ปีการศึกษา 2532

การพัฒนา ดิสเครต พูเรียร์ ทรานสฟอร์ม

โดย

สมบัติ ตระกูลโกศล 326327

สอน ทุมเกษร 326330

อาจารย์ที่ปรึกษา

อาจารย์ เกษตร์ ศิริสันติสัมฤทธิ์

ปริญญานิพนธ์ปีการศึกษา 2532
ภาควิชา วัดคุมทางอุตสาหกรรม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง
เรื่อง การพัฒนา ดิสเครต ฟูเรียร์ ทรานสฟอร์ม

ผู้จัดทำ

1. สมบัติ ตระกูลโกศล 326327
2. สอน ทุมเกษร 326330

อาจารย์ที่ปรึกษา

.....

(อาจารย์ เกษตร ศิริสันติสัมฤทธิ์)

ชอร์ท-คัท ดิสเครท ฟูเรียร์ ทรานสฟอร์ม

สมบัติ ตระกูลโกศล
สอน ทุมเกษร
เกษตร ศิริสันติสัมฤทธิ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2533

บทคัดย่อ

ชอร์ท-คัท ดิสเครท ฟูเรียร์ ทรานสฟอร์ม เป็นเครื่องมือการคำนวนอย่างหนึ่งที่ช่วยในการวิเคราะห์สัญญาณ ตัวอย่างเช่นการวิเคราะห์เพาเวอร์สเปกตรัม และการจำลองการกรองสัญญาณ โดยการใช้ดิจิตอลคอมพิวเตอร์. การทรานสฟอร์มนี้ เป็นวิธีการคำนวน ดิสเครท ฟูเรียร์ ทรานสฟอร์ม ข้อมูลที่สุ่มแบบเรียงติดต่อกันที่มีประสิทธิภาพ. ในปริญญานิพนธ์ฉบับนี้ จะให้คำจำกัดความคุณสมบัติบางประการ ของดิสเครท ฟูเรียร์ ทรานสฟอร์ม , วิธีการคำนวนที่เร็ว สำหรับการคำนวนการทรานสฟอร์มที่ได้ทำให้อยู่ในรูปที่ง่ายแล้ว , และวางแผนวิธีการคำนวน. พร้อมกันนี้จะรวมตัวอย่าง ที่แสดงเนื้อหาที่เกี่ยวพันธ์ และเราจะได้ทำการพัฒนาเพื่อที่จะได้ อัลกอริทึม ของดิสเครท ฟูเรียร์ ที่ดีที่สุด สำหรับการคำนวนเท่าที่จะสามารถทำได้

SHORT-CUT DISDRETE FOURIER TRANSFORM

SOMBAT TRAGOONGOSON
SORN TOOMKASORN
KASET SIRISANTISAMRID ADVISOR
ACADEMIC YEAR 1989

Abstract

The short-cut discrete fourier transform is a computional tool which facilitates signal analysis such as power spectrum analysis and filter simulation by means of digital computers. It is a method for efficiently computing the discrete fourier transform of a series of data samples(refer to as time series). In this paper, the discrete fourier transform of a time series is defined , some of its properties are discussed , the associated fast method for computing this transform is derived , and some of the computation aspects of the method are pressented. Examples are included to demonstrate the concept involved , and we will develop to have got the best discrete fourier transform algorithm for computing it as possibly.

สารบัญ

บทที่ 1.
บทนำ

ถึงประโยชน์ของ DFT จะมีมากมายก็ตาม เนื่องจากมีการคำนวณที่ยุ่งยาก และใช้เวลานานมากในการประมวลในแต่ละครั้ง ถ้านำไปใช้กับ REAL TIME แล้ว จะไม่สามารถที่จะประมวลสัญญาณทันเลย แต่ถึงอย่างไรในปัจจุบันเทคโนโลยีในการสร้าง VLSI CHIP (Very Large Scale Integrate Circuit) ได้มีการพัฒนาไปมาก สามารถที่จะนำ DFT ALGORITHM ที่มีอยู่มาสร้างใส่ใน CHIP ได้ทำให้มีความเร็วสูงมาก สามารถที่จะนำมาใช้ใน REAL TIME ได้ แต่ในส่วนของ PROJECT นี้จะพัฒนา ทางด้าน SOFTWARE เพื่อนำมาใช้ในเครื่อง PC ที่มีอยู่ทั่วๆไป ซึ่งนับวันจะยิ่งมีความเร็วสูงมากและให้เกิดประโยชน์สูงสุด แต่ก่อนที่จะนำอัลกอริทึมที่มีอยู่มาใช้งานนั้น
เราจะต้องหาอันที่ดีที่สุดมาใช้ โครงงานนี้จึงเกิดขึ้นโดยเราตั้งความหวังว่าจะพยายามนำอัลกอลิทึมที่มีอยู่มาปรับปรุง หรือดัดแปลงเพื่อให้ได้อัลกอริทึมใหม่ที่ดีที่สุด

สำหรับพื้นฐานของ DFT นั้น เราไม่ได้ทำการเขียนไว้ให้กระจ่างท่านที่สนใจจะศึกษาจะต้องมีพื้นฐานมาก่อน ในเนื้อหาที่เขียนไว้ในหนังสือเล่มนี้ อาจจะไม่สมบูรณ์เท่าที่ควร อีกอย่างคณิตศาสตร์ บางสูตรเป็นคณิตศาสตร์ชั้นสูง ทำให้ยากที่จะทำความเข้าใจ ถ้ามีสิ่งใดผิดพลาดก็ต้องขออภัยท่านผู้อ่านด้วย

ฟูเรียร์ทรานสฟอร์ม เป็นสูตรที่ใช้ในการวิเคราะห์ ความถี่ของสัญญาณอนาลอกซึ่งเป็นสัญญาณต่อเนื่อง ซึ่งสัญญาณทุกๆสัญญาณที่มีคาบเวลา จะประกอบด้วย สัญญาณ ไซเวฟ(sine wave) ที่มีความถี่ต่างๆประกอบกันอยู่ โดยแต่ละความถี่จะมี Amplitude ต่างๆกัน

เมื่อเราต้องการใช้คอมพิวเตอร์ วิเคราะห์ความถี่ของสัญญาณ อนาลอก ซึ่งอยู่ในรูปของ Time Domain ให้เราจะใช้ DFT ทำการประมวลผล ซึ่งจะต้องทำการสุ่มสัญญาณเหล่านั้น เข้ามาให้อยู่ในรูปของสัญญาณดิจิตอลก่อน โดยความถี่ของสัญญาณที่ใช้ทำการสุ่มนั้น จะต้องมีความถี่เป็น 2 เท่า หรือมากกว่า

ความถี่ของสัญญาณอินพุท เพื่อให้ได้ข่าวสารครบถ้วน และถูกต้องในประมวลสัญญาณนั้น จะนำสัญญาณมาทีละช่วงๆไป เราไม่สามารถที่จะกระทำการประมวลได้ทั้งหมดพร้อมกัน ถ้าสัญญาณนั้นมี ความยาวมากๆ เมื่อทำการประมวลสัญญาณเสร็จแล้ว ผลลัพท์ที่ได้จะอยู่ในรูปของความถี่ สัญญาณที่จะนำมาวิเคราะห์นั้นอาจจะเป็น สัญญาณเสียง สัญญาณภาพ สัญญาณจากดาวเทียม หรือ สัญญาณอื่น ๆ ฯลฯ

เมื่อสัญญาณทำการวิเคราะห์แล้ว จะอยู่ในรูปของความถี่ โดยที่เราสามารถที่จะตรวจดูได้ว่า มีความถี่ไหนบ้าง ที่มีอยู่ในสัญญาณนั้น และยังสามารถที่จะนำค่าเหล่านี้ แปลงกลับให้อยู่ในรูปของสัญญาณเดิมได้ และสามารถที่จะตัดค่าความถี่ไหนออกไป หรือเพิ่มความถี่ไหนเข้าไปก่อนที่จะแปลงกลับก็ได้ ซึ่งมีประโยชน์มากสำหรับเป็นตัวกรองสัญญาณ ในขณะที่เรารู้ว่ามีความถี่ไหนบ้างที่อยู่ในสัญญาณ เมื่อนำไปวิเคราะสัญญาณเสียงพูด สามารถที่จะรู้ความหมายของคำนั้นได้ เมื่อทำการเปรียบเทียบกับ Pattern ที่เก็บไว้ก่อน หรือสามารถที่จะสร้างเสียงออกมาได้ซึ่งสำคัญมากสำหรับทางด้านหุ่นยนต์ เพื่อให้การโต้ตอบกับคนเป็นธรรมชาติมากที่สุด หรือ ถ้าเป็นสัญญาณภาพที่ถ่ายมา มีสัญญาณรบกวน เมื่อเรานำสัญญาณภาพนั้น มาทำการประมวลด้วย DFT และเรารู้ความถี่ของสัญญาณนั้น เราสามารถจะตัดออก และทำให้ภาพชัดเจนขึ้นได้ หรือสามารถที่จะวิเคราะห์ได้ว่าภาพนั้นเป็นภาพของอะไร

บทที่ 2.

พีชคณิต และคณิตศาสตร์เบื้องต้น

ก่อนที่เราจะทำการศึกษา DFT นั้นเราจำเป็นจะต้องทำการศึกษาพีชคณิตเกี่ยวกับ แมทริคส์เบื้องต้น ,Prime number, Relativelry Prime, ตัวหารร่วมมาก,ตัวหารร่วมน้อย ,Ring,Polynomaial และสิ่งอื่น ๆ ที่จำเป็นที่จะต้องใช้ใน DFT ALGORITHM เพราะมันเป็นพื้นฐาน ที่จะทำให้เราสามารถที่จะทำความเข้าใจได้ง่าย และเร็ว อัลกอริทัมที่ดี จะต้องมีหลักการที่สามารถที่จะอธิบาย โดยใช้หลักคณิตศาสตร์ได้ และสามารถที่จะแสดงให้เห็นจริงได้

2.1 พีชคณิตแมทริคส์(Matrix Algrebra)

หลักการพีชคณิตสำหรับ แมทริคส์ ในทางทั่วไปจะเรียนเฉพาะเลขจำนวนจริง(Real number) และจำนวนเชิงซ้อน(Complex Number)

แมทริคส์ A ขนาด n x m จะประกอบด้วยตัวประกอบ เท่ากับ n x m ในการจัดลง ARRAY สี่เหลี่ยมผืนผ้า จะมี n แถว และ m คอลัมน์

$$A = \begin{bmatrix} a11 & a12 & \dots & a1m \\ a21 & a22 & \dots & a2m \\ . & . & & . \\ . & . & & . \\ an1 & an2 & \dots & anm \end{bmatrix} = [a_{ij}]$$

ถ้า n = m เรียกแมทริคส์นี้ว่า Square Matrix แมทริคส์ขนาด n x m

แมทริคส์ 2 อัน คือ A และ B สามารถที่จะบวกกัน ได้โดยใช้กฎการบวก

$$A + B = \begin{bmatrix} a11+b11 & a12+b12 & \dots & a1m+b1m \\ a21+b21 & a22+b22 & \dots & a2m+b2m \\ . & . & & \\ . & . & & \\ an1+bn1 & an2+bn2 & \dots & anm+bnm \end{bmatrix}$$

แมทริคส์ A มีขนาด n x m สามารถที่จะคูณด้วย β ได้ ด้วยกฎการคูณแบบ Scalar

$$\beta A = \begin{bmatrix} \beta a11 & \beta a12 & \dots & \beta a1m \\ \beta a21 & \beta a22 & \dots & \beta a2m \\ \vdots & & & \vdots \\ \beta an1 & \beta an2 & \dots & \beta anm \end{bmatrix}$$

แมทริคส์X A มีขนาด 1 x n และ แมทริคส์ B มีขนาด n x m สามารถที่จะคูณกัน และได้ แมทริคส์ C ขนาด 1 x m โดยกฎการคูณแมทริกส์

$$C_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$i = 1, \dots, l$$

$$j = 1, \dots, m$$

ในแมทริคส์ A จำนวนคอลัมน์ และจำนวนแถวเท่ากัน เรียกว่า MAIN DIAGONAL และIDENTITY MATRIX ซึ่งแสดงโดย I คือ MATRIX มีขนาด n x n ซึ่งมีค่าที่ MAIN DIAGONAL เท่ากับ 1. ส่วนค่าอื่นๆ จะเท่ากับศูนย์

EXCHANGE MATRIX แสดงโดย J เป็นแมทริคส์มีค่าเป็น 1 ทุกค่าของ ANTIDIAGONAL ส่วนค่าอื่นๆ จะเท่ากับศูนย์หมด(J จะมีค่า เท่ากับ n+1-i) สังเกตุว่า $J^2 = I$

ตัวอย่าง

IDENTITY MATRIX ขนาด 3 x3 และ EXCHANGE MATRIX ขนาด 3 x 3 คือ

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad J = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

TRANSPOSE ของแมทริคส์ A ขนาด n x m ก็คือแมทริคส์ m x n แสดงโดย A^T ซึ่ง $a_{ij}^T = a_{ji}$ ซึ่งคือแถวของ A^T คือคอลัมน์ของ A และคอลัมน์ของ A^T ก็คือแถวของ A มันง่ายที่จะแสดงว่า C = AB แล้ว $C^T = B^T A^T$

INVERSE ของ SQUARE MATRIX A ก็คือ SQUARE MATRIX A^{-1}ถ้า $A^{-1}A = AA^{-1} = I$ MATRIX ที่มี INVERSE เรียกว่า NONSINGULAR MATRIX ส่วน MATRIX ที่ไม่มี INVERSE เรียกว่า SINGULAR MATRIX ให้ C = AB ถ้า A และBไม่มี INVERSE แล้ว C ก็ไม่มี INVERSE ด้วยแต่ถ้า A และ B มี INVERSE แล้ว $C^{-1} = B^{-1}A^{-1}$ เพราะว่า $(B^{-1}A^{-1})C = I = C(B^{-1}A^{-1})$

2.2 RING

คือ SET ของตัวกระทำ(OPERATOR) 2 ตัว คือการบวก(ADDITION) และการคูณ(MULTIPLE)

2.3. FIELD F

คือ SET ที่ประกอบไปด้วยอย่างน้อย 2 ตัวประกอบ กระทำกันโดย 2 ตัวกระทำ คือการบวก และการคูณ FIELD ที่รู้จักกันดี คือ

1. R : คือ SET ของจำนวนจริง (REAL)
2. C : คือ SET ของจำนวนเชิงซ้อน (COMPLEX NUMBER)
3. Q : คือ SET ของเลขเศษส่วน (RETIONAL NUMBER)

2.4 PRIME NUMBER

จำนวนเต็มบวก P ที่มีค่ามากกว่า 1 ที่สามารถหารได้เฉพาะตัวมันเอง หรือ 1 เท่านั้น เราเรียก P ว่า PRIME INTEGER ค่า PRIME ที่น้อยที่สุดมี เช่น 2,3,5,7,11,13,...;

ค่า 1ไม่ใช่ PRIME NUMBER ค่าจำนวนเต็มบวกที่ไม่ใช่ PRIME เราเรียกว่า COMPOSITE

ตัวหารร่วมมาก(GREATEST COMMON DIVISOR) ของจำนวนเต็ม 2 จำนวน คือ r และ s แสดงได้โดย GCD(r,s) ก็คือ ค่าจำนวนเต็มบวกที่มากที่สุดที่หารทั้ง 2 ค่าได้ลงตัว

ตัวคูณร่วมน้อย(LEAST COMMON MULTIPLE)ของ 2 จำนวนเต็มคือ r และ s แสดงโดย LCM(r,s) คือค่าจำนวนเต็มที่น้อยที่สุดที่ทั้ง 2 ค่านี้หารได้ลงตัว

จำนวนเต็ม 2 ค่า จะเรียกว่า RELATIVELY PRIME ก็ต่อเมื่อค่าหาร

ร่วมมากทั้ง 2 มีค่าเป็น 1

ทฤษฎี(DIVISION ALGORITHM) สำหรับทุกค่าของจำนวนเต็ม c และจำนวนเต็มบวก d และมี Q เป็นผลหารและ s เป็นค่า เศษจะได้ว่า

$$c = dQ + s \qquad \text{ซึ่ง } 0 < s < d$$

ผลหารสามารถแสดงได้โดย

$$Q = \lfloor c/d \rfloor$$

โดยทั่วไปเราจะให้ความสนใจเศษ มากกว่าผลหาร s และ c มีเศษเดียวกันเมื่อถูกหารด้วย d เราจะเขียนได้ว่า

$$s \equiv c \pmod{d}$$

ค่าที่เหลือจากการหารเราจะเขียนได้ว่า

$$s = R_d[c]$$

ซึ่งอ่านว่า s คือ เศษของ c เมื่อถูกหารด้วย d หรือ s คือ ค่าที่มีอยู่เมื่อ c MODULO d

ทฤษฎี. การกระทำกับตัวคงที่

1. $R_d[a+b] = R_d[\, R_d[a] + R_d[b] \,]$
2. $R_d[a.b] = R_d[\, R_d[a] \cdot R_d[b] \,]$

การหาตัวหารร่วมมากของจำนวนเต็มบวก 2 จำนวนใดๆ สามารถที่จะหาได้ โดยบระยุกต์ใช้ ALGORITHM การหาร ซึ่งขบวนการ(PROCEDURE) นี้เรียกว่า EUCLIDEAN ALGORITHM

สมมุติว่า $t < s$ EUCLIDEAN ALGORITHM จะได้ดังนี้

$$s = Q^{(1)}t + t^{(1)}$$
$$t = Q^{(2)}t^{(1)} + t^{(2)}$$
$$t^{(1)} = Q^{(3)}t^{(2)} + t^{(3)}$$
$$\vdots$$
$$t^{(n-2)} = Q^{(n)}t^{(n-1)} + t^{(n)}$$
$$t^{(n-1)} = Q^{(n+1)}t^{(n)}$$

ซึ่งขบวนการนี้จะหยุดได้เมื่อเศษเท่ากับศูนย์ ค่าเศษสุดท้ายที่ไม่เท่ากับศูนย์ $t^{(n)}$คือค่า หารร่วมมาก เสต็ปการทำงานของ EUCLIDEAN ALGORITHM สามารถที่จะแสดงง่ายๆในแมทริคส์ได้เป็น

$$\begin{bmatrix} s^{(r)} \\ t^{(r)} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)} \end{bmatrix} \begin{bmatrix} s^{(r-1)} \\ t^{(r-1)} \end{bmatrix}$$

ทฤษฎี. EUCLIDEAN ALGORITHM ให้ s และ t เป็นจำนวนเต็มบวก 2 ค่า ซึ่ง

$s > t$ ให้ $s^{(0)} = s$ และ $t^{(0)} = t$ เมื่อ $r = 1,\ldots,n$:

$$Q^{(r)} = [s^{(r-1)}/t^{(r-1)}]$$

$$\begin{bmatrix} s^{(r)} \\ t^{(r)} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)} \end{bmatrix} \begin{bmatrix} s^{(r-1)} \\ t^{(r-1)} \end{bmatrix}$$

และจะได้ว่า

$$s^{(n)} = GCD[s,t]$$

ซึ่ง n คือจำนวนเต็มที่น้อยที่สุดที่ทำให้ $t^{(n)} = 0$

จะเห็นได้ว่า $t^{(r+1)} < t^{(r)}$ และค่าของเศษไม่เป็นลบ เพราะฉะนั้นจะต้องมีสักค่าของ n ที่จะทำให้ $t^{(n)} = 0$ และเรารู้ว่ามันต้องจบแน่นอน

ตามการ INVERSE MATRIX เราจะได้ว่า

$$\begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)} \end{bmatrix}^{-1} = \begin{bmatrix} Q^{(r)} & 1 \\ 1 & 0 \end{bmatrix}$$

เพราะฉะนั้นจะได้ว่า

$$\begin{bmatrix} s \\ t \end{bmatrix} = \left[\prod_{l=1}^{n} \begin{bmatrix} Q^{(l)} & 1 \\ 1 & 0 \end{bmatrix}\right] \begin{bmatrix} s^{(n)} \\ 0 \end{bmatrix}$$

เมื่อ $s^{(n)}$ จะต้องหาได้ทั้ง s และ t เราจะได้ว่า

$$\begin{bmatrix} s^{(n)} \\ 0 \end{bmatrix} = \left[\prod_{l=n}^{1} \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(l)} \end{bmatrix}\right] \begin{bmatrix} s \\ t \end{bmatrix}$$

จะได้ว่า

$$s^{(n)} = GCD[s,t]$$

2.5 POLYNOPMIAL

จะแสดงได้ด้วย

$$f(x) = f_n x^n + f_{n-1} x^{n-1} + \ldots + f_1 x + f_0$$

$$= \sum_{i=0}^{n} f_i x^i$$

x คือ indeternate

f_i คือ ตัวประกอบ (coefficients)

ZERO POLYNOMIAL คือ $f(x) = 0$

DEGREE ของ POLYNOMIAL f(x) แสดงโดย deg f(x) คือค่าตัวชี้ที่มากที่สุด ที่มีสัมประสิทธิ์ไม่เป็นศูนย์ กำลังของ NON - ZERO POLYNOMIAL จะจำกัดเสมอ ส่วน Degree ของ ZERO POLYNOMIAL จะเป็น NEGATIVE INFINITY(ลบ อินฟินิตี้)

MONIC POLYNOMIAL คือ POLYNOMIAL ที่สัมประสิทธิ์ของ f(n) ที่มีค่าตัวชี้มากที่สุด มีค่าเท่ากับ 1 POLYNOMIAL 2 ชุดจะเท่ากันเมื่อสัมประสิทธิ์ของ f(i) ทุกๆตัวมีค่าเท่ากัน

ผลบวกของ 2 POLYNOMIAL ใดๆจะเท่ากับ

$$f(x) + g(x) = \sum_{i=0}^{\infty} (f_i + g_i) x^i.$$

กำลังของผลคูณของ Polynomial 2 อัน เท่ากับผลบวกของกำลังของตัวประกอบ 2 ตัว ถ้า f(x) และ g(x) ไม่เท่ากับศูนย์แล้ว f(x).g(x)จะไม่เท่ากับศูนย์เพราะว่า deg p(x) = $-\infty$ ถ้าเพียงแต่ p(x) เป็นศูนย์.

