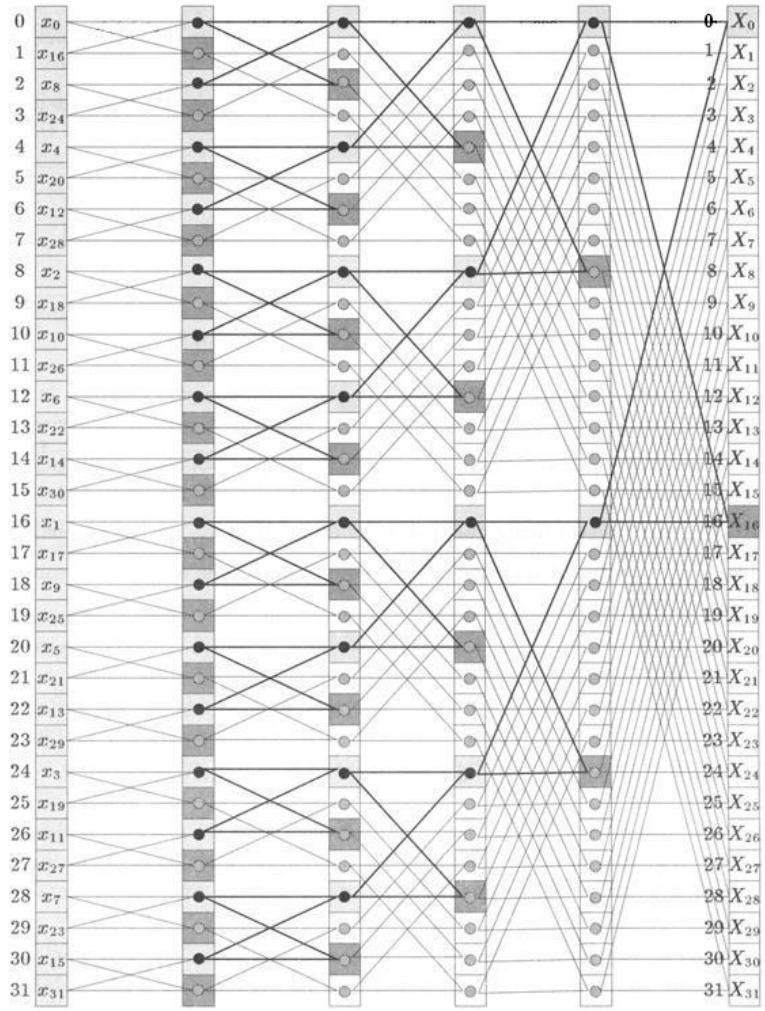
```
function [X] = dftuciterRN (x)
    N = length(x);
    NumOfProblems = 1;
    ProblemSize = N;
    Distance = 1;

    X = bitrevorder(x);                    % ulazne podatke postavimo u bit-
                                           % inverzni redosljed
    w = zeros(1,ProblemSize/2);            % alociramo memoriju za smještanje
    Theta = 2*pi/ProblemSize;              % težinskih faktora u niz
    for twiddle = 1:(ProblemSize/2)
        w(twiddle) = exp(-1i*Theta*(twiddle-1)); % popunjavanje niza
    end                                    % težinskim faktorima

    w = bitrevorder(w);                    % težinske faktore postavimo u
                                           % bit - inverzni redosljed

    while ProblemSize > 1                  % primjenjumo ‘‘Gentleman - Sande
        for JFirst = 1:NumOfProblems       % butterfly’’ algoritam
            J = JFirst;
            Jtwiddle = 1;
            while J < N
                W = w(Jtwiddle);
                Temp = X(J);
                X(J) = Temp + X(J + Distance);
                X(J + Distance) = (Temp - X(J + Distance))* W;
                Jtwiddle = Jtwiddle + 1;
                J = J + 2 * NumOfProblems;
            end
        end
        NumOfProblems = NumOfProblems * 2;   % sada imamo duplo manje,
        ProblemSize = ProblemSize/2;         % ali duplo veće probleme
        Distance = Distance * 2;
    end

end
```

2.3.1 Primjer iterativnog DIF FFT po bazi 2

Za $N = 2^3 = 8$ DIF FFT se sastoji od tri faze “butterfly” računanja. Sada želimo da na izlazu dobijemo podatke u prirodnom redosljedu, da bi to dobili na ulazu moramo imati podatke u bit - inverznom redosljedu. Na početku algoritma primijenit ćemo funkciju “bitrevorder” na ulazne podatke i rezultat smjestiti u X. Nakon toga izračunat ćemo težinske faktore i smjestiti ih u bit - inverzni redosljed. Zbog ispremještanosti podataka potrebna nam je dodatna varijabla “Distance” koju smo na početku algoritma postavili na jedan, da bi sa unutrašnjom “while” petljom mogli u prvoj fazi “butterfly” algoritma dobiti četiri susjedna “butterfly” para. Nakon završetka prve “butterfly” faze podatke grupišemo u dvije grupe i povećavamo varijablu “Distance” i ponovo ulazimo u vanjsku “while” petlju i dobijamo po dva “butterfly” para u dvije grupe. Isti postupak ponavljamo i u trećoj “butterfly” fazi, gdje sada imamo smo jednu grupu koja se sastoji od četiri “butterfly” para.



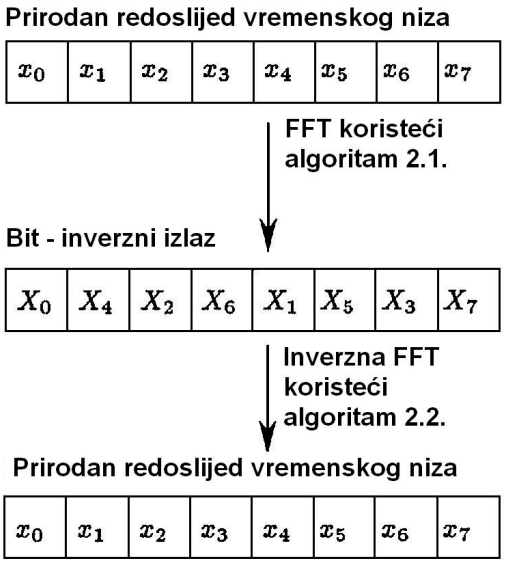
Slika 2.7. *"Butterfly" računanje za* $N = 8$



Slika 2.8. *"Butterfly" računanje za* $N = 32$

2.3.2 Ispremještani izlaz za ulaz u inverzni FFT

Kao što je to ranije pokazano DFT i inverzna diskretna Fourierova transformacija (engl. **I**nverse **D**iscrete **F**ourier **T**ransform, IDFT) imaju isto računanje osim što imamo faktor skaliranja i konjugaciju težinskih faktora W_N. Ranije spomenute algoritme možemo iskoristiti za implementaciju IDFT. Možemo kombinovati algoritam dftuciterNR (koji ćemo modifikovati tako što ćemo izbaciti dio o ispremještanju izlaznih podataka u prirodni redosljed) sa inverznim FFT koristeći inverzni iterativni FFT algoritam sa decimiranjem po učestanosti sa bit - inverznim ulazom i izlaznom sekvencom u prirodnom redosljedu. Ovaj proces je opisan za $N = 8$ na slici 2.9.



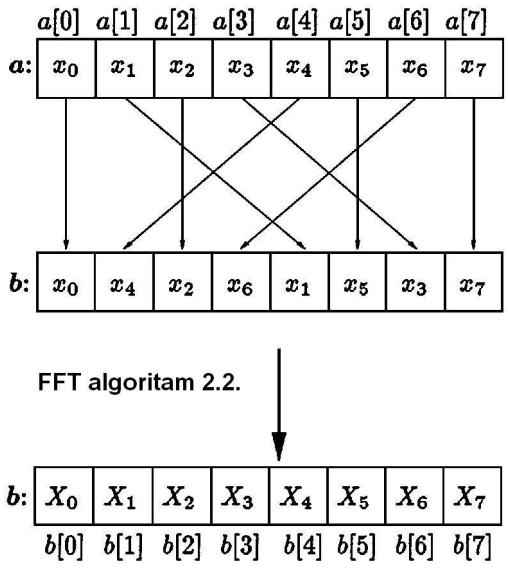
Slika 2.9. *Računanje inverzne FFT*

U slučaju da su dobijeni ulazni podaci dobiveni u bit - reverznom redosljedu možemo iskoristiti algoritam dftuciterRN u računanju FFT-a i komplementarni algoritam sa ulazom datim u prirodnom redosljedu a dobijenim izlazom u bit - reverznom redosljedu.

U svakom slučaju, sa odgovarajućom kombinacijom algoritama za računanje FFT i inverzne FFT ne moramo se brinuti o redosljedu podataka. Također, korisno je imati dvije različite implementacije.

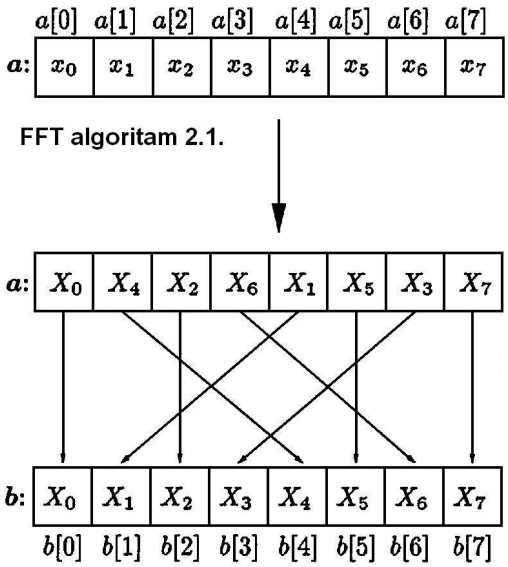
2.4 Izvođenje DIF FFT po bazi 2 sa ponavljajućim permutacijama međurezultata

U prethodnom podpoglavlju pokazano je da prvo moramo ispremiještati ulazne podatke u bit - reverzni redosljed da bi dobili izlazne podatke u prirodnom redosljedu. Ovaj proces za $N = 8$ prikazan je na slici 2.10. Kada se korak statičke permutacije ne izvodi u - mjestu, bit - reverzni ulazni podaci su dostupni u nizu b poslije izmjenjivanja redosljeda.



Slika 2.10. *Bit - inverzno premještanje ulaznih podataka prije izvođenja algoritma*

Također, možemo prvo izvršiti FFT algoritam i onda ispremiještati bit - reverzne podatke u podatke sa prirodnim redosljedom, kao što je pokazano na slici 2.11.



Slika 2.11. *Ispremještanje izlaznih podataka u bit - inverzni redosljed nakon primjene FFT algoritma*

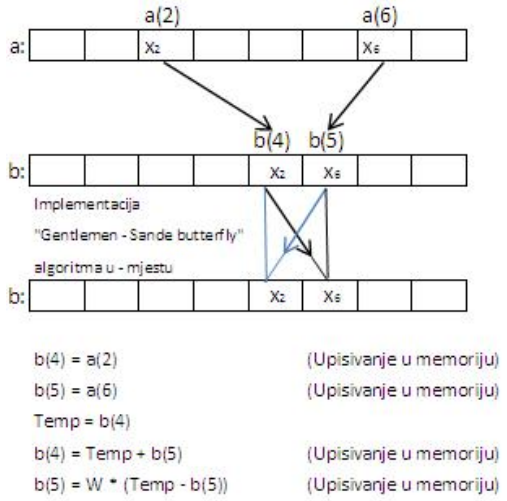
Pošto se koristi isti pristup memoriji za sve "butterflies" u svakoj fazi, ova implementacija eliminiše svu dodatnu memoriju u ispremiještanju međurezultata.

2.4.1 Kombinacija permutacija sa "butterfly" računanjem

Cijena dodatne memorije zbog odvojene bit - inverzne faze može se kompletno eliminirati ako permutaciju podataka ukombinujemo sa "butterfly" računanjem na svakom koraku.

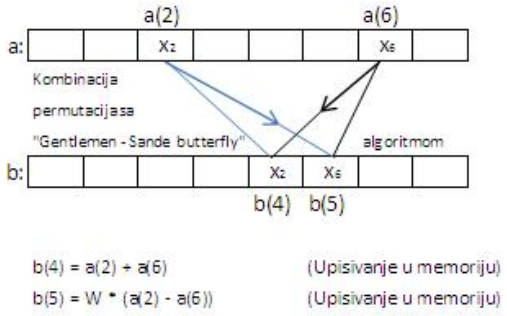
Ulazni podaci x i izlazni podaci X nalaze u prirodnom redosljedu. Potrebno je shvatiti da se svako "butterfly" računanje sastoji od jednog koraka permutacije koji je praćen sa jednim korakom računanja u - mjestu. Ovi koraci permutacije ispremiještaju ulazne podatke kao i ulaze u svaku podsekvencu.

Na slici 2.12 algoritam se izvodi logičkim načinom razmišljanja, odnosno korak permutacije se izvodi prije "butterfly" računanja. Sa slike se vidi da se memorijske lokacije $b[4]$ i $b[5]$ modifikuju dva puta.



Slika 2.12. *Prirodna implementacija algoritma*

Na slici 2.13 vidimo da se permutacija i "butterfly" računanje izvode istovremeno i rezultat se direktno upisuje u memoriju. Pošto se pristup memoriji odnosi na sva "butterflies" računanja u svakoj fazi, ovaj način implementacije eliminira sav dodatni pristup memoriji u premještanju međurezultata i ovaj način implementacije je mnogo efikasniji.



Slika 2.13. *Implementacija bez dodatnog pristupa memoriji*

Kompletan algoritam napisan u Matlab-u je dat ispod.

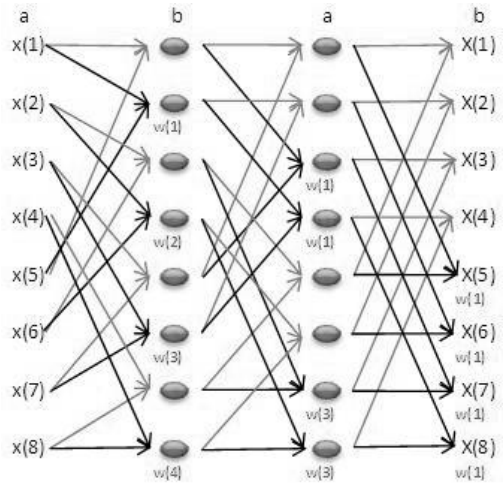
Algoritam 2.3. FFT algoritam sa decimiranjem po učestanosti po bazi 2

```
function [X] = dftuciterNN (x)
    N = length(x);
    NumOfProblems = 1;
    ProblemSize = N;
    Distance = 1;
```

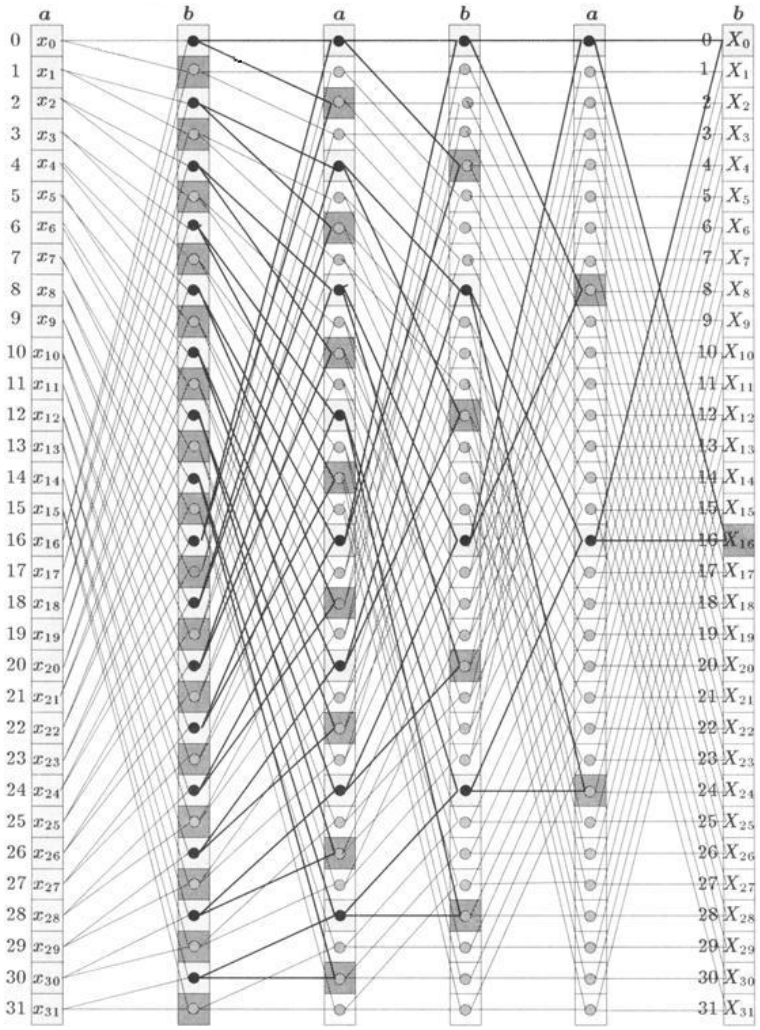
```
    NotSwitchInput = true;
    a = x;
    b = zeros(1,N);

    w = zeros(1,ProblemSize/2);            % alociramo memoriju za smještanje
    Theta = -2i*pi/ProblemSize;            % težinskih faktora u niz
    for twiddle = 1:(ProblemSize/2)
        w(twiddle) = exp(Theta*(twiddle-1));% popunjavanje niza težinskim
    end                                     % faktorima

    while ProblemSize > 1                   % podijelimo svaki problem
        if NotSwitchInput                   % niz a sadrži ulazne podatke,
             for JFirst = 1:NumOfProblems   % niz b sadrži izlazne podatke
                J = JFirst;                 % primjenjujemo "Gentleman -
                Jtwiddle = 1;               % Sande butterfly" algoritam
                K = JFirst;
                while (J < N)
                    W = w(Jtwiddle);
                    b(J) = a(K) + a(K + N/2);
                    b(J + Distance) = (a(K) - a(K+N/2)) * W;
                    Jtwiddle = Jtwiddle + NumOfProblems;
                    J = J + 2 * NumOfProblems;
                    K = K + NumOfProblems;
                end
            end
            NotSwitchInput = false;
        else                                 % niz b sadrži ulazne podatke,
            for JFirst = 1:NumOfProblems     % primjenjujemo "Gentleman -
                J = JFirst;                  % Sande butterfly" algoritam
                Jtwiddle = 1;
                K = JFirst;
                while (J < N)
                    W = w(Jtwiddle);
                    a(J) = b(K) + b(K + N/2);
                    a(J + Distance) = (b(K) - b(K + N/2)) * W;
                    Jtwiddle = Jtwiddle + NumOfProblems;
                    J = J + 2 * NumOfProblems;
                    K = K + NumOfProblems;
                end
            end
            NotSwitchInput = true;
        end
        NumOfProblems = NumOfProblems * 2;
        ProblemSize = ProblemSize/2;
        Distance = Distance * 2;
    end
    if NotSwitchInput
     X = a;
    else
        X = b;
    end
end
```

2.4.2 Primjer iterativnog DIF FFT po bazi 2

Kao i u prethodnim slučajevima, za $N = 8$ postoje tri faze "butterfly" računanja. Zbog korištenja implementacije bez dodatnog pristupa memoriji potrebno je da imamo dva niza, niz a i niz b. Iz tog razlika koristimo varijablu "NotSwitchInput" da bi zavisno da li je logička jedinica ili nula prebacivali niz a ili b u niz sa ulaznim podacima ili u niz sa izlaznim podacima. Prvo smo postavili da nam je niz a jednak ulaznim podacima, a niz b smo alocirali u memoriji. Nakon toga izračunali smo težinske faktore. Varijabla "NotSwitchInput" je postavljena na logičku jedinicu, pa iz tog razloga ulazimo u prvu "for" petlju i primjenjujemo "Gentleman - Sande butterfly" algoritam. U ovom slučaju ulazni podaci nam nisu ispremještani, pa iz tog razloga u prvoj "butterfly" fazi dobijeni parovi koji su upisani redom u niz b se sastoje od članova koji su udaljeni za četiri mjesta u nizu a. Rezultat se upisuje u niz b. Sada varijabla "NotSwitchInput" postaje logičko nula i ulazimo u drugu "for" petlje. U drugoj "butterfly" fazi "butterfly" parovi koji se upisuju u niz a se sastoje od članova koji su također udaljeni za četiri mjesta u nizu b, ali su prilikom upisa u niz a pomjereni za dva mjesta. I da bi na kraju treće "butterfly" faze dobili jednu grupu koja se sastoji od četiri "butterfly" para. Odnosno, izlazni podaci se sada nalaze u nizu b u prirodnom redoslijedu.



