การสร้างวงจรกรองสัญญาณเชิงเลข 2 มิติโดย DISTRIBUTED ARITHMETICS

IMPLEMENTATION OF 2-D DIGITAL FILTER-BASED DISTRIBUTED ARITHMETICS

นายสมจินต์ ทองปลิว

MR. SOMCHIN THONGPLEW

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2538

ISBN 974-621-257-5

IMPLEMENTATION OF 2-D DIGITAL FILTER-BASED DISTRIBUTED ARITHMETICS

SOMCHIN THONGPLEW

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE
MASTER OF ELECTRICAL ENGINEERING
GRADUATE SCHOOL
KING MONGKUT 'S INSTITUTE OF TECHNOLOGY LADKRABANG
1995

ISBN 974-621-257-5

หัวข้อวิทยานิพนธ์	การสร้างวงจรกรองสัญญาณเชิงเลข 2 มิติโดย Distributed Arithmetics
นักศึกษา	นายสมจินต์ ทองปลิว
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.กอบชัย เดชหาญ
ระดับการศึกษา	วิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมโทรคมนาคม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.	2538

บทคัดย่อ

วิทยานิพนธ์นี้จะเกี่ยวกับวงจรกรองสัญญาณเชิงเลข (digital filter) ซึ่งเป็นแขนงหนึ่งของการประมวลสัญญาณเชิงเลข (digital signal processing) โดยปกติแล้วจะเป็นวิชาที่เกี่ยวข้องกับคณิตศาสตร์ในรูปของทฤษฎี และอัลกอริทึมต่าง ๆ แต่บทความนี้จะแสดงให้เห็นวิธีการนำเอาฮาร์ดแวร์เข้าไปประยุกต์ใช้กับงานการกรองสัญญาณเชิงเลขสองมิติแบบ Low-pass filtering , High-pass filtering , Sobel edge detector และ Laplacian filtering โดยใช้วิธีการหนึ่งที่เรียกว่าการกระจายทางคณิตศาสตร์ (Distributed Arithmetics , DA) ซึ่งเป็นวิธีที่ทำให้ฟังก์ชันถ่ายโอน (transfer function) ของวงจรกรองสัญญาณเชิงเลขแปลงไปสู่วงจรอิเล็กทรอนิกส์ได้ในรูปของการเปิดตาราง อีกทั้งยังแก้ไขข้อมูลที่ผิดพลาดจากการประมวลสัญญาณของ DA แบบเก่า วิธีการนี้จะช่วยให้เห็นรูปแบบทางฮาร์ดแวร์ที่ประมวลข้อมูลด้วยความเร็วสูง และถูกต้องมากขึ้น

Thesis Title	Implementation of 2-D digital filter-based Distributed Arithmetics
Student	Mr.Somchin Thongplew
Thesis Advisor	Assoc. Prof. Dr. Kobchai Dejhan
Level of Study	Master of Electrical Engineering
Department	Communication Engineering King Mongkut 's Institute of Technology Ladkrabang
Year	1995

ABSTRACT

This thesis describes the digital filtering, it is a part of Digital Signal Processing (DSP) section.Generally, DSP is presented by arithmetics and algorithms. This paper proposes an implementation of two dimensional low pass filter, high pass fillter,Sobel edge detector and Laplacian filter by using the method so called distributed arithmetics with this method the transfer function is transformed into a look up table. Not only the ease in implementation, the proposed method has also included the error-checking procedure. The implementation with conventional LSI chips shows its functionality perfectly.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี ผู้ทำวิจัยขอกราบขอบพระคุณ คุณพ่อบัญญัติ ทองปลิว และคุณแม่มาลี ทองปลิว ที่ให้ความอุปการะในทุก ๆ ด้าน

ขอกราบขอบพระคุณท่านอาจารย์ รศ.ดร.กอบชัย เดชหาญ ที่ให้คำแนะนำตลอดจนช่วยเหลือด้านเอกสารข้อมูลอันเป็นประโยชน์ต่องานวิจัย ขอขอบคุณ คุณอาโมทย์ สมบูรณ์แก้ว ที่ช่วยเหลือทางด้านข้อมูลภาพถ่ายและแนะนำเทคนิคการเขียนโปรแกรม

นายสมจินต์ ทองปลิว

สารบัญ

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

หน้า

บทที่ 1
ทฤษฎีเบื้องต้น

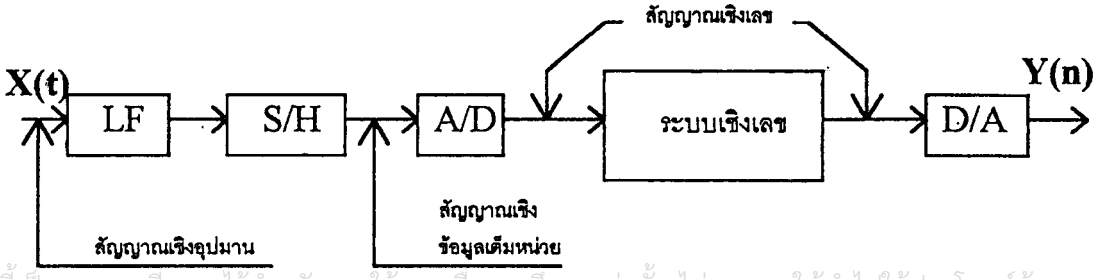
1.1 บทนำ

การประมวลผลสัญญาณที่เรียกว่า ดิจิตอลซิกแนลโปรเซสซิ่ง (Digital Signal Processing)[11] หรือใช้ตัวย่อว่า DSP เป็นขบวนการที่ใช้ความรู้ทางคณิตศาสตร์เกี่ยวกับตัวเลขมาทำการจัดการกับสัญญาณต่าง ๆ โดยปกติการจัดการสัญญาณที่เป็นอะนาลอกเป็นทั้งขบวนการแบบเชิงเส้น และไม่เชิงเส้น มีอุปกรณ์ช่วยในการประมวลผลสัญญาณอยู่มาก เช่น ตัวต้านทาน ตัวเก็บประจุ ทรานซิสเตอร์ ออปแอมป์ และอุปกรณ์อิเลคทรอนิคส์อีกมากมาย ฯลฯ การสร้างและประมวลสัญญาณเหล่านั้น สามารถประมวลให้ได้ผลลัพธ์ตามต้องการด้วยขบวนการทางดิจิตอล และนี่เป็นเหตุผลที่ทำให้บทบาทของ DSP จะเพิ่มมากขึ้นเป็นลำดับ จนสามารถนำมาประยุกต์ใช้งานได้อย่างกว้างขวางต่อไป ทั้งนี้เพราะอุปกรณ์ที่เกี่ยวกับ DSP เริ่มมีราคาถูกลงและทำงานได้รวดเร็วยิ่งขึ้น การประยุกต์ใช้งานที่พบเห็นได้มากได้แก่ เรื่องต่าง ๆ ต่อไปนี้

- ฟิลเตอร์ อะแดปตีฟฟิลเตอร์ อีควอไลเซอร์ และตัวเลื่อนเฟส
- มิกเซอร์ มอดูเลเตอร์ และตัวเปรียบเทียบเฟส
- วงจรกำเนิดสัญญาณ ออสซิเลเตอร์ปรับค่าได้ และแหล่งสัญญาณรบกวน
- ฟาสฟูเรียร์ทรานส์ฟอร์ม (Fast Fourier Transform)
- อุปกรณ์ไม่เชิงเส้น ลิมิตเตอร์ คอมพาราเตอร์
- การควบคุม วงจรควบคุมแบบต่าง ๆ เซอร์โว ฯลฯ
- วงจรประมวลผลสัญญาณภาพ เสียงพูด ฯลฯ

การประมวลผลสัญญาณในรูปแบบ DSP จะใช้ไมโครโปรเซสเซอร์ที่มีความเร็วสูง ดังนั้นจึงประยุกต์ใช้งานได้ง่าย และเป็นการออกแบบระบบด้วยซอล์ฟแวร์ ดังนั้นจึงมีการนำไปใช้ในงานด้านต่าง ๆ ที่เป็นผลิตภัณฑ์สำเร็จรูปหลายอย่าง เช่น โมเดม อุปกรณ์โทรศัพท์ วงจรสังเคราะห์เสียง วงจรควบคุมขบวนการผลิต ฯลฯ การประมวลผลสัญญาณอะนาลอกจึงประกอบด้วยขั้นตอนต่าง ๆ ดังนี้

