TAMPEREEN TEKNILLINEN YLIOPISTO

PETRI LEPPÄNEN

RNS-ARITMETIIKKA DSP-JÄRJESTELMISSÄ

Diplomityö

Tarkastajat: prof. Jukka Vanhala ja
prof. Jarmo Takala
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan
tiedekuntaneuvoston
kokouksessa 8. kesäkuuta 2011

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Sähkötekniikan koulutusohjelma
LEPPÄNEN, PETRI: RNS-aritmetiikka DSP-järjestelmissä
Diplomityö, 50 sivua
Toukokuu 2012
Pääaine: Elektroniikka: Mikroelektroniikka
Tarkastajat: prof. Jukka Vanhala ja prof. Jarmo Takala
Avainsanat: RNS, aritmetiikka, DSP, DFT, FFT

Digitaalisella signaalinkäsittelyllä (Digital Signal Processing, DSP) tarkoitetaan kaikenlaista äänen, kuvan ja muiden signaalien käsittelyä. Suosituimmat sovellukset signaalinkäsittelyssä ovat suodattimet ja Fourier-muunnokset. Signaalinkäsittelyllä tarkoitetaan monesti myös diskreettiä Fourier-muunnosta (Discrete Fourier Transform, DFT) ja tarkemmin sen jatkomuunnosta, nopeaa Fourier-muunnosta (Fast Fourier Transform, FFT). DSP-tekniikat ovat viimeaikoina olleet suuressa tutkimuskohteen suosiossa ja sen ansiosta ne ovat kehittyneet huomattavaa tahtia tällä vuosituhannella.

Tärkeimpänä signaalinkäsittelyn osa-alueena on ollut alusta lähtien FFT. Tämän avulla muunnos on mahdollista tehdä paljon nopeammin ja tehokkaammin kuin normaalilla DFT-muunnoksella. Residue-numerojärjestelmän (Residue Number System, RNS) aritmetiikalla pystytään vielä lisäämään nopeutta ja tehokkuutta entisestään.

RNS on numerojärjestelmä kuten tavallinen luonnollinen kymmenkanta, tai kuten digitaalisissa järjestelmissä binääriluku. Tosin se eroaa edellisistä huomattavasti, sillä RNS-luvuilla ei ole kiinteää kantaa vaan se on numeromanipulaatio. RNS koostuu moduuleista, joilla jokaisella on oma painotus.

RNS-järjestelmän suurimmat hyödyt ovat nopeudessa, rinnakkaisuudessa ja virheenkestossa. Järjestelmän rinnakkaisuus tuo esiin suurimmat hyödyt sillä jokainen moduuli voidaan laskea samanaikaisesti. Tämä näkyy suoraan nopeudessa ja yleisesti tehokkuudessa.

RNS-järjestelmällä on kuitenkin suuria rajoituksia ja ongelmakohtia. Nämä ovat selvästi rajoittaneet sen suosiota ja kunnollista läpilyöntiä erilaisissa digitaalisissa sovelluksissa. Jotta RNS-järjestelmää on soveliasta käyttää, on sen suunnitteluun siis panostettava paljon. Tietyt moduulivariaatiot ovat yleisesti tulleet suosituksi ja näin ollen näitä on eniten käytetty sekä tutkittu.

RNS-järjestelmän käyttöä on rajoittanut myös paljon sen käännökset binääriluvusta RNS-luvuksi ja takaisin. Varsinkin takaisinkäännös binääriluvuksi on ollut koko RNS-järjestelmän suurin haaste alusta asti, mikä on jarruttanut sen suurempaa suosiota.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Electrical Engineering
LEPPÄNEN, PETRI: RNS Arithmetic in DSP Systems
Master of Science Thesis, 50 pages
May 2012
Major: Electronics: Microelectronics
Examiner: Professors Jukka Vanhala and Jarmo Takala
Keywords: RNS, Arithmetic, DSP, DFT, FFT

With digital signal processing (DSP) we can manipulate like audio, picture, video-signals and so on. Most popular applications in DSP are different filters and Fourier transforms. Talking about DSP we usually mean discrete Fourier transform (DFT) and especially DFT's modification fast Fourier transform (FFT). Lately DSP has been under quite big research and taken major improvements in this millennium.

With FFT same process can do faster and more efficient than with normal DFT. When doing FFT with Residue Number System (RNS) arithmetic we can get better results in speed and power.

RNS is number system as our conventional numbers or binaries but it's differs a lot from those. RNS is number manipulation and it hasn't base. RNS consist of many different modulus which everyone has own weight.

RNS brings many advantages in speed, paralleling and fault tolerance. Parallelism brings huge advantages to systems when each modulus can calculate same time. This shows directly in speed and power.

RNS brings also many restrictions. Restrictions and problems has been slowed popularity in different DPS systems. It's very important to invest into design to get most out from whole RNS system. Some modulus variations are more popular now days than others. Those give big advantages in speed and in implementation.

Also conversions from binary and back has restricted RNS popularity in DSP systems. Especially reverse conversion from RNS to binary is major problem in whole RNS system and it takes most time and power from it.

ALKUSANAT

Tämä Diplomityö on kirjallisuusselvitys RNS-aritmetiikan ominaisuuksista ja käytöstä sekä sen toteutuksista digitaalisissa sovelluksissa varsinkin FFT-muunnoksessa. Matkan ja vuosien saatossa työ ja sen sisältö on muuttunut monta kertaa.

Työssä on käyty läpi RNS-järjestelmän aritmetiikkaa ja määritelmiä sekä sen vaikutusta digitaaliseen signaalinkäsittelyyn, varsinkin sen tärkeimpiin ominaisuuksiin kuten DFT-muunnokseen ja sen yleisempään muotoon eli FFT-muunnokseen.

Haluan kiittää Tampereen teknillisen yliopiston Elektroniikan sekä Tietokonetekniikan laitosta ja diplomityöni tarkastajia Jukka Vanhalaa ja Jarmo Takalaa tuesta ja joustavuudesta sekä varsinkin vaimoani siitä mitä tämä ikuisuusprojekti on vaatinut.

Lempäälässä, 24.4.2012

Petri Leppänen

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

DSP	Digitaalinen signaalinkäsittely (engl. Digital Signal Processing).
DFT	Diskreetti Fourier-muunnos on Fourier-muunnoksen diskreettiaikainen esitys (engl. Discrete Fourier Transform).
FFT	Nopea Fourier-muunnos (engl. Fast Fourier Transform).
RNS	Residue-numerojärjestelmä, ei-painollinen numerojärjestelmä (engl. Residue Number System).
DIT	engl. Decimation in Time.
DIF	engl. Decimation in Frequensy.
CRNS	Kompleksinen RNS (engl. Complex Residue Number System).
QRNS	engl. Quadratic Residue Number System.
QLRNS	engl. Quadratic Like Residue Number System.
MQRNS	engl. Modified Quadratic Residue Number System.
PRNS	engl. Polynomial Residue Number System.
CRT	Kiinalainen jakojäännös teoreema (engl. Chinese Reminder Theorem).
MRC	Mixed-radix-käännös (engl. Mixed-Radix Conversion).
MRS	Mixed-radix-järjestelmä (engl. Mixed-Radix System).
BR	Eteenpäin käännös binääriluvusta Residue-luvuksi (engl. Binary-to-residue).
RB	Takaisinkäännös Residue-luvusta binääriluvuksi (engl. Residue-to-binary).
CPA	Muisti-bitin monistava summain (engl. Carry-propagate Adder).
CSA	Muisti-bitin muistava summain (engl. Carry-save Adder).
PS	Osittainen summa (engl. Partial sum).
PC	Osittainen muistibitti (engl. Partial carry).
MUX	Multiplekseri, valitsee sisäänmenoista yhden ulostulon.
ROM	Muisti mitä vain voi lukea (engl. Read-only memory).
VLSI	Tekniikka, jossa tuhansia transistoreja pakataan yhteen mikrosiruun (engl. Very-Large-Scale Integration).
DAC	Digitaali-analogia muunnin (engl. Digital-to-analog converter).
DCT	Diskreetti kosini muunnos (engl. Discrete Cosine Transform).

1. JOHDANTO

Digitaalinen signaalinkäsittely (Digital Signal Processing, DSP) on nykypäivänä tärkeä osa-alue monessa sovelluksessa jokapäiväisessä tekemisessä. Tällä vuosituhannella signaalinkäsittely on lyönyt todella joka paikassa itsensä läpi ja sen takia myös sen tutkimiseen on panostettu koko ajan enemmän ja enemmän.

Eri tekniikoita kehitetään koko ajan parantamaan tehokkuutta ja muita signaalikäsittelyn ominaisuuksia. Näistä yksi on jo 1800-luvulla keksitty RNS-järjestelmä (Residue Number System, RNS), joka on nyt tietokoneistumisen myötä tullut uudestaan tutkimusten alle.

Tässä työssä esitellään nopeasti mitä signaalin käsittely yleisesti on ja sen tärkeimpiä ominaisuuksia kuten diskreetti Fourier-muunnos (Discrete Fourier Transform, DFT) ja sen manipulaatio nopea Fourier-muunnos (Fast Fourier Transform, FFT) sekä sen erilaisia ominaisuuksia ja erikoisuuksia. Lisäksi perehdytään RNS-aritmetiikkaan ja -määritelmiin sekä erilaisiin muunnelmiin. Viimeisessä osiossa keskitytään RNS-tekniikkaan ja millä eri tavoin sitä voidaan toteuttaa erilaisissa järjestelmissä.

RNS-järjestelmän suurimpina etuina luetaan nopeus ja tehokkuus sekä virheen sietokyky. Järjestelmän tehokkuus perustuu siihen, että monet perusoperaatiot kuten summaus ja kertolaskut pystytään laskemaan rinnakkaisesti kaikki yhtä aikaa, mitä luonnollisesti tiedetään olevan paljon erilaisissa DSP-sovelluksissa varsinkin FFT-muunnoksessa. Myös virheensietokyky on eduksi RNS-järjestelmissä, sillä virhe ei pääse kumuloitumaan juuri itsenäisten moduulien ansiosta. Näin virheet saadaan pysymään pieninä.

RNS tuo kuitenkin paljon rajoituksia koko järjestelmään ja lisätyötä toteutukseen, joka ei ole ihan suoraviivaista. Kokonaisprosessiin tarvitsee suunnitella käännökset RNS-järjestelmään ja takaisin. Kuitenkin RNS on hyödyksi tarpeeksi suurissa sovelluksissa, joissa on paljon summa- ja kertolaskuja. RNS-aritmetiikan mukaantulo tuo siis hyviä ja huonoja puolia, jolloin sen käyttö pitää ottaa huomioon kokonaistilanteessa päätettäessä onko sitä järkevää käyttää.

Kriittisimmät toteutukset ovat käännöksissä, joissa RNS-järjestelmän tuoma etu voidaan tuhota täysin. Ongelmallisinta on takaisin käännös RNS-järjestelmästä takaisin binääriluvuksi. Tosin monet erilaiset yritykset ja tekniikat jotka helpottavat takaisinkäännöstä ovat tuoneet edistysaskeleita parempaan suuntaan ja näin ollen on saatu käännöstä tehokkaammaksi ja nopeammaksi.

Takaisinkäännökseen vaikuttavat eniten valitut moduulit ja menetelmät. Suuressa mittakaavassa RNS-järjestelmää suunniteltaessa on valittava oikea määrä ja oikeanlaiset moduulit. Tietyt moduulit on helpompi toteuttaa kuin toiset. Myös moduulien määrällä pystytään vaikuttamaan tehokkuuteen huomattavasti.

Perinteisille ja yleisimmille RNS-järjestelmän toteutuksille on olemassa jo vakiintuneita tapoja. Nämä helpottavat tekniikan yleistymistä ja RNS-aritmetiikan mukaantuloa muihinkin sovelluksiin. Tosin edelleen RNS:n ongelmat ja sen tuomat rajoitukset ovat hillinneet sen suosiota sekä käyttöä.

2. DIGITAALINEN SIGNAALINKÄSITTELY

Digitaalinen signaalinkäsittely on tämän vuosituhannen yksi tutkituimmista ja käytetyimmistä teknologioista, jota on kehitetty suurin harppauksin ja hyödynnetty useissa sovelluksissa. Nykyään signaalinkäsittelyä tehdään pääasiassa tietokoneiden avulla, joten digitaalisen signaalinkäsittelystä on tullut yksi merkittävimmistä teknologioista tänä päivänä.

Yksi suosituimmista digitaalisen signaalikäsittelyn sovelluksista on diskreetti Fourier-muunnos. Kun puhutaan diskreetistä Fourier-muunnoksesta, tarkoitetaan yleensä nopeaa Fourier-muunnosta.

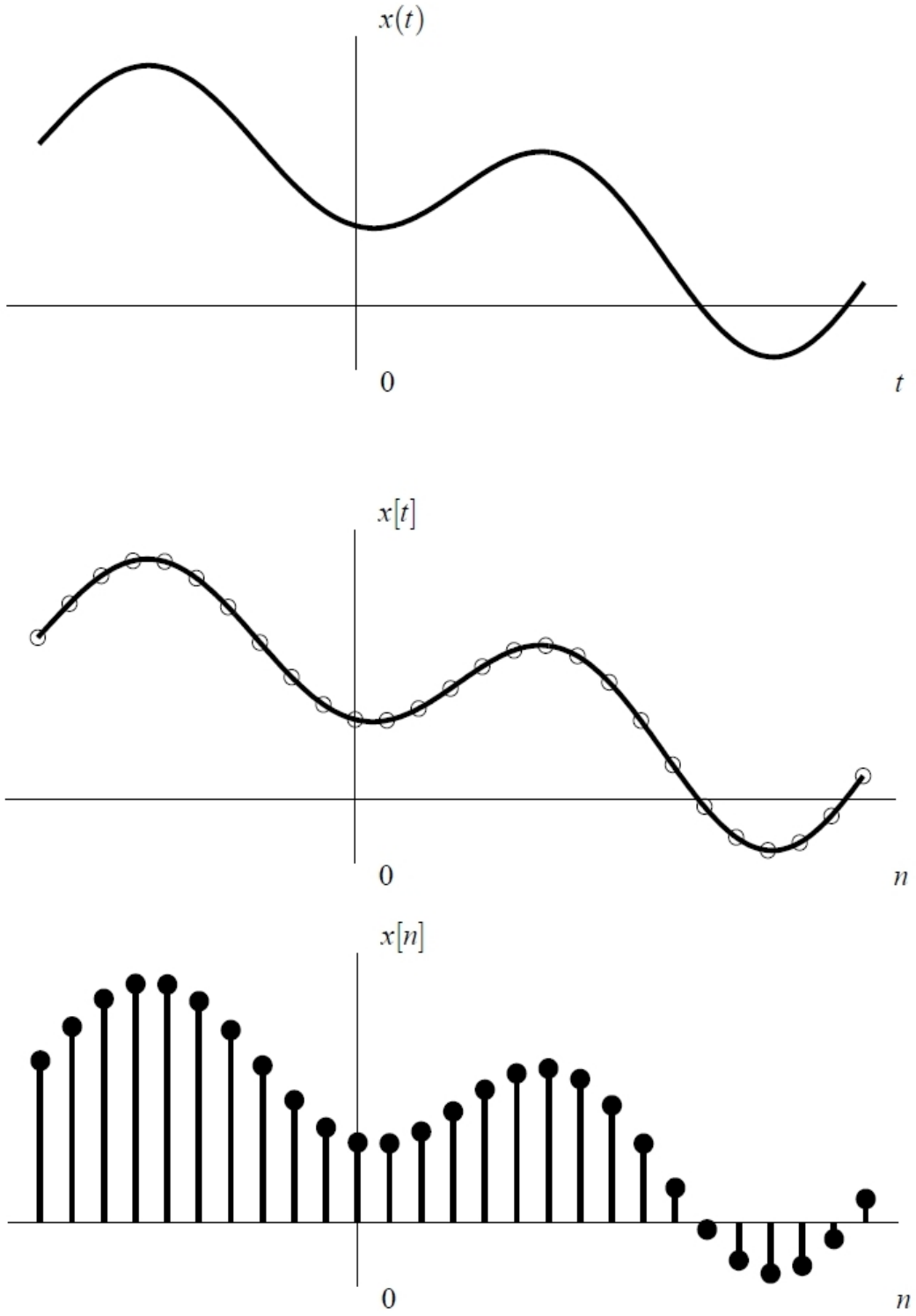
Historiallisesti signaalinkäsittelyn juuret ovat lähtöisin elektroniikasta ja signaaleilla yleisesti tässä yhteydessä tarkoitettiin sähköisiä signaaleja kuten puhelinlinjoja taikka radioaaltoja. Termi digitaalinen tulee englantilaisesta sanasta digital, joka tarkoittaa numeroa. Digitaalinen signaali täten tarkoittaa sarja numeroita, yleisesti sarja binäärinumeroita.

Digitaalisen signaalinkäsittelyn tavoite yleisesti on mitata, suodattaa tai pakata pienempään tilaan analogisia signaaleja. Jotta signaalia voidaan muokata, on se ensin muunnettava digitaaliseen muotoon. Esimerkiksi analoginen sähköinen signaali voidaan digitalisoida käyttämällä analog-to-digital-muunninta. Tämä mahdollistaa sähköisestä signaalisen esittäminen digitaalisena, jossa jokainen bitti edustaa sähköistä signaalia näytteenottohetkellä.

Yleisesti halutaan, että myös lopullinen signaali olisi analoginen. Tätä varten tarvitsee tehdä toinen muunnos käyttämällä digital-to-analog-muunninta. Vaikka tämä operaatio on hankala, niin signaalit kannattaa silti muuntaa välillä digitaaliseen muotoon, sillä digitaalinen signaalinkäsittely mahdollistaa monipuolisemman ja nopeamman käsittelyn halutulle signaalille.

2.1. Diskreetti Fourier-muunnos

Analogisen signaalin muuntaminen digitaaliseksi yleisesti hoidetaan ottamalla analogisesta signaalista $x(t)$ näytteitä tasaisin väliajoin, jolloin tuloksena on diskreettiaikainen digitaalinen signaalijono $x[n]$. Signaali täten saa arvoja vain tietyillä ajanhetkillä. Kuva 1. havainnollistaa tätä tekniikkaa.



Kuva 1. Näytteenotto analogisesta signaalista

Signaalit yleensä esitetään aika-akselin sijaan taajuusakselilla. Aikatason signaalien muuntaminen taajuustasoon tehdään usein Fourier-muunnoksen avulla. Täten signaalin eri taajuuksia on helpompi tulkita ja muokata.

Fourier-muunnos on matematiikassa käytetty jatkuva integraalimuunnos. Sitä käytetään yleisesti signaalin sisältämien taajuuksien analysointiin. Fourier-muunnos voidaan tehdä jatkuvalle tai diskreetille signaalille.

Fourier-muunnos on jatkuva integraalimuunnos. Jatkuva-aikaisen signaalin $x(t)$ ($t \in \mathbf{R}$) Fourier-muunnos ja sen käänteismuunnos määritellään seuraavasti: [8]

$$X(e^{i\omega t}) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t} dt \tag{1}$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(e^{i\omega}) e^{i\omega t} d\omega, \tag{2}$$

missä $i = \sqrt{-1}$ ja $e^{i\omega t}$ -Eulerin kaavan mukaan $e^{i\omega t} = \cos(\omega t) + i\sin(\omega t)$

Fourier-muunnoksen tuloksena on muuttujan $\omega \in \mathbf{R}$ funktio, jota kutsutaan yleisesti kulmataajuudeksi. Yleisesti Fourier-muunnoksen tuloksena on kompleksiarvoinen funktio. Fourier-muunnettu funktio voidaan ajatella alkuperäisen funktion esityksenä taajuustasossa.