ภายใน Ring ของ Polynomial การลบกระทำได้เสมอ เราเขียน r(x)|s(x) เราจะอ่านได้ว่า Polynomial s(x) ได้กระทำการหารโดย Polynomial r(x) หรือr(x) เป็นตัวประกอบหนึ่งของ s(x) ถ้ามี Polynomial a(x) แล้ว r(x).a(x)=s(x) ถ้า p(x) เป็น Non-Zero Polynomial p(x) ซึ่งสามารถหารเฉพาะp(x) หรือ α เท่านั้น เมื่อ α เป็น Field Element หนึ่งที่กำหนดขึ้นเอง เรียกว่า Polynomial ที่แยกตัวประกอบไม่ได้(Irreducible Polynomial) และMonic Irreducible Polynomialเรียกว่า Prime Polynomial เราจะเรียก Polynomial หนึ่งว่าเป็น Prime Polynomial นั้นจำเป็นที่จะรู้ Field ภายในPolynomial

ซึ่งได้กล่าวมาแล้ว Polynomial $p(x) = x^4 - 2$ เป็น Prime Polynomial บน Field ของเศษส่วนแต่ไม่เป็น Prime บน Field ที่เป็นจำนวนจริงบน Field ที่เป็นจำนวนจริง $p(x) = (x^2 - \sqrt{2})(x^2 + \sqrt{2})$ เป็นการคูณของ Polynomial 2 จำนวน. บน Complex Field Polynomial เหล่านี้ ไม่เป็น Prime.

เมื่อใดก็ตามที่ r(x)สามารถกระทำทั้งการหาร s(x) และหารได้ด้วย s(x) แล้ว $r(x) = \alpha s(x)$ เมื่อ α เป็น Element หนึ่งของ Field F เป็นที่ยอมรับตามนี้ว่า ต้องมีการมีอยู่ของ Polynomial a(x) และ b(x)ดังนั้น $r(x) = s(x).a(x)$ และ $s(x) = r(x).b(x)$ เพราะฉะนั้น $r(x) = r(x).a(x).b(x)$แต่กำลังของทางด้านขวามือเป็นผลบวกของกำลังของr(x), b(x) และ a(x) เพราะว่ากำลังทางด้านขวามือต้องเท่ากับกำลังทางด้านซ้ายมือ a(x) และ b(x) ต้องมีกำลังเป็น ศูนย์นั่นคือมันเป็น Scalar

ตัวหารร่วมมากของ Polynomial 2 จำนวน คือ r(x) และ s(x) เขียนย่อเป็น GCD[(r(x),s(x)] เป็น Monic Polynomial ของกำลังสูงสุดที่กระทำการหาร r(x) และ s(x) ถ้าตัวหารร่วมมากของ 2 Polynomialเป็น 1 แล้ว เราจะพูดได้ว่ามันเป็น Relatively Prime

ตัวคูณร่วมน้อยของ s(x)และ r(x) เขียนย่อเป็น LCM[r(x),s(x)] เป็น Monic Polynomial ของกำลังต่ำสุด สามารถกระทำการหารได้โดย r(x) และs(x) เราจะเห็นว่าตัวหารร่วมมาก และตัวคูณร่วมน้อยของ r(x) และ s(x) มีค่าเดียว

2.6 CHINESE REMAINDER THEORY

เป็นไปได้ที่จะพิจารณาความเป็นหนึ่งเดียวของจำนวนเต็มบวก ที่กำหนดให้เพียง MODULI ของแต่ละกลุ่มของจำนวนเต็ม โดยมีเงื่อนไขว่าจำนวนเต็มค่านั้นจะต้องน้อยกว่าผลคูณของ MODULI ซึ่งเป็นที่รู้จักกันในหมู่ของ ชาวจีนในสมัยโบราณ มีชื่อว่า Chinese Remainder

THEOREM.

Chinese Remainder ได้รวบรวมไว้ในรูป 2.1 พิสูจน์เป็น 2 ส่วน ส่วนแรกเราจะพิสูจน์ คือความเป็นหนึ่ง (Uniqueness) ของคำตอบ และส่วนที่ 2 เราจะพิสูจน์การมีอยู่ของคำตอบ โดยมีกระบวนความ(Procedure) ในการหามัน

ก่อนที่เราจะทำความเข้าใจทฤษฎี เราจะกำหนดตัวอย่างง่ายๆ. ดังจะแสดงดังต่อไปนี้เลือก Modoli $m_0 = 3, m_1 = 4$, และ $m_2 = 5$ และกำหนด $M = m_0.m_1.m_2 = 60$. กำหนดจำนวนเต็ม c อยู่ในช่วง $0 < c < 60$ กำหนด $c_i = R_m[c]$. Chinease Remainder บอกว่ามีอยู่ค่าเดียว ระหว่าง 60 ค่า ที่ map, vector ของ residuces(c_0,c_1,c_2). สมมติว่า $c_0=2$,$C_1=1$ และ $c_2=2$. มี 3 เงื่อนไขที่เป็นไปได้

$c \in \{2,5,8,11,14,17,20,23,26,...\}$

$c \in \{1,5,9,13,17,21,25,29,33,...\}$

$c \in \{2,7,12,17,22,27,32,37,...\}$

คำตอบที่เป็นหนึ่งเดียว(Unique Solution) คือ 17. หลังจากนี้เราจะแสดง Algorithm ที่หาค่า c จาก Residues ของมัน

THEOREM 1. กำหนดให้ Set ของจำนวนเต็ม $m_0,m_1,m_2,m_3,...,m_k$ ที่เป็น Relatively Prime และ Set ของจำนวนเต็ม $c_0,c_1,c_2,c_3,...,c_k$ กับ $c_i < m_i$ แล้วระบบของสมการ (System of Equation)

$$c_i = c \pmod{m_i} \quad i=0,1,...,k$$

จะมีคำตอบที่มากที่สุดเพียงค่าเดียวสำหรับค่าของ c ในช่วง

Direct Equations

$C_i = R_{m_i}[c]$ $i=0,...k$ เมื่อ m_i เป็น Relatively prime

Inverse Equations

$$C = \sum_{i=0}^{k} c_i N_i M_i \pmod{M}$$

เมื่อ $M = \prod_{i=0}^{k} m_i$ $\qquad M_i = M/m_i$

และ N_i เป็นคำตอบของ

$$N_i M_i + n_i m_i = 1$$

รูปที่ 2.1. Chinese Remainder Theorem.

$$0 \leq c < \prod_{i=0}^{k} m_i$$

PROOF สมมุติว่า c และ c' เป็นคำตอบในช่วงนี้แล้วจะได้ว่า

$$c = Q_i m_i + c_i$$
$$c' = Q'_i m_i + c_i$$

เพราะว่า M_i เป็น Relative Prime แต่ $c - c'$ จะอยู่ในช่วง

$$-\prod_{i=0}^{k} m_i < (c - c') < \prod_{i=0}^{k} m_i$$

เป็นไปได้อย่างเดียวเท่านั้นที่ค่า $c - c' = 0$ นั่นคือ $c = c'$ มีวิธีที่ง่ายที่จะหาคำตอบของระบบ (Congruences) Theorem 1. บนพื้นฐานของ Euclidean Algorithm เขาแสดงให้เห็นว่าแต่ละ s และ t มีการมีอยู่ของจำนวนเต็ม a และ b ที่ได้จาก

$$GCD(s,t) = as + bt$$

เพราะฉะนั้นจาก set ของจำนวนเต็มที่เป็น relatively prime $m_0, m_1, \ldots, m_k$ ใช้เป็น moduli กำหนดให้

$$M = \prod_{r=0}^{k} m_r$$

และ $M_i = M / m_i$ แล้ว $GCD(M_i, m_i) = 1$ ดังนั้นแต่ละ i มีการมีอยู่ของจำนวนเต็ม M_i และ n_i กับ $N_i M_i + n_i m_i = 1$, $i = 0, \ldots, k$ เราพร้อมที่จะพิสูจน์ Theorem ต่อไป

THOREM 2. กำหนดให้

$$M = \prod_{r=0}^{k} m_r$$

เป็นการคูณของเลขจำนวนเต็มที่เป็น relatively prime และกำหนดให้ $M_i = M/m_i$ ในแต่ละ i กำหนด N_i หาได้จาก $N_i M_i + n_i m_i = 1$ แล้วจะแสดงได้ดังนี้

$$c_i = c \pmod{m_i} , \quad i = 0, \ldots, k$$

จะหาคำตอบได้โดยใช้

$$C = \sum_{i=0}^{k} c_i N_i M_i \mod(M)$$

PROOF เราเพียงต้องการแสดงว่า c นี้คือคำตอบ เพราะว่าเรารู้ว่าคำตอบมีเพียงค่าเดียวแต่สำหรับ c นี้

$$C = \sum_{r=0}^{k} c_r N_r M_r = c_i N_i M_i \ (\mod(m_i),$$

เพราะว่า m_i หาร M_r ถ้า r ไม่เท่ากับ i สุดท้ายเพราะว่า

$$N_i M_i + n_i m_i = 1$$

และเรามี

$N_i M_i = 1 \quad (\mod m_i)$ และ $c = c_i \ (\mod m_i)$

ตัวอย่าง แสดงการใช้ THEOREM 2. ซึ่งจะได้ $M = 60$, $M_o = 20$, $M_1 = 15$, $M_2 = 12$ จะได้ว่า

$$1 = (-1)M_o + 7m_o$$
$$1 = (-1)M_1 + 4m_1$$
$$1 = (-1)M_2 + 5m_2$$

เราสามารถคำนวณจาก Euclidean Algorithm หรือจะเขียนให้เห็นได้ง่ายๆได้ ดังนี้

$$N_o M_o = -20 \ , \ N_1 M_1 = -15 \ , N_1 M_1 = -24$$

และกระทำ Inverse ได้ว่า

$$c = -20c_o - 15c_1 - 24c_2 \quad (\mod 60)$$

ถ้าให้ $c_o = 2$ $c_1 = 1$ และ $c_2 = 2$

$$c = -103 \mod (60)$$
$$= 17$$

ซึ่งเราได้เห็นตามตัวอย่างมาแล้ว

บนพื้นฐานของ Chinese Remainder Theorem , มันสามารถใช้กับการคูณได้ สมมติว่าเราต้องการที่จะกระทำ การคูณ

$$c = a.b$$

สำหรับแต่ละค่า i,กำหนด $a_i = R_{m_i}[a], b = R_{m_i}[b]$ และ $c = R_{m_i}[c]$. แล้วสำหรับแต่ละค่า $i = 0,1,\ldots,k$

$$c_i = a_i.b_i \ (\mod m_i)$$

มันง่ายมากที่จะทำการคำนวนเพราะว่า a,b เป็นจำนวนเต็มค่าน้อยๆ, ถ้าใช้กับการบวกจะแสดงได้ดังต่อไปนี้

$$c = a + b$$

แล้วสำหรับแต่ละค่า $i = 0,1,\ldots,k$

$$c_i = a_i + b_i \pmod{m_i}$$

ไม่ว่าจะเป็นกรณีใดๆ ค่าคำตอบสุดท้ายของ c สามารถที่จะหาค่าของมันได้จาก Residues ของมัน โดยใช้ Chinese Remainder Theorem.

ทีนี้เราจะทำการ Residues,ค่าจำนวนเต็นค่ามากๆ จะแยกออกเป็นค่าเล็กๆ ที่ง่ายต่อการบวก,การลบ และ การคูณ. ตามที่ใช้ในการคำนวณการแทนนี้จะจัดโดย Alternative System ทางคณิตศาสตร์ได้ โดยการ MAP ของข้อมูลโดยการใช้ Residue เมื่อทำการ MAP เข้ามาแล้วสามารถที่จะใช้ Residue ทำการ MAP กลับคืนค่าเดิมได้

ในเทอมของ Polynomial ที่นำมาใช้กับ Chinese Remainder Theory ซึ่งสรุปไว้ในรูปที่ 2.2 มีลักษณะคล้ายๆกับเลขจำนวนเต็ม

THEOREM 3. กำหนด Set ของ Polymomials $m^{(0)}(x)$, $m^{(1)}(x),\ldots,m^{(k)}(x)$ เป็น Relatively Prime และ คู่กับ Set ของ Polynomials $c^{(0)}(x),c^{(1)}(x),\ldots,c^{(k)}(x)$ ซึ่งมี $\text{Deg } c^{(i)}(x) < \text{Deg } m^{(i)}(x)$ แล้ว สมการสามารถแสดงได้ดังต่อไปนี้

$$c^{(i)}(x) = c(x) \pmod{m^{(i)}(x)} \quad i = 0,\ldots,k$$

มีคำตอบที่มากที่สุดเพียงค่าเดียวสำหรับ c(x)

$$\deg c(x) < \sum_{i=0}^{k} \deg m^{(i)}(x)$$

PROOF สมมุติว่า c(x) และ c'(x) เป็นคำตอบของสมการ

$$c(x) = Q^{(i)}(x)m^{(i)}(x) + c^{(i)}(x)$$

$$c'(x) = Q'^{(i)}(x)m^{(i)}(x) + c^{(i)}(x)$$

ดังนั้น $c(x)-c'(x)$ เป็นผลคูณของ $m_{(i)}$ ในแต่ละค่าของ i นั่นคือ

$c(x) - c'(x)$ เป็นผลคูณของ

$$\prod_{i=0}^{k} m^{(i)}(x)$$

และกำลังของ $c(x) - c'(x)$ จะต้องน้อยกว่ากำลังของ

$$\prod_{i=0}^{k} m^{(i)}(x)$$

นั่นคือ $c(x) - c'(x) = 0$

ระบบของ Congruences สามารถที่จะคำตอบได้ ในทำนองเดียวกันกับเลขจำนวนเต็ม ในรูปของ Polynomial ให้ $s(x)$ และ $t(x)$ มีการมีอยู่ของ Polynomial $a(x)$ และ $b(x)$ ซึ่งจะได้ว่า

$GCD[s(x),t(x)] = a(x)s(x) + b(x)t(x)$ เมื่อให้

$$M(x) = \prod_{i=0}^{k} m^{(i)}(x)$$

และ $M^{(i)}(x) = M(x)/m^{(i)}(x)$ แล้ว

$GCD\ [\ M^{(i)}(x)\ ,\ m^{(i)}(x)\ ] = 1$

ให้ $N^{(i)}(x)$ และ $n^{(i)}(x)$ ได้ว่า

$N^{(i)}(x).M^{(i)}(x) + n^{(i)}(x).m^{(i)}(x) = 1$

THOEREM 4. กำหนดให้

$$M(x) = \prod_{r=0}^{k} m^{(r)}(x)$$

เป็นผลคูณของ Relative Prime Polynomials

กำหนดให้ $M^{(i)}(x) = M(x) / m^{(i)}(x)$ และ $N^{(i)}(x)$ ที่ต้องการคือ $N^{(i)}(x).M^{(i)}(x) + n^{(i)}(x).m^{(i)}(x) = 1$ แล้วระบบของ Congruences คือ

$c^{(i)}(x) = c(x) \pmod{m^{(i)}(x)}, \quad i = 0,1,\ldots,k$

หาคำตอบได้โดย

$$c(x) = \sum_{I=0}^{k} c^{(i)}(x).N^{(i)}(x).M^{(i)}(x) \pmod{M(x)}$$

Direct Equations

$$C^{(i)}(x) = R_{m^{(i)}(x)}[c(x)]$$

$i = 0,\ldots k$ เมื่อ $m^{(i)}(x)$ เป็น relatively prime

Inverse Equations

$$C(x) = \sum_{i=0}^{k} c^{(i)}(x)N^{(i)}(x)M^{(i)}(x) \mod(M(x))$$

เมื่อ $$M(x) = \prod_{i=0}^{k} m^{(i)}(x)$$

$$M^{(i)}(x) = M(x)/m^{(i)}(x)$$

และ $N^{(i)}(x)$ เป็นคำตอบของ

$$N^{(i)}(x)M^{(i)}(x) + n^{(i)}(x)m^{(i)}(x) = 1$$

รูปที่ 2.2. Chinese Remainder theorem สำหรับ Polynomial

PROOF เราจำเป็นที่จำเป็นที่จะต้องแสดงว่า $c(x)$ นี้ ให้ Congruence ทุกๆอัน ในระบบของ Congruences

$$c(x) = c^{(i)}(x).N^{(i)}(x).M^{(i)}(x) \pmod{m^{(i)}(x)}$$

เพราะว่า $M^{(r)}(x)$ มี $m^{(i)}(x)$ เป็น Factor ถ้า r ไม่เท่ากับ i.
เพราะว่า

$$N^{(i)}(x).M^{(i)}(x) + n^{(i)}(x).m^{(i)}(x) = 1$$

เราจะมี

$$N^{(i)}(x).M^{(i)}(x) = 1 \pmod{m^{(i)}(x)}$$

และ

$$c(x) = c^{(i)}(x) = 1 \pmod{m^{(i)}(x)}$$

ที่ทำให้ข้อพิสูจน์ของ Theorem สมบูรณ์

บทที่ 3.

