การออกแบบสร้างวงจรกรองเชิงเลขอันดับ 4

A Fourth Order Digital Filter Implementation

โดย

นาย ชานนท์ พนาวรรต

นาย สุรพงษ์ วัฒนา

นาย อัครพล ตรีรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2536

การออกแบบสร้างวงจรกรองเชิงเลขอันดับ 4

A Fourth Order Digital Filter Implementation

โดย นาย ชานนท์ พนาวรรต
นาย สุรพงษ์ วัฒนา
นาย อัครพล ตรีรัตน์

อาจารย์ที่ปรึกษา ผศ.ดร. กอบชัย เดชหาญ

บทคัดย่อ

บทความนี้จะเกี่ยวกับวงจรกรองสัญญาณเชิงเลข (digital filter) ซึ่งเป็นแขนงหนึ่งของการประมวลผลสัญญาณเชิงเลข (digital signal processing) โดยปกติแล้วจะเป็นวิชาที่เกี่ยวข้องกับคณิตศาสตร์ในรูปทฤษฎี และ อัลกอลิทึมต่างๆ แต่ในบทความนี้จะแสดงให้เห็นถึง การนำเอาฮาร์ดแวร์เข้าไปประยุกต์ใช้กับงานการกรองเชิงเลขป้อนกลับอันดับที่ 4 โดยวิธีการหนึ่งที่เรียกว่า การกระจายทางคณิตศาสตร์ (Distributed Arithmetic) ซึ่งเป็นวิธีที่ทำให้ฟังก์ชันถ่ายโอน หรือ ทรานสเฟอร์ฟังก์ชัน (Transfer function) ของวงจรกรองสัญญาณเชิงเลขแปลงไปสู่วงจรทางอิเลคทรอนิกส์ได้

ABSTRACT

This paper describes about the digital filtering,which includes in digital signal processing. In general DSP presents by arithemetics and algorithms.This paper demonstrates an implementation of 4th order digital filter which is presented by distributed arithemetics.Distributed arithemetics is a process that transforms transfer function to look-up table digital hardware.

สารบัญ

บทนำ

การออกแบบสร้างวงจรกรองเชิงเลขที่นิยมสร้างกันมีอยู่ 2 วิธี คือ

1. สร้างโดยวิธีซอฟแวร์ (Software) วิธีนี้เป็นการจำลองเลียนแบบ (Simulation) การทำงาน หรือ การประมวลผลของตัวกรองเชิงเลขลักษณะนี้เราเรียกว่าเป็นการทำงานในระบบเวลาไม่จริง (non-real time system)

2. สร้างโดยฮาร์ดแวร์ (Hardware implementation) วิธีนี้เป็นการสร้างวงจรกรองเชิงเลขให้ทำงาน หรือ ประมวลผลในระบบเวลาจริง (real time system) ซึ่งมีขั้นตอนต่างๆในการออกแบบดังนี้

(ก) เลือกโครงสร้างของตัวกรอง

(ข) เลือกการคำนวณระหว่างการประมวลผลโดยใช้รูปแบบจำนวนโดยตรง (Fixed-point Arithmetic) และ การประมวลผลโดยใช้รูปแบบของระบบตัวเลขอ้างอิงดรรชนี (Floating Arithmetic)

(ค) เลือกระบบตัวเลขที่ใช้แทนค่าต่างๆในสมการ เช่น เลือกระบบตัวเลขแบบส่วนเติมเต็มเป็น 1 (one's complement) ระบบตัวเลขแบบส่วนเติมเต็มเป็น 2 (two's complement) หรือระบบตัวเลขแบบอิงดรรชนี เป็นต้น

(ง) เลือกการประมวลผลว่าเป็นแบบใดเช่น แบบอนุกรม (cascade) หรือแบบขนาน (parallel) เป็นต้น

(จ) เลือกอุปกรณ์ที่เหมาะสมในการสร้าง เช่น IC ประเภท TTL, CMOS ซึ่งขึ้นอยู่กับความต้องการของผู้สร้าง เช่น ความเร็วในการประมวลผล การประหยัดกำลังไฟฟ้าที่ป้อนแก่ตัวกรอง เป็นต้น

ซึ่งส่วนประกอบต่างๆที่ต้องนำมาพิจารณา คือ ส่วนที่ใช้เก็บข้อมูล (Storage) ส่วนประมวลผลทางคณิตศาสตร์ (Arithmetic) ส่วนที่เกี่ยวข้องกับสัญญาณเข้าและออก (Input and Output) รวมถึงส่วนควบคุมการทำการ (Control)

ซึ่งขั้นตอนเหล่านี้จะขึ้นอยู่กับระเบียบวิธีการ (Algolithm) ของตัวกรองเป็นสำคัญ

ในการออกแบบในบทความนี้จะเลือกการคำนวณโดยใช้รูปแบบของระบบตัวเลขแบบ รูปแบบจำนวนโดยตรง (Fixed-point Arithmetic) และใช้ระบบตัวเลขที่ใช้แทนจำนวนต่างๆเป็นแบบ ส่วนเติมเต็มเป็น 2 (two's complement) และใช้การประมวลผลเป็นแบบอนุกรมกัน (cascade)

บทที่1. ระบบตัวเลขที่ใช้

สำหรับระบบเชิงเลข ตัวเลข หรือจำนวนเลขต่างๆมักถูกแทนด้วยเลขฐานสอง ซึ่งโดยทั่วไปนิยมใช้กันสองรูปแบบ คือ รูปแบบจำนวนโดยตรง (fixed point format) และ รูปแบบจำนวนอิงดรรชนี (floating point format) ซึ่งในทั้งสองรูปแบบนี้ แบบจำนวนโดยตรงจะมีวงจรฮาร์ดแวร์ที่ใช้ในการคำนวณง่ายกว่าแต่ให้ค่าการคูณค่อนข้างจำกัด ส่วนรูปแบบจำนวนอิงดรรชนีแทนค่าของสัญญาณได้มากกว่า แต่ต้องใช้วงจรฮาร์ดแวร์ที่สลับซับซ้อนและแพงกว่ามาก ส่วนรูปแบบที่ใช้ในบทความนี้จะใช้รูปแบบจำนวนโดยตรง ซึ่งจะกล่าวถึงรายละเอียดต่อไปนี้

ตัวเลขฐานสองแบบจำนวนโดยตรง (Fixed point format)

โดยทั่วไปเลขฐานสองแบบจำนวนโดยตรงนี้แบ่งได้ 3 รูปแบบด้วยกันคือ

(ก.) แบบขนาดและเครื่องหมาย (sign magnitude)

(ข.) แบบส่วนเติมเต็มเป็นหนึ่ง (one's complement)

(ข.) แบบส่วนเติมเต็มเป็นสอง (two's complement)

ในทั้ง 3 รูปแบบนี้ได้เลือกใช้แบบส่วนเติมเต็มเป็นสองเนื่องจากว่า

1. เป็นตัวเลขที่ให้ค่าศูนย์เพียงค่าเดียว
2. การสร้างฮาร์ดแวร์สำหรับการบวก ลบ และการคูณ ทำได้ง่าย
3. ในระหว่างผลการบวกย่อยของการบวกเลขส่วนเติมเต็ม สอง สามหรือสี่จำนวน ถึงแม้จะเกิดโอเวอร์โฟล์ (ตัวทดจากผลบวกล้นข้ามไปทับบิทเครื่องหมาย) แต่ผลลัพธ์สุดท้ายให้ค่าถูกต้องเสมอ ถ้าผลบวกอยู่ในช่วง -1 ถึง $1-2^{-B}$

ตัวเลขแบบส่วนเติมเต็มเป็นสอง (Two's complement)

เนื่องจากการประมวลผลในระบบเชิงเลข จำนวนต่างๆมักมีค่าเป็นเลขทศนิยม (fraction) ที่มีค่าขนาดของมันน้อยกว่าหรือเท่ากับ 1 ($|x|<=1$) เสมอ ดังนั้นจำนวนเหล่านี้สามารถแทนเลขทศนิยมด้วยเลขฐานสองจำนวน B บิท นับจากบิทที่มีนัยสำคัญสูงสุด นั่นคือตัวเลขเหล่านั้นได้ถูกเสกลด้วยค่า 2^{-B} นั่นเอง

นิยามสำหรับตัวเลขส่วนเติมเต็มเป็นสองดังมีต่อไปนี้

$$\bar{x} = \begin{cases} x & \text{เมื่อ } x >= 0 \\ 2-|x| & \text{เมื่อ } |x| < 0 \end{cases}$$

โดยที่ $\bar{x}$ คือ ส่วนเติมเต็มเป็นสองของเลขจำนวน x

x คือ จำนวนเลขที่เป็นเศษส่วน (fraction number)

$\bar{x} = +- 0.b_{-1}b_{-2}b_{-3}.....b_{-m}$

การใช้เครื่องหมายของเลขส่วนเติมเต็มเป็นสอง

(Two's complement notation)

ในระบบเลขส่วนเติมเต็มเป็นสอง จะใช้บิทที่มีนัยสำคัญสูงสุด เป็นบิทแสดงเครื่องหมาย ถ้าเป็นบวกแทนด้วย "0" ส่วนลบแทนด้วย "1" ดังตัวอย่างดังต่อไปนี้

+0.828125 = 0110101

-0.828125 = 1110101

จะเห็นได้ว่าเลขฐานสองเมื่อแทนด้วยเลขฐานสองในลักษณะนี้จะมีค่าอยู่ในช่วง $-(1-2^{-B+1}) <= x <= (1-2^{-B+1})$ โดยที่มีค่าศูนย์อยู่เพียงค่าเดียวเท่านั้น

ถ้าให้ x แทนด้วยเลขไบนารีจำนวน B บิท ดังนั้นรูปแบบส่วนเติมเต็มเป็นสองซึ่งอาจเขียนในรูปแบบทางคณิตศาสตร์ได้ดังต่อไปนี้

$$\bar{x} = x_o.x_1x_2 .....x_B$$

. . ค่าของ x ในรูปแบบของเลขฐานสิบหาได้ดังสมการต่อไปนี้

$$x = -x_o + \sum_{i=1}^{B-1} x_i 2^{-i}$$

ตัวอย่าง เช่น $\bar{x}$ = 1101 สามารถแทนด้วยเลขฐานสิบได้ดังต่อไปนี้

$$x = -1+1*2^{-1} +0*2^{-2} +1*2^{-3}$$

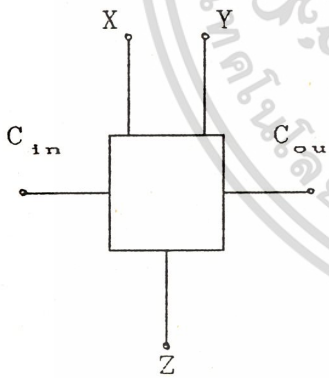
= -1+0.5+0.125

. . x = -0.375

การบวกและลบตัวเลขแบบส่วนเติมเต็มเป็นสอง

(Two's complement Addition and Subtraction)

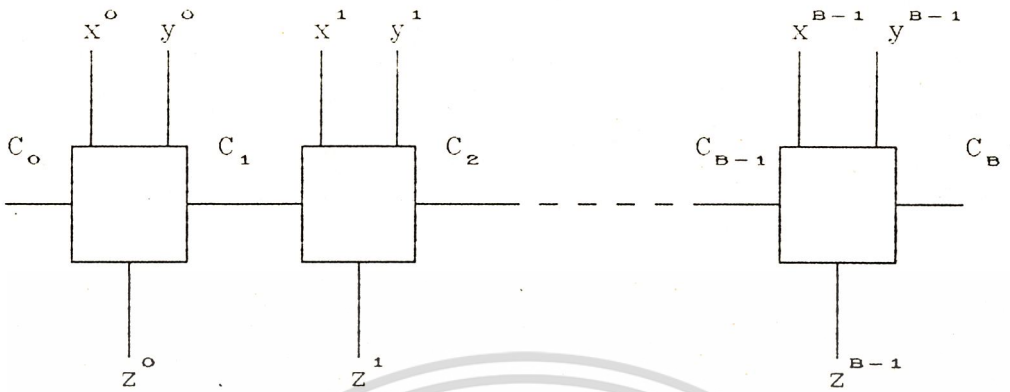
การบวกและลบตัวเลขแบบส่วนเติมเต็มเป็นสอง สามารถสร้างได้โดยใช้วงจรบวกเลขแบบคิดตัวทดเข้า (Full Adder) ขนาด 2 บิท โดยมีอินพุทของวงจรคือ ตัวตั้ง X,ตัวบวก/ลบ Y,ตัวทด C_{in} โดยที่ผลลัพธ์ที่ได้คือ Z และตัวทดออกC_{out}ดังแสดงในรูปที่1(a)พร้อมทั้งตารางแสดงค่าความจริง(Truth Table)



X	Y	C_{in}	Z	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

รูปที่1 ตารางค่าความจริงและวงจรบวกเลขแบบคิดตัวทดเข้า

สำหรับวงจรที่ใช้ในการบวกเลขแบบส่วนเติมเต็มเป็นสอง (รวมทั้งบิทเครื่องหมาย) B บิท สามารถแสดงได้ดังรูปที่ 2



รูปที่2 วงจรบวกเลขขนาด B บิท

โดยที่บิททดมายังบิทนัยสำคัญต่ำสุด (LSB) จะถูกกำหนดให้มีค่าเป็น 0 ($C_0=0$) สำหรับการลบตัวเลขแบบส่วนเติมเต็มเป็นสอง โดยการเปลี่ยนตัวลบให้เป็นตัวเลขแบบส่วนเติมเต็มเป็นหนึ่ง และบิททดมายังบิทนัยสำคัญต่ำสุดจะถูกกำหนดให้มีค่าเป็นหนึ่ง ($C_0=1$)

การคูณตัวเลขแบบส่วนเติมเต็มเป็นสอง
(Two's comolement multiplication)

พิจารณาการคูณตัวเลขแบบส่วนเติมเต็มเป็นสองขนาด B บิทสองจำนวน(x,y) โดยผลลัพธ์ที่ได้จากการคูณคือ z สามารถที่จะแสดงการหาผลลัพธ์ z ได้ดังนี้

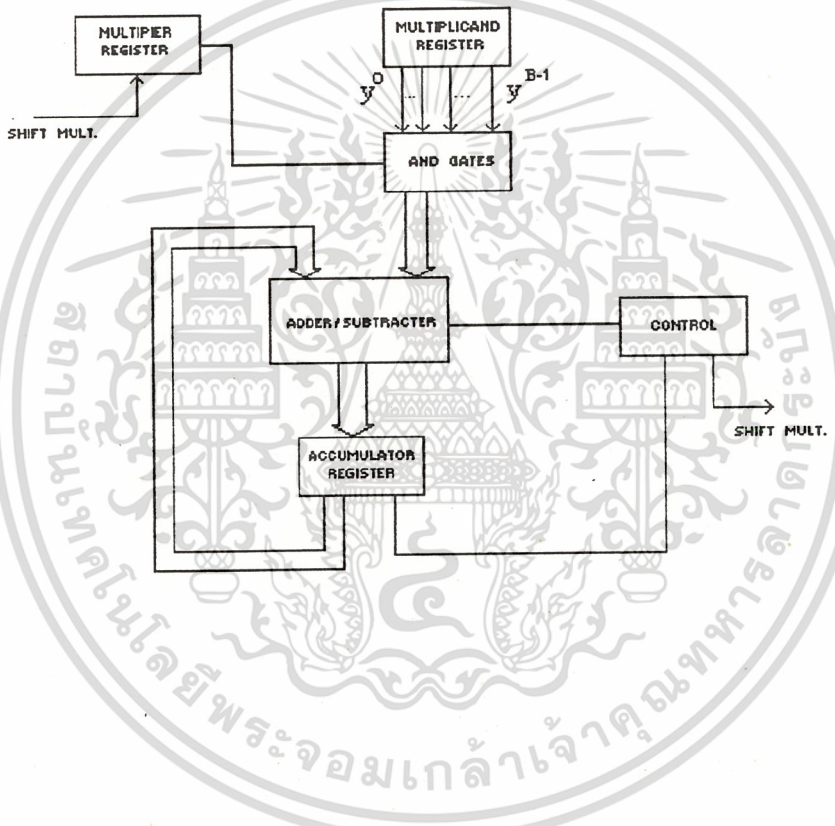
$$z = x_0 + \sum_{i=1}^{B-1} x_i 2^{-i} \quad)y = -x_0 y + \sum_{i=1}^{B-1} (x_i y) 2^{-i}$$

จากสมการข้างต้นจะเห็นได้ว่า การคูณ y เข้ากับแต่ละบิทของ x นั้น จะมีค่าเพียง 0 หรือ y เท่านั้นเนื่องจาก x_i มีค่า 0 หรือ 1 เท่านั้น การคูณจะเริ่มคูณ y

กับบิทนัยสำคัญต่ำสุด (x_{B-1}) ก่อน และภายหลังการคูณ y เข้ากับแต่ละบิทของ x จะนำผลลัพธ์ที่ได้มาบวกกันกับผลลัพธ์ก่อนหน้านั้นเสมอ สำหรับผลของการคูณ x_{B-1} กับ y นี้ค่าที่นำมาบวกเป็นค่าเริ่มแรกคือ "0"หลังจากที่บวกกับผลลัพธ์ก่อนหน้าแล้วจะเลื่อนผลลัพธ์ที่ได้นี้ไปทางขวาหนึ่งบิท หรือเท่ากับเป็นการคูณด้วย 2^{-1} เมื่อการคูณดำเนินไปถึงบิทสุดท้ายซึ่งเป็นบิทนัยสำคัญสูงสุด (x_0) แล้วการลบจะนำมาใช้แทนการบวกสำหรับในการคูณบิทนี้ จะเห็นได้ว่าการคูณตัวเลขแบบส่วนเติมเต็มเป็นสองขนาด B บิท สองจำนวนเข้าด้วยกันนี้มีการแอนด์กัน B^2 ครั้ง , มีการบวกกันและเลื่อนข้อมูลอย่างละ B-1 ครั้ง และผลลัพธ์ที่ได้มีขนาด 2B-1 บิท รูปที่ 3 แสดงถึงแผนภาพของการคูณตัวเลขแบบส่วนเติมเต็มเป็นสอง

<u>ตัวอย่าง</u> สำหรับการคูณเมื่อ x = -0.4375 -> 110010 และ
y = -0.40625 -> 110011

x_5y	0 * 110011	=	000000
SHIFT ACCUMULATOR		=	0000000
x_4y	1 * 110011	=	110011
ADD			1100110
SHIFT ACCUMULATOR		=	11100110
x_3y	0 * 110011	=	000000
ADD		=	11100110
SHIFT ACCUMULATOR		=	111100110
x_2y	0 * 110011	=	000000
ADD			111100110
SHIFT ACCUMULATOR		=	1111100110
x_1y	1 * 110011	=	110011
ADD			1100010110
SHIFT ACCUMULATOR		=	11100010110
x_0y	1 * 110011	=	110011
SUBTRACT			00010110110
	00010110110	->	0.177734375



รูปที่3 การคูณตัวเลขแบบส่วนเติมเต็มเป็นสอง

บทที่2.โครงสร้างแบบเลขคณิตแจกแจง

(Distributed Arithmetic Structure)

โครงสร้างแบบเลขคณิตแจกแจง หรือเรียกย่อๆว่า DA นี้ หนังสือบางเล่มเรียกชื่อว่า โครงแบบวิธีการของรอม (ROM Method) หรือ โครงสร้างแบบ ROM/ACC (ROM-ACCUMULATOR) หรือบางแห่งอาจเรียกว่า โครงสร้างการคูณเวกเตอร์ (Vector multiplication based structure) ซึ่งโครงสร้างนี้ใช้กันมากในการจัดรูปแบบทางคณิตศาสตร์ โดยเฉพาะจะปรากฏอยู่ในการประมวลผลสัญญาณดิจิตอล โดยจัดสมการให้อยู่ในรูปแบบการคำนวณแบบบิท เพื่อให้สมการเปลี่ยนเป็นวงจรดิจิตอลได้ โดยเฉพาะความเร็วในการคูณในบทความนี้จะออกแบบระบบการคูณแบบเปิดตารางค่าสัมประสิทธิ์ที่ถูกเก็บไว้ในหน่วยความจำ EPROM