1. สุ่มสัญญาณอะนาลอกที่เป็นอินพุท ออกเป็นสัญญาณอะนาลอกแบบเป็นช่วง ๆ
2. เปลี่ยนสัญญาณอะนาลอกเป็นสัญญาณดิจิตอล
3. ประมวลข้อมูลดิจิตอลนั้นตามอัลกอริทึมทางคณิตศาสตร์
4. แปลงผลลัพธ์จากดิจิตอลกลับเป็นอะนาลอกใหม่



รูปที่ 1.1 ระบบการประมวลสัญญาณเชิงอุปมานโดยใช้ระบบประมวลสัญญาณเชิงเลข

กล่าวโดยทั่วไปการประมวลผลสัญญาณก็คือการนำเอาสัญญาณมาเข้าระบบการประมวลผลที่ข้างในประกอบขึ้นด้วยระบบอุปกรณ์ที่ทำปฏิบัติการทางคณิตศาสตร์ อาทิเช่น การบวกสัญญาณ การคูณ การหาร ถอดรากที่สอง หรือ การอินทิเกรตสัญญาณ ซึ่งอาจเป็นปฏิบัติการอย่างใดอย่างหนึ่ง หรือผสมผสานกันแล้วแต่ความยุ่งยากของระบบแต่เดิมนั้นเรานิยมใช้ระบบการประมวลสัญญาณในรูปแบบระบบการประมวลผลสัญญาณเชิงอุปมาน ซึ่งสัญญาณเข้า และ ออก จากระบบเป็นสัญญาณต่อเนื่องหรือเชิงอุปมาน ส่วนปฏิบัติการทางคณิตศาสตร์ทำโดยการใช้วงจรเชิงเส้นที่อาจต่อโดยใช้กลอุปกรณ์เอ็กทีฟว์ (เช่น ทรานซิสเตอร์ หรือ ออปแอมป์) และ กลอุปกรณ์พาสซีฟว์ (เช่น ตัวความต้านทาน ตัวเก็บประจุ หรือ ตัวอินดักเตอร์) ข้อดีของระบบการประมวลผลแบบนี้ก็คือ ราคาถูก การออกแบบทำได้ง่าย เนื่องจากมีการค้นคิดวิธีการ และทำบันทึกหรือตารางไว้มากมาย อย่างไรก็ตามจุดจำกัดของระบบการประมวลผลสัญญาณเชิงอุปมานอยู่ที่ ประสิทธิภาพ และ ความแม่นยำ ในการประมวลผลที่ มีความละเอียดมากได้ นอกจากนี้คุณสมบัติของกลอุปกรณ์ก็ยังแปรค่าตามสภาวะแวดล้อม เช่น อุณหภูมิ อายุการใช้งาน ความชื้นทำให้ความเชื่อถือได้ของระบบมีได้แค่ระดับหนึ่งเท่านั้น และยังมีสิ่งรบกวนมากด้วย ผลจากพัฒนาการทางเทคโนโลยีการออกแบบ และผลิตวงจรรวม ทำให้กลอุปกรณ์เชิงเลข เช่น คอมพิวเตอร์ ไมโครคอมพิวเตอร์ ไมโครโปรเซสเซอร์ อุปกรณ์ลอจิกเกต หรือ อุปกรณ์สนับสนุนต่าง ๆ มีราคาถูกลงเป็นอย่างมาก อีกทั้งประสิทธิภาพดีขึ้นมาก ทำให้ความสนใจในการนำเอาระบบประมวลผลเชิงเลขมาทำการประมวลผลสัญญาณมากขึ้น และเริ่มนำเข้ามาทดแทนระบบการประมวลผลเชิงอุปมานเดิมมากขึ้นเป็นลำดับ ผลที่ได้ก็คือ ได้ระบบการประมวลผลสัญญาณที่มีประสิทธิภาพ และความเชื่อถือได้สูง และราคาไม่แพงอีกต่อไป อย่างไรก็ตามถึงแม้แนวคิดในการประมวลผลสัญญาณก็ยังเป็นการนำเอาสัญญาณมาทำปฏิบัติการทางคณิตศาสตร์ แต่เนื่องจากสัญญาณเข้าระบบเป็นแบบเชิงเลขที่ได้จากการสุ่มตัวอย่างสัญญาณเดิมแล้วทำการแปลงเป็นตัวเลข การทำอย่างนี้ทำให้ธรรมชาติของสเปกตรัมของสัญญาณเปลี่ยนไป การวิเคราะห์และออกแบบจึงไม่สะดวกที่จะใช้ชุดเครื่องมือทางคณิตศาสตร์ของระบบเชิงอุปมานเดิม เช่น การแปลงลาปลาซ หรือ สมการดิฟเฟอเรนเชียลได้ ดังนั้นการเรียนรู้ถึงการวิเคราะห์การออกแบบระบบการประมวลผลสัญญาณเชิงเลข เราจึงต้องทำความเข้าใจกับเครื่องมือทางคณิตศาสตร์ที่ใช้ในสาขาวิชานี้ เช่น การแปลง-แซด และสมการผลต่างสืบเนื่อง เป็นต้น ซึ่งสิ่งเหล่านี้มีพัฒนาการมาจากระบบเชิงอุปมานนั่นเอง

โดยทั่วไปอาจกล่าวได้ว่างานวิจัยออกแบบ และพัฒนาในสาขาวิชาการประมวลผลสัญญาณเชิงเลข (Digital Signal Processing : DSP) อาจแบ่งได้เป็น 2 หัวข้อใหญ่ คือ

1. การกรองเชิงเลข (Digital Filtering): ความจริงแล้ว การกรองเชิงเลขอาจมีอยู่หลายประเภทด้วยกัน อย่างไรก็ตามสำหรับในงานวิจัยนี้เราสนใจเฉพาะการกรองเชิงเลขแบบเป็นระบบเชิงเส้น ซึ่งมีตัวกรองอยู่ 2 ชนิด คือ ตัวกรองไม่ป้อนกลับเชิงเลข (non-recursive filter) และ ตัวกรองป้อนกลับเชิงเลข (recursive digital filter) ตัวกรองทั้ง 2 ชนิดเป็นวงจรพื้นฐานของวงจรกรองเชิงเลขแบบอื่น ๆ รายละเอียดเกี่ยวกับคุณสมบัติ

หลักการออกแบบ หลักการสร้างจะได้กล่าวถึงในบทต่อ ๆ ไป

2. การแปลงเชิงเลข (Digital Transform): การแปลงเชิงเลขโดยเฉพาะ การแปลงฟูริเยอร์เต็มหน่วย (Discrete Fourier Transform : DFT) ที่ทำการประมวลผลโดยใช้ขั้นตอนวิธีที่เรียกว่า การแปลงฟาสท์ฟูริเยอร์ (Fast Fourier Transform : FFT) มีส่วนทำให้ การประมวลผลสัญญาณเชิงเลขได้รับความนิยมมากขึ้น

ตัวอย่างการนำไปประยุกต์ใช้ของการประมวลผลสัญญาณเชิงเลขมีให้เห็นอยู่มากในปัจจุบัน เช่น เทคนิคการสื่อสารเชิงเลขซึ่งอาจเป็น การผสมสัญญาณ (modulatoin) หรือ การเข้ารหัส (coding) สัญญาณเชิงเลข นอกจากนี้ก็ยังใช้ในระบบควบคุมเชิงเลข (digital control system) และอุปกรณ์เครื่องมือวัดเชิงเลข (digital Instrumentation) เป็นต้น โดยทั่วไปแล้วการประมวลผลสัญญาณเชิงเลขอาจ หมายถึง

(ก) การประมวลผลเชิงเลขของสัญญาณใด ๆ ที่ไม่ได้อยู่ในรูปสัญญาณเชิงเลข หรือ

(ข) การประมวลผล (ด้วยวิธีใดก็ตาม) ของสัญญาณที่อยู่ในรูปแบบสัญญาณเชิงเลขแล้ว

ความหมายในข้อ (ก) นั้น หมายถึง การประมวลผลสัญญาณในธรรมชาติทั่วไป ซึ่งสัญญาณในธรรมชาตินั้นปกติอยู่ในรูปแบบ สัญญาณเชิงอุปมาน (analog signal) ก่อนการประมวลผลต้องทำการแปลงสัญญาณเชิงอุปมานให้เป็นสัญญาณเชิงเลขเสียก่อน ส่วนความหมายในข้อ (ข) เป็นการประมวลผลสำหรับสัญญาณที่มาจากอุปกรณ์เชิงเลข เช่น คอมพิวเตอร์ หรือ ระบบควบคุมเชิงเลข สัญญาณจากอุปกรณ์เหล่านี้เป็นแบบสัญญาณเชิงเลขอยู่แล้วจึงไม่จำเป็นต้องมีการแปลงสัญญาณอีก