Muunnettaessa jaksollinen ja diskreettiaikainen signaali muunnetaan taajuusulotteiseksi, puhutaan diskreetistä Fourier-muunnoksesta. Tulokseksi saadaan diskreettiaikainen jaksollinen signaali taajuustasossa.

Diskreettiaikainen signaali on jaksollinen, jos on olemassa sellainen $N \in \mathbf{N}$, että $x[n] = x[n + N]$, kaikilla indeksin arvoilla n. Näin ollen lukua N sanotaan jakson pituudeksi.

Diskreetti Fourier-muunnos on Fourier-muunnoksen diskreettiaikainen yleistys, jossa signaali ajatellaan sarjaksi. Tällöin se voidaan esittää äärellisenä Fourier-sarjana ja näin integraali muunnoksessa korvautuu summalausekkeeksi seuraavasti: [8]

$$X(n) = \sum_{k=0}^{N-1} x(k) e^{-i2\pi kn/N}, n = 1,..., N-1 \tag{3}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{i2\pi kn/N}. \tag{4}$$

Yleisesti vakiota $e^{-i2\pi/N}$ kutsutaan yleisemmin englantilaisessa kirjallisuudessa twiddle-kerroin, ja siitä käytetään merkintää w_N. [8]

Kun on kysymys jaksollisesta signaalista, tarvitsee muunnoksessa tietää vain yksi jakso. Koko muunnos ja käänteismuunnos voidaan näin esittää vektoreiden avulla. Tällöin itse koko operaatio on matriisikertolasku.

DFT matriisikertolaskuna on yleinen ja helpoin tulkitsemismuoto ja näin voidaan helpoiten käyttää tietokoneen laskentatehoa hyödyksi. Twiddle-kerroin muuntuu isoksi matriisiksi ja näin DFT-muunnos on vektorin ja twiddle-kerroin matriisin kertolasku.

N-pisteen DFT-muunnos esitettynä *N* x *N*-matriisin ja signaalin kertolaskuna *X = W x*, missä *x* on muunnettava signaali ja *X* signaalin DFT. Muunnosmatriisi ja käänteismatriisi ovat näin ollen muotoa

$$\left[F_N\right]_{nm} = W_N^{nm}, n, m = 0, \ldots, N-1 \tag{5}$$

$$\left[F_N\right]_{nm}^{-1} = \frac{1}{N} W_N^{-nm}, \tag{6}$$

jossa *n* ja *m* merkitsevät matriisin elementtejä ja W_N^{nm} twiddle-kerroinmatriisia [9]

$$W_N^{nm} = \begin{bmatrix} w_N^0 & w_N^0 & w_N^0 & \cdots & w_N^0 \\ w_N^0 & w_N^{-1} & w_N^{-2} & \cdots & w_N^{-(N-1)} \\ w_N^0 & w_N^{-2} & w_N^{-4} & \cdots & w_N^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_N^0 & w_N^{-(N-1)} & w_N^{-2(N-1)} & \cdots & w_N^{-(N-1)(N-1)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w_N^{-1} & w_N^{-2} & \cdots & w_N^{-(N-1)} \\ 1 & w_N^{-2} & w_N^{-4} & \cdots & w_N^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w_N^{-(N-1)} & w_N^{-2(N-1)} & \cdots & w_N^{-(N-1)(N-1)} \end{bmatrix}. \tag{7}$$

Fourier-muunnosten laskeminen suoraan määritelmän mukaan on käytännössä hankalaa, mutta koska DFT sisältää redundansseja kertoimia, niin sopivalla matriisihajotelmalla säästetään huomattavasti aikaa. *N*-pisteen vektorin ja *N* x *N*-matriisin kertolasku vaatii N^2 kertolaskua ja yhtä monta yhteenlaskua. Kun näytteitten määrä kasvaa, tulee DFT-menetelmä liian hitaaksi ja se rajoittaa huomattavasti Fourier-muunnoksen käyttökelpoisuutta. Tätä varten on kehitetty parempia tekniikoita jotka suoriutuvat operaatiosta paljon nopeammin. Näitä kutsutaankin nopeiksi Fourier-muunnoksiksi.

2.2. Nopea Fourier-muunnos

Nopea Fourier-muunnos mahdollistaa suorittaa DFT-muunnoksen vähemmällä määrällä aritmeettisia operaatioita kuin suora matriisikertolasku DFT-matriisin mukaan. FFT on yhtäpitävä ja muunnoksessa pätee samat ominaisuudet kuin DFT:ssä. DFT sisältää redundantteja laskutoimituksia ja nopeissa algoritmeissa otetaan nämä huomioon ja vältetään jo kertaalleen laskettujen välitulosten laskeminen uudelleen. FFT algoritmit perustuvat täten yleensä muunnelmiin, jossa twiddle-kerroin saa uuden muodon käyttämällä hyödyksi DFT-matriisin symmetriaa ja jaksollisuutta. [8]

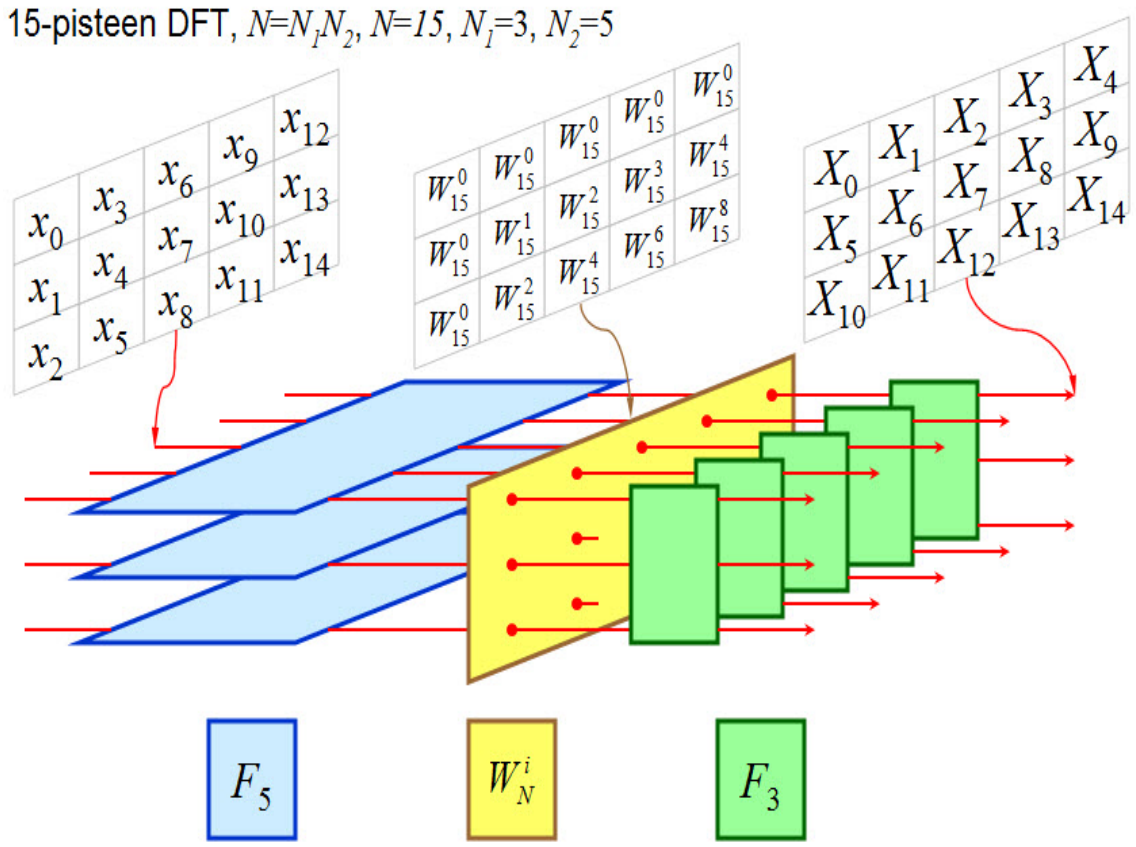
Historian aikana on kehitetty erilaisia FFT algoritmeja, mutta ensimmäinen ja tunnetuin FFT algoritmi on Cooley-Tukey algoritmi. Tämän algoritmin keksivät sen nimen mukaan J.W. Cooley IBM:stä ja John W. Tukey Princetonin yliopistosta vuonna 1965. [9]

Algoritmi on tosin peräisin jo vuodelta 1805, jonka kehitti Carl Friedrich Gauss. Cooley ja Tukey keksivät menetelmän uudelleen, kun he keksivät tavan käyttää algoritmia tietokoneen avulla.

Kuten DFT:n määritelmästä nähdään, tarvittavien laskentaoperaatioiden määrä on verrannollinen näytepisteiden määrän neliöön $O(N^2)$. Cooley ja Tukey esittivät DFT-muunnoksen uudella tavalla, jonka avulla DFT-muunnos voidaan laskea aritmeettisella kompleksisuudella $O(N \log N)$. Näin ollen FFT:n nopeusero verrattuna DFT:n laskemiseen on hyvin merkittävä, kun näytepisteiden määrä on suuri.

2.2.1. Cooley-Tukey FFT

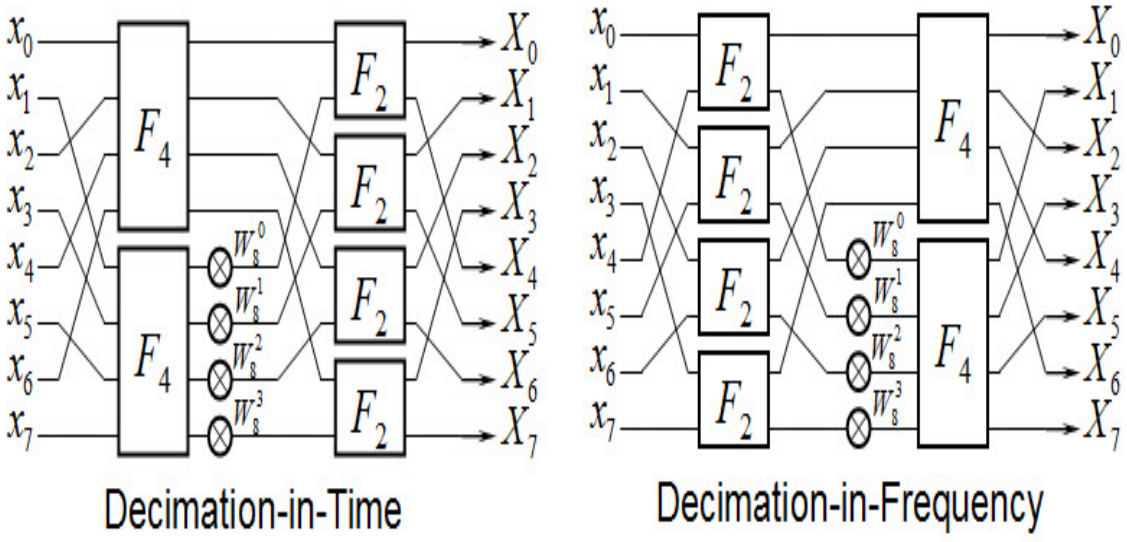
Cooley-Tukey esitys perustuu suuren näytepisteiden omaavan DFT-muunnoksen jakamisen pienempiin osiin. Näin tuloksena on monta pienempää DFT-muunnosta yhden ison sijaan. Perusajatus on siis, että signaali N-näytteinen signaali desimoidaan pienempiin kokonaisuuksiin N_1 ja N_2, $N = N_1N_2$. Seuraavaksi suoritetaan N_2-kokoinen DFT N_1 kertaa. Tämä tulos kerrotaan ns. twiddle-kerroinmatriisilla. Lopuksi suoritetaan N_1 kokoinen DFT N_2 kertaa. Kuva 2. havainnollistaa tätä menetelmää.



Kuva 2. 15-pisteen DFT jaettuna kahteen pienempään muunnokseen.

Menetelmällä saadaan etua, koska N_1 tai N_2-mittainen DFT voidaan laskea vähemmällä määrällä aritmeettisia operaatioita ja sitä yleisesti kutsutaan englannin termeillä radix. Jos N_1 on radix, algoritmia kutsutaan decimation-in-time algoritmiksi (DIT), kun taas N_2 ollessa radix, algoritmia kutsutaan decimation-in-frequensy algoritmiksi (DIF). DIT- ja DIF-tekniikoilla on eroa karkeasti vain siinä, kummasta päästä signaalia muokataan diskreettiin muotoon. Toisin sanoen muokkaamalla sisään

menevää signaalia $x(n)$ pienempiin kokonaisuuksiin, saadaan joukko muunnoksia, jota kutsutaan DIT-algoritmiksi. Muunnosta kutsutaan taas DIF-algoritmiksi, kun signaalia muokataan signaalin ulostulon päästä $X(n)$. [10] [11]



Kuva 3. 8-pisteen Fourier-muunnoksen ero DIT- ja DIF-algoritmeilla.

Kuvassa 3. on esitetty edelliset algoritmit vuokaaviona. F2 ja F4 kuvaavat lohkoa, joissa suoritetaan tässä tapauksessa joko kahden pisteen tai neljän pisteen muunnos nimen mukaisesti. Näistä yleisesti puhutaan englannin termistöllä radix-2 tai radix-4 kuvannollistaen monenko pisteen muunnos on kyseessä.

Cooley-Tukey algoritmin idea perustuu muunnokseen, tutkimalla DFT:n määritelmää (kaava 3) tarkemmin ja erityisesti twiddle-kerrointa $w_N = e^{-i2\pi/N}$, huomataan, että

$$w_{2k}^2 = e^{2(-i2\pi)/2k} = e^{-i2\pi/k} = w_k . \tag{8}$$

Kun näytteen pituus on 2^k, se voidaan jakaa kahteen pienempään muunnokseen, parillisiin ja parittomiin, joiden pituus on $N/2$. Diskreetti Fourier-muunnos jaksolliselle $x(n)$ lukujonolle, jonka pituus on $N = 2^k$

$$X(n) = \sum_{k=0}^{N-1} x(k) w_N^{nk} = \underbrace{\sum_{k=0}^{N/2-1} x(2k) w_N^{n(2k)}}_{parilliset} + \underbrace{\sum_{k=0}^{N/2-1} x(2k+1) w_N^{n(2k+1)}}_{parittomat}, \tag{9}$$

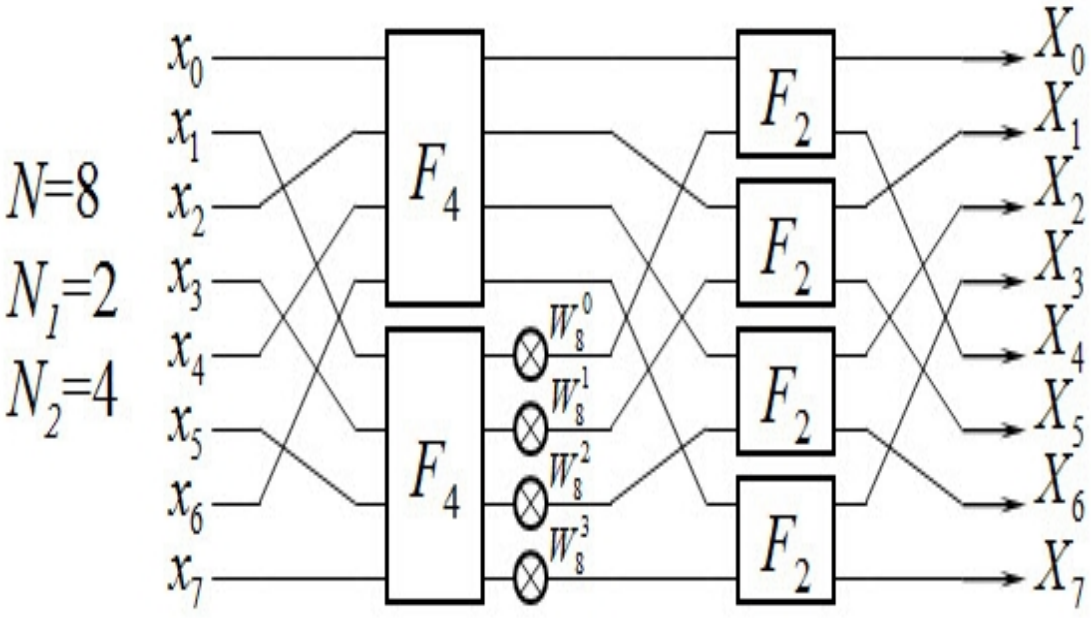
koska $w_N^2 = w_{N/2}$, niin

$$X(n) = \underbrace{\sum_{k=0}^{N/2-1} x(2k) w_{N/2}^{nk}}_{DFT\ parillisille} + w_N^n \underbrace{\sum_{k=0}^{N/2-1} x(2k+1) w_{N/2}^{nk}}_{DFT\ parittomille}. \quad (10)$$

Nyt pienemmillä muunnoksilla on pituus $N/2$, joten tarvitsee laskea vain $N/2$ ulostuloa. Kun merkitään $x_0(k) = x(2k)$ ja $x_1(k) = x(2k + 1)$, DFT jaksollisen määritelmän mukaan ulostulot ovat samat $X_0(k + N/2) = X_0(k)$ ja $X_1(k + N/2) = X_1(k)$ sekä $w_N^{k+N/2} = -w_N^k$. Nyt saadaan yhtälö muotoon,

$$X(n) = \begin{cases} X_0(k) + w_N^k X_1(k), \ jos\ k < N/2 \\ X_0(k - N/2) - w_N^k X_1(k - N/2), \ jos\ k > N/2 \end{cases}. \quad (11)$$

Kuvassa 4. on vuokaavio 8-pisteen muunnoksesta. Siinä on jaettu 8-pisteen muunnos kahteen pienempään muunnokseen desimoimalla signaali kahteen pienempään osaan. Ensin on kaksi 4-pisteen DFT-muunnosta, joissa sisääntuloissa ovat parilliset sekä parittomat pisteet keskenään. Tämän perään on neljä 2-pisteen muunnosta.



Kuva 4. N-pisteen vuokaavio DIT-algoritmin DFT-muunnoksesta jakamalla se ensin kahteen N/2-pisteen DFT-muunnokseen. Tässä tapauksessa N = 8.

2.2.2. Radix-2 FFT

Cooley-Tukey algoritmin perusta on kahden pisteen Fourier-muunnos. Siinä muunnos, jonka pituus on $N = 2^k$, muunnetaan moneksi Fourier-muunnokseksi, joissa pituus on vain kaksi. Kuvassa 4 on perusajatus tästä menetelmästä.