การประยุกต์ใช้เลขคณิตแจกแจง (DA) นี้จะแปลงสมการผลบวกของผลคูณให้กระทำการคำนวณแบบบิท ซึ่งจะทำให้หาผลลัพธ์ได้เร็ว

อุปกรณ์สำคัญที่ใช้ในโครงสร้างแบบเลขคณิตแจกแจงนี้มี

1. ชีฟรีจีสเตอร์ (Shift register) เพื่อเลื่อนข้อมูลจากเวิร์ดเป็นบิท
2. หน่วยความจำประเภทรอม (ROM) สำหรับเก็บค่าสัมประสิทธิ์
3. ตัวรวมข้อมูล (ACCUMULATOR) เพื่อรวมข้อมูลจากรอม

จากที่กล่าวมาข้างต้น DA เป็นกรรมวิธีทางคณิตศาสตร์ที่จัดรูปแบบการคำนวณที่ส่วนใหญ่อยู่ในรูปผลบวกของผลคูณ (Sum of Products) ของเลขฐานสิบ ให้กระจายออกเป็นบิท หรือในรูปของเลขฐานสอง เพื่อให้การกระทำทางคณิตศาสตร์เปลี่ยนไปในรูปของวงจรดิจิตอลได้ วงจรดังกล่าวอาจเป็นวงจรซึ่งใช้แทนทรานสเฟอร์ฟังชันแบบต่างๆ ที่มีลักษณะสัญญาณเชิงอินพุทหรือเอาท์พุทคูณกับค่าคงที่

การใช้ DA ในการประมวลผลจะมีประสิทธิภาพมากโดยเฉพาะสมการแบบผลบวกของผลคูณของสัญญาณดิจิตอลที่เป็นอินพุทกับค่าสัมประสิทธิ์ของสมการ

จะกล่าวถึงรายละเอียดที่จะนำเอาโครงสร้างแบบเลขคณิตแจกแจงนี้ไปสร้างเป็นวงจรกรองเชิงเลขแบบป้อนกลับอันดับที่สอง ได้ดังนี้

เมื่อพิจารณาถึงสมการผลต่างสืบเนื่อง (Difference equation)อันดับที่สองให้อยู่ในรูปส่วนเติมเต็มเป็นสองได้ดังนี้

$$y(n) = a_0x(n)+a_1x(n-1)+a_2x(n-2)-b_1y(n-1)-b_2y(n-2) \quad ---(1)$$

ก่าหนดให้

$$\overline{x(n)} = x_o(n)x_1(n).x_2(n)...x_B(n)$$

และ

$$\overline{y(n)} = y_o(n).y_1(n).y_2(n)...y_B(n)$$

คือส่วนเติมเต็มเป็นสองของ x(n) และ y(n) ตามล่าดับ

$\overline{a_1}, \overline{b_1}$ คือส่วนเติมเต็มเป็นสองของ a,b ตามล่าดับ ดังนั้น x(n) และ y(n) สามารถเขียนในรูปของส่วนเติมเต็มเป็นสองได้ดังนี้

$$x(n) = -x_o(n) + \sum_{i=1}^{B-1} x_i(n)2^{-i}$$

และ

$$y(n) = -y_o(n) + \sum_{i=1}^{B-1} y_i(n)2^{-i} \qquad \text{----------(2)}$$

B คือจ่านวนบิท ; แทนค่า (2) ใน (1)

$$y(n) = a_o\left[-x_o(n)+\sum_{i=1}^{B-1} x_i(n)2^{-i}\right] + a_1\left[-x_o(n-1)+\sum_{i=1}^{B-1} x_i(n-1)2^{-i}\right]$$

$$+a_2\left[-x_o(n-2)+\sum_{i=1}^{B-1} x_i(n-2)2^{-i}\right] + b_1\left[-y_o(n-1)+\sum_{i=1}^{B-1} y_i(n-1)2^{-i}\right]$$

$$-b_2\left[-y_o(n-2)+\sum_{i=1}^{B-1} y_i(n-2)2^{-i}\right]$$

$$y(n) = \Big[a_o \sum_{i=1}^{B-1} x_i(n)2^{-i} + a_1 \sum_{i=1}^{B-1} x_i(n-1)2^{-i} + a_2 \sum_{i=1}^{B-1} x_i(n-2)2^{-i}$$

$$-b_1 \sum_{i=1}^{B-1} y_i(n-1)2^{-i} - b_2 \sum_{i=1}^{B-1} y_i(n-2)2^{-i} \Big]$$

$$-[\, a_o x_o(n)+a_1 x_o(n-1)+a_2 x_o(n-2)-b_1 y_o(n-1)-b_2 y_o(n-2)]$$

$$y(n) = \sum_{i=1}^{B-1} 2^{-i} [a_o x_i(n)+a_1 x_i(n-1)+a_2 x_i(n-2)-b_1 y_i(n-1)-b_2 y_i(n-2)]$$

$$-[a_o x_o(n)+a_1 x_o(n-1)+a_2 x_o(n-2)-b_1 y_o(n-1)-b_2 y_o(n-2)]$$

คูณ 2^{-1} ทั้งสองข้าง

$$2^{-1} y(n) =$$

$$\sum_{i=1}^{B-1} 2^{-i} \big[2^{-1} a_o x_i(n)+2^{-1} a_1 x_i(n-1)+2^{-1} a_2 x_i(n-2)-2^{-1} b_1 y_i(n-1)$$

$$-2^{-1} b_2 y_i(n-2)] - [2^{-1} a_o x_o(n)+2^{-1} a_1 x_o(n-1)+2^{-1} a_2 x_o(n-2)$$

$$-2^{-1} b_1 y_o(n-1)-2^{-1} b_2 y_o(n-2)] \qquad \text{--------(3)}$$

พิจารณาส่วนเติมเต็มเป็นสอง (Two's complement) ของเลขจำนวน

$2^{-1} u$ ซึ่ง $\bar{u} = u_o.u_1.u_2.u_3.....u_m$ สำหรับ $u > 0$, (หรือ $u_o=0$)

จะได้ว่า

$$\text{Two's complement } (2^{-1} u) = 2^{-1} \bar{u}$$

และสำหรับ $u<0$ (หรือ $u_{o=1}$) จะได้ว่า

$$\text{Two's complement } (2^{-1}u) = 2^{-1} - |2^{-1}u|$$

$$= 1 + 2^{-1}[2-|u|]$$

$$= 1 + 2^{-1}\bar{u}$$

$$\therefore \text{Two's complement } (2^{-1}u) = \begin{cases} 2^{-1}\bar{u} & \text{ถ้า } u_o=0 \\ 1 + 2^{-1}\bar{u} & \text{ถ้า } u_o=1 \end{cases} \quad \text{--------(4)}$$

จะเห็นได้ว่าการทำงานของส่วนเติมเต็มเป็นสอง (Two's complement) เป็นการ shift right ดังนั้นสมการที่ (3) โดยอาศัยสมการที่ (4) ได้ดังนี้

$$2^{-1}\bar{y}(n) =$$

$$\sum_{i=1}^{B-1} 2^{-i}\left[2^{-1}\bar{a}_o x_i(n)+2^{-1}\bar{a}_1 x_i(n-1)+2^{-1}\bar{a}_2 x_i(n-2)-2^{-1}\bar{b}_1 y_i(n-1)\right.$$

$$\left.-2^{-1}\bar{b}_2 y_i(n-2)\right] - \left[2^{-1}\bar{a}_o x_o(n)+2^{-1}\bar{a}_1 x_o(n-1)+2^{-1}\bar{a}_2 x_o(n-2)\right.$$

$$\left.-2^{-1}\bar{b}_1 y_o(n-1)-2^{-1}\bar{b}_2 y_o(n-2)\right]$$

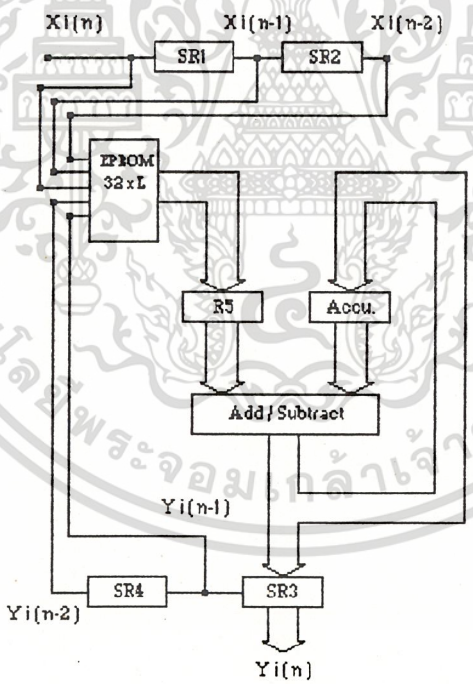
$$\therefore \bar{y}(n) = \sum_{i=0}^{B-1} 2^{-i} F_i - F_o \quad \text{---------(5)}$$

โดยให้

$$F_i = \bar{a}_o x_i(n)+\bar{a}_1 x_i(n-1)+\bar{a}_2 x_i(n-2)-\bar{b}_1 y_i(n-1)-\bar{b}_2 y_i(n-2) \quad \text{---(6)}$$

$$F_o = \bar{a}_o x_o(n) + \bar{a}_1 x_o(n-1) + \bar{a}_2 x_o(n-2) - \bar{b}_1 y_o(n-1) - \bar{b}_2 y_o(n-2) \quad ---(7)$$

จะเห็นได้ว่าจากสมการที่ (5) มีตัวแปรอยู่ 5 ตัวคือ $x_i(n), x_i(n-1), x_i(n-2), y_i(n-1), y_i(n-2)$ ซึ่งแต่ละตัวแปรจะมีค่าเพียง 1 และ 0 เท่านั้น ค่าที่เป็นไปได้ที่ต้องแมพอยู่ใน EPROM คือ 2 ยกกำลัง 5 = 32 ค่า แต่ B เป็นจำนวนบิท ดังนั้นเราจะต้องใช้หน่วยความจำหรือ EPROM มีขนาด 32 * B bits และจากสมการที่ (5) สามารถสร้างเป็นวงจรได้ ดังรูปที่ 1.



รูปที่1. แสดงภาคประมวลผลของตัวกรองเชิงเลขอันดับสอง

SR1,SR2,SR3,SR4 เป็นวงจรเลื่อนข้อมูลแบบอนุกรม (Serial Shift register) ขนาด12 bits, R5 เป็นรีจิสเตอร์ชนิด เข้า/ออกแบบขนาน (Parallel in - Parallel out register) ลำดับขั้นตอนการทำงานของรูปที่ 1. อธิบายได้ดังนี้

ก. ทำการ Clear Accumulator

คำนวณค่า F_i เมื่อ i=B โดยการ Load ค่า F_i ลงใน Register R5

ข. บวก F_i ซึ่งอยู่ใน R5 กับค่าที่อยู่ใน Accumulator (เป็นการบวกแบบ two's complement)

ค. เลื่อนค่าที่อยู่ใน Accumulator ไปทางขวา 1 บิท (two's complement shift)

ง. กระทำซ้ำในข้อ (ข) และ (ง) สำหรับ i = B-1,B-2...จนกระทั่งถึง 1

จ. คำนวณค่า F_o

ฉ. นำ F_o ไปลบออกจากค่าที่อยู่ใน Accumulator (ลบแบบ two's complement)

ช. ผลลัพธ์ที่ได้จาก (ฉ) จะเป็นเอาท์พุท y(n) และ y(n) ถูกป้อนกลับไปเป็น Address ของ EPROM (ค่าสัมประสิทธิ์ของตัวกรองที่อยู่ใน EPROM จะถูกนำมาคูรกับ x(n) ในรูปแบบของการบวกและลบเลขแบบ two's complement) จากนั้นวนไปกระทำซ้ำข้อ (ก) ถึง (ช)

การคำนวณค่าที่แมพในหน่วยความจำ EPROM

จากสมการ (5) คือ $\bar{y}(n) = \sum_{i=1}^{B-1} 2^{-1} F_i - F_o$

B คือจำนวนบิท สำหรับบทความนี้กำหนดให้ B = 12 bits. ดังนั้นต้องใช้หน่วยความจำขนาด 12*32 = 384 bits. การคำนวณหาค่า F_iได้จากสมการที่ (6) โดยมี Address EPROM ที่ไม่ซ้ำกันอยู่เท่ากับ 2 ยกกำลัง 5 = 32 Address โดยเริ่มจาก 00000 จนถึง 11111 จากคุณสมบัติของตัวกรองผ่านความถี่ต่ำ มีความถี่ Cut off หรือ f_c = 36 KHz. และมีฟังก์ชันถ่ายโอนดังนี้

$$H(z) = \frac{0.009604+0.010563591+0.01Z^{2}}{0.021611325-0.056774998Z+0.065131304Z^{2}}$$

$a_o = 0.01$ $a_1 = 0.010563591$ $a_2 = 0.009604$

$b_o = 0.065131304$ $b_1 = -0.056774998$ $b_2 = 0.021811325$

แปลง a_o , a_1 , a_2 , b_o , b_1 , b_2 ให้อยู่ในรูปของเลขส่วนเติมเต็มสอง (two's complement) ได้ดังนี้

$\bar{a}_o = 0.000000101000$ $\bar{a}_1 = 0.000000101011$

$\bar{a}_2 = 0.000000100111$ $\bar{b}_o = 0.000100001010$

$\bar{b}_1 = 0.000011101000$ $\bar{b}_2 = 1.111110100110$

นำค่า $\bar{a}_o, \bar{a}_1, \bar{a}_2, \bar{b}_o, \bar{b}_1, \bar{b}_2$ ไปแทนในสมการที่(6)

จากนั้นแทนค่าของ $x_1(n), x_1(n-1), x_1(n-2), y_1(n-1), y_1(n-2)$ ซึ่งมีค่าอยู่ระหว่าง 00000-11111 ลงในสมการ (6)

$$F_1 = \bar{a}_o x_1(n)+\bar{a}_1 x_1(n-1)+\bar{a}_2 x_1(n-2)-\bar{b}_1 y_1(n-1)-\bar{b}_2 y_1(n-2)$$

ก็จะได้ค่า F_1 ดังตารางที่ 1.

ตารางที่1.แสดงข้อมูลที่ตำแหน่งแอดเดรสต่างๆ

$x_1(n)$	$x_1(n-1)$	$x_1(n-2)$	$y_1(n-1)$	$y_1(n-2)$	F_1
0	0	0	0	0	0.000000000000
0	0	0	0	1	1.111110100110
0	0	0	1	0	0.000011101000
0	0	0	1	1	0.000010001110
0	0	1	0	0	0.000000100111
0	0	1	0	1	1.111111001101
0	0	1	1	0	0.000100001111
0	0	1	1	1	0.000010110101
0	1	0	0	0	0.000000101011
0	1	0	0	1	1.111111010001
0	1	0	1	0	0.000100010011
0	1	0	1	1	0.000010111001
0	1	1	0	0	0.000001010010
0	1	1	0	1	1.111111111000
0	1	1	1	0	0.000100111010
0	1	1	1	1	1.111111100000
1	0	0	0	0	0.000000101000
1	0	0	0	1	1.111111001110
1	0	0	1	0	0.000100010000
1	0	0	1	1	0.000010110110
1	0	1	0	0	0.000001001111
1	0	1	0	1	1.111111110101
1	0	1	1	0	0.000100110111
1	0	1	1	1	0.000011011101
1	1	0	0	0	0.000001010011
1	1	0	0	1	1.111111111001
1	1	0	1	0	0.000100111011
1	1	0	1	1	0.000011100001
1	1	1	0	0	0.000001111010
1	1	1	0	1	0.000110001010
1	1	1	1	0	0.000101100010
1	1	1	1	1	0.000100001000

บทที่3. การออกแบบตัวกรองป้อนกลับเชิงเลข

โดยทั่วไปในการออกแบบตัวกรองนั้น เราต้องการให้ได้ตัวกรองที่มีผลตอบสนองต่อความถี่(ถ้าเป็นไปได้) ให้มีทั้งผลตอบสนองแอมปลิจูด และ ผลตอบสนองเฟสตามผลตอบสนองอุดมคติที่ต้องการมากที่สุด อย่างไรก็ตามในการประยุกต์ใช้ในงานบางลักษณะ เราต้องการให้มีผลตอบสนองทางแอมปลิจูดตามต้องการเพียงอย่างเดียวก็พอ สำหรับการออกแบบตัวกรองให้มีผลตอบสนองแอมปลิจูดอย่างเดียวนั้น โดยทั่วไปตัวกรองป้อนกลับเชิงเลขใช้อันดับต่ำกว่าตัวกรองไม่ป้อนกลับเชิงเลขมาก ซึ่งข้อดีของตัวกรองอันดับต่ำก็คือ ประหยัดในเทอมของเวลาในการคำนวณ และขนาดของหน่วยความจำที่ต้องใช้ ส่วนข้อเสียที่สำคัญของการใช้ตัวกรองป้อนกลับเชิงเลขก็คือ มีผลตอบสนองเฟสไม่เป็นเชิงเส้น ซึ่งข้อเสียนี้จะทำให้การประมวลสัญญาณกับสัญญาณที่ต้องการคงรูปร่างของสัญญาณเดิมผิดเพี้ยนไป ซึ่งข้อเสียนี้อาจแก้ได้โดย เพิ่มส่วนของวงจรปรับเฟสเท่า (phase equalizer)

ในการออกแบบตัวกรองป้อนกลับเชิงเลข มีการออกแบบค่อนข้างยุ่งยาก ทั้งนี้เนื่องมาจากตัวกรองแบบนี้มีการนำสัญญาณออกมาใช้ในการคำนวณหาสัญญาณลำดับถัดไป ในการนี้จึงต้องคำนึงถึงเสถียรภาพของตัวกรองด้วย

โดยทั่วไปมีวิธีการออกแบบตัวกรองป้อนกลับเชิงเลข 2 วิธีใหญ่

วิธีแรก คือ วิธีออกแบบโดยทำการแปลงจากฟังก์ชันถ่ายโอนของตัวกรองเชิงอุปมานที่มีเสถียรภาพดี วิธีการนี้ถ้าหากมีการแปลงที่ดีการออกแบบก็ไม่ต้องคำนึงถึงเสถียรภาพของตัวกรองอีก โดยที่ได้ผลตอบสนองต่อแอมปลิจูดตามต้องการ

วิธีที่สอง คือ การออกแบบตัวกรองป้อนกลับเชิงเลขในโดเมน z โดยตรง ซึ่งวิธีนี้ยังรวมถึงการออกแบบโดยคอมพิวเตอร์ช่วย

สำหรับในบทความนี้จะใช้วิธีแรก คือ การออกแบบโดยทำการแปลงจากฟังก์ชันถ่ายโอนของตัวกรองที่มีเสถียรภาพดี ซึ่งมีรายละเอียดสำหรับการออกแบบตัวกรองเชิงเลขต่างๆดังนี้

การออกแบบตัวกรองป้อนกลับเชิงเลขชนิดผ่านความถี่ต่ำ
(Design of Lowpass IIR Filter)