เหตุผลสำคัญที่ทำให้การประมวลผลสัญญาณเชิงเลข เริ่มได้รับความสนใจมากขึ้นก็เนื่องมาจากเทคโนโลยีในการสร้างวงจรรวม หรือ วงจรไอซี (integrated circuit) สามารถทำได้อย่างมีประสิทธิภาพดีขึ้นมาก วงจรยุ่งยาก และ มีขนาดใหญ่สามารถนำมาสร้างเป็นวงจรรวมที่ผลิตบนสารกึ่งตัวนำชิ้นเดียวกันได้ และสามารถผลิตครั้งละจำนวนมากขึ้นได้ จึงทำให้ราคาต่อวงจรถูกเป็นอย่างมาก โดยเฉพาะอย่างยิ่งวงจรรวมแบบวีแอลเอสไอ (VLSI : Very Large Scale Integration) สำคัญ ๆ เช่น ไมโครโปรเซสเซอร์ (microprocessor) ซึ่งมีราคาถูกลงมากและมีผู้สนใจนำเอามาทำเป็นตัวประมวลผลสัญญาณมากขึ้นนอกจากนี้ การประมวลผลเชิงเลขยังให้ความเชื่อถือได้ (reliability) และความแม่นยำในการคำนวณ (accuracy) ดีกว่าการประมวลผลสัญญาณเชิงอุปมานมาก

เนื่องจากการประมวลผลสัญญาณเชิงอุปมานได้มีการศึกษาวิจัย และมีการพัฒนากันมานาน ทำให้อุปกรณ์ และระบบวงจรที่ใช้กันอยู่ในปัจจุบันอยู่ในรูปแบบของการประมวลผลสัญญาณเชิงอุปมานเกือบทั้งหมด การที่จะออกแบบสร้างระบบประมวลผลสัญญาณใหม่ ให้อยู่ในรูปแบบของการประมวลผลสัญญาณเชิงเลข อุปกรณ์ใหม่นั้นก็ต้องสามารถเชื่อมโยงกับอุปกรณ์เดิมซึ่งเป็นระบบการประมวลผลเชิงอุปมานได้ ดังนั้นความพยายามของนักวิจัยสมัยแรก ก็คือ การวิจัย และ ศึกษาเพื่อหาวิธีที่นำเอาความสัมพันธ์นี้ได้แก่ สาขา วิชาทฤษฎีโครงข่าย (network theory) กับ ทฤษฎีกราฟ (graph theory) สาขาวิชาการทำนายข้อมูล (data prediction) กับ การอัดแถบความถี่ปฏิบัติงาน (bandwidth compression) สาขาวิชาการแปลงฟูริเยอร์ (Fourier transform) กับ การแปลงอาดามาด์ (Hadamard transform) หรือ สาขาวิชา

การกรองเชิงเลข (digital filter) กับ การกรองเชิงอุปมาน (analog filter) เป็นต้น ซึ่งในแต่ละสาขาวิชาที่กล่าวถึงนี้เป็นที่รู้จักกันเป็นอย่างดี และ มีการประยุกต์ใช้กันมากในปัจจุบัน

1.2 ข้อได้เปรียบและข้อด้อยของการประมวลสัญญาณเชิงเลข

อาจกล่าวได้ว่า ข้อเด่นเป็นอย่างมากของการประมวลผลสัญญาณเชิงเลขก็คือ ความเที่ยงตรงความแม่นยำในการประมวลผล อย่างไรก็ตามการนำเอาระบบประมวลผลสัญญาณเชิงเลขมาใช้ร่วมกับระบบการประมวลผลสัญญาณเชิงอุปมานที่ยังมีการใช้กันมากในปัจจุบัน ก็ต้องมีข้อเสียเปรียบอยู่มากเช่นกัน ดังนั้นการกล่าวถึงข้อได้เปรียบและข้อด้อย บางประการของการประมวลผลสัญญาณเชิงเลขก็อาจเป็นประโยชน์บ้างและนอกจากนี้ยังเป็นการให้ภาพอย่างกว้าง ๆ แก่ผู้ที่จะนำเอาระบบประมวลผลสัญญาณเชิงเลขไปประยุกต์ใช้ได้อย่างเหมาะสมยิ่งขึ้น

- **ข้อดีของการประมวลสัญญาณเชิงเลข**

ข้อได้เปรียบของการประมวลผลสัญญาณเชิงเลขอาจกล่าวได้เป็นหัวข้อดังนี้

1.เหมาะสำหรับอุปกรณ์ที่ข้อมูลอยู่ในรูปแบบสัญญาณเชิงเลขอยู่แล้วเช่น ผลลัพธ์จากคอมพิวเตอร์ ไมโครโปรเซสเซอร์ หรือ ข้อมูลจากระบบควบคุมเชิงเลข เป็นต้น ทั้งนี้เนื่องจากสัญญาณหรือข้อมูลจากอุปกรณ์เหล่านี้ถ้าหากต้องนำไปประมวลผลในระบบประมวลผลสัญญาณเชิงอุปมานก็จำเป็นต้องมีวงจรแปลงรูปแบบสัญญาณเชิงเลขเป็นเชิงอุปมาน (Digital to Analog converter : D/A) เพื่อแปลงสัญญาณเชิงเลขให้เป็นเชิงอุปมานก่อน หลังจากประมวลผลด้วยระบบประมวลผลเชิงอุปมานแล้วก็ต้องแปลงผลลัพธ์ที่เป็นสัญญาณเชิงอุปมานไปเป็นสัญญาณเชิงเลขด้วย วงจรแปลงรูปแบบสัญญาณเชิงอุปมานเป็นเชิงเลข (Analog to Digital converter : A/D) เพื่อส่งกลับไปยังอุปกรณ์เชิงเลขต่อไปซึ่งจะเห็นว่าระบบประมวลผลมีความซับซ้อนมากขึ้น นอกจากนี้ถ้าหากเป็นระบบที่ต้องการทำการประมวลผลสัญญาณพร้อมกันหลายสัญญาณ (ซึ่งในคอมพิวเตอร์มักทำในลักษณะนี้) ก็จะทำให้ระบบยุ่งยากมากขึ้น อีกทั้งในปัจจุบัน A/D หรือ D/A ที่มีประสิทธิภาพสูงและมีความเร็วในการแปลงสูงมีราคาแพง

2. อุปกรณ์ทางด้านเชิงเลขหรือ ดิจิตอล มีราคาถูก ขนาดเล็ก มีประสิทธิภาพสูง ความแม่นยำและความแน่นอนสูง นอกจากนี้การพัฒนาให้มีประสิทธิภาพสูงขึ้นก็เป็นไปอย่างรวดเร็ว ข้อนี้เป็นข้อได้เปรียบที่สำคัญ เพราะว่าการประมวลผลสัญญาณเชิงเลขสามารถคำนวณได้อย่างแม่นยำ และต้องการให้มีความละเอียดเท่าใดก็ได้ แต่สำหรับการประมวลผลเชิงอุปมาน การคำนวณที่ให้ความละเอียดเกินกว่าหนึ่งในพันส่วนหรือ 0.001 นั้นทำได้ยากมาก นอกจากนี้อุปกรณ์ (device) ของการประมวลผลเชิงอุปมาน เช่น ตัวความต้านทาน ตัวขยายสัญญาณ ยังมีคุณสมบัติการแปรค่าไปตามสภาวะแวดล้อม เช่น อุณหภูมิ ความชื้น อายุการใช้งานด้วย ทำให้ความแม่นยำ และความเชื่อถือได้ของระบบการประมวลผลสัญญาณเชิงอุปมานต่ำ อย่างไรก็ตามในปัจจุบัน การประมวลผลเชิงเลขที่ให้ความแม่นยำสูงต้องใช้อุปกรณ์ราคาแพง ถ้าหากมีการพัฒนาให้อุปกรณ์ประมวลผลเชิงเลขมีราคาถูกก็จะเป็นการผลดีอย่างยิ่ง