Slika 2.14. *"Butterfly" računanja za $N = 8$*



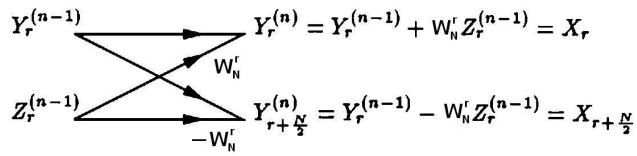
Slika 2.15. *"Butterfly" računanje za* $N = 32$

2.5 DIT FFT po bazi 2 u - mjestu za dobijene ulazne podatke u prirodnom redosljedu

U podpoglavljima 2.2, 2.4 i 2.3 date su implementacije FFT algoritama sa decimiranjem po učestanosti. Također, postoje i tri odgovarajuće varijante FFT algoritma sa decimiranjem po vremenu i oni će biti dati u ovom i u sljedeća dva podpoglavlja. Ovdje ćemo koristiti drugačiji pristup, izvest ćemo iterativne FFT algoritme sa decimiranjem po vremenu iz rekurzivne definicije.

2.5.1 Razumijevanje rekurzivnog DIT FFT algoritma i njegova implementacija u - mjestu

U poglavlju 2.1 dali smo rekurzivni DIT FFT po bazi 2. Na slici 2.16 prikazan je posljednji korak "Cooley - Tukey butterfly" algoritma u računanju transformacije (2.2).



Slika 2.16. *"Cooley - Tukey butterfly" algoritam*

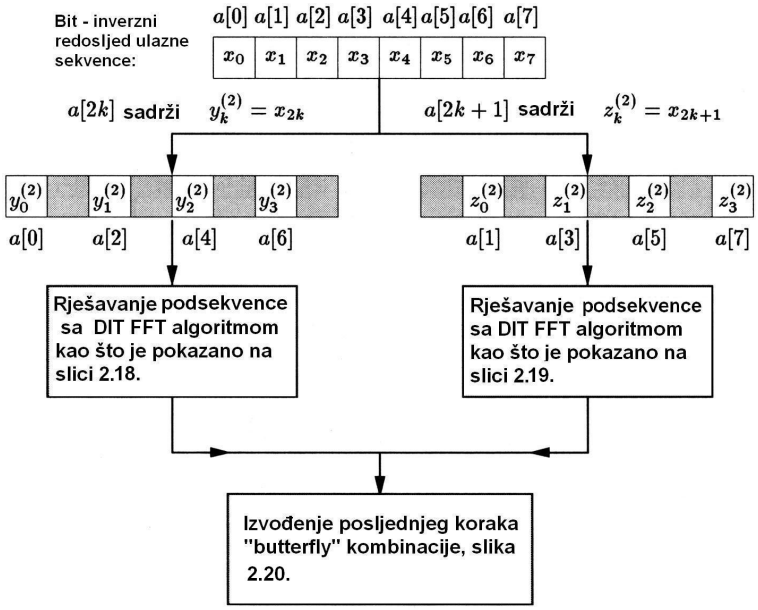
Pretpostavimo da je $N = 2^n$. $Y_r^{(n-1)}$ i $Z_r^{(n-1)}$ na slici 2.16 predstavljaju rješenje dvije podsekvence koji su definisani sa

$$Y_r^{(n-1)} = \sum_{k=0}^{\frac{N}{2}-1} x_{2k} W_N^{r(2k)} = \sum_{k=0}^{\frac{N}{2}-1} y_k^{(n-1)} W_{\frac{N}{2}}^{rk}, \quad r = 0, 1, \ldots, {}^N/_2 - 1$$

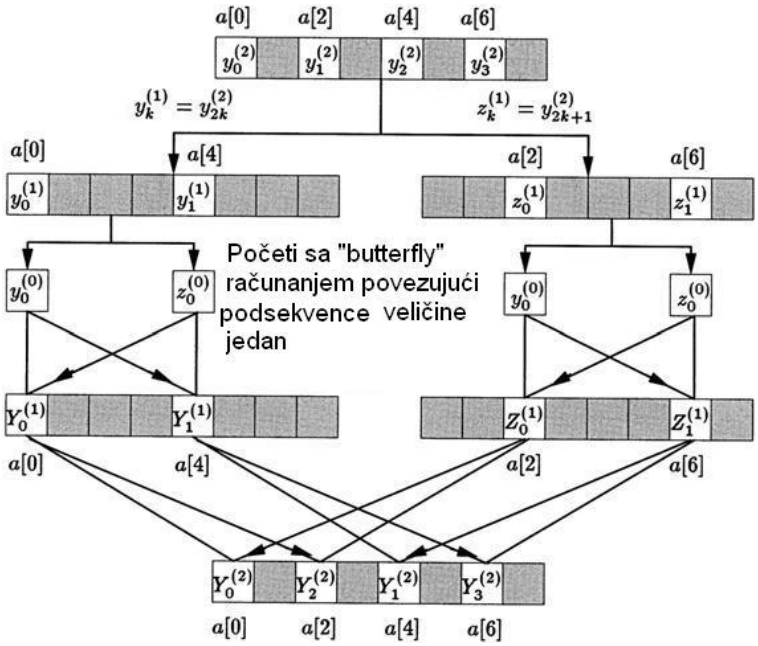
i

$$Z_r^{(n-1)} = \sum_{k=0}^{\frac{N}{2}-1} x_{2k+1} W_N^{r(2k)} = \sum_{k=0}^{\frac{N}{2}-1} z_k^{(n-1)} W_{\frac{N}{2}}^{rk}, \quad r = 0, 1, \ldots, {}^N/_2 - 1.$$

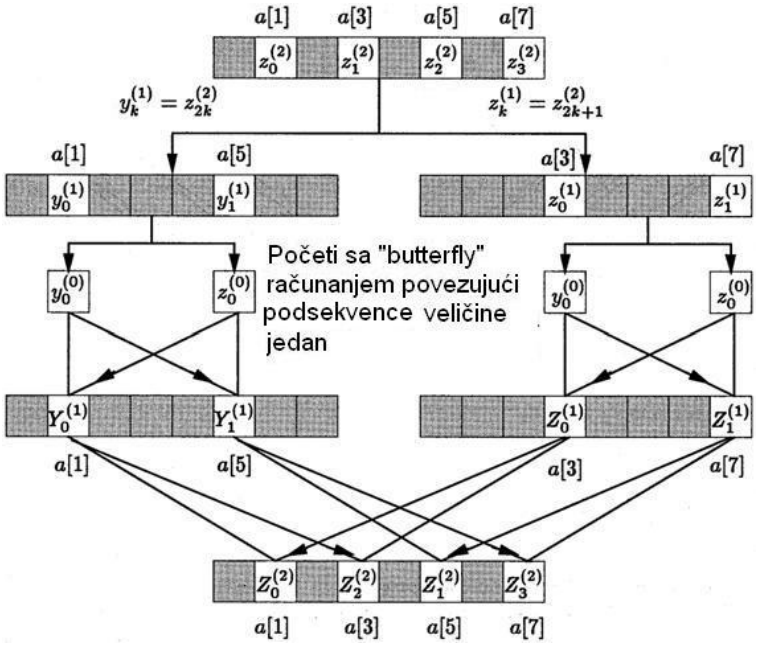
Primjećujemo da $y_k^{(n-1)} \equiv x_{2k}$ i $z_k^{(n-1)} \equiv x_{2k+1}$, za $k = 0, 1, \ldots, 2^{n-1} - 1$ određuju ulazne elemente ove dvije podsekvence dužine 2^{n-1} kao što je opisano na slici 2.17 za $N = 8$. Pošto se svaka podsekvenca rješava sa istim DIT FFT algoritmom rekurzivno, dvije podsekvence dužine $N = 2^{n-2}$ definisane su sa parnim i neparnim elementima iz $y_k^{(n-1)}$, odnosno iz $y_k^{(n-2)} \equiv y_{2k}^{(n-1)}$ i $z_k^{(n-2)} \equiv y_{2k+1}^{(n-1)}$ kao što je opisano na slici 2.18. Druge dvije podsekvence dužine $N = 2^{n-2}$ su definisana sa parnim i neparnim elementima iz $z_k^{(n-1)}$, odnosno iz $y_k^{(n-2)} \equiv z_{2k}^{(n-1)}$ i $z_k^{(n-2)} \equiv z_{2k+1}^{(n-1)}$ kao što vidimo na slici 2.19. Koraci podjele nastavljaju se sve dok dužina podsekvence ne postane jedan i rješenje svake podsekvence je jednostavno samo po sebi, $Y_0^{(0)} = y_0^{(0)}$ i $Z_0^{(0)} = z_0^{(0)}$. Iz tog razloga DIT FFT algoritam počinje sa računanjem svih kombinacija parova $y_0^{(0)}$ i $z_0^{(0)}$ da bi dobili rješenje podsekvence dužine dva i tako dalje. Za $N = 8$, tri koraka kombinacije su opisana na slikama 2.18 i 2.19. Poslije posljednjeg koraka kombinacije dobijemo DIT FFT algoritam u - mjestu sa prirodnim redosljedom ulaza i sa bit - reverznim izlazom, slika 2.20.



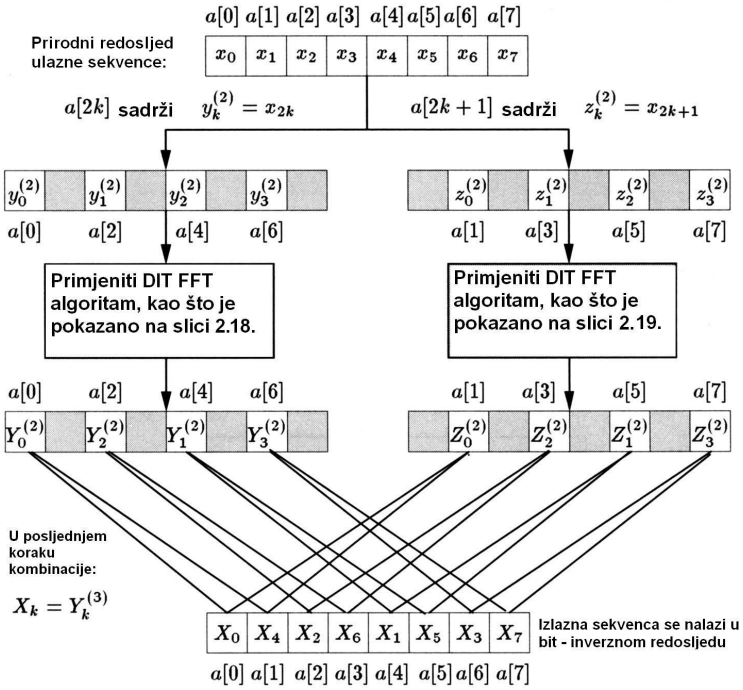
Slika 2.17. *Prvi korak podjele sekvence*



Slika 2.18. *Rješenje prve podsekvence*



Slika 2.19. *Rješenje druge podsekvence*

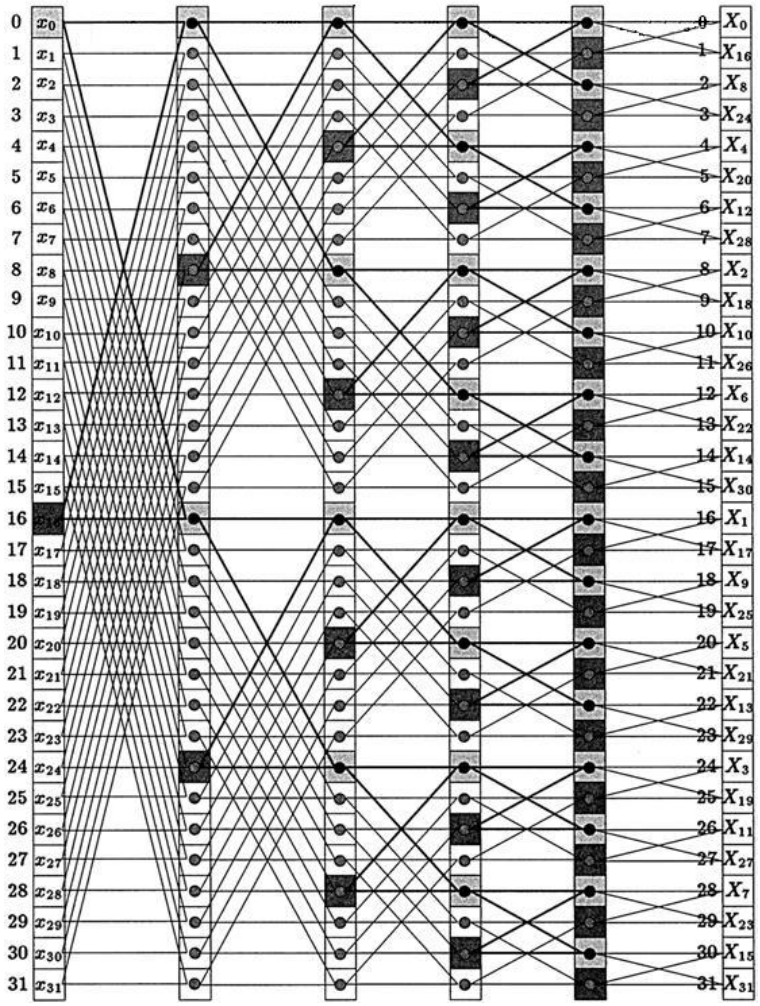


Slika 2.20. *Rješenje cijele sekvence*

Algoritam 2.4. Algoritam napisan u Matlab-u

```
function [X] = dftvriter (x)
    N = length(x);
    PairsInGroup = N/2;        % počinjemo sa N/2 "butterflies" u jednoj grupi
    NumOfGroups = 1;
    Distance = N/2;

    w = zeros(1,N/2);                     % alociramo memoriju za smještanje
    Theta = 2*pi/N;                       % težinskih faktora u niz
    for twiddle = 1:(N/2)
        w(twiddle) = exp(-1i*Theta*(twiddle-1)); % popunjavanje niza
    end                                          % težinskim faktorima

    w = bitrevorder(w);                    % težinske faktore postavimo
    while NumOfGroups < N                  % primjena "Cooley - Tukey
        for K = 0 : (NumOfGroups - 1)      % butterfly" algoritma
            JFirst = 2 * K * PairsInGroup+1;
            JLast = JFirst + PairsInGroup-1;
            Jtwiddle = K+1;
            W = w(Jtwiddle);
            for J = JFirst : JLast
                Temp = W * x(J + Distance);
                x(J + Distance) = x(J) - Temp;
                x(J) = x(J) + Temp;
            end
        end
        PairsInGroup = PairsInGroup/2;      % sada imamo duplo vise grupa,
        NumOfGroups = NumOfGroups * 2;      % ali duplo manje veličine
        Distance = Distance/2;
    end
                                            % izlazni podaci se nalaze u
                                            % bit - inverznom redosljedu
    X = bitrevorder(x);                     % podatke ispremještano da ih
                                            % dobijemo u prirodnom redosljedu
end
```

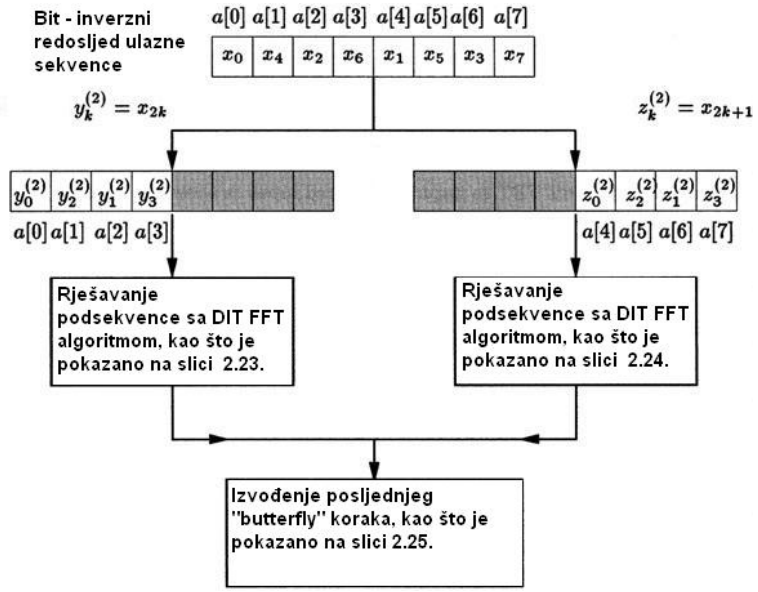
2.5.2 Primjer iterativnog DIT FFT po bazi 2