รูปแบบของผลตอบสนองทางแอมปลิจูดของตัวกรองชนิดนี้ สามารถแสดงได้ดังรูปที่(1) แถบผ่านจะเริ่มจาก $\lambda = 0$ จนกระทั่งถึงตำแหน่งความถี่ตัด (cutoff

frequency) ที่ $\lambda = \Lambda_c$, บริเวณแถบหยุด (stop band) จะเริ่มจาก $\lambda = \Lambda_s$ จนถึง $\lambda = \pi$ ส่วนบริเวณระหว่างช่วงความถี่ $\Lambda_c \leq \lambda \leq \Lambda_s$ เป็นบริเวณแถบเปลี่ยนสถานะ (transition band) ส่วน δ_1 และ δ_2 เป็นค่าที่กำหนดขอบเขตของแถบผ่านและแถบหยุดตามลำดับ ในการออกแบบตัวกรองชนิดนี้จะเริ่มพิจารณาที่ฟังก์ชันถ่ายโอนของตัวกรองซึ่งมีขนาดแอมปลิจูดอยู่บนวงกลมหนึ่งหน่วย (unit circle)

$$1 - \delta_1 \leq |H(e^{j\lambda})| \leq 1 + \delta_1 \qquad \text{เมื่อ} \quad 0 \leq \lambda \leq \Lambda_c$$

และ

$$|H(e^{j\lambda})| \leq \delta_2 \qquad \text{เมื่อ} \quad \Lambda_s \leq \lambda \leq \pi$$

หลังจากนั้นจะหาความสัมพันธ์ระหว่าง ความถี่ของตัวกรองเชิงอุปมาน และความถี่ของตัวกรองเชิงเลขแบบป้อนกลับ ถ้ากำหนดให้ความถี่ที่ใช้ในการสุ่มตัวอย่างสัญญาณมีค่าเป็น $F_s = 1/T$ โดยที่ T เป็นช่วงเวลาที่ใช้ในการสุ่มตัวอย่างสัญญาณแต่ละตัว จะสามารถหาความสัมพันธ์ระหว่างความถี่ในระบบเชิงอุปมานและในระบบเชิงเลขได้จากสมการที่ (1)

$$\lambda = 2\pi f / F_s \qquad \text{----------(1)}$$

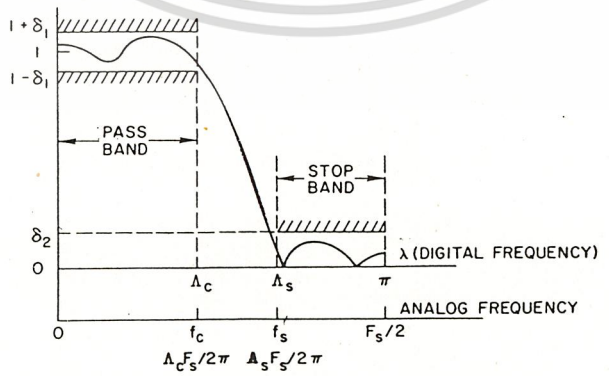
ดังนั้น

$$f_c = \Lambda_c F_s / 2\pi$$

และ

$$f_s = \Lambda_s F_s / 2\pi$$

เมื่อกำหนดให้ f_c และ f_s เป็นความถี่ตัด และ ความถี่เริ่มต้นของแถบหยุดในระบบเชิงอุปมานตามลำดับ ซึ่งสามารถแสดงความสัมพันธ์ดังกล่าวได้ดังรูปที่(1)



รูปที่1. ผลตอบสนองทางแอมปลิจูดตัวกรองชนิดผ่านความถี่ต่ำ

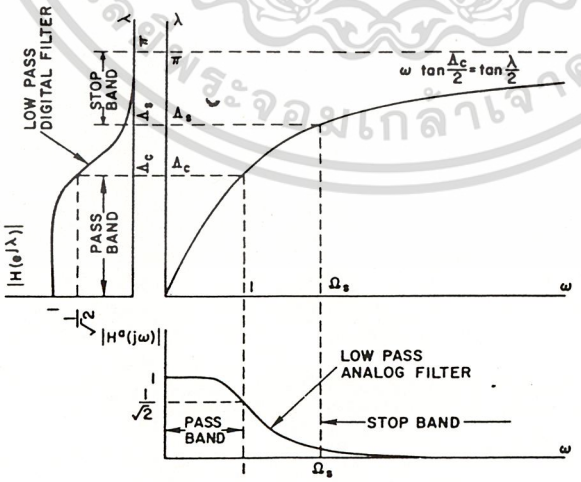
ขั้นตอนสำคัญประการหนึ่งสำหรับการออกแบบตัวกรองป้อนกลับเชิงเลข คือ การเลือกตัวกรองต้นแบบ (Prototype) ในระบบเชิงอุปมานให้เหมาะสมกับการใช้งานในกรณีต่างๆกัน ซึ่งตัวกรองเชิงอุปมานต้นแบบที่กล่าวถึงในบทความนี้จะประกอบด้วย ตัวกรองบัตเตอร์เวิร์ด (Butterworth) และ ตัวกรองเชฟบีเชฟ (Chebyshev)

ภายหลังที่เลือกตัวกรองเชิงอุปมาได้แล้ว จากนั้นจะทำการแปลงจากฟังก์ชันถ่ายโอนของตัวกรองเชิงอุปมานไปเป็นฟังก์ชันถ่ายโอนของตัวกรองป้อนกลับเชิงเลขซึ่งมีรายละเอียดดังนี้

กำหนดให้การแปลงจากความถี่ w ซึ่งเป็นความถี่ในระบบเชิงอุปมานไปเป็น ซึ่งเป็นความถี่ในระบบเชิงเลขได้ดังนี้

$$\omega \tan(\Lambda_c/2) = \tan(\lambda/2) \qquad \text{----------(2)}$$

รูปที่(2) เป็นกราฟแสดงความสัมพันธ์ระหว่าง w และ λ ตามสมการที่(2)ซึ่งจะเห็นได้ว่าในช่วง $0 <= w <= \infty$ จะถูกส่ง (map) ไปยังช่วง $0 <= \lambda <= \pi$ ที่ความถี่ $w = 1$ จะสัมพันธ์กับ $\lambda = \Lambda_c$. นอกจากนี้รูปที่(2) ยังแสดงถึงความสัมพันธ์ของฟังก์ชันถ่ายโอนในระบบเชิงอุปมาน $|H(jw)|$ กับฟังก์ชันถ่ายโอนของตัวกรองป้อนกลับเชิงเลขชนิดผ่านความถี่ต่ำ



รูปที่2 แสดงการเปลี่ยนโดเมนความถี่ตัวกรองชนิดผ่านความถี่ต่ำ

ถ้ากำหนดให้ $S = jw$ และ $Z = e^{j\lambda}$ สมการที่(2)สามารถเขียนได้ดังนี้

$$s = j\cot(\Lambda_c/2)\frac{\sin(\lambda/2)}{\cos(\lambda/2)}$$

$$= j\cot(\Lambda_c/2)\frac{(e^{j\lambda/2} - e^{-j\lambda/2})/2j}{(e^{j\lambda/2} + e^{-j\lambda/2})/2}$$

$$= c\frac{z-1}{z+1} \qquad \text{----------(3)}$$

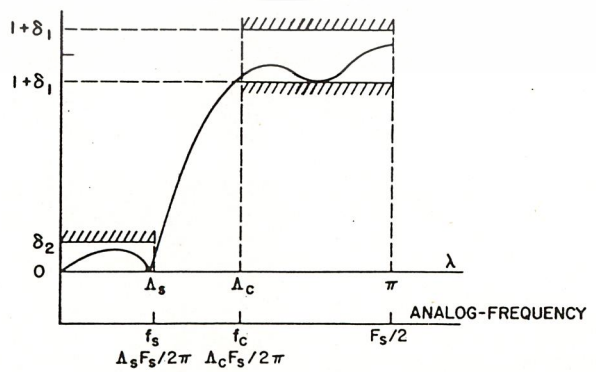
โดยที่ $C = \cot(\Lambda_c/2)$ สมการที่ (3) นี้มีชื่อเรียกว่าการแปลงเชิงเส้นคู่ (bilinear transform) สำหรับตัวกรองป้อนกลับเชิงเลขชนิดความถี่ต่ำ

$$H(z) = H^a[c(z-1)/(z+1)]$$

การออกแบบตัวกรองป้อนกลับเชิงเลขชนิดผ่านความถี่สูง

(Design of Highpass Recursive Filters)

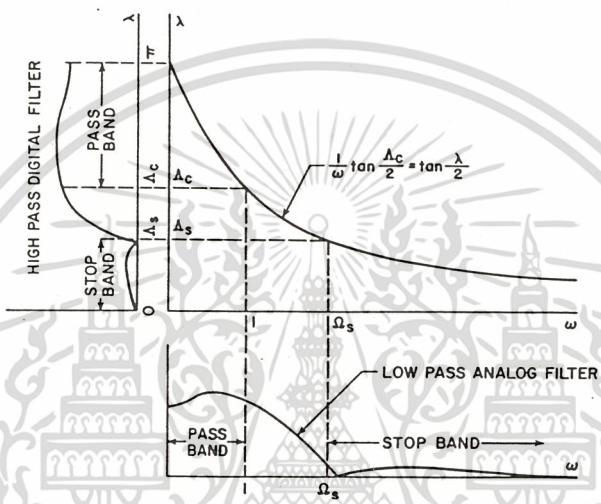
รูปแบบของผลตอบสนองทางแอมปลิจูดของตัวกรองชนิดผ่านความถี่สูงสามารถแสดงได้ดังรูปที่ (3) การออกแบบตัวกรองป้อนกลับเชิงเลขชนิดผ่านความถี่สูงก็จะมีลักษณะคล้ายกับการออกแบบตัวกรองป้อนกลับเชิงเลขชนิดผ่านความถี่ต่ำ กล่าวคือ จะต้องทำการออกแบบตัวกรองชนิดผ่านความถี่ต่ำในระบบเชิงอุปมานเสียก่อน จากนั้นจึงทำการแปลงเป็นตัวกรองในระบบเชิงเลขชนิดผ่านความถี่สูงได้ตามต้องการ สำหรับสมการที่ใช้หาความสัมพันธ์ระหว่างระบบเชิงเลข และ ระบบเชิงอุปมานจะต่างกับการแปลงในหัวข้อที่ผ่านมาเล็กน้อย ดังแสดงได้ดังสมการที่ (4)



รูปที่3 แสดงผลตอบสนองทางแอมปลิจูดตัวกรองชนิดผ่านความถี่สูง

$$\omega \tan(\lambda/2) = \tan(\Lambda_c/2) \qquad \text{----------(4)}$$

ในลักษณะคล้ายหัวข้อที่ผ่านมาสามารถพิจารณาจากรูปที่(4) ซึ่งแสดงความสัมพันธ์ตามการแปลงในสมการที่ (4) และ ความสัมพันธ์ระหว่างฟังก์ชันถ่ายโอนของตัวกรองชนิดผ่านความถี่ต่ำและตัวกรองป้อนกลับเชิงเลขชนิดผ่านความถี่สูง ในช่วง $0 <= w < \alpha$ จะถูกส่งไปยังบริเวณ $0 < \lambda <= \pi$ แต่ในบริเวณ $0 < w < 1$ จะสัมพันธ์กับ $\Lambda_c < \lambda < \pi$



รูปที่4 แสดงการเปลี่ยนโดเมนความถี่ตัวกรองชนิดผ่านความถี่สูง

ถ้ากำหนดให้ $S = jw$ และ $Z = e^{j\lambda}$ สมการที่ (4) สามารถเขียนได้ดังนี้

$$s = j\tan(\Lambda_c/2)\cot(\lambda/2)$$

$$= j\tan(\Lambda_c/2)\frac{\cos(\lambda/2)}{\sin(\lambda/2)}$$

$$= j\tan(\Lambda_c/2)\frac{(e^{j\lambda/2} + e^{-j\lambda/2})/2}{(e^{j\lambda/2} - e^{-j\lambda/2})/2j}$$

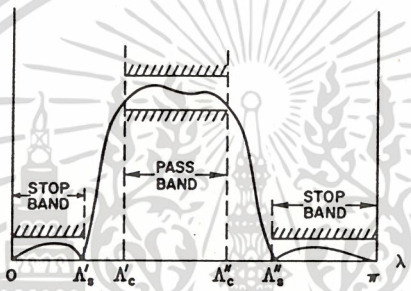
$$= c'\frac{z+1}{z-1} \qquad \text{----------(5)}$$

โดยที่ $C' = \tan(\Lambda_c/2)$ สมการที่ (5) คือการแปลงเชิงเส้นคู่ที่ใช้แปลงจากตัวกรองต้นแบบเชิงอุปมาชนิดผ่านความถี่ต่ำไปเป็นตัวกรองป้อนกลับเชิงเลขชนิดผ่านความถี่สูง

$$H(z) = H^a[c'(1+z)/(1-z)]$$

การออกแบบตัวกรองป้อนกลับเชิงเลขชนิดผ่านแถบความถี่และชนิดกำจัดแถบความถี่ (Design of Bandpass and Bandstop Fikters)

รูปแบบของผลตอบสนองทางแอมปลิจูดของตัวกรองชนิดผ่านแถบความถี่สามารถแสดงได้ดังรูปที่(5) ซึ่งมีแถบผ่านอยู่ในช่วง Λ_c' จนถึง Λ_c'' แถบหยุดอยู่ในช่วงแรกคือ $0 <= \lambda <= \Lambda_s'$ และในช่วงที่สองคือ $\Lambda_s'' <= \lambda <= \pi$ สำหรับความสัมพันธ์ระหว่างความถี่ในระบบตัวกรองเชิงอุปมานชนิดผ่านความถี่ต่ำ และ ความถี่ในระบบเชิงเลขสำหรับการสร้างตัวกรองชนิดนี้มีความสัมพันธ์ดังสมการที่ (6)



รูปที่5 แสดงผลตอบสนองทางแอมปลิจูดตัวกรองชนิดผ่านแถบความถี่

$$\omega = \alpha(\beta - \cos\lambda)/\sin\lambda \qquad \text{----------(6)}$$

เมื่อกำหนดให้

$$\alpha = \cot[(\Lambda_c'' - \Lambda_c')/2]$$

$$\beta = \frac{\sin(\Lambda_c' + \Lambda_c'')}{\sin\Lambda_c' + \sin\Lambda_c''}$$

รูปที่(6) แสดงความสัมพันธ์ของการแปลงตามสมการที่ (6) ซึ่งแสดงความสัมพันธ์ระหว่าง $-\alpha < w < \alpha$ กับค่าบวกของ λ จะเห็นได้ว่าในช่วง $-1 <= w <= 1$ ซึ่งเป็นแถบผ่านของตัวกรองเชิงอุปมานชนิดผ่านความถี่ต่ำ จะถูกส่งไปยังแถบผ่านของตัวกรองป้อนกลับเชิงเลข $\Lambda_c' <= \lambda <= \Lambda_c''$ ส่วนที่ขอบของแถบหยุดจะถูกส่งจาก $\alpha(\beta-\cos\Lambda_s')/\sin\Lambda_s'$ และ $\alpha(\beta-\cos\Lambda_s'')/\sin\Lambda_s''$ ตามลำดับ ดังนั้นสามารถสรุปเงื่นไขที่สำคัญในการออกแบบตัวกรองชนิดนี้คือ

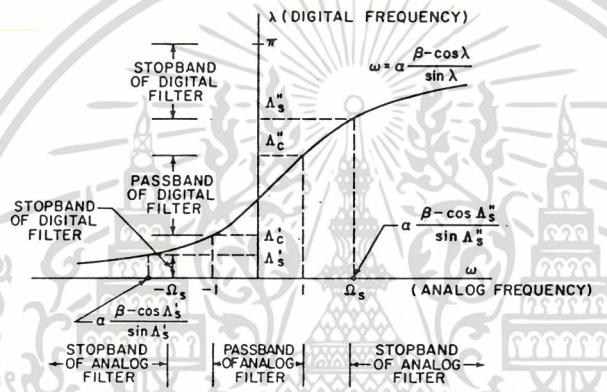
$$\alpha(\beta - \cos\Lambda_s')/\sin\Lambda_s' \leq -\Omega_s$$

$$\alpha(\beta - \cos\Lambda_s'')/\sin\Lambda_s'' \geq \Omega_s$$

จากความสัมพันธ์ข้างต้นสามารถแปลงจากโดเมน S ไปยังโดเมน Z โดยใช้การแปลงดังต่อไปนี้

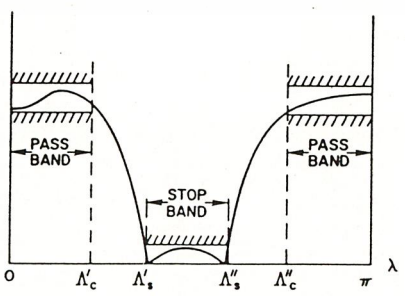
$$s = \alpha \frac{z^2 - 2\beta z + 1}{z^2 - 1} \qquad \text{----------(7)}$$

จากสมการที่ (7) จะเห็นได้ว่าภายหลังจากการแปลง อันดับของตัวกรองจะสูงขึ้นจากอันดับของตัวกรองเชิงอุปมาน 2 เท่า กล่าวคือ ถ้าอันดับของตัวกรองเชิงอุปมานคือ N ภายหลังจากการแปลงโดยใช้สมการที่ (7) แล้วจะได้ตัวกรองเชิงเลขที่มีอันดับของตัวกรองเท่ากับ 2N



รูปที่6 แสดงการเปลี่ยนโดเมนความถี่ตัวกรองชนิดผ่านแถบความถี่

สำหรับการออกแบบตัวกรองป้อนกลับเชิงเลขชนิดกำจัดแถบความถี่ซึ่งมีรูปแบบของผลตอบสนองทางแอมปลิจูดตามรูปที่(7) มีความสัมพันธ์ระหว่างความถี่ของตัวกรองความถี่ต่ำเชิงอุปมานกับตัวกรองป้อนกลับเชิงเลขชนิดนี้ตามสมการที่ (8)



รูปที่7 แสดงผลตอบสนองทางแอมปลิจูดตัวกรองชนิดกำจัดแถบความถี่

$$\omega = \frac{\alpha \sin\lambda}{\beta - \cos\lambda} \qquad \text{----------(8)}$$

เมื่อกำหนดให้

$$\alpha = \frac{\cos\Lambda_c' - \cos\Lambda_c''}{\sin\Lambda_c' + \sin\Lambda_c''}$$

$$\beta = \frac{\sin(\Lambda_c' + \Lambda_c'')}{\sin\Lambda_c' + \sin\Lambda_c''}$$

เพื่อที่จะให้การส่งเป็นไปอย่างถูกต้องเงื่อนไขที่สำคัญคือ

$$\frac{\alpha \sin\Lambda_s'}{\beta - \cos\Lambda_s'} \leq -\Omega_s$$

$$\frac{\alpha \sin\Lambda_s''}{\beta - \cos\Lambda_s''} \geq \Omega_s$$

ดังนั้นการแปลงจากโดเมน S ไปยังโดเมน Z จะเป็นไปตามสมการที่ (9)

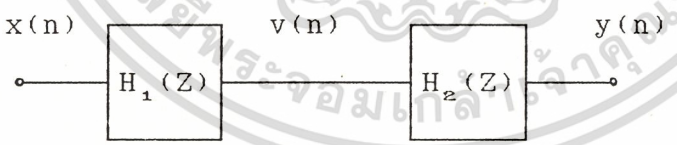
$$s = \frac{\alpha(z^2 - 1)}{z^2 - 2\beta z + 1} \qquad \text{----------(9)}$$