3. การรับและส่งข้อมูลหรือสัญญาณเชิงเลขทำได้แน่นอนกว่า ทั้งนี้เนื่องจากสัญญาณเชิงเลขมีแค่ 2 ระดับ คือ ศูนย์ (0) กับหนึ่ง (1) เท่านั้น ถ้าหากรูปคลื่นสัญญาณผิดเพี้ยนไปก็สามารถแก้ไข และสร้างขึ้นมาใหม่ให้เหมือนเดิมได้โดยง่าย

4. การประมวลผลสัญญาณเชิงเลขทำได้ง่าย ทั้งนี้เนื่องจากขั้นตอนวิธี (algorithm) การประมวลผลสัญญาณมักประกอบด้วย การบวก การลบ การคูณ การหาร และการเลื่อนตัวเลขเท่านั้น ซึ่งข้อความนี้สามารถเห็นจริงได้ในบทต่อ ๆ ไป

5. ระบบประมวลผลสัญญาณเชิงเลข สามารถทำเป็นแบบระบบแบ่งส่วนเวลา (time-shared system) ได้ ดังนั้นจึงสามารถทำการประมวลผลได้พร้อม ๆ กันหลายช่องสัญญาณได้ และนอกจากนี้ในระบบเดียวกันยังสามารถโปรแกรมให้ทำงานได้หลายรูปแบบด้วย

6. ระบบประมวลผลสัญญาณเชิงเลขมีความคล่องตัวสูง ทั้งนี้เนื่องจากสามารถทำการ มัลติเพล็กกับข้อมูลหรือ สัญญาณเสียงหรือ สัญญาณภาพได้ การมัลติเพล็กยังสามารถทำเป็นแบบ การมัลติเพล็กแบบแบ่งเวลา (Time Division Multiplex : TDM) หรือ การมัลติเพล็กแบบแบ่งรหัส (Code Division Multiplex : CDM) ได้

- **ข้อด้อยของการประมวลผลเชิงเลข**

นอกจากข้อดีที่กล่าวมา การประมวลผลสัญญาณเชิงเลขทำให้เกิดข้อด้อยหลายประการคือ

1. ระบบการประมวลสัญญาณเชิงเลขต้องมีสัญญาณสำหรับ การซิงค์ (synchronization) การจัดเวลา (timing) และ การกำหนดขอบ (framing) ของข้อมูลไว้ด้วย ถ้าหากสัญญาณเหล่านี้สูญหายหรือผิดพลาดไป การทำงานของระบบก็จะผิดพลาดไปด้วย

2. มีปัญหาการเชื่อมต่อ (interfacing) กับระบบการประมวลสัญญาณเชิงอุปมาน ทั้งนี้เนื่องมาจากระบบประมวลสัญญาณแบบเดิม ส่วนมากอยู่ในรูปแบบระบบเชิงอุปมาน ดังนั้นระบบที่สร้างขึ้นใหม่ก็ต้องสามารถเชื่อมต่อกับระบบเดิมให้ได้ ซึ่งอาจทำให้ระบบประมวลสัญญาณซับซ้อนขึ้น

3. สัญญาณหรือข้อมูลเชิงเลขไม่ใช่สัญญาณตามธรรมชาติที่แท้จริงแต่ถูกสร้างขึ้นมา ดังนั้นการส่งสัญญาณชนิดนี้ไปในตัวกลางตามธรรมชาติทั่วไป ซึ่งมักมีแถบความถี่ปฏิบัติงาน (bandwidth) จำกัด และยังมีผลตอบสนองเฟสไม่เป็นเชิงเส้น (non-linear phase response) จึงอาจทำให้สัญญาณเกิดความผิดเพี้ยนได้ เช่น เมื่อเราส่งสัญญาณรูปเหลี่ยมไปบนสายส่ง สัญญาณที่รับได้ไม่เป็นรูปเหลี่ยมเหมือนเดิม

4. เนื่องจากเราต้องการออกแบบระบบประมวลผลสัญญาณเชิงเลขให้มีความคล่องตัวสูง สามารถใช้งานได้หลายรูปแบบ ดังนั้นอาจทำให้การออกแบบระบบประมวลสัญญาณเชิงเลขมีความซับซ้อนมาก ซึ่งรวมทั้งด้านการซ่อมแซม บำรุงรักษา และจัดทำคู่มือใช้งานด้วย

5. ช่วงกว้างความถี่ปฏิบัติงานของระบบประมวลผลสัญญาณเชิงเลขต่ำกว่าของระบบประมวลผลเชิงอุปมานมาก ข้อจำกัดนี้เนื่องมาจากอุปกรณ์ที่ใช้หรือประกอบขึ้นเป็นระบบการประมวลผล

สัญญาณเชิงเลข เช่น วงจรเกท วงจรซีพรีจีสเตอร์ วงจรสุ่มและคงค่าสัญญาณ (Sampling and Hold circuit หรือ S/H) หรือ วงจร A/D และ D/A เป็นต้น วงจรเหล่านี้ต่างมีความเร็วสูงสุดในการทำงานจำกัดอยู่ค่าหนึ่ง ซึ่งในปัจจุบันยังมีค่าต่ำมาก จึงเป็นผลทำให้ระบบการประมวลผลเชิงเลขมีความเร็วต่ำ ตัวอย่างเช่น ถ้าใช้ไมโครโปรเซสเซอร์ที่มีสัญญาณนาฬิกาขนาด 1 เมกกะเฮิร์ท เมื่อนำไปสร้างเป็นระบบประมวลผลสัญญาณเชิงเลขจะใช้ได้กับสัญญาณที่มีความถี่สูงสุดประมาณ 10 ถึง 20 กิโลเฮิร์ท เท่านั้น อย่างไรก็ตาม เมื่อพิจารณาข้อดีและข้อด้อยต่าง ๆ นี้แล้ว ถ้าเราต้องการระบบประมวลสัญญาณที่มีประสิทธิภาพดี ความแม่นยำสูงโดยไม่คำนึงถึงว่าราคาจะสูงเท่าใดแล้ว ระบบการประมวลผลสัญญาณเชิงเลขก็ให้ประโยชน์ได้มาก

1.3 บทนิยามของคำศัพท์เทคนิคในระบบการประมวลผลเชิงเลข

ในการกล่าวถึงระบบ DSP ในที่นี้จะเกี่ยวข้องกับคำนิยามของคำศัพท์ทางอิเล็คทรอนิคส์ซึ่งในที่นี้ขอนิยามคำศัพท์ที่เกี่ยวข้องก่อน

1. สัญญาณเชิงอุปมาน (Analog signal) : ใช้กับการกล่าวถึงสัญญาณที่มีรูปคลื่น (waveform) แปรไปอย่างต่อเนื่องกับพิสัยเวลา โดยที่แอมพลิจูด (amplitude) หรือค่าขนาดของสัญญาณก็มีการแปรไปอย่างต่อเนื่องด้วย เช่น สัญญาณไซน์ หรือ $\sin(\omega t)$ เป็นต้น คำว่าอุปมานมีต้นกำเนิดมาจากการพัฒนาการของคอมพิวเตอร์เชิงอุปมาน (analog computer) ที่เลียนแบบปริมาณทางฟิสิกส์ต่าง ๆ (ซึ่งเป็นปริมาณแบบต่อเนื่องกับเวลา) ด้วยค่าขนาดศักย์หรือกระแสไฟฟ้า และใช้อุปกรณ์เชิงอุปมาน เช่น วงจรขยายออปแอมป์ (op-amp) มาใช้ช่วยในการเลียนแบบ

2. สัญญาณเชิงเวลาต่อเนื่อง (Continuous-time signal) : สัญญาณแบบนี้รูปคลื่นของสัญญาณแปรค่าไปอย่างต่อเนื่องกับพิสัยเวลา แต่แอมพลิจูดไม่ได้เจาะจงว่าต้องแปรไปอย่างต่อเนื่อง ซึ่งก็หมายถึงว่าอาจแปรอย่างไม่ต่อเนื่องก็ได้ เพราะฉะนั้นอาจกล่าวได้ว่า สัญญาณเชิงอุปมานเป็นชนิดหนึ่งของสัญญาณเชิงเวลาต่อเนื่องได้