FFT ALGORITHM

ในบทนี้ เราจะกล่าวถึง Algorithm ที่มีอยู่แล้ว และมีประสิทธิภาพที่ได้แก้ปัญหาทำให้ผลคูณ -และผลบวกลดลง- จริงๆแล้ว Algorithm ที่ได้คิดค้นขึ้นมานั้น พยายามที่จะลดผลคูณให้มากที่สุด แม้ว่าจะต้องเพิ่มผลบวกขึ้นก็ตาม เพราะเราถือว่าการคูณ และการบวกตัวเลขที่มีขนาดเท่ากันแล้ว นั้นการคูณจะยุ่งยากกว่า และใช้เวลานานกว่า โดยเฉพาะเมื่อก่อนนั้นคำสั่งการคูณใน CPU ไม่มีเลยในการคูณแต่ละครั้งนั้น จะต้องมี Algorithm ในการคูณโดยเฉพาะ แต่ถึงอย่างไรแม้ว่าปัจจุบันนี้ CPU จะสามารถจะกระทำการคูณ ได้ในคำสั่งเดียวก็ตาม แต่เป็นเพียง 8 BIT หรือ 16 BIT เท่านั้น สำหรับการคำนวณ DFT แล้วจะใช้ FLOATING POINT NUMBER ซึ่งใช้ 4 BYTE หรือ 2 WORD ในการคำนวณ

สำหรับ Algorithm ที่เราได้ทำการศึกษานั้นมีดังนี้ Cooley-Tukey, Algorithm,Good - Thomas Algorithm และ Winograd Algorithm

3.1 THE COOLEY-TUKEY FAST FOURIER TRANSFORM

FOURIER TRANSFORM ของ VECTOR V คือ

$$V_k = \sum_{i=0}^{n-1} W^{ik} v_i$$

จากสูตรที่เขียนมาข้างบน ต้องการจำนวนการคูณถึง N^2 ครั้ง และจำนวนการบวก N (N-1) ครั้งด้วย ถ้า N มีตัวประกอบ สามารถที่จะทำการทรานสฟอร์ม FOURIER จาก ONE-DIMENTION เป็น TWO-DIMENSION ได้ ซึ่งมีประสิทธิภาพมากกว่า แต่เพิ่มความยุ่งยากที่จะทำความเข้าใจ

Algorithm นี้ เรียกว่า FAST FOURIER TRANSFORM(FFT)ในรูปที่ 3.1 ได้รวบรวมโครงสร้างของ COOLEY-TUKEY FFT ALGORITHM ที่เราจะได้ศึกษาต่อไปนี้

COOLEY - TUKEY FFT (1965)

$n = n'n"$

$i = i' + n'i"$

$k = n"k' + k"$

$i' = 0,\ldots,n'-1$

$i" = 0,\ldots,n"-1$

$k' = 0,\ldots,n'-1$

$k" = 0,\ldots,n"-1$

$$V_{k',k"} = \sum_{i'=0}^{n'-1} \beta^{i'k'} \left[w^{i'k"} \sum_{i"=0}^{n"-1} \delta^{i"k"} V_{i',i"} \right]$$

รูปที่ 3.1 Cooley - Tukey FFT

เพื่อที่จะพิสูจน์ Cooley - Tukey Algorithm สมมติให้ $n = n'n"$

$i = i' + n'i"$

$k = n"k' + k"$

$i' = 0,\ldots,n'-1$

$i" = 0,\ldots,n"-1$

$k' = 0,\ldots,n'-1$

$k" = 0,\ldots,n"-1$

$$V_{n"k'+k"} = \sum_{i'=0}^{n'-1} \sum_{i"=0}^{n"-1} w^{(i'+n'i")(n"k'+k")} V_{i'+n'i"}$$

$$i.k = i'.n".k' + i'.k" + n'.i".n".k' + n'.i".k"$$

เนื่องจาก

$$w^{(n'n")(i"k')} = 1$$

ดังนั้น

$$w^{ik} = w^{(i'n"k')} \cdot w^{(n'i"k")} \cdot w^{(i'k")}$$

ให้ $\beta = w^{n"}$, $\delta = w^{n'}$

เพราะฉะนั้น

$$V_{k',k''} = \sum_{i'=0}^{n'-1} \beta^{i'k'} \left[w^{i'k''} \sum_{i''=0}^{n''-1} \delta^{i''k''} v_{i',i''} \right]$$

ถึงแม้ว่ารูปแบบนี้มันยากที่จะทำความเข้าใจกว่า Algorithm เดิมของมันแต่จำนวน การคูณ และการบวก จ ะน้อยลงกว่าเดิมมาก คือมันต้องการจำนวนการคูณเชิงซ้อนเท่ากับ n (n' + n" + 1) และต้องการจำนวนการบวกเชิงซ้อนเท่ากับ n(n' + n" -2)

COOLEY - TUKEY FFT สามารถที่จะแสดงให้เห็นจากการ MAP จาก ONE - DIMENSION ไปเป็น TWO - DIMENSION ดังได้แสดงในรูปที่ 3.2 สำหรับค่า n ที่มีค่า เท่ากับ 15 และ 21 การคำนวณจะประกอบไปด้วย n" POINT ในแต่ละคอลัมน์ เมื่อเสร็จแล้วจะนำแต่ละค่ามาทำการ MULTIPLY ด้วยค่า $w^{i'k''}$ ตามด้วยการ ทรานสฟอร์ม n' POINT ในแต่ละแถว

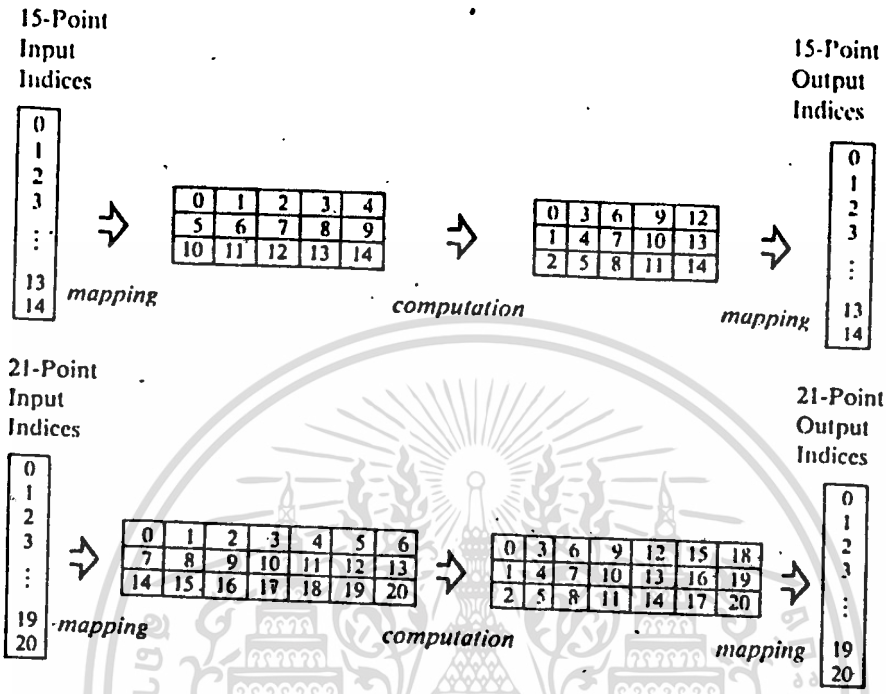
การนับจำนวนของ การคูณเชิงซ้อน และการบวกเชิงซ้อนเมื่อ Input vector(v) เป็นค่าเชิงซ้อน เมื่อการคำนวณในแต่ละครั้งของ เทอมใน และเทอมนอกใช้การคูณเชิงซ้อนเท่ากับ $(n'')^2$ และ $(n')^2$ ตามลำดับ และมีการคำนวณการบวกเลขเชิงซ้อนเท่ากับ n"(n"-1) และ n'(n'-1)ครั้ง ตามลำดับ และการคำนวณเทอมในมีการคำนวณ n' ครั้ง ส่วนเทอมนอกมีการคำนวณ n" ครั้ง นอกจากนี้ยังมีการคูณเพิ่มขึ้นมาอีก n'n" ครั้ง โดยการคูณแต่ละค่าด้วย $w^{i'k''}$ เพราะฉะนั้นจะมีการคูณเชิงซ้อน $M_c(n)$ และการบวกเชิงซ้อน $A_c(n)$ เป็นดังนี้

$$M_c(n) = n'(n'')^2 + n''(n')^2 + n'n''$$
$$= n(n' + n'' + 1)$$
$$A_c(n) = n'n''(n''-1) + n''n'(n'-1)$$
$$= n(n' + n'' - 2)$$

ในส่วนที่ค่าของ i' หรือ k" มีค่าเป็นศูนย์ นั้นเราก็คิดผลคูณด้วยแต่จริงๆแล้วเราไม่ต้องทำการคูณเพราะเท่ากับคูณกับ 1 แล้วเราจะได้จำนวนผลคูณใหม่เป็น $M_c(n) = n(n' + n'') + (n' - 1)(n''-1)$

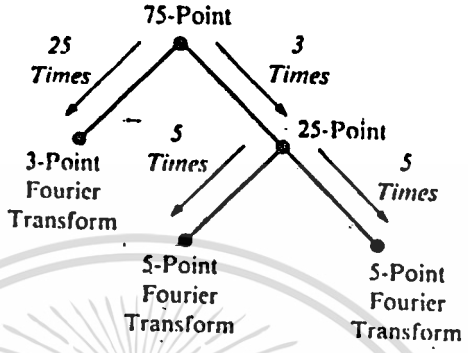
$$= (n - 1)(n' + n'') + (n + 1)$$

การคำนวนเทอมนอก และเทอมในที่เป็น Short Algorithm นั้นสามารถที่จะนำไปคำนวณด้วย Algorithm อื่นได้ไม่เฉพาะของ Cooley-Tukey Algorithm เท่านั้น



รูปที่ 3.2 แสดงการ MAP ค่า ของ INPUT และ OUTPUT.

ในขณะเดียวกันค่าของ n' และ n" ถ้าสามารถที่จะทำการแบ่งย่อยลงไปอีก สามารถที่จะแบ่งย่อย และนำมาคำนวณโดย COOLEY - TUKEY FFT ได้ในรูปที่ 3.3 ได้แสดงแนวทางหนึ่งในหลายๆแนวทางในการคำนวณ ฟูเรียร์มีขนาด 75 Point โดยการคำนวนจะทำเป็นส่วนโปรแกรมย่อยดังแสดงในรูปที่ 3.4.



รูปที่ 3.3โครงสร้างการคำนวณ COOLEY - TUKEY FFT ขนาด 75 POINT

FFT เป็น ALGORITHM สำหรับคำนวณ DFT ที่สามารถแก้ปัญหาการคำนวณของ DFT (เมื่อ INPUT DATA เป็นREAL DATA อย่างเดียว)ที่ต้องการจำนวนผลคูณ และผลบวกมากถึง N^2 และ N(N-1) ตามลำดับ ลดเทอมผลคูณและเทอมผลบวกจากการคำนวณโดยตรงมีหลาย ALGORITHM และทั้งการปรับปรุงพิมพ์ออกเผยแพร่ และทำเป็น SOFT-WARE PACKAGE ออกมามากมาย

เพื่อให้เข้าใจการทำงาน และคุณสมบัติของ ALGORITHM ต่างๆพร้อมทั้ง TECHNIQUE ที่มีอยู่เพื่อใช้งานได้อย่างมีประสิทธิภาพ

สูตร DFT ในรูปของ COMPLEX NUMBER แสดงได้ดังนี้

$$Xn = \sum_{i=0}^{N-1} x_i \, w^{in} \qquad (3.2)$$

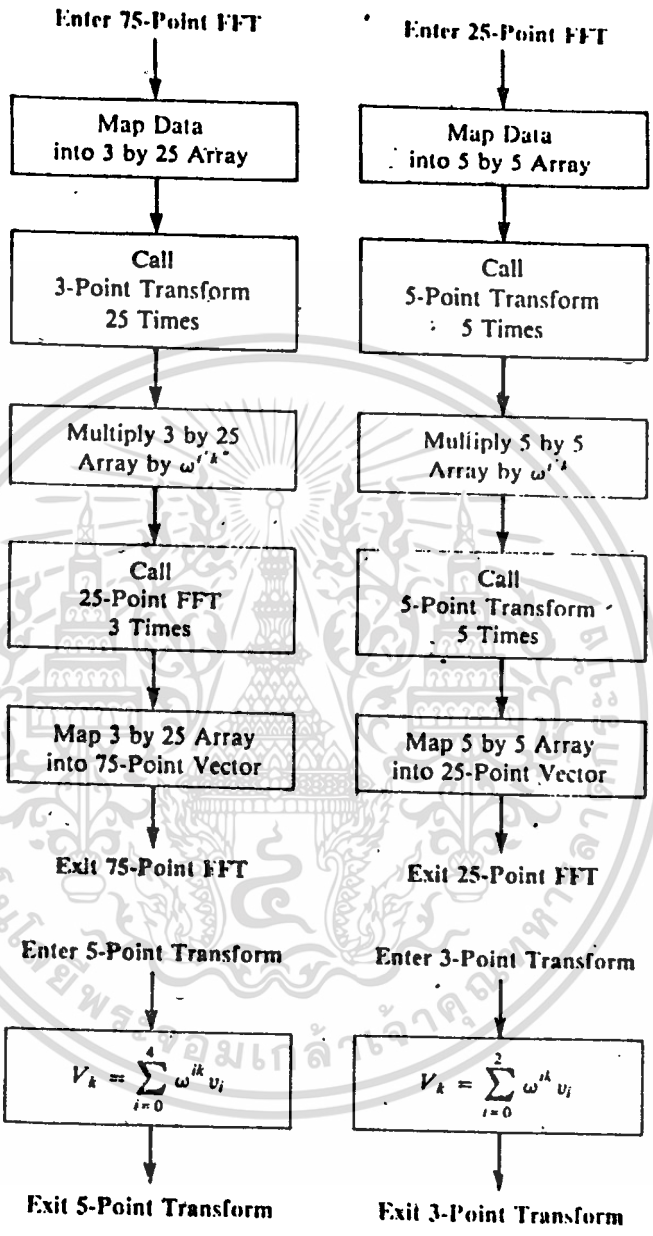
(i,n = 0,1,2,3,...,N-1), $w = \exp(-j2\pi/N)$

ส่วน 1/N นั้นเราจะไม่นำมาเขียนในตอนนี้ เพราะเป็นเพียงการ SCALE ซึ่งในตอนนี้ เราจะพิจารณาการทำงานของ FFT

ถ้าใช้วิธีการคำนวณ DFT โดยตรงจากสูตร (3.2) คำตอบจะได้เป็น

$$[\,Xn\,] = [\,W^{in}\,][\,xi\,]$$

ซึ่งทั้งหมดอยู่ในรูป COMPLEX NUMBER



รูปที่ 3.4. การแยก COOLEY - TUKEY FFT เป็น SUBROUTINE

ตัวอย่าง ให้ค่า N = 8 จะได้ MATRIXดังรูปที่ 3.5 ถ้าใช้การคำนวณโดยตรงแบบ DFT จะต้องคูณ 64 COMPLEX และบวก 64 COMPLEX ซึ่งเท่ากับ N^2 ของการคูณ และการบวก ในรูปของ COMPLEX NUMBER

ALGORITHM COOLEY-TUKEY เป็นการคำนวณที่อยู่บนฐานของการแตกออกของ MATRIX ซึ่งจะแยก MATRIX ออกเป็นหลายๆ MATRIX (เพราะในแต่ละชุดของ MATRIX ที่แยกออกมาจะมีเทอมที่เป็นศูนย์อยู่มาก)การใช้ TECHNIQUE นี้จะลดผลคูณที่มีอยู่มากมายในสมการ (3.2) นั้นลง ซึ่งจะลดเวลาลงมากเวลาคำนวณ

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ W^0 & W^2 & W^4 & W^6 & W^8 & W^{10} & W^{12} & W^{14} \\ W^0 & W^3 & W^6 & W^9 & W^{12} & W^{15} & W^{18} & W^{21} \\ W^0 & W^4 & W^8 & W^{12} & W^{16} & W^{20} & W^{24} & W^{28} \\ W^0 & W^5 & W^{10} & W^{15} & W^{20} & W^{25} & W^{30} & W^{35} \\ W^0 & W^6 & W^{12} & W^{18} & W^{24} & W^{30} & W^{36} & W^{42} \\ W^0 & W^7 & W^{14} & W^{21} & W^{28} & W^{35} & W^{42} & W^{49} \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix}$$

รูปที่ 3.5. MATRIX ของ DFT

ให้เราลองพิจารณา MATRIX ที่ประกอบไปด้วย R แถว S คอลัมน์ ซึ่งมีจำนวนทั้งหมด N = r.s การจัดเพื่อทำการแปลง FOURIER TRANSFORM สามารถจัดเป็น s กลุ่มโดยแต่ละกลุ่มประกอบด้วย r ตัวแล้วนำมาบวกกันเพื่อให้ได้ผลลัพท์ เราแก้ปัญหาโดยการใช้ SHORTER FOURIER TRANSFORM แทนการคำนวณแบบทั้งหมดพร้อมกัน ซึ่งลดเวลาได้มากทีเดียว ข้อดีของหลักการอีกอย่างหนึ่งคือ ต้องการMEMORY ระหว่างการคำนวณน้อยกว่าเพื่อให้เห็นภาพการทำงานของ ALGORITHM ในเทอมของ MATRIX ง่าย ๆ แยก N ออกเป็น 2 ค่า ซึ่ง TOTAL N = r.s สมมุติว่าเราแสดงตัวแปร N , n , i ตามสมการ (4.2) เป็นตัวย่อยๆ 4 ตัวคือ i_0, i_1, n_0 และ n_1

$$n = n_1 r + n_0 \qquad (3.3)$$

และ

$$i = i_1 s + i_0 \qquad (3.4)$$

ซึ่ง $n_0 = 0,1,2,3,\ldots,r-1$

$n_1 = 0,1,2,3,\ldots,s-1$

$i_0 = 0,1,2,3,\ldots,s-1$

$i_1 = 0,1,2,3,\ldots,r-1$

เราจะเห็นได้ว่า i และ n ยังเป็นค่า N อยู่ ถ้าเรากำหนดค่าเหล่านี้เป็นค่าตายตัว และแทนลงในสมการข้างบน

ตัวอย่าง ถ้า N = 20 , r = 5 และ s = 4 ดังนั้น n_0 จะมีค่าตั้งแต่ 0 ถึง 4, n_1 มีค่าจาก 0 ถึง 3, i_0 มีค่าจาก 0 ถึง 3 และ i_1 มีค่าจาก 0 ถึง 4 ตาราง 3.1 แสดงค่า n มีทั้งหมด 20 ค่าซึ่งเหมือนค่า i ที่ให้ไว้ในตาราง 3.2

จากสมการ 3.2 เราสามารถจัดใหม่ ตามตัวอย่าง เป็นสองผลบวก

$$x_{n_0,n_1} = \sum_{i0=0}^{i0=s-1} \sum_{i1=0}^{i1=r-1} x_{i_1,i_0} w^{in} \qquad (3.5)$$

ซึ่งเราจะทำการคูณ และบวกลูปในของสมการก่อน เมื่อเสร็จแล้วจึงมาทำลูปนอก

$$w^{in} = w^{(i1*+i0)(n1r+n0)}$$

$$= w^{n1i1rs} w^{n0i1*} w^{i0(n0+n1r)}$$

แต่ว่า i1n1rs = i1n1N เมื่อกำหนดให้ i1,n1 เป็น INTEGER

$$w^{i1n1N} = \exp(-j2\pi/N)^{i1n1N} = 1 \qquad (3.7)$$

ซึ่งเราจะได้ต่อมาว่า

$$w^{N/2} = \exp(-j2\pi/2) = -1 \qquad (3.8)$$

สมการ (3.5) เขียนใหม่ได้เป็น

$$x(n_1,n_0) = \sum_{i_0=0}^{s-1} \sum_{i_1=0}^{r-1} x(i_0,i_1)\, w^{n0i1*} w^{i0(n0+n1r)} \qquad (3.9)$$

ผลบวกลูปในของ i1 ขึ้นอยู่กับ i0 และ n0 สามารถเขียนใหม่ได้เป็น

$$x_1(i_0,n_0) = \sum_{i1=0}^{r-1} x(i_0,i_0)\, w^{n0i1*} \qquad (3.10)$$

สมการ (3.9) สามารถเขียนใหม่ได้เป็น

$$x(n_0,n_1) = \sum_{i0=0}^{s-1} x_{1(i0,n0)}\, w^{i0(n0+n1r)} \qquad (3.11)$$

Table 4.1 Factorized values of n

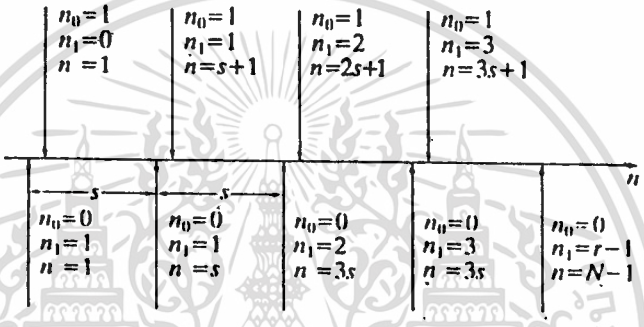
$n_1=$	0	1	2	3	n_0
$n=$	0	5	10	15	0
	1	6	11	16	1
	2	7	12	17	2
	3	8	13	18	3
	4	9	14	19	4

$r=5$

Table 4.2 Factorized values of i

$i_1=$	0	1	2	3	4	i_0
$i=$	0	4	8	12	16	0
	1	5	9	13	17	1
	2	6	10	14	18	2
	3	7	11	15	19	3

$s=4$



รูปที่ 3.6. แสดงถึงการแยกออกของ FFT

มี N ตัวแปรใน ARRAY X1 แต่มันจะไม่เหมือน ARRAY แรก ซึ่งก็คือ Xi ในการหาค่าของ x1 นั้น TOTAL OPERATE = N.r(COMPLEX OPERATION)ในทำนองเดียวกันเราต้อง การ N.s OPERATION เพื่อคำนวณหา X1 จาก Xi นี่เป็น ALGORITHM ง่ายๆ 2 เสต็ป ต้องการการคำนวณ