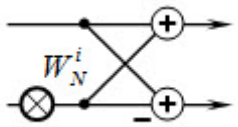
Radix-2 DIT ja DIF FFT-muunnokset ovat yksinkertaisimmat FFT-muunnokset. Cooley-Tukey FFT ja siis myös radix-2-muunnos perustuu ”hajota ja hallitse” -

menetelmään. Nopeus muunnokseen saadaan käyttämällä uudestaan pienempien muunnosten tuloksia. [11]

Radix-2-menetelmää jatketaan rekursiivisesti vielä eteenpäin kunnes kaikki muunnokset ovat vain kahden pisteen muunnoksia. Kahden pisteen muunnoksessa on siis vain kaksi sisääntuloa ja kaksi ulostuloa. Ottamalla huomioon kaavan 11 määritelmä, tulokseksi saadaan

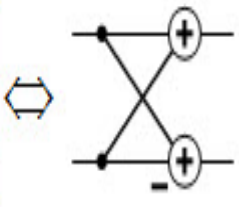
$$y_0 = x_0 + w_N^k x_1$$
$$y_1 = x_0 - w_N^k x_1$$

=>



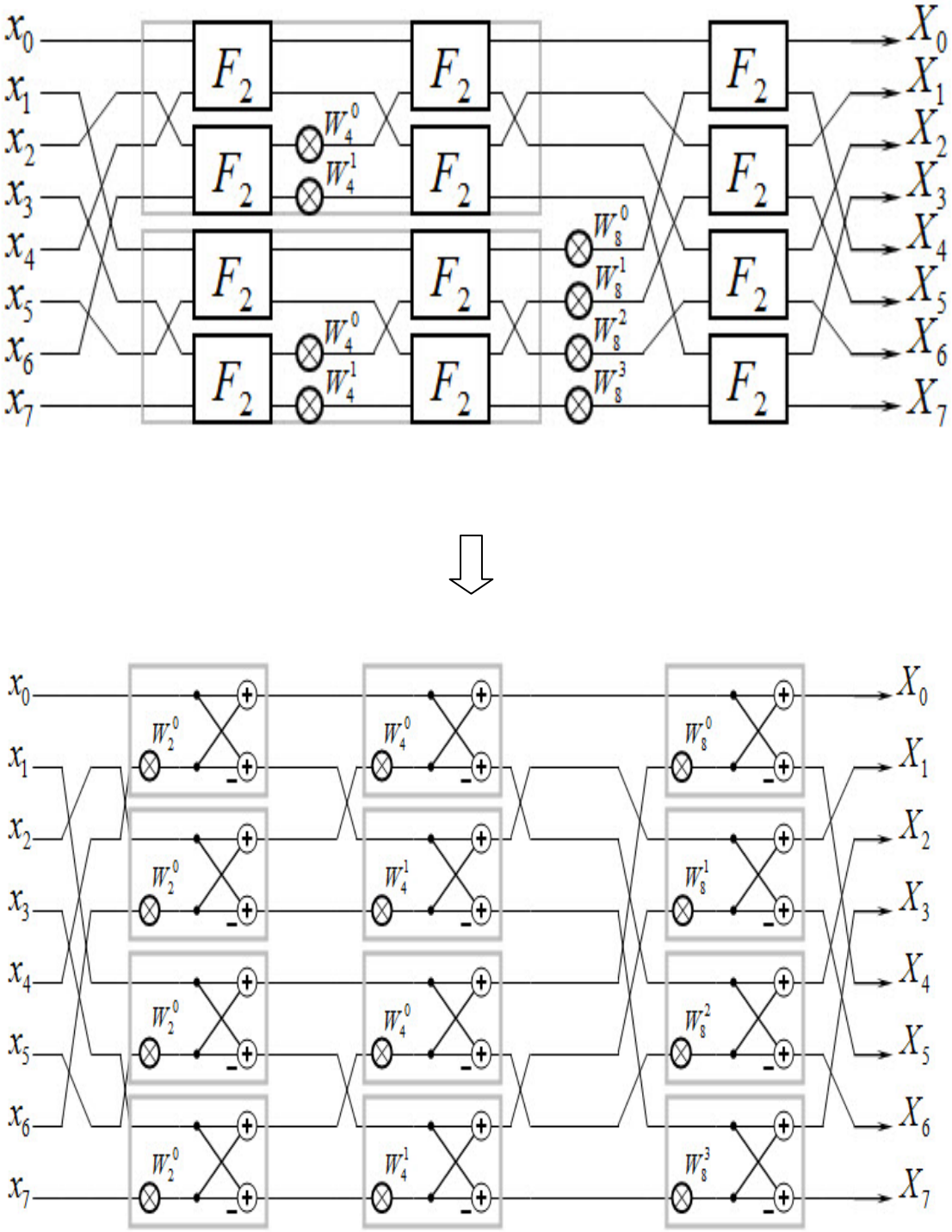
Tätä yhtälöparia kutsutaan termillä ”butterfly”, joka tulee perhosen muotoisesta signaalivuokaaviosta, kuten kuvasta 5. voidaan nähdä. Radix-2 DFT-muunnosmatriisi on hyvin yksinkertainen ja juuri tähän algoritmin tehokkuus perustuu. Yksi radix-2-muunnos saadaan siis tehtyä vain yhdellä kompleksisella tulolla ja kahdella summalla.

$$F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$



Kuva 5. Radix-2-muunnosmatriisi ja ns. butterfly

Kuvassa 6. on sama muunnos kuin kuvassa 4. Erotuksena on, että muunnos on nyt pilkottu kolmeen vaiheeseen jotta saadaan aikaiseksi vain kahden pisteen muunnoksia ja koko muunnos on näin ollen tehty kokonaan radix-2-muunnoksella. Tällaisia muunnoksia, missä käytetään vain yhtä radix-menetelmää, kutsutaan muunnosta radixin mukaan.



Kuva 6. Radix-2 DFT. 8-pisteen DFT-muunnoksen vuokaavio hajotettuna kolmeen osavaiheeseen, jossa tuloksena vain radix-2-muunnoksia.

Radix-4-algoritmi on taas muunnos, missä laskenta tapahtuu käyttämällä 4-pisteen DFT-muunnoksia. Tällöin signaalin pituus on $N = 4^k$. Erityisesti radix-2 ja radix-4-algoritmit ovat tehokkaita, koska ne sisältävät triviaaleja kertoimia 0, 1 ja -1. Nämä muunnokset ovat myös suosittuja. Tehokkuus perustuu yksinkertaiseen twiddle-kertoimeen ja näin ollen perhoslaskenta ja etenkin kertolasku on hyvin yksinkertainen.

Mixed-radix-algoritmissa on käytetty useampaa radix-algoritmia. Esim. kuvassa 4. on 8-pisteen signaalin muunnos toteutettu radix-4 ja radix-2 menetelmin.

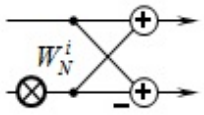
2.2.3. Radix-4 FFT

Radix-4 FFT-muunnos perustuu nimensä mukaisesti menetelmään, jossa ryhmien muunnosten pituus on vain neljä. Radix-4-menetelmässä vaaditaan vain 75% kompleksisia kertolaskuja kuin radix-2-menetelmässä. Tämä tuo huomattavaa etua käyttää radix-4-menetelmää. Radix-2- ja radix-4-menetelmät ovatkin suosituimmat menetelmät mitä käytetään FFT-muunnoksissa. [12]

Radix-4-menetelmässä muunnos jaetaan neljään osaan joiden pituus on $N/4$, jotka summataan yhteen, aivan kuten radix-2-menetelmässä jaettiin kahteen osaan. Tässä tapauksessa FFT-muunnoksen pituus pitää olla $N = 4^k$. Radix-4-muunnoksen tehokkuus perustuu siihen, että lyhyiden FFT-muunnoksien jaksollisuuden $N/4$ vuoksi voidaan laskea toistuvasti $X(n)$, $X(n + N/4)$, $X(n + N/2)$ ja $X(n + 3N/4)$. Perusmenetelmä on sama kuin radix-2-muunnoksessa, mutta nyt perhoslaskenta on monimutkaisempi. Kuvassa 7. nähdään radix-2 ja radix-4-muunnosmatriisit, sekä kuvassa 8. 16-pisteen radix-4-muunnos.

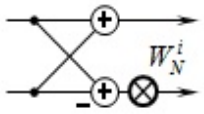
•Radix-2 Decimation-in-Time Butterfly

$$B_2^{DIT} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & \\ & W_N^i \end{pmatrix}$$



•Radix-2 Decimation-in-Frequency Butterfly

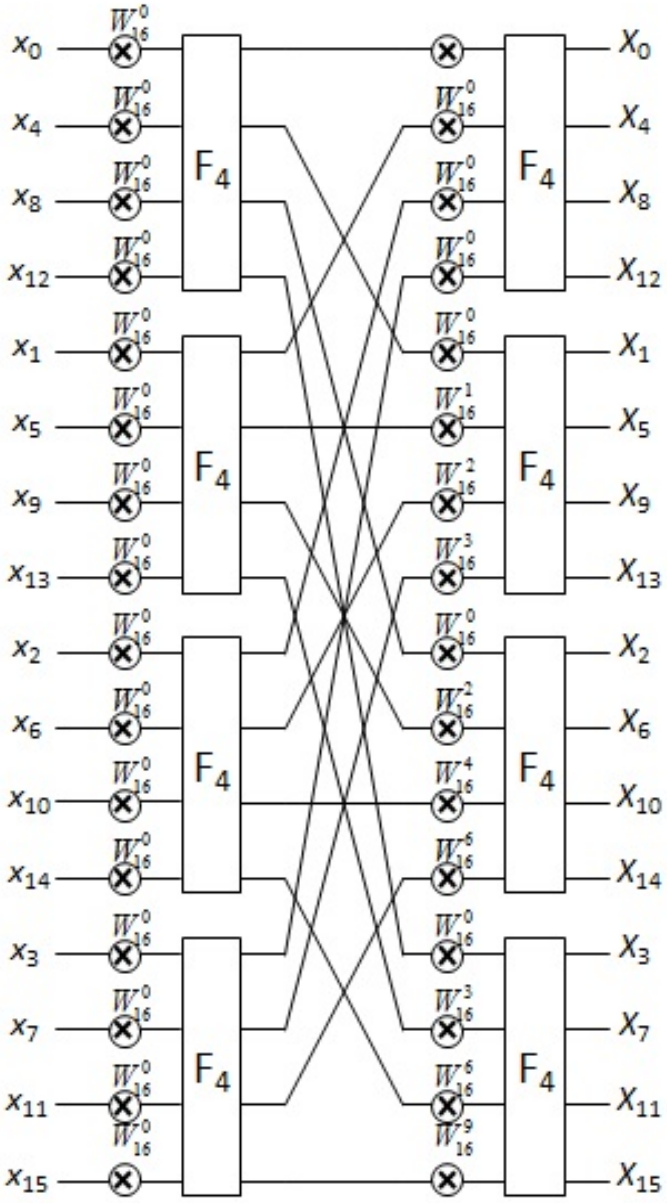
$$B_2^{DIF} = \begin{pmatrix} 1 & \\ & W_N^i \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$



•Radix-4 DIT and DIF Butterfly

$$B_4^{DIT} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \begin{pmatrix} 1 & & & \\ & W_N^i & & \\ & & W_N^{2i} & \\ & & & W_N^{3i} \end{pmatrix} \quad B_4^{DIF} = \begin{pmatrix} 1 & & & \\ & W_N^i & & \\ & & W_N^{2i} & \\ & & & W_N^{3i} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix}$$

Kuva 7. Radix-2- ja radix-4-muunnosmatriisit.



Kuva 8. Radix-4 DIT FFT-muunnos.

3. RESIDUE-NUMEROJÄRJESTELMÄ

Digitaalinen maailma ja tietokoneet ovat riippuvaisia numeroista ja numerosysteemeistä, sekä siitä miten ne esittävät itse numerot järjestelmissään. Nykyaikana vallitsevimmat ja tunnetuimmat numerojärjestelmät ovat kymmenjärjestelmä sekä digitaalimaailmassa yleisesti käytetty binäärinen järjestelmä. Nämä kaksi numerojärjestelmää ovat jo niin itsestäänselvyyksiä, että on unohdettu muut vaihtoehdot melkein kokonaan ja näitä kahta pidetään itsestään selvyyksinä

Residue-numerojärjestelmä on aivan erilainen numerojärjestelmä kuin kaksi edellä kuvattua. RNS poikkeaa niin paljon sekä kymmenjärjestelmästä että binääristä kuin nämä kaksi edellistä toisistaan. RNS ei perustu kiinteään kantaan (esim. 10- ja 2-kanta), vaan on ns. numeromanipulaatio. [1]

RNS-järjestelmän teoreettiset perusteet ovat keksineet suuret matemaatikot kuten Euler, Fermat ja Gauss 1700- ja 1800-luvulla. RNS-järjestelmä sai uutta tuulta alleen 1960-luvulla, kun sitä käytettiin digitaalisessa signaalinkäsittelyssä ja -laskennassa. Tällöin huomattiin RNS-järjestelmän potentiaali nopeudessa, rinnakkaisuudessa ja virheensiedossa. [2]

3.1. RNS-menetelmän kuvaus

RNS on kokonaisluku, ei-painollinen numerojärjestelmä, jolla on eri painotus jokaisella numerolla. RNS koostuu moduuleista, jotka ovat positiivisia alkulukuja. Moduulit ovat keskenään alkulukuja ja esitysmuoto riippuu täysin valitusta kannasta, sillä painotukset riippuvat valituista moduuleista. RNS-järjestelmän poikkeavuus tavallisista numerojärjestelmistä aiheuttaa sen, että jopa normaalit laskutoimitukset ovat erilaisia totutuista. RNS-järjestelmällä ei edes ole kaikkia toimintoja kuten voisi ajatella kymmenjärjestelmässä olevan.

Järjestelmänsä vuoksi myös kaikki operandit ja tulokset aritmeettisista operaatioista täytyy esittää kokonaislukuina. Moduuleittensa ansiosta summaus, vähennys ja kertolasku ovat muistinumeroista vapaita. Jokainen moduuli on ns. itsenäinen eikä sen arvo siis riipu muista moduuleista. Tällöin myös laskutoimitukset ovat itsenäisiä ja muistinumeroita ei tarvita. Tähän ominaisuuteen suurimmaksi osaksi RNS-tekniikka perustuukin ja sen takia sitä on kannattavaa käyttää erilaisissa sovelluksissa.

Perustuen edelliseen, RNS-tekniikkaa käytetään monissa digitaalisien signaalinkäsittelyn tehtävissä, kuten digitaalisissa suotimissa, konvoluutioissa, korrelaatioissa, DFT:ssä ja FFT-laskennassa. RNS-järjestelmästä on myös monia sovelluksia ja jatko kehityksiä kuten, Quadratic Residue Number System (QRNS), Quadratic Like Residue Number System (QLRNS), Modified Quadratic Residue

Number System (MQRNS) ja Polynomial Residue Number System (PRNS). Kaikilla näillä on merkityksensä digitaalisessa signaalinkäsittelyssä, vähentäen laskutoimituksien kompleksisuutta ja samalla suurentaen toimintojen rinnakkaisuuksia. [3]

3.2. RNS-aritmetiikka

RNS-järjestelmä perustuu täysin sen kantaan. Kuitenkin järjestelmä ei koostu vain yhdestä kannasta vaan useammasta kokonaisluvusta, moduuleista $\{m_1, m_2, \ldots, m_n\}$. Nämä moduulit luovat dynaamisen alueen M, missä $M = \prod m_i$. [1]

3.2.1. RNS-määritelmä ja -esitys

Jokaisella kokonaisluvulla x, joka kuuluu dynaamiselle alueelle M, on olemassa esitysmuoto RNS-järjestelmässä $\{r_1, r_2, \ldots, r_n\}$, *jossa* $r_i = |x|_{m_i}$ *ja* $|\cdot|$ *tarkoittaa* m_i*-kantaista moduloa luvusta x.* [1]

Eli, $r_i = x \bmod m_i$ (12)

Esimerkki 1.

Esitetään luku 21 RNS-kannassa {8, 7, 5, 3}. Jolloin dynaaminen alue, M = 840

$x = 21 = \{|21|_8, |21|_7, |21|_5, |21|_3\} = \{5, 0, 1, 0\}$

RNS-järjestelmässä jokaisella numerolla, joka kuuluu dynaamiseen alueeseen, on yhdenlainen esitys, sillä jokaisella numerolla on olemassa vain yksi ei-negatiivinen jakojäännös. Kuitenkin edellisen esimerkin kannassa myös luvulla 861 on sama esitys RNS-kannassa. Huomataan, että RNS-järjestelmä on jaksollinen, joka toistuu dynaamisen alueen välein. Dynaaminen alue tekee RNS-järjestelmästä yksiselitteisen, kun otetaan vain yksi jakso huomioon. [1]

Moduulien koosta johtuen itse muunneltavien lukujen koot muuttuvat merkittävästi pienemmiksi, jolloin selvitään vähemmällä määrällä bittejä. Jotta RNS-järjestelmässä pystytään esittämään yhtä monta numeroa kuin kahden komplementissa, täytyy RNS-järjestelmän dynaaminen alue vastata vastaavan kahdenkomplementti-järjestelmän aluetta. RNS-järjestelmän dynaamista aluetta voidaan kasvattaa joko suurentamalla moduulien kantaa tai lisäämällä moduulien määrää.

3.2.2. Negatiivinen RNS

RNS-järjestelmä perustuu siis valittujen moduulien jakojäännöksiin ($x \bmod m_i$). Tästä huomataan, että luku x voi olla mikä tahansa kokonaisluku, myös negatiivinen. Mutta jakojäännöksen määritelmän perusteella $\left|x\right|_{m_i}$ täytyy olla positiivinen, tällöin

$$r_i = \left|x\right|_{m_i}, \qquad \text{jos } x > 0 \tag{13}$$

$$r_i = \left|M - x\right|_{m_i}, \qquad \text{jos } x < 0. \tag{14}$$

Dynaaminen alue muuttuu, jos otetaan huomioon myös negatiiviset luvut:

$$M = [0 \qquad M - 1], \qquad \text{jos } x > 0 \tag{15}$$

$$M = [-M/2 \qquad -M/2 - 1], \qquad \text{jos } x < 0 \text{ ja } x \text{ on parillinen} \tag{16}$$

$$M = [-(M-1)/2 \quad (M-1)/2], \qquad \text{jos } x < 0 \text{ ja } x \text{ on pariton.} \tag{17}$$

3.2.3. Multiplikatiivinen käänteisluku

Jos luku a, $0 \leq a < m$ ja $|ab|_m = 1$, niin lukua a kutsutaan luvun b mod m multiplikatiiviseksi käänteisluvuksi ja merkitään $a = \left|\frac{1}{b}\right|_m$.

Luku $\left|\frac{1}{b}\right|_m$ on olemassa, jos ja vain jos $(b,m) = 1$ ja $b \neq 0$. Tässä tapauksessa $\left|\frac{1}{b}\right|_m$ on yksiselitteinen. [1]

Yleisesti ottaen ei ole olemassa yksiselitteistä selkeää ilmaisua ilmoittaa multiplikatiivinen käänteisluku, lyhyemmin käänteisluku. Kuitenkin, jos moduuli *m* on alkuluku, niin *Fermatin teoreemalla* voidaan käänteisluku laskea helposti. [1]

Fermatin teoreema

Jos luku p on alkuluku, niin $|a^p|_p = |a|_p$ kaikilla kokonaisluvuilla a.