3. สัญญาณเชิงเลขเวลาเต็มหน่วย (Discrete-time signal) : เป็นสัญญาณ $x(t)$ ที่ค่าของฟังก์ชันกำหนดเฉพาะเซตของเวลาที่แน่นอนอันหนึ่งเท่านั้นสัญญาณแบบนี้อาจแบ่งตามลักษณะของแอมพลิจูดได้เป็น 2 แบบ คือ

- สัญญาณเชิงข้อมูลเต็มหน่วย (Discrete data signal) : สัญญาณแบบนี้แอมพลิจูดจะต่อเนื่อง หรือกล่าวได้ว่า แอมพลิจูดมีค่าเท่ากันทุกประการกับสัญญาณเชิงอุปมาน ที่เป็นตัวต้นแบบในการสุ่มตัวอย่าง (sampling) ตัวอย่างเช่น สัญญาณออกของวงจรสุ่มและคงค่าสัญญาณจะจัดเป็นสัญญาณเชิงเต็มหน่วย สัญญาณแบบนี้บางที่อาจเรียกว่า ข้อมูล หรือ สัญญาณเชิงเต็มหน่วย

- สัญญาณเชิงเลข (Digital signal) : สัญญาณแบบนี้แอมพลิจูดของสัญญาณมีค่าเฉพาะเซตของค่าที่แน่นอนเซตหนึ่งเท่านั้น เช่น สัญญาณที่ออกจากวงจร A/D เป็นต้น

ในระบบประมวลผลสัญญาณถ้าหากเรามีสัญญาณเชิงอุปมาน x(t) แล้วเราทำการสุ่มตัวอย่างทุกคาบเวลา T วินาที ด้วยวงจร S/H เพราะฉะนั้น x(t) ที่ได้จากการสุ่มตัวอย่าง หรือ มีการกำหนดค่า x(nT) เฉพาะเวลาที่ t = nT โดยที่ n เป็นค่าคงตัวใด ๆ สัญญาณที่จุดสัญญาณออกของวงจร S/H นี้เรียกว่าเป็นสัญญาณเชิงเต็มหน่วย เพราะว่าแอมพลิจูดยังเท่ากับสัญญาณ x(t) เดิม แต่ถ้าผ่านสัญญาณ x(nT) ไปยัง วงจร A/D ขนาด N บิท แล้วสัญญาณจะถูกแปลง หรือตีความเป็นเลขฐานสองที่มี 2^n ค่า โดยใช้ค่าที่ใกล้เคียงกับ x(nT) มากที่สุด ตัวเลขที่ได้จากการตีความหรือเรียกว่าเป็น สัญญาณออกจากวงจร A/D นี้จะเรียกว่า สัญญาณเชิงเลข

4. ระบบเวลาจริง (Real-time system) : โดยทั่วไปคำนี้ใช้กับระบบการประมวลสัญญาณที่การคำนวณการประมวลผลทำได้เสร็จสิ้น ก่อนที่จะมีการสุ่มตัวอย่างสัญญาณลำดับใหม่ อย่างไรก็ตามคาบเวลาในการสุ่มตัวอย่างสัญญาณควรมีค่าน้อย

5. ลำดับ (Sequence) : บางครั้งอาจเรียกเป็นลำดับข้อมูล หรือ ลำดับสัญญาณ ซึ่งคำว่าลำดับอาจใช้แทนสัญญาณเชิงเต็มหน่วย หรือ สัญญาณเชิงเลขก็ได้

1.4 ทฤษฎีการสุ่มตัวอย่าง

อาจกล่าวได้ว่าสัญญาณในธรรมชาติส่วนมาก เช่น สัญญาณเสียง, สัญญาณการสั่นของพื้นโลก, คลื่นหัวใจ หรือ การแปรค่าไปของอุณหภูมิ เหล่านี้เป็นไปในลักษณะแบบต่อเนื่องกับพิสัยเวลาหรือกล่าวได้ว่าเป็นสัญญาณเชิงอุปมาน การนำสัญญาณเหล่านี้ไปประมวลผลในลักษณะการประมวลผลสัญญาณเชิงเลข หรือ การประมวลผลสัญญาณเชิงเต็มหน่วย (discrete signal processing) ได้ต้องใช้ระบบการประมวลผลตามรูปที่ 1.1 ซึ่งตามรูปวงจร S/H (sample & Hold) เป็นวงจรสุ่มและคงค่าสัญญาณไว้เพื่อให้วงจร A/D ทำการแปลงเป็นตัวเลขอีกทีหนึ่ง

อย่างไรก็ตามการเปลี่ยนสัญญาณเชิงอุปมานมาเป็นสัญญาณเชิงเลขนั้น ในเบื้องแรกต้องมีการสุ่มตัวอย่างก่อน ซึ่งอัตราหรือความถี่ในการสุ่มตัวอย่างโดยไม่ทำให้สูญเสียข้อมูลสำคัญไปนั้น ทฤษฎีการสุ่มตัวอย่าง (sampling theory) ของแชนนอน (Shannon) กล่าวไว้ว่าถ้าหากเรามีสัญญาณเชิงอุปมาน x(t) ที่ค่าการแปลงฟูริเยอร์ หรือ สเปกตรัมกำลัง (power spectrum) ของมันมีแถบความถี่ปฏิบัติงาน (bandwidth) เท่ากับ f_0 แล้ว เราสามารถทำการสุ่มตัวอย่าง โดยที่สัญญาณที่ได้ไม่สูญเสียเนื้อหาสำคัญก็ต่อเมื่อความถี่ในการสุ่มตัวอย่าง f_s มีค่ามากกว่าหรือ เท่ากับสองเท่าของความถี่ f_0 หรือโดยทั่วไปเราอาจสุ่มตัวอย่างด้วยค่าความถี่ $f_{sn} = 2f_0$ พอดี ค่าความถี่นี้มีชื่อเรียกว่าความถี่นีควิท (Nyquist frequency) และคาบเวลา $T_n = 1/(2f_0)$ นี้เรียกว่า ช่วงการสุ่มตัวอย่างนีควิท (Nyquist interval) ในทางปฏิบัติเพื่อหลีกเลี่ยงผลของปรากฏการณ์ไม่เป็นเชิงเส้น (nonlinearity) ที่อาจเกิดจากการสุ่มตัวอย่าง เรามักใช้ความถี่ในการสุ่มตัวอย่าง f_s มากกว่าค่าความถี่นีควิท หรือ f_{sn} ขึ้นไป ส่วนจะมีค่ามากกว่าเท่าใดนั้นขึ้นกับลักษณะงานไม่ได้มีการกำหนดค่าที่แน่นอน ซึ่งอาจใช้เป็น

2.5 f_0 หรือ $10f_0$ ก็ได้ตามทฤษฎีการสุ่มตัวอย่างของแชนนอนนั้นเห็นได้ว่าการจะสุ่มตัวอย่าง สัญญาณได้ถูกต้องก็ต่อเมื่อต้องรู้ค่าแถบความถี่ปฏิบัติงานของสัญญาณ หรือพูดอีกนัยหนึ่งสัญญาณต้องมีแถบความถี่ปฏิบัติงานจึงจะทำการสุ่มตัวอย่างได้ ดังนั้นในบางครั้งเพื่อให้มั่นใจได้ว่าสัญญาณที่ทำการประมวลผลถูกประมวลผลอย่างถูกต้อง ในภาคแรกของระบบประมวลผลเชิงเต็มหน่วย และเชิงเลขจึงอาจมีวงจรกรองผ่านความถี่ต่ำ (low-pass filter) ไว้เป็นตัวกำหนดแถบความถี่ปฏิบัติงานของสัญญาณดังได้แสดงในรูป 1.1

1.5 ทฤษฎีเบื้องต้นเกี่ยวกับการกรองสัญญาณเชิงเลข[11]

วงจรฟิลเตอร์กล่าวได้ว่าเป็นวงจรที่มีความสำคัญในระบบอิเล็คทรอนิคส์ ทฤษฎีของวงจรฟิลเตอร์แบ่งตามประวัติได้เป็น 2 แขนงคือ ทฤษฎีดั้งเดิม (classical filter theory) และทฤษฎียุคใหม่ (modern filter theory) ทฤษฎีแบบยุคใหม่นี้จะให้ผลที่ละเอียดและแน่นอนกว่าทฤษฎียุคดั้งเดิม วงจรฟิลเตอร์แบ่งออกได้เป็นประเภทใหญ่ ๆ เช่น Passive Filter, Active Filter, Electromechanical Filter, Digital Filter, Microwave Filter