T = N(r+s) COMPLEX OPERATION

เราสามารถที่จะแสดงให้เห็นว่าเราจะขยายจำนวนเสต็ปขึ้นได้ถึงค่า P ซึ่งจำนวน OPERATION จะเป็น T = N(s1+s2+s3+s4+s5,,...,+sp) ซึ่ง N = s1,s2,s3,s4,...,sp ถ้า s1=s2=s3=s4=,...,=sp=s แล้ว $N = s^p$ ฉะนั้นเราจะแสดง s เป็นฐานได้ ซึ่งจะได้ว่า

$$p = \log_s N$$

$$T = sN \log_s N = N(s/\log_2 s)\log_2 N \qquad (3.12)$$

ถ้าเรา PLOT s กับ $s/\log_2 s$ จะเห็นว่า RADIX s = 3 มีประ

สิทธิภาพสูงสุด และขณะเดียวกันก็ลงความเห็นว่า มีการสูญเสียประสิทธิภาพน้อยมากเมื่อใช้กำลัง 2

ถ้าใช้ RADIX 2 เพื่อปรับปรุงความเร็วจากการคำนวณโดยตรง จะได้เป็น

$$N^2 / (2N\log_2 N) = N/(2\log_2 N) \qquad (3.13)$$

3.1.1 RADIX-2 FFT ALGORITHM

ALGORITHM FFT RADIX 2 ของ COOLEY-TUKEY ต้องการ N ที่เป็นกำลังของ 2 INPUT ที่เข้ามาจะถูกเปลี่ยนไป ทีละ STAGE ซึ่งแต่ละ STAGE ระหว่างการ TRANSFORM นั้น จะใช้ INPUT ARRAY แปลงเลยจนกระทั่งสุดท้าย INPUT นั้นก็จะกลายเป็น OUTPUT เลยจากที่เป็น INPUTเข้ามา

จากค่า N ซึ่งมีกำลัง 2 อย่างเดียว เป็นไปได้ที่จะแสดง i และ n เทอมในรูปของ BINARY - WEIGHTED SERIES

$$i = i_{p-1}2^{p-1}+i_{p-2}2^{p-2}+,\ldots,+i_2 2^2+i_1 2+i_0 \qquad (3.14)$$

$$n = n_{p-1}2^{p-1}+n_{p-2}2^{p-2}+,\ldots,+n_2 2^2+n_1 2+n_0 \qquad (3.15)$$

ในลักษณะนี้เป็นไปได้ ที่เราจะแสดงค่าของ N ในรูปของ BINARY NUMBER INPUT และ OUTPUT ของ FOURIER TRANSFORM Xi และ Xn จะถูกเก็บใน MEMORY ตำแหน่งที่กำหนด โดย BINARY ตามค่าของ i และ n

สมการ(3.2) สามารถที่จะเขียนใหม่ ในเทอมของผลบวกของ ตัวประกอบซึ่งการแยกออก จะแยกในรูปของ BINARY - WEIGHTED

$$X(np-1,np-2,\ldots,n0) = \sum_{i0=0}^{1}\sum_{i1=0}^{1},\ldots,\sum_{ip-1=0}^{1} x(ip-1,ip-2,\ldots,i0).$$

$$w^{n(i_{p-1}2^{p-1}+i_{p-2}2^{p-2}+\ldots+i0)} \qquad (3.16)$$

ตัวอย่าง $N = 2^2 = 4_{(10)} = 1\,0\,0_{(2)}$

ในรูปของฐาน สิบ

$$\sum_{i=0}^{i=3} xi_{(10)} = x_0 + x_1 + x_2 + x_3$$

ในรูปของฐาน สอง

$$\sum_{i=0}^{1}\sum_{i=0}^{1} x i_{(2)} = x_{00} + x_{01} + x_{10} + x_{11}$$

ทีนี้เราจะมากระจายเทอม EXPONENTIAL ของสมการ (3.16)

$$w^{n_1 p-1^{\frac{p-1}{2}}} = w^{n p-1^{\frac{p-1}{2}} i p-1^{\frac{p-1}{2}}} . w^{n p-2^{\frac{p-1}{2}} i p-1^{\frac{p-1}{2}}}, \ldots,$$

$$. w^{n 1^{2} i p-1^{2}}{}^{p-1} . w^{n 0 i p-1^{2}}{}^{p-1}$$

เราจะพบว่าเทอมกลางทั้งหมด มีค่า = 1 และจะได้ว่า

$$w^{n_1 p-1^{2}}{}^{p-1} = w^{n 0 i p-1^{\frac{p-1}{2}}}$$

ผลบวกชุดในสุดของสมการ (4.16)ในช่วงของ i_{p-1} จะเขียนได้เป็น

$$X_1(n_o, i_{p-2}, \ldots, i_o) = \sum_{i_{p-1}=0}^{i_{p-1}=1} x(i_{p-1}, i_{p-2}, \ldots, i_o) . w^{n 0 i p-1^{2}}{}^{p-1} \quad (3.19)$$

ซึ่งขึ้นอยู่กับ $n_o, i_{p-2}, \ldots, i_o$

มันไม่เหมือนกับการคำนวณ ของ FOURIER แบบสมบูรณ์ ซึ่งการบวกจะกระทำเป็น SET ของ N จำนวนเท่านั้น แต่วิธีนี้จะเป็นการคำนวณทีละ 2 จุด ของข้อมูล การบวกครั้งต่อมาจะกระทำ ตามสมการ (3.16) สามารถที่จะคำนวณได้โดยใช้สมการ EXPONENTIAL TERM

$$w^{n_1 p-q^{2}}{}^{p-q} = w^{(n q-1^{2}}{}^{q-1} + n q-2^{2}{}^{q-2} + \ldots + n 0)^{i p-q^{2}}{}^{p-q} \quad (4.20)$$

$$q = 1,2,3,\ldots,p$$

ผลบวกที่ได้จะเป็นไปตามลำดับตามสมการ

$$X_q(n_o, n_1, \ldots, n_{q-1}, i_{p-q-1}, i_{p-q-2}, \ldots, i_o)$$

$$= \sum_{i_{p-q}=1}^{i_{p-q}=1} x_{q-1}(n_o, n_1, \ldots, n_{q-2}, i_{p-q}, \ldots, i_o)$$

$$w^{(n q-1^{2}}{}^{q-1} + n q-2^{2}{}^{q-2} + \ldots + n 0)^{i p-q^{2}}{}^{p-q} \quad (3.21)$$

$$q = 1,2,3,\ldots,p$$

ในการใช้สูตรนี้ ค่าเริ่มแรก คือ DATA INPUT ครั้งแรกคือ $x_o(q=1)$

ดังนั้น

$$x_1 = x(i_{p-1}, i_{p-2}, \ldots, i_0) = x_0(i_{p-1}, i_{p-2}, \ldots, i_0)$$

ซึ่งค่าของ x_q ก็จะเพิ่มขึ้นเรื่อยๆ จนกระทั่งสุดท้ายจะเป็น Xn เมื่อเราได้แสดง SET 2 SET P $X_q(n_0, \ldots, n_{q-1}, i_{p-q-1}, \ldots, i_0)$ เป็น BINARY ตามตำแหน่งที่เก็บ ซึ่งใช้ในการทำงานของ ALGORITHM และลดการเก็บระหว่างการบวกนี้รวมทั้งการนำค่ามาจาก 2 ตำแหน่งของการเก็บ และนำมาทำการคูณและบวกตามสมการ (3.2) และเก็บค่าผลลัพท์ที่ได้ไว้ในตำแหน่งเดิม โดยใส่แทนเข้าไปยัง INPUT เดิมข้อเสียอีกอย่างหนึ่ง คือค่า Xn สุดท้ายที่ได้จะเก็บไว้ไม่ถูกต้องตามตำแหน่ง ดังนั้นสำหรับการคำนวณครั้งสุดท้าย

$$X(n_{p-1}, n_{p-2}, \ldots n_0) = X_p(n_0, \ldots, n_{p-2}, n_{p-1}) \quad (3.22)$$

นี่เป็นการแสดงการหาค่า Xn BIT ใน BINARYตามค่า n โดยจะต้อง REVERSE ตัวชี้ MEMORY ซึ่ง Xn ก็จะได้จาก Xp ARRAY จากสมการ 3.21 เราจะเห็นว่าแต่ละค่าของ ตัวแปรของ x_q ใช้ 1 COMPLEX ของการคูณและ 1 COMPLEX ของการบวก รวมทั้งการคูณและการบวกของ COMPLEX EXPONENTIAL ดังนั้นจำนวนการกระทำ $T = 2Np$ แต่ $N = 2^p$ ค่าในแต่ละ X_q ซึ่ง $q = 1,2,3,\ldots,p$ การปรับปรุงความเร็ว จากการคำนวณโดยตรงคือ $T/T = N/2\text{Log}_2 N$ อ้างถึงสมการ (3.2) ถ้าเราลองแบ่ง x_1 เป็น 2 ชุด ดังนี้

$$x_{2i} = y_i$$

$$x_{2i+1} = z_i$$

$(i = 0,1,2,3,\ldots,(N/2)-1)$ ซึ่ง y_i ประกอบด้วย INPUT ที่เป็นเลขคู่ ส่วน z_i ประกอบด้วย INPUT ที่เป็นเลขคี่สามารถที่จะคำนวณ FOURIER TRANSFORM ได้ดังนี้

$$Y_n = \sum_{i=0}^{(N/2)-1} y_i W^{2in}$$

$$Z_n = \sum_{i=0}^{(N/2)-1} z_i W^{2in}$$

ซึ่ง i และ n มีค่าเพียงครึ่งหนึ่งของค่า N $W_{N/2} = \exp(-2\pi j/N/2) = W^2_N$

ถ้าเราจะนำ Yn และ Zn มารวมกันเพื่อให้ได้ N POINT

$$Xn = \sum_{i=0}^{(N/2)-1} [\, y_i W^{n(2i)} + z_i W^{n(2i+1)} \,]$$

$$= \sum_{i=0}^{(N/2)-1} y_i W^{2in} + W^n \sum_{i=0}^{(N/2)-1} z_i W^{2in}$$

ในเมื่อ W^n เป็นค่าคงที่ตามค่าของ n ดังนั้น

$$Xn = Yn + W^n Zn$$

แต่ n มีค่าถึงเพียง (N/2)-1 ดังนั้นเพื่อให้สมการ Xn สมบูรณ์เราต้องหาค่าเทอมที่อยู่ระหว่าง N/2 ถึง N-1 ด้วย

จาก ทฤษฎี ของ DFT ซึ่ง n>N/2 การ TRANSFORM Yn และ Zn จะเป็นคลื่นที่ซ้ำกับ ค่าที่อยู่ในช่วง $0 < n < N/2$ ดังนั้นสามารถที่จะแทนค่าได้ด้วย(n + N/2) ในรูปของ PHASE SHIFT

$$X(n + N/2) = Yn + W^{n+N/2} Zn$$

แต่

$$W^{n+N/2} = \exp(-j2\pi n/N - j\pi) \qquad (3.29)$$

จะได้ว่า

$$-W^n = -\exp(-j2\pi n/N) \qquad (3.30)$$

ดังนั้น

$$X(n + N/2) = Yn - W^n Zn \qquad (3.31)$$

สมการ(3.28) และ(3.31) สามารถที่จะหาค่า N/2 แรก และหลังได้พร้อมกัน เราจะเห็นว่าเราต้องการ $N + 2(N/2)^2$ ตัวคูณ

สมการ(3.28) และ(3.31) อธิบายการทำงานเบื้องต้นของ RADIX 2 FFT W^n เป็นCOMPLEX OPERATION ซึ่งเราจะเรียกว่า TWIDDLE FACTOR

การทำงานของ PROCESS คือการหาร INPUT ออกเป็น 2 ส่วนแล้วหาทีละส่วนซึ่งจะลดเวลาในการคำนวณลง ซึ่งมีข้อจำกัดว่า INPUT ค่า N จะต้องหารด้วย 2 ลงตัวเท่านั้นซึ่งหาได้ตั้งแต่ 2 จุดขึ้นไป ส่วน DFT ของ INPUT จุดเดียว คือค่าของมันเอง

$$Xn = \sum_{i=0}^{i=0} xi\, W^{in} = xi \qquad (3.32)$$

TECHNIQUE ที่กล่าวมาเรียกว่า DECIMATION IN TIME มี TECH-

NIQUE คล้ายๆกันนี้อีก TECHNIQUE หนึ่ง ซึ่งแบ่ง xi ออกเป็น 2 ส่วนเลย คือ N/2 แรกและ N/2 หลัง

$$xi = yi'$$
$$xi+N/2 = zi'$$

i = 0,1,2,3,...,(N/2)-1 TECHNIQUE นี้เรียกว่า DECIMATION IN FREQUENCY และ TRANSFORM Xn จะกลายเป็น

$$Xn = \sum_{i=0}^{(N/2)-1} (yi' + zi' W^{Nn/2})W^{in} \qquad (3.34)$$

สำหรับจำนวณคู่และคี่ แยกกันหาจะได้เป็น

$$Yn' = \sum_{i=0}^{(N/2)-1} (yi' + zi')W^{2in} \qquad (3.35)$$

เปลี่ยนค่า n เป็น 2n + 1 ซึ่งเป็นเลขคี่จะได้เป็น

$$Zn' = \sum_{i=0}^{(N/2)-1} (yi' - zi')W^{i}W^{2in} \qquad (3.36)$$

เมื่อ $W^{Nn/2} = (-1)^n$

ดังนั้น Xn จะคำนวณได้ทีละ 2 ค่าของความยาว N/2 ตามสมการ (3.35) และ (3.36) คล้ายกันเพียงแต่ว่า สมการ (3.36) ต้องคูณด้วย W^i กับ INPUT ซึ่งไม่มีผลต่อการคำนวณรวมทาง คณิตศาสตร์

3.1.2 แมทริคส์ และ Signal Diagram

แมทริคส์จัตุรัส ของการ TRANSFORM พิจารณาในเทอมของของการคูณแมทริคส์ สามารถแสดงได้ดังนี้

$$Xn = W^{in}xi \qquad (3.37)$$

ซึ่ง W^{in} คือ COMPLEX MATRIX ซึ่งมีขนาด N.N TERM MATRIX สำหรับ W^{in} ซึ่ง N = 8 ในรูป (3.5) สามารถที่จะยุบโดยใช้สมการ (3.30)จะได้รูปทั่วๆไป

$$W^{m} = W^{(m \bmod N)} \qquad (3.38)$$

ซึ่ง m mod (N) คือค่าที่เหลือเมื่อ m ถูกหารด้วย N

$$m \bmod(N) = m - \lfloor m/N \rfloor N \qquad (3.39)$$

ซึ่ง $\lfloor . \rfloor$ แสดงถึงค่าจำนวนเต็ม

ดังนั้นเมื่อ N = 8 กำลังของ W จึงมีค่าไม่เกิน 3 ซึ่ง MATRIX จากรูป 3.6 จะถูกลดลงได้ดังรูป 3.7 และเราสามารถเขียนเป็น

$$Xn = F8x_i \qquad (3.40)$$

(i,n = 0,1,2,...,7) และ F8 ก็คือ 8.8 DFT ซึ่งจะแยก FACTOR ออกในรูปของ

$$Xn = [B1.C1.C2.P.P1]x_i \qquad (3.41)$$

ซึ่ง B1,C1 และ C2 เป็น MATRIX ย่อยของ F8 และ P กับ P1เป็น PERMUTATION MATRIX (คือ MATRIX ที่ใช้สลับที่ของผลลัพท์)

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & -1 & -W & -W^2 & -W^3 \\ 1 & W^2 & -1 & -W^2 & 1 & W^2 & -1 & -W^2 \\ 1 & W^3 & -W & W & -1 & -W^3 & W^2 & -W \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -W & W & -W^3 & -1 & W & -W^2 & W^3 \\ 1 & -W^2 & -1 & W^2 & 1 & -W^2 & -1 & W^2 \\ 1 & -W^3 & -W^2 & -W & -1 & W^3 & W^2 & W \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix}$$

รูปที่ (3.7) การยุบค่า MATRIX ของ DFT

P,P1 ก็เป็น MATRIX ย่อยด้วย ซึ่งจะรวมเป็น MATRIX เดียวกัน เพื่อจัดเรียงลำดับใหม่ซึ่งจะเป็น INPUT หรือ OUTPUT ของค่า TRANSFORM ซึ่งแยกจากการคำนวณให้ N = N1N2 = 2.4 แล้ว ตัดเอาส่วนบนของ F8 มา N1 แถวแล้วแบ่งเป็น N1 ส่วน แต่ละส่วนมี N2 สัมประสิทธิ

$$\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & W & W^2 & W^3 & -1 & -W & -W^2 & -W^3 \end{array} \qquad (3.42)$$

เราสามารถที่จะสร้าง MATRIX B1 โดยจัดเรียงตามเส้นทะแยงมุมเป็น แมทริคส์ย่อยขนาด N2.N2 ซึ่งนำมาจากส่วนที่ตัดมาจะได้ดังนี้

$$B1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & W & 0 & 0 & 0 & -W & 0 & 0 \\ 0 & 0 & W^2 & 0 & 0 & 0 & -W^2 & 0 \\ 0 & 0 & 0 & W^3 & 0 & 0 & 0 & -W^3 \end{bmatrix} \qquad (3.43)$$

MATRIX B2 เป็นการจัด MATRIX ทะแยงขวา ซึ่งมี MATRIX ย่อย ขนาด N2.N2 ตามเส้นทะแยง ตัวแปรของ SUBMATRIX จะมีค่าเป็น $W^{N_{a,b}}$ ซึ่ง a,b = 0,1,2,3 ดังนั้น ค่า W จะเป็น (เมื่อค่า N1 = 2)

b \ a	0	1	2	3
0	0	0	0	0
1	0	2	4	6
2	0	4	8	12
3	0	6	12	18

$$B2 = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & & & & \\ 1 & W^2 & -1 & -W^2 & & & & \\ 1 & -1 & 1 & -1 & & 0 & & \\ 1 & -W^2 & -1 & -W^2 & & & & \\ \hline & & & & 1 & 1 & 1 & 1 \\ & & & & 1 & W^2 & -1 & -W^2 \\ & 0 & & & 1 & -1 & 1 & -1 \\ & & & & 1 & -W^2 & -1 & W^2 \end{array}\right] \qquad (3.44)$$

ซึ่ง 0 หมายถึงค่าของ MATRIX ย่อยนั้น มีค่าเป็น 0 ทั้งหมดการสับเปลี่ยนตำแหน่งของ MATRIX P จะเห็นได้จากตัวประกอบของ N2แถวแรกใน MATRIX B1 ตัวประกอบของแถวเหล่านี้ นำมาเป็นแถวต่อแถวเพื่อใส่ใน MA-

TRIX P ใน COLUMN ที่ตรงกับ B1 แต่ตัวชี้ของแถวเพิ่มขึ้นเท่ากับ N1

$$P1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.45)$$

MATRIX ย่อย B2 มันสามารถที่จะแบ่งย่อยตัวเอง ออกเป็น 2 MATRICES คือ C1 และC2 ทำเหมือนกับ B1 เราจะได้ C1 จาก 2 แถวบนของ B2 จะได้เป็น

$$C1 = \left[\begin{array}{cccc|cccc} 1 & 0 & 1 & 0 & & & & \\ 0 & 1 & 0 & 1 & & & 0 & \\ 1 & 0 & -1 & 0 & & & & \\ 0 & W^2 & 0 & -W^2 & & & & \\ \hline & & & & 1 & 0 & 1 & 0 \\ & & 0 & & 0 & 1 & 0 & 1 \\ & & & & 1 & 0 & -1 & 0 \\ & & & & 0 & W^2 & 0 & -W^2 \end{array}\right] \qquad (3.46)$$

แมทริคส์ C2 จะเป็น MATRIX ทะแยงขวา ซึ่งมี MATRIX ย่อยเป็น N2/2.N2/2 ตามเส้นทะแยง ตัวประกอบเหล่านี้ได้จาก W^{n2ab} a,b = 0,1

b\a	0	1
0	0	0
1	0	4

และจะได้ C2 เป็น

$$C2 = \left[\begin{array}{cccccccc} 1 & 1 & & & & & & \\ 1 & -1 & & & & & 0 & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & 0 & & & & & 1 & 1 \\ & & & & & & 1 & -1 \end{array}\right] \qquad (3.47)$$

และ PERMUTATION MATRIX จะได้เป็น

$$P1 = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & & & & \\ 0 & 0 & 1 & 0 & & 0 & & \\ 0 & 1 & 0 & 0 & & & & \\ 0 & 0 & 0 & 1 & & & & \\ \hline & & & & 1 & 0 & 0 & 0 \\ & & & & 0 & 0 & 1 & 0 \\ & 0 & & & 0 & 1 & 0 & 0 \\ & & & & 0 & 0 & 0 & 1 \end{array}\right] \qquad (3.48)$$