Fermatin teoreema on tärkeä, koska se selvästi esittää luvun $|a|_p$ käänteisluvun, jos luku *p* on alkuluku ja $a \neq 0$. Fermatin teorian avulla luvun $|a|_p$ käänteisluku on muotoa $|a^{p-2}|_p$, sillä $|a^{p-2}a|_p = 1$. [1]

Siis toisin sanoen $|a^{-1}|_p = |a^{p-2}|_p$ (18)

Esimerkki 2.

$$\left|\frac{1}{3}\right|_7 = \left|3^{7-2}\right|_7 = \left|3^5\right|_7 = \left|243\right|_7 = 5$$

3.2.4. Käännös RNS-järjestelmästä kymmenjärjestelmään.

On olemassa kaksi yleistä tekniikkaa muuntaa RNS-numero takaisin kymmenjärjestelmään. Yleisin on kiinalainen jäännöslause (Chinese Reminder Theorem, CRT). Toinen on mixed-radix-käännös (Mixed Radix Conversion, MRC), jossa RNS-luku muutetaan ensin mixed-radix-järjestelmään (Mixed Radix System, MRS).

Annetun luvun x RNS esityksestä $\{r_1, r_2, \ldots, r_N\}$ kiinalainen jäännöslause mahdollistaa määrittämään takaisin luvun $|x|_M$, kun moduulien suurin yhteinen tekijä on 1. Tällaisia moduuleja kutsutaan parillisesti suhteellisiksi alkuluvuiksi.

Kiinalainen jäännöslause (CRT)

$$|x|_M = \left| \sum_{j=1}^{N} \hat{m}_j \left| \frac{r_j}{\hat{m}_j} \right|_{m_j} \right|_M \qquad (19)$$

missä $\hat{m}_j = \frac{M}{m_j}$, $M = \prod_{j=1}^{N} m_j$ ja $(m_j, m_k) = 1$ kun $j \neq k$

CRT vaatii, että käänteisluku moduuleista on olemassa, eli että moduulit ovat keskenään pareittain suhteellisesti alkulukuja. Tästä johtuen RNS-järjestelmä on varsin rajattu, kun moduuleja ei voi mielivaltaisesti valita, vaan ne täytyy harkiten laskea ja sovittaa jokaiseen tilanteeseen erikseen.

3.2.5. Käännös suoraan binääriluvusta

Käännös RNS-luvuksi binääriluvusta voidaan tehdä helposti suoraan, jolloin ei tarvitse binäärilukua muuntaa ensin kymmenjärjestelmään. Käännös binääriluvusta suoraan RNS-järjestelmään hoituu helposti, kun muunnetaan ns. yksi bitti kerrallaan.

Kun annettu bitti on muotoa $x = 2^n b_n + \ldots + 2^2 b_2 + 2b_1 + b_0$, niin sen RNS-käännös on muotoa $|x|_{mi} = ||2^n|_{mi}\, b_n + \ldots + |2^2|m_i b_2 + |2|m_i b_1 + b_0|_{mi}$. [1]

3.3. Aritmetiikka

Aritmetiikka RNS-järjestelmässä perustuu kiinalaiseen jäännöslauseeseen. Moduulit siis oletetaan tässä olevan keskenään parillisesti suhteellisiksi alkuluvuiksi.

Summa, vähennys ja kertolasku voidaan suorittaa itsenäisesti jokaiselle luvulle erikseen. Kun moduulit ovat itsenäisiä ja jokaisella on oma painonsa järjestelmässä, myös nämä perusoperaatiot voidaan suorittaa muista moduuleista välittämättä.

Yhteen- ja vähennyslasku

Olkoon x ja y jäännöslukuja ja numerojärjestelmä koostuu moduuleista m_1, m_2, … , m_N. Tällöin jäännösesitys summalle ja erotukselle $|x \pm y|_M$ on,

$$|x \pm y|_M \leftrightarrow \left\{ \left| |x|_{m_1} \pm |y|_{m_1} \right|_{m_1}, \left| |x|_{m_2} \pm |y|_{m_2} \right|_{m_2}, \ldots, \left| |x|_{m_N} \pm |y|_{m_N} \right|_{m_N} \right\} \qquad (20)$$

Yksinkertaisuudessaan summa (tai vähennys) tuo RNS-järjestelmälle sen tärkeimmän ominaisuuden ja edun esille. Operaatioissa ei synny moduulin sisäisiä muistinumerobittejä vaan jokainen tuloksen jakojäännös on riippuvainen vain operandin vastaavasta jakojäännöksestä. Tämä suuri ero painollisiin numerojärjestelmiin antaa RNS-järjestelmälle merkittävän edun.

Muistinumerobitin poissaolo antaa järjestelmälle luontaisen edun nopeudessa. Järjestelmään ei tarvitse erikseen toteuttaa mitään ylimääräistä sovellusta järjestämällä ja odottamalla muistinumerobittejä. Täten vältytään ylimääräisiltä viiveiltä. Moduulien itsenäisyys antaa toisen merkittävän edun. Jokaisen moduulin operandi voidaan laskea samanaikaisesti. Rinnakkaisuudella saavutetaan lisähyötyä nopeudessa. [1]

Kertolasku

Tulo voidaan aina toteuttaa toistamalla summia, jolloin tulo on analoginen logiikaltaan kuin summa.

Olkoon x ja y jäännöslukuja ja numerojärjestelmä koostuu moduuleista m_1, m_2, …, m_N. Tällöin jäännösesitys tulolle $|x \cdot y|_M$ on,

$$|xy|_M \leftrightarrow \left\{ \left| |x|_{m_1} |y|_{m_1} \right|_{m_1}, \left| |x|_{m_2} |y|_{m_2} \right|_{m_2}, \ldots, \left| |x|_{m_N} |y|_{m_N} \right|_{m_N} \right\} \qquad (21)$$

Selvästi huomataan, että muistinumerobitin poissaolo myös tulo-operandissa. Nämä perusoperaatiot antavat suuren hyödyn rinnakkaisuudessa, kun jokainen moduuli voidaan laskea samanaikaisesti välttyen muistinumerobitiltä moduulien välillä. Moduulien itsenäisyys mahdollistaa, jos prosessin aikana tapahtuu virhe yhdessä moduulissa, niin virhe ei laajene ja kertaannu muihin moduuleihin. RNS-järjestelmä pystyy edelleen toimimaan ainoastaan eristämällä virheellinen moduuli. Tämä ominaisuus antaa korkean potentiaalin virheen kestossa. [3]

Jakolasku on RNS-järjestelmässä ongelmallista. Kun RNS ei ole samalla tavalla painollinen systeemi kuin kymmenjärjestelmä tai binäärijärjestelmä, niin RNS-lukua ei voi suoraan vain jakaa toisella RNS-luvulla. RNS-järjestelmässä kaikki luvut ovat kokonaislukuja. Sen myötä jakolasku on vaikea, kun suurimmassa osassa jako-operaatioissa tuloksena on muuta kuin kokonaisluku.

3.4. RNS-menetelmän edut ja ongelmat

Yleisesti ottaen RNS-järjestelmää käytetään todella vähän ja sen käyttö onkin minimaalista johtuen sen aiheuttamasta suuresta määrästä rajoituksia ja hankaluuksista toteutuksessa.

Rinnakkaisuuden ja moduulien suuruuksien pienuuden ansiosta perusoperaatiot pystytään tekemään pienillä look-up taulukoilla tai jopa kombinatorisella logiikalla järkevällä kompleksisuudella. Kun moduulit ovat pienikokoisia ja vievät vain muutaman bitin, niin myös näiden operaatiot ovat nopeita ja yksinkertaisia. Rinnakkaisuus sekä muistibitin huomioimattomuus tuovat entisestään lisää nopeutta järjestelmään.

Suurimpia miinuspuolia RNS-järjestelmässä on, että se ei lukua päältäpäin nopeasti katsoessa kerro itsestään paljon mitään. RNS-luvusta ei päältä päin voi nähdä sen etumerkkiä, eikä voida tietää onko joku toinen RNS-luku suurempi kuin toinen. Eli, lukujen vertailu on todella hankalaa.

Myös ylivuodon tarkkailu on vaikeaa. Tästä johtuen täytyy erikseen laskea ja varata riittävän suuri määrä bittejä järjestelmään, jotta ylivuotoa ei synny. Käytännössä jakolaskun laskemisen mahdottomuus tuo myös lisää rajoituksia ja vaikeuttaa laskuoperaatioita järjestelmässä.

Suuret vaikeudet RNS-järjestelmässä rajoittavat suuresti sen käyttöä. Sen takia RNS-tekniikkaa käytetään vain sovelluksissa joissa summaus ja kertominen ovat suuressa osassa. Samoin tuloksen täytyy olla tiedossa tietyn rajan sisällä, jotta vältytään ylivuodolta.

3.5. Kompleksinen RNS-järjestelmä

RNS-menetelmää voi käyttää myös kompleksilukujen laskuoperaatioissa. Kun RNS menetelmään lisätään imaginääriosa, niin puhutaan kompleksisesta RNS-järjestelmästä (Complex Residue Number System, CRNS). Kompleksinen luku täytyy vain saavuttaa samat ehdot niin reaaliosalta kuin imaginääriosalta. Eli kompleksiluvun täytyy olla kokonaisluku molemmilta osin. Käännös kompleksiselle luvulle tehdään reaaliosalle ja imaginääriosalle erikseen.

$Z = X + jY$, missä $j = sqrt(-1)$ sekä X ja Y *modulo* m RNS-kannassa (22)

Käännös takaisin tehdään myös erikseen reaaliosalle ja imaginääriosalle CRT-menetelmällä. Laskuoperaatiot summan ja tulon suhteen menevät normaaliin tapaan ja operaatiot suoritetaan eri osille erikseen.

Summa ja tulo

$$Z_3 = Z_1 + Z_2 = X_3 + jY_3 = |X_1 + X_2|_m + j|Y_1 + Y_2|_m \quad (23)$$

$$Z_3 = Z_1 \cdot Z_2 = X_3 + jY_3 = |X_1X_2 - Y_1Y_2|_m + j|X_1Y_2 + X_2Y_1|_m \quad (24)$$

Reaaliosa ja imaginääriosa ovat tässä tapauksessa myös itsenäisiä eivätkä ne riipu toisistaan. Moduulien väliset operaatiot siis voidaan suorittaa kaikki yhtä aikaa ja näin rinnakkaisuudesta hyödytään nopeuden kasvuna.

Kompleksisesta summasta ja tulosta huomataan, että summa vaatii kaksi yhteenlaskuoperaatiota ja tulo vaatii neljä kertolaskuoperaatiota sekä kaksi yhteenlaskuoperaatiota. Tämä on siis sama määrä laskuoperaatioita kuin kahden komplementissa. Ainoa etu siis tässä tapauksessa saadaan rinnakkaisuudesta.

3.6. Quadratic RNS-menetelmä

Vuonna 1981 *S. H. Leung* kehitti Quadaratic RNS-menetelmän CRNS-menetelmän jatkeeksi. QRNS on muunnelma RNS-muunnoksesta tai lähinnä jatkokehitys. [4]

QRNS-menetelmällä on samat ehdot ja määritelmät kuin RNS-menetelmällä. QRNS-menetelmä on todella tehokas kompleksisessa laskennassa ja siksi se on varteenotettava vaihtoehto digitaalisessa signaalinkäsittelyssä.

QRNS-menetelmä hyöty perustuu siihen, että sen avulla voidaan kompleksisen kertolaskun operaatioiden määrä puolittaa. QRNS mahdollistaa siis laskea kompleksinen tulo ainoastaan kahdella kertolaskulla. Tämän kautta QRNS nopeuttaa laskentaa ja pienentää olennaisesti kompleksisuuden määrää. QRNS mahdollistaa huomattavat resurssien säästöt verrattuna kahdenkomplementtiin. Sen takia se on todella hyödyllinen esim. FFT-laskennassa, jossa on paljon kompleksisia kertolaskuja. [5] [6]

3.6.1. QRNS-käännös ja -aritmetiikka

QRNS tuo paljon lisää rajoituksia koko RNS-menetelmään. QRNS-menetelmä perustuu siihen, että imaginääri yksikkö *j* kuvataan RNS-luvulla. Ts.

$$r_i^2 = -1 \ mod \ m_i \quad (25)$$

on olemassa ja sen esitys pystytään laskemaan.

QRNS-käännös on olemassa ainoastaan, jos moduulit m ovat alkulukuja ja ovat muotoa $m_i = 4k_i + 1$, jossa luku k on positiivinen kokonaisluku. Tai vaihtoehtoisesti, jos yhtälö $x^2 + 1$ voidaan jakaa tekijöihin $(x - r_i)(x + r_i)$, jossa $r_i \in \{0, 1, \ldots, m_i - 1\}$.

Kaavan (14) luvun r juuret ovat reaalisia kokonaislukuja. *Leung* esitti yhteyden CRNS ja QRNS menetelmien välillä. Hän esitti, että QRNS luku pari *(Z, Z*)* ja CRNS luku $X+jY$ voidaan muuntaa seuraavilla kaavoilla: [7]

$$Z = |X + rY|_m \tag{26}$$

$$Z^* = |X - rY|_m \tag{27}$$

$$X = |2^{-1} (Z + Z^*)|_m \tag{28}$$

$$Y = |(2r)^{-1} (Z - Z^*)|_m. \tag{29}$$

Kaavoissa (15)-(18) luvut 2^{-1} ja r^{-1} ovat multiplikatiiviset käänteisluvut modulossa m ja luvut X ja Y ovat RNS-lukuja.

Olkoon $z_1 = (Z_1, Z_1^*)$ ja $z_2 = (Z_2, Z_2^*)$, *jolloin summa, erotus ja tulo voidaan esittää QRNS-järjestelmässä seuraavasti:*

$$z_1 \,¤\, z_2 = (Z_1, Z_1^*) \,¤\, (Z_2, Z_2^*) = (|Z_1 \,¤\, Z_2|_m, |Z_1^* \,¤\, Z_2^*|_m), \tag{30}$$

jossa merkki ¤ kuvaa summaa, erotusta tai tuloa.

Kaavasta (19) huomataan, että tulo vaatii ainoastaan kaksi kertolaskua, eikä summaa ollenkaan. QRNS-menetelmällä on näin saatu poistettua ns. ristitermien summaukset. Tämän takia QRNS-menetelmä on todella hyödyllinen ja antaa edun nopeudessa verrattuna CRNS-menetelmään. Lisäksi kun moduulit ovat itsenäisiä, ne voidaan ajaa yhtä aikaa ja näin rinnakkaisuudella saadaan kaikki hyöty irti. [7]

3.6.2. QRNS-menetelmän ongelmat

Vaikka QRNS-menetelmänä on nopea vähentyneen kompleksisuuden ja monimutkaisen laskennan kautta, sekä rinnakkaisuuden ja vähentyneiden kertolaskuoperaatioiden myötä, niin se on todella rajoittunut järjestelmä.

Moduulien tarkemmat rajoitukset (oltava alkuluku ja muotoa $4k + 1$) vaikeuttavat suunnittelua ja tekevät menetelmästä harvinaisen. Myös moduulien skaalaus on huomattavasti hankalampaa, ellei miltei mahdotonta. [7]

Systeemin suunnitteluun täytyy lisätä konversiot, jotka syövät vähän aikaa ja tilaa, vaikka ei merkitsevästi. Yleensäkin muunnokset eri menetelmien välillä vievät tehoa ja aikaa varsinaisista laskuoperaatioista. Toisaalta ainoastaan käännös RNS-luvusta

binäärilukuun CRT-menetelmällä on todella hidas ja kallis pinta-alaltaan, tehon ja kustannuksien suhteen. Se yleensä onkin järjestelmän pullonkaula ja vie järjestelmästä eniten laskenta-aikaa. Muut konversiot hoituvat helposti ja nopeasti eikä niinkään ole järjestelmälle haitallinen. [6]

RNS-menetelmää kannattaa siis käyttää ainoastaan, jos muunnoksien välillä tehdään paljon summaus- ja kertolaskuoperaatioita, jotta menetelmä olisi järkevää käyttää. RNS-menetelmä on sitä tehokkaampi mitä enemmän laskuoperaatioita on muunnoksien välissä.

3.7. Mixed-radix-järjestelmä

Mixed-radix-järjestelmä on vastaavanlainen järjestelmä kuin RNS. Siinä mikä tahansa positiivinen kokonaisluku voidaan esittää seuraavanlaisesti:

$$x = a_N \prod_{i=1}^{N-1} R_i + ... + a_3 R_1 R_2 + a_2 R_1 + a_1 , \qquad (31)$$

jossa R_i esittää kantalukuja ja a_i mixed-radix-lukuja ja $a_i = [0, R_i)$. Luvun x mixed-radix-esitys on muotoa $<a_n, a_{n-1}, ..., a_1>$ ja lukualue $[0, \Pi R_i - 1]$.
Edellisen määritelmän mukaan nähdään, että jokaisella numerolla on yksi yksikäsitteinen esitys. Kymmenjärjestelmä on erikoismuoto MRS-järjestelmässä, missä jokainen $R_i = 10$ ja numeroiden painot ovat kymmenen potensseja.

MRS-järjestelmä on hyvä ja tärkeä lisäapu RNS-tekniikassa ja -laskennassa. MRS on painollinen järjestelmä, joten sen avulla pystytään vertailemaan eri lukuja keskenään ja myös päättelemään lukujen etumerkki. Käännökset RNS-järjestelmästä MRS-järjestelmään ovat nopeita ja tehokkaita. MRS-menetelmässä helpotusta tuo se ettei lukujen vertailuissa tarvitse kääntää lukuja erikseen 10-järjestelmään, joka on paljon raskaampaa ja hitaampaa. [1]

3.7.1. RNS-käännös mixed-radix-järjestelmään

RNS-luvun kääntäminen MRS-järjestelmään ovat nopeita tietyissä tilanteissa ja on täten järkevämpää muuntaa RNS-luku MRS-järjestelmään 10-kantajärjestelmän sijaan. Varsinkin, jos on vain tarkoitus tehdä lukujen suuruus vertailuja tai tarkistaa luvun etumerkki. [1]

Jos moduulit m_1, m_2, …, m_N on valittu siten, että $m_i = R_i$, niin mixed-radix ja residue-numerojärjestelmän sanotaan assosioivan toisiaan. Tässä tilanteessa molemmilla systeemeillä on sama lukualue, joka siis on $\prod m_i$.

Jos $m_i = R_i$, niin mixed-radix-esitys on muotoa, niin

$$x = a_N \prod_{i=1}^{N-1} m_i + ... + a_3 m_1 m_2 + a_2 m_1 + a_1 , \qquad (32)$$

jossa a_i ovat mixed-radix-kantalukuja. Luvut a_i määritellään järjestelmällisesti seuraavalla tavalla aloittaen luvusta a_1.

Yhtälöstä (32) nähdään, että kaikki kantaluvut paitsi viimeinen luku a_1 on jaollinen luvulla m_1, niin täten saamme helposti $\left|x\right|_{m_1} = a_1$. Näin ollen ensimmäinen mixed-radix luku on a_1.