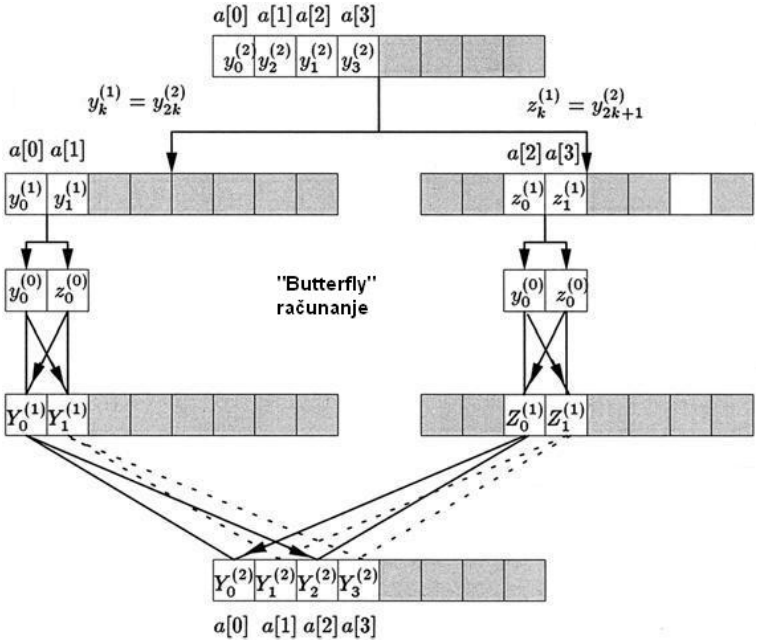
Slika 2.21. *Pet faza "butterfly" računanja*

2.6 DIT FFT po bazi 2 algoritam u - mjestu sa ulaznim podacima u bit - inverznom redosljedu

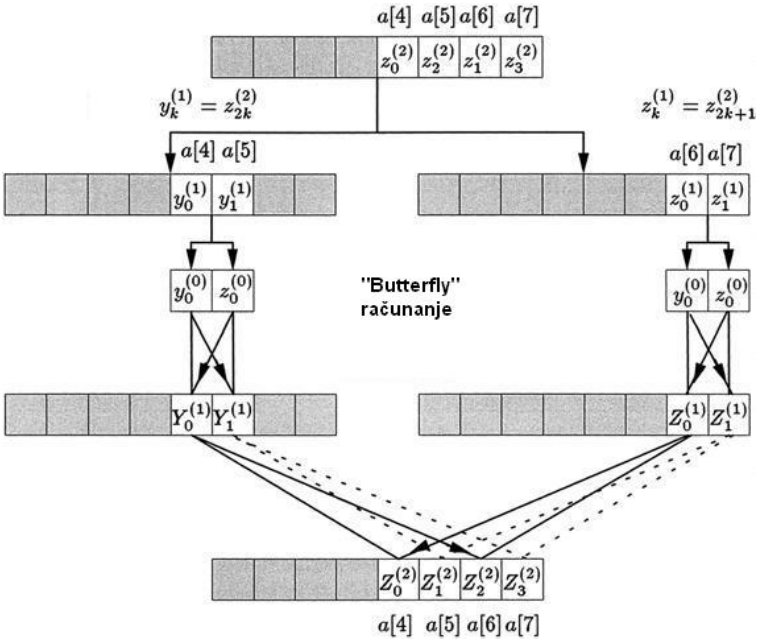
Ovo podpoglavlje je slično prethodnom podpoglavlju. Primjena DIT FFT algoritma u - mjestu na ulazne podatke smještene u bit - inverznom redosljedu za $N = 8$ opisano je na slikama 2.22 - 2.25. Prirodan redosljed izlaznih podataka je upisan preko ulaznih podataka.



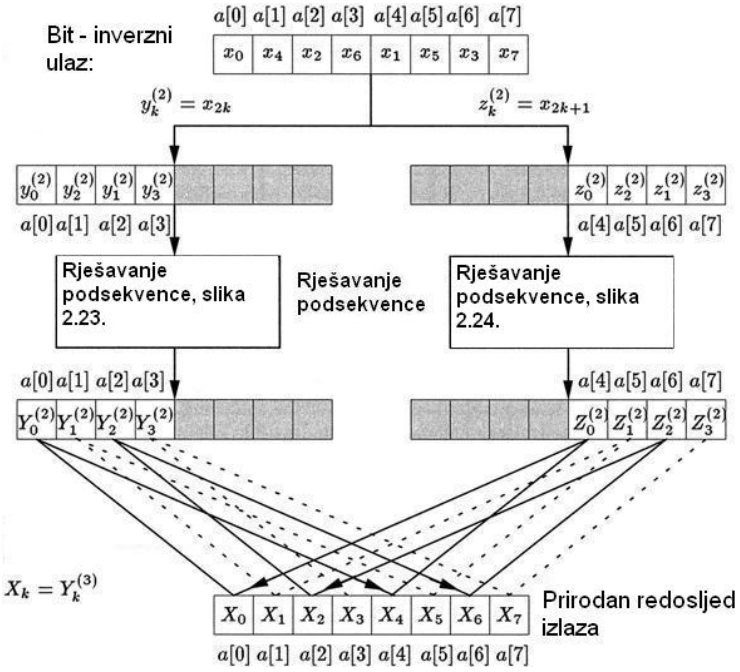
Slika 2.22. *Prvi korak podjele*



Slika 2.23. *Rješavanje prve podsekvence*



Slika 2.24. *Rješavanje druge podsekvence*

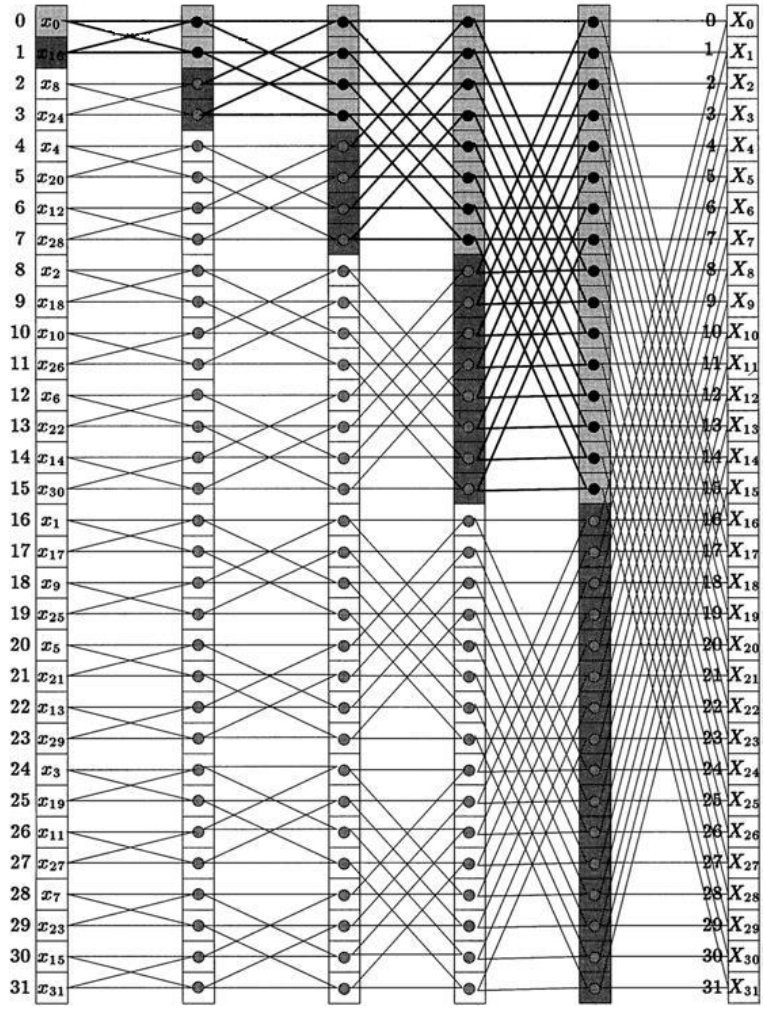


Slika 2.25. *Rješavanje cijele sekvence*

Algoritam 2.5. Algoritam napisan u Matlab-u

```
function [X] = dftvriter2 (x)
    N = length(x);
    PairsInGroup = N/2;       % počinjemo sa N/2 "butterflies" u jednoj grupi
    NumOfGroups = 1;
    Distance = 1;
    X = bitrevorder(x);                          % ulazne podatke postavimo u
                                                 % bit - inverzni redosljed
    w = zeros(1,N/2);                            % alociramo memoriju za
    Theta = 2*pi/N;                              % smještanje težinskih
    for twiddle = 1:(N/2)                        % faktora u niz
        w(twiddle) = exp(-1i*Theta*(twiddle-1)); % popunjavanje niza
    end                                          % težinskim faktorima

    while NumOfGroups < N                        % primjena "Cooley - Tukey
        GapToNextPair = 2 * NumOfGroups;         % butterfly" algoritma
        GapToLastPair = GapToNextPair * (PairsInGroup - 1);
        for K = 0 : (NumOfGroups - 1)            % modifikujemo po jednu grupu
            J = K+1;                             % prvi par
            JLast = K + GapToLastPair+1;         % posljednji par
            Jtwiddle = K * PairsInGroup+1;
            W = w(Jtwiddle);
            while J <= JLast                     % modifikujemo sve parove u
                Temp = W * X(J + Distance);      % istoj grupi
                X(J + Distance) = X(J) - Temp;
                X(J) = X(J) + Temp;
                J = J + GapToNextPair;           % nastavljamo dalje do
            end                                  % sljedećeg para u grupi
        end
        PairsInGroup = PairsInGroup/2;           % sada imamo duplo manje
        NumOfGroups = NumOfGroups * 2;           % grupa, ali duplo vece
        Distance = Distance * 2;                 % veličine
    end
end
```

2.6.1 Primjer iterativnog DIT FFT po bazi 2

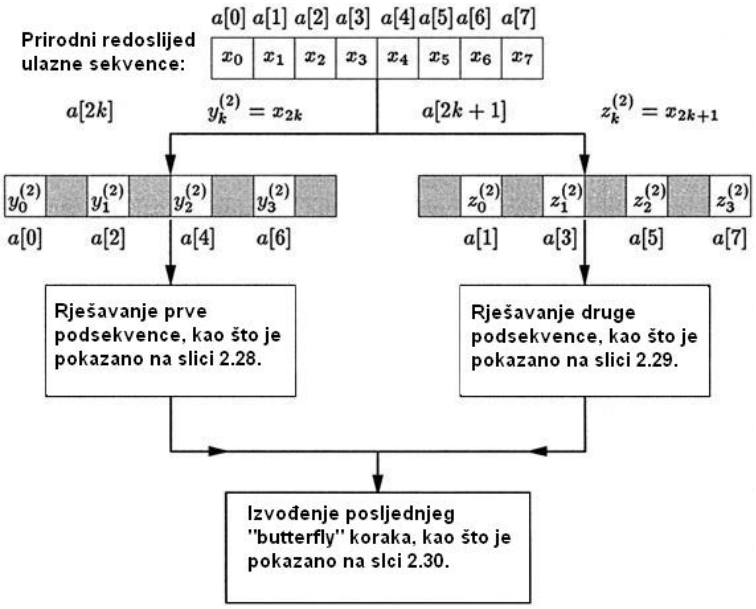


Slika 2.26. *Pet faza "butterfly" računanja*

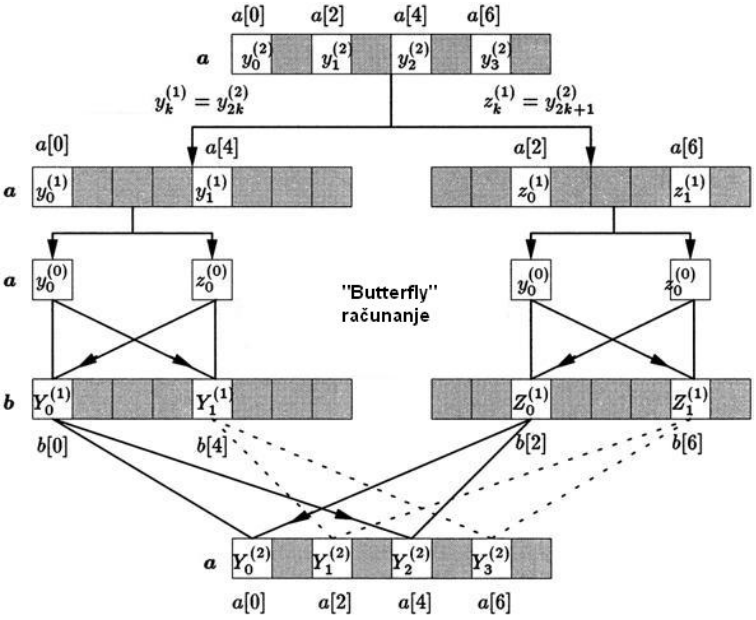
2.7 DIT FFT sa bazom 2

Ovo podpoglavlje predstavlja šestu varijantu FFT algoritma u kojem primjenjujemo DIT FFT da transformišemo prirodni redosljed ulaznog vremenskog niza u prirodni redosljed izlaznog frekventnog niza. Ovaj algoritam je sličan algoritmu koji smo predstavili u poglavlju 2.4. Ovdje će biti primijenjen već opisani princip kombinacije permutacija sa "butterfly" računanjem, koji nam dozvoljava implementaciju ponavljajućih permutacija međurezultata bez dodatnog pristupa memoriji.

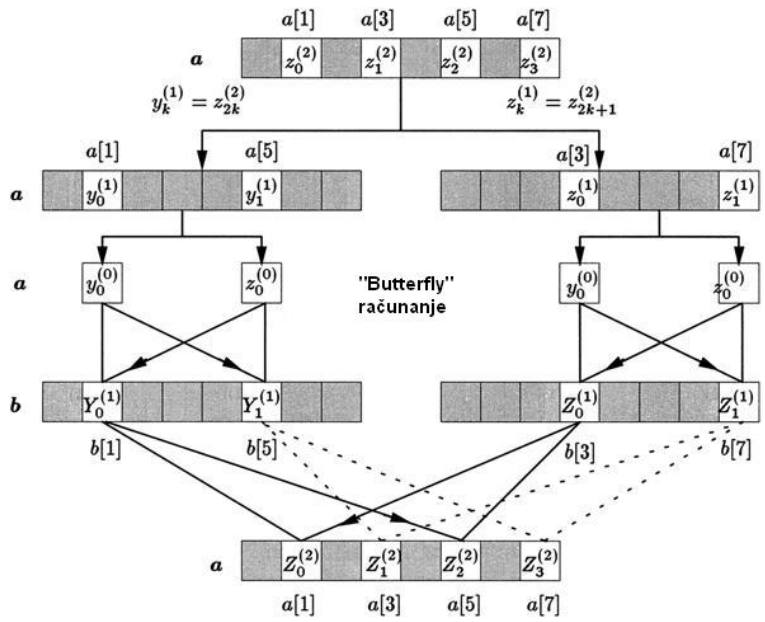
Izvođenje DIT FFT algoritma iz svoje rekurzivne formule je opisano na slikama 2.27 - 2.30 za $N = 8$.