บทที่4. การสร้างตัวกรองเชิงเลขอันดับที่สี่

ในทางปฏิบัตินั้น การนำตัวกรองไปสร้างใช้งาน มักไม่นิยมสร้างเป็นตัวกรองอันดับสูงโดยตรง เนื่องจากเกิดผลของปรากฏการณ์จากความยาวจำกัดซึ่งยิ่งตัวกรองอันดับสูงยิ่งมีผลมาก ผลเช่นนี้มี เช่น ผลตอบสนองความถี่ผิดเพี้ยนไป หรือมีการกวัดแกว่งของสัญญาณออก เป็นต้น ส่วนในแง่การออกแบบ เราต้องการตัวกรองที่เสถียร โดยตรวจสอบได้จากการแยกตัวประกอบของฟังก์ชันถ่ายโอนเพื่อดูตำแหน่งโพลของตัวกรอง ซึ่งการแยกตัวประกอบของฟังก์ชันอันดับสูงทำได้ค่อนข้างยุ่งยาก ด้วยเหตุผลเหล่านี้ โครงสร้างของตัวกรองที่ใช้ในการออกแบบจึงเลือกเป็นแบบ ตัวกรองอันดับสองต่อเรียงกัน (cascade) จำนวน 2 ภาค ซึ่งสามารถเขียนเป็นสมการในรูปตัวแปรแซดได้เป็น

$$H(Z) = A \prod_{k=1}^{2} \frac{1 + a_k Z^{-1} + b_k Z^{-2}}{1 + c_k Z^{-1} + d_k Z^{-2}}$$

โดยที่ a_k, b_k, c_k และ d_k เป็นค่าสัมประสิทธิ์ของตัวกรองภาคที่ K และ A เป็นค่าขยายของตัวกรอง จากความสัมพันธ์ดังกล่าวสามารถนำมาเขียนเป็นแผนภาพได้ดังรูปที่(1)



รูปที่1. การจัดรูปแบบโครงสร้างแบบอนุกรม

ตัวกรองป้อนกลับเชิงเลขอันดับที่ 4 สามารถสร้างจากการนำเอาตัวกรองป้อนกลับเชิงเลขมาต่ออนุกรมกันสองตัว ในแต่ละส่วนสามารถเขียนสมการเพื่อแทนความสัมพันธ์ของตัวกรองทั้งสองได้ดังนี้

$$v_n = a_{0,1}x_n + a_{1,1}x_{n-1} + a_{2,1}x_{n-2} - b_{1,1}v_{n-1} - b_{2,1}v_{n-2} \quad ----(1)$$

และ

$$y_n = a_{0,2}v_n + a_{1,2}v_{n-1} + a_{2,2}v_{n-2} - b_{0,2}y_{n-1} - b_{2,2}y_{n-2} \quad ----(2)$$

โดยที่ $\{x_n\}$ คือ ลำดับสัญญาณอินพุท, $\{v_n\}$ คือลำดับสัญญาณเอาท์พุทของตัวกรองส่วนแรก, $\{y_n\}$ เป็นลำดับสัญญาณเอาท์พุทของตัวกรองป้อนกลับเชิงเลขอันดับที่ 4 และ $\{a_{i,j}\}$, $\{b_{i,j}\}$ เป็นสัมประสิทธิ์ของตัวกรอง (j คือหมายเลขที่แสดงถึงตัวกรองแต่ละส่วน)

จากสมการที่ (1) และ (2) สามารถนำมาเขียนใหม่ให้อยู่ในรูปแบบของระบบเลขส่วนเติมเต็มเป็นสอง

$$v_n = \sum_{i=1}^{B-1} 2^{-i} F(x_n^i, x_{n-1}^i, x_{n-2}^i, v_{n-1}^i, v_{n-2}^i) - F(x_n^o, x_{n-1}^o, x_{n-2}^o, v_{n-1}^o, v_{n-2}^o) \quad -----(3)$$

และ

$$y_n = \sum_{i=1}^{B-1} 2^{-i} F(v_n^i, v_{n-1}^i, v_{n-2}^i, y_{n-1}, y_{n-2}^i) - F(v_n^o, v_{n-1}^o, v_{n-2}^o, y_{n-1}^o, y_{n-2}^o) \quad ------(4)$$

บล็อกไดอะแกรมของวงจรป้อนกลับเชิงเลขอันดับ 4 สามารถแสดงได้ดังรูปที่(2) จะเห็นว่าส่วนต่างๆของวงจรกรองจะมีส่วนควบคุมที่สอดคล้องกับอุปกรณ์ภายในตัวกรองต่างๆกัน ตามหน้าที่ของแต่ละตัว จากรูปที่(2) สามารถอธิบายถึงการทำงานได้ดังต่อไปนี้

อุปกรณ์เลื่อนข้อมูล (shift register) ที่มีใช้สำหรับเก็บพักข้อมูลจะต้องได้รับสัญญาณควบคุมที่เป็นสัญญาณนาฬิกาเพื่อใช้ในการเลื่อนข้อมูลในแต่ละบิท โดยที่จะใช้สัญญาณนาฬิกา 1 ตัว เพื่อใช้ในการเลื่อนข้อมูล 1 บิท ดังนั้นสำหรับข้อมูลขนาด 8 บิท จะต้องใช้สัญญาณนาฬิกา 8 ตัว สัญญาณควบคุมอีกแบบหนึ่ง คือ สัญญาณที่ใช้ควบคุมการโหลดข้อมูลเข้าแบบขนานออกแบบอนุกรม (parallel to serial convertion) สำหรับสัญญาณที่สำคัญอีกสัญญาณ คือ สัญญาณที่ใช้ควบคุมการเลือกหน่วยความจำประเภทรอม เพื่อใช้ในการคำนวณฟังก์ชัน F(...) ให้สอดคล้องกับขั้นตอนการประมวลว่าเป็นขั้นตอนในการประมวลผล v_n หรือ y_n ตัวประมวลผลทางคณิตศาสตร์ก็เช่นกัน ต้องมีสัญญาณที่คอยควบคุมการโหลดข้อมูลเข้ามายังตัวประมวลผลนี้

และ ยังรวมถึงการเคลียร์ (clear)สัญญาณควบคุมอีกสัญญาณหนึ่ง คือ สัญญาณที่ใช้ในการเลือกที่ตัวประมวลผลทางคณิตศาสตร์ว่าต้องการประมวลผลในลักษณะการบวกหรือการลบ

ในระหว่างการคำนวณหาลำดับสัญญาณ (V_n) จะเรียกกระบวนการนี้ว่า " วัฏจักร ROM1 " (ROM1 cycle) ซึ่งมีลำดับสัญญาณที่เกี่ยวข้องในการคำนวณ คือ $X_n, X_{n-1}, X_{n-2}, V_{n-1}$ และ V_{n-2}

ในขั้นตอนนี้อุปกรณ์เก็บพักสัญญาณ RX_0 และ RY_1 จะได้รับสัญญาณควบคุม S/L เป็นหนึ่ง เพื่อโหลดสัญญาณอินพุทเข้ามาเก็บพักไว้ ในทางกลับกัน ถ้าหาก RX_0 และ RX_1 ได้รับสัญญาณควบคุม S/L ที่มีค่าเป็นศูนย์ ข้อมูลที่เก็บพักไว้ดังกล่าว จะถูกเลื่อนออกมาอย่างอนุกรมอย่างสอดคล้องกับสัญญาณควบคุม

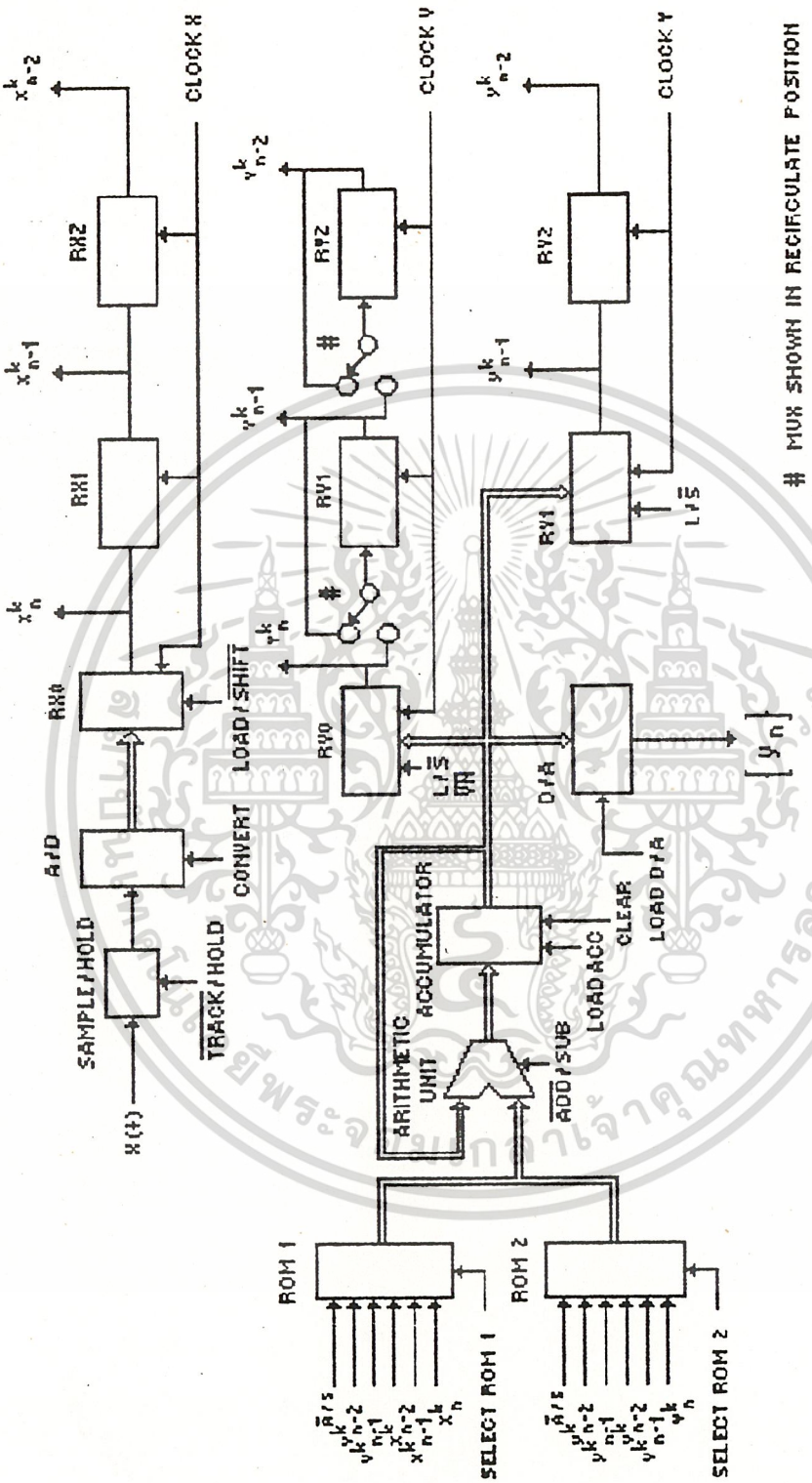
เนื่องจากเราใช้ระบบเลขแบบส่วนเติมเต็มเป็นสองขนาด 8 บิทแทนลำดับสัญญาณแต่ละตัว ดังนั้นสัญญาณนาฬิกาที่ใช้ควบคุมชิพรีจีสเตอร์จะต้องมี 8 ลูกเช่นกัน สำหรับสัญญาณนาฬิกา CKX,CKV จะใช้เป็นตัวควบคุมการเลื่อนข้อมูลแบบอนุกรมผ่าน RX0,RX1,RX2,RV1และRV2 โดยที่ชิพรีจีสเตอร์ RX0,RX1และRX2 จะถูกต่อด้วยกันอย่างอนุกรมและเนื่องจาก V_{n-1} และV_{n-2} เป็นลำดับสัญญาณที่ใช้ทั้งวัฏจักร ROM1และวัฏจักร ROM2 ดังนั้นลำดับสัญญาณทั้งสองจึงต้องมีการเลื่อนวนอยู่ภายในรีจีสเตอร์ RV1และ RV2 ในระหว่างการประมวลผลในวัฏจักร ROM1

ในระหว่างวัฏจักร ROM2 เพื่อที่จะคำนวณลำดับสัญญาณ V_n ชิพรีจีสเตอร์ RV1 และ RV2จะถูกเชื่อมต่อเข้าด้วยกัน เพื่อลำดับสัญญาณ V_n สามารถที่จะถูกเลื่อนผ่านชิพรีจีสเตอร์ RV1 และลำดับสัญญาณ V_{n-1} จะถูกเลื่อนผ่านชิพรีจีสเตอร์ RV2

ผลลัพธ์จากวัฏจักร ROM1 คือการคำนวณลำดับสัญญาณ V_n หลังจากนั้นจะถูกโหลดไปเก็บไว้ที่ตัวเก็บพักข้อมูล RVO เมื่อระดับสัญญาณควบคุม VN มีค่าเป็น 0 ต่อจากนั้นลำดับสัญญาณ V_n จะถูกเลื่อนผ่านชิพรีจีสเตอร์ RV1และRV2เพื่อที่จะนำลำดับสัญญาณ V_n, V_{n-1}, V_{n-2}, รวมทั้งY_{n-1} และ Y_{n-2} มาใช้ในการประมวลผลในวัฏจักร ROM2เช่นเดียวกับการเลื่อนข้อมูลผ่านชิพรีจีสเตอร์ RX1และ RX2ในวัฏจักร ROM1 สัญญาณนาฬิกา 8 ตัว ก็จะถูกนำมาควบคุมการเลื่อนลำดับสัญญาณผ่าน RVO,RV1,RV2,RY1และRY2 เมื่อสิ้นสุดการประมวลผลในวัฏจักร ROM2 ค่า Y_n ค่าใหม่ที่ประมวลผลได้จะถูกโหลดเก็บไว้ใน RY1 เพื่อเตรียมพร้อมไว้สำหรับการคำนวณในวัฏจักร ROM2 ในครั้งต่อไป

จากสมการที่ (3) และ (4) สามารถนำมาพิจารณาโครงสร้างเพื่อสร้างตัวกรองป้อนกลับเชิงเลขอันดับที่ 4 ได้ดังต่อไปนี้

ในส่วนแรกที่ต้องนำมาพิจารณา คือ จำนวนบิทที่ใช้ในแต่ละเวอร์ด และ ระบบ



รูปที่2. แผนภาพแสดงวงจรกรองเชิงเลขอันดับสอง

ตัวเลขที่ใช้ จากที่กล่าวมาแล้วในบทที่เกี่ยวกับระบบตัวเลข ในการสร้างตัวกรองจะเลือกใช้ระบบตัวเลขแบบรูปแบบจำนวนโดยตรง โดยมีขนาดความยาวของคำ 8 บิท โดยที่มีบิทนัยสำคัญสูงสุด (MSB) เป็นบิทแสดงเครื่องหมาย ส่วนจำนวนเลขที่แทนเลขทศนิยม คือ บิททางขวาของเลขบิทนัยสำคัญสูงสุด

ส่วนที่ใช้เก็บข้อมูล (Storage)

จากสมการที่ (3) และ (4)จะเห็นได้ว่าจะต้องมีการจัดลำดับสัญญาณให้ถูกต้องก่อนที่จะนำไปประมวลผล นั่นคือต้องสร้างลำดับข้อมูล X_n, X_{n-1}, X_{n-2}, V_n,V_{n-1},V_{n-2},Y_{n-1},Y_{n-2} เพื่อส่งลำดับสัญญาณเหล่านี้ไปชี้แอดเดรสของหน่วยความจำ ซึ่งลำดับสัญญาณเหล่านี้สามารถนำไปเก็บไว้ในรีจีสเตอร์ขนาด8บิทที่สามารถทำงานได้ทีละบิทในแต่ละครั้งที่ได้รับสัญญาณนาฬิกา ซึ่งรีจีสเตอร์เหล่านี้ก็คือชิพรีจีสเตอร์นั่นเอง(SHIFT REGISTER) จากวงจรในรูปที่(3) จะเห็นได้ว่าใช้วงจรรวม(INTRIGRATED CIRCUIT)ชนิด TTLเบอร์ 7491

นอกจากนี้จากรูปที่ (3) ยังประกอบไปด้วยรีจีสเตอร์ลักษณะอื่นๆอีกตามลักษณะการทำงาน เช่น RX0 ซึ่งเป็น IC TTL เบอร์ 74165 ซึ่งทำงานแบบเข้าขนานแต่ออกอนุกรม(PARALLEL IN SERIAL OUT)ซึ่งทำหน้าที่เป็นตัวเก็บพักข้อมูล

การออกแบบข้อมูลลงหน่วยความจำ (MEMORY MAP)

การออกแบบข้อมูลที่จะเก็บลงหน่วยความจำเป็นสิ่งที่ยุ่งยากสำหรับการออกแบบวงจรกรองเชิงเลขโดยใช้โครงสร้างแบบเลขคณิตแจกแจง เนื่องจากจะต้องเอาเวลาที่ใช้ในการอ่านข้อมูลแต่ละค่าจากหน่วยความจำ(ACCESS TIME)มาคำนึงถึงเพื่อให้จังหวะการทำงานของวงจรเป็นไปด้วยความถูกต้อง เมื่อเข้าใจการทำงานของหลักการกระจายทางคณิตศาสตร์ก็จะทำให้ทราบว่าจะต้องใช้ขนาดหน่วยความจำเท่าใด กรณีการประมวลผลของวงจรกรองเชิงเลขอันดับ4 หน่วยความจำประเภท

ROM หรือ EPROMที่ใช้จะมีขนาดเท่ากับ 32*8(256บิท) โดยที่32เป็นจำนวนแอดเดรสที่จะแมพค่าต่างๆในหน่วยความจำ จากวงจรดังรูปที่2จะเลือกใช้หน่วยความจำประเภท EPROM มาใช้งานแทน ROM ทั้งนี้เนื่องจากสามารถที่จะโปรแกรมค่าสัมประสิทธิ์ค่าใหม่แทนค่าเดิมได้ โดยเลือกใช้ EPROM เบอร์ 2716 จำนวน 2 ตัว

สำหรับการคำนวนค่าที่จะแมพลงแอดเดรสต่างๆภายใน EPROM สามารถคำนวณจากค่าสัมประสิทธิ์ของตัวกรองโดยดูที่ฟังก์ชันถ่ายโอน

ส่วนประมวลผลทางคณิตศาสตร์ (Arithmetic Unit)

จากสมการที่ (3) และ (4) สังเกตุเห็นได้ว่าในแต่ละสมการดังกล่าว จะประกอบไปด้วย การบวก การเลื่อนข้อมูล ตลอดจนการลบเท่านั้น เนื่องจากเลือกใช้ขนาดความยาวของคำเป็น 8 บิท โดยเป็นระบบตัวเลขแบบส่วนเติมเต็มเป็นสอง ดังนั้นในแต่ละสมการของสมการที่ (3) และ (4) จะประกอบไปด้วยการบวก และการเลื่อนข้อมูล 7 ครั้ง และอีกหนึ่งครั้งสำหรับการลบ เพื่อที่จะคำนวณค่า v_n และ y_n ที่เป็นลำดับสัญญาณเอาท์พุทในแต่ละส่วน ค่าของฟังก์ชัน F(...) สามารถที่จะนำไปเก็บไว้ภายในหน่วยความจำประเภทรอมได้อย่างสะดวก ในลักษณะของข้อมูลที่เข้าไปในลักษณะขนาน ดังนั้นจึงต้องเลือกส่วนประมวลผลที่มีลักษณะสอดคล้องกับการกระทำดังกล่าว คือ ส่วนประมวลผลคณิตศาสตร์ในลักษณะขนาน (parallel arithmetic) ถ้าหากเลือกใช้ส่วนประมวลผลแบบอนุกรม (serial arithmetic) จะต้องเปลืองทางฮาร์ดแวร์เพื่อที่เปลี่ยนรูปแบบส่วนประมวลผลมาเป็นลักษณะขนาน