Saadakseen selville seuraava mixed-radix-luku a_2, täytyy ensin laskea $x - a_1$. Kaavan (32) mukaan luku $x - a_1$ on jaollinen luvulla m_1. Määritelmän mukaan täten luku m_1 on suhteellinen alkuluku muille moduuleille ja täten voidaan suorittaa laskutoimitus $\frac{x-a_1}{m_1}$. Tarkastelemalla kaavaa (32) huomataan, että $a_2 = \left|\frac{x-a_1}{m_1}\right|_{m_1}$. Täten jatkamalla tätä samaa tapaa, voidaan selvittää aina seuraava mixed-radix-luku. Erityisesti huomataan, että $a_1 = \left|x\right|_{m_1}$, $a_2 = \left|\left[\frac{x}{m_1}\right]\right|_{m_2}$, $a_3 = \left|\left[\frac{x}{m_1 m_2}\right]\right|_{m_3}$ ja yleisesti

$a_i = \left|\left[\frac{x}{m_1 m_2 \cdots m_{i-1}}\right]\right|_{m_i}$, $i > 1$. [1]

Mixed-radix-käännös vaatii $2(N - 1)$ aritmeettista operaatiota jakautuen tasan vähennyslaskuihin ja kertolaskuihin (kun N on sama kuin moduulien määrä). On myös olemassa kaksi nopeampaa menetelmää, mutta ne ovat erityistapauksia ja niitä voi käyttää vain kun moduulit ovat tiukasti rajatut. [1]

Ensimmäinen menetelmä on, missä moduulit ovat valittu siten, että moduulien kaikki käänteisluvut mixed-radix-käännöksessä ovat yksiselitteiset. Tällöin kertolaskut käännöksessä voidaan jättää huomiotta. Tällainen järjestelmä on esimerkiksi $m_1 = 2$, $m_2 = 3$, $m_3 = 7$ ja $m_4 = 43$. [1]

Toinen erityistapaus on, missä tietyillä numero kombinaatiolla käännösvaiheessa loput moduulit saadaan heti suoraan selville ilman lisälaskutoimituksia. Tässä on kaksi sääntöä. Ensimmäinen sääntö on, että jos määritellessä moduloa a_i ja huomataan, että loput residue-luvut ovat $\left|a_i\right|_{m_{j1}}$, jossa m_j on mikä tahansa jäljellä oleva modulus, niin jäljellä olevat mixed-radix-moduulit ovat nollia. Toinen sääntö on, että jos määritellessä moduloa a_i ja huomataan, että loput residue-luvut ovat $\left|m_j - a_i\right|_{m_{j1}}$, jossa m_j on mikä tahansa jäljellä oleva modulus, niin jäljellä olevat mixed-radix-moduulit ovat muotoa $a_i = m_j - 1$. Tätä menetelmää käyttäessä operaatiot muunnoksessa vähenevät merkittävästi, jos moduulit ovat pieniä. [1]

3.8. RNS-moduulien valinta

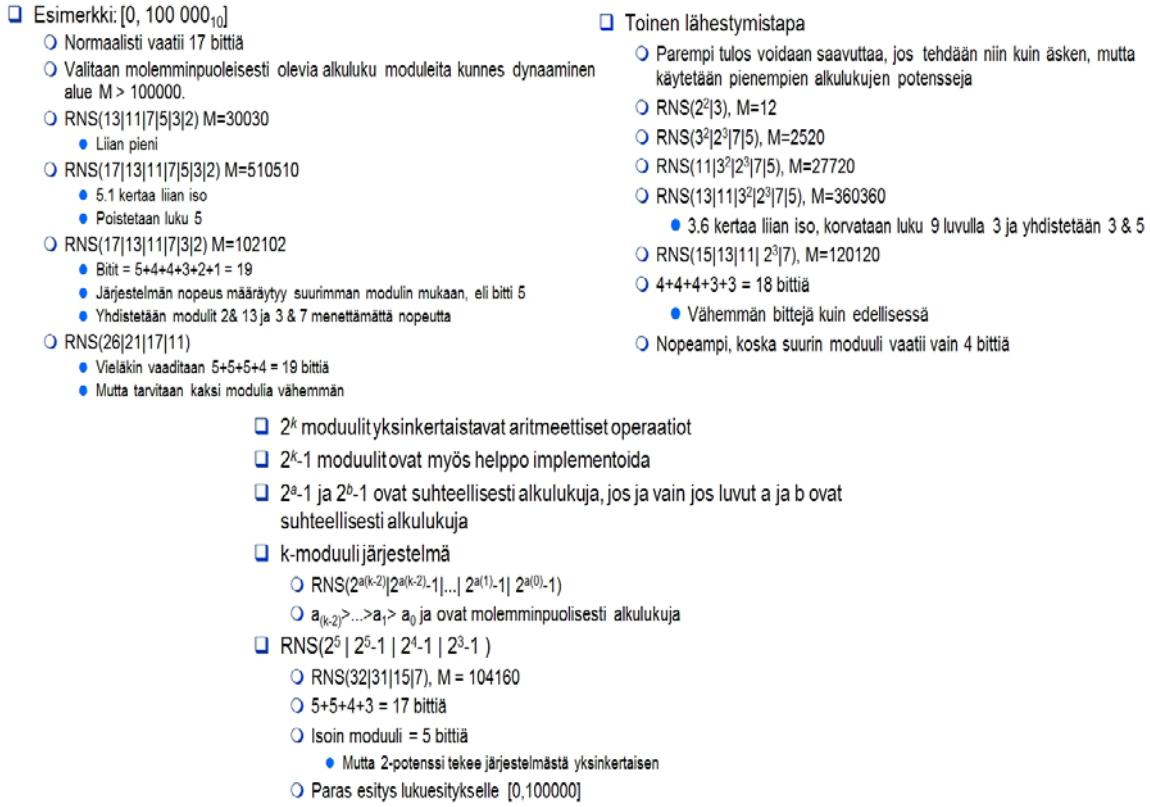
RNS-moduulien valinta vaikuttaa suoraan menetelmän tehokkuuteen ja nopeuteen. Siksi on tärkeää myös valita oikeat moduulit sen mukaan mihin tarkoitukseen kyseistä järjestelmää käytetään. Moduulien valintaan vaikuttaa esitystehokkuus sekä algoritmin

monimutkaisuus. Esitystehokkuus lasketaan kuinka suuri dynaaminen alue saadaan moduuleilla sama määrä bittejä käyttäen kuin binääriluvuissa.

Moduulien valinta on tärkeässä roolissa suunniteltaessa RNS-järjestelmää. Tietyt erikoiset sovellukset vaativat erilaisia toteutuksia, joka vaikuttaa myös moduulien valintaan. Myös järjestelmän toteutus vaikuttaa siihen millaisia moduuleita pystyy käyttämään.

Pääsääntö moduulia valittaessa täytyy muistaa, että suurin moduuli määrittää nopeuden operaatioissa sekä myös käännöksissä järjestelmien välillä. Kun suurin moduuli on valittu, on järkevä valita muut moduulit suurimman mukaan ja näin saada suurempi dynaaminen alue aikaiseksi. Myös tietyt moduulit on helpompi toteuttaa fyysisesti kuin toiset.

Ensimmäinen ja helpoin menetelmä määrittää moduulit on valita alkulukuja moduuleiksi kunnes dynaaminen alue saadaan katettua. Tästä eteenpäin muokkaamalla ja yhdistelemällä moduuleita voidaan moduuleitten määrää pienentää. Pienempien alkulukujen potenssit muiden alkulukujen kanssa moduuleina on myös hyvä valinta. Kahden potenssiin nojautuvat moduulit ovat tehokkaita niin moduulien määrän suhteen kuin myös niiden toteutuksen helppouden suhteen. Kuva 9. kuvastaa yksinkertaista esimerkkiä miten moduulit voidaan valita, niin että aikaansaadaan haluttu dynaaminen alue.



Kuva 9. Esimerkki moduulien valinnasta RNS-järjestelmään.

RNS-järjestelmän moduulien valinta voidaan kategorisoida edellisen kuvan mukaisesti. Suosituin menetelmä kuitenkin on kahden potenssin moduulit. 2^n-moduulien tehokkuus toteutuksissa on ylivoimainen, koska niiden yhteydessä voidaan käyttää hyödyksi samoja menetelmiä kuin mitä käytettäessä pelkkiä binäärilukuja. Myös yksinkertaisuus aritmeettisissa operaatioissa tuo suuren edun muihin moduulivalintoihin verrattuna. [13]

Kaikista suosituin ja käytetyin moduulisarja on $\{2^n - 1, 2^n, 2^n + 1\}$, joka on esitetty jo vuonna 1977. Tämän moduulisarjan edut ovat yksinkertaisuudessa ja hyvin muodostetuissa ja balansoiduissa moduuleissa. Kuitenkin aritmeettiset operaatiot ($2^n + 1$)-moduulilla ovat kompleksisia verrattuna kahteen muuhun moduuliin, joten tämän takia on kehitelty muutamia korvaavia moduuli-sarjoja, jotka ovat myös hyvin suosittuja. Näitä ovat mm. moduulisarjat $\{2^{n-1} - 1, 2^n - 1, 2^n\}$ ja $\{2^n - 1, 2^n, 2^{n+1} - 1\}$. Näissä moduuli $2^n + 1$ on korvattu joko $2^{n-1} - 1$ tai $2^{n+1} - 1$-moduulilla ja näin saatu järjestelmän nopeutta kasvatettua. [13]

Kolmen moduulin järjestelmässä tulee nopeasti dynaaminen alue vastaan. Tämän takia on kehitelty useita vastaavia neljän moduulin ja jopa viiden 2^n-moduulin järjestelmiä. Näissä kuitenkin kompleksisuus kuitenkin kasvaa huomattavasti. Näistä seikoista huomaakin miten tärkeä rooli moduulien valinnalla on. Valinta on usein kompromissi ja mitä osa-alueita siinä halutaan painottaa. [13]

On myös kehitelty moduulisarja $\{2^\alpha, 2^\beta - 1, 2^\beta + 1\}$, jossa $\alpha < \beta$. Tässä muuttujilla α ja β säädetään dynaamista aluetta halutunlaiseksi, mutta samalla saaden hyvin balansoitu moduulijärjestelmä. Järjestelmä tarjoaa myös korkean suorituskyvyn RNS-järjestelmässä ja erityisesti RNS-takaisinkäännöksessä. [13]

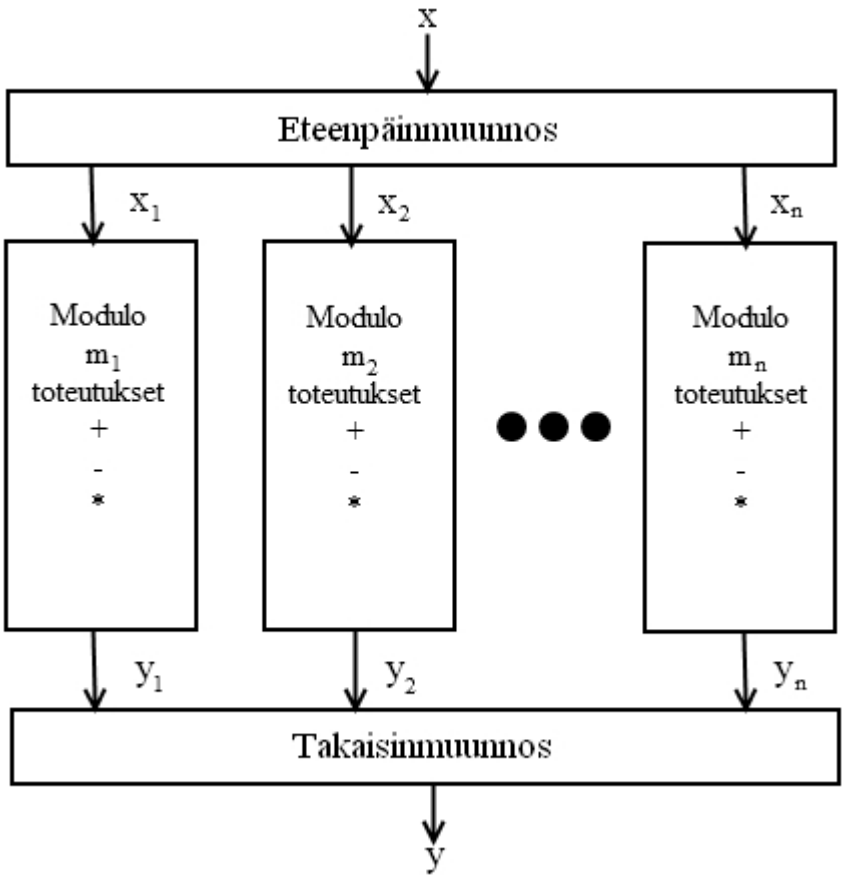
4. RNS-TEKNIIKAN SOVELTAMINEN

Tässä kappaleessa perehdytään muutamiin RNS-tekniikoihin ja -sovelluksiin, joissa konkreettisesti näemme järjestelmän tuomat edut. RNS-järjestelmää on toteutettu eri tekniikoilla ja tavoilla vuosien saatossa. RNS-tekniikkaa on käytetty hyödyksi jo muutamia vuosikymmeniä, mutta lopullista läpimurtoa ei ole vielä saavutettu ainakaan suuremmassa mittakaavassa järjestelmän monimutkaisuuden ja rajoituksien takia. Monimutkaisuus ja suuret rajoitukset ovat hidastaneet myös järjestelmän tunnettavuutta maailmalla ja leviämistä tutkijoiden sekä suunnittelijoiden tietoisuuteen.

Kuvassa 10. on esitetty lohkokaavio tyypillisestä RNS-järjestelmästä. Kymmenkanta-luku muunnetaan ensin RNS-kantaan, joissa kaikki moduulioperaatiot voidaan suorittaa rinnakkaisesti. Laskutoimituksien jälkeen RNS-luku käännetään takaisin kymmenkantaan.

Takaisinkäännös RNS-järjestelmästä binääriluvuksi on koko prosessin vaikein osa ja myös vaikuttaa suuresti koko järjestelmän tehokkuuteen. Käännös vie suuren osan ajasta koko prosessista ja sen vuoksi tutkimuksia on tehty juuri tälle osa-alueelle kaikkein eniten. Käänteismuunnokseen vaikuttaa suurimmalta osin moduulien valinnat sekä menetelmät joita on käytetty RNS-muunnoksessa. [13]

Myös kokonaisvaltainen moduulien valinta vaikuttaa toteutukseen ja mitä useampi moduuli on valittu sitä enemmän järjestelmä vie piiriltä tilaa. Siksi RNS-järjestelmissä käytetään suurimmaksi osaksi kolmen tai neljän moduulin menetelmiä. [13]



Kuva 10. Kuvaus tyypillisestä RNS-järjestelmästä.[13]

4.1. Eteenpäin käännös

Eteenpäin käännös (Binary-to-residue, RB) on tässä tapauksessa joko binääri- tai desimaaliluvun käännös RNS muotoon. Perusperiaate jakojäännöksien laskemisessa on jakolasku, jossa jakajana on moduuli. Kuitenkin rautatasolla jakojäännös on kallis operaatio joten sen käyttöä on syytä välttää. Jakolaskun rautatason toteutusta voidaan helpottaa tietynlaisilla moduuleilla, mutta tämä taas rajoittaa muuten koko järjestelmän tehokkuutta sillä tietyillä moduuleilla taasen saavutetaan parempia etuja esim. takaisinmuunnoksessa. [20]

Rautatason toteutukset perustuvat yleisesti look-up-taulukoihin, kombinatorilogiikkaan tai näiden yhdistelmiin. Yleisesti erilaiset sovellukset, jotka perustuvat tiettyihin erityisiin moduuleihin, ovat suunniteltu kombinatorilogiikalla. Näiden muunnoksien kompleksisuus riippuu täysin valituista moduuleista. [20]

Digitaalinen signaalikäsittely vaatii yleisesti suurta dynaamista aluetta, joten tämän tyyppiset sovellukset joko koostuvat useasta pienestä moduulista tai muutamasta isosta moduulista. Yleensä valinta osuu jälkimmäiseen. [20]

4.1.1. BR-käännös erikoismoduulille $\{2^n - 1, 2^n, 2^n + 1\}$

Suositulle moduulisarjalle $\{2^n - 1, 2^n, 2^n + 1\}$ myös BR-käännös on suoraviivainen ja koko käännös pystytään toteuttamaan pelkästään kombinatorilogiikalla ja moduulisummaimia käyttäen. Tässä tapauksessa siis hyötyä saadaan jo eteenpäin muunnoksessa, vaikka koko moduulisarja on suunniteltu ja kehitetty takaisinkäännöstä ajatellen. [20]

Jos määritellään, $m_1 = 2^n + 1$, $m_2 = 2^n$ ja $m_3 = 2^n - 1$, niin jokaiselle luvulle X, jonka dynaaminen alue on $M = [0, 2^{3n} - 2^n - 1]$, on ainutlaatuisesti määritelty residue-luku $\{r_1, r_2, r_3\}$, jossa $r_i = |X|_{m_i}$ ja X on $3n$ bittinen luku $X = x_{3n-1}x_{3n-2}x\ldots x_{2n}x_{2n-1}\ldots x_n x_{n-1}\ldots x_0$. [20]

Jakojäännökset r_i saadaan jakamalla luku X moduulilla m_i. Täten jakojäännös r_2 on helpoiten laskettavissa, sillä n vähiten merkitsevää bittiä määrittää jakojäännöksen kun X jaetaan luvulla 2^n. Nämä bitit saadaan helposti siirtämällä oikealle n bittiä. Saadakseen selville jakojäännökset r_1 ja r_3 jaetaan luku X kolmeen n-bittiseen lohkoon seuraavasti: $B_1 = \sum_{j=2n}^{3n-1} x_j 2^{j-2n}$, $B_2 = \sum_{j=n}^{2n-1} x_j 2^{j-n}$ ja $B_3 = \sum_{j=0}^{n-1} x_j 2^j$. [17]

Täten $X = B_1 2^{2n} + B_2 2^n + B_3$.

r_1 jakojäännös saadaan nyt laskettua seuraavasti:

$$\begin{aligned} r_1 &= |X|_{2^n+1} \\ &= \left|B_1 2^{2n} + B_2 2^n + B_3\right|_{2^n+1} \\ &= \left|\left|B_1 2^{2n}\right|_{2^n+1} + \left|B_2 2^n\right|_{2^n+1} + \left|B_3\right|_{2^n+1}\right|_{2^n+1} \end{aligned}$$

B_1 ja B_2 ovat n-bittisiä lukuja ja niiden on oltava vähemmän kuin $(2^n + 1)$ joten jakojäännös luvulle 2^{2n} suhteessa luvulle $(2^n + 1)$ on

$$\begin{aligned} \left|2^{2n}\right|_{2^n+1} &= \left|2^n 2^n\right|_{2^n+1} \\ &= \left|2^n + 1 - 1\right|_{2^n+1}\left|2^n + 1 - 1\right|_{2^n+1} \\ &= -1 \times -1 \\ &= 1 \end{aligned}$$