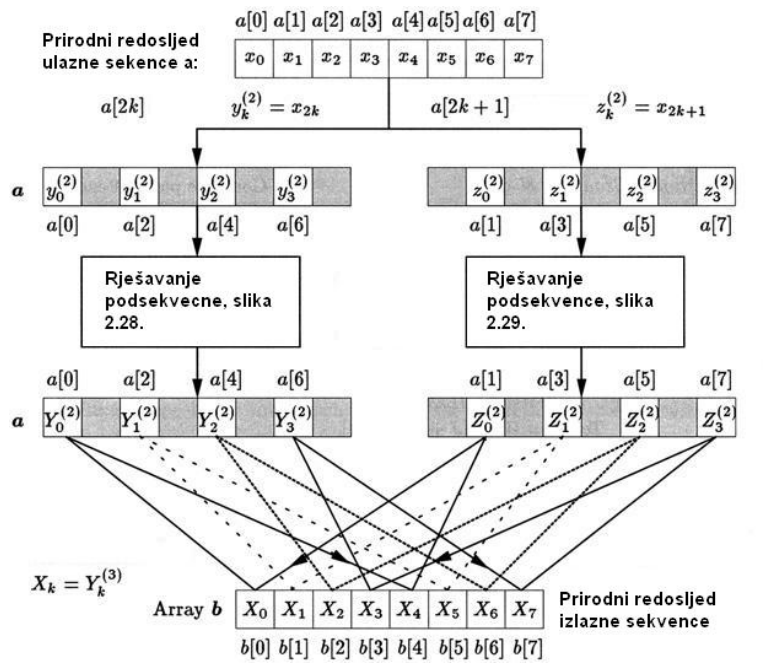
Slika 2.27. *Prvi korak podjele algoritma*



Slika 2.28. *Rješavanje prve podsekvence*



Slika 2.29. *Rješavanje druge podsekvence*



Slika 2.30. *Rješavanje cijele sekvence*

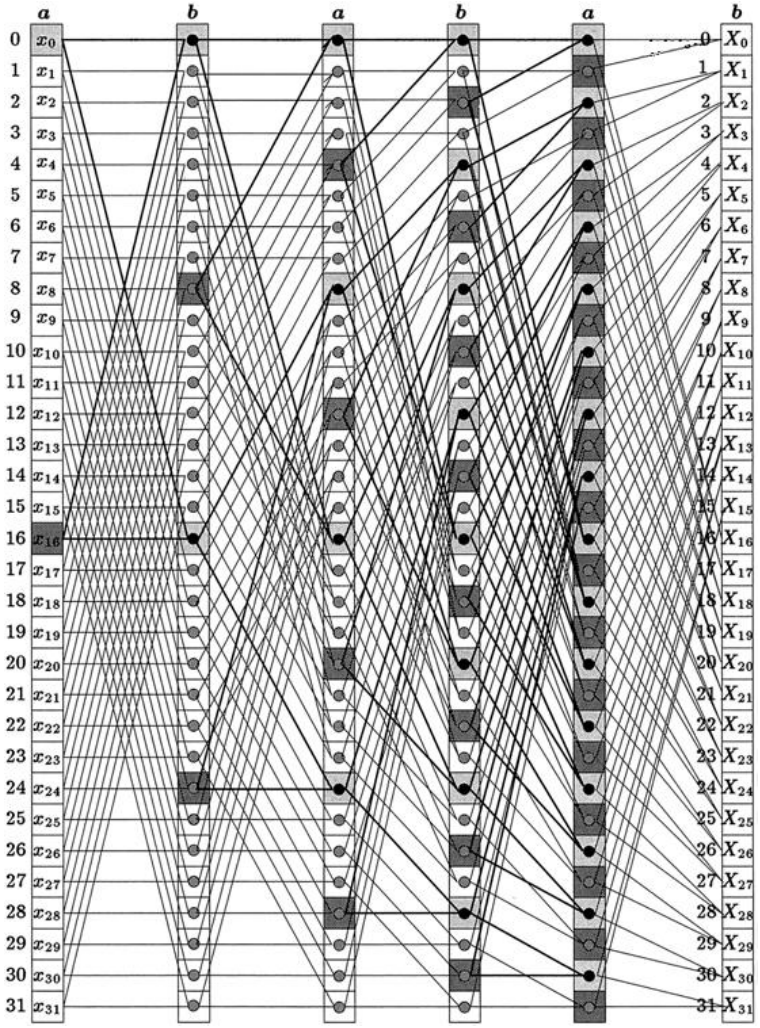
Algoritam 2.6. Algoritam napisan u Matlab-u

```
function [X] = dftvriterNN (x)
    N = length(x);
    ProblemSize = N;
    PairsInGroup = N/2;        % počinjemo sa N/2 "butterflies" u jednoj grupi
```

```
NumOfGroups = 1;
Distance = N/2;
NotSwitchInput = true;
a = x;
b= zeros(1,N);

w = zeros(1,ProblemSize/2);             %alociramo memoriju za smještanje
Theta = -2i*pi/ProblemSize;             % težinskih faktora u niz
for twiddle = 1:(ProblemSize/2)         % popunjavanje niza težinskim
    w(twiddle) = exp(Theta*(twiddle-1));% faktorima
end

while NumOfGroups < N                   % primjenjujemo "Cooley - Tukey
    if NotSwitchInput                   % butterfly" algoritam
        L = 1;                          % kombinujemo parove svake grupe
        for K = 0 : (NumOfGroups -1)    % niz a sadrži ulazne podatke, niz
            JFirst = 2 * K * PairsInGroup+1;% b sadrži izlazne podatke
            JLast = JFirst + PairsInGroup-1;
            Jtwiddle = K * PairsInGroup+1;
            W = w(Jtwiddle);
            for J = JFirst : JLast
                Temp = W * a(J + Distance);
                b(L) = a(J) + Temp;
                b(L + N/2) = a(J) - Temp;
                L = L + 1;
            end
        end
        NotSwitchInput = false;
    else
        L = 1;                           % primjenjujemo "Cooley - Tukey
        for K = 0 : (NumOfGroups - 1)    % butterfly" algoritam
            JFirst = 2 * K * PairsInGroup+1;%niz b sadrži ulazne podatke,
            JLast = JFirst + PairsInGroup-1;% niz a sadrži izlazne podatke
            Jtwiddle = K * PairsInGroup+1;
            W = w(Jtwiddle);
            for J = JFirst : JLast
                Temp = W * b(J + Distance);
                a(L) = b(J) + Temp;
                a(L + N/2) = b(J) - Temp;
                L = L + 1;
            end
        end
        NotSwitchInput = true;
    end
    PairsInGroup = PairsInGroup/2;
    NumOfGroups = NumOfGroups * 2;
    Distance = Distance / 2;
end

if NotSwitchInput
 X = a;
else
    X = b;
end
```

```
end
```

2.7.1 Primjer iterativnog DIF FFT algoritma



Slika 2.31. *Pet faza "butterfly" računanja*

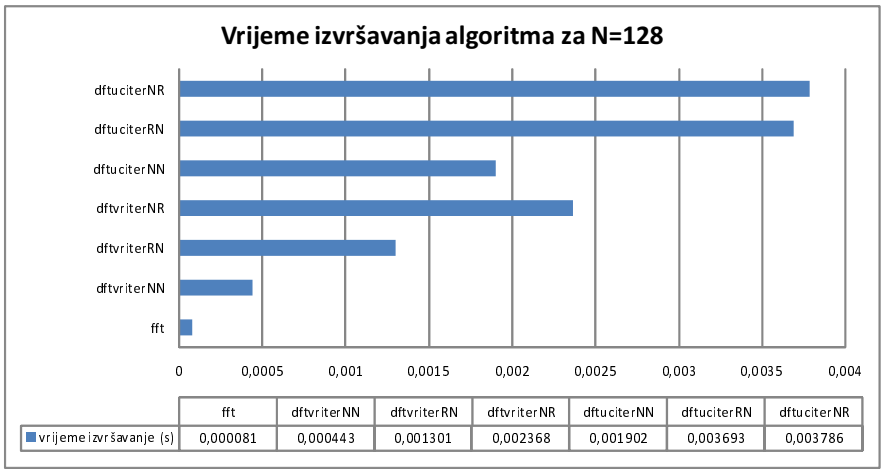
2.7.2 FFT algoritmi

Na kraju možemo zaključiti da imamo ukupno šest algoritama, od kojih smo na četiri algoritma primjenili funkciju ispremještanja podataka ili na prirodni redosljed ulazne sekvence ili na bit - inverznu izlaznu sekvencu. Preostala dva algoritma (dftuciterNN i dftvriterNN) su specifični zbog svoje implementacije, odnosno zato što ne postoji dodatni pristup memoriji.

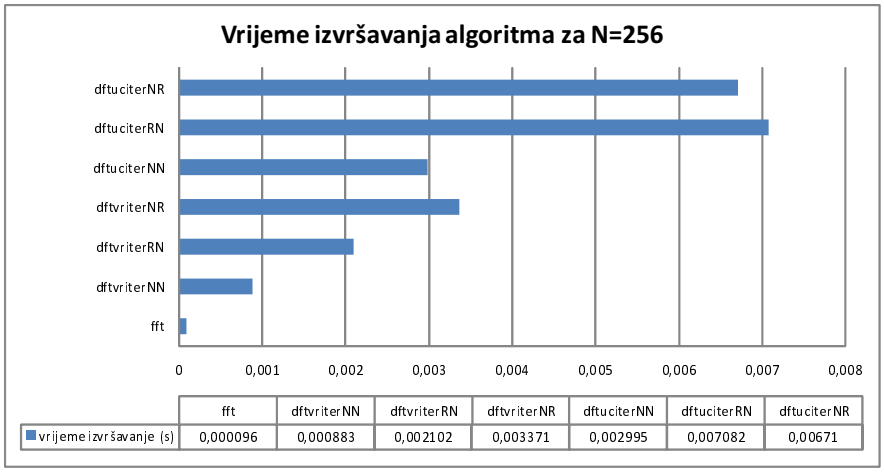
2.8 Vrijeme izvršavanja algoritama

Na slikama od 2.32 do 2.36 prikazana su vremena izvršavanja algoritama koji su dati u radu i funkcije *fft* u Matlab-u za različit broj uzoraka. Kao i što smo pretpostavljali, zbog načina na koji algoritam funkcioniše,

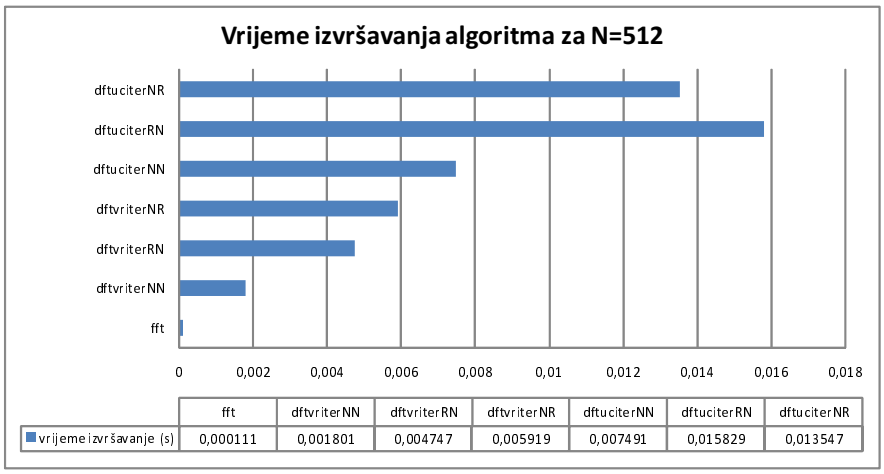
vremena izvršavanja algoritama *dftuciterNR* i *dftuciterRN* su približno ista. Zbog kombinacije permutacije podataka sa "butterfly" računanjem na svakom koraku algoritam *dftuciterNN* je skoro duplo brži od algoritama *dftuciterNR* i *dftuciterRN*. Za algoritme *dftuvriterNR* i *dftvriterRN* možemo reći da imaju skoro isto vrijeme izvršavanja za različit broj uzoraka, također im se vremena izvršavanja skoro podudaraju sa vremenom izvršavanja algoritma *dftuciterNN*. To se dešava zato što se u DIT FFT algoritmima koriste isti težinski faktori za cijelu jednu grupu. I naravno algoritam *dftvriterNN* je zbog kombinacije podataka sa "butterfly" računanjem duplo brži od algoritama *dftvriterNR* i *dftvriterRN*.



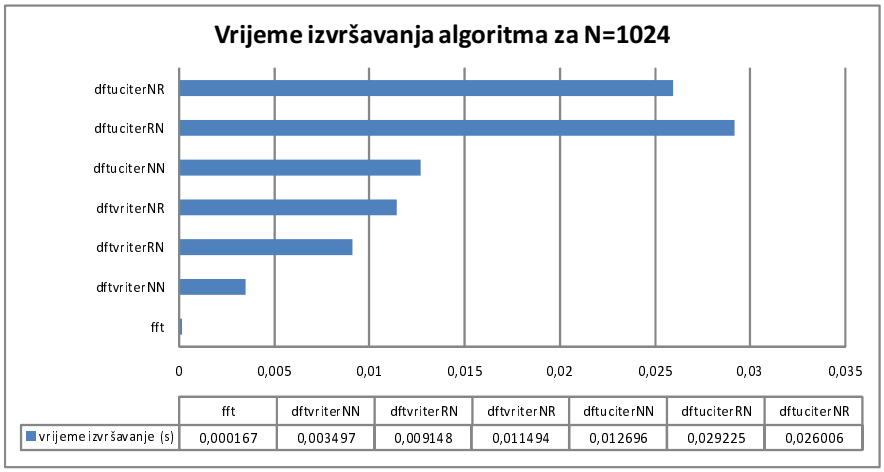
Slika 2.32. *Vrijeme izvršavanja algoritma za* $N = 128 = 2^7$



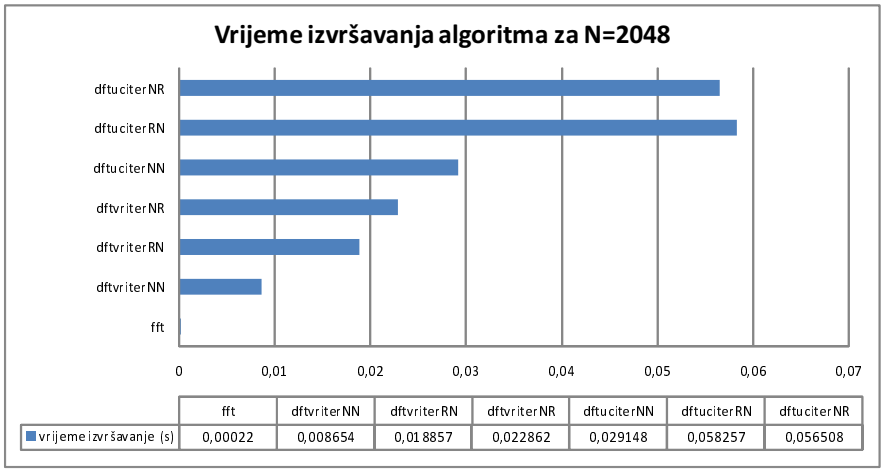
Slika 2.33. *Vrijeme izvršavanja algoritma za* $N = 256 = 2^8$



Slika 2.34. *Vrijeme izvršavanja algoritma za* $N = 512 = 2^9$

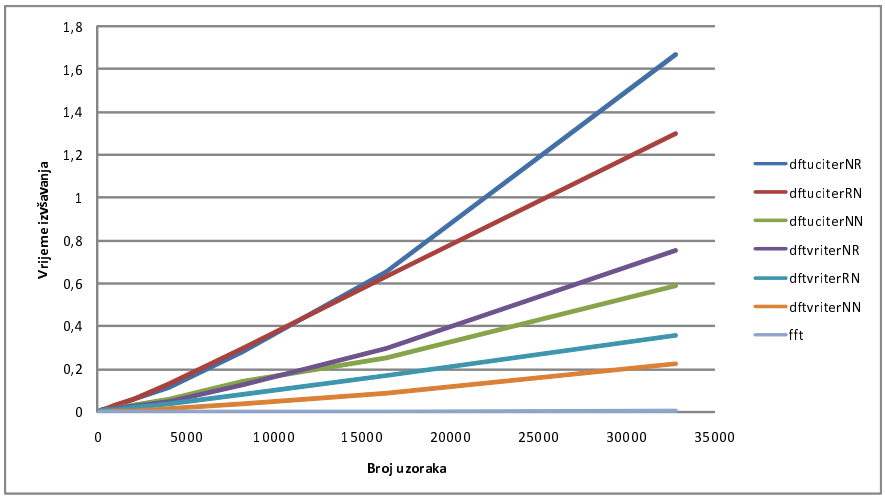


Slika 2.35. *Vrijeme izvršavanja algoritma za* $N = 1024 = 2^{10}$



Slika 2.36. *Vrijeme izvršavanja algoritma za* $N = 2048 = 2^{11}$

Na slici 2.37 prikazan je uporedni prikaz vremena izvršavanja pojedinih implementacija FFT algoritama za različit broj uzorkovanja. Kao i što smo očekivali dobili smo funkciju $N \log_2 N$.

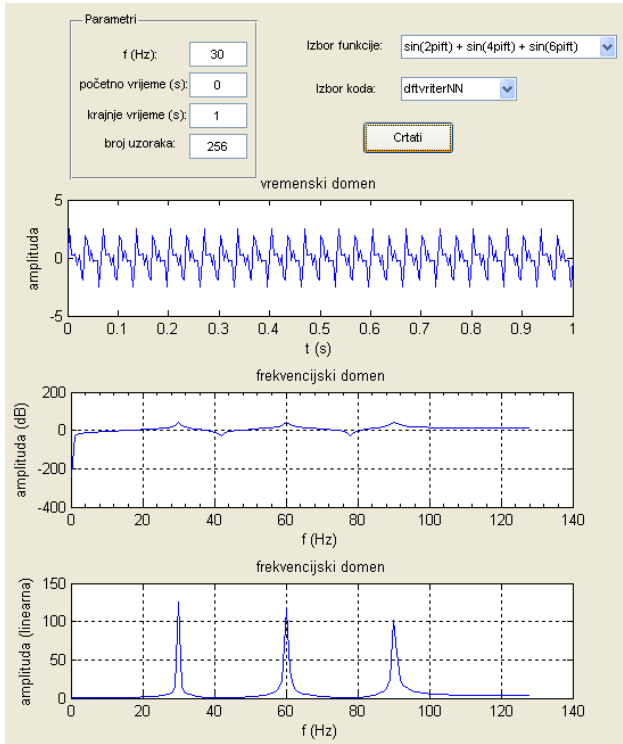


Slika 2.37. *Uporedni prikaz vremena izvršavanja pojedinih implementacija FFT algoritama za različite ulazne veličine*

2.9 Grafičko korisničko okruženje

Grafičko korisničko okruženje (engl. **G**raphical **U**ser **I**nterface, GUI) je način interakcije čovjeka s računarom kroz manipulaciju grafičkim elementima i dodacima uz pomoć tekstovnih poruka i obavijesti. Za rad u GUI - u postoje dva načina rada: preko Guide ili da programiramo direktno u Matlabu u m -file. Izabrali smo rad preko Guide. Osnovni format GUI-a prikazan je na slici 2.38. Nakon što smo izabrali koje elemente ("Push Button", "Edit Text", "Static Text", "Pop-up Menu", "Panel" i "Axes") treba da ubacimo u Guide da bi mogli upisivati parametre i mjesta gdje će se parametri i slike ispisivati, potrebno je da u m-file dodamo funkcije da bi se određena akcija u GUI-u mogla izvršiti. Prvo smo dodali funkciju koja će datoj varijabli vraćati vrijednost parametra koji smo upisali. Dodali smo funkcije *axes* i *plot*. Funkcija *Axes* omogućava da se rezultati simulacije predstave na grafiku, a funkcija *plot* crta grafike.

M - file u kojem su implementirane potrebne funkcije dat je u prilogu rada (na CD-u).



Slika 2.38. *Format GUI-a kreiran sa Matlab Guide*

2.10 FFT po bazi 4 i po bazama 2^s

Algoritam podijeli i riješi koji je predstavljen u poglavlju 2.1 nije ograničen da podijeli sekvencu u dvije podsekvence. Jednačina

$$T(N) = \begin{cases} \alpha T\left(\frac{N}{\alpha}\right) + bN, & N = \alpha^k > 1, \\ \gamma & N = 1, \end{cases}$$

predstavlja računanje kompleksnosti algoritma koji rješava originalnu sekvencu dužine N kombinujući dobijene rezultate rješavajući α sekvenci dužine N/α. U ovom poglavlju razmatrat ćemo slučajeve za $\alpha = 4$ i za $\alpha = 2^s$.