สำหรับการคำนวณในกระบวนการลบ ตัวเลขแบบส่วนเติมเต็มเป็น 2 สามารถเลือกตัวประมวลผลทางคณิตศาสตร์ ที่สามารถคำนวณการลบได้ภายในตัวเอง (built-in) นอกจากนี้ ภายหลังที่มีการประมวลทางคณิตศาสตร์ที่ตัวประมวลผล จะต้องมีการเลื่อนผลลัพธ์ที่ได้มาเก็บที่ตัวรวบรวมข้อมูล (Accumulator) ขนาด10บิท (IC 8202)ซึ่งเป็นอุปกรณ์ประเภท D FLIP FLOP ที่ตัวรวบรวมข้อมูลนี้จะมีการป้อนกลับ พร้อมกับเลื่อนข้อมูลดังกล่าวไปทางขวา 1 บิทไปยังอินพุทของตัวประมวลผลทางคณิตศาสตร์

รูปที่ (4),(5) แสดงวงจรของส่วนประมวลผลทางคณิตศาสตร์ จะเห็นได้ว่ามี

การใช้วงจรบวกเลข(ADDER)ขนาด4บิทจำนวน2ตัวเป็นALU(IC TTL เบอร์ 7483) โดยผ่านวงจรเอกซ์คลูซีฟออร์เกท(EXCLUSIVE OR GATE)IC TTL เบอร์ 7486 เพื่อเลือกว่าจะให้วงจรบวกทำหน้าที่บวกหรือลบ โดยจะใช้สัญญาณควบคุมCOมาควบคุมซึ่งจะทำให้ALUทำการลบที่รอบของสัญญาณนาฬิกาที่ 8 ของแต่ละรอบของการสุ่มสัญญาณแต่ละครั้งซึ่งก็คือบิทที่มีนัยสัมคัญสูงสุด(MSB)นั่นเอง

นอกจากนี้ส่วน ALU ยังมีส่วนที่เพิ่มเติมขึ้นมาอีกในการทำงาน กล่าวคือก่อนการโหลดค่าจากแอคคิวมูเลเตอร์ขนาด 10 บิทอย่างขนานมายังบัพเฟอร์ เนื่องจากสัญญาณอินพุทที่เข้ามายังแอคคิวมูเลเตอร์ก็คือเอาท์พุทของวงจรบวกนั่นเอง และเอาท์พุทส่วนหนึ่งจะถูกป้อนกลับไปเข้าวงจรบวกอีกครั้งหนึ่งพร้อมทั้งเลื่อนข้อมูลไปทางขวา1บิท เพื่อที่ต้องการจะหลีกเลี่ยงการเลื่อนข้อมูลผิดพลาดไปยังตำแหน่งของบิทเครื่องหมาย(SIGN BIT)ซึ่งจะก่อให้เกิดโอเวอร์โฟลขึ้นที่วงจรบวก ดังนั้นจึงต้องมีวงจรเพิ่มเติมสำหรับการตรวจสอบบิทเครื่องหมาย โดยใช้เอกซ์คลูซีฟออร์เกท (IC TTL เบอร์ 7486)นั่นคือจะเอาบิทเครื่องหมายของข้อมูลที่เข้าวงจรบวกและบิทตัวทด($a^{\circ}, b^{\circ}, c^{\circ}$) จากวงจรบวกถ้า a+b=d ดังนั้นบิทเครื่องหมายของผลลัพธ์หลังจากเลื่อนข้อมูล(d)ไปทางขวา 1 บิท($2^{-1}d$) หาได้จากสมการ

$$d^{\circ} = a^{\circ} \oplus b^{\circ} \oplus c^{\circ}$$

อินพุท/เอาท์พุท (INPUT/OUTPUT)

ลำดับสัญญาณอินพุท (x_n)เกิดจากการแซมปลิ้งสัญญาณเข้าด้วยคาบ $T = 1/f_s$ วินาที โดยที่ f_s เป็นค่าความถี่ที่ใช้ในการแซมปลิ้งสัญญาณ ในขั้นตอนนี้อุปกรณ์ที่ใช้เปลี่ยนสัญญาณต่อเนื่องที่เข้ามาเพื่อแปลงให้เป็นลำดับสัญญาณ คือ อุปกรณ์แซมเปิลแอนด์โฮลด์(sample and hold) โดยสามารถใช้ลิเนียร์ไอซีเบอร์ LF 398 ซึ่งเป็นไอซีสำเร็จรูปโดยความถี่ที่ใช้แซมปลิ้งสัญญาณจะขึ้นอยู่กับสัญญาณนาฬิกาที่ป้อนเข้าที่ตัว LF 398 หลังจากสัญญาณผ่านวงจรแซมเปิลแอนด์โฮลด์แล้วจะถูกต่อผ่านมายังอุปกรณ์แปลงสัญญาณเชิงอุปมานไปเป็นสัญญาณเชิงเลข(A/D) โดยจะใช้ IC เบอร์ ADC 0800 ซึ่งมีคอนเวอร์ชั่นไทม์(CONVERSION TIME)เท่ากับ 50μs หรือความถี่ที่A/D ใช้คอนเวิตท์สัญญาณก็คือ20khzนั่นเอง แต่เนื่องจากในวงจรกรองเชิงเลขระบบตัวเลขที่ใช้ก็คือระบบเลขส่วนเติมเต็มเป็นสอง ดังนั้นเอาท์พุทของADC0800เป็นระบบตัว

เลขแบบคอมพลีเมนท์ทารี(COMPLEMENTARY) กล่าวคือสัญญาณเอาท์พุทที่แบ่งออกเป็น 2^8(256)ระดับ ที่ระดับต่ำสุดคือ1111 1111 และระดับสูงสุดคือ0000 0000 ดังนั้นจึงต้องเอาบิทที่ 0 ถึงบิทที่ 6(ยกเว้นบิทMSB) มาผ่านอินเวร์ทเตอร์ (IC 7404) ก่อนที่จะผ่านเข้าไปยังส่วนประมวลผลของวงจรกรอง

สำหรับทางด้านเอาท์พุทจะใช้วงจรแปลงสัญญาณเชิงเลขไปเป็นสัญญาณเชิงอุปมาน(D/A)เพื่อแปลงลำดับสัญญาณY_nกลับไปเป็นสัญญาณเชิงอุปมานตามต้องการ โดยที่ D/A ที่ใช้คือ DAC0800 ซึ่งมีคอนเวร์ชันไทม์เท่ากับ100μs และเนื่องจากD/AC0800นี้ลำดับสัญญาณที่เข้ามาจะต้องเป็นระบบตัวเลขฐานสอง ดังนั้นจะต้องเอาบิทที่ 7(MSB) ของลำดับสัญญาณมาผ่านอิเวร์ทเตอร์(IC 7404)ก่อนที่จะผ่านD/A

ส่วนควบคุม (Control)

วงจรควบคุม มีไว้เพื่อ ควบคุมอุปกรณ์ต่างๆภายในวงจรกรองเชิงเลขให้ทำงานสอดคล้องซึ่งกันและกันเพื่อที่จะสร้างเอาท์พุทที่ถูกต้องตามต้องการ วงจรควบคุมโดยทั่วไปจะผลิตสัญญาณนาฬิกาให้สอดคล้องกับการทำงาน และ คงสภาพของระดับสัญญาณนั้นๆไว้ตามช่วงเวลาที่เหมาะสมกับการทำงานนั้นๆ

ตามรูปที่(6)แสดงวงจรสำหรับส่วนควบคุมที่ใช้ควบคุมส่วนต่างๆของวงจรกรอง ส่วนรูปที่(7)แสดงไทม์มิ่งไดอะแกรมของสัญญาณควบคุมต่างๆที่ใช้ โดยที่สัญญาณควบคุมแต่ละสัญญาณจะมีลักษณะที่แตกต่างกันออกไป โดยอาศัยคณิตศาสตร์บูลีนดังตารางที่(1) เพื่อสร้างสัญญาณควบคุมจากสัญญาณเอาท์พุท A, B, C, D, E,และFจากเคาท์เตอร์ขนาด 6 บิท(ซึ่งนับสัญญาณนาฬิกาที่ป้อนเข้ามาเริ่มจาก 0 ถึง 63 จึงจะรีเซ็ท) ดังนั้นสัญญาณนาฬิกาที่เข้ามายังส่วนควบคุมจะมีทั้งสิ้น 64 ลูกต่อการประมวลผลหนึ่งรอบ โดยจะถูกใช้ในวัฏจักรของROM1(ROM1 CYCLE) 32 ลูกและวัฏจักรของROM2(ROM2 CYCLE)อีก32ลูก

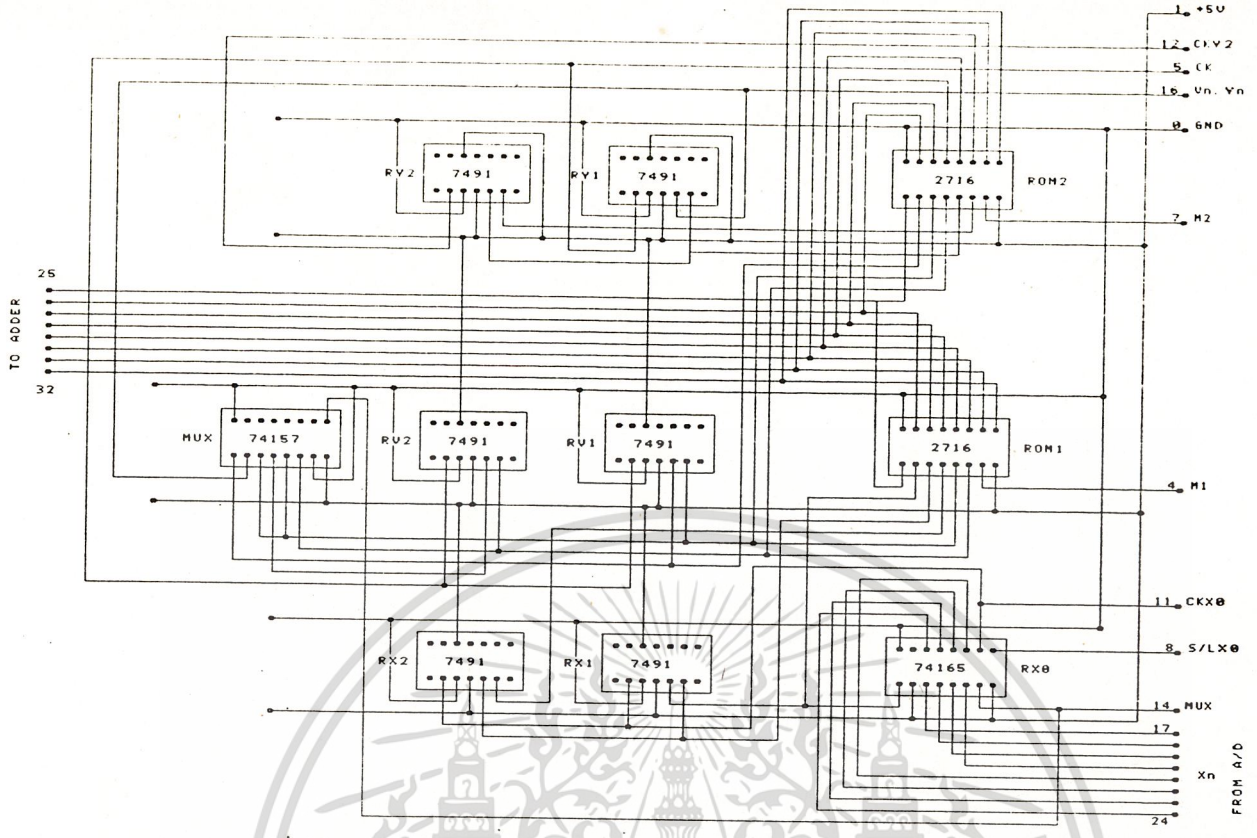
ตารางที่ (2) เป็นตารางที่แสดงเหตุการณ์ต่างๆที่เกิดขึ้นในแต่ละส่วนของการประมวลผลในส่วนต่างๆของวงจรกรอง

ตารางที่1. แสดงสัญญาณควบคุมที่ต้องการใช้งาน

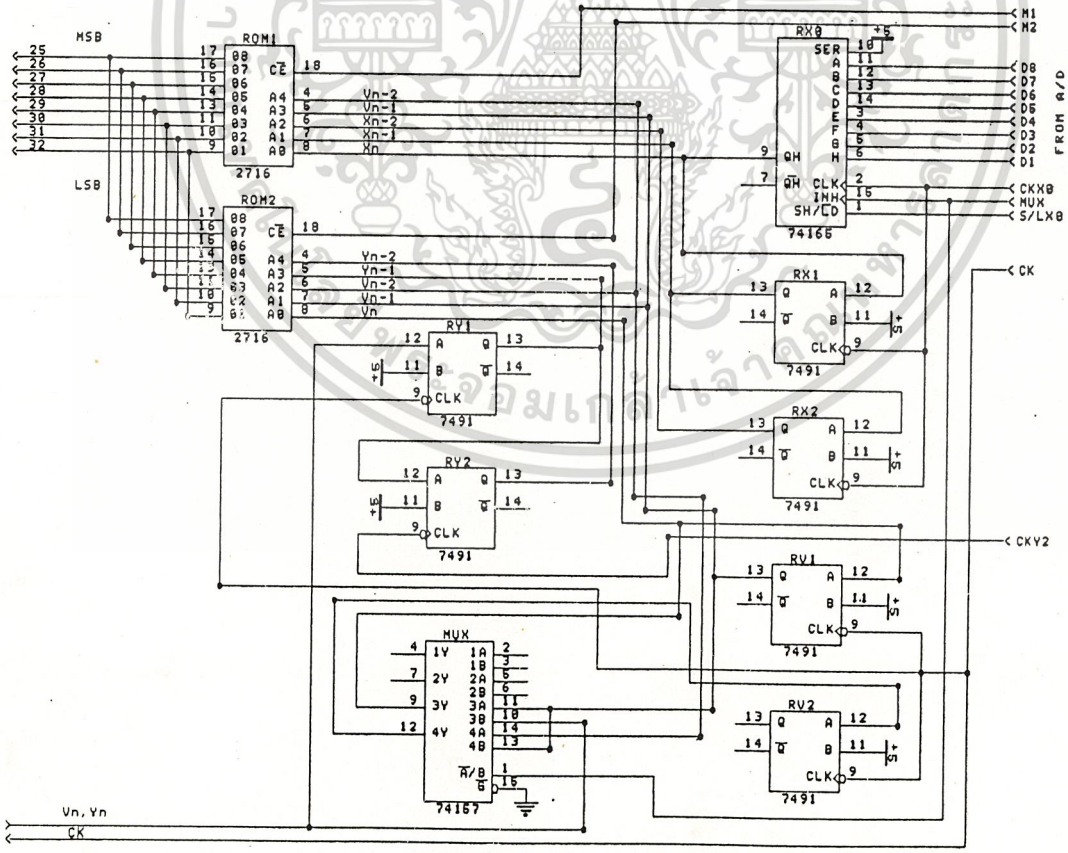
A/D Converter:	
Sample & Hold(LF 398)	
A / D (ADC0800)	$SC=F.(E.\overline{D}.\overline{C}.).B$
Memory:	ME = A.B
EPROM1(2716)	M1 = F + ME
EPROM2(2716)	$M2 = \overline{F} + ME$
Adder:	
Carry-in & Complement signal	$CO = E.\overline{D}.C$
Round	Round = E.D.C
Shift Register:	
RV1,RV2,RY1(7491)	$Clock = CK = \overline{A.B}$
RX1,RX2(7491)	$Clock = CKX0 = \overline{CK.MUX}$
RX0(74165)	$Load = S/LX0 = \overline{ME.CO}$
	Clock Inhibit = MUX
	Clock = CKX0
RY2(7491)	Clock = CKY2 = CK.MUX
BRF1(74165)	Load = S/LX0
	Clock Inhibit = GND
Multiplexer:	
MUX(74157)	Select = MUX
Buffer Register:	
ACC(8202)	$Clear = CLACC = \overline{\overline{A.B}.(\overline{C.D.E})}$
	Load = LACC = CK.CI
BFR2(74198)	Clock = F.CO.ME

ตารางที่2. แสดงถึงเหตุการณ์ที่เกิดขึ้น ณ ช่วงเวลาต่างๆ

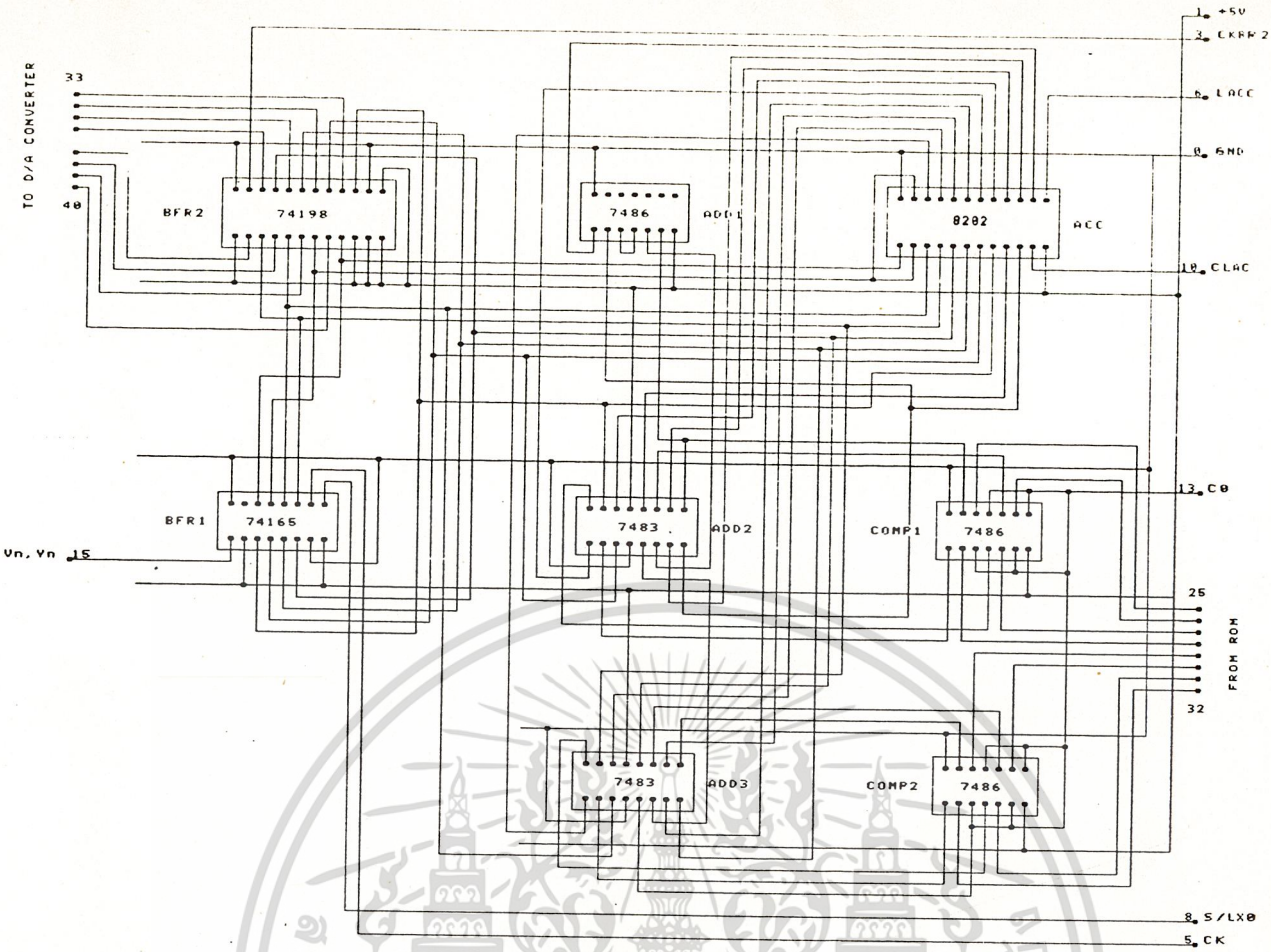
Count Computation	Event
Period 64	Cycle
0-31	คำนวณ v_n
	Sample Analog Signal
32-63	คำนวณ y_n
34-63	A/D Conversion
63	ข้อมูลถูกโหลดเข้า RX0,BFR2
การคำนวณ v_n/y_n	
Period 32	Cycle
28-31	Complementเอาท์พุทจากEPROM
0	Clear ACC
One Accumulation	
Period 6	Cycle
0	Enable หน่วยความจำ EPROM1
	และ EPROM2
2	Load ACC
	Shift(RX0,RX1,RX2),(RV1,
	RV2,BFR1),(RY2)
3	Disable หน่วยความจำ