Edellisestä huomataan, että jakojäännös luvulle 2^n suhteessa luvulle $2^n + 1$ on -1. Täten tästä seuraa:

$$r_1 = \left|B_1 - B_2 + B_3\right|_{2^n+1}. \qquad (33)$$

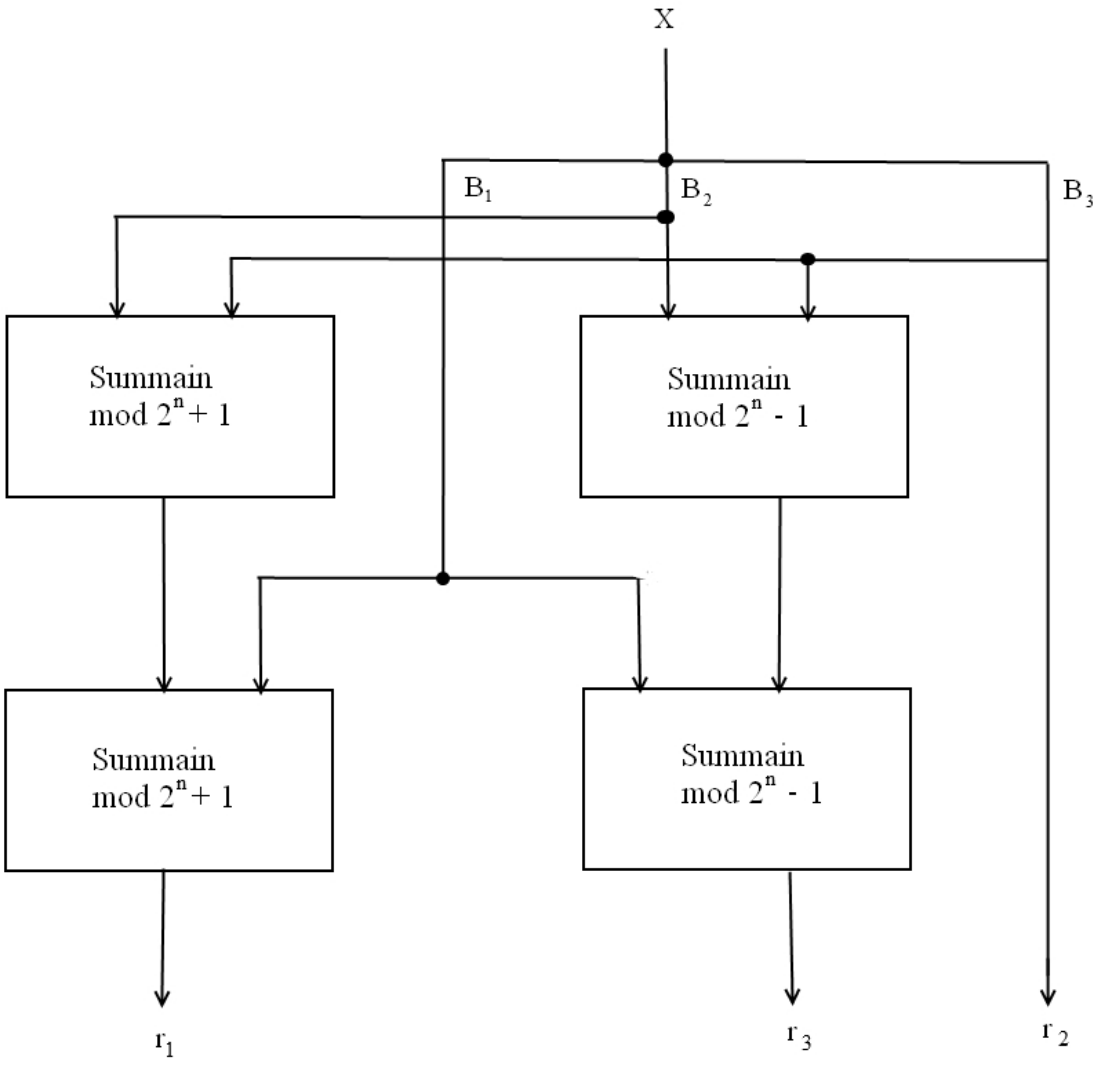
Samalla lailla voidaan laskea jakojäännös r_3

$$\left|2^{2n}\right|_{2^n-1} = \left|2^n - 1 + 1\right|_{2^n-1} \left|2^n - 1 + 1\right|_{2^n-1} = 1 \times 1 = 1.$$

Myös jakojäännös 2^n suhteessa luvulle 2^n- 1 on 1, joten

$$r_3 = \left|B_1 + B_2 + B_3\right|_{2^n-1}. \qquad (34)$$

Kokonaisvaltaisesti yllä oleva menetelmä ei ole järin kompleksinen ja tällä tavalla muunnoksesta saadaan tehokas niin pinta-alan kuin kompleksisuudenkin suhteen. [20]

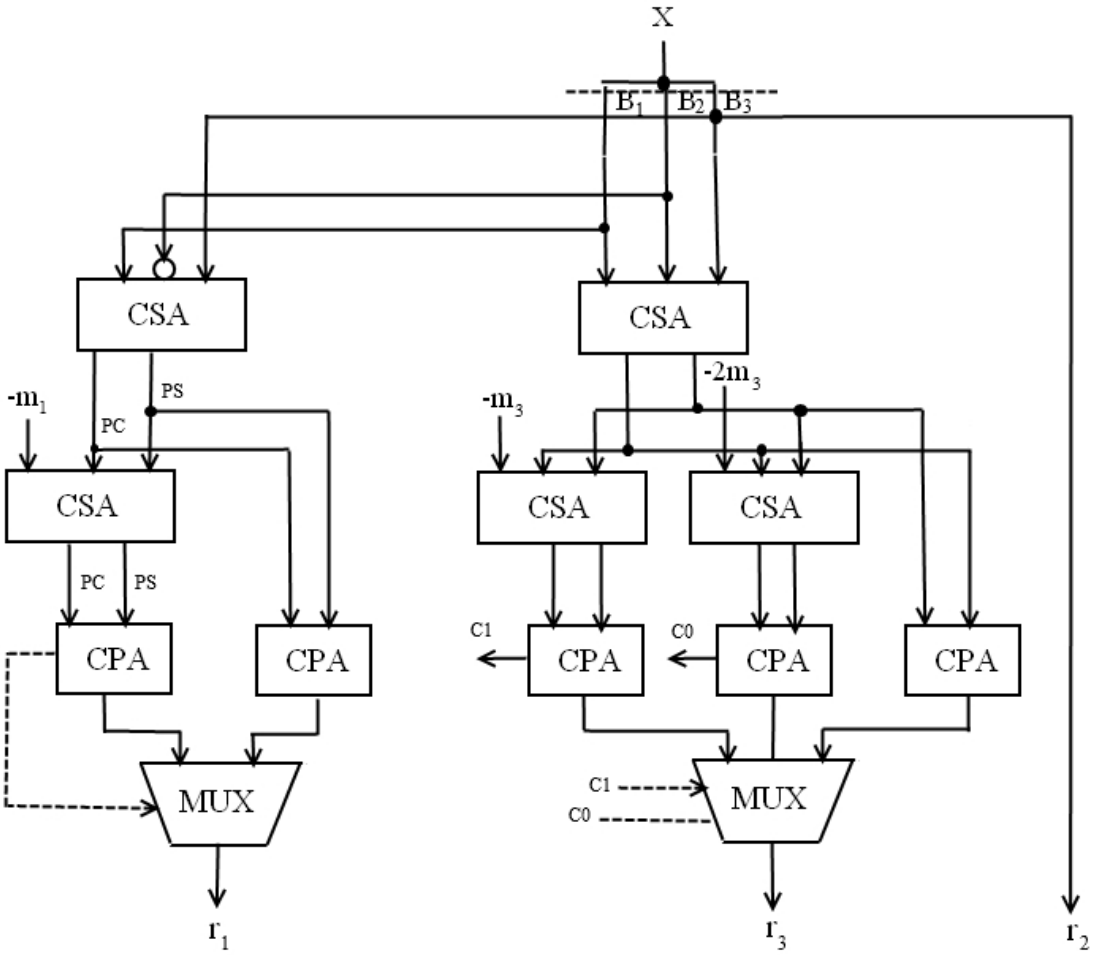


Kuva 11. Moduulisarjan {2^n – 1, 2^n, 2^n + 1} eteenpäin käännös. [20]

Edelliseen menetelmään nojaten kuvassa 11. nähdään perusperiaate miten BR-käännös on tehty {2^n – 1, 2^n, 2^n + 1} moduulisarjalla. Vaikka suunnitelma on suoraviivainen, niin moduulisummainten on oltava täysin muisti-bitin monistavia summaimia (Carry-propagate Adder, CPA), joka saattaa näkyä tehokkuuden

menettämisenä. Suorituskykyä voidaan edistää monin tavoin, mutta kaikki tavat vaativat lisää logiikkaa. [20]

Keskeiset tavat, joilla suorituskykyä saadaan lisättyä, ovat muisti-bitin muistavien summaimen (Carry-save Adder, CSA) käyttämien ja rinnakkaisuuden lisääminen muunnokseen. Kuva 12. esittää jatkokehitettyä mallia samalle muunnokselle kuin kuvassa 11. Kuvassa 12. CSA-summaimet ottavat kolme sisäänmenoa ja tuottavat kaksi ulostuloa, joista toinen on osittainen summa (partial sum, PS) ja toinen on osittainen muistibitti (partial carry, PC), jotka syötetään CPA-summaimelle. Jakojäännöksille r_1 ja r_3 joudutaan tarvittaessa tekemään vähennyslaskuja oikean lopputuloksen saamiseksi joten nämä kaikki erilaiset laskuoperaatiot on otettu huomioon ja nämä mahdolliset kombinaatiot on laskettu rinnakkaisesti ja oikea vaihtoehto valitaan multiplekserin avulla. [20]



Kuva 12. Moduulisarjan {$2^n - 1$, 2^n, $2^n + 1$} paranneltu eteenpäin käännös. [20]

4.1.2. BR-käännös mielivaltaisille moduuleille

Mielivaltaisia moduuleita käytetään yleensä sovelluksissa, joissa dynaaminen alue on huomattavan suuri ja tietyt erityiset moduulit aiheuttavat joitain rajoituksia.

Mielivaltaisien moduulien suuri määrä ja suuri koko aiheuttavat monimutkaisuutta muunnokseen ja vaativat erityistä rautatason toteutusta.

Tapoja, joilla tälläisiä muunnoksia suunnitellaan, ovat yleisesti look-up-taulukot, eli pääsääntöisesti ROM-muistit (Read-only memory), joihin tallennetaan kaikki mahdolliset jakojäännökset käytettävistä moduuleista. Tämän tekniikan huonona puolena tulee helposti ja nopeasti vastaan käytettävän mustin määrä joka vie koko toteutuksessa valtavasti tilaa. [20]

Perusperiaate ROM-muisteja käytettäessä on pilkkoa suuri käännös pienempiin osiin ja näin saada koko muunnoksesta nopeampi ja tehokkaampi. Avainasemassa on kahdenpotenssin jakojäännös, sillä

$$X = x_{n-1}x_{n-2}x_{n-3}\ldots x_0 = \sum_{j=0}^{n-1} x_j 2^j \quad \text{ja}$$

$$|X|_m = \left|\sum_{j=0}^{n-1} x_j 2^j\right|_m = \left|\sum_{j=0}^{n-1} \left|x_j 2^j\right|_m\right|_m .$$

Tapoja, millä näitä osittaisia summia ja kokonaistulos lasketaan yhteen, on monia. Voidaan laskea sarjassa, peräkkäin, rinnakkain tai näiden erilaisilla yhdistelmillä. [20]

4.2. Takaisinkäännös

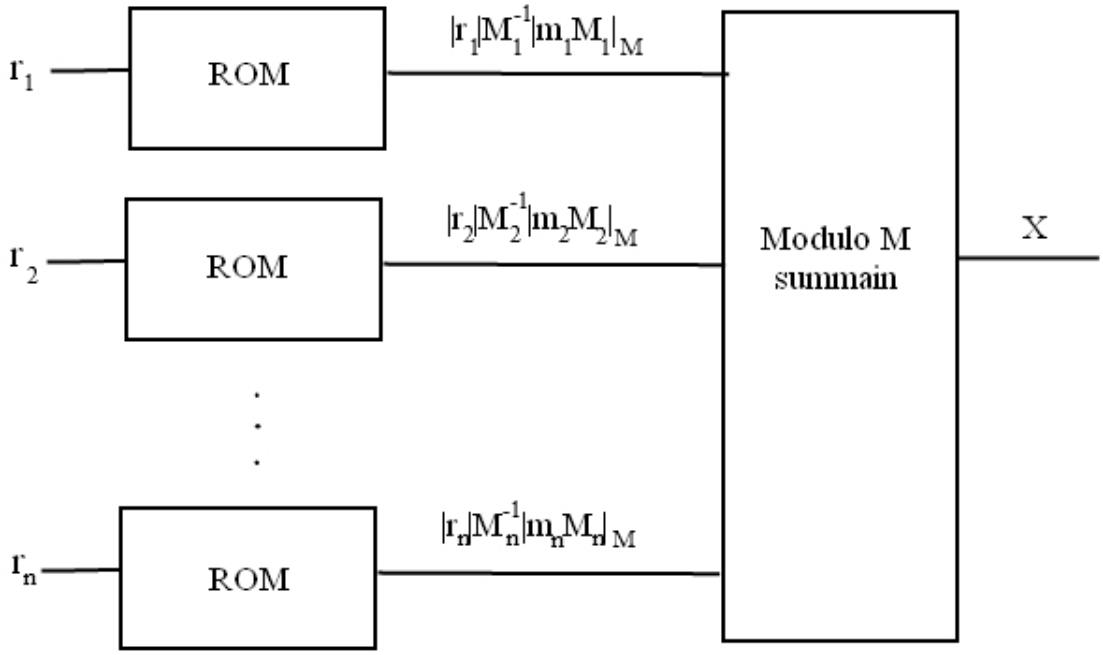
RNS-tekniikoita on vältelty juuri sen takia, että järjestelmä on vähän vaivalloinen ja hyödyntämisen suurimpana ongelmana on ollut takaisinkäännös (Residue-to-binary, RB). Siksi tämä on ollut jo pitkään keskeisimpiä tutkimuskohde RNS-tekniikassa. Takaisinkäännös on pääsääntöisesti koko järjestelmän pullonkaula ja näin ollen vie koko prosessissa paljon aikaa ja myös pinta-alaa koko järjestelmästä. Täten takaisinkäännös on ollut luonnollinen tutkimuskohde, koska sitä parantamalla saadaan koko järjestelmästä paljon parempi, nopeampi ja tehokkaampi.

Takaisinkäännökseen käytetään lähinnä CRT- tai MRC-menetelmää. MRC-tekniikan hankaluuden takia on vuosien saatossa kehitelty eri malleja ja uusia käännös variaatioita alkuperäisestä muunnoksesta. Näillä saavutetaan tiettyjä etuja kuten nopeudessa ja pinta-alassa tietyissä tapauksissa.

MRC-menetelmä ei ole niin rinnakkainen kuin CRT-menetelmä vaan yleisemmin peräkkäinen joka aiheuttaa turhaa viivettä järjestelmään. MRC-menetelmää voidaan vain käyttää tapauksissa joissa moduulien määrä on rajoitettu. Neljä moduulia on suurin määrä jossa vielä voidaan käyttää takaisinkäännöksessä MRC-menetelmää. Tosin kahden moduulin järjestelmissä MRC-menetelmää käyttäen saadaan käännös aikaiseksi helpommin kuin CRT-menetelmällä ja näin koko järjestelmä voidaan tehokkaasti toteuttaa laitteistolla. [13]

Kirjallisuudessa melkein kaikki takaisinkäännökset perustuvat siis joko CRT- tai MRC-menetelmiin. Kaikki muut käännökset ovat edellisien variaatioita tai

jatkokehityksiä, jotka ovat kehitetty yleensä vain tietylle sovellukselle. VLSI-tekniikassa (Very-large-scale integration) näiden algoritmien toteutukset on hoidettu monin eri tavoin. Käyttäen joko look-up-taulukoita (ROM-muisteja), puhdasta kombinatoorilogiikkaa tai erilaisia yhdistelmiä molempia. Lopputulos on kuitenkin monimutkainen ja kallis. [18]



Kuva 13. Havainnollinen kuva kolmen moduulin RB-käännös CRT-menetelmää käyttäen. [18]

Kuva 13 esittää yksinkertaista mallia RB-muunnoksesta, jossa on käytetty look-up-taulukoita kertolaskuissa ja nämä on summattu yhteen suurella modulo-summaimella. Takaisinkäännös siis näyttää todella yksinkertaiselta ja suoraviivaiselta, mutta suurimmaksi huoleksi tulee moduulien pienentäminen, sillä takaisinkäännöksessä joudutaan käyttämään todella suurta summainta. Tämä summain vie fyysistesti pinta-alaltaan niin paljon tilaa, että takaisinkäännöksessä suunnittelu on todella tärkeä osa koko järjestelmän suunnittelussa. [18] [20]

Takaisinkäännökseen on jo olemassa jonkinlaisia ratkaisuja, joilla sen kompleksisuutta ja kokoa sekä nopeutta on saatu parempaan suuntaan. Yksi näistä on mm. M.Griffinin kehittämä autoskaalaus-CRT (ε-CRT). Siinä skaalataan RNS-järjestelmän moduuleita, jolloin takaisinkäännös saadaan järjestettyä vain käyttämällä pienemmän kokoisia look-up-taulukoita ja normaaleita binäärisummaimia. [19]

4.2.1. Moduulien valinta

VLSI-tekniikan päätavoite on vähentää kustannuksia ja parantaa tehokkuutta järjestelmän nopeudessa, tehokkuudessa ja kompleksisuudessa. RNS-järjestelmä tuo

uuden avun myös VLSI-tekniikkaan, jonka kautta voidaan suunnitella nopeampia ja parempia sovelluksia. [14]

CRT-muunnoksessa suurin ongelma on pääasiassa laitteistototeutus ja sen tehokkuuden saaminen tarpeeksi korkealle tasolle. Suurin vaikutus tässä on moduulien valinnalla. Oikeilla moduuleilla saadaan huomattavia parannuksia niin nopeuteen kuin myös käännökseen vaativaan pinta-alaan. Moduulisarja $\{2^n - 1, 2^n, 2^n + 1\}$ on tosi paljon käytetty juuri edellisten seikkojen vuoksi. On tutkittu, että moduulisarja $\{2^n - 1, 2^n, 2^n + 1\}$ on tehokkain kaikilla RNS-järjestelmän osa-alueilla varsinkin tehokkuuden ja pinta-alan suhteen keskisuurilla dynaamisilla alueilla (vähemmän kuin 22 bittiä). [15]

Kun dynaaminen alue kasvaa suuremmaksi, ei edellinen kolmen moduulin valinta ole enää tehokkain. Näin suuremmilla dynaamisilla alueilla joudutaan kasvattamaan moduulien määrää, jotta saavutettaisiin parhaimmat nopeudet ja pienimmät pinta-alat. Yleisesti 2^n-moduuli valinnoilla nopeimmat ja tehokkaimmat järjestelmät saadaan, kun moduuli-sarja on muotoa $\left\{2^{n_l}, 2^{n_1}+1, 2^{n_1}-1, 2^{n_2}\pm 1, 2^{n_r}\pm 1\right\}$. [15]

4.2.2. Uudet CRT-tekniikat

Suurimmat ongelmat tulevat ilmi RB-käännöksessä. Suora CRT-menetelmän käännös VLSI-tekniikalle ei ole tehokasta suurien modulo-operaatioiden takia. Tämän takia on kehitetty monia uusia CRT-menetelmiä, jossa alkuperäistä CRT-käännöstä on pilkottu pienempiin osiin tai muuten muokattu haluttuun suuntaan. Nämä uudet CRT-muunnokset mahdollistavat paremman ja nopeamman tekniikan. Suurin tekijä kuitenkin saavutetaan muunnoksien toteutuksissa, jotka vievät huomattavasti vähemmän pinta-alaa. [14]

Monia käännöksiä on kehitetty alkuperäisestä CRT-käännöksestä. Ns. New CRT1- ja New CRT2-menetelmät ovat muokkauksia alkuperäisestä. Nämä uudet CRT1- ja CRT2-menetelmät käyttävät hyväkseen tiettyjä arkkitehtuureita ja algoritmeja helpottaakseen käännösprosessia. Nämä pääasiassa muokkaavat CRT-lauseketta toiseen muotoon, jotta saadaan aikaiseksi haluttu ja toivottu muoto joka on tietyillä arkkitehtuureilla mahdollista saada kompaktimpaan muotoon. Näillä uusilla algoritmeilla saadaan nopeampi toteutus sekä pienempi pinta-ala. [13]

Kolmas uusi CRT-käännös on kehitetty pääasiassa kolmen 2^n-moduulien järjestelmään. Tässä tapauksessa käytetään moduulien valintoja hyväksi ja näin saadaan uuden muunnoksen avulla yksinkertaistettua käännöstä huomattavasti.