การกรองสัญญาณดิจิตอล (Digital Filter) คือ กระบวนการคำนวณ (computational process) หรืออัลกอริทึม (algorithm) ซึ่งสัญญาณดิจิตอลหรือลำดับของตัวเลขที่เป็นสัญญาณเข้า (input) ถูกแปลงให้เป็นลำดับของตัวเลขใหม่ที่เรียกว่า สัญญาณออกดิจิตอล (output digital signal) กระบวนการคำนวณอาจจะเป็นแบบการกรองความถี่ต่ำ (low pass filter) การกรองความถี่สูง (high pass filter) การหาค่าอนุพันธ์ (derivative) เป็นต้น

1.5.1 สมการแตกต่างเชิงเส้นสัมประสิทธิ์คงที่

(Linear Constant Coefficient Differential Equations)

ระบบ shift-invarient สามารถเขียนแทนด้วยสมการแตกต่างเชิงเส้นสัมประสิทธิ์คงที่อันดับที่ N ได้ดังนี้

$$\sum_{k=0}^{N} a_k Y(n-k) = \sum_{r=0}^{M} b_r X(n-r) \qquad ;a \neq 0 \tag{1.1}$$

$$a_0 Y(n) + \sum_{k=1}^{N} a_k Y(n-k) = \sum_{r=0}^{M} b_r X(n-r) \tag{1.2}$$

เมื่อ $a_0, a_1, \ldots, a_n$ และ $b_0, b_1, \ldots b_m$ เป็นค่าคงที่ซึ่งใช้ในการกำหนดคุณสมบัติ (characteristic) ของระบบ ถ้าเรากำหนดให้ $a_0 = 1$ เราจะได้

$$Y(n) = \sum_{r=0}^{M} b_r X(n-r) + \sum_{k=1}^{N} a_k Y(n-k) \qquad (1.3)$$

เราเรียกสมการที่ (1.3) ว่าสมการ digital filter โดยที่ a_k และ b_r เป็นค่าสัมประสิทธิ์จะมีค่าเท่าไรขึ้นอยู่กับชนิดของฟิลเตอร์ซึ่งเราเรียกฟิลเตอร์แบบนี้ว่า **รีเคอร์ซีฟฟิลเตอร์** (recursive filter) ถ้าสัมประสิทธิ์ a_k ของ Y(n-k) มีค่าเป็นศูนย์เราเรียกฟิลเตอร์แบบนี้ว่า **นอนรีเคอร์ซีฟฟิลเตอร์** (non-recursive filter) ซึ่งเขียนความสัมพันธ์ได้ดังสมการที่ (1.4)

$$Y(n) = \sum_{r=0}^{M} b_r X(n-r) \qquad (1.4)$$

1.5.2 การสร้างวงจรกรองความถี่ดิจิตอลแบบรีเคอร์ซีฟ (Recursive Filter Synthesis)

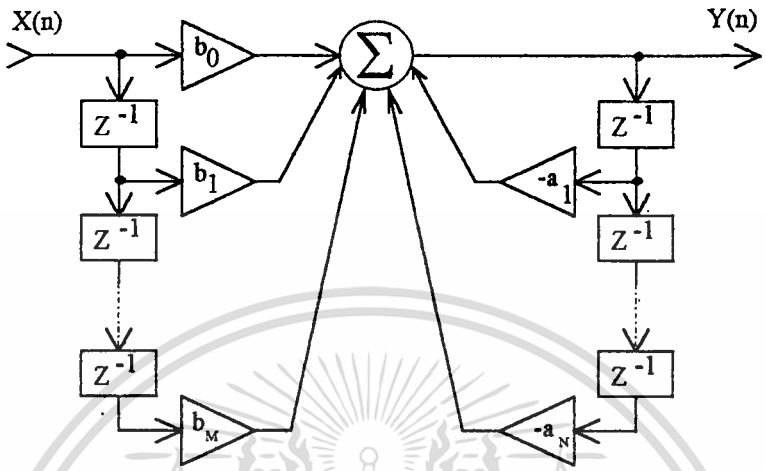
การสร้างวงจรกรองความถี่ดิจิตอลจะต้องเขียนความสัมพันธ์ระหว่างสัญญาณ input และสัญญาณ output ให้อยู่ในรูปแบบหรือ algorithm ที่ใช้คำนวณก่อน algorithm นี้จะใช้ตัวบวก (adder) ตัวหน่วง (delay) และ ตัวคูณด้วยค่าคงที่ (constant multiplier) มาช่วยเขียนโครงสร้างของวงจรจากสมการของ digital filter สมการที่ (1.3) กรณี r = k เราจะได้

$$Y(n) = \sum_{k=0}^{M} b_k X(n-k) - \sum_{k=1}^{N} a_k Y(n-k) \qquad (1.5)$$

Z - Transform

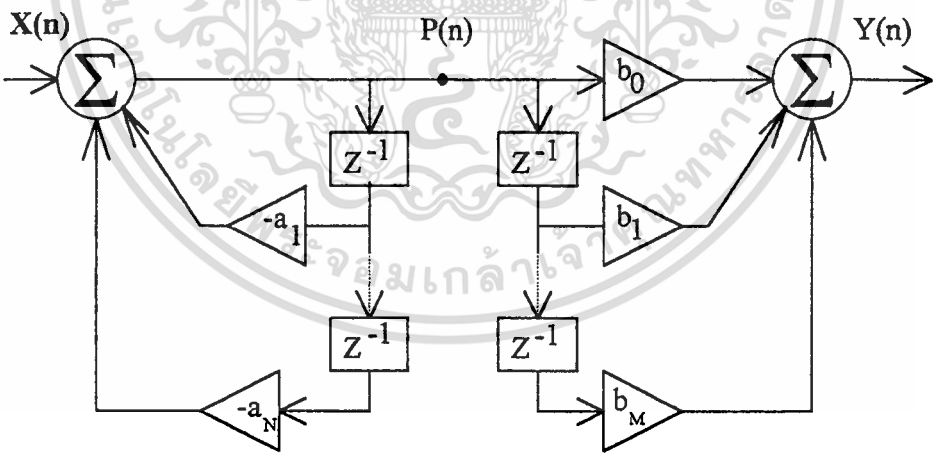
$$Y(z) = \sum_{k=0}^{M} b_k X(z) Z^{-k} - \sum_{k=1}^{N} a_k Y(z) Z^{-k} \qquad (1.6)$$

ซึ่งเราสามารถแทนสมการ (1.6) ด้วย algorithm ในการคำนวณได้โดยตรงนั้นคือเอาค่า delay ของสัญญาณ input คูณด้วยสัมประสิทธิ์ b_k และค่า delay ของสัญญาณ output คูณด้วยสัมประสิทธิ์ a_k แล้วนำผลคูณ ทั้งหมดเอามาเข้าวงจรรวมข้อมูล (Summing) โดยสามารถเขียนเป็น block diagram ดังรูปที่ 1.2 ซึ่งเรียกว่า direct form I



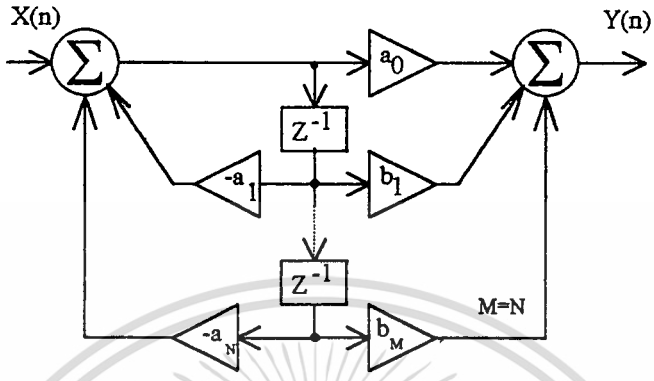
รูปที่ 1.2 Direct form I

จาก direct form I เราจะแยกให้อยู่ในรูปของ direct form II ดังรูปที่ 1.3



รูปที่ 1.3 Direct form II

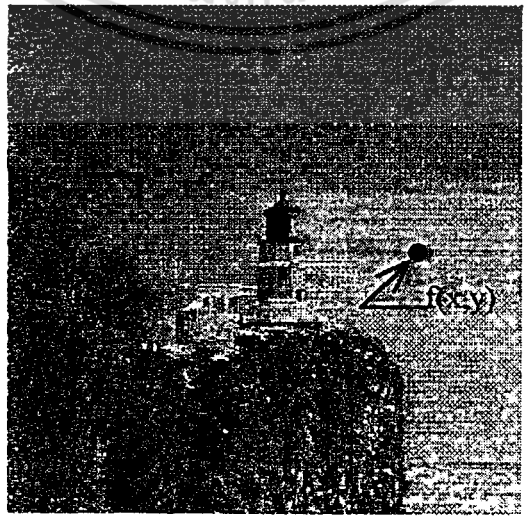
จากรูปที่ 1.3 จะเห็นว่าเราสามารถใช้ delay ร่วมกันได้ เมื่อยุบ delay รวมกันแล้วเราเรียกการเขียนรูปแบบนี้ว่า direct form II canonical แสดงดังรูปที่ 1.4