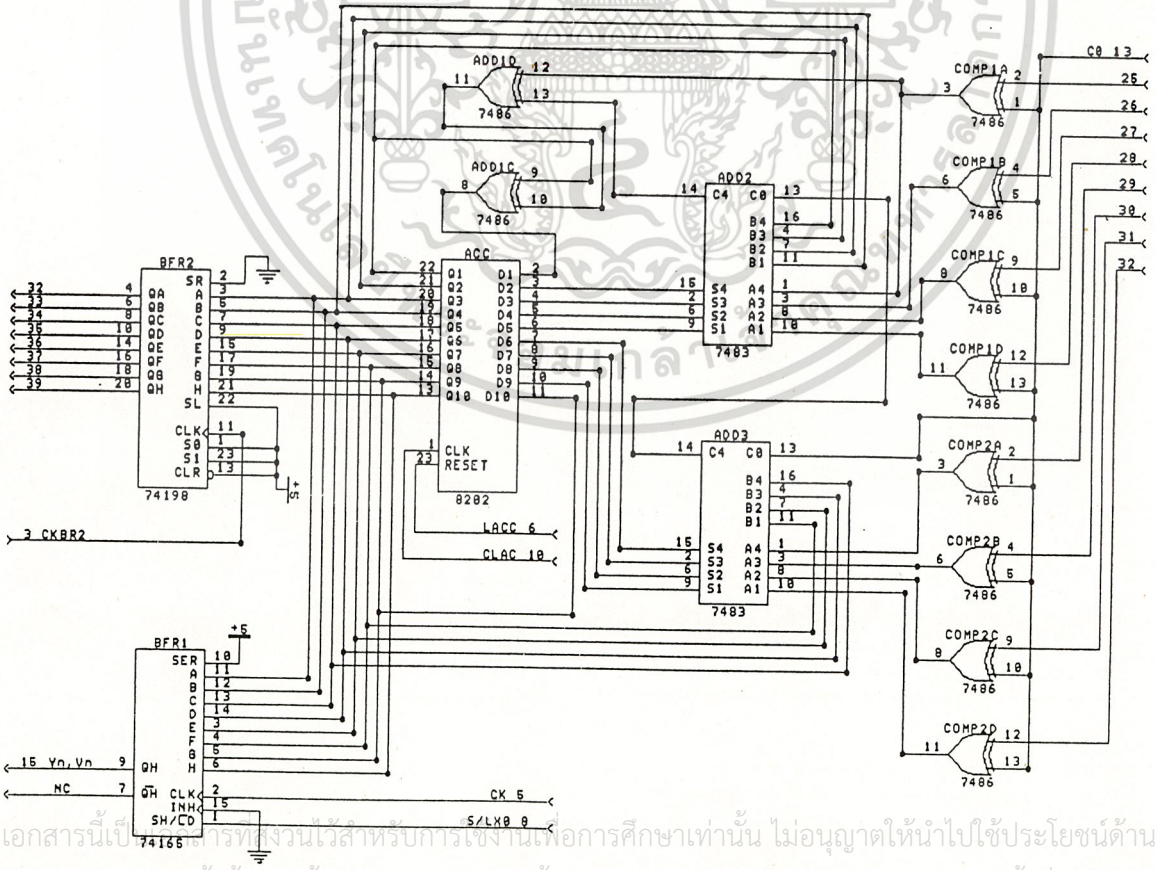


MEMORY AND REGISTER SECTION

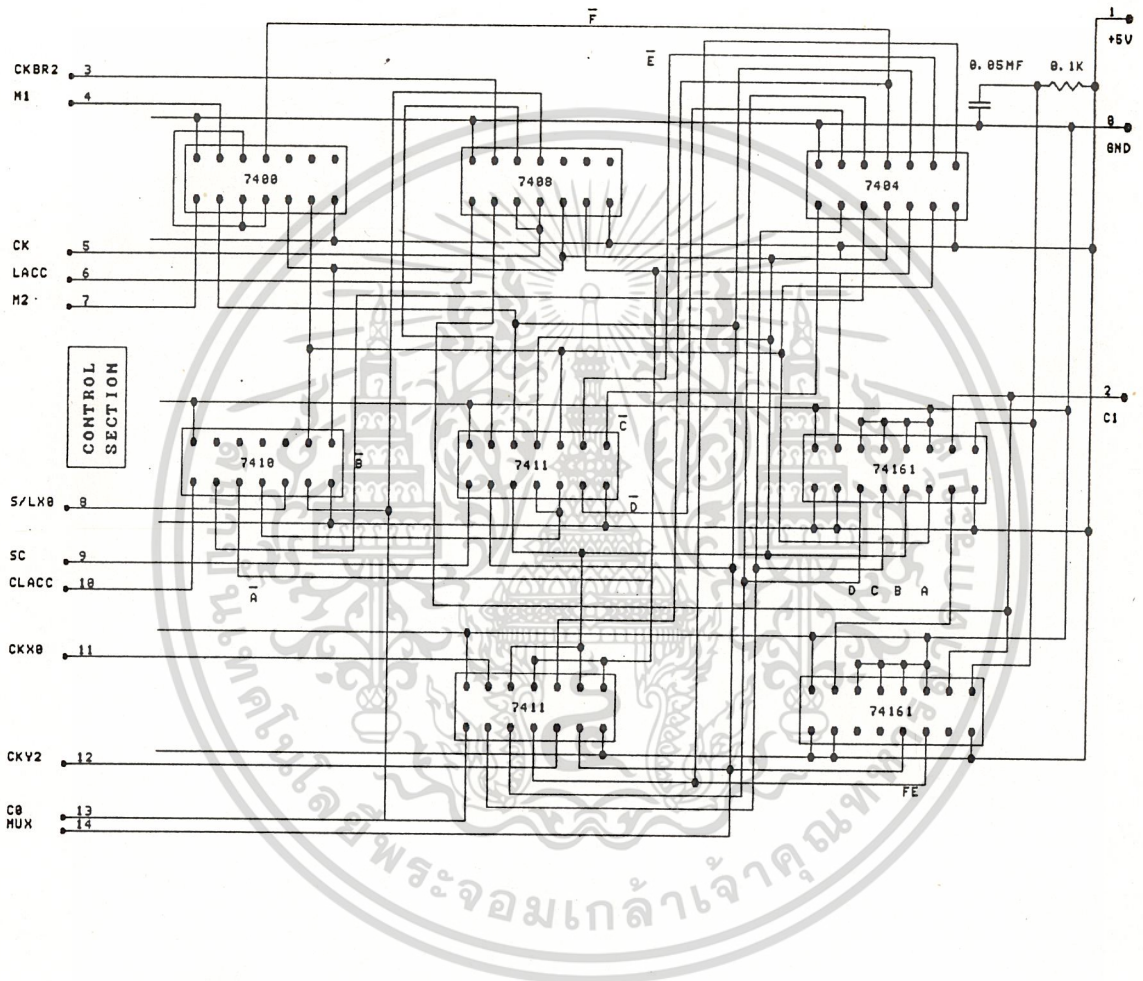


รูปที่3. วงจรของหน่วยความจำและรีจีสเตอร์

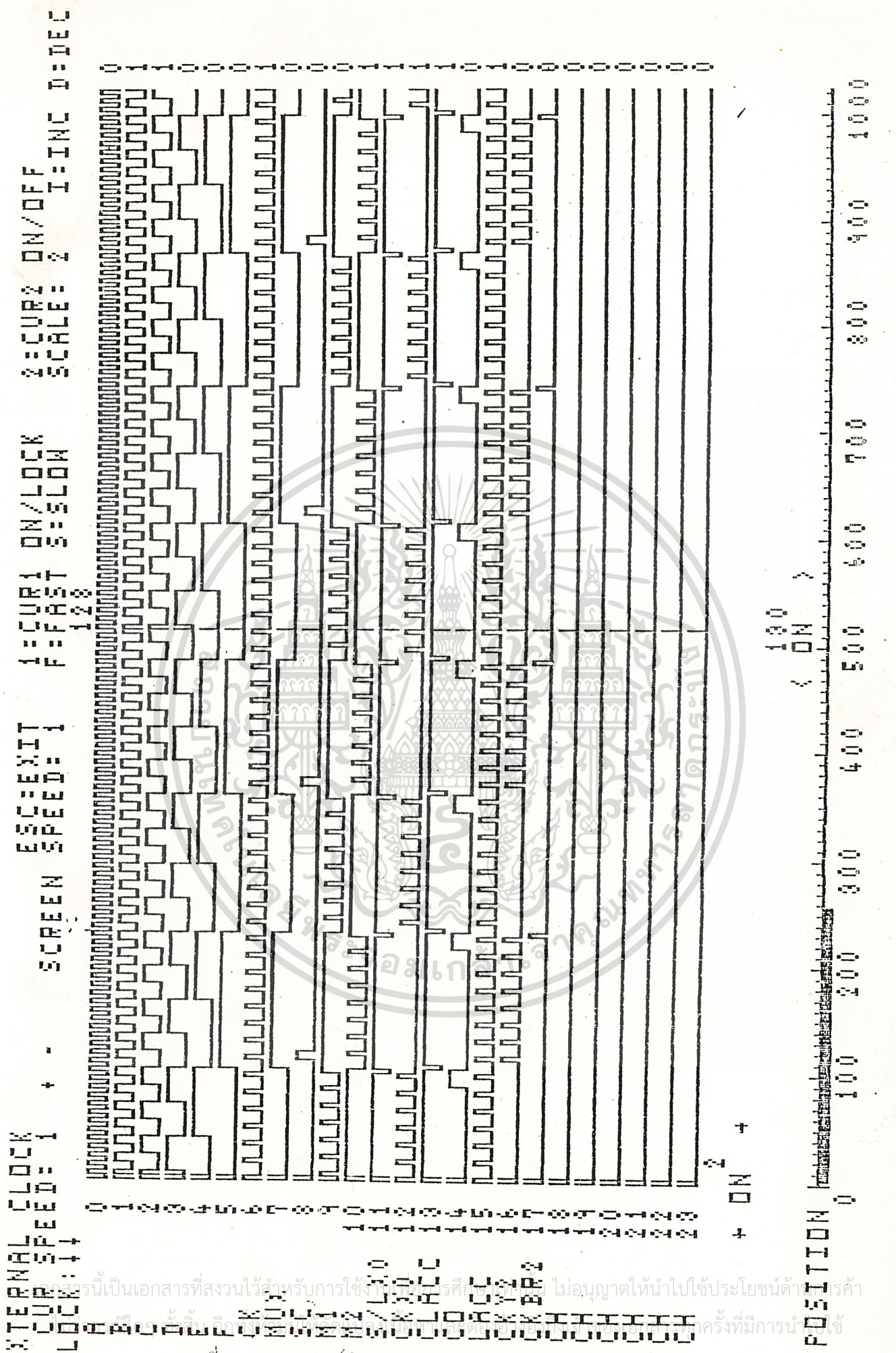




รูปที่4และ5. วงจรของส่วนประมวลผลทางคณิตศาสตร์



รูปที่6. วงจรของส่วนควบคุม



รูปที่7. แสดงไทม์มิงไดอะแกรมของสัญญาณควบคุม

บทที่5. การทดสอบวงจรกรองเชิงเลขอันดับที่4

ก่อนเริ่มการทดลองขั้นแรกจะต้องเลือกฟังก์ชันถ่ายโอนของตัวกรอง เพื่อที่จะหาค่าสัมประสิทธิ์ของตัวกรองมาเก็บไว้ในหน่วยความจำ(EPROM 2716) ในการทดสอบฟังก์ชันถ่ายโอนที่เลือกมาทดสอบคือ

$$\frac{1-3.05271Z+4.28714Z^2-3.05271Z^3+Z^4}{0.611078-2.58952Z+4.29167Z^2-3.29306Z^3+Z^4} \qquad \text{-----(1)}$$

ซึ่งเป็นฟังก์ชันถ่ายโอนของตัวกรองชนิดผ่านความถี่ต่ำแบบอีลิปติก (Elliptic Filter)ที่มีจุดตัดความถี่(Cut Off Frequency)อยู่ที่ 1.5 KHz และโพล(Pole) ของระบบอยู่ที่ 0.771962+j0.2276434และ 0.8745678+j0.422517 ส่วนซีโร (Zero) อยู่ที่0.6599549+j0.751302และ 0.8664001+j0.4993505

จากสมการที่ (1) สามารถแยกออกเป็นฟังก์ชันถ่ายโอนของตัวกรองอันดับสองได้ดังต่อไปนี้

ส่วนที่1. $\dfrac{0.5-0.8664Z+0.5Z^2}{0.4717145-0.87459Z+0.5Z^2}$

$a_o = 0.5$	$\bar{a}_o$ = 0100 0000
$a_1 = -0.8664$	$\bar{a}_1$ = 1001 0001
$a_2 = 0.5$	$\bar{a}_2$ = 0100 0000
$-b_1 = 0.87459$	$-\bar{b}_1$ = 0110 1111
$-b_2 = -0.4717145$	$-\bar{b}_2$ = 1100 0011

ส่วนที่2. $\dfrac{0.5-0.659955Z+0.5Z^2}{0.32386-0.77194Z+0.5Z^2}$

$a_o = 0.5$	$\bar{a}_o$ = 0100 0000
$a_1 = -0.659955$	$\bar{a}_1$ = 1010 1011
$a_2 = 0.5$	$\bar{a}_2$ = 0100 0000
$-b_1 = 0.77194$	$-\bar{b}_1$ = 0110 0010
$-b_2 = -0.32386$	-b2 = 1101 0110

ดังนั้นค่าที่จะถูกแมพลงในหน่วยความจำ(EPROM 2716)ทั้งสองตัวคือ

ตารางที่1. แสดงค่าที่อยู่ภายในหน่วยความจำ

ADDRESS					EPROM1	EPROM2
0	0	0	0	0	0000 0000	0000 0000
0	0	0	0	1	1100 0011	1101 0110
0	0	0	1	0	0110 1111	0110 0010
0	0	0	1	1	0011 0010	0011 1000
0	0	1	0	0	0100 0000	0100 0000
0	0	1	0	1	0000 0011	0001 0110
0	0	1	1	0	1010 1111	1010 0010
0	0	1	1	1	0111 0010	0111 1000
0	1	0	0	0	1001 0001	1010 1011
0	1	0	0	1	0101 0100	1000 0000
0	1	0	1	0	0000 0000	0000 1101
0	1	0	1	1	1100 0011	1110 0011
0	1	1	0	0	1101 0001	1110 1011
0	1	1	0	1	1001 0100	1100 0001
0	1	1	1	0	0100 0000	0100 1101
0	1	1	1	1	0000 0011	0010 0011
1	0	0	0	0	0100 0000	0100 0000
1	0	0	0	1	0000 0011	0001 0110
1	0	0	1	0	1010 1111	1010 0010
1	0	0	1	1	0111 0010	0111 1000
1	0	1	0	0	1000 0000	1000 0000
1	0	1	0	1	0100 0011	0101 0110
1	0	1	1	0	1110 1111	1110 0010
1	0	1	1	1	1011 0010	1011 1000
1	1	0	0	0	1101 0001	1110 1011
1	1	0	0	1	1001 0100	1100 0001
1	1	0	1	0	0100 0000	0100 1101
1	1	0	1	1	0000 0011	0010 0011
1	1	1	0	0	0001 0001	0010 1011
1	1	1	0	1	1101 0100	0000 0001
1	1	1	1	0	1000 0000	1000 1101
1	1	1	1	1	0100 0011	0110 0011

ประสิทธิภาพสูงสุดของวงจร

- บิทเรท(Bit Rate)ของวงจรสามารถคำนวณได้จากสมการที่ 2เมื่อ$\log_2 B$ มีค่าเป็นจำนวนเต็มแต่ถ้า $\log_2 B$ มีค่าเป็นเลขทศนิยมแล้วให้ใช้สมการที่ (3)โดยที่ $[1+\log_2 B]$หมายถึงเลขจำนวนเต็มที่ปัดเศษทิ้งไป

$$\text{บิทเรท} = \frac{1}{2\{(1+\log_2 B)T\}} \qquad \text{----------(2)}$$

$$= \frac{1}{2\{(1+[1+\log_2 B])T\}} \qquad \text{----------(3)}$$

เมื่อ B = จำนวนบิทที่ใช้

T = ค่าที่มากที่สุดระหว่างแอกเซสไทม์ของหน่วยความจำ(t_m)และแอกเซสไทม์ของวงจรบวก(t_a)

ในการทดสอบ B = 8 ,t_m = 350 ns,t_a = 23 ns

ดังนั้นค่าบิทเรทสูงสุดของวงจรจึงเท่ากับ 0.35 MHz

- ความถี่ที่ใช้ในการแซมปลิง(Sampling Frequency)สูงสุด เนื่องจากวงจรกรองประมวลผลสัญญาณที่สุ่มมา 1 รอบจะใช้สัญญาณ CK 16ลูก

$$\text{แซมปลิงเรทสูงสุด} = \frac{0.35}{16} = 21.875 \text{ KHz}$$

- ความถี่ที่ A/D ใช้คอนเวอร์ทต้องมีค่าไม่เกิน 21.875 KHz

- ความกว้างของแถบความถี่ของสัญญาณอินพุท = $\frac{21.875}{2}$ = 10.9 KHz

- ความถี่ของสัญญาณ CI = 21.875*64 = 1.4 MHz

- ความถี่ของสัญญาณ CK = 1.4/4 = 0.35 MHz (จากไทม์มิงไดอะแกรมความถี่ของสัญญาณCIเป็นสี่เท่าของสัญญาณCK)

การทดลอง

ความถี่ของสัญญาณอินพุทที่ใช้

เนื่องจากวงจรแซมเปิลแอนด์โฮล (LF398)ทำการสุ่มตัวอย่างสัญญาณที่ความถี่ 20 KHz ดังนั้นจากทฤษฎีของไนควิสต์(Nyquist Theorem) สัญญาณที่เข้ามาจะต้องมีค่าไม่เกิน 10 KHz แต่จากการทดลองความถี่ที่ใช้ได้ดีเมื่อสัญญาณมีค่าอยู่ในช่วง 0 - 3.5 KHz เท่านั้น(แซมปลิงเรท>=6เท่าของสัญญาณอินพุท)

กล่าวโดยสรุปคือสัญญาณที่ใช้จะมีความกว้างของแถบความถี่ (Band Width) เท่ากับ 3.5 KHz นั่นเอง สำหรับในการทดสอบจะใช้ความถี่เท่ากับ 2.5 KHz

ความถี่ของสัญญาณนาฬิกา(CI)