Tälle viimeisimmälle mainitulle CRT-muunnokselle voidaan CRT-algoritmi muuntaa seuraavasti. Yleisesti annetun moduulijärjestelmän $\{m_1, m_2, \ldots, m_n\}$ ja jakojäännös numeron $(r_1, r_2, \ldots, r_n)$ voidaan kääntää binääriluvuksi X seuraavasti:

$$X = r_1 + m_1 \sum_{j}^{n} \left| w_j r'_j \right|_{m_2 \cdots m_n} \tag{35}$$

$$\text{missä} \quad \text{n} > 1, \text{w}_1 = \frac{\hat{m}_1 \left| \frac{1}{\hat{m}_1} \right|_{m_1} - 1}{m_1}, r'_1 = r_1, w_j = \frac{\hat{m}_j}{m_1} \; ja \; r'_j = \left| \frac{r_j}{\hat{m}_j} \right|_{m_j} \quad j = 2, 3, \cdots, N$$

Suosituin ja ehkä eniten käytetyin moduuli-sarja $\{2^n - 1, 2^n, 2^n + 1\}$hyötyy tästä uudesta muunnoksesta paljon ja käännösprosessi yksinkertaistuu huomattavasti. Vaihtamalla hieman moduulien järjestystä muotoon $\{2^n, 2^n + 1, 2^n - 1\}$, niin saadaan kaava 22 tässä tapauksessa muotoon,

$$X = r_1 + 2^n Y \tag{36}$$

$$\text{missä} \quad \text{n} > 1 \text{ ja } \text{Y} = \left| (r_1 - r_2) + 2^{n-1}(2^n + 1)(r_3 - 2r_1 + r_2) \right|_{2^{2n}-1}$$

Kaavan 36 avulla saadaan CRT-käännös todella yksinkertaiseen muotoon, joka nopeuttaa prosessia ja käytettävä pinta-ala pienentyy murto-osaan alkuperäisestä. On huomattava, että tämä menetelmä onnistuu vain 2^n muotoisiin moduuleihin. Tämä ei ole kuitenkaan niin suuri haitta, sillä juuri 2^n-moduulisarjat ovat olleet jo pidemmän aikaan suositumpien moduulisarjojen joukossa ja siksi myös tutkituin. [14]

RB-käännökseen voidaan käyttää hyödyksi myös MRC-tekniikkaa, jossa suurin hyöty on tehonsäästö. MRC-tekniikassa tulee helposti dynaaminen alue vastaan ja myös suurilla moduulien määrällä MRC-tekniikan hyöty pienenee. Dynaamista aluetta voidaan kasvattaa muokkaamalla olemassa olevia moduuleita.

Yksi tällainen on $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$ moduulisarja. Perinteiseen $\{2^n - 1, 2^n, 2^n + 1\}$ moduulisarjaan verrattuna tässä on korvattu $\{2^n + 1\}$-moduuli $\{2^{2n+1} - 1\}$-moduulilla. Tämän avulla saadaan hankalin käsiteltävä moduuli korvattu toisella ja samalla kasvatettua dynaamista aluetta. Tämän yksinkertaisen vaihdoksen ansiosta muunnoksesta saadaan samalla tehokas ja nopea. [16]

Toinen moduulisarja, jossa käytetään MRC-tekniikkaa hyväksi, on edeltävän tapainen, mutta muokkaamalla viimeistä moduulia saadaan siitä vielä tehokkaampi. Tällainen on moduulisarja $\{2^n - 1, 2^n, 2^{2n-1} - 1\}$. Samalla käännökseen on käytetty ainoastaan summaajia ja muutamia loogisia portteja. Tällä tavalla käännöksestä on saatu tehokkaampi kuin vastaavilla neljän moduulin sarjoilla, joilla on sama dynaaminen alue. Järjestelmästä on saatu nopea sekä raudan osalta että käännöksen viiveitten pienenemisenä. [17]

4.2.3. $\{2^n - 1, 2^n, 2^n + 1\}$ RB-käännös

Takaisinkäännös $\{2n - 1, 2n, 2n + 1\}$ moduulisarjalle ja myös $\{2^n - 1, 2^n, 2^n + 1\}$ moduulisarjalle voidaan helposti johtaa CRT-muunnoksen avulla. Niin kuin eteenpäin muunnoksessa, niin tässäkin tapauksessa käännös jaetaan pienempiin osiin ja kokonaisuuksiin, jotka sitten lopuksi parsitaan yhteen saaden koko käännös aikaiseksi. [20]

Yhtälölle $X = w_i x_i$ saadaan painokertoimet w_i laskettua seuraavasti, kun moduulisarjana on $m_3 = 2n - 1$, $m_2 = 2n$ ja $m_1 = 2n + 1$:

$$w_1 = \frac{(m_1 + 1)m_2 m_3}{2}$$
$$w_2 = m_1(m_2 - 1)m_3$$
$$w_3 = \frac{m_1 m_2 (m_3 + 1)}{2}.$$

Painokertoimien avulla saadaan laskettua luvun $\langle x_1, x_2, x_3 \rangle$, jossa siis $x_i = |X|_{m_i}$, luonnollinen luku X seuraavasti,

$$X = \left| \sum_{i=1}^{3} w_i x_i \right|_{m_1 m_2 m_3} . \qquad (37)$$

Jotta vältyttäisiin suuren moduulin M käyttämistä, kaavaa 37 voidaan kehittää eteenpäin. Tällöin säästytään suurilta kertolaskuilta jotka vie paljon tilaa ja ovat hitaita operaatioita.

$$|X|_M = |A + m_1(m_2 - 1)m_2 x_2 + C|_M \text{, jossa}$$

$$A = \frac{(m_1 + 1)m_2 m_3}{2} x_1 \text{ ja } C = \frac{m_1 m_2 (m_3 + 1)}{2} x_3$$

Merkitsemällä moduuleitten tuloa $M = m_1 m_2 m_3$ ja siistimällä kaavaa edelleen saadaan,

$$|X|_M = |\hat{A} + m_1(m_2 - 1)m_2 x_2 + \hat{C}|_M \text{, jossa}$$

$$\hat{A} = \left(\frac{M}{2} + \frac{m_2 m_3}{2} \right) x_1 \text{ ja } \hat{C} = \left(\frac{M}{2} + \frac{m_1 m_2}{2} \right) x_3 .$$

Josta edelleen saadaan,

$$|X|_M = \left| \frac{M}{2}(x_1 + x_3) + \frac{m_2 m_3}{2} x_1 - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \right|_M . \qquad (38)$$

Viimeisessä lausekkeessa termi (x_1 + x_3) voi olla joko parillinen tai pariton. Jos se on parillinen niin, $\left| \frac{M}{2}(x_1 + x_3) \right|_M = 0$. Jos se taas on pariton niin, $\left| \frac{M}{2}(x_1 + x_3) \right|_M = \frac{M}{2}$. Tällöin kaava 27 saadaan muotoon:

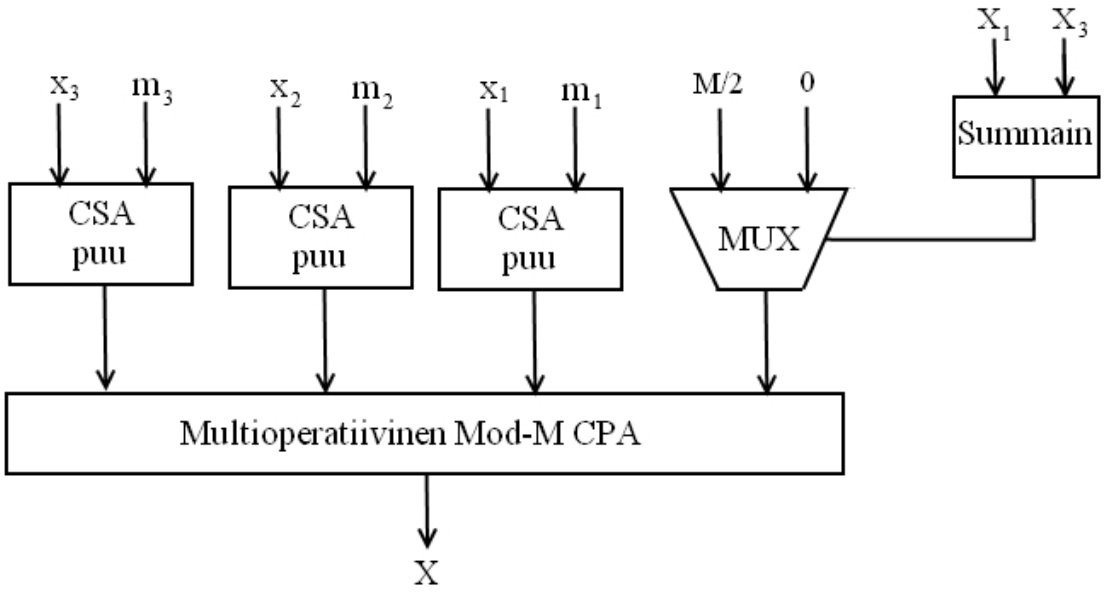
Jos (x_1 + x_3) on parillinen,

$$|X|_M = \left| \frac{m_2 m_3}{2} x_1 - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \right|_M = |M_1 x_1 M_2 x_2 + M_3 x_3|_M . \qquad (39)$$

Jos (x_1 + x_3) on pariton,

$$|X|_M = \left| \frac{M}{2} + \frac{m_2 m_3}{2} x_1 - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \right|_M = \left| \frac{M}{2} + M_1 x_1 M_2 x_2 + M_3 x_3 \right|_M . \qquad (40)$$

Näiden kahden uuden yhtälön avulla on saatu pilkottua kertolaskuja pienemmiksi, kuin mitä ne olivat alkuperäisessä yhtälössä. Kaavojen 39 ja 40 mukainen toteutus on kuvassa 14. Termiä (x_1 + x_3) ei periaatteessa tarvitse laskea sillä, siitä riittää vain tieto onko se parillinen vai pariton. Siksi riittää vain summaimen viimeisimmän bitin tieto onko se joko 0 vai 1. Tämä bitti valitsee multiplekseristä oikean vaihtoehdon joko $\frac{M}{2}$ tai 0. [20]



Kuva 14. Takaisinkäännös moduulisarjalle {2n – 1, 2n, 2n +1}. [20]

Moduulisarjan {2n – 1, 2n, 2n + 1} tulosta voidaan käyttää samalla lailla hyödyksi moduulisarjalle {2^m – 1, 2^m, 2^m + 1} koska tahansa milloin n on parillinen.

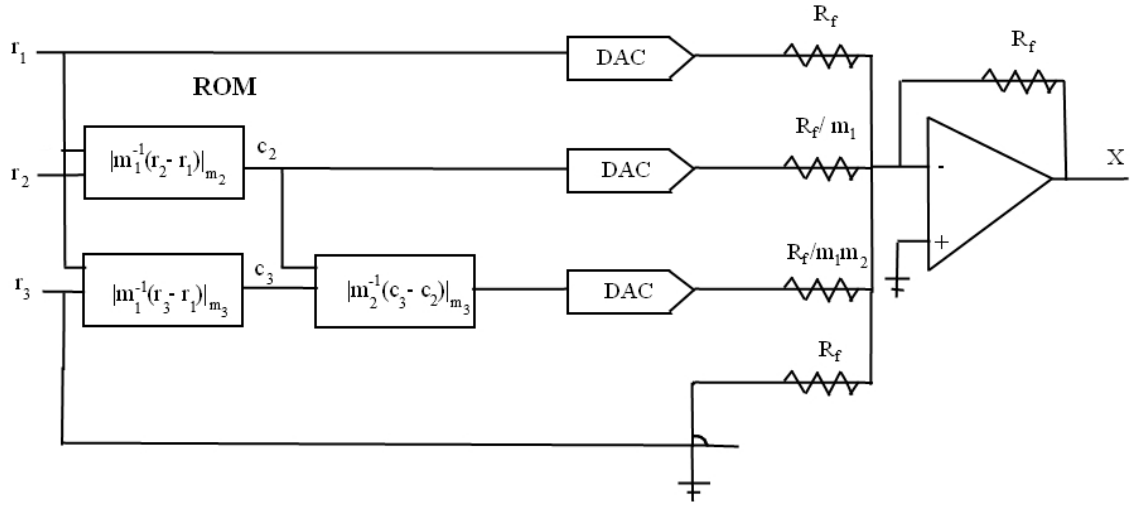
Moduulisarjan $\{2^m - 1, 2^m, 2^m + 1\}$ suosio on perusteltua sillä sen toteutus on helppo suunnitella ja lopputuloksena on hyvä hyötysuhde ja alhaiset kustannukset. [16]

4.3. RNS suoraan analogiseksi käännös

RNS-käännös voidaan tehdä myös suoraan analogiseksi ilman, että luku muutetaan ensin binääriluvuiksi. Tämä tapa on kuitenkin harvinaista, sillä suurin osa sovelluksista hyödyntää binäärilukuja. Lisäksi binääriluvuilla voidaan tehdä kaikki tärkeät operaatiot mitä taas ei voida RNS-tekniikalla tehdä. [14]

Tämän menetelmän hyöty on lähinnä analogisien signaalien aritmeettisissa muokkauksissa. Suoralla käännöksellä analogiseksi saavutetaan lähinnä tehosäästöjä kun vältytään turhalta välivaiheelta. Käännöksestä saadaan myös nopeampi ja samalla päästään eroon binäärikääntäjästä ja sen suunnittelusta. Näin päästään keskivaiheesta eroon ja koko järjestelmän viive pienenee. Jotkut sovellukset myös vaativat aktiivista kommunikointia reaalisen analogimaailman kanssa, joissa tämä menetelmä on todella hyödyksi. [14]

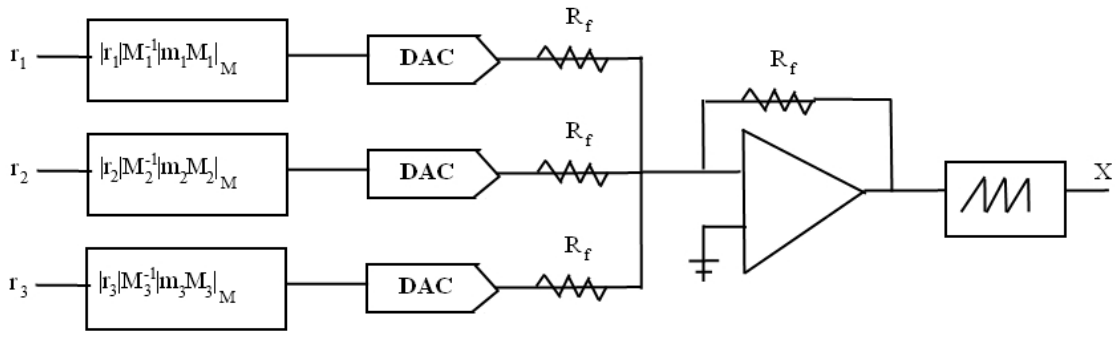
Koko RNS-prosessi menee samalla tavalla kuin binäärimuunnoksissakin, mutta tässä tapauksessa signaali käännetään suoraan analogisesta RNS-signaaliksi ja takaisin. Tässäkin tapauksessa signaalin muunnos takaisin analogiseksi on koko prosessin pullonkaula ja kaikista haastavin osa. Takaisinkäännös vie eniten tehoa koko järjestelmässä kuten binäärikäännös tapauksessakin. Menetelmää käytettäessä on suurin painoarvo takaisinmuunnoksessa ja näin saadaan koko järjestelmästä mahdollisimman tehokas. [14]



Kuva 15. Kolmen moduulin RNS-analogia käännös MRC-menetelmää käyttäen. [18]

Käänteismuunnoksen voi tehdä kuten binäärilukujen tapauksessakin, joko CRT-taikka MRC-menetelmää käyttäen. Kuvassa 15. näkyy MRC-menetelmälle arkkitehtuuri suoralle RNS-analogia muunnokselle. Kuvassa 15. on käytetty ROM-toteutusta ja analogisen summaimen yhdistelmää. Jos jokaisella moduulilla on enintään *k*-bittiä, niin

takaisinkäännös voidaan toteuttaa kolmella (2^{2k} x k) kokoisella ROM:lla ja kolmella k-bittisellä Digital-to-Analog-muuntimella (Digital-to-analog converter, DAC). Analogiset ulostulot on yhdistetty analogisella summavahvistimella. [14]



Kuva 16. Kolmen modulin RNS-analogia käännös CRT-menetelmää käyttäen. [18]

Vastaavasti käännös voidaan toteuttaa perinteisellä CRT-menetelmällä. Kuvassa 16. on yksi arkkitehtuuri suoralle RNS-analogia muunnokselle käyttäen CRT-menetelmää. kuvassa on käytetty ROM-toteutusta, jossa jokainen moduuli on käännetty rinnakkain ja ulostulo summattu analogisella summavahvistimella. Jos jokainen moduuli on k-bittiä, niin osasumma saadaan käyttämällä kolme (2^k x $3k$) ROM-moduulia. Ulostulo on näin ollen modulo M, jonka suuruus on $3k$-bittiä. Nämä osasummat käännetään analogiseksi $3k$-bittisellä DAC-muuntimella. Lopuksi osasummat yhdistetään analogisella summavahvistimella. [18]

Edellisissä esimerkeissä ROM-muistien koko määrää suurimmalta osin koko järjestelmän nopeuden. Toisaalta pienentämällä ROM-muistien kokoa saadaan koko järjestelmästä pienempi kokoinen. ROM-muistien koko on tärkeässä roolissa järjestelmää suunniteltaessa ja on pullonkaula nopeuden ja järjestelmän koon suhteen. Myös suurin hyöty normaaliin käännökseen binäärimaailmaan on se, että tällä tekniikalla päästää eroon suuresta modulosummaimesta, joka on suurin resurssien haukkaava osa RB-muunnoksessa. [18]

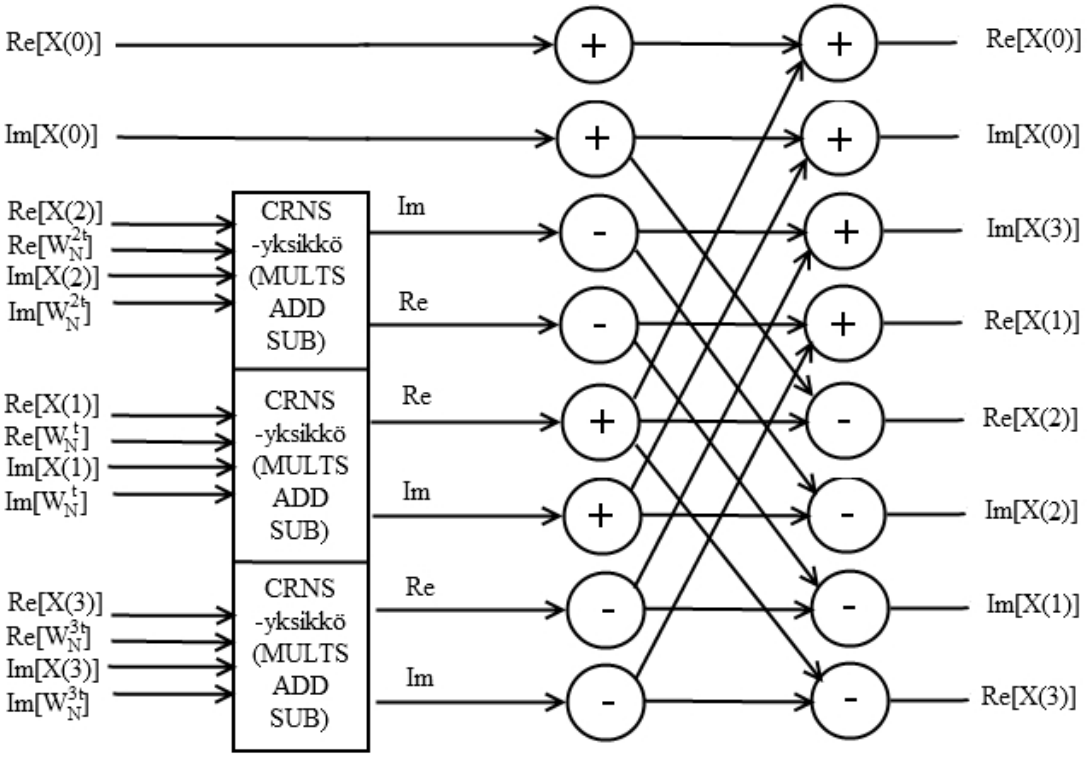
4.4. RNS-tekniikan sovelluksia

RNS-järjestelmä soveltuu mainiosti Fourier-muunnoksien esittämiseen sillä muunnoksissa on paljon yhteenlaskuja ja kertolaskuja. Tämän takia RNS-järjestelmä on ylipäänsä olemassa, sillä RNS-järjestelmässä summa- ja kertolaskut ovat todella helppoja operaatioita ja nopeita toteuttaa. RNS-tekniikka antaa suurimman hyödyn nopeudessa, mutta sen käyttöönotossa on omat kynnyksensä ja se aiheuttaa lisätyötä koko järjestelmän suunnittelussa.