ผลคูณของ แมทริคส์กระจาย(SPARSE MATRIX) $B_1 . C_1 . C_2$ คูณกับ INPUT VECTOR x_1 ซึ่งจะให้ค่าที่ได้จากการทรานสฟอร์มเป็น X_n ที่เรียงลำดับแบบ bit-reverse การคูณแมทริคส์ที่สมบูรณ์ สำหรับ N = 8 ได้ทำการรวบรวมไว้ในรูปที่ 3.9 การแปลง bit-reverse แมทริคส์ กลับมาจะทำการคูณด้วย แมทริคส์ P . P_1 เพราะฉะนั้น สมการของ แมทริคส์จะได้ว่า

$$X_n = B_1 . C_1 . C_2 . x_1$$

$$\left[\begin{array}{c} X_0 \\ X_4 \\ X_2 \\ X_6 \\ X_5 \\ X_3 \\ X_7 \end{array}\right] = \left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & W & 0 & 0 & 0 & -W & 0 & 0 \\ 0 & 0 & W^2 & 0 & 0 & 0 & -W & 0 \\ 0 & 0 & 0 & W^3 & 0 & 0 & 0 & -W^3 \end{array}\right] .$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & W^2 & 0 & -W^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & W^2 & 0 & -W^2 \end{bmatrix}$$

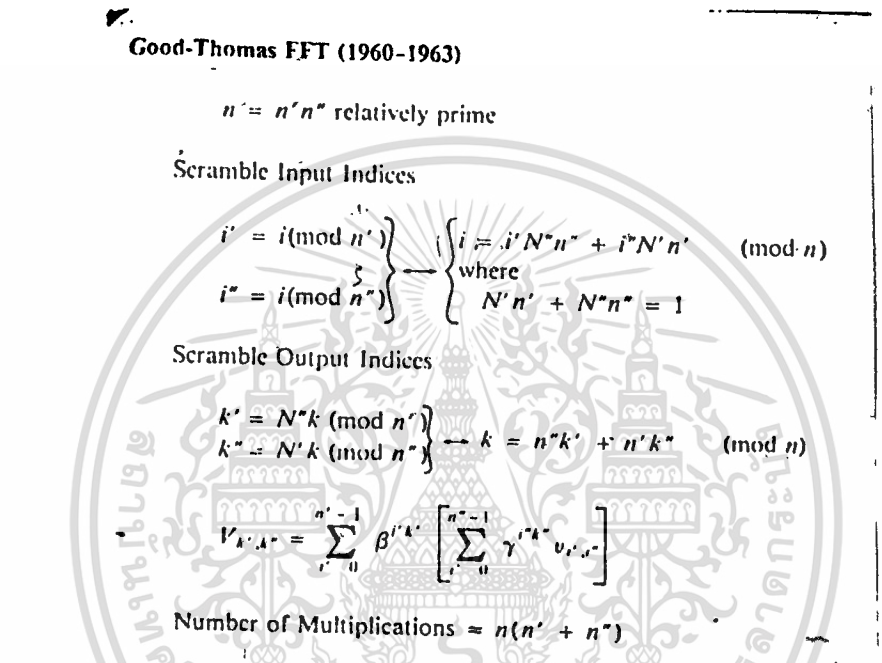
$$\begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & 1 \\ & & & & & & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

รูปที่ 3.8. Factorizing the DFT matrix

3.2 GOOD THOMAS PRIME ALGORITHM

Good-Thomas Algorithm ได้ทำการสรุปไว้ที่ 3.9 เป็นวิธีการหนึ่งเหมือนกันที่ทำการจัด Input จาก n = n'n" เข้าใส่ใน array 2 มิติ ที่มีขนาด n' . n" ถึงจะเป็นการจัดจากมิติเดียวไปเป็น 2 มิติ ก็จริงอยู่ แต่ว่าแนวความคิด ในการออกแบบได้แตกต่างจาก Algorithm ของ Cooley-Tukey ขณะเดียวกันค่า n' และ n" จะต้องเป็น Relative prime และการ map จะใช้หลักการของ Chinese Remainder theorem ในรูปที่ 9. จะแสดงให้เห็นว่า Input ได้ถูกจัดอย่างไร มันจะถูกเก็บใน Array ขนาด 2 มิติ โดยเริ่มจากมุมบนทางด้านซ้ายสุด และเรียงลงตาม "Extended Diagonal "เพราะว่าจำนวนแถว และจำนวนคอลัมน์เป็นเลข Relative Prime ค่าทุกค่าจึงถูกจัดลง Array ได้พอดีหลังจากการ Transformแล้ว หลังจากนั้น จะถูกจัดเรียงลง Array 2 มิติอีก อันหนึ่งแล้วจัดลำดับ Output ให้ถูกต้องใส่ใน OUTPUT Array ถึงอย่างไรก็ตามการจัดเรียงทางด้าน Input

กับ Output มีความแตกต่างกัน การจัดเรียงทางด้าน Input และ Output จะได้ทำการอธิบายดังข้างล่าง



รูปที่ 3.9. GOOD - THOMAS FFT.

การเปลี่ยนแปลง ของ GOOD - THOMAS ALGORITHM อยู่บนพื้นฐานของ CHINESE REMAINDER THEORY สำหรับจำนวนเต็มการจัดเรียงINPUT จะอธิบายได้โดย RESIDUES ของมันดังนี้

$$i' = i \pmod{n'}$$

$$i'' = i \pmod{n''}$$

นี่คือ การ MAP INPUT ชี้โดย i ลงใน Extended Diagonal ของ Array ขนาด 2 มิติที่ถูกชี้โดย (i', i") โดย Chinese Remainder Theorem มีการมีอยู่ของจำนวนเต็ม N' และ N" ซึ่งชี้ input ที่จะสามารถถูกนำกลับคืนมาได้ตาม

$$i = i'N''n'' + i''N'n' \pmod{n}$$

ซึ่ง N' และ N" เป็นจำนวนเต็มที่ได้มาจาก

$$N'n' + N"n" = 1$$

แต่ตัวชี้ทางด้าน output จะมีความแตกต่างกันอธิบายได้ดังนี้

$$k' = N"k \pmod{n'}$$
$$k" = N'k \pmod{n"}$$

หรือสามารถที่จะเขียนได้โดย

$$k" = ((N' \bmod n")k \bmod n")$$

และ

$$k' = ((N" \bmod n')k \bmod n')$$

output ตัวชี้โดย k สามารถที่จะถูกนำกลับมาได้โดย

$$k = n"k' + n'k" \pmod{n}$$

สามารถที่จะพิสูจน์ได้โดย

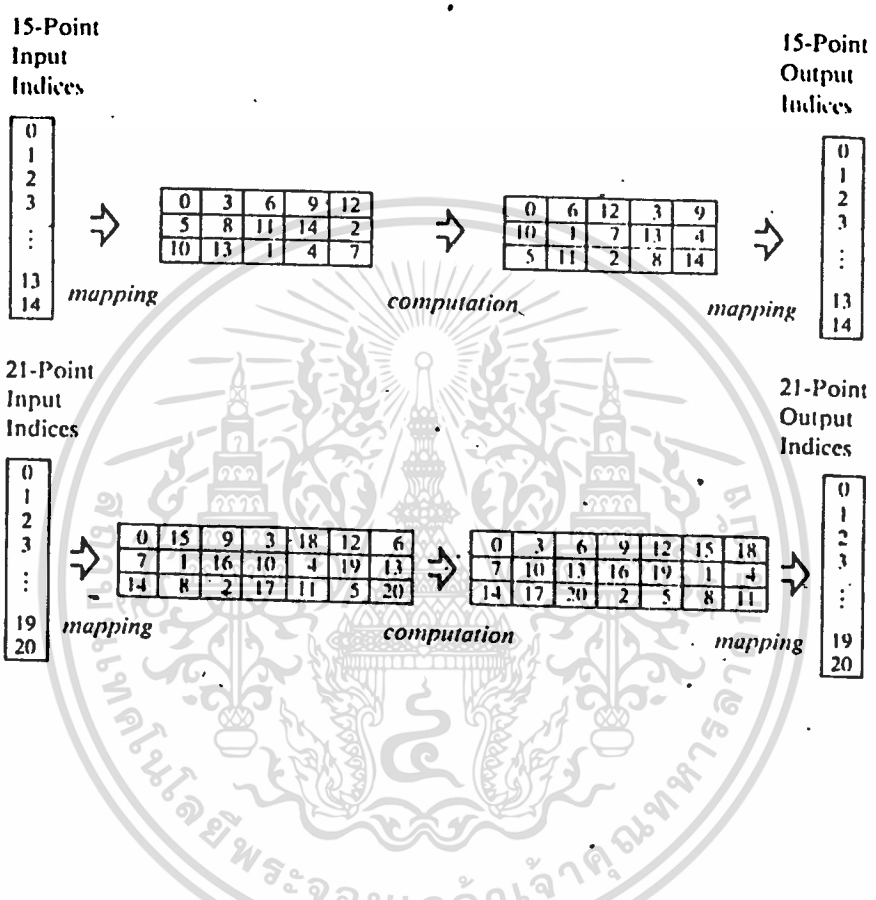
$$k = n"(N"k + Q_1 n') + n'(N'k + Q_2 n") \pmod{n'n"}$$
$$= k(n"N" + n'N') \pmod{n}$$
$$= k$$

เราจะทำการเปลี่ยนตัวชี้ใหม่โดยใช้จากสูตร

$$V_k = \sum_{i=0}^{n-1} w^{ik} \cdot v_i$$

ไปเป็น

$$V_{k',k"} = \sum_{i"=0}^{n"-1} \sum_{i'=0}^{n'-1} w^{(i'N"n" + i"N'n')(n"k' + n'k")} V_{i'N"n" + i"N'n'}$$



รูปที่ 3.10 ตัวอย่างการจัดการแยก N ของ GOOD THOMAS FFT

3.3 WINOGRAD ALGORITHM

วินอกราดได้แสดงวิธีการแก้ปัญหาการทรานสฟอร์มฟูเรียสำหรับ ค่าตัวประกอบเล็ก ๆ ตามทฤษฎีของ ไชนิสริเมนเดอร์(Chinese Remainder Theory) ในรูปของโพริโนเมียล (Polynomial) เพื่อหาคำตอบที่ทำให้ผลบวกเท่ากับวิธีการหาคำตอบของคูเลย์ และเทิคคีย์(COOLEY - TUKEY) แต่ผลคูณประมาณ 20 เปอร์เซนต์ของจำนวนการคูณของวิธีการคำนวนของ คูเลย์และเทิคคีย์ ที่ต้องการใช้เท่านั้น นอกเหนือจากนั้นเขายังได้แสดงให้เห็นจริง

ว่า โดยการจัดเรียงลำดับข้อมูลอินพุทใหม่ก่อน ที่จะทำการทรานสฟอร์มตามวิธีการของกูด-โทมัส บนพื้นฐานตามทฤษฎีของไชนิสริเมนเดอร์ ซึ่งได้กล่าวว่าตัวประกอบจะต้องเป็น Prime Number เท่านั้น ยังไม่สมสมบูร์ณทีเดียวนัก จริงๆแล้วตัวประกอบต้องเป็น Relatively Prime จึงจะสมบูรณ์

ทฤษฎีพื้นฐาน

กำหนดให้ Polynomial 2 พจน์ ที่มีค่า สัมประสิทธิ์ไม่แน่นอน

$$R_1(u) = \sum_{i=0}^{1} x_i u^i \quad , \quad S_m(u) = \sum_{i=0}^{m} y_i u^i$$

และกำหนด $P(u) = u^n + \sum_{i=0}^{n-1} a_i u^i$ เป็น Monic Polynomial ของ Degree n ที่มีค่าสัมประสิทธิ์ใน Field G หนึ่ง.(ในการประยุกต์เราจะใช้ $\mathcal{G}$ เป็น Field Q ของเศษส่วน) สมมติว่า $P(u) = P_1(u) \cdot P_2(u)$ โดย $P_1(u)$ และ $P_2(u)$ เป็น Relatively prime และ กำหนด $n_1 = \deg(P_1)$ และ $n_2 = \deg(P_2)$

ใช้ Chinese Remainder เราจะได้ว่า

(1) $R_1 \cdot S_m \bmod P$

$= (Q_2 \cdot P_2 (R_1 \cdot S_m \bmod P_1) + Q_1 \cdot P_1 (R_1 * S_m \bmod P_2)) \bmod P$

เมื่อ Q_1 และ Q_2 เป็น Polynomial ดังนี้

(2) $Q_1 \cdot P_1 + Q_2 \cdot P_2 = 1 \bmod P$

กำหนดให้ T เป็น set ของค่าสัมประสิทธิ์ ของ $R_1 \cdot S_m$ และกำหนด T_p เป็น Set ของค่าสัมประสิทธิ์ของ $R_1 \cdot S_m \bmod P$ เมื่อ P มีตัวประกอบที่แยกไม่ได้เพียงหนึ่งตัวเราไม่สามารถใช้สมการ(1)ได้เลย แต่เราจะใช้ T_p โดยการคำนวน T ก่อนเพื่อลด Modolo P มี 2 ทางที่จะคำนวน T ที่ใช้การคูณเพียง m+1+1เท่านั้น วิธีที่หนึ่งแสดงได้ดังนี้

(3) $R_1(u) \cdot S_m(u) = R_1(u) \cdot S_m(u) \bmod \prod_{i=0}^{m+1} (u - \alpha_i)$

เมื่อ α_i เป็น Element ที่แจ่มแจ้งของ G

วิธีที่ 2.

(4) $R_1(u) \cdot S_m(u) = R_1(u) \cdot S_m(u) \bmod \prod_{i=1} (u - \beta_i)$

$$+ x_1 y_m \prod_{i=1}^{m+1} (u - \beta i) .$$

เมื่อ β_i เป็น element ที่แจ่มแจ้งของ G

CYCLIC CONVOLUTION

เราจะมาพิจารณาปัญหา การคำนวนของ 2 set ของ n จุดที่มีการหมุนแบบครบวงรอบต่อไปนี้คือ $(x_o, x_1 u, x_2 u^2, x_3 u^3, \ldots, x_{n-1} u^{n-1})$และ $(y_o, y_{n-1} u, y_{n-2} u^2, \ldots, y_1 u^{n-1})$สามารถจะเขียนเป็นแมทริคส์ได้ดังนี้

(10)
$$\begin{bmatrix} x_o & x_1 & \cdots & x_{n-1} & x_{n-1} \\ x_1 & x_2 & & x_{n-1} & x_o \\ \cdot & & & & \cdot \\ x_{n-1} & x_o & & x_{n-3} & x_{n-2} \end{bmatrix} \begin{bmatrix} y_o \\ y_1 \\ \cdot \\ y_{n-1} \end{bmatrix}$$

และ เขียนอยู่ในรูป Polynomial ได้ดังนี้

(11) $(x_o + x_1 u + x_2 u^2 + \ldots + x_{n-1} u^{n-1}) \cdot$
$(y_o + y_{n-1} u + y_{n-2} u^2 + \ldots + y_1 u^{n-1}) \bmod u^n - 1$

และเราสามารถที่จะใช้ทฤษฎีที่ได้กล่าวมาทำการแก้ปัญหาเพื่อหาคำตอบได้ ตัวอย่างเช่น สมมุติ n = 3 นั้นคือ

(12)
$$\begin{bmatrix} z_o \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_o & x_1 & x_2 \\ x_1 & x_2 & x_o \\ x_2 & x_o & x_1 \end{bmatrix} \begin{bmatrix} y_o \\ y_1 \\ y_2 \end{bmatrix}$$

ดังนั้น สัมประสิทธิ์เทอมแสดงได้ดังนี้

(13)

$(x_o + x_1 u + x_2 u^2) \cdot (y_o + y_2 u + y_1 u^2) \bmod (u^3 - 1)$

แต่เนื่องจาก $u^3 - 1 = (u + 1)(u^2 + u + 1)$ เราต้องคำนวน

(14)

$$(x_0 + x_1u + x_2u^2) \cdot (y_0 + y_2u + y_1u^2) \bmod (u+1) = (x_0 + x_1 + x_2)(y_0 + y_1 + y_2)$$

สังเกตุว่า สมการ(14) กระทำการคูณเพียง 1 ครั้ง และ

(15)

$$(x_0 + x_1u + x_2u^2) \cdot (y_0 + y_2u + y_1u^2) \bmod (u^2 + u + 1) = ((x_0 - x_2) + (x_1 - x_2)u)((y_0 - y_1) + (y_2 - y_1)u) \bmod (u^2 + u + 1)$$

เพื่อที่จะคำนวน สมการ(15) เราจะต้องใช้ T นั้นคือ เราต้องใช้ผลสัมประสิทธิ์เทอมของสมการ(15) $((x_0-x_2)+(x_1-x_2)u)((y_0-y_1)+(y_2-y_1)u)$ หาต่อโดยใช้สมการที่(4) แสดงได้ดังนี้

(16)

$$((x_0-x_2)+(x_1-x_2)u)((y_0-y_1)+(y_2-y_1)u) = ((x_0 - x_2) + (x_1 - x_2)u)((y_0 - y_1) + (y_2 - y_1)u) \bmod u(u+1) + (x_1 - x_2)(y_2 - y_1)u(u+1)$$

ซึ่งจะนำเราไปสู่ Algorithm

$m_1 = (x_0 - x_2)(y_0 - y_1)$, $m_2 = (x_1 - x_2)(y_2 - y_1)$

$m_3 = ((x_0-x_2)-(x_1-x_2))((y_0-y_1)-(y_2-y_1)) = (x_0-x_1)(y_0-y_2)$

$(x_0-x_2)(y_0-y_1) = m_1$,

(17)

$$(x_0-x_2)(y_2-y_1)+(x_1-x_2)(y_0-y_1) = m_1 + m_2 - m_3,$$

$$(x_1-x_2)(y_0-y_1) = m_2.$$

เพราะฉะนั้น สัมประสิทธิ์ของ (15) จะได้ว่า

$m_1 - m_2$ และ $m_1 + m_2 - m_3 - m_2 = m_1 - m_3$.