ด้วยเหตุที่ ADC0800 ซึ่งเป็นอุปกรณ์แปลงสัญญาณเชิงอุปมานเป็นสัญญาณเชิงเลขมีคอนเวอร์ชันไทม์เท่ากับ 50μs หรือความถี่ในการคอนเวิร์ทเท่ากับ 20 KHz ดังนั้นสัญญาณ CI ที่ใช้จะมีความถี่เท่ากับ 64*20 = 1.28 MHz(ไม่เกิน1.4MHz)

นอกจากนี้ยังมีอีกวิธีหนึ่งที่ใช้คำนวณค่าของสัญญาณ CI โดยพิจารณาจากไทม์มิงไดอะแกรมได้ดังนี้

ประมวลผล 1 รอบ(แซมปลิง 1 ครั้ง)ใช้สัญญาณ CK 16 ลูก
ทำการแซมปลิงที่ 20 KHz ดังนั้นใช้สัญญาณ CK = 16*20 KHz
= 320 KHz
(ไม่เกิน350KHz)
จากไทม์มิงไดอะแกรมสัญญาณ CI มีความถี่เป็น4เท่าของสัญญาณ CK
ดังนั้น สัญญาณ CI จะมีความถี่ = 4*320 KHz
= 1.28 MHz

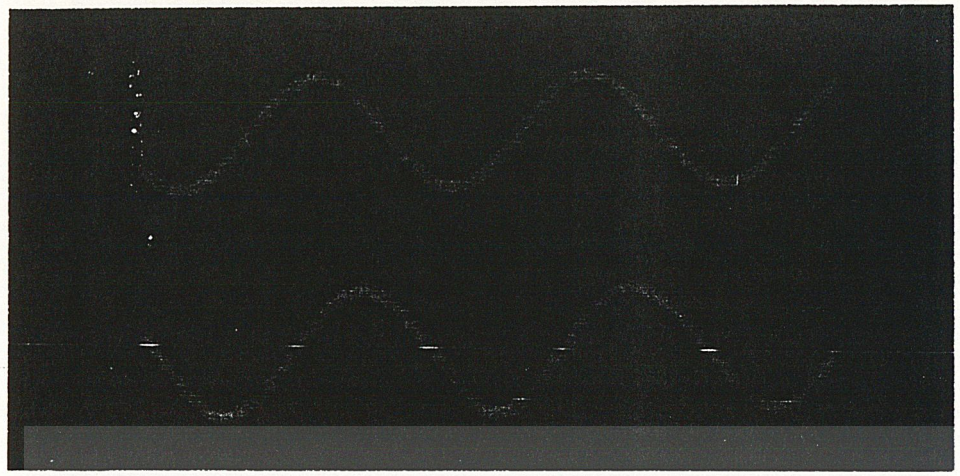
สำหรับในการทดสอบใช้คริสตอลเป็นตัวผลิตความถี่(Crystal Oscilator) ได้สัญญาณ CI ความถี่ประมาณ 1.2 MHz

ผลการทดลอง

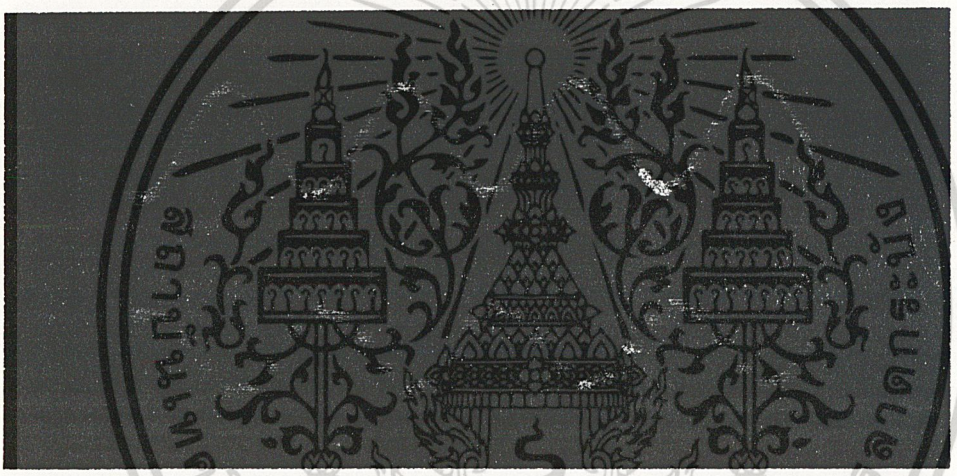
ตาราง**ที่2.** คือตารางบันทึกผลการทดลองโดยบันทึก ณ ค่าความถี่ต่างๆ(เพิ่มความถี่ทีละ100Hz)

ตารางที่2. แสดงผลการทดลอง

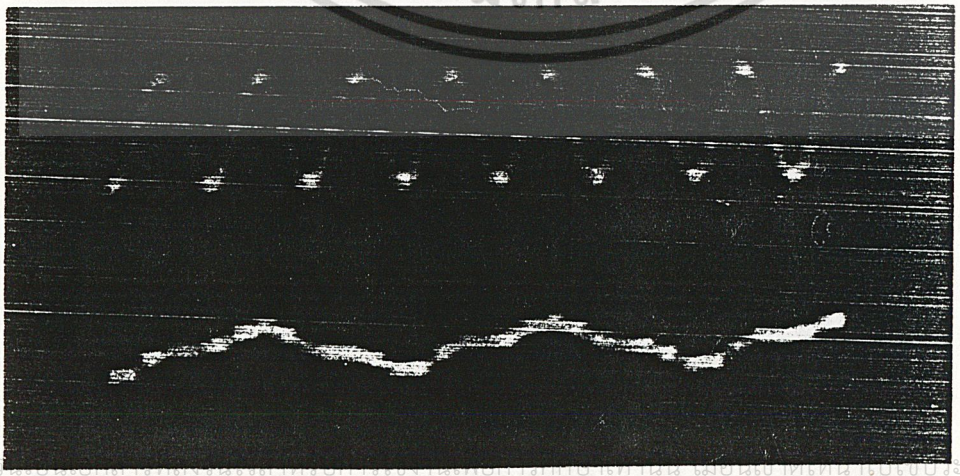
ความถี่ (KHz)	Vpp(v)	Gain
0	20.0	1.00
0.1	20.0	1.00
0.2	20.2	1.01
0.3	20.6	1.03
0.4	21.6	1.04
0.5	21.4	1.08
0.6	21.0	1.07
0.7	20.6	1.05
0.8	20.6	1.03
0.9	19.6	0.98
1.0	19.2	0.96
1.1	18.8	0.94
1.2	21.6	1.08
1.3	20.0	1.00
1.4	14.0	0.70
1.5	13.6	0.68
1.6	1.2	0.06
1.7	0.8	0.04
1.8	1.6	0.08
1.9	1.6	0.08
2.0	1.4	0.07
2.1	1.4	0.07
2.2	1.2	0.06
2.3	1.0	0.05
2.4	0.6	0.03
2.5	0.2	0.01
2.6	0.0	0.00
2.7	0.4	0.02
2.8	0.4	0.02
2.9	0.6	0.03
3.0	0.8	0.04



รูปที่1. แสดงสัญญาณที่ความถี่800Hz



รูปที่2. แสดงสัญญาณที่ความถี่1.5KHz



รูปที่3. แสดงสัญญาณที่ความถี่2KHz

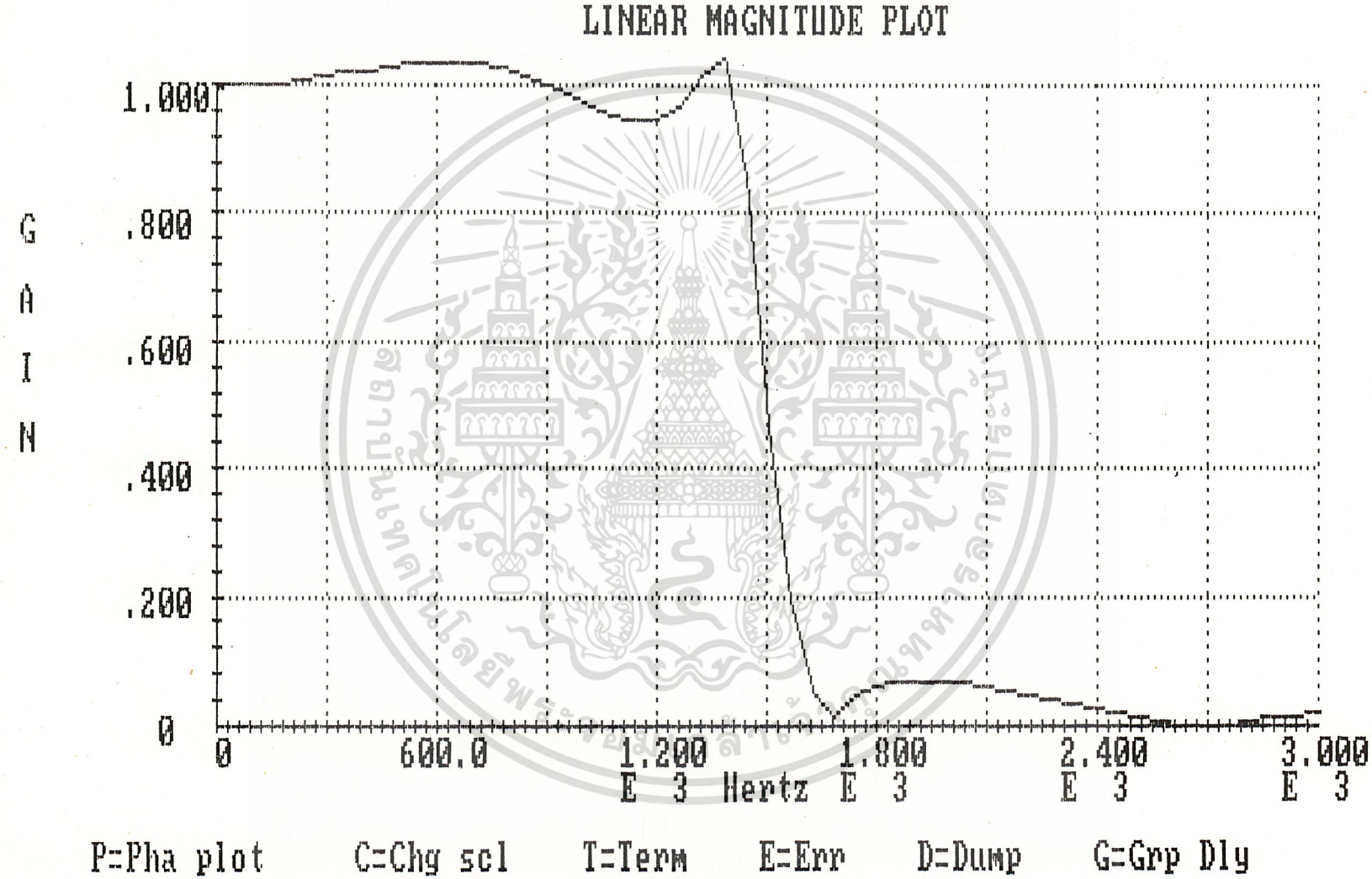
จากผลการทดลองที่ได้ที่ความถี่1.5Kz ซึ่งเป็นจุดตัดความถี่ที่กำหนดไว้สามารถวัดค่าแอมปลีจูด(Vpp)เท่ากับ 13.6Voltหรือมีเกนประมาณ0.68 ซึ่ง ณ ความถี่นี้ทำให้แอมปลีจูดลดลง32เปอร์เซนต์

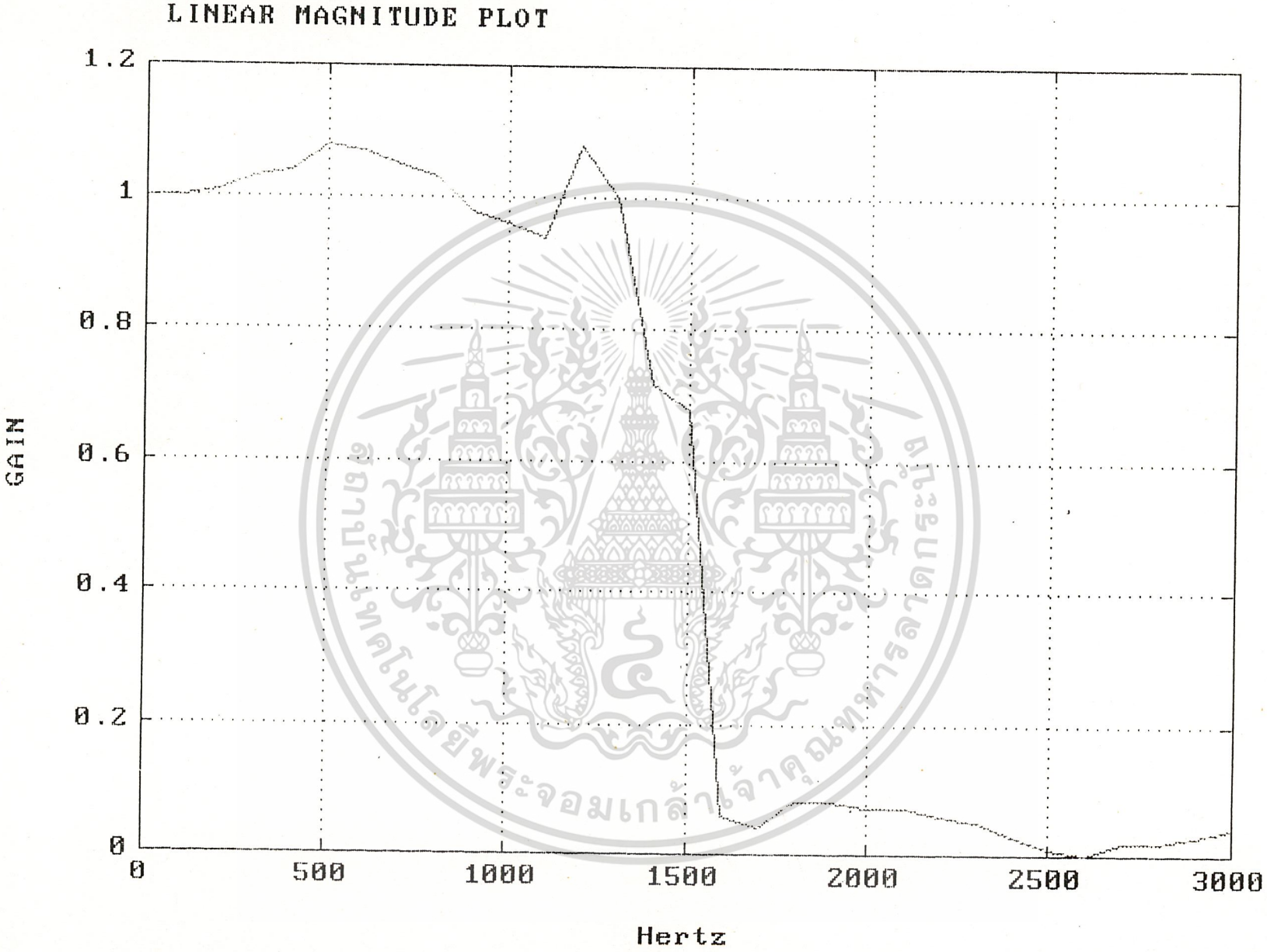
เมื่อทำการเปรียบเทียบกับทฤษฎีกล่าวคือ ณ ตำแหน่งจุดตัดความถี่จะมีผลตอบสนองทางแอมปลีจูด(H(w))เท่ากับ-3.01db หรือเป็นจุดที่กำลังไฟฟ้าออกลดลงครึ่งหนึ่ง นั่นแสดงว่าจุดตัดความถี่ของตัวกรองจะอยู่ที่ความถี่ประมาณ 1.550 KHz โดยดูที่กราฟแสดงลักษณะทางแอมปลีจูดของผลการทดสอบข้างต้น ซึ่งจะเห็นได้ว่าตัวกรองจะมีจุดตัดความถี่ผิดพลาดไปประมาณ 50 Hz จากจุดตัดความถี่ที่กำหนดไว้แต่เดิมทั้งนี้เป็นผลเนื่องมาจากการใช้ความยาวของคำจำกัด(Finite Word Length) (บทแทรก1) ซึ่งมีผลทำให้ตำแหน่งโพลของตัวกรองเลื่อนไปทำให้ผลตอบสนองทางความถี่เปลี่ยนแปลงไปด้วย

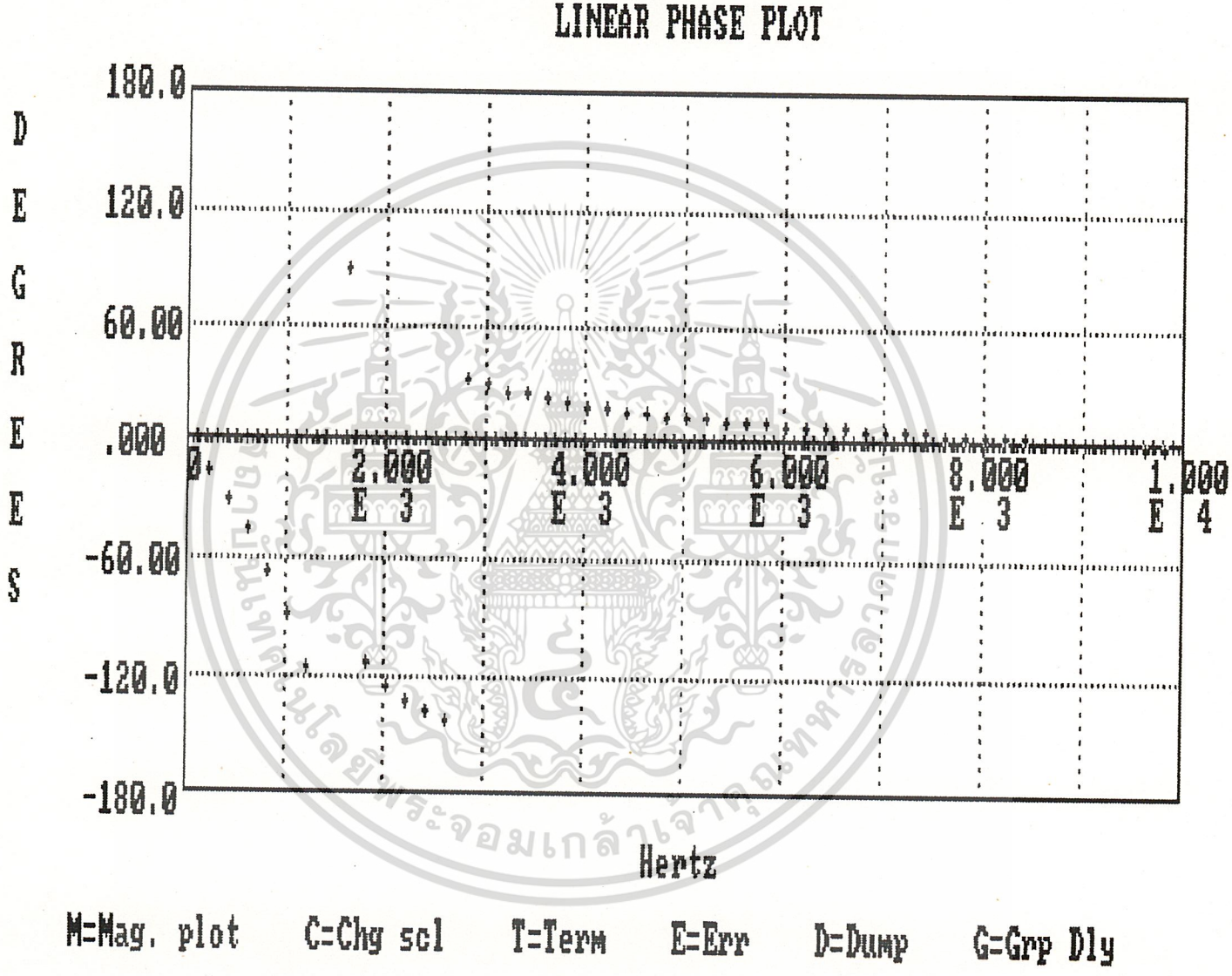
รูปที่4. แสดงสัญญาณที่ความถี่1.550KHz

สรุปผลการทดสอบ

จากการเปรียบเทียบกราฟแสดงคุณสมบัติของตัวกรอง (Linear Magnitude Plot) ระหว่างกราฟที่ได้จากการออกแบบในรูปที่(5) กับกราฟที่ได้จากการทดลองตามตารางที่2. ในรูปที่(6)มีค่าใกล้เคียงกัน ดังนั้นทำให้สามารถสรุปได้ว่าการออกแบบวงจรกรองเชิงเลขอันดับ4โดยใช้โครงสร้างแบบเลขคณิตแจกแจง (Distributed Arithematics)สามารถทำงานได้จริง โดยที่ความกว้างของแถบความถี่(Band Width)ของสัญญาณอินพุทเท่ากับ3.5KHzและสำหรับความเร็วในการ