FFT:n laskennassa ja suunnittelussa RNS tuo lisää kapasiteettia silloin, kun laskenta-aikaa ja rautatason kompleksisuutta halutaan pienentää. RNS-järjestelmän tapa hajottaa isot operaatiot moniin pienempiin kokonaisuuksiin tuo lisää tarkkuutta,

yksinkertaisuutta ja nopeutta. RNS-järjestelmä antaa myös hyvän vaihtoehdon sovelluksille, jotka tarvitsevat suurta tarkkuutta. [20]

FFT:n RNS-toteutuksissa kaikki ei-kokonaisluvut, kuten esim. twiddle-kertoimet ja välitulokset, tarvitsee normalisoida kokonaisluvuiksi. Kaikkien kertolaskujen tulokset pitää säilyttää täydellä tarkkuudella ja useiden perättäisen aritmeettisten operaatioiden takia lukuarvot alkavat kasvaa liian suuriksi. RNS:n käyttämisellä on suuret rajoitteet lukualueen ylivuodon takia. Ylivuodon estämisestä joudutaan käyttämään skaalausta, joka on todella vaivalloinen operaatio RNS-järjestelmässä. Radix-4 FFT on ominaisuuksien puolesta paras vaihtoehto, jolloin tarvitaan mahdollisimman vähän skaalausta. Siinä on vähiten perättäisiä kertolaskuoperaatioita muihin verrattuna, jolloin säästytään vähemmällä skaalaamisella. [20]



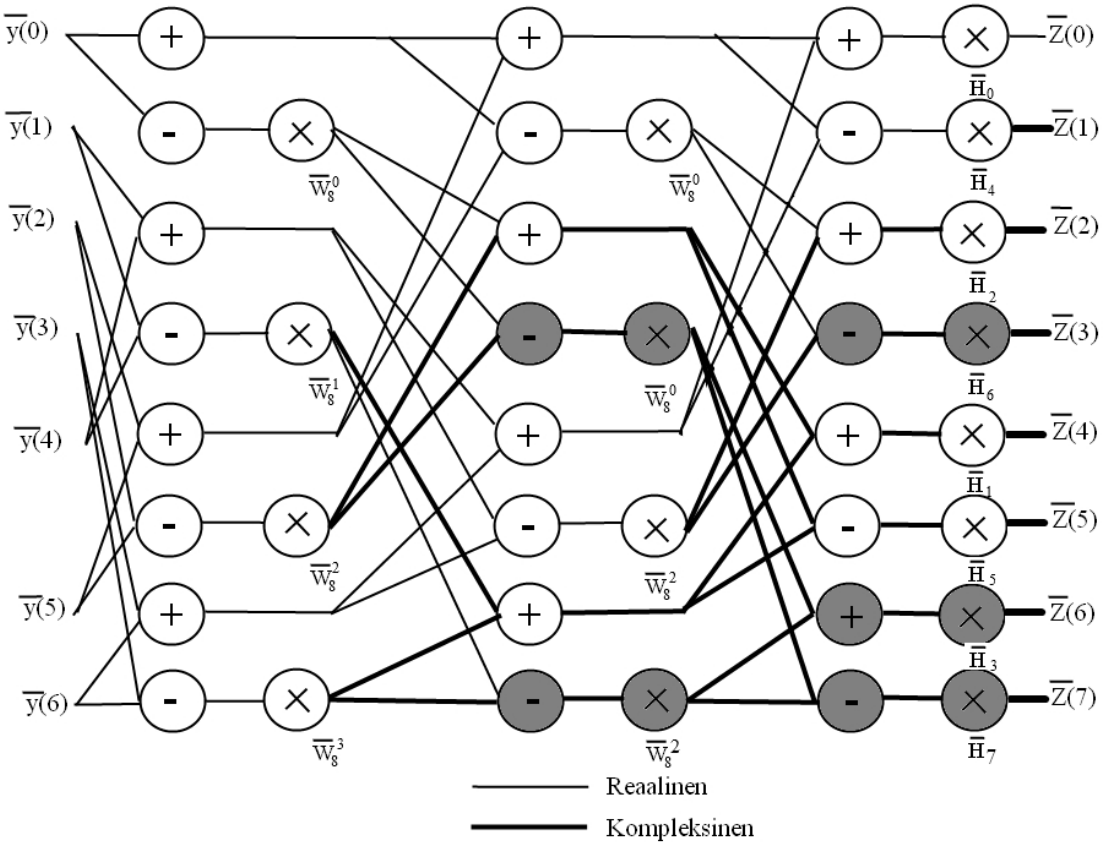
Kuva 17. Radix-4 CRNS FFT arkkitehtuuri. [21]

Kuvassa 17. on arkkitehtuuri radix-4 CRNS-toteutukselle. Tämä sisältää useita kompleksiarvoisia RNS-yksiköitä ja summaimia. Yllä olevalle arkkitehtuurille voidaan suunnitella erilaisia variaatioita, jotka vaikuttavat sen nopeuteen ja toteutuksen kompleksisuuteen. Kuva 17. on näistä kaikkein nopein ja alla oleva taulukosta nähdään miten järjestelmän nopeus riippuu toteutuksesta. [20]

	Radix-4 CRNS Butterfly		
Viive Δ (ns)	Kertojia	Summaimia	Tasoja
1	12	11	4
2	6	6	4
4	3	3	4
8	2	2	4

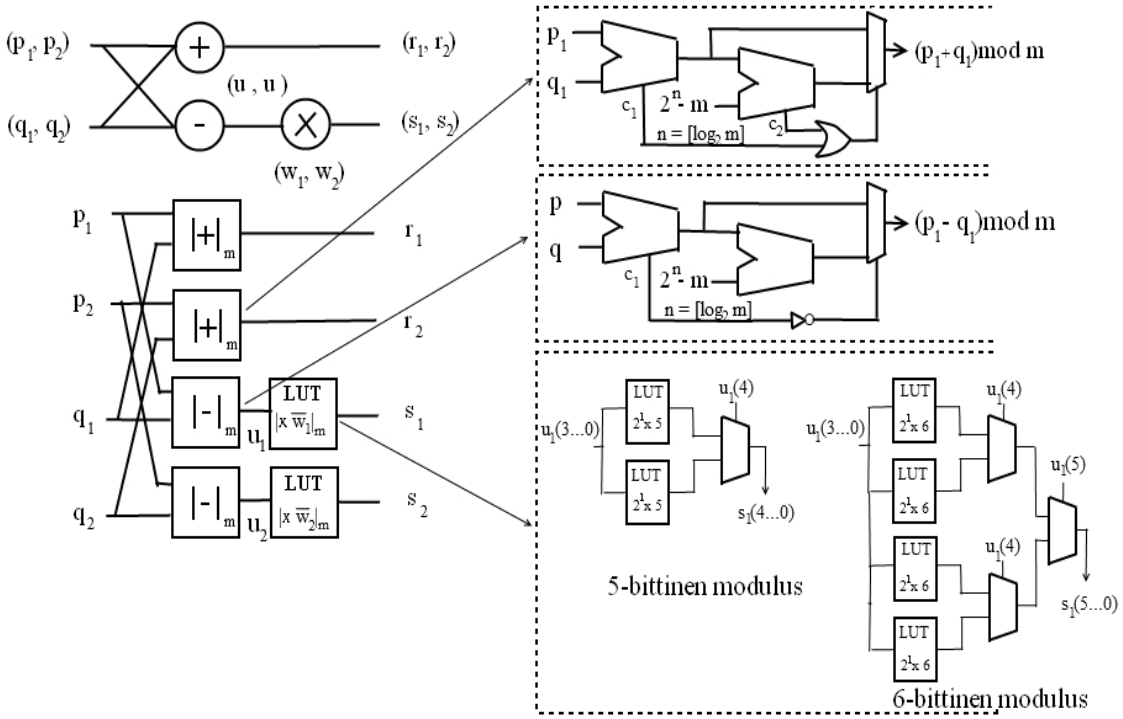
Taulukko 1. Viive erilaisilla Radix-4 CRNS Butterfly toteutuksilla. [20]

Kuvassa 18. on QRNS-arkkitehtuuri, jolla pystytään laskemaan 8-pisteen Diskreetti kosini muunnos (Discrete Cosine Transform, DCT). Arkkitehtuuri perustuu radix-2 DIF FFT algoritmiin. [21]



Kuva 18. QRNS Radix-2 FFT arkkitehtuuri [21]

Kun kompleksiset binäärisummaimet ja kertojat korvataan tässä tapauksessa QRNS-summaimilla ja kertojilla, niin jokainen summain, vähentäjä tai kertoja koostuu vastaavasti ainoastaan kahdesta modulaarisesta summaimesta, taikka kertojasta. Koska sisään menevä sekvenssi ja moni twiddle-kerroin ovat reaalisia, niin osa QRNS-aritmeetiikan moduuleista sisältää vain reaaliset sisäänmenot ja nämä voidaan pienentää vain yhteen modulaariseen summaimeen taikka kertojaan. [21]



Kuva 19. QRNS radix-2 FFT butterfly. [21]

Kuvassa 19. esiintyy QRNS radix-2 perhonen, joka koostuu kahdesta summaimesta ja vakiokertoimisesta modulaarisesta kertojasta. Nämä aritmeettiset moduulit ovat helppo kuvata laitetasolle tarjoten parhaan tehokkuuden niin pinta-alan kuin nopeuden suhteen. Kuvassa 19. nähdään myös miten on suunniteltu 5- ja 6-bittiset kertojat käyttäen LUT-kertojia, sillä ne tarjoavat suuren nopeuden ja ne voidaan rakentaa pienillä ROM-muisteilla. Yllä olevalla tekniikalla saavutetaan jopa 140% parannus nopeuteen ja 20% parannus viiveeseen kuin vastaavalla tehty kahdenkomplementti-metodilla. [21]

5. YHTEENVETO

Signaalinkäsittely on ollut vahvassa kasvussa koko uuden vuosituhannen ja yhä useampi sovellus käyttää edes jotenkin signaalinkäsittelyä osana tekniikkaansa. Tällä vuosituhannella on signaalinkäsittelystä tullut ylivoimaisesti suurin osa-alue, sillä kaikki kuva ja videosysteemit aina sisältävät signaalinkäsittelyä jossain muodossa. Kun otetaan tietokoneiden ja puhelimien hurja kehittyminen niin DSP-järjestelmien ja niiden käyttö kyseisissä laitteissa on vaan kasvanut.

Signaalinkäsittelyn kasvaessa on myös RNS-tekniikalle kehitelty paljon uusia tekniikoita ja RNS-järjestelmät ovat saaneet enemmän huomiota useissa toteutuksissa. RNS-järjestelmän huonot puolet ja rajoitukset eivät silti ole hävinneet minnekään, mutta näille ongelmille on saatu kehitettyä erilaisia tapoja kiertää tai vähentää niitä.

RNS-järjestelmän todellinen läpilyönti vielä odottaa itseään. Tämän näkee myös siinä, että RNS itse terminä ei vielä esiinny kovinkaan paljon ja siihen törmääminen voi normaalissa olosuhteissa olla vaikeaa.

RNS tutkimuskohteena on kuitenkin ollut koko ajan kasvavaa nyt vuosituhanteen vaihtuessa ja sen tuomat edut ovat olleet tosissaan tutkimuksen alla. Paljon on kehitetty pienempiä ratkaisuja tietyille sovelluksille, mutta sitä isompaa ja uudelleen käytettävää tekniikkaa ei ole vielä saatu kunnolla kehitettyä, jotta RNS-järjestelmän käyttö lisääntyisi huomattavasti.

RNS-järjestelmän tuomat edut ovat kiistattomat ja useat tutkimukset jopa 1980-luvulta lähtien tukevat sitä. Nykyajan tekniikat ja tietokoneet sekä laitteet ovat mahdollistaneet tutkia ja kehittää uusia tapoja ratkaista RNS-järjestelmän huonoja puolia.

RNS-tekniikan käyttöönotossa täytyy ottaa aina huomioon kokonaiskuva ongelmaa ratkaistaessa. Täytyy myös muistaa, että RNS-tekniikan käyttö vaatii muunnokset RNS-järjestelmään ja takaisin, joka täytyy ottaa huomioon koko systeemin nopeuteen ja tehonkulutukseen sekä pinta-alaan. Sovelluksen ollessa tarpeeksi suuri on RNS-tekniikan käyttö soveliasta. Myös, jos sovellus sisältää paljon perus aritmeettisia operaatioita saadaan RNS-tekniikalla tuotua tehoa sovellukseen.

On myös muistettava, että RNS-tekniikka mahdollistaa hyvän virheeneston, sillä jokainen moduuli RNS-järjestelmässä on itsenäinen ja näin mahdolliset virheet eivät pääse monistumaan ja kertaantumaan koko toteutukseen. Tämän ansiosta RNS:n käyttö on perusteltavaa, vaikka ei ns. tehohyötyjä saavutettaisikaan.

LÄHTEET

[1] Szabó N.S., Tanaka R.I. “Residue Arithmetic and Its Applications to Computer Technology”. United States of America 1967. McGraw-Hilll, Inc. 235p.

[2] York T.A., Srisuchinwong B., Tsalides, Hicks P.J., Thanailakis A. “Design and VLSI Implementation of Mod-127 Multiplier Using Cellular Automaton-based Data Compression Techniques”. IEEE Proceedings-e, Vol. 138, No. 5, September 1991. pp. 351-356.

[3] Ahmad A.H. “New Efficient Structure for a Modular Multiplier for RNS”. IEEE Transactions on Computers, Vol. 49, No. 2, February 2000. pp. 170-174.

[4] Leung S.H. “Application of Residue Number Systems to Complex Digital Filters”. Proc. 25th Asilomar Conf. on Circuits, Systems and Computers, Pacific Grove, CA, Nov. 1981.

[5] Michael A.S., Gregory D.P. “Applications of Quadratic-like Complex Residue Number System Arithmetic to Ultrasonics”. IEEE International Conference on ICASPP 1984. pp. 484-487.

[6] Ramirez J., Garcia A., Fernández P.G., Parrilla L., Lloris A. “A New Architecture to Compute the Discrete Cosine Transform Using The Quadratic Residue Number System”. IEEE ISCAS 2000. pp. v321-v324.

[7] Stephen Oh, Domingo G. “Implementation of a Parallel DFE using Residue Number System”. IEEE International Conference on ICASPP 1994. pp.III237-III240.

[8] Huttunen H. “Johdatus signaalin käsittelyyn 1”. Opetusmoniste 2001.

[9] linkki: http://en.wikipedia.org/wiki/fft. 15.6.2011

[10] linkki: http://cnx.org/content/m12018. 15.6.2011

[11] linkki: http://cnx.org/content/m12016. 15.6.2011

[12] linkki: http://cnx.org/content/m12027. 15.6.2011

[13] Navi K., Molahosseini A.S., Esmaeildoust M. “How to Teach Residue Number System to Computer Scientist and Engineers”. IEEE Transactions on Education 2011. pp. 156-163.

[14] Wang W., Swamy M.N.S., Ahmad M.O., Wang Y. “A study of the Residue-to-Binary Converters for the Three-moduli Sets”. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 2003. pp. 235-243.

[15] Wang W., Swamy M.N.S., Ahmad M.O. “Moduli Selection in RNS for Efficient VLSI Implementation”. IEEE Proceedings of the 2003 International Symposium on Circuits and Systems 2003. pp. 512-515.

[16] Molahosseini A.S., Navi K., Rafsanjani M.K. “A New Residue to Binary Converter Based on Mixed-Radix Conversion”. IEEE 3rd International Conference on Information and Communication Technologies: From Theory to Applications 2008. pp. 1-6.

[17] Molahosseini A.S., Sezavar S., Navi K. “A New Design of Reverse Converter for A Three-moduli Set”. IEEE International Symposium on Intelligent Signal Processing and Communication Systems 2009. pp. 57-60.

[18] Abdelfattah, O., Swidan, A., Zilic, Z. “Direct Residue-to-Analog Conversion Scheme Based on Chinese Remainder Theorem”. IEEE International Conference on Electronics, Circuits, and Systems ICECS 2010. pp. 687-690.

[19] Griffin M., Taylor F. J., Sousa M. “New Scaling Algorithms for the Chinese Remainder Theorem”, Proc of 22nd Asilomar Conference on Signals, Systems and Computers. 1988.

[20] Omondi A., Premkumar B. “Residue Number Systems, Theory and Implementation”. Imperial College Press 2007. 296p.

[21] Ramirez J., Garcia A. “A Fast QRNS-based Algorithm for the DCT And It’s Field-programmable Logic Implementation”. Journal of Circuit, Systems and Computers 2003. Vol 12, pp 111-123.