2.10.1 DIT FFT po bazi 4

U ovom dijelu ćemo razmatrati DFT vremenskog niza koji se sastoji od $N = 4^n$ diskretnih uzoraka. Pošto je $N = 4^n = 2^{2n}$ bilo koja verzija FFT-a sa bazom 2 koja je razmotrena u podpoglavljima 2.1.1 i 2.1.2 može se koristiti da bi se izračunala transformacija. Razlog zbog kojeg ćemo izvesti implementaciju po bazi 4 umjesto da jednostavno koristimo bazu 2 jeste što možemo smanjiti kompleksnost algoritma i ova prednost se koristi i kod paralelnih FFT-ova. U suštini, transformacije i po bazi 2 i po bazi 4 su specijalni slučajevi FFT-a po bazi 2^s.

DIT FFT po bazi 4 ćemo izvesti iz jednačine (2.1), koja definiše diskretnu Fourierovu transformaciju kompleksnog vremenskog niza. Koristeći da je $W_N^{\frac{N}{4}} = -j$ i $W_N^4 = W_{\frac{N}{4}}$ jednačinu (2.1) možemo razdvojiti na četiri parcijalne sume:

$$
\begin{aligned}
X_r &= \sum_{k=0}^{N-1} x_k W_N^{rk}, \quad r = 0, 1, \ldots, N-1 \qquad (2.13)\\
&= \sum_{k=0}^{\frac{N}{4}-1} x_{4k} W_N^{r(4k)} + \sum_{k=0}^{\frac{N}{4}-1} x_{4k+1} W_N^{r(4k+1)} + \sum_{k=0}^{\frac{N}{4}-1} x_{4k+2} W_N^{r(4k+2)} + \sum_{k=0}^{\frac{N}{4}-1} x_{4k+3} W_N^{r(4k+3)}\\
&= \sum_{k=0}^{\frac{N}{4}-1} x_{4k} W_N^{r(4k)} + W_N^{r} \sum_{k=0}^{\frac{N}{4}-1} x_{4k} W_N^{r(4k)} + W_N^{2r} \sum_{k=0}^{\frac{N}{4}-1} x_{4k} W_N^{r(4k)} + W_N^{3r} \sum_{k=0}^{\frac{N}{4}-1} x_{4k} W_N^{r(4k)}.
\end{aligned}
$$

Decimiranjem vremenskog niza u četiri skupa, to jest skup $\left\{y_k | y_k = x_{4k}, \quad 0 \le k \le \frac{N}{4} - 1\right\}$, skup $\left\{z_k | z_k = x_{4k+1}, \quad 0 \le k \le \frac{N}{4} - 1\right\}$, skup $\left\{g_k | g_k = x_{4k+2}, \quad 0 \le k \le \frac{N}{4} - 1\right\}$ i skup $\left\{h_k | h_k = x_{4k+3}, \quad 0 \le k \le \frac{N}{4} - 1\right\}$ dobili smo četiri podsekvence sa periodom $N/4$ koja možemo definisati nakon što uvedemo $W_N^4 = W_{\frac{N}{4}}$. Četiri podsekvence su:

$$
Y_r = \sum_{k=0}^{\frac{N}{4}-1} x_{4k} W_N^{r(4k)} = \sum_{k=0}^{\frac{N}{4}-1} x_{4k} \left(W_N^4\right)^{rk} = \sum_{k=0}^{\frac{N}{4}-1} y_k W_{\frac{N}{4}}^{rk}, \quad r = 0, 1, \ldots, N/4 - 1. \qquad (2.14)
$$

$$
Z_r = \sum_{k=0}^{\frac{N}{4}-1} x_{4k+1} W_N^{r(4k)} = \sum_{k=0}^{\frac{N}{4}-1} x_{4k+1} \left(W_N^4\right)^{rk} = \sum_{k=0}^{\frac{N}{4}-1} z_k W_{\frac{N}{4}}^{rk}, \quad r = 0, 1, \ldots, N/4 - 1. \qquad (2.15)
$$

$$
G_r = \sum_{k=0}^{\frac{N}{4}-1} x_{4k+2} W_N^{r(4k)} = \sum_{k=0}^{\frac{N}{4}-1} x_{4k+2} \left(W_N^4\right)^{rk} = \sum_{k=0}^{\frac{N}{4}-1} g_k W_{\frac{N}{4}}^{rk}, \quad r = 0, 1, \ldots, N/4 - 1. \qquad (2.16)
$$

$$
H_r = \sum_{k=0}^{\frac{N}{4}-1} x_{4k+3} W_N^{r(4k)} = \sum_{k=0}^{\frac{N}{4}-1} x_{4k+3} \left(W_N^4\right)^{rk} = \sum_{k=0}^{\frac{N}{4}-1} h_k W_{\frac{N}{4}}^{rk}, \quad r = 0, 1, \ldots, N/4 - 1. \qquad (2.17)
$$

Dužina svake podsekvence je $N/4$, što je jednako broju ulaznih podataka ili broju izračunatih izlaznih podataka jednog perioda. Nakon što riješimo pojedinačno sve četiri podsekvence, rješenje početne sekvence dužine N može se izraziti pomoću jednačine (2.13) za $r = 0, 1, \ldots, N-1$. Pošto niz Y_r ima period $N/4$, $Y_r = Y_{r+\frac{N}{4}} = Y_{r+\frac{N}{2}} = Y_{r+\frac{3N}{4}}$, i isto vrijedi za nizove Z_r, G_r i H_r izlaz X_r možemo izraziti preko Y_r, Z_r, G_r i H_r za $r = 0, 1, \ldots, \frac{N}{4} - 1$, kao što je ispod pokazano.

$$
X_r = Y_r + W_N^r Z_r + W_N^{2r} G_r + W_N^{3r} H_r. \qquad (2.18)
$$

$$
X_{r+\frac{N}{4}} = Y_r + W_N^{r+\frac{N}{4}} Z_r + W_N^{2\left(r+\frac{N}{4}\right)} G_r + W_N^{3\left(r+\frac{N}{4}\right)} H_r. \qquad (2.19)
$$

$$
X_{r+\frac{N}{2}} = Y_r + W_N^{r+\frac{N}{2}} Z_r + W_N^{2\left(r+\frac{N}{2}\right)} G_r + W_N^{3\left(r+\frac{N}{2}\right)} H_r. \qquad (2.20)
$$

$$
X_{r+\frac{3N}{4}} = Y_r + W_N^{r+\frac{3N}{4}} Z_r + W_N^{2\left(r+\frac{3N}{4}\right)} G_r + W_N^{3\left(r+\frac{3N}{4}\right)} H_r. \qquad (2.21)
$$

Uvođenjem $W_N^{\frac{N}{4}} = e^{-j\frac{2\pi}{4}} = -j$, $W_N^{\frac{N}{2}} = (-j)^2 = -1$ i $W_N^{\frac{3N}{4}} = (-j)^3 = j$ možemo jednačine (2.18), (2.21), (2.20) i (2.19) pojednostaviti i dobijamo:

$$X_r = Y_r + W_N^r Z_r + W_N^{2r} G_r + W_N^{3r} H_r, \tag{2.22}$$

$$X_{r+\frac{N}{4}} = Y_r - jW_N^r Z_r - W_N^{2r} G_r + jW_N^{3r} H_r, \tag{2.23}$$

$$X_{r+\frac{N}{2}} = Y_r - W_N^r Z_r + W_N^{2r} G_r - W_N^{3r} H_r, \tag{2.24}$$

$$X_{r+\frac{3N}{4}} = Y_r + jW_N^r Z_r - W_N^{2r} G_r - jW_N^{3r} H_r, \tag{2.25}$$

gdje je $r = 0, 1, \ldots, {}^N/_4 - 1$.

Ako je algoritam po bazi 4 implementiran na osnovu jednačina (2.22), (2.23), (2.24) i (2.25), jedan korak algoritma po bazi 4 zahtjevat će više aritmetičkih operacija od dva koraka algoritma po bazi 2, zato što se neki parcijalni rezultati računaju više puta. Ako možemo identifikovati te parcijalne rezultate i izračunati samo jednom, jedan korak algoritma po bazi 4 zahtjevat će manje aritmetičkih operacija nego dva koraka algoritma po bazi 2. Algoritam po bazi 4 možemo smanjiti da bi bio manji od algoritma po bazi 2. Četiri parcijalna rezulata su grupisana po parovima kako je pokazano u sljedećim jednačinama:

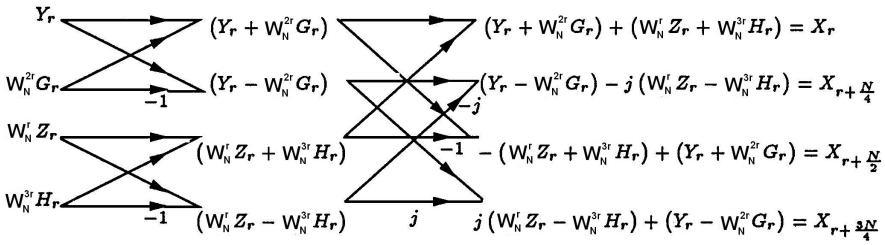
$$X_r = \left(Y_r + W_N^{2r} G_r\right) + \left(W_N^r Z_r + W_N^{3r} H_r\right), \tag{2.26}$$

$$X_{r+\frac{N}{4}} = \left(Y_r - W_N^{2r} G_r\right) - j\left(W_N^r Z_r - jW_N^{3r} H_r\right), \tag{2.27}$$

$$X_{r+\frac{N}{2}} = -\left(W_N^r Z_r + W_N^{3r} H_r\right) + \left(Y_r + W_N^{2r} G_r\right), \tag{2.28}$$

$$X_{r+\frac{3N}{4}} = j\left(W_N^r Z_r - jW_N^{3r} H_r\right) + \left(Y_r - W_N^{2r} G_r\right), \tag{2.29}$$

gdje je $r = 0, 1, \ldots, {}^N/_4 - 1$. Računanje predstavljeno jednačinama (2.26), (2.27), (2.28) i (2.29) sada se može predstaviti u dvije faze "butterfly" računanja, kao što vidimo na slici 2.39.



Slika 2.39. *Računanje FFT-a sa decimiranjem po vremenu po bazi 4*

- **Analiza kompleksnosti algoritma**

Da bi odredili kompleksnost FFT algoritma sa bazom 4, potrebno je da izračunamo $W_N^{2r} G_r$, $W_N^r Z_r$ i $W_N^{3r} H_r$ prije nego sastavimo četiri parcijalne sume. Pošto je dužina svake podsekvence ${}^N/_4$, ${}^{3N}/_4$ kompleksnih množenja i N kompleksnih sabiranja se izvode u toku prve faze "butterfly" računanja. Druga faza "butterfly" računanja izvodi se bez množenja težinskih faktora W_N, tako da je potrebno samo N kompleksnih sabiranja, odnosno, potrebno nam je ${}^{3N}/_4$ kompleksnih množenja i $2N$ kompleksnih sabiranja da bi izveli

"butterfly računanja", što je prikazano na slici 2.39. Podsjetimo se da se kompleksnost kompjuterskih algoritama mjeri brojem realnih aritmetičkih operacija. Prema tome, potrebno nam je $^{9N}/_{4}$ realnih množenja i $^{25N}/_{4}$ realnih sabiranja ili ukupno $^{17N}/_{2}$ flopova za izvođenje jednog koraka FFT DIT algoritma sa bazom 4.

Cilj nam je da smanjimo broj realnih operacija da bi minimizirali algoritam sa bazom 4. Pažljivom analizom možemo isključiti trivijalno množenje sa $W_N^0 = 1$ i sa $W_4^l = (-j)^l = \pm 1$ ili $\pm j$,također, množenje sa težinskim faktorom $W_8 = {}^{(1-j)}/\sqrt{2}$, jer je

$$W_8^{2l+1} = W_4^l \times W_8 = (-j)^l \left(\frac{1-j}{\sqrt{2}}\right) = \pm\left(\frac{1-j}{\sqrt{2}}\right) \quad ili \quad \pm\left(\frac{1+j}{\sqrt{2}}\right).$$

$0 \le r \le \frac{N}{4} - 1$	$r = 0$	$r = \left(\frac{1}{2}\right)\frac{N}{4}$	$r = \left(\frac{1}{4}\right)\frac{N}{4}$	$r = \left(\frac{3}{4}\right)\frac{N}{4}$
$W_N^{2r} G_r$	$1 \times G_r = G_r$	$W_4 G_r = -jG_r$	$\omega_8 G_r$	$\omega_8^3 G_r$
$W_N^{r} Z_r$	$1 \times Z_r = Z_r$	$W_8 Z_r$	–	–
$W_N^{3r} H_r$	$1 \times H_r = H_r$	$W_8^3 H_r$	–	–

Tablica 11. *Specijalni slučajevi množenja težinskim faktorima W_N u algoritmu sa bazom* 4

Postoji osam specijalnih slučajeva: četiri slučaja koja uključuju množenje sa 1 ili sa $-j$ su trivijalna, ostala četiri slučaja uključuju množenje sa W_8 sa neparnim eksponentom koji se tretiraju specijalno. Ukupan broj netrivijalnih kompleksnih množenja se smanjuje na $^{3}/_{4}N - 8$. Pošto nam je potrebno samo 4 flopa da bi izračunali svaki od $W_8 G_r$, $W_8^3 G_r$, $W_8 Z_r$ i $W_8^3 H_r$, ukupan broj flopova iznosi

$$(3+3) \times \left(\frac{3}{4}N - 8\right) + 4 \times 4 + 2 \times (2N) = \frac{17}{2}N - 32.$$

Ove specijalne faktore također imamo i u svakom kasnijem koraku, spašavanja istih možemo uključiti u rekurzivnu jednačinu. Zbog kompletnosti, pretpostavimo da je dužina sekvence $N = 4^n$ i ovi specijalni faktori se definišu za $r = 0, 1, \ldots, {}^{N}/(4^{i+1})$ u svakom i-tom koraku za $i = 0, 1, \ldots, n-2$.

$0 \le r \le \frac{N}{4^{i+1}} - 1$	$r = 0$	$r = \left(\frac{1}{2}\right)\frac{N}{4^{i+1}}$	$r = \left(\frac{1}{4}\right)\frac{N}{4^{i+1}}$	$r = \left(\frac{3}{4}\right)\frac{N}{4^{i+1}}$
$W^{2r}_{\frac{N}{4^{i+1}}} G_r$	$1 \times G_r = G_r$	$W_4 G_r = -jG_r$	$W_8 G_r$	$W_8^3 G_r$
$W^{r}_{\frac{N}{4^{i+1}}} Z_r$	$1 \times Z_r = Z_r$	$W_8 Z_r$	–	–
$W^{3r}_{\frac{N}{4^{i+1}}} H_r$	$1 \times H_r = H_r$	$W_8^3 H_r$	–	–

Tablica 12. *Periodični specijalni slučajevi množenja težinskim faktorima W_N u algoritmu sa bazom* 4

Da bi postavili rekurzivnu jednačinu potreban nam je granični uslov, $N = 4$. Podsjetimo se da kada je $N = 4$, težinski faktori su: 1,-1,j i $-j$, tako da prva "butterfly" faza računanja uključuje ne trivijalno kompleksno množenje. Zato, kada je $N = 4$, potrebno je $2 \times N = 8$ kompleksnih sabiranja ili $4 \times N = 16$ realnih sabiranja.

Sada možemo napisati jednačinu za kompleksnost FFT algoritma po bazi 4:

$$T(N) = \begin{cases} 4T\left(\frac{N}{4}\right) + \frac{17}{2}N - 32, & N = 4^n > 4, \\ 16, & N = 4, \end{cases}$$

ili

$$T(N) = N \log_2 N - \frac{43}{6}N + \frac{32}{3}.$$

Ako ovu kompleksnost uporedimo sa kompleksnosti algoritma po bazi 2, koja iznosi $T(N) = 5N \log_2 N$, zaključujemo da smanjenje kompleksnosti algoritma po bazi 4 iznosi 15%.