รูปที่ 1.4 Direct form II canonical

1.6 การประมวลสัญญาณภาพ (Image Processing)[10]

ในที่นี้จะกล่าวถึงการประมวลสัญญาณเชิงเลขสองมิติซึ่งเป็นข้อมูลภาพ (Image Processing) และองค์ประกอบพื้นฐานในการประมวลภาพข้อมูล รูปแบบของอิมเมจ การฟิลเตอร์ การแทนภาพในระบบดิจิตอล (Digital Image Representation) ในระบบภาพโดยทั่วไปสามารถแทนด้วยฟังก์ชั่นของความเข้มแสงในระนาบสอง มิติ : f(x,y) โดยที่ x และ y แทนตำแหน่งคู่ลำดับที่เกิดขึ้นในภาพจริง และค่าฟังก์ชั่น f ณ จุด (x,y) ใด ๆ จะเป็นสัดส่วนโดยตรงกับความสว่างหรือระดับสีเทา (gray level) ของภาพที่จุดนั้น ๆ แสดงได้ดังรูปที่ 1.5



รูปที่ 1.5 ค่าของฟังก์ชั่น f(x,y) ในระบบภาพดิจิตอล

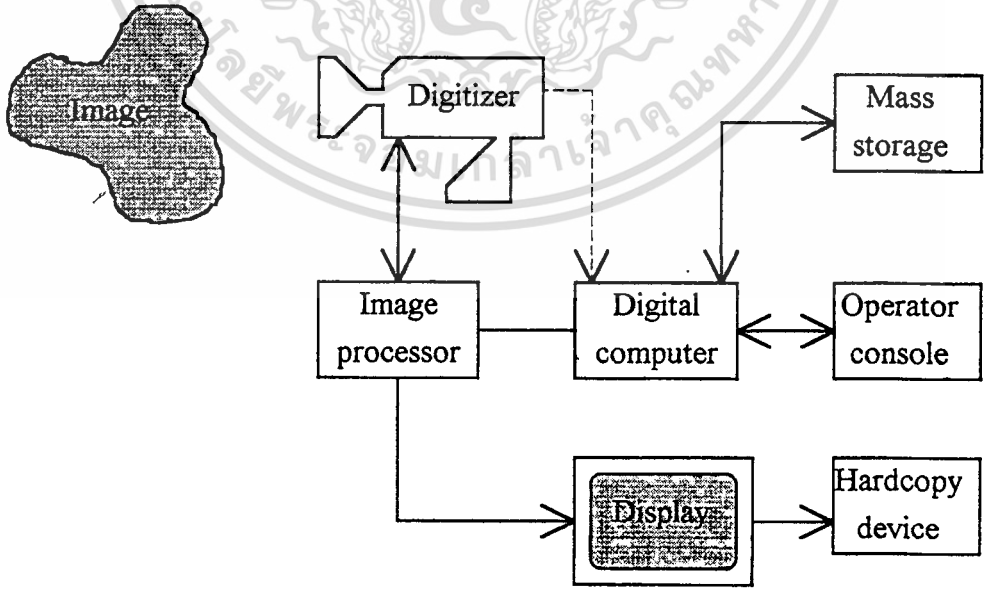
สำหรับข้อมูลภาพแบบดิจิตอล f(x,y) มีลักษณะที่ไม่ต่อเนื่องทั้งในระนาบสเปเชียล (spatial) ดังนั้นจึงพิจารณาให้ข้อมูลภาพแบบดิจิตอลแทน อยู่ในรูปของเมตริกซ์ ซึ่งเมตริกซ์จะมีแถว และหลักที่มีลักษณะเป็นเอกลักษณ์ และมีความสัมพันธ์กับค่าสมาชิกในแถวและหลัก ที่แทนด้วยระดับสีเทาของภาพที่จุดนั้น ๆ ถึงแม้ว่าขนาดของข้อมูลภาพแบบดิจิตอล จะแปรเปลี่ยนไปตามการประยุกต์ใช้งาน แต่จะมีข้อดีที่ได้เปรียบ ถ้ากำหนดให้มีลักษณะเฉพาะเป็นอะเรย์แบบสมมาตร (square array) และจำนวนของระดับสีเทาเป็นเลขจำนวนเต็มกำลังสอง

1.6.1 องค์ประกอบของระบบประมวลข้อมูลภาพดิจิตอล

(Elements of A Digital Image Processing System)

องค์ประกอบพื้นฐานโดยทั่วไปในระบบการประมวลข้อมูลภาพ ประกอบด้วย 5 ส่วนใหญ่ ๆ คือ

1. ส่วนเปลี่ยนสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (Digitizer)
2. เครื่องประมวลผลข้อมูลภาพ (Image Processor)
3. เครื่องคอมพิวเตอร์แบบดิจิตอล (Digital computer)
4. อุปกรณ์ที่ใช้ในการแสดงผล และบันทึกผล (Display and Recording Devices)
5. อุปกรณ์ที่ใช้ในการเก็บรักษาข้อมูล (storage Devices)



รูปที่ 1.6 แสดงองค์ประกอบของระบบการประมวลผลแบบดิจิตอล[10]

1. ส่วนเปลี่ยนสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอล (Digitizer)

ส่วนของดิจิไตเซอร์ ทำหน้าที่เปลี่ยนสัญญาณภาพให้อยู่ในรูปของตัวเลขเพื่อใช้เป็นข้อมูลป้อนเข้าดิจิตอลคอมพิวเตอร์ ส่วนนี้ได้แก่ กล้องโทรทัศน์ ดิจิไตเซอร์ ภายในประกอบด้วยหลอดวิดิคอน ทำหน้าที่เป็นสื่อนำไฟฟ้าทางแสง ภาพจะถูกโฟกัสลงบนผิวของหลอด ซึ่งถูกเปลี่ยนให้เป็น สัญญาณไฟฟ้า ซึ่งจากการควอนไทซิ่งสัญญาณนี้จะได้ภาพดิจิตอลเกิดขึ้น

2. เครื่องประมวลข้อมูลภาพ (Image Processor)

เครื่องประมวลข้อมูลภาพแบบดิจิตอลเป็นหัวใจของระบบการประมวลข้อมูลภาพเครื่องประมวลข้อมูลภาพ ประกอบด้วยชุดฮาร์ดแวร์ที่มี 4 ฟังก์ชั่นพื้นฐานคือ การหยุดภาพนิ่ง (image acquisition) การเก็บข้อมูลภาพ (storage) การประมวลภาพในระดับต่ำ (แต่รวดเร็ว) (Low Level (fast) processing) และการแสดงผล (display) โดยทั่วไปโมดูลของการหยุดภาพนิ่ง จะมีสัญญาณโทรทัศน์เป็นหน่วยอินพุท และทำการเปลี่ยนสัญญาณนี้ให้เป็นข้อมูลแบบดิจิตอล เครื่องประมวลข้อมูลภาพสมัยใหม่ส่วนมากจะใช้การดิจิไตซ์ภาพโทรทัศน์ ในเวลา 1 เฟรม (one frame-time) เช่น ในเวลา 1/30 วินาที จึงมักเรียกว่าเครื่องจับภาพนิ่ง (frame grabber)