กำหนดให้ $m_0 = (x_0 + x_1 + x_2)(y_0 + y_1 + y_2)$ เราจะได้ว่า

$(x_0 + x_1u + x_2u^2)(y_0 + y_2u + y_1u^2) \bmod (u-1) = m_0$,

(18)

$$(x_0 + x_1u + x_2u^2)(y_0 + y_2u + y_1u^2) \bmod (u^2 + u + 1) = (m_1 - m_2) + (m_1 - m_3)u;$$

และใช้ (1) เราจะได้ว่า

(19)

$$(x_0 + x_1u + x_2u^2)(y_0 + y_2u + y_1u^2) \bmod (u^3 - 1)$$

$$= (u^2+u+1)m_0/3 + \{-(u-1)(u+2)/3\}((m_1-m_2) + (m_1 - m_3)u) \mod (u^3-1)$$

$$= (m_0/3 + m_1/3 - 2m_2/3 + m_3/3) + (m_0/3 + m_1/3 + m_2/3 - 2m_3/3)u + (m_0/3 + 2m_1/3 + m_2/3 + m_3/3)u^2$$

ในการประยุกต์ใช้ ของ x_1 หรือ y_1 อย่างใดอย่างหนึ่ง เป็นที่รู้กันในกลุ่มของนักคณิตศาสตร์

ตัวอย่างเช่น มันเป็น tap value ดังนั้นการคำนวณเกี่ยวพันธ์กับตัวแปรเหล่านี้เท่านั้น สามารถที่จะทำด้วยมือก่อน และไม่ควรจะนับ การคำนวณเข้าไปด้วย สมมุติว่าการคำนวณด้วย x_1 ไม่ถูกนับเข้าไปเราจะกำหนดได้ว่า

(20)

$$m'_0 = (x_0 + x_1 + x_2)/3(y_0 + y_1 + y_2)$$
$$m'_1 = (x_0 - x_2)/3(y_0 - y_1),$$

(20a)

$m'_2 = (x_1 - x_2)/3(y_0 - y_1)$, $m'_3 = (x_0 - x_1)/3(y_0 - y_2)$, และค่าคำตอบ 3 ค่าก็จะแสดงได้ดังนี้

(20b)

$$m'_0 + m'_1 - 2m_2 + m'_3,$$
$$m'_0 + m'_1 + m'_2 - 2m'_3,$$
$$m'_0 - 2m'_1 + m'_2 + m'_3.$$

และ ALGORITHM อื่นที่เหลือ จะหาได้จากการ TRANSPOSE ของสมการที่ (12) คือ

(21)

$$\begin{bmatrix} z_0 & z_2 & z_1 \\ z_1 & z_0 & z_2 \\ z_2 & z_1 & z_0 \end{bmatrix}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} z_0 & z_1 & z_2 \\ z_1 & z_2 & z_0 \\ z_2 & z_0 & z_1 \end{bmatrix}\begin{bmatrix} y_0 \\ y_2 \\ y_1 \end{bmatrix}$$

เมื่อเราทำการ TRANSPOSE สมการ (20) เราจะได้ว่า

(22a)

$$m_0 = (z_0 + z_1 + z_2)/3(y_0 + y_1 + y_2)$$
$$m_1 = (z_0 + z_1 - 2z_2)/3(y_0 - y_1),$$
$$m_2 = (-2z_0 + z_1 + z_2)/3(y_0 - y_1),$$

$$m_3 = (z_0 - 2z_1 + z_2)/3(y_0 - y_2),$$

และค่าคำตอบ 3 ค่าจะสามารถคำนวนได้ดังนี้

(22b)

$$m_0 + m_1 + m_3 \ , \ m_0 + m_2 - m_3 \ , \ m_0 - m_1 - m_2 .$$

Matrix ในสมการ (10) สามารถที่จะแสดงเป็น " MULTIPLICATION TABLE "

สำหรับกลุ่มของ z_n ของการบวก Modulo n เมื่อ $n = n_1 \ , \ n_2$ ซึ่ง n_1 และ n_2 เป็น Relative prime แล้ว z_n เป็น isomorphic ที่ $z_{n_1} \cdot z_{n_2}$ เพราะฉะนั้นจะมีการหมน ของแถว และ คอลัมน์ ของ matrix ในสมการ (10) ซึ่งผลที่ได้จะแบ่งเป็น block ขนาด $n_2 \ . \ n_2$ Cyclic Matices และจะมีขนาด $n_1 \ . \ n_1$ Cyclic Matrices

ตัวอย่าง เนื่องจาก $6 = 2 \ . \ 3$ เราจะได้ว่า

(23)

$$0 \ -> \ (0,0), \ 1 \ -> \ (1,1), \ 2 \ -> \ (0,2),$$
$$3 \ -> \ (1,0), \ 4 \ -> \ (0,1), \ 5 \ -> \ (1,2) \ ;$$

(24)

$$\begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{bmatrix} = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_0 \\ x_2 & x_3 & x_4 & x_5 & x_0 & x_1 \\ x_3 & x_4 & x_5 & x_0 & x_1 & x_2 \\ x_4 & x_5 & x_0 & x_1 & x_2 & x_3 \\ x_5 & x_0 & x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

เราจะจัดให้มันอยู่ตาม ลำดับ 0,4,2,3,1,5 Block Structure นี้สามารถใช้แก้ปัญหา โดยใช้ $u^2 - 1 = (u+1)(u-1)$ สามารถที่จะแสดงการแก้ปัญหาซึ่งจะนำไปสู่ Algorithm ที่มี Block Length

(25)

$$\left[\begin{array}{c} \phi_0 \\ \phi_4 \\ \phi_2 \\ \hline \phi_3 \\ \phi_1 \\ \phi_5 \end{array}\right] = \left[\begin{array}{ccc|ccc} x_0 & x_4 & x_2 & x_3 & x_1 & x_5 \\ x_4 & x_2 & x_0 & x_1 & x_5 & x_3 \\ x_2 & x_0 & x_4 & x_5 & x_3 & x_1 \\ \hline x_3 & x_1 & x_5 & x_0 & x_4 & x_2 \\ x_1 & x_5 & x_3 & x_4 & x_2 & x_0 \\ x_5 & x_3 & x_1 & x_2 & x_0 & x_4 \end{array}\right] \left[\begin{array}{c} y_0 \\ y_4 \\ y_2 \\ \hline y_3 \\ y_1 \\ y_5 \end{array}\right]$$

เท่ากับ 2 ได้ดังนี้

(26)

$$\left[\begin{array}{cc} x_0 & x_1 \\ x_1 & x_0 \end{array}\right] \left[\begin{array}{c} y_0 \\ y_1 \end{array}\right]$$

$$= \left[\begin{array}{c} (x_0 + x_1)/2(y_0 + y_1) + (x_0 - x_1)/2(y_0 - y_1) \\ (x_0 + x_1)/2(y_0 + y_1) - (x_0 - x_1)/2(y_0 - y_1) \end{array}\right]$$

สำหรับ Cyclic Convolution ของ 2 ตัวประกอบเราจะกำหนดให้เป็น

$$\theta_0 = \left[\begin{array}{c} \phi_0 \\ \phi_4 \\ \phi_2 \end{array}\right] \qquad \theta_1 = \left[\begin{array}{c} \phi_3 \\ \phi_1 \\ \phi_5 \end{array}\right]$$

$$Y_0 = \left[\begin{array}{c} y_0 \\ y_4 \\ y_2 \end{array}\right] \qquad Y_1 = \left[\begin{array}{c} y_3 \\ y_1 \\ y_5 \end{array}\right]$$

(27)

$$X_0 = \left[\begin{array}{ccc} x_0 & x_4 & x_2 \\ x_4 & x_2 & x_0 \\ x_2 & x_0 & x_4 \end{array}\right] \qquad X_1 = \left[\begin{array}{ccc} x_3 & x_1 & x_5 \\ x_1 & x_5 & x_3 \\ x_5 & x_3 & x_1 \end{array}\right]$$

แล้วเราสามารถเขียนสมการ (25) ใหม่ได้ว่า

(28)

$$\begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} X_0 & X_1 \\ X_1 & X_0 \end{bmatrix} \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix}$$

ทำการแทนสมการ (28) นี้ลงในสมการ (26) เราจะได้ว่า

(29)

$$M_1 = (X_0 + X_1)/2(Y_0 + Y_1)$$
$$M_0 = (X_0 - X_1)/2(Y_0 - Y_1)$$
$$\theta_0 = M_1 + M_2$$
$$\theta_1 = M_1 - M_2$$

การคำนวณหาค่าของ M_1 และ M_2 เราใช้ ALGORITHM (22) ดังนั้นเมื่อนำมาใช้ กับสมการที่ (24) เราจะได้จำนวนการคูณเท่ากับ 8 และจำนวนการ บวกเท่ากับ 34

3.4 การทรานสฟอร์มฟูเรียร์ทางตรีโกณมิติ

เนื่องจากเมื่อค่าตัวประกอบที่เป็น Prime Number ของจำนวน N มีค่ามากกว่า 9 ในการแก้ปัญหาเพื่อหาคำตอบ จะทำได้ค่อนข้างยุ่งยาก เพราะจำนวนตัวแปรที่ใช้ จะมากตามไปด้วย ดังนั้นจึงใช้การทรานสฟอร์มฟูเรียด้วยวิธีการแยกทางตรีโกณมิติเข้าช่วย แต่อย่างไรก็ตาม ถ้าค่าตัวประกอบมีค่ามากกว่า 23 แล้ว ผลจะทำให้ความสามารถทางด้านความเร็วไม่ดี ซึ่งเป็นผลมาจากต้องใช้การคูณจำนวนจริงถึง $(p/2 - 1)^2$ เมื่อ p เป็นตัวประกอบที่เป็น Prime number ของ N ซึ่งสามารถแสดงได้ดังนี้

$$a_k + {}_1b_k = \sum_{j=0}^{p-1} (x_j + {}_1y_j)[\cos(2\pi jk/p) + {}_1\sin(2\pi jk/p)]$$

$$= x_0 + \sum_{j=1}^{p-1} x_j \cos(2\pi jk/p) - \sum_{j=1}^{p-1} y_j \sin(2\pi jk/p)$$

$$+ {}_1[y_0 + \sum_{j=1}^{p-1} y_j \cos(2\pi jk/p)$$

$$+ \sum_{j=1}^{p-1} x_j \sin(2\pi jk/p)]$$

$$= x_0 + \sum_{j=1}^{(p-1)/2} (x_j + x_{p-j}) \cos(2\pi jk/p)$$

$$- \sum_{j=1}^{(p-1)/2} (y_j - y_{p-j}) \sin(2\pi jk/p)$$

$$+ i [y_0 + \sum_{j=1}^{(p-1)/2} (y_j - y_{p-j}) \cos(2\pi jk/p)$$

$$+ \sum_{j=1}^{(p-1)/2} (x_j + x_{p-j}) \sin(2\pi jk/p)]$$

สำหรับ $k = 0,1,2,3,...,p-1.$ และเราจะได้ว่า

$$a_0 + ib0 = \sum_{j=0}^{p-1} (x_j + iy_j)$$

ซึ่งจะคำนวนโดยปราศจากการคูณ และ ค่าสัมประสิทธิ์ของฟูเรียที่เหลือสามารถแสดงได้ดังต่อไปนี้

$$a_k = a_k^+ - a_k^-$$

$$a_{p-k} = a_k^+ + a_k^-$$

$$b_k = b_k^+ + b_k^-$$

$$b_{p-k} = b_k^+ - b_k^-$$

สำหรับ $k = 0,1,2,3,4,...,(p-1)/2.$ เมื่อ

$$a_k^+ = x_0 + \sum_{j=1}^{(p-1)/2} (x_j + x_{p-j}) \cos(2\pi jk/p)$$

$$a_k^- = \sum_{j=1}^{(p-1)/2} (y_j - y_{p-j}) \sin(2\pi jk/p)$$

$$b_k^+ = y_0 + \sum_{j=1}^{(p-1)/2} (y_j - y_{p-j}) \cos(2\pi jk/p)$$

$$b_k^- = \sum_{j=1}^{(p-1)/2} (x_j + x_{p-j}) \sin(2\pi jk/p)$$

บทที่ 4.

ซอร์ท-คัท ดิสเครท ฟูเรียร์ ทรานสฟอร์ม

อัลกอริทัมต่าง ๆ ที่มีอยู่ในปัจจุบันนี้ ได้พัฒนามาจากหลักการ พื้นฐานเดียวกัน ซึ่งอัลกอริทัมเหล่านี้ ได้นำเอาคุณสมบัติเฉพาะ มาประยุกต์ใช้อย่างอย่างเหมาะสม จนมากระทั่งบัดนี้ คุณลักษณะเฉพาะก็แทบจะไม่เหลือ เป็นสัญญาณบอกให้ทราบว่า การพัฒนาได้มาถึงขีดสุดของมันแล้ว จึงเป็นการยากที่จะจะพัฒนาต่อไป แต่อย่างไรก็ตามเพื่อที่จะให้ได้อัลกอริทัมที่ดีที่สุด ในการทรานสฟอร์ม ฟูเรียร์ ก็จำเป็นที่จะต้องพัฒนาต่อไป เนื่องจากยังมีบางสิ่งที่แต่ละอัลกอริทัม นั้นแก้ปัญหาไม่ได้ ซึ่งเป็นสิ่งที่ไม่เหมือนกัน ดังนั้น ในบทนี้จะได้กล่าวถึงการนำเอาอัลกอริทัม ที่มีการพัฒนามาเกือบถึงขีดสุดแล้ว แต่พวกมันยังขาดบางสิ่งที่ไม่เหมือนกัน มาทำการรวมกันไว้ในอัลกอริทัมเดียว เพื่อทดแทนบางสิ่ง ที่แต่ละอัลกอริทัมไม่มี แม้ว่าจะไม่ได้ถึง 80 % แต่มันก็ทำให้ จุดที่บกพร่องของอัลกอริทัมใหม่น้อยลงกว่า อัลกอริทัมเดิมเหล่านั้น เนื้อหาที่จะพูดส่วนใหญ่จะเป็น แนวคิดที่จะนำเอาอัลกอริทัมหลายอัน มารวมกัน จะไม่กล่าวถึงหลักการของแต่ละอัลกอริทัม เพราะว่า หลักการของแต่ละอัลกอริทัม เหมือนเดิมทุกประการ ถ้าหากมีข้อสงสัยสามารถที่จะย้อนกลับไปดูบทที่ 2 ในเล่มนี้ได้

จากการศึกษา พบว่าการคำนวน ดิส เครท ฟูเรียร์ ทราสฟอร์ม จะทำได้เร็ว ค่าของ N จะต้องสามารถแยกออกเป็นตัวประกอบค่าเล็ก ๆ ซึ่งสามารถเขียนเป็นสูตรทางคณิตศาสตร์ ได้ดังนี้

$$N = \prod_{i=1}^{k} n_i \qquad , i = 1,2,\ldots,k$$

เมื่อ n_i เป็นตัวประกอบตัวที่ i ของ N และ k เป็นค่าเลขจำนวนของตัวประกอบทั้งหมดของ N และนอกเหนือจากนั้นก็ยังพบอีกว่า ค่าของ n_i จะต้องเป็นเลข Prime ที่มีค่าไม่ควรเกิน 23 เพราะว่าเวลาที่ใช้ในการคำนวนจะไม่ดี และ ต้องมีการจองหน่วยความจำมากที่ n_i มีค่ามาก สำหรับตัวแปรอีกทั้ง ยังทำให้โปรแกรมมีขนาดใหญ่ขึ้นอีกด้วย จากความรู้เบื้องต้นที่กล่าวมาทำให้พอมองเห็นแนวทางที่จะพัฒนา อัลกอรทัม ในการคำนวน DFT โดยนำเอาคุณลักษณะเฉพาะของ 3-ALGORITHM คือ COOLEY-TUKEY FFT, GOOD

THOMAS PRIME FFT , และ WINOGRAD'S SMALL BLOCK FFT ALGORITHM มารวมกันไว้ใน ALGORITHM เดียว และให้ชื่อมันว่า SHORT-CUT DFT ALGORITHM ซึ่งจะได้กล่าวรายละเอียดเพิ่มเติมต่อไป

สิ่งแรกที่เราควรจะทราบคือคุณลักษณะเฉพาะที่กล่าวถึงพอสังเขป เพื่อจะได้มองเห็นความเป็นไปได้ทุกแง่มุมของมัน

4.1. คุณลักษณะที่สำคัญ(PROPER PRINCIPAL CHARACTORS)

4.1.1 COOLEY-TUKEY ALGORITHM

อัลกอริทึม นี้สามารถคำนวน DFT ได้ ค่าของ N เป็นเลข PRIME หรือ N สามารถแยกตัวประกอบ ซึ่งตัวประกอบแต่ละตัวเป็น PRIME หรือจำ PRIME ยกกำลัง แทนเป็นสูตรทางคณิตศาสตร์ได้ดังนี้

$$N = \prod_{i=1}^{k} r_i \quad , i = 1,2,\ldots,k$$

เมื่อ r_i เป็นเลข PRIME หรือ PRIME ยกกำลังตัวที่ i และ k เป็นเลขจำนวนสูงสุดของตัวประกอบของ N

อย่างไรก็ตาม เมื่อเรามาพิจารณาที่สูตรของ COOLEY - TUKEY FFT ALGORITHM กรณีที่ง่ายที่สุด คือ N = r1.r2 เราสามารถที่จะทำการ MAP จาก 1 มิติ เป็น 2 มิติได้ โดยใช้สมการเชิงเส้นดังจะแสดงต่อไปนี้

$$x(n) = \sum_{k=0}^{N-1} x(k).w^{(nk)}$$

$$n = n1.r1 + n0$$

$$k = k1.r2 + k0$$

$$n.k = n1.k1.r1.r2 + n1.k0.r1 + k1.n0.r2 + n0.k0$$

แต่ $w^{(r1.r2).n1.k1} = 1$

ดังนั้น $w^{(n.k)} = w_{r1}^{(k1.n0)}.w_{r2}^{(n1.k0)}.w_N^{(n0.k0)}$

นั่นคือ

$$x(n1,n0)=\sum_{k0=0}^{r2-1}\left(w_N^{(n0.k0)}\sum_{k1=0}^{r1-1} x(k1,k0).w_{r1}^{(k1.n0)}\right)w_{r2}^{(n1.k0)}$$

แต่ถ้าเราตั้งข้อสมติฐานว่า เมื่อ N = r1.r2 เทอม $w_N^{(n0.k0)}$ ไม่น่าจะมี ซึ่งก็เป็นไปได้ ดังจะกล่าวถึงอีกครั้งในข้อต่อไป แต่ในกรณีของ เลข PRIME ยกกำลังแล้ว จะต้องมีเทอมนี้คูณเข้าหลังจากที่ได้หา r1-POINT DFT แล้วตามสมการซึ่งมันแสดงให้เห็นถึงคุณลักษณะเฉพาะได้อย่างชัดเจน อย่างไรก็ดีกรณีเช่นนี้จะทำให้ความเร็วในการคำนวน DFT ช้าลง

4.1.2 GOOD-THOMAS PRIME ALGORITHM

จากข้อสมติฐานที่ได้ตั้งขึ้นในข้อ 4.1.1 ในกรณีแรกที่กล่าวถึงสามารถที่จะแสดงให้เห็นอย่างแจ่มแจ้งว่าการคำนวน DFT ที่มีค่า N = r1.r2 นั้น สามารถที่จะทำการคำนวนเพียง r1-POINT DFT และ r2-POINT DFT เท่านั้น โดยการจัดเรียงลำดับข้อมูลทางอินพุทใหม่ก่อนที่จะทำการคำนวน DFTและหลังจากที่ทำการคำนวนเสร็จก็จะทำการจัดเรียงลำดับข้อมูลทางเอ้าท์พุท ซึ่งการจัดเรียงลำดับข้อมูลอินพุทและเอ้าท์พุทจะไม่เหมือนกัน แต่อย่างไรก็ตามทั้งสองนั้น ก็มีหลักการตั้งอยู่บน Chinese Remainder Theorem เพื่อความเข้าใจยิ่งขึ้น เราจะมาพิจารณาที่สูตรการคำนวนตาม GOOD THOMAS PRIME FFT ALGORITHM ที่เขาได้แสดงไว้

$$x(n) = \sum_{k=0}^{N-1} x(k).w^{(nk)}$$

การจัดเรียงลำดับทางอินพุท

$$i1 = n \pmod{r1}$$

$$i0 = n \pmod{r2}$$

$$i = i1.r2.s + i0.r1.t \pmod{N}$$

เมื่อ $r2.s + r1.t = 1 \pmod{N}$

การจัดเรียงลำดับทางเอาท์พุท

$$k1 = s.k \pmod{r1}$$

$$k0 = t.k \pmod{r2}$$

นั้นคือ

$$x(k1,k0) = \sum_{i1=0}^{r2-1} \left(\sum_{i0=0}^{r1-1} x(i1,i0).w_{r1}^{(i0.k0)} \right) w_{r2}^{(i1.k1)}$$

แต่อย่างไรก็ตามค่าตัวประกอบของ N จะอยู่ในขอบเขตที่จำกัดคือ ตัวประกอบทุกตัวจะต้องเป็น PRIME สัมพันธ์ (RELATIVELY PRIME) เขียนเป็นสูตรทางคณิตศาสตร์ได้ดังนี้

$$N = \prod_{i=1}^{k} r_i \quad , i = 1,2,\ldots,k$$

เมื่อ r_i เป็นเลข Prime ตัวที่ i และ k เป็นเลขจำนวนสูงสุดของตัวประกอบของ N โดยมีเงื่อนไขเพิ่มเติมว่า ตัวประกอบทุกตัวจะต้องเป็น Prime ที่ไม่ซ้ำกัน

4.1.3 WINOGRAD'S SMALL BLOCK FFT ALGORITHM

SMALL BLOCK FFT ALGORITHM เป็นวิธีการที่ทำการแก้ปัญหา ผลคูณของ DFT ที่มีค่าของ N เป็นเลข PRIME โดยยืนอยู่บนหลักการของ RADER'S PRIME ALGORITHM ที่ทำการลดผลคูณลงประมาณ 20 % ส่วนผลบวกจะเท่ากับจำนวนผลบวกที่ต้องการใน ALGORITHM อื่น นอกเหนือจากนั้นเขายังได้แสดงให้เห็นจริงว่า คำกล่าวของ GOOD-THOMAS ไม่สมบูรณ์ที่จริงแล้วตัวประกอบของ N จะเป็น RELATIVELY PRIME โดยได้ทำ SMALL BLOCK FFT เป็น 2,3,4,5,7,8,9, และ 16 เพื่อใช้ในการคำนวน DFT ที่มีค่า N มาก

อย่างไรก็ตามหลักการต่างๆก็ตั้งอยู่บน Chinese Remainder THEOREM เหมือนกันกับ GOOD-THOMAS PRIME ALGORITHM จากข้อคิดเกี่ยวกับคุณลักษณะเฉพาะที่สำคัญของแต่ละ ALGORITHM สามารถที่จะนำมาตั้งเป็นกฎเกณฑ์ได้ 2 ข้อ

4.2 กฎเกณฑ์ของ SHORT-CUT DFT

4.2.1 การแยกตัวประกอบของ N

ค่า N สามารถที่จะแยกออกเป็นตัวประกอบ สามารถแสดงได้เป็นสูตรทางคณิตศาสตร์ได้ดังต่อไปนี้

$$N = \prod_{i=1}^{k} n_i \quad , \; i = 1,2,..,k$$

ดังนั้นค่า N สามารถที่จะแจกแจงได้ 2 แบบคือ

- ค่าตัวประกอบ n_i แต่ละตัวเป็นเลข PRIME ยกกำลังที่ ค่าเลขจำนวนของ PRIME ไม่ซ้ำกัน