2.10.2 DIF FFT po bazi 4

Algoritam FFT sa decimiranjem po učestanosti po bazi 4 izvodimo rekurzivno decimiranjem frekventne sekvence u četiri podskupa. Skupovi su označeni kao $Y_k = X_{4k}$ za $0 \leq k \leq \frac{N}{4} - 1$, $Z_k = X_{4k+1}$ za $0 \leq k \leq \frac{N}{4} - 1$, $G_k = X_{4k+2}$ za $0 \leq k \leq \frac{N}{4} - 1$ i $H_k = X_{4k+3}$ za $0 \leq k \leq \frac{N}{4} - 1$ kao što je ispod pokazano. Izvođenje počinjemo sa definicijom DFT-a, odnosno jednačinom (2.1).

$$
\begin{aligned}
X_r &= \sum_{l=0}^{N-1} x_l W_N^{rl}, \quad r = 0, 1, \ldots, N-1, \qquad (2.30)\\
&= \sum_{l=0}^{\frac{N}{4}-1} x_l W_N^{rl} + \sum_{l=\frac{N}{4}}^{\frac{N}{2}-1} x_l W_N^{rl} + \sum_{l=\frac{N}{2}}^{\frac{3N}{4}-1} x_l W_N^{rl} + \sum_{l=\frac{3N}{4}}^{N-1} x_l W_N^{rl}\\
&= \sum_{l=0}^{\frac{N}{4}-1} x_l W_N^{rl} + \sum_{l=0}^{\frac{N}{4}-1} x_{l+\frac{N}{4}} W_N^{r\left(l+\frac{N}{4}\right)} + \sum_{l=0}^{\frac{N}{4}-1} x_{l+\frac{N}{2}} W_N^{r\left(l+\frac{N}{2}\right)} + \sum_{l=0}^{\frac{N}{4}-1} x_{l+\frac{3N}{4}} W_N^{r\left(l+\frac{3N}{4}\right)}\\
&= \sum_{l=0}^{\frac{N}{4}-1} x_l W_N^{rl} + W_4^r \sum_{l=0}^{\frac{N}{4}-1} x_{l+\frac{N}{4}} W_N^{rl} + W_4^{2r} \sum_{l=0}^{\frac{N}{4}-1} x_{l+\frac{N}{2}} W_N^{rl} + W_4^{3r} \sum_{l=0}^{\frac{N}{4}-1} x_{l+\frac{3N}{4}} W_N^{rl}\\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(x_l + x_{l+\frac{N}{4}} W_4^r + x_{l+\frac{N}{2}} W_4^{2r} + x_{l+\frac{3N}{4}} W_4^{3r}\right) W_N^{rl}.
\end{aligned}
$$

Četiri podsekvence ćemo jednostavno izvesti uvrštavajući odgovarajuće smjene, $r = 4k$, $r = 4k+1$, $r = 4k+2$ i $r = 4k+3$ u jednačinu (2.30).

$$
\begin{aligned}
Y_k &= X_{4k} = \sum_{l=0}^{\frac{N}{4}-1} \left(x_l + x_{l+\frac{N}{4}} W_4^{4k} + x_{l+\frac{N}{2}} W_4^{2\times 4k} + x_{l+\frac{3N}{4}} W_4^{3\times 4k}\right) W_N^{4kl} \qquad (2.31)\\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(x_l + x_{l+\frac{N}{4}} + x_{l+\frac{N}{2}} + x_{l+\frac{3N}{4}}\right) W_{\frac{N}{4}}^{kl}\\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(\left(x_l + x_{l+\frac{N}{2}}\right) + \left(x_{l+\frac{N}{4}} + x_{l+\frac{3N}{4}}\right)\right) W_{\frac{N}{4}}^{kl}\\
&= \sum_{l=0}^{\frac{N}{4}-1} y_l W_{\frac{N}{4}}^{kl}, \quad r = 0, 1, \ldots, {}^{N}/_{4} - 1.
\end{aligned}
$$

$$
\begin{aligned}
Z_k &= X_{4k+1} = \sum_{l=0}^{\frac{N}{4}-1} \left(x_l + x_{l+\frac{N}{4}} W_4^{4k+1} + x_{l+\frac{N}{2}} W_4^{2\times(4k+1)} + x_{l+\frac{3N}{4}} W_4^{3\times(4k+1)}\right) W_N^{(4k+1)l} \qquad (2.32)\\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(x_l - j x_{l+\frac{N}{4}} - x_{l+\frac{N}{2}} + j x_{l+\frac{3N}{4}}\right) W_N^{l} W_{\frac{N}{4}}^{kl}\\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(\left(x_l - x_{l+\frac{N}{2}}\right) - j\left(x_{l+\frac{N}{4}} - x_{l+\frac{3N}{4}}\right)\right) W_N^{l} W_{\frac{N}{4}}^{kl}\\
&= \sum_{l=0}^{\frac{N}{4}-1} z_l W_{\frac{N}{4}}^{kl}, \quad r = 0, 1, \ldots, {}^{N}/_{4} - 1.
\end{aligned}
$$

$$
\begin{aligned}
G_k &= X_{4k+2} = \sum_{l=0}^{\frac{N}{4}-1} \left(x_l + x_{l+\frac{N}{4}} W_4^{4k+2} + x_{l+\frac{N}{2}} W_4^{2\times(4k+2)} + x_{l+\frac{3N}{4}} W_4^{3\times(4k+2)} \right) W_N^{(4k+2)l} \qquad (2.33)\\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(x_l - x_{l+\frac{N}{4}} + x_{l+\frac{N}{2}} - x_{l+\frac{3N}{4}} \right) W_N^{2l} W_{\frac{N}{4}}^{kl}\\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(\left(x_l + x_{l+\frac{N}{2}} \right) - \left(x_{l+\frac{N}{4}} + x_{l+\frac{3N}{4}} \right) \right) W_N^{2l} W_{\frac{N}{4}}^{kl}\\
&= \sum_{l=0}^{\frac{N}{4}-1} g_l W_{\frac{N}{4}}^{kl}, \quad r = 0, 1, \ldots, {}^{N}/_{4} - 1.
\end{aligned}
$$

$$
\begin{aligned}
H_k &= X_{4k+3} = \sum_{l=0}^{\frac{N}{4}-1} \left(x_l + x_{l+\frac{N}{4}} W_4^{4k+3} + x_{l+\frac{N}{2}} W_4^{2\times(4k+3)} + x_{l+\frac{3N}{4}} W_4^{3\times(4k+3)} \right) W_N^{(4k+3)l} \qquad (2.34)\\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(x_l + j x_{l+\frac{N}{4}} - x_{l+\frac{N}{2}} - j x_{l+\frac{3N}{4}} \right) W_N^{3l} W_{\frac{N}{4}}^{kl}\\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(\left(x_l - x_{l+\frac{N}{2}} \right) + j \left(x_{l+\frac{N}{4}} - jx \right) \right) W_N^{3l} W_{\frac{N}{4}}^{kl}\\
&= \sum_{l=0}^{\frac{N}{4}-1} h_l W_{\frac{N}{4}}^{kl}, \quad r = 0, 1, \ldots, {}^{N}/_{4} - 1.
\end{aligned}
$$

Da bi formirali ove podsekvence koristeći dvije faze "butterfly" računanja, potrebno je prvo pregrupisati parcijalne sume.

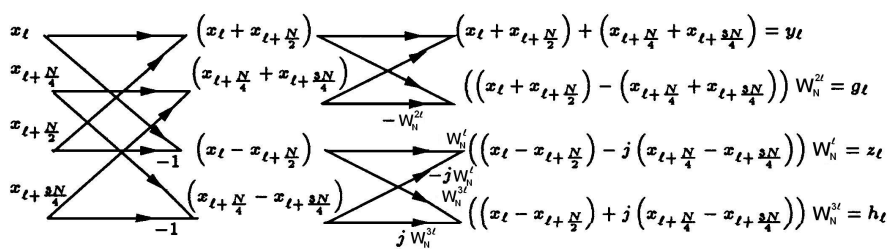
$$
y_l = \left(x_l + x_{l+\frac{N}{2}} \right) + \left(x_{l+\frac{N}{4}} + x_{l+\frac{3N}{4}} \right), \quad 0 \le l \le \frac{N}{4} - 1. \qquad (2.35)
$$

$$
z_l = \left(\left(x_l - x_{l+\frac{N}{2}} \right) - j \left(x_{l+\frac{N}{4}} - x_{l+\frac{3N}{4}} \right) \right) W_N^{l}, \quad 0 \le l \le \frac{N}{4} - 1. \qquad (2.36)
$$

$$
g_l = \left(- \left(x_{l+\frac{N}{4}} + x_{l+\frac{3N}{4}} \right) + \left(x_l + x_{l+\frac{N}{2}} \right) \right) W_N^{2l}, \quad 0 \le l \le \frac{N}{4} - 1. \qquad (2.37)
$$

$$
h_l = \left(j \left(x_{l+\frac{N}{4}} - jx \right) + \left(x_l - x_{l+\frac{N}{2}} \right) \right) W_N^{3l}, \quad 0 \le l \le \frac{N}{4} - 1. \qquad (2.38)
$$

Računanje predstavljeno jednačinama (2.35), (2.36), (2.37) i (2.38) može se predstaviti sa dvije faze "butterfly" računanja, kao što je prikazano na slici 2.40.



Slika 2.40. *Računanje FFT-a sa decimiranjem po učestanosti po bazi 4*

2.10.3 DIT I DIF FFT po bazi 2^s

Tehnike koje smo koristili za algoritme po bazi 2 i po bazi 4 mogu se generalizovati za algoritam po bazi 2^s. Stavljajući da je $q = 2^s$ FFT algoritam sa decimiranjem po vremenu po bazi q možemo dobiti rastavljanjem jednačine (2.1) u q parcijalnih suma:

$$\begin{aligned} X_r &= \sum_{k=0}^{N-1} x_k W_N^{rk}, \quad r = 0, 1, \ldots, N-1 \\ &= \sum_{u=0}^{q-1} \sum_{k=0}^{\frac{N}{q}-1} \left(x_{qk} + u W_N^{r(qk+u)} \right) \\ &= \sum_{u=0}^{q-1} W_N^{ur} \sum_{k=0}^{\frac{N}{q}-1} \left(x_{qk} + u W_N^{rqk} \right) \\ &= \sum_{u=0}^{q-1} W_N^{ur} \sum_{k=0}^{\frac{N}{q}-1} \left(x_{qk} + u \left(W_N^{q} \right)^{rk} \right) \\ &= \sum_{u=0}^{q-1} W_N^{ur} \sum_{k=0}^{\frac{N}{q}-1} \left(x_{qk} + u W_{\frac{N}{q}}^{rk} \right). \end{aligned} \tag{2.39}$$

Primjećujemo da su jednačine (2.2) i (2.13) specijalni slučajevi jednačine (2.39) kada je $q = 2$ i $q = 4$ respektivno. Prema jednačini (2.39), vremenski niz možemo decimirati u $q = 2^s$ skupova tako da se svaka od q parcijalnih suma predstavljena kao $\sum_{k=0}^{\frac{N}{q}-1} \left(x_{qk} + u W_N^{r(qk+u)} \right)$ za $u = 0, 1, \ldots, q-1$ može rekurzivno izračunati neovisno jedna od druge. Svaka parcijalna suma predstavlja DFT podsekvence dužine N/q.

Izlazne frekvencije se računaju kao q odvojeni segmenti i svaki segmenat je opisan sa $X_{l+\lambda\frac{N}{q}}$ koji ima N/q rastućih elemenata indeksiranih sa l, gdje je $0 \leq l \leq N/q - 1$ i $0 \leq \lambda \leq q-1$. Uvrštavajući zamjenu $r = l + \lambda\frac{N}{q}$ u jednačini (2.39) dobijemo jednačinu za računanje izlazne frekvencije za svaki q-ti segment. Jednačina za λ-ti segment je pokazana ispod, gdje je $0 \leq \lambda \leq q-1$.

$$\begin{aligned} X_{l+\lambda\frac{N}{q}} &= \sum_{u=0}^{q-1} W_N^{ul} \left(\omega_N^{\frac{N}{q}} \right)^{u\lambda} \left(\sum_{k=0}^{\frac{N}{q}-1} x_{qk} + u W_{\frac{N}{q}}^{k\left(l+\lambda\frac{N}{q}\right)} \right) \\ &= \sum_{u=0}^{q-1} W_N^{ul} W_q^{u\lambda} \left(\sum_{k=0}^{\frac{N}{q}-1} x_{qk} + u W_{\frac{N}{q}}^{kl} \right), \quad l = 0, 1, \ldots, N/q - 1. \end{aligned} \tag{2.40}$$

Naravno, da bi minimizirali kompleksnost algoritma računanje q frekvencije segmenata trebali bi pregrupisati da izbjegnemo suvišno računanje kao što je pokazano ranije u izvođenju FFT algoritma sa decimiranjem po vremenu po bazi 4. Ako stavimo da je $q = 2$ i $\lambda = 0, 1$ u jednačinu (2.40) dobit ćemo jednačine (2.6) i (2.7) za računanje dva frekventna segmenata u FFT algortimu sa decimiranju po vremenu po bazi 2. Ako stavimo da je $q = 4$ i $\lambda = 0, 1, 2, 3$ u jednačinu (2.40) dobit ćemo četiri jednačine (2.18), (2.19), (2.20) i (2.21) za računanje četiri frekventna segmenta u FFT algoritmu sa decimiranjem po vremenu po bazi 4.

Da bi razvili DIF FFT algoritam po bazi q potrebno je jednostavno decimirati frekventni skup X_r u q skupova gdje će svaki skup sadržavati $\left\{ Y_k^{(u)} | Y_k^{(u)} = X_{qk+u},\ 0 \leq k \leq N/q - 1 \right\}$ za $u = 0, 1, \ldots q-1$. Svaki od q podsekvence dužine N/q i definisan je zamjenom $r = qk + u$ u jednačinu (2.41) kao što je pokazano za $q = 2$ i $q = 4$ u izvođenjima FFT algoritma sa decimiranjem po frekvenciji po bazi 2 i po bazi 4 u podpoglavljima

2.1.2 i 2.10.2. Za potpunost ovog podpoglavlja dato je kratko izvođenje koje će generalizovati algoritam po bazi 2 i po bazi 4:

$$\begin{aligned} X_r &= \sum_{l=0}^{N-1} x_l W_N^{rl}, \quad r = 0, 1, \ldots, N-1 \\ &= \sum_{\lambda=0}^{q-1} \sum_{l=0}^{\frac{N}{q}-1} x_{l+\lambda\frac{N}{q}} W_N^{r\left(l+\lambda\frac{N}{q}\right)} \\ &= \sum_{l=0}^{\frac{N}{q}-1} \sum_{\lambda=0}^{q-1} x_{l+\lambda\frac{N}{q}} W_N^{r\left(l+\lambda\frac{N}{q}\right)} \\ &= \sum_{l=0}^{\frac{N}{q}-1} \sum_{\lambda=0}^{q-1} x_{l+\lambda\frac{N}{q}} W_N^{r\lambda\frac{N}{q}} W_N^{rl} \\ &= \sum_{l=0}^{\frac{N}{q}-1} \left(\sum_{\lambda=0}^{q-1} x_{l+\lambda\frac{N}{q}} W_N^{r\lambda\frac{N}{q}} \right) W_N^{rl}. \end{aligned} \tag{2.41}$$

q podsekvenci bit će izvedeni zamjenom $r = qk + u$ u jednačini (2.41) za $u = 0, 1, \ldots q-1$.

$$\begin{aligned} Y_k^{(u)} &= X_{qk+u} = \sum_{l=0}^{\frac{N}{q}-1} \left(\sum_{\lambda=0}^{q-1} x_{l+\lambda\frac{N}{q}} W_N^{r(qk+u)} \right) W_N^{(qk+u)l} \\ &= \sum_{l=0}^{\frac{N}{q}-1} \left\{ \left(\sum_{\lambda=0}^{q-1} x_{l+\lambda\frac{N}{q}} W_N^{r(qk+u)} \right) W_N^{ul} \right\} W_{\frac{N}{q}}^{kl} \\ &= \sum_{l=0}^{\frac{N}{q}-1} y_l^{(u)} W_{\frac{N}{q}}^{kl}, \quad k = 0, 1, \ldots, {}^{N}/_{q} - 1. \end{aligned} \tag{2.42}$$

Vidimo da ${}^{N}/_{q}$ ulaznih podataka za svaku podsekvencu su obilježeni sa $y_l^{(u)}$ za $u = 0, 1, \ldots q-1$. Da bi pokazali da su algoritmi FFT sa decimiranjem po učestanosti po bazi 2 i po bazi 4 su specijalni slučajevi kada je $q = 2$ i $q = 4$, općenite formule za formiranje svake od $q = 2^s \geq 2$ podsekvence određene su jednačinom (2.42). Pimjećujemo da kada je $q = 2$, y_l u jednačini (2.10) i z_l u jednačini (2.12) odgovaraju $y_l^{(0)}$ i $y_l^{(1)}$ u općoj formuli; kada je $q = 4$, y_l, z_l, g_l i h_l u jednačinama (2.31), (2.32), (2.33) i (2.34) odgovaraju $y_l^{(0)}$, $y_l^{(1)}$, $y_l^{(2)}$ i $y_l^{(3)}$ u općoj formuli.

$$y_l^{(u)} = \left(\sum_{\lambda=0}^{q-1} x_{l+\lambda\frac{N}{q}} W_N^{r(qk+u)} \right) W_N^{ul}, \quad l = 0, 1, \ldots, {}^{N}/_{q} - 1. \tag{2.43}$$

Pošto je poznato da je algoritam FFT po bazi q za $q = 2^2 > 4$ manje efikasan nego optimalni algoritam sa razdvojenom bazom koji se rekurzivno primjenjuje na oba algoritma i po bazi 2 i po bazi 4 za rješavanje podsekvence nećemo dalje ulaziti u razmatranje algoritama sa većom bazom.

2.11 FFT algoritam sa razdvojenom bazom

2.11.1 FFT algoritam sa decimiranjem po vremenu sa razdvojenom bazom

Nakon što smo proučili FFT algoritme po bazi 2 i po bazi 4 interesantno je da vidimo da se kompleksnost FFT algoritma može smanjiti kombinujući ova dva algoritma u algoritam sa razdvojenom bazom (engl. the

Split-Radix FFT). Pristup FFT sa razdvojenom bazom prvi su put predložili 1984. godine Duhamel-a i Hollmann-a . Također postoje verzije decimiranje po vremenu i po učestanosti ovog algoritma, zavisno od toga da li decimiramo ulazni vremenski niz ili izlaznu frekventnu sekvencu .