โมดูลของการเก็บภาพแบบที่เรียกว่า เฟรมบัฟเฟอร์ (frame buffer) จะทำหน้าที่เก็บรักษาหน่วยความจำของภาพแบบดิจิตอล โดยทั่วไปจะมีเครื่องมือ 2-3 ชุด ที่รวมเข้าไว้ในเครื่องประมวลข้อมูลภาพ คุณสมบัติการหยุดภาพของโมดูลการเก็บภาพประกอบด้วยหน่วยความจำที่สามารถอ่านข้อมูลจากอัตราของโทรทัศน์ปกติ มีอัตรา 30 ภาพ ต่อวินาที ลักษณะนี้เป็นไปตามการทำงานของโมดูลการหยุดภาพนิ่ง ที่ประกอบไปด้วยภาพที่สมบูรณ์ในการเก็บภาพที่เร็วเท่าที่ภาพถูกจับให้นิ่ง หน่วยความจำสามารถเปลี่ยนให้เป็นแอดเดรสของโทรทัศน์ได้โดยชุดแสดงผลของภาพที่จอมอนิเตอร์

เครื่องประมวลข้อมูลภาพยังมีหน้าที่ในฟังก์ชั่นระดับต่ำอีกด้วย (Low-level function) เช่น การประมวลผลทางคณิตศาสตร์ และทางตรรกศาสตร์ หรือที่เรียกว่า หน่วยตรรกเลขคณิต (Arithmetic Logic Unit : ALU) โดยถูกออกแบบพิเศษเพื่อที่จะเพิ่มความเร็วในการประมวลผลแบบขนาน ส่วนโมดูลแสดงผล (display module) ทำหน้าที่อ่านหน่วยความจำของภาพ แล้วทำการเปลี่ยนข้อมูลของภาพที่เป็นแบบดิจิตอลให้เป็นสัญญาณวีดีโอแบบอนาล็อก และแสดงผลสัญญาณที่แปลงได้นั้น ให้ออกทางมอนิเตอร์โทรทัศน์ หรืออุปกรณ์วีดีโอ ชนิดอื่น

3. เครื่องคอมพิวเตอร์แบบดิจิตอล (Digital Computer)

ถึงแม้ว่าเครื่องประมวลข้อมูลภาพ อาจเปรียบได้กับความสามารถในการประมวลผลภายใน ดังที่ได้กล่าวมาแล้วแต่ว่าระดับของการประมวลผลยังอยู่ในระดับค่อนข้างต่ำ ในด้านความทันสมัยซึ่งโดยทั่วไปพบว่า เครื่องประมวลข้อมูลภาพนี้จะถูกต่อเชื่อม (interface) กับเครื่องคอมพิวเตอร์ระบบ

จะมีความเหมาะสมเฉพาะกับงาน ๆ หนึ่งเท่านั้น ไม่สามารถนำเทคนิคเดียวกันนี้มาใช้ร่วมกันได้หมด ในหัวข้อนี้จะกล่าวถึงรูปแบบของอิมเมจ วิธีการฟิลเตอร์ การตัดค่าเทรสโฮลด์ และวิธีการเลือกค่าฮีสโตรแกรม เป็นต้น

1. โมเดลข้อมูลภาพ (an Image Model)

ในที่นี้คำว่าข้อมูลภาพ หมายถึง ฟังก์ชั่นของความเข้มแสง 2 มิติโดยมีรูปแบบ f(x,y) ซึ่งขนาดหรือ แอมปิลจูด (amplitute) ของฟังก์ชั่น ที่พิกัดเสปเชี่ยล (spatial coordinates) ที่จุด (x,y) จะแสดงค่าความสว่าง (brightness) ของข้อมูลภาพที่จุดนั้น ๆ เมื่อแสงเป็นรูปแบบพลังงานแบบหนึ่ง ดังนั้น f(x,y) จึงต้องมีขนาดไม่เท่ากับศูนย์ และจะต้องมีขนาดไม่เท่ากับอนันต์ (finite) ดังนั้น จะได้ว่า

$$0 < f(x,y) < \alpha$$

พื้นฐานโดยทางธรรมชาติของ f(x,y) อาจจะได้รับการพิจารณาเกี่ยวกับคุณสมบัติ สองประการดังต่อไปนี้ คือ ประการแรก จะพิจารณาเกี่ยวกับแหล่งกำเนิดแสงทั้งหมดที่เกิดขึ้นเองบนฉาก ประการที่สอง จะพิจารณาเกี่ยวกับแสงสะท้อนทั้งหมด ที่เกิดขึ้นจากการสะท้อนของวัตถุในฉากเอง คุณสมบัติทั้งสองประการนี้ จะถูกเรียกว่า การส่องสว่าง (illumination) และการสะท้อน (reflectance) ซึ่งมีรูปแบบฟังก์ชั่นดังนี้คือ i(x,y) และ r(x,y) ตามลำดับ ทั้งสองฟังก์ชั่นนี้รวมกันจะกลายเป็นรูปแบบของ f(x,y) จะได้ว่า $f(x,y) = i(x,y) * r(x,y)$ เมื่อ $0 < i(x,y) < \alpha$ และ $0 < r(x,y) < \alpha$

2. ฟิลเตอร์ (Filter)

ฟิลเตอร์เป็นตัวกรองสัญญาณภาพเบื้องต้น ให้ได้ภาพที่มีความคมชัดมากยิ่งขึ้น เพื่อที่จะนำภาพไปประยุกต์ใช้ได้ดียิ่งขึ้น ซึ่งมีมากหลายชนิด แต่ในที่จะกล่าวเฉพาะที่นิยมใช้กันมากเท่านั้น

- การเฉลี่ยค่ารอบย่าน (Neighbourhood Average)

การเฉลี่ยค่ารอบย่านเป็นเทคนิคที่ใช้กันในเสปเชียลโดเมนโดยตรง สำหรับการทำภาพให้ชัดเจนยิ่งขึ้น ให้ภาพอินพุท f(x,y) มีขนาด nxn และ g(x,y) เป็นผลของภาพที่ได้จากการปฏิบัติของระดับสีเทาทุก ๆ จุด (x,y) ได้จากการเฉลี่ยค่าระดับสีเทาของพิเซลในภาพ f(x,y) ซึ่งสามารถอธิบายได้ดังนี้

$$g(x,y) = \frac{1}{M}\sum_{n,m} f(n,m)$$

สำหรับ x,y = 0,1,..., N-1 และ M คือ ผลรวมของจำนวนจุดในย่านนั้น ๆ

- ฟิลเตอร์แบบมัธยฐาน (Median Filter)

สิ่งสำคัญประการหนึ่งจากการเฉลี่ยค่ารอบย่าน คือ การเกิดความเบลอร์ที่ขอบของภาพ และมีรายละเอียดสัญญาณรบกวนที่ชัดเจน อาจแก้ไขได้โดยการใช้เทรสโฮลด์ (threshold) ซึ่งโดยทั่วไป แล้วค่าเทรสโฮลด์จะได้จากการลองผิดลองถูก แต่ถ้าใช้ฟิลเตอร์แบบมัธยฐาน แทนระดับสีเทาของแต่ละพิเซล โดยการใช้ค่าเฉลี่ยมัธยฐานของระดับสีเทาในรอบย่านนั้น ๆ ของพิเซล จะทำให้ภาพที่ได้มีประสิทธิภาพดีขึ้น การหาค่ามัธยฐาน (M) คือ เซ็ตที่มีค่าในเซ็ตครึ่งหนึ่งมากกว่า M และอีกครึ่งหนึ่งน้อยกว่า M การหาค่ามัธยฐาน สามารถหาได้โดยทำการเรียงข้อมูลจากน้อยไปหามาก แล้วนำค่าที่อยู่ตรงกลางนั้นมาใช้แทนค่า M

ตัวอย่าง

สมมติว่า มีย่านขนาด 3x3 มีข้อมูล

(10, 20, 20, 20, 15, 20, 20, 25, 100)

สามารถเรียงข้อมูลใหม่ได้เป็น

(10, 15, 20, 20, 20, 20, 20, 25, 100)

ดังนั้นค่าที่อยู่ตรงกลางเป็นค่ามัธยฐานมีค่าเท่ากับ 20 นั่นเอง ความคิดที่มีอิทธิพลต่อฟังก์ชั่นของฟิลเตอร์แบบมัธยฐาน คือ จะทำให้จุดที่มีความเข้มที่แตกต่างมากกว่าย่านรอบ ๆ