LINEAR MAGNITUDE PLOT
GAIN
1.2
1
0.8
0.6
0.4
0.2
0
0
500
1000
1500
2000
2500
3000
Hertz


LINEAR PHASE PLOT
180.0
120.0
60.00
.000
-60.00
-120.0
-180.0
DEGREES
0+
2.000 E 3
4.000 E 3
6.000 E 3
8.000 E 3
1.000 E 4
Hertz
M=Mag. plot
C=Chg scl
T=Term
E=Err
D=Dump
G=Grp Dly

ประมวลผลกระทำได้สูงสุดประมาณ0.20μs(4800KHz) จะเห็นได้ว่าการสร้างวงจรกรองสัญญาณเชิงเลขโดยใช้โครงสร้างแบบเลขคณิตแจกแจงนี้ ทำให้วงจรกรองประมวลสัญญาณด้วยความเร็วสูง

การสร้างวงจรกรองเชิงเลขโดยใช้โครงสร้างแบบเลขคณิตแจกแจงนี้ หลักสำคัญอยู่ที่จะต้องหาค่าสัมประสิทธิ์จากฟังก์ชันถ่ายโอนให้ได้ก่อน แล้วนำค่าสัมประสิทธิ์นี้ไปเก็บไว้ในหน่วยความจำ แล้วสัญญาณเข้าจะไปเปิดตารางหน่วยความจำออกมาประมวลผล ดังนั้นการออกแบบวงจรด้วยวิธีนี้สัญญาณออกจะถูกต้องขึ้นอยู่กับค่าสัมประสิทธิ์การทำงานของฮาร์ดแวร์เป็นหลัก จากการทดสอบยังมีปัญหาเกี่ยวกับฮาร์ดแวร์ที่อุปกรณ์แปลงสัญญาณเชิงอุปมานเป็นสัญญาณเชิงเลข(A/D)ที่มีคอนเวอร์ชันไทม์ (Conversion Time)ต่ำและที่หน่วยความจำที่มีค่าแอกเซสไทม์(Acces Time)สูงจึงมีผลทำให้ความกว้างของแถบความถี่(Band Width)ของสัญญาณอินพุทต่ำไปด้วย ยังผลให้ไม่สามารถใช้งานวงจรกรองที่ออกแบบไว้ให้ได้ประสิทธิภาพสูงสุดตามต้องการ

แนวการพัฒนาต่อไป

การสร้างวงจรกรองโดยใช้โครงสร้างแบบเลขคณิตแจกแจงนี้ นอกจากสร้างฮาร์ดแวร์เป็นแบบอนุกรมแล้วยังสามารถสร้างโดยใช้โครงสร้างแบบขนานได้อีกซึ่งจะทำให้ความเร็วในการประมวลผลรวดเร็วยิ่งขึ้น นอกจากนี้โครงสร้างแบบเลขคณิตแจกแจงยังสามารถประยุกต์เข้ากับอัลกอริทึม(Algorithm) อื่นๆได้อีกมากเช่น ฟาสฟูเรียทรานฟอร์ม(Fast Fourier Transform) ดิสครีทฟูเรียทรานฟอร์ม(Discrete Fourier Transform) หรือสามารถสร้างวงจรกรองให้มีอันดับของตัวกรองสูงขึ้นเช่นวงจรกรองอันดับหกได้

บทแทรกที่1. การสร้างตัวกรองเชิงเลขด้วยรีจีสเตอร์ความยาวค่าจำกัด

(Implementation of Digital Filter with Finite Register Length)

การออกแบบตัวกรองเชิงเลขตามหัวข้อที่ผ่านมา เป็นการออกแบบที่ไม่มีเงื่อนไขบังคับกับค่าสัมประสิทธิ์ของตัวกรอง โดยถือเสมือนว่าสัมประสิทธิ์ของตัวกรองมีความละเอียดสูง การออกแบบเหล่านี้อาจมีปัญหาในทางปฏิบัติที่เมื่อนำไปสร้างใช้งานตัวกรองที่ได้อาจมีผลตอบสนองความถี่ผิดเพี้ยนไปจากที่ได้ออกแบบเอาไว้ โดยเฉพาะกับตัวกรองผลตอบสนองแอมปลิจูดแคบ การนำไปสร้างใช้งานอาจทำให้ตัวกรองไม่เสถียรได้ ทั้งนี้เนื่องจากว่าเมื่อนำตัวกรองไปสร้างเป็นฮาร์แวร์ ค่าสัมประสิทธิ์ของตัวกรอง และ ค่าของสัญญาณต่างๆต้องถูกแทนด้วยเลขฐานสองที่มีจำนวนบิทจำกัด นั่นคือ เราต้องทำการควอนไทซ์ (quantize) ให้มีจำนวนบิทตามเลขฐานสองตามต้องการ จากนั้นอาจทำการปัดเศษให้ได้ค่าใกล้เคียงกับ ค่าสัมประสิทธิ์ที่ออกแบบไว้มากที่สุด ผลก็คือทำให้ค่าสัมประสิทธิ์ผิดพลาดไป ตำแหน่งของโพลและตัวกรองเลื่อนไป ทำให้ผลตอบสนองความถี่ของตัวกรองเปลี่ยนไป

ผลในทางปฏิบัติเหล่านี้ เป็นข้อจำกัดทำให้ตัวกรองบางแบบที่ทางทฤษฎีทำการออกแบบได้แต่ไม่สามารถใช้งานได้

ต่อไปนี้จะพิจารณาผลต่างๆที่กล่าวข้างต้น จากตัวอย่างของตัวกรองป้อนกลับเชิงเลขอันดับที่ 2 ซึ่งเป็นตัวกรองชนิดผ่านแถบความถี่ที่มี ซีโร ที่ +-1 และ โพลที่ $0.9e^{+-j\pi/4}$ หรือที่ 0.6363961+-j6363961 จะได้ฟังก์ชันของตัวถ่ายโอนของตัวกรองนี้ คือ

$$H(Z) = \frac{1-z^{-2}}{1-1.2727922z^{-1}+0.81z^{-2}} \qquad \text{------(1)}$$

จากสมการที่ (1) สามารถเขียนแทนด้วย สมการผลต่างสืบเนื่องได้ดังสมการที่ (2)

$$y_n = x_n - x_{n-2} + 1.2727922y_{n-1} - 0.81y_{n-2} \qquad \text{---(2)}$$

ตามสมการที่ (2) สามารถที่จะนำมาสร้างเป็นโครงสร้างของตัวกรองนี้ได้โดยใช้อุปกรณ์ต่างๆเหมือนหัวข้อที่ผ่านมา

ในการที่จะนำสมการที่ (2) ไปสร้างเป็นตัวกรอง จะต้องแทนค่าสัมประสิทธิ์ดังกล่าวด้วยเลขฐานสอง เพื่อที่จะได้นำเอาค่าเหล่านี้ไปประมวลผลตามขั้นตอนต่างๆ

ภายในตัวกรอง ถ้าพิจารณาที่สัมประสิทธิ์ 0.81 ซึ่งสามารถเขียนแทนด้วยเลขฐานสอง ได้ดังนี้ คือ 0.110011110101

$$0.81 = (1/2)^{1}+(1/2)^{2}+(1/2)^{5}+(1/2)^{6}+...$$

จะเห็นได้ว่าจะต้องใช้ตัวเลขฐานสองเป็นจำนวนอนันต์เพื่อที่จะสามารถแทนค่าที่แท้จริงของค่าสัมประสิทธิ์ 0.81 นี้ แต่เนื่องจากอุปกรณ์ต่างๆที่นำมาประกอบขึ้นเป็นตัวกรอง สามารถประมวลผลด้วยตัวเลขฐานสองที่มีจำนวนบิทจำกัด ดังนั้นตัวเลขฐานสองจากเดิมที่ใช้แทนด้วยพจน์เป็นอนันต์ ต้องเปลี่ยนแปลงให้อยู่ในรูปที่แทนได้ด้วยตัวเลขฐานสองด้วยจำนวนบิทที่จำกัดที่สอดคล้องกับอุปกรณ์ที่นำมาประกอบเป็นตัวกรองดังกล่าว

สมมติว่าอุปกรณ์ต่างๆที่นำมาประกอบขึ้นเป็นโครงสร้างของตัวกรอง เป็นโครงสร้างขนาด 6 บิท ดังนั้นค่าสัมประสิทธิ์ 0.81 สามารถประมาณแทนได้ด้วยเลขฐานสอง คือ 0.11001 (เลือกเอา 6 บิทที่มีนัยสำคัญสูงสุด) เมื่อนำเอาค่า 0.11001 มาพิจารณาจะเห็นได้ว่าค่านี้แทนด้วยจำนวน 0.78125 ในระบบเลขฐานสิบ ดังนั้นสัมประสิทธิ์ของตัวกรองที่นำมาสร้างจะใช้ค่า 0.78125 แทนค่าสัมประสิทธิ์จริงๆของตัวกรองคือ 0.81จะเห็นได้ว่าจะเกิดค่าความผิดพลาดจากสัมประสิทธิ์จริงๆอยู่ 0.2875 ในทำนองเดียวกับค่าสัมประสิทธิ์ 1.2727922 สามารถแทนด้วยเลขฐานสองจำนวนพจน์อนันต์ได้คือ 1.010001011101 เมื่อทำการประมาณให้เหลือจำนวนบิท 6 บิท จะได้ 1.01000 หรือ 1.25 ซึ่งเกิดค่าผิดพลาดจากค่าสัมประสิทธิ์จริงๆอยู่ 0.0227922 ดังนั้นสมการที่ (2) เมื่อนำไปสร้างเป็นโครงสร้างของตัวกรองจะถูกแทนด้วยสมการที่ (3)

$$y_n = x_n - x_{n-2} + 1.25y_{n-1} - 0.78125y_{n-2} \quad \text{-----(3)}$$

จากสมการที่ (3) ฟังก์ชันถ่ายโอนที่ได้ก็จะเปลี่ยนแปลงไปจากเดิม โดยที่ตำแหน่ง ซีโร ของตัวกรองยังอยู่ที่เดิม แต่สำหรับตำแหน่งของโพลจะเปลี่ยนไปอยู่ที่ตำแหน่ง 0.625+-j0.625

จากผลดังกล่าวที่เกิดขึ้น ทำให้ผลตอบสนองความถี่ของตัวกรอง และ ผลตอบสนองทางแอมปลิจูดของตัวกรองก็จะเปลี่ยนแปลงไปจากเดิม

ความผิดพลาดที่พบอีกชนิดหนึ่งก็ คือ ความผิดพลาดที่เกิดจากการควอนไตซ์สัญญาณอินพุทซึ่งเป็นสัญญาณเชิงอุปมาน สมมติว่าสัญญาณอินพุท คือ สัญญาณที่เป็นฟังก์ชันของไซน์ (sine) และ โคไซน์ (cosine) ภายหลังจากการควอนไตซ์สัญญาณเพื่อที่

จะทำการเข้ารหัสให้กับลำดับสัญญาณ 0.2955,0.5564,0.8912,0.9320,... เพื่อให้ค่าต่างๆเหล่านี้จะต้องถูกแทนด้วยเลขฐานสองที่มีจำนวนบิทจำกัด เพื่อที่จะนำไปเก็บไว้ในตัวเก็บข้อมูลในการที่จะนำมาประมวลผลข้อมูลที่เหมาะสม

ในที่นี้ถ้าหากเราใช้ตัวเก็บข้อมูลขนาด 8 บิท ดังนั้นข้อมูลต่างๆเหล่านี้จะต้องถูกแทนด้วยเลขฐานสองขนาด 8 บิทได้ดังต่อไปนี้

...,0.0100101,0.1000111,0.111010,0.1110111,...

ซึ่งค่าที่ถูกต้องสอดคล้องกับเลขฐานสองขนาด 8 บิท ดังกล่าวคือ

...,0.2890625,0.5546875,0.890625,0.9265,...

จะเห็นได้ว่าค่าเหล่านี้จะแตกต่างกับค่าที่ได้มาจากลำดับสัญญาณจริงๆ

สำหรับความผิดพลาดที่อาจเกิดขึ้นได้อีกอันก็คือ ความผิดพลาดที่เกิดจากการปัดเศษข้อมูล (rounding) กล่าวคือ ถ้าจะพิจารณาเจาะจงไปที่การประมวลผลทางคณิตศาสตร์ จากสมการที่ (3) จะเห็นได้ว่าการคูณระหว่างค่าสัมประสิทธิ์ซึ่งแทนด้วยเลขฐานสองขนาด 6 บิทและ ลำดับสัญญาณ y_{n-2} ซึ่งแทนด้วยเลขฐานสองขนาด 8 บิท ผลลัพธ์ที่ได้คือเลขฐานสองขนาด 14 บิท ดังนั้นจึงต้องมีการปัดเศษข้อมูลนี้ให้เหลือ 8 บิท เพื่อที่จะนำไปเก็บไว้ในตัวเก็บข้อมูลขนาด 8 บิท ได้ นั่นคือการปัดเศษนี้ จะทำให้เกิดค่าผิดพลาดขึ้นเช่นเดียวกันกับการผิดพลาดที่ผ่านมาแล้วทั้ง 2 แบบ

กล่าวโดยสรุป ในการสร้างตัวกรองเพื่อนำไปใช้งานจริง ผู้สร้างจะต้องคำนึงถึงข้อผิดพลาดต่างๆที่เกิดจากการใช้ความยาวคำจำกัดไว้เสมอ

ซึ่งสามารถสุปได้เป็นความผิดพลาด 3 ประการดังต่อไปนี้

1. ความผิดพลาดที่เกิดจากการควอนไตซ์ลำดับสัญญาณ $\{x_n\}$
2. ความผิดพลาดที่เกิดจากการแทนค่าสัมประสิทธิ์ของตัวกรอง$\{a_k\}$และ$\{b_k\}$
3. ความผิดพลาดที่เกิดจากการปัดเศษที่ตัวประมวลผลทางคณิตศาสตร์

บทแทรก2. โปรแกรมแปลงและบวกเลขส่วนเติมเต็มเป็นสอง

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
     /**                                                    **/
     /**    This  function  is Decimal to  Binary conversion  **/
     /**                                                    **/
     /**          in  2'complement Form                     **/
     /**                                                    **/

void DTB(double a[],unsigned bin[],int h)
  {
     int  x;
     int     p;

     bin[h] = 0;

     if(a[h] < 0.0 )
      {
         a[h]  = 1.0+a[h];
         bin[h]  |= 1 << 13;    /**** to keep sign of value at 13th bit of bin[h] ****/
      }
     for(p=1 ; p <= 12 ; p++)
        {
          a[h]   *= 2.0;
          x    = (int)a[h];
          bin[h] |= x << (13-p);    /**** to keep 0 or 1 at (1-12)th bit of bin[h] ****/
          if(a[h] >= 1.0 )  a[h] = a[h]-1.0;   /**** if a[h] more than 1 then minus 1 from a[h]   ****/
        }
    }
     /****                                  ****/
     /****       Plus Binary value Function  ****/
     /****                                  ****/

 void  Plus(unsigned a1,unsigned a2,unsigned total[1])
   {
     int  plus,p,x,y,carry;

     plus  = 0;
     carry = 0;
     /*printf("\n\n\n\n");*/
     for(p=1; p <= 13 ; p++)
        {
          x = (a1 >> p) & 1;
          y = (a2 >> p) & 1;
        /*  printf(" x= %d  y= %d  carry=%d ",x,y,carry); */
          x += y;
          x += carry;
          if( (x == 0) || (x == 1) )
            {
              carry = 0;
              x = x;
            }
          else if(x == 2)
```

```
            {
                carry = 1;
                    x = 0;
             }
           else if(x == 3)
             {
                carry = 1;
                    x = 1;
             }
        /*   printf("total= %d ",x);  */
           plus |= x << p;
        /*   buf  =  (plus >> p) & 1;
           printf("  plus=%d\n",buf);*/
        }
     total[1] = plus;
   }

   /****    Use to converse and plus binary value by use    ****/
   /****    Plus  and DBT function and show in monitor      ****/

void PlusBinary(int number,double a[],unsigned bin[12],unsigned total[1])
  {
     int        i,j,buf;

     printf("\n\n");
     for(j=0;j < number ; j++)
        {
          printf("   a[%d]  = ",j);
          printf(" %10f   ->   ",a[j]);
          DTB(a,bin,j);
          buf = (bin[j] >> 13 ) & 1;
          printf("  %d.",buf);
          for(i=0 ; i <= 11 ; i++)
            {
              buf = (bin[j] >> (12-i)) & 1 ;
              printf("%d",buf);
            }
         printf("\n\n");
        }

     total[1] = 0;
     for(i=0 ; i < number ; i++)
        Plus(total[1],bin[i],total);
     buf = (total[1] >> 13 ) & 1;
     printf("                          total = ");
     printf(" %d.",buf);
     for(i=0 ; i <= 11 ; i++)
        {
          buf = (total[1] >> (12-i)) & 1 ;
          printf("%d",buf);
        }
     printf("\n");
  }

void main()
  {
```

```
  unsigned  bin[12];        /*** to keep binary value   ****/
  unsigned  total[1];  /*** to keep binary value which plus together  ****/
  int       number;    /*** all value which you want to plus  ****/
  int       p;
  double    a[12];     /*** to keep value that you want to plus  ****/
  char      c[20];

  clrscr();

  printf("\n\n  The number of value :  ");
  scanf("%s",c);
  number = atoi(c);
  printf("\n\n");
  for(p=0 ; p < number ; p++)
     {
       printf("  a[%d] =   ",p);
       scanf("%s",c);
       a[p] = atof(c);
     }

  PlusBinary(number,a,bin,total);
  getch();
}
```

เอกสารอ้างอิง

1. Abraham Peled and Bede Liu ; "A New Hardware Realization of Digital Filter",IEEE Trans. Acoust.,Speech and Signal Processing,Vol. ASSP-22,1972
2. Abraham Peled and Bede Liu ; "Digital Signal Processing" : John Wiley & Sons,1976
3. J.L. Schmalzel , D.N. Heinand,N. Ahmed; "Some Pedagogical Considerations of Digital Hardware Implementation", IEEE Circuit and System Magazine, Vol.2 No.1,1980
4. Fred J. Taylor;" An Analysis of Distributed Arithmetic Digital Filter ",IEEE Trans. Acoust.,Speech and Signal Processing ,Vol. ASSP-34,1986
5. Curtis F. Gerald, Patrick O. Wheatley ;"Applied Numerical Analysis ":Addison-Wesley ,1987
6. Lawrence R. Rabiner and Bernard Gold;"Theory and Application of Digital Signal Processing" : Prentice Hall ,1988
7. วัลลภ สุระกำพลธร ; "การประมวลผลสัญญาณเชิงตัวเลข"กรุงเทพ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ,2533