ตัวอย่างเช่น N = 1000 = $(2^4) \cdot (5^4)$ เมื่อ $n1 = 2^3$ และ $n2 = 5^3$

- ค่าตัวประกอบ ni ทุกตัวเป็น RELATIVELY PRIME

ตัวอย่างเช่น N = 1000 = (2x5)(2x5)(2x5) เมื่อ n_1 = 2x5, n_2 = 2x5, และ n_3 = 2x5

4.2.2 การเลือก SMALL BLOCK FFT

SMALL BLOCK FFT จะใช้ BLOCK LENGHT เป็น 2,3,4,5,7,8, และ 9 ที่สร้างมาจาก WINOGRAD ' S SMALL BLOCK FFT ALGORITHM และ นอกเหนือจากนี้ก็เป็น SMALL BLOCK DFT ที่เป็น BLOCK พิเศษ สามารถคำนวนหา DFT ที่เป็นเลข PRIME คือ 11,13,17,19, และ 23 โดยใช้หลักการทรานสฟอร์มฟูเรียทางตรีโกณมิติ ค่า PRIME ที่มีค่ามากกว่า 23 จะไม่ใช้เนื่องจาก เป็นตัวประกอบที่มีค่ามากเกินไป และยิ่งไปกว่านั้น ยังต้องการหน่วยความจำมาก ในการเก็บค่าที่ต้องการใช้ ในการทรานสฟอร์มฟูเรียร์ ซึ่งจะไม่เกิดผลดีจึงไม่นำมาใช้

4.3 OVERVIEW OF THE SHORT-CUT DFT ALGORITHM

SHORT-CUT DFT ALGORITHM เป็นการนำเอาหลักการของ COOLEY TUKEY และ GOOD-THOMAS PRIME FFT ALGORITHM มาใช้รวมกัน ซึ่งสามารถที่จะทำได้ 2 แบบ ตามกฎเกณฑ์การแยกตัวประกอบของ N

แบบที่ 1.

$$N = \prod_{i=1}^{k} r_i \quad , i=1,2,\ldots,k$$

ค่าตัวประกอบ r_i แต่ละตัวเป็นเลข PRIME ยกกำลัง ที่ค่าเลขจำนวนของ PRIME ไม่ซ้ำกัน

ในการอธิบายจะใช้ที่ง่ายที่สุด คือ 2 มิติ สมมติว่า N = 100 ดังนั้น $r_1 = 4$ และ $r_2 = 25 = 5^2$

ขั้นตอนการคำนวน(Step by step of computation)

1. ทำการ MAP ข้อมูลอินพุทตาม GOOD THOMAS PRIME FFT ALGORITHM จาก 1->2 มิติ แบบ 1:2 ,ผลลัพธ์ข้อมูลอินพุทจะจัดเรียงลำดับกันใน Array 2 มิติขนาด 4 แถว 25 หลัก
2. ตั้งค่า i = 1
3. เก็บค่าข้อมูลอินพุทหลักที่ i ทุกแถวลง STACK
4. เรียกโปรแกรมย่อยคำนวน 4-point DFT
5. คืนค่าข้อมูลจาก STACK ไปไว้ที่เดิม
6. เพิ่มค่า i ขึ้นหนึ่ง
7. ถ้า i น้อยกว่า 25 จริง ไปข้อ 3.
8. ทำการ TRANSPOSE จากแถวเป็นหลัก
9. ตั้งค่า i = 1
10. เก็บค่าข้อมูลอินพุทหลักที่ i ทุกแถว ลง STACK
11. เรียกโปรแกรมย่อยคำนวน 25-point DFT
12. คืนค่าข้อมูลจาก STACK ไปไว้ที่เดิม
13. เพิ่มค่า i ขึ้นหนึ่ง
14. ถ้า i น้อยกว่า 4 จริง ไปข้อ 10.
15. ทำการ MAP ข้อมูล ตาม GOOD THOMAS PRIM FFT ALGORITHM จาก 2->1 มิติ ผลลัพธ์ข้อมูลเอ้าท์พุทจะจัดเรียงลำดับกันใน Array 1 มิติ

จะเห็นว่าใช้ GOOD THOMAS PRIME FFT ALGORITHM เป็นหลักในการคำนวน ในขั้นที่ 4 และ 11 เป็นการเรียกโปรแกรมย่อยขนาด 4-POINT และ 25-POINT DFT จากกฎเกณฑ์การเลือก SMALL BLOCK FFT ค่าของ BLOCK LENGHT = 4 เราสามารถคำนวนหาได้โดยตรง แต่ค่า BLOCK LENGHT=25 ไม่สามารถหาค่าได้โดยตรง เนื่องจากค่า 25 = 5x5 เราสามารถทำการคำนวนโดยใช้ COOLEY-TUKEY FFT ALGORITHM เนื่องจากค่าตัวประกอบเป็นเลข PRIME ยกกำลัง และนอกจากนี้แล้วได้แสดงเป็น Flow Chart ในรูปที่ 4.1

แบบที่ 2.

$$N = \prod_{i=1}^{k} r_i \quad , i=1,2,\ldots,k$$

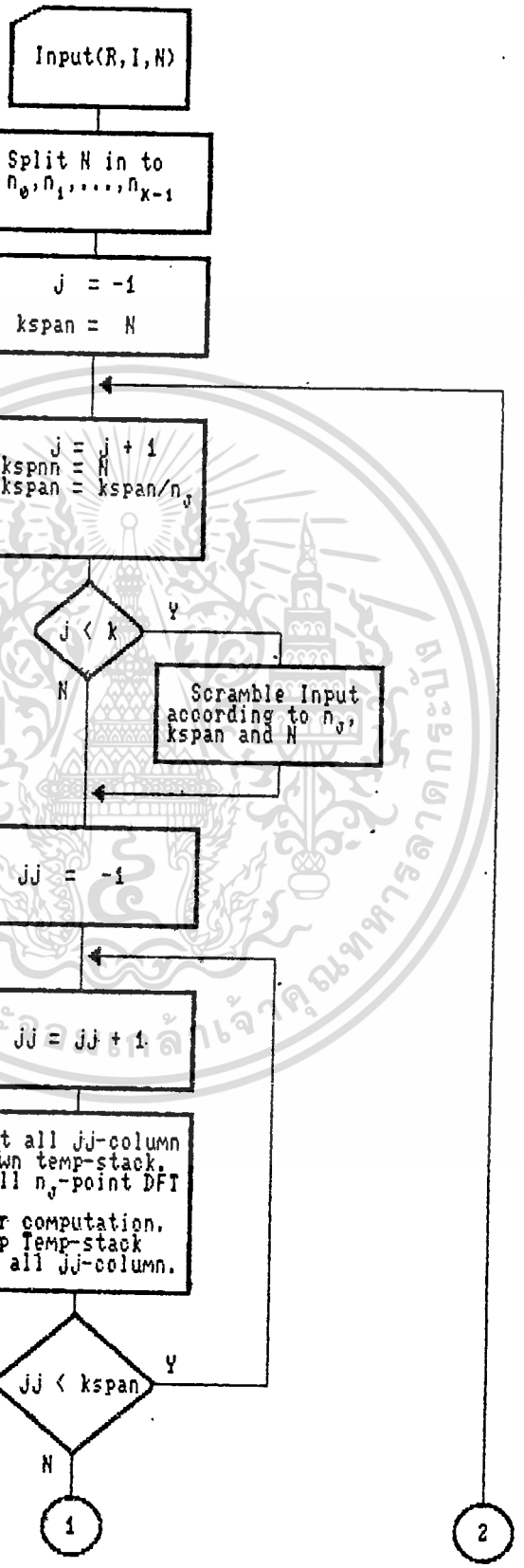
ค่าตัวประกอบ r_i ทุกตัวเป็นเลข RELATIVELY PRIME ในการอธิบายจะใช้ SIMPLE CASE คือ 2 มิติ สมมติว่า N = 100 ดังนั้น $r_1 = 2x5$ และ $r_2 = 2x5$

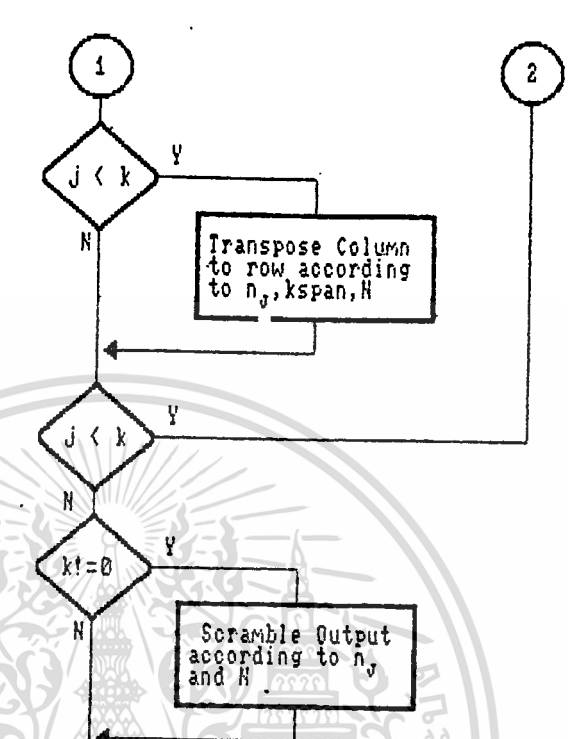
ขั้นตอนการคำนวน(Step by step of computation)

1. ทำการ MAP ข้อมูลอินพุท ตาม COOLEY-TUKEY FFT ALGORITHM จาก 1->2 มิติ แบบ 1:1, ผลลัพธ์ข้อมูลอินพุท จะจัดเรียงลำดับกันใน Array 2 มิติขนาด 10 แถว 10 หลัก
2. ตั้งค่า i = 1
3. เก็บค่าข้อมูลอินพุทหลักที่ i ทุกแถว ลง STACK
4. เรียกโปรแกรมย่อยคำนวน 10-POINT DFT
5. คืนค่าข้อมูลจาก STACK ไปไว้ที่เดิม
6. เพิ่มค่า i ขึ้นหนึ่ง
7. ถ้า i น้อยกว่า 10 จริง ไปข้อ 3.
8. ทำการ TRANSPOSE จากแถวเป็นหลัก
9. MULTIPLY BY ROTATION FACTOR (--->$w_N^{(n_0 \cdot k_0)}$)
10. ตั้งค่า i = 1

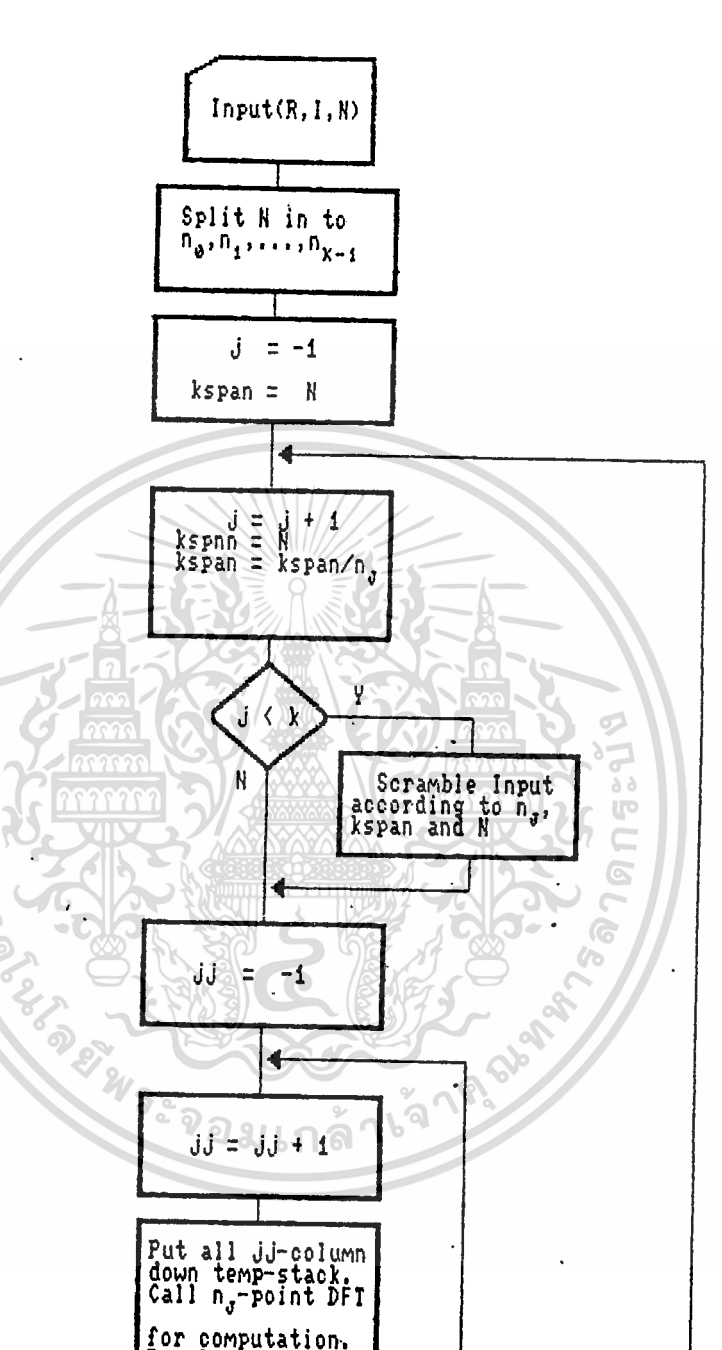
11. เก็บค่าข้อมูลอินพุทหลักที่ i ทุกแถว ลง STACK
12. เรียกโปรแกรมย่อยคำนวน 10-POINT DFT
13. คืนค่าข้อมูลจาก Stack ไปไว้ที่เดิม
14. เพิ่มค่า i ขึ้นหนึ่ง
15. ถ้า i น้อยกว่า 10 จริง ไปข้อ 10.
16. ทำการ Map ข้อมูล ตาม COOLEY - TUKEY FFT ALGORITHM จาก 2->1 มิติ ผลลัพธ์ข้อมูลเอ้าท์พุทจะจัดเรียงลำดับกันใน ARRAY 1 มิติ

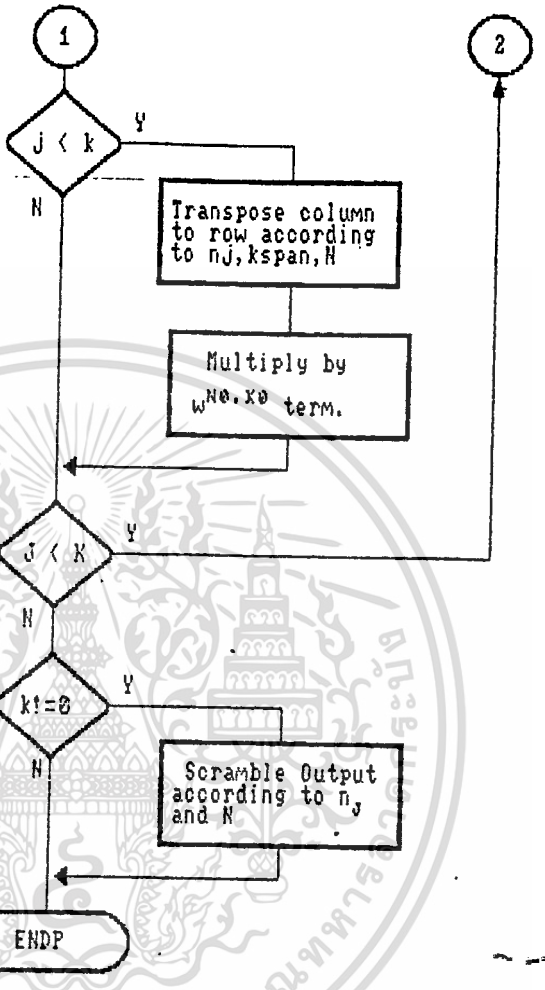
จะเห็นว่าใช้ COOLEY-TUKEY FFT ALGORITHM เป็นหลัก ในการคำนวน ในขั้นที่ 4 และ 12 เป็นการเรียกโปรแกรมย่อยขนาด 10-POINT และ 10-POINT DFT จากกฎเกณฑ์ การเลือก SMALL BLOCK FFT ค่า BLOCK LENGHT = 10 ไม่มี จึงไม่สามารถหาค่าได้โดยตรง ดังนั้น เนื่องจากค่า 10 = 2x5 เราสามารถทำการคำนวนโดยใช้ GOOD THOMAS PRIME FFT ALGORITHM เนื่องจาก ค่าตัวประกอบเป็น RELATIVELY PRIME และ ได้แสดงเป็น Flow Chart ได้ดังรูปที่ 4.2





รูปที่ 4.1 TYPE 1 SHORT-CUT DFT FLOW CHART.





รูปที่ 4.2 TYPE 2 SHORT-CUT DFT FLOW CHART.

4.4 โครงสร้างของโปรแกรม

โครงสร้างของโปรแกรมมีส่วนสำคัญ 3 ส่วนดังต่อไปนี้

- SMALL BLOCK FFT
- MULTIPLY BY ROTATION FACTOR
- PERMUTATION

4.4.1 SMALL BLOCK FFT

จะมี BLOCK LENGTH เป็น 2,3,4,5,7,8,9 และ BLOCK พิเศษที่ใช้คำนวน DFT ที่มี BLOCK LENGTH 11,13,17,19,และ23

Small Block FFT ที่มี BLOCK LENGTH,2,3,4,5,7,8,9 สามารถแสดงจำนวนการคูณและการบวกได้ดังในตารางที่ 4.1

BLOCK LENGTH	ADDITION $A_c(n)$	MULTIPICATION $M_c(n)$
2	2	-
3	12	4
4	16	2
5	34	10
7	72	16
8	52	6
9	90	20

ตารางที่ 4.1

สำหรับ BLOCK LENGTH 11,13,17,19, และ23 สามารถคำนวนได้จาก

จำนวนการบวก = $4(p/2 - 1)$ และ คูณด้วย $(p/2 - 1)$

จำนวนการคูณ = $((p-1)/2)^2$

4.4.2 MULTIPLY BY ROTATION FACTOR

จะใช้ใน COOLEY-TUKEY ALGORITHM การหมุนของมุม θ จะเป็นในลักษณะที่เรียกว่า DECIMATION IN FREQUENCY เราจะสังเกตุเห็นได้จาก SIGNAL DIAGRAM ค่าของมุมจะเป็นดังนี้คือ θ , 2θ , 3θ , 4θ,.. และ 2θ, 4θ, 6θ,... ดังนั้นเราจะนำสูตรทางตรีโกณมาประยุกต์ใช้ในคำนวนค่า COSINE และ SINE ที่เกิดการหมุนของมุมที่เพิ่มขึ้นอย่างที่เรียกว่า DECIMATION IN FREQUENCY สูตรที่ใช้มี 2 สูตร

$$\cos(\alpha+\beta) = \cos(\alpha)\sin(\beta) + \cos(\beta)\sin(\alpha)$$

$$\sin(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$$

สาเหตุที่ใช้วิธีนี้เพื่อที่จะลดการใช้หน่วยความจำสำหรับเก็บค่า COSINE และ SINE แต่ในด้านความเร็วจะช้าลงไม่มากจนไม่นำมาคิด

เพื่อที่จะแสดงให้กระจ่างขึ้น เพื่อที่จะนำเอาหลักการที่กล่าวมาไปใช้สามารถที่จะทำความเข้าใจได้โดยการไล่ขั้นตอน ตาม FLow Chart รูปที่ 4.3. ควบกับ SIGNAL DIAGRAM

SYMBOLIC DEFINITION

c หมายถึงมีค่า $2.0 \times \sin^2(\theta/2)$ ซึ่งค่าของ cosine ของมุม θ มีค่าเท่ากับ $1.0 - 2.0 \times \sin^2(\theta/2)$

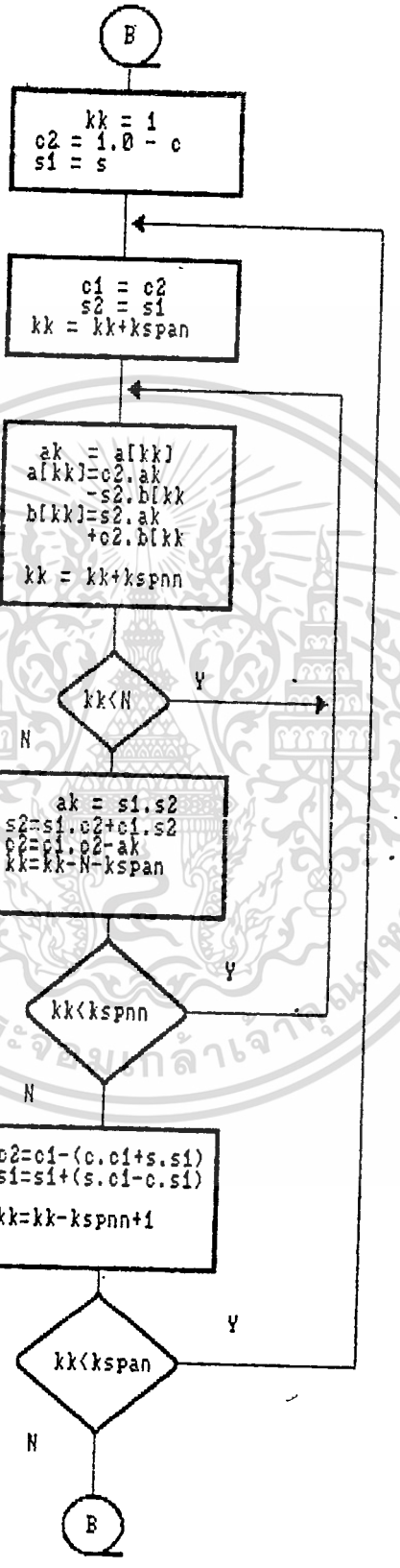
s หมายถึงค่าของ $\sin(\theta)$

c1 หมายถึงค่าของ cosine ของมุมที่เป็น BASE การหมุนของมุม θ

s1 หมายถึงค่าของ sine ของมุมที่เป็น BASE การหมุนของมุม θ

c2 หมายถึงค่าของ cosine ของมุมที่เป็นผลจากการหมุนของมุม θ ขึ้นอยู่กับมุม BASE

s2 หมายถึงค่าของ sine ของมุมที่เป็นผลจากการหมุนของมุม θ ขึ้นอยู่กับมุม BASE



รูปที่ 4.3 การคูณด้วย ROTATION FACTOR.