FFT algoritam sa decimiranjem po vremenu sa razdvojenom bazom izveden je iz jednačine (2.1), koja definiše diskretnu Fourierovu transformaciju kompleksnog vremenskog niza:

$$\begin{aligned} X_r &= \sum_{k=0}^{N-1} x_k \omega_N^{rk}, \quad r = 0, 1, \ldots, N-1 \\ &= \sum_{k=0}^{\frac{N}{2}-1} x_{2k} \omega_N^{r(2k)} + \sum_{k=0}^{\frac{N}{4}-1} x_{4k+1} \omega_N^{r(4k+1)} + \sum_{k=0}^{\frac{N}{4}-1} x_{4k+3} \omega_N^{r(4k+3)} \\ &= \sum_{k=0}^{\frac{N}{2}-1} x_{2k} \omega_N^{r(2k)} + \omega_N^{r} \sum_{k=0}^{\frac{N}{4}-1} x_{4k+1} \omega_N^{r(4k)} + \omega_N^{3r} \sum_{k=0}^{\frac{N}{4}-1} x_{4k+3} \omega_N^{r(4k)}. \end{aligned} \tag{2.44}$$

Decimiranjem vremenskog niza u tri skupa, to jest skup $\{y_k | y_k = x_{2k}, \quad 0 \le k \le {}^{N}/_{2} - 1\}$, skup $\{z_k | z_k = x_{4k+1}, \quad 0 \le k \le {}^{N}/_{4}$ i skup $\{h_k | h_k = x_{4k+3}, \quad 0 \le k \le {}^{N}/_{4} - 1\}$, tri podsekvence će biti definisane nakon što uvedemo $W_{\frac{N}{2}} = W_N^2$ i $W_{\frac{N}{4}} = W_N^4$.

$$Y_r = \sum_{k=0}^{\frac{N}{2}-1} x_{2k} W_N^{r(2k)} = \sum_{k=0}^{\frac{N}{2}-1} x_{2k} \left(W_N^2\right)^{rk} = \sum_{k=0}^{\frac{N}{2}-1} y_k W_{\frac{N}{2}}^{rk}, \quad r = 0, 1, \ldots, {}^{N}/_{2} - 1. \tag{2.45}$$

$$Z_r = \sum_{k=0}^{\frac{N}{4}-1} x_{4k+1} W_N^{r(4k)} = \sum_{k=0}^{\frac{N}{4}-1} x_{4k+1} \left(W_N^4\right)^{rk} = \sum_{k=0}^{\frac{N}{4}-1} z_k W_{\frac{N}{4}}^{rk}, \quad r = 0, 1, \ldots, {}^{N}/_{4} - 1. \tag{2.46}$$

$$H_r = \sum_{k=0}^{\frac{N}{4}-1} x_{4k+3} W_N^{r(4k)} = \sum_{k=0}^{\frac{N}{4}-1} x_{4k+3} \left(W_N^4\right)^{rk} = \sum_{k=0}^{\frac{N}{4}-1} h_k W_{\frac{N}{4}}^{rk}, \quad r = 0, 1, \ldots, {}^{N}/_{4} - 1. \tag{2.47}$$

Nakon što su sve tri podsekvence riješene algoritmom sa razdvojenom bazom, rješenje početne sekvence dužine N može se dobiti prema jednačini(2.44) za $r = 0, 1, \ldots, N-1$. Zato što je $Y_{r+k\frac{N}{2}} = Y_r$ za $0 \le r \le {}^{N}/_{2} - 1$, $Z_{r+k\frac{N}{4}} = Z_r$ za $0 \le r \le {}^{N}/_{4} - 1$ i $H_{r+k\frac{N}{4}} = H_r$ za $0 \le r \le {}^{N}/_{4} - 1$ jednačina (2.44) se može napisati sa uvjetima Y_r, $Y_{r+\frac{N}{4}}$, Z_r, i H_r za $0 \le r \le {}^{N}/_{4} - 1$:

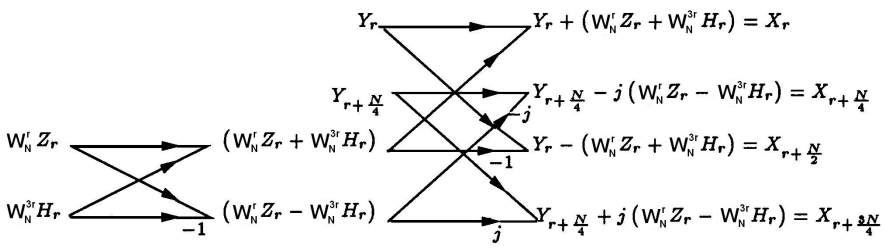
$$\begin{aligned} X_r &= Y_r + W_N^r Z_r + W_N^{3r} H_r \\ &= Y_r + \left(W_N^r Z_r + W_N^{3r} H_r\right), \quad 0 \le r \le {}^{N}/_{4} - 1, \end{aligned} \tag{2.48}$$

$$\begin{aligned} X_{r+\frac{N}{4}} &= Y_{r+\frac{N}{4}} + W_N^{r+\frac{N}{4}} Z_r + W_N^{3\left(r+\frac{N}{4}\right)} H_r \\ &= Y_{r+\frac{N}{4}} - j\left(W_N^r Z_r - W_N^{3r} H_r\right), \quad 0 \le r \le {}^{N}/_{4} - 1, \end{aligned} \tag{2.49}$$

$$\begin{aligned} X_{r+\frac{N}{2}} &= Y_{r+\frac{N}{2}} + W_N^{r+\frac{N}{2}} Z_r + W_N^{3\left(r+\frac{N}{2}\right)} H_r \\ &= Y_{r+\frac{N}{2}} - \left(W_N^r Z_r + W_N^{3r} H_r\right), \quad 0 \le r \le {}^{N}/_{4} - 1, \end{aligned} \tag{2.50}$$

$$\begin{aligned} X_{r+\frac{3N}{4}} &= Y_{r+\frac{N}{4}} + W_N^{r+\frac{3N}{4}} Z_r + W_N^{3\left(r+\frac{3N}{4}\right)} H_r \\ &= Y_{r+\frac{N}{4}} + j\left(W_N^r Z_r - W_N^{3r} H_r\right), \quad 0 \le r \le {}^N/_4 - 1. \end{aligned} \tag{2.51}$$

Računanje predstavljeno jednačinama (2.48), (2.49), (2.50) i (2.51) se odnosi na nesimetrično "butterfly" računanje decimiranja po vremenu, kao što je pokazano na slici 2.41.



Slika 2.41. *Računanje FFT algoritma sa decimiranjem po vremenu sa razdvojenom bazom*

- **Analiza kompleksnosti algoritma**

Da bi odredili kompleksnost DIT FFT algoritma sa razdvojenom bazom vidimo da se $\omega_N^r Z_r$ i $\omega_N^{3r} H_r$ moraju izračunati prije nego što formiramo dvije parcijalne sume. Pošto su ove dvije podsekvence iste dužine ${}^N/_4$, potrebno nam je ${}^N/_2$ kompleksnih množenja i ${}^N/_2$ kompleksnih sabiranja da bi dobili parcijalne sume. Postoje četiri specijalna slučaja koja smo već definisali: dva slučaja množenja sa 1 i dva slučaja množenja sa težinskim faktorom ω_8 sa neparnim eksponentom, tako da imamo $(3+3) \times ({}^N/_2 - 4) + 4 \times 2 + 2 \times {}^N/_2 = 4N - 16$ ne trivijalnih realnih operacija koji se izvode u prvoj fazi "butterfly" računanja. U drugoj fazi "butterfly" računanja potrebno nam je N kompleksnih sabiranja ili $2N$ realnih sabiranja. Ukupni $6N - 16$ netrivijalnih realnih operacija ili flopova. Da bi mogli napisati jednačinu za računanje kompleksnosti algoritma potrebni su nam uslovi ograničenja za $N = 2$ i $N = 4$. Kao što je ranije izvedeno, $T(2) = 4$ i $T(4) = 16$ flopova.

$$T(N) = \begin{cases} T\left(\frac{N}{2}\right) + 2T\left(\frac{N}{4}\right) + 6N - 16, & N = 4^n > 4, \\ 16, & N = 4, \\ 4, & N = 2 \end{cases}$$

ili

$$T(N) = 4N \log_2 N - 6N + 8.$$

2.11.2 DIF FFT sa razdvojenom bazom

Izvođenje FFT algoritma sa decimiranjem po učestanosti sa razdvojenom bazom može se rekurzivno primijeniti na FFT algoritme sa bazom 2 i sa bazom 4 sa decimiranjem po učestanosti da bi riješili podsekvence koji nastaju decimiranjem izlaznog frekventnog niza na sličan način. Frekvenntni niz se rekurzivno desetkuje na tri podskupa, to jest skup opisan sa $Y_k = X_{2k}$ za $0 \le k \le {}^N/_2 - 1$, skup $Z_k = X_{4k+1}$ za $0 \le k \le {}^N/_4 - 1$ i skup $H_k = X_{4k+3}$ za $0 \le k \le {}^N/_4 - 1$. Za početak izvođenja koristimo jednačinu (2.1). Koristeći ranije dobijene rezultate za FFT algoritam sa decimiranjem po učestanosti po bazi 2 u jednačini (2.8), imamo:

$$
\begin{aligned}
X_r &= \sum_{l=0}^{\frac{N}{2}-1} x_l W_N^{rl} + \sum_{l=\frac{N}{2}}^{N-1} x_l W_N^{rl} \\
&= \sum_{l=0}^{\frac{N}{2}-1} \left(x_l + x_{l+\frac{N}{2}} W_N^{r\frac{N}{2}} \right) W_N^{rl}, \quad r = 0, 1, \ldots, N-1.
\end{aligned} \tag{2.52}
$$

Zamjenom $Y_k = X_{2k}$, $y_l = x_l + x_{l+\frac{N}{2}}$ jedna podsekvenca polovine dužine je definisana sa:

$$
\begin{aligned}
Y_k &= X_{2k} = \sum_{l=0}^{\frac{N}{2}-1} \left(x_l + x_{l+\frac{N}{2}} \right) W_{\frac{N}{2}}^{kl} \\
&= \sum_{l=0}^{\frac{N}{2}-1} y_l W_{\frac{N}{2}}^{kl}, \quad k = 0, 1, \ldots, {}^{N}/_{2} - 1.
\end{aligned} \tag{2.53}
$$

Da bi definisali druge dvije podsekvence dužine ${}^{N}/_{4}$ počinjemo od definicije DFT, jednačina (2.1), i rezultata koje smo ranije izveli za FFT algoritam sa decimiranjem po učestanosti po bazi 4:

$$
\begin{aligned}
X_r &= \sum_{l=0}^{N-1} x_l W_N^{rl}, \quad r = 0, 1, \ldots, N-1, \\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(x_l + x_{l+\frac{N}{4}} W_4^{r} + x_{l+\frac{N}{2}} W_4^{2r} + x_{l+\frac{3N}{4}} W_4^{3r} \right) W_N^{rl}.
\end{aligned} \tag{2.54}
$$

Zamjenom $r = 4k+1$ i $r = 4k+3$ dobijamo:

$$
\begin{aligned}
Z_k &= X_{4k+1} = \sum_{l=0}^{\frac{N}{4}-1} \left(x_l + x_{l+\frac{N}{4}} W_4^{4k+1} + x_{l+\frac{N}{2}} W_4^{2(4k+1)} + x_{l+\frac{3N}{4}} W_4^{3(4k+1)} \right) W_N^{(4k+1)l} \\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(\left(x_l - x_{l+\frac{N}{2}} \right) - j \left(x_{l+\frac{N}{4}} - x_{l+\frac{3N}{4}} \right) \right) W_N^{l} W_{\frac{N}{4}}^{kl} \\
&= \sum_{l=0}^{\frac{N}{4}-1} z_l W_{\frac{N}{4}}^{kl}, \quad , k = 0, 1, \ldots {}^{N}/_{4} - 1.
\end{aligned} \tag{2.55}
$$

$$
\begin{aligned}
H_k &= X_{4k+3} = \sum_{l=0}^{\frac{N}{4}-1} \left(x_l + x_{l+\frac{N}{4}} W_4^{4k+3} + x_{l+\frac{N}{2}} W_4^{2(4k+3)} + x_{l+\frac{3N}{4}} W_4^{3(4k+3)} \right) W_N^{(4k+3)l} \\
&= \sum_{l=0}^{\frac{N}{4}-1} \left(\left(x_l - x_{l+\frac{N}{2}} \right) + j \left(x_{l+\frac{N}{4}} - x_{l+\frac{3N}{4}} \right) \right) W_N^{3l} W_{\frac{N}{4}}^{kl} \\
&= \sum_{l=0}^{\frac{N}{4}-1} h_l W_{\frac{N}{4}}^{kl}, \quad , k = 0, 1, \ldots {}^{N}/_{4} - 1.
\end{aligned} \tag{2.56}
$$

Da bi formulisali ove tri podsekvence koristeći dvije faze nesimetričnog "butterfly" računanja grupisat ćemo parcijalne sume.

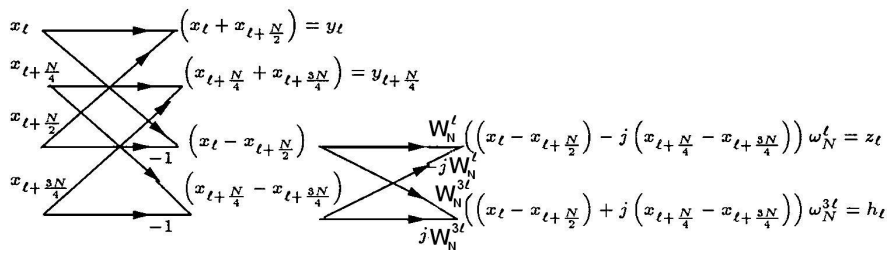
$$y_l = x_l + x_{l+\frac{N}{2}}, \quad 0 \leq l \leq {}^{N}/_{4} - 1. \tag{2.57}$$

$$y_{l+\frac{N}{4}} = x_{l+\frac{N}{4}} + x_{l+\frac{3N}{4}}, \quad 0 \leq l \leq {}^{N}/_{4} - 1. \tag{2.58}$$

$$z_l = \left(\left(x_l - x_{l+\frac{N}{2}}\right) - j\left(x_{l+\frac{N}{4}} - x_{l+\frac{3N}{4}}\right)\right) W_N^l, \quad 0 \leq l \leq {}^{N}/_{4} - 1. \tag{2.59}$$

$$h_l = \left(j\left(x_{l+\frac{N}{4}} - x_{l+\frac{3N}{4}}\right) + \left(x_l - x_{l+\frac{N}{2}}\right)\right) W_N^{3l}, \quad 0 \leq l \leq {}^{N}/_{4} - 1. \tag{2.60}$$

Računanje predstavljeno jednačinama (2.57), (2.58), (2.59) i (2.60) odnosi se na nesimetrično "butterfly" računanje decimiranja po učestanosti, kao što je pokazano na slici 2.42.



Slika 2.42. *Računanje FFT algoritma sa decimiranjem po učestanosti sa razdvojenom bazom*

2.12 FFT algoritmi za realni ulaz

U poglavlju 2.1, diskretna Fourierova transformacija N diskretnih uzoraka kompleksnog vremenskog niza definisana je jednačinom (2.1):

$$\begin{aligned} X_r &= \sum_{l=0}^{N-1} x_l W_N^{rl}, \quad r = 0, 1, \ldots, N-1, \\ &= \sum_{l=0}^{N-1} \left(Re\left(x_l\right) + jIm\left(x_l\right)\right) W_N^{rl}. \end{aligned}$$

Kada su uzorci uzeti iz realnog vremenskog niza možemo ih tretirati kao kompleksne brojeve sa imaginarnim dijelom jednakim nuli, $Im(x_l) = 0$ za $0 \leq l \leq N-1$. Drugim riječima, realni podatak predstavlja specijalan slučaj kada je približno jedna polovina aritmetičkih operacija jednaka nuli. Pošto su mnogi FFT algoritmi izvedeni na realnoj vrijednosti vremenskog niza imamo mogućnost mnogo efikasnijeg rada sa realnim ulazom . Dvije vrste takvih algoritama ćemo obraditi u ovom poglavlju. Prvi algoritam dopušta nam da računamo dva realna FFT-a dužine N računajući jedan kompleksni FFT dužine N. Drugi algoritam dopušta nam računanje realnog FFT-a dužine N računajući kompleksni FFT dužine ${}^{N}/_{2}$.

2.12.1 Istovremeno računanje dva realna FFT-a

U ovom podpoglavlju upoznat ćemo se sa algoritmom koji računa dva realna FFT-a dužine N računajući jedan kompleksni FFT dužine N. Dva skupa realnih brojeva su obilježena sa f_l i g_l za $0 \leq l \leq N-1$. Stavljajući da je $Re\,(x_l) = f_l$ i $Im\,(x_l) = g_l$ dobijamo kompleksne brojeve $x_l = f_l + jg_l$ za $0 \leq l \leq N-1$.

Definicija DFT-a implicira da je:

$$F_r = \sum_{l=0}^{N-1} f_l W_N^{rl} \quad i \quad G_r = \sum_{l=0}^{N-1} g_l W_N^{rl} \quad za\ 0 \leq l \leq N-1,$$

i

$$\begin{aligned} X_r &= \sum_{l=0}^{N-1} x_l W_N^{rl} \\ &= \sum_{l=0}^{N-1} (f_l + jg_l)\, W_N^{rl} \\ &= \sum_{l=0}^{N-1} f_l W_N^{rl} + j \sum_{l=0}^{N-1} g_l W_N^{rl} \\ &= F_r + jG_r. \end{aligned} \tag{2.61}$$

Jedan kompleksni FFT od x_l-ova može se izračunati tako da sadrži X_r-ove i skoro pola aritmetičkih operacija možemo sačuvati ako se F_r-ovi i G_r-ove mogu ponovno vratiti od izračunatih X_r-ova. Ovo se može uraditi koristeći osobinu simetrije za DFT realnih vrijednosti.

Osobina simetrije omogućava da konjugovano kompleksni F_{N-r} je jednak F_r. Ova osobina je izvedena koristeći DFT algoritam i činjenice da je konjugovano kompleksna vrijednost od realne vrijednosti f_l jednaka samoj sebi, $W_N^N = 1$ i da je konjugovano kompleksna vrijednost od W_N^{-rl} jednaka je W_N^{rl}.

$$\bar{F}_{N-r} = \sum_{l=0}^{N-1} \bar{f}_l \left(\bar{W}_N^{Nl}\right) \bar{W}_N^{-rl} = \sum_{l=0}^{N-1} f_l W_N^{rl} = F_r. \tag{2.62}$$

Pošto su g_l-ovi realni imamo da je $\bar{G}_{N-r} = G_r$. Sada kompleksno konjugovan X_{N-r} može se izraziti preko F_r i G_r.

$$\bar{X}_{N-r} = \bar{F}_{N-r} - j\bar{G}_{N-r} = F_r - jG_r. \tag{2.63}$$

Kombinujući jednačine (2.61) i (2.63) dobijamo:

$$F_r = \frac{1}{2}\left(X_r + \bar{X}_{N-r}\right) \qquad i \qquad G_r = \frac{j}{2}\left(\bar{X}_{N-r} - X_r\right).$$

Potrebno je samo $2N$ kompleksnih sabiranja/oduzimanja da bi dobili dva realna FFT-a nakon što je izvedena kompleksna FFT.