ถ้าดูจาก SIGNAL FLOW GRAPH และ FLOW CHART รูปที่ 4.3 ถ้าสมมติว่า ค่า N = 12,n0 = 3 และ n1 = 4

ขั้นตอน

1. ค่า kspan = 12
2. ค่า kspnn = kspan
3. kspan = kspan / 3
4. คำนวน 3-point DFT เป็นจำนวน 4 ครั้ง

หลังจากนี้ก็จะเป็นขั้นตอนที่ได้กล่าวถึง คือ การคูณด้วย Rotation factor($w^{n0.k0}$) ซึ่งสามารถที่จะทำความเข้าใจ Step by Step ตาม Flow Chart ที่ 4.3 ผู้เขียนเองได้ตระหนักว่า ผู้อ่านสามารถที่จะทำความเข้าใจได้อย่างง่าย จึงจะขอกล่าวเรื่องนี้ไว้เพียงเท่านี้

4.4.4 PERMUTATION

PERMUTATION เป็นการจัดเรียงลำดับข้อมูลให้อยู่ในตำแหน่งที่ต้องการ ในที่นี้จะใช้วิธีการ MAP ลำดับที่ของข้อมล ที่ต้องการลงใน ARRAY มีขนาด N หน่วยที่เป็น Array แบบข้อมูลจำนวนเต็ม ซึ่งทำให้ประหยัดหน่วยความจำครึ่งหนึ่งของ N ซึ่งปกติจะต้องใช้หน่วยความจำสำหรับขอ้มูล และหน่วยความจำที่เป็น TEMPOLARY MEMORY ถึง 2N หน่วยที่เป็น FLOATING POINT วิธีการจะพิจารณาจากตารางผลลัพธ์ที่จัดเรียงลำดับข้อมูลแล้ว จะแสดงให้เห็นได้ดังนี้

กำหนดตารางการ MAP ข้อมูลจาก ARRAY 2 มิติ ไปเป็น ARRAY ขนาด I มิติเป็นดังนี้จะได้ว่า

		0	1	2	3
3->1	0	0	3	6	9
9->3	4	1	4	7	10
13->9	8	2	6	8	11
11->13					
5->1					
1->3					

มันจะเป็น LOOP จากคุณสมบัตินี้ สามารถที่จะนำมาใช้ในการจัดเรียงลำดับข้อมูลที่ต้องการ โดยจะต้องมีลำดับการจัดเรียงข้อมูลในตารางการ MAP จากที่กล่าวมาข้างต้นสามารถที่จะแสดงให้เห็นชัดโดยจะใช้ตัวแปรแทน

$A_J = A_1$

$A_1 = A_3$

$A_3 = A_9$

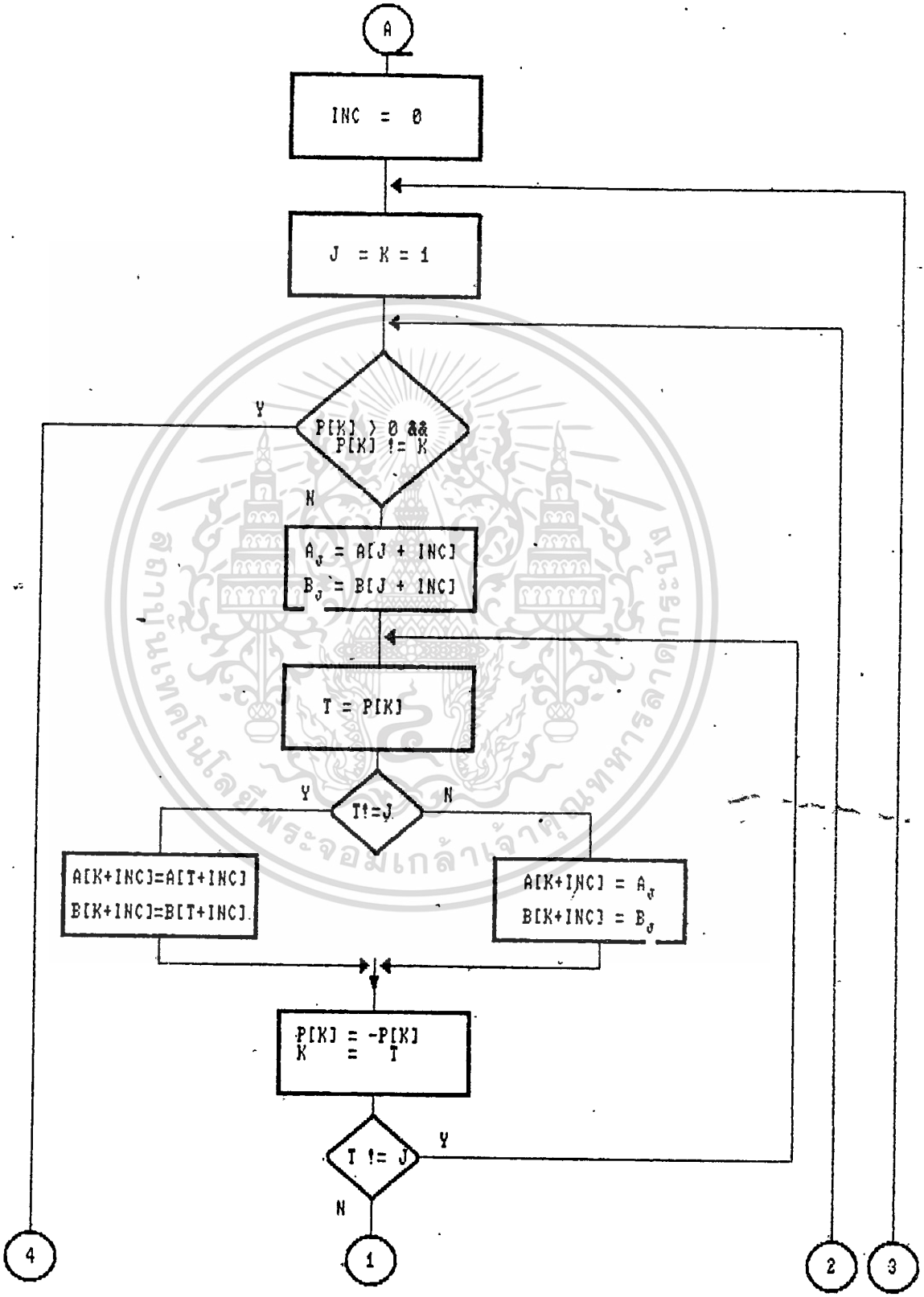
$A_9 = A_{13}$

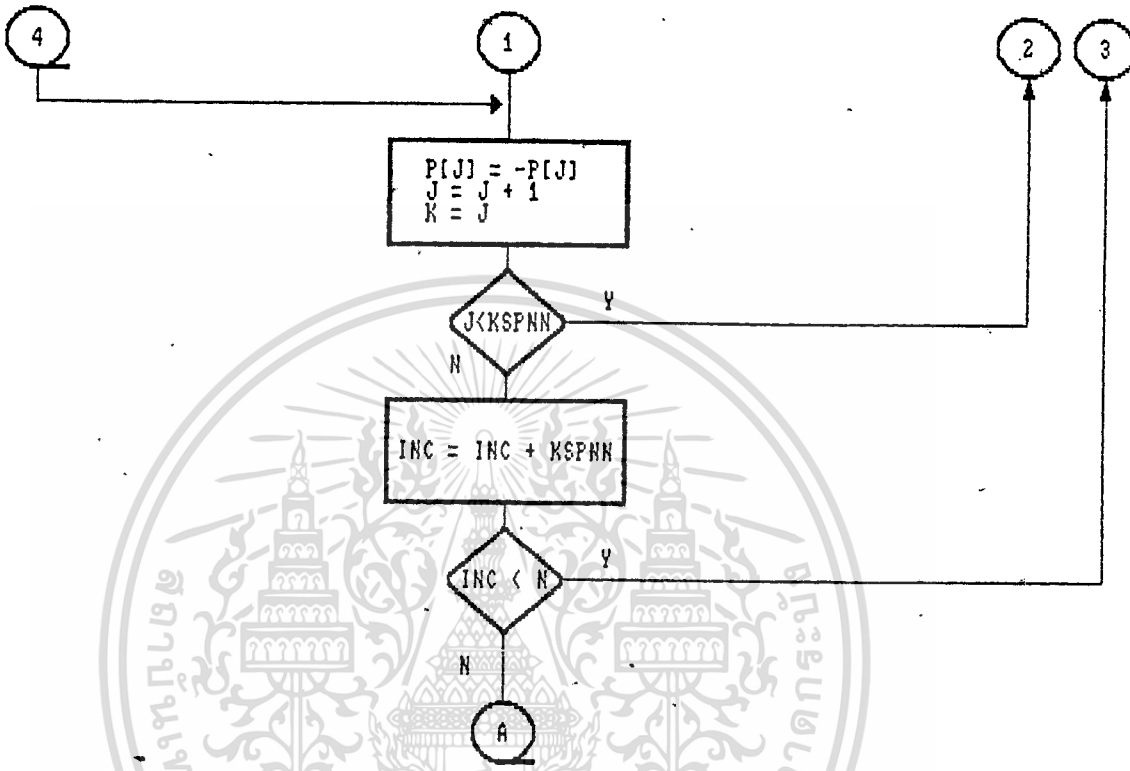
$A_{13} = A_{11}$

$A_{11} = A_5$

$A_5 = A_J$

เพื่อที่จะแสดงให้เห็นการใช้ คุณสมบัติอันนี้ เราจะแสดงใน Chartรูปที่ 4.4 มันง่ายมากที่จะทำความเข้าใจ โดยการไล่ที่ละ Step เพราะเป็น Flow Chart ที่เขียนแบบตรงไปตรงมาตามหลักการที่ได้กลาวถึง มีข้อที่เพิ่มเติม คือเมื่อเราทำการจัดเรียงข้อมูล ที่ LOOP ไหน จะต้องมีการ mark เพื่อเป็นสัญญาณบอกว่าได้ทำการจัดเรียงแล้ว เนื่องจากการที่เราไม่อาจทราบได้เลยว่าจะมีกี่ LOOP และได้ทำการจัดเรียงหรือยัง คิดว่ามันง่ายมากที่จะ ทำความเข้าใจจึงขอกล่าวเพียงเท่านี้





รูปที่ 4.4 FLOW CHART OF THE PERMUTATION.

4.5 การนับจำนวนผลคูณ และผลบวก

เพื่อที่จะลดความยุ่งยากของการนับจำนวนผลคูณ และผลบวกเราจะมาพิจารณากรณีที่ง่ายที่สุดคือ 2 มิติ

สมมติว่า N=100 สามารถแยกตัวประกอบได้เป็น 2 แบบ ตามกฎการแยกตัวประกอบของ N

แบบที่ 1. จะได้ $n_0 = 2^2$ และ $n_1 = 5^2$ สามารถที่จะหาจำนวนผลคูณ และผลบวกตามขั้นตอนดังต่อไปนี้

ตัวแปรที่ควรจะทราบคือ

$M_c(n)$ คือจำนวนผลคูณของ n-Point Small Block FFT

$A_c(n)$ คือจำนวนผลบวกของ n-Point Small Block FFT

ซึ่งแสดงไว้ในตารางที่ 4.1

1. เนื่องจากในการคำนวณจะต้องคำนวณ n_0-Point DFT ก่อน แต่ค่า $n_0 = 4 = 2 \times 2$ ดังนั้นจะต้องคิดจำนวนผลคูณและผลบวก 2 ขั้นตอน

1.1 ต้องคำนวน 2 - Point DFT แล้วทำการคูณด้วย Rotation Factor หลังจากนั้นก็คำนวณ 2-Point DFT

- คำนวณ 2-Point DFT จำนวนผลคูณ และผลบวกแสดงได้ดังนี้

จำนวนผลคูณ $= 2 \times M_c(2)$

$= 2 \times 0$

$= 0$

จำนวนผลบวก $= 2 \times A_c(2)$

$= 2 \times 2$

$= 4$

- ทำการคูณด้วย Rotation Factor หนึ่งครั้ง เนื่องจากทำการคำนวณ 2-Point DFT

จำนวนผลคูณที่เพิ่มขึ้น = 2 (image data)

- ทำการคำนวณ 2-Point DFT จะได้จำนวนผลคูณ และผลบวกเท่ากับข้อ 1.1.1

1.2 ทำข้อ 1.1 เป็นจำนวน n_1 ครั้ง

1.3 คำนวน 5-point DFT,คูณด้วย ROTATION FACTOR หลังจากนั้นก็คำนวณ 5-Point DFT

- คำนวณ 5-Point DFT จำนวนผลคูณและผลบวกแสดงได้ดังนี้

จำนวนผลคูณ = $5 \times M_c(5)$
= 5×10
= 50

จำนวนผลบวก = $5 \times A_c(5)$
= 5×34
= 170

- ทำการคูณด้วย Rotation Factor จำนวน 16 ครั้ง เนื่องจากทำการคำนวณ 5-Point DFT

จำนวนผลคูณที่เพิ่มขึ้น = 16

- ทำการคำนวณ 5-Point DFT จะได้จำนวนผลคูณ และผลบวกเท่ากับที่หาแล้วข้างบน

- ทำข้อ 1.3 เป็นจำนวน n_0 ครั้ง

เพราะฉะนั้น

จำนวนผลคูณทั้งหมด = (2 x 2 x 25) + (100 + 32) x 4
= 630

จำนวนผลบวกทั้งหมด = (4 x 25) + (170 x 4)
= 100 + 680
= 780

แบบที่ 2.

จะได้ $n_0 = 2 \times 5$ และ $n_1 = 2 \times 5$ สามารถที่จะหาจำนวนผลคูณและผลบวกตามขั้นตอนดังต่อไปนี้

2.1 การคูณด้วย Rotation Factor เนื่องจาก $n_0 = n_1 = 10$ ดังนั้นจะได้ว่าจำนวนผลคูณด้วย Rotation Factor (Mr) จะเป็น

$Mr = 2 \times (n_0-1)(n_1-1)$ (image data)
$= 2 \times 9 \times 9$
$= 162$

2.2 พิจารณา n_0 จะได้ว่าต้องทำการคำนวณ 2-Point DFT เป็น

จำนวน 5 ครั้ง และหลังจากนั้นจะทำการคำนวณ 5-Point DFT เป็นจำนวน 2 ครั้ง

จำนวนผลคูณ = $M_c(2) \times 5 + M_c(5) \times 2$
= $0 \times 5 + 10 \times 2$
= 20

จำนวนผลบวก = $A_c(2) \times 5 + A_c(5) \times 2$
= $(2 \times 5) + (34 \times 2)$
= 78

2.3 ทำข้อ 2.2 ซ้ำเป็นจำนวน 10 ครั้ง

ดังนั้นจะได้ว่า

จำนวนผลคูณทั้งหมด = 162 + (10 x 20)
= 362

จำนวนผลบวกทั้งหมด = 78 x 10
= 780

จะเห็นว่า Short-Cut DFT แบบที่ 1 จะมีจำนวนผลคูณมากกว่าแบบที่ 2 จำนวนผลบวกจะเท่ากัน ดังนั้นในการสร้าง Program จึงใช้แบบที่ 2

บทที่ 5.

สรุป

1.ในการทำการทรานสฟอร์มูเรียร์ แบบเร็ว จะทำได้ก็ต่อเมื่อสามารถที่จะแยก N ออกเป็นตัวประกอบ ที่แต่ละค่าของมันมีค่าน้อยๆ และ ตัวประกอบนั้นจะเป็น PRIME NUMBER

2.COOLEY-TUKEY FFT ALGORITHM สามารถที่จะหาค่าได้ทุกค่าของ N ที่สามารถแยกตัวประกอบได้ ไม่ว่าจะเป็น PRIME ยกกำลัง หรือ อื่นๆ แต่มันจะต้องมีการปรับค่าโดยการคูณด้วย ROTATION FACTOR ทำให้ประสิทธิภาพทางด้านความเร็วไม่ดี

3.GOOD THOMAS PRIME ALGORITHM สามารถที่จะหาค่า N ได้เฉพาะค่าที่ตัวประกอบของ N ทุกตัวเป็น PRIME สัมพันธ์ ทำให้การหาค่า N จำกัดกว่า COOLEY-TUKEY ALGORITHM แต่ทางด้านความเร็วจะดีกว่า

4.SHORT-CUT DFT ALGORITHM เป็นกรรมวิธีใหม่ที่มีความสามารถทำการทรานสฟอร์มูเรียได้อย่างมีประสิทธิภาพ กล่าวคือทางด้านความเร็วจะอยู่ระหว่างความเร็วของ COOLEY-TUKEY FFT ALGORITHM และ GOOD THOMAS PRIME FFT ALGORITHM และทางด้านการคำนวนค่า สามารถที่หาค่าเฉพาะของ N เท่าที่ COOLEY-TUKEY ALGORITHM สามารถหาได้

เพื่อแสดงความแตกต่างทางด้านความเร็วของ SHORT-CUT DFT และ COOLEY-TUKEY FFT เราได้ทำการทดสอบประมวลผลบนเครื่อง MICROCOMPUTER ที่ CLOCK SPEED เป็น 16 Mhz ดังตารางที่ 4.3

จำนวน N	COOLEY-TUKEY FFT	SHORT-CUT DFT
1000	13 s	12 s
900	11 s	9 s
800	9 s	8 s
700	9 s	7 s
600	7 s	6 s
500	6 s	5 s
400	4 s	4 s
300	3 s	3 s
200	2 s	1 s
100	1 s	น้อยกว่า 1 s

ตารางที่ 4.3 การเปรียบเทียบความเร็วระหว่าง COOLEY-TUKEY FFT และ SHORT-CUT DFT ALGORITHM

กิตติกรรมประกาศ

หนังสือฉบับนี้เสร็จสมบูรณ์ได้ ด้วยความอนุเคราะห์จากอาจารย์ทุกท่าน และเพื่อนๆทุกคน โดยเฉพาะอาจารย์ เกษตร์ ศิริสันติสัมฤทธิ์ อาจารย์ที่ปรึกษา ที่ได้ให้ข่าวสารข้อมูลที่จำเป็น ตลอดจนคำแนะนำต่างๆ จึงขอขอบคุณทุกท่านไว้ ณ.โอกาสนี้ด้วย

หนังสืออ้างอิง

1. Rabiner,Lawrence R ," Digital Signal Processing ", IEEE.,518 P., 1972
2. K.G. Beauchamp ," Transform for Engineers A Guide to Signal Processing ", OXFORD.,269 P.,1987
3. Richard E. Blahut ," Fast Algorithm For Digital Signal Processing ", ADDISON - WESLEY ,NEW YORK., 441 P.,1987