2.12.2 Računanje realne FFT

Dobijene rezultate iz prethodnog podpoglavlja ćemo razdvojiti na dva dijela u pola manje dužine. Izvođenje je slično izvođenju FFT sa derivacijom po vremenu u dijelu pod 2.1.1.

$$\begin{aligned} X_r &= \sum_{l=0}^{N-1} x_l W_N^{rl}, \quad r = 0, 1, \ldots, N-1, \\ &= \sum_{l=0}^{\frac{N}{2}-1} x_{2l} W_N^{r(2l)} + W_N^r \sum_{l=0}^{\frac{N}{2}-1} x_{2l+1} W_N^{r(2l)} \\ &= \sum_{l=0}^{\frac{N}{2}-1} x_{2l} \omega_{\frac{N}{2}}^{rl} + W_N^r \sum_{l=0}^{\frac{N}{2}-1} x_{2l+1} W_{\frac{N}{2}}^{rl}. \end{aligned} \tag{2.64}$$

Stavljajući da je $f_l = x_{2l}$, $g_l = x_{2l+1}$ za $0 \leq l \leq {}^N/_2 - 1$ DFT od dva realna niza i DFT od ${}^N/_2$ kompleksnih brojeva $y_l = f_l + jg_l$ je ispod definisan.

$$F_r = \sum_{l=0}^{\frac{N}{2}-1} f_l W_{\frac{N}{2}}^{rl}, \qquad G_r = \sum_{l=0}^{\frac{N}{2}-1} g_l W_{\frac{N}{2}}^{rl},$$

i

$$Y_r = \sum_{l=0}^{\frac{N}{2}-1} y_l W_{\frac{N}{2}}^{rl} = \sum_{l=0}^{\frac{N}{2}-1} \left(f_l + jg_l\right) W_{\frac{N}{2}}^{rl} = \sum_{l=0}^{\frac{N}{2}-1} f_l W_{\frac{N}{2}}^{rl} + j \sum_{l=0}^{\frac{N}{2}-1} g_l W_{\frac{N}{2}}^{rl} = F_r + jG_r.$$

Koristit ćemo rezultate iz prethodnog podpoglavlja, kompleksna FFT y_l-ova može se izračunati da sadrži Y_r-ove, i F_r-ove i G_r-ove (za $0 \leq l \leq {}^N/_2 - 1$) mogu se ponovo dobiti koristeći sljedeće jednačine.

$$F_r = \frac{1}{2}\left(Y_r + \bar{Y}_{N-r}\right), \qquad G_r = \frac{j}{2}\left(\bar{Y}_{N-r} - Y_r\right).$$

Cilj nam je da izračunamo X_r definisano jednačinom (2.64). Sada ćemo uvrstiti i dobijene F_r i G_r.

$$X_r = \sum_{l=0}^{\frac{N}{2}-1} x_{2l} W_{\frac{N}{2}}^{rl} + W_N^r \sum_{l=0}^{\frac{N}{2}-1} x_{2l+1} W_{\frac{N}{2}}^{rl} = F_r + W_N^r G_r, \quad r = 0, 1, \ldots, {}^N/_2 - 1. \tag{2.65}$$

Pošto je x_l realan X_r ima osobinu simetrije izvedenu u jednačini (2.62); također $X_{\frac{N}{2}+1}, X_{\frac{N}{2}+2}, \ldots, X_{N-1}$ možemo dobiti koristeći konjugovano kompleksnu vrijednost prethodno izračunate X_r.

$$X_{N-r} = \bar{X}_r, \quad r = 0, 1, \ldots, {}^N/_2 - 1. \tag{2.66}$$

Koristeći jednačine (2.65) i (2.66) svi X_r-ovi se mogu dobiti izuzev $X_{\frac{N}{2}}$. Da bi izračunali $X_{r+\frac{N}{2}} = F_{r+\frac{N}{2}} + W_N^{r+\frac{N}{2}} G_{r+\frac{N}{2}}$ podsjetimo se da je $F_{r+\frac{N}{2}} = F_r$, $G_{r+\frac{N}{2}} = G_r$ i $W_N^{r+\frac{N}{2}} = -W_N^r$. Koristeći ove osobine sa $r = 0$, $X_{r+\frac{N}{2}} = X_r$ može se izračunati sa

$$X_{\frac{N}{2}} = F_0 + G_0. \tag{2.67}$$

Na kraju potrebno nam je N kompleksnih sabiranja/oduzimanja da bi ponovo dobili F_r i G_r poslije kompleksne FFT koja se izvodi na $N/2$ kompleksnih brojeva i potrebno je još dodatnih $N/2$ kompleksnih množenja i $N/2 + 1$ kompleksnih sabiranja/oduzimanja da bi se izračunali X_r pomoću jednačina (2.65), (2.66) i (2.67). Za veće N gotovo polovina aritmetičkih operacija može se sačuvati izvodeći FFT od $N/2$ kompleksnih brojeva umjesto koristeći N kompleksnih brojeva.

U ovom poglavlju korištena je literatura [1],[2],[9] i [12].

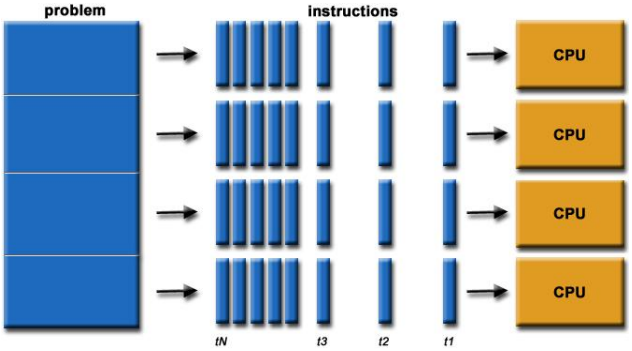
3 Paralelno računanje FFT-a

3.1 Principi paralelnih algoritama

Sekvencijalni algoritam se u osnovi sastoji od osnovnih koraka za rješavanje dobijenog problema. Slično, paralelni algoritam govori nam kako da riješimo problem koristeći multiprocesorske strukture. Zbog preobimnosti problematike nismo ulazili u izvođenje FFT algoritama. U praksi, paralelni algoritam se sastoji od sljedećih koraka:

- identificirati dijelove problema koji se mogu izvršavati istovremeno,
- pridružiti dijelove problema multprocesorima koji se izvršavaju paralelno,
- dijeljenje ulaznih, izlaznih i međupodataka koji su u vezi sa problemom,
- imati pristup dijeljenim podacima,
- sinhronizovati procesore u različitim fazama izvršavanja.

Dijeljenje računanja u manja računanja i pridruživanje istih različitim procesorima predstavljaju dva ključna koraka za izvedbu paralelnih algoritama.

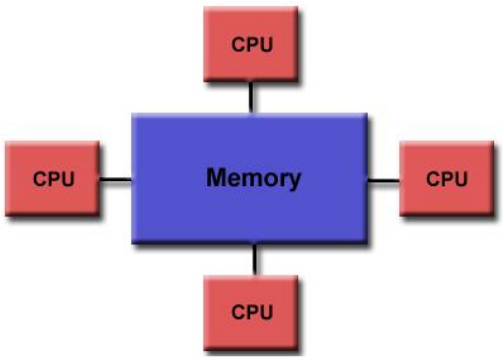


Slika 3.1. *Paralelno računanje*

3.2 Pridruživanje podataka procesorima

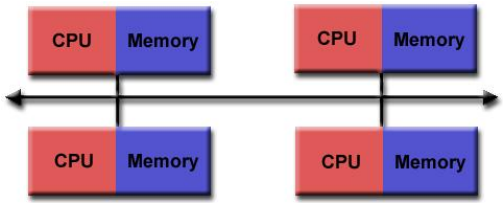
Multiprocesorske strukture se dijele na multiprocesorske strukture sa dijeljenom memorijom i multiprocesorske strukture sa lokalnom memorijom. Kao što im sama imena impliciraju, razlikuju se da li procesor može direktno pristupati memoriji ili je memorija podijeljena na dijelove koji su zasebni za svaki procesor.

Za dijeljenu memoriju potrebno je paralelizirati sekvencijalni algoritam odnosno podijeliti računanje između procesora. Više procesora može raditi zasebno, ali dijele istu memoriju. Promjene u lokaciji memorije uzrokovane od jednog procesora su vidljive drugim procesorima.



Slika 3.2. *Dijeljena memorija*

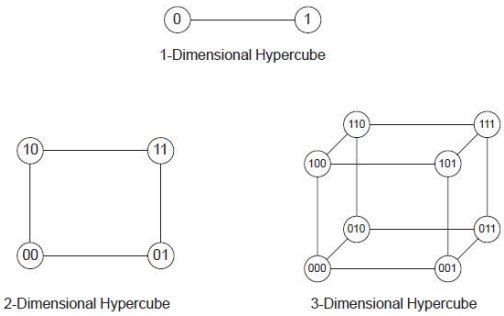
Lokalna memorija zahtjeva da i podatak i računanje budu podijeljeni. Da bi identificirali paralelizam u računanju i dodijelili računanje individualnom procesoru podatak povezan sa računanjem mora biti raspoređen među procesorima i komunikacija između njih je neophodna. Zadatak je uraditi ovo na takav način da svaki procesor ima podatak koji mu je potreban u svojoj lokalnoj memoriji na vrijeme i da je ostvarena komunikacija između procesora tokom računanja.



Slika 3.3. *Lokalna memorija*

3.3 Paraleni 1 - dimenzionalni FFT

Teorija paralelnih FFT algoritama je obrađena u [2], [5], [6], [7] i [8]. Za paralelnu implementaciju 1-dimenzionalnog FFT-a sa dijeljenom memorijom sa p procesora povezanih u kubusa d- tog reda mreža potrebna je topologija. d- dimenzionalni kubus je sistem multiprocesora koji se sastoji od 2^d procesora spojenih u kubuse d-tog reda. Svaki član formira vrh kubusa i čvorište identifikacije razlikuje se tačno za jedan bit od svog d susjeda. Na slici 3.4 pokazane su tipične strukture 1−, 2− i 3− reda kubuse.

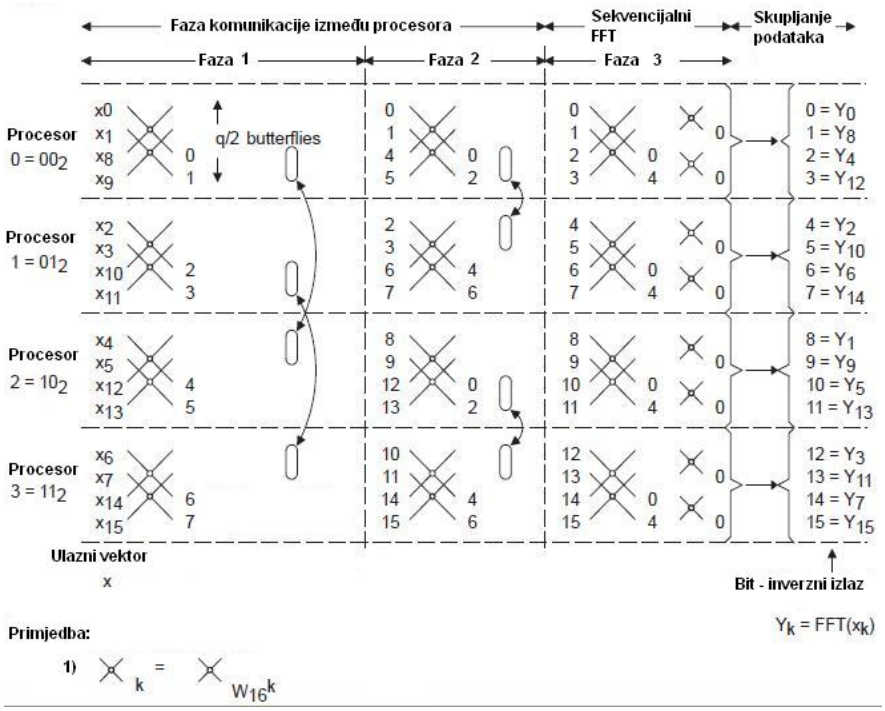


Slika 3.4. *Strukture kubusa* 1−, 2− *i* 3− *reda*

Solowiejczk i Petzinger predložili su interesantan pristup za rješavanje većeg broj uzoraka (> 10000) na dijeljenoj memoriji [6]. Da bi bolje shvatili korake paralelnog algoritma proći ćemo kroz strukturu paralelnog DIF FFT algoritma.

3.3.1 Paralelni DIF FFT

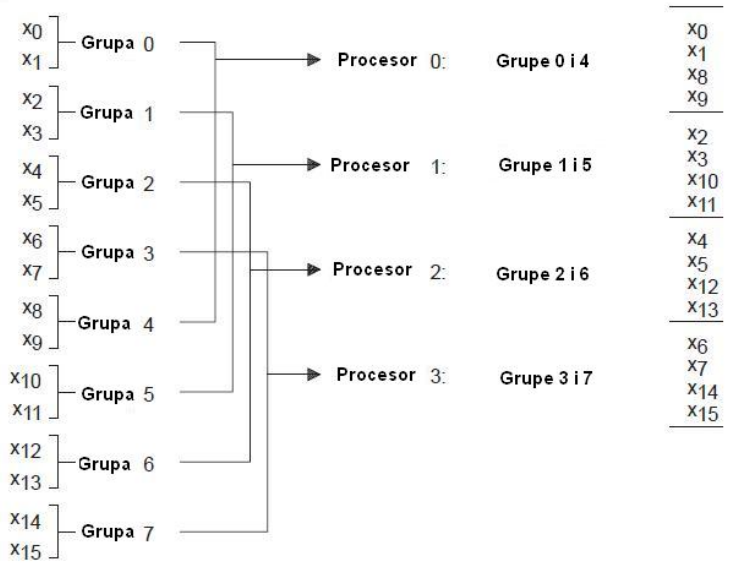
Neka je $n = qp$, gdje n predstavlja dužinu sekvence, p je broj procesora predstavljen u strukturi kubusa i $q \geq 1$ je cijeli broj. Ulazni podaci su predstavljeni u prirodnom redosljedu, a izlazni podaci se nalaze u bit - inverznom redosljedu. Struktura paralelnog algoritma za $n = 16$ i $p = 4$ prikazana je na slici 3.5.



Slika 3.5. *Struktura paralelnog DIF FFT algoritma*

Faza 1. Podjela podataka

Ulazna sekvenca x je podjeljena u $2p$ grupa od kojih se svaka sastoji od $q/2$ kompleksnih brojeva i pridružena je procesoru i, grupama i i $i + p$. Na kraju podjele, procesor i će sadržavati vektor elemenata $(i \cdot q/2) + j$ i $(i \cdot q/2) + n/2 + j$ gdje su $0 \leq j < q/2$ i $0 \leq i \leq p - 1$. Ovaj proces je prikazan na slici 3.6 za $n = 16$ i $p = 4$.



Slika 3.6. *Koraci podjele DIF FFT*

Faza 2. Komunikacija unutar procesora

Komunikacija između procesora je potrebna zato što podsekventna računanja na jednom procesoru zavise od međurezultata na drugim procesorima. U svakom od ovih koraka, svako čvorište:

- računa "butterfly" operacije na svakom od dodjeljenom "butterfly" paru i
- šalje pola izračunatih rezultata ($q/2$ kompleksnih brojeva) čvorištu kojem je potrebno za sljedeći korak računanja i čeka na informacije iste dužine iz drugog čvorišta da bi se nastavilo računanje.

Faza 3. Sekvencijalno računanje

Tokom posljednjih $n-d$ koraka nije potrebna komnunikacija između procesora i sekvencijalni FFT veličine q se nezavisno izvodi na svakom od procesora. Primjećujemo na slici 3.5 da koraci 2 i 3 odgovaraju sekvencijalnom FFT algoritmu sa bazom 4.

Faza 4. Skupljanje podataka

Na kraju FFT računanja, procesor i sadrži q kompleksnih elemenata sa rednim brojevima pozicije $i \cdot q + j$ u izlaznom vektoru.

Za daljnje obradu paralelnih FFT algoritama preporučuje se literatura [2] i [8]

Zaključak

Tema rada je omogućila pored odgovarajuće teorije, izvođenja formula, implementacije odgovarajućih algoritama u Matlabu i eksperimentalnu analizu. U radu su definisani pojmovi Fourierova transformacija, diskretna Fourierova transformacija i brza Fourierova transformacija. FFT je jedna od najlakših i najkorištenijih tehnika za opisivanje i obradu signala. Također, FFT se koristi u električnim filterima, avionima, telekomunikacionim sistemima i u mnogim drugim aspektima modernog života. Analizirane su dvije varijante FFT algoritama, FFT algoritmi sa decimiranjem po vremenu i FFT algoritmi sa decimiranjem po učestanosti. Analizirani su različiti algoritmi u - mjestu, pri tome se ulazna ili izlazna sekvenca nalazi u prirodnom redosljedu ili u bit - inverznom redosljedu, kao i algoritmi koji se ne izvršavaju u - mjestu kojima se i ulazna i izlazna sekvenca nalaze u prirodnom redosljedu. Na uporednom prikazu vremena izvršavanja pojedinih implementacija FFT algoritama za različite ulazne veličine dobijeno je da su FFT algoritmi u funkciji od $N \log_2 N$ što je bilo i za očekivati, pošto smo ranije u izvođenjima dobijali da nam je kompleksnost algoritma u zavisnosti od funkcije $N \log_2 N$. Da bi obezbjedili komunikaciju između korisnika i računara u korištenju datih algoritama na određene funkcije napravljeno je grafičko korisničko okruženje (GUI). GUI je povećao efikasnost i prezentaciju podataka. Pristup digitalnom osciloskopu je omogućio nam je provjeru podataka iz teorije, odnosno obradu signala. Pri tome, postignuto je dobro slaganje rezultata dobijenih na osciloskopu (spektralnom analizatoru) i onih dobijenih korištenjem Matlaba, sa razlikom postojanja šuma mjerenja kod spektralnog analizatora.

Popis slika

Literatura

[1] Melita Ahić - Đokić, *Predavanja iz predmeta "Analiza signala i sistema"*, Elektrotehnički fakultet, Sarajevo, www.etf.unsa.ba, akademska 2008/2009. godina

[2] Inside the FFT Black Box

[3] Bruno A. Olshausen, *PSC 129 - Sensory Processes*, 2000.

[4] Mujo Hebibović, *Predavanja iz predmeta "Digitalni sistemi upravljanja"*, Elektrotehnički fakultet, Sarajevo, www.etf.unsa.ba, akademska 2008/2009. godina

[5] Hwang, K., F. A. Briggs, *Computer Architecture and Parallel Processing*, New York: McGraw-Hill, 1984.

[6] Solowiejczk, Y., J. Petzinger, *Large 1-D Fast Fourier Transforms on a Shared-Memory System*, ICPP91 Proceedings, 1991.

[7] Rose Marie Piedra, *Parallel 1-D FFT Implementation With TMS320C4x DSPs*

[8] Behrooz Parhami, *Introduction to Parallel Processing*, New York, Plenum Press 1999.

[9] dokumentacija za Matlab 7.7.0 softverski paket, R2008b

[10] www.wikipedia.org

[11] Hewlett Packard, *HP 54600 - Series Oscilloscope* , USA, 1993.

[12] Eleanor Chu, *Discrete and Continuous Fourier Transforms*, USA, Chapman & Hall/CRC, 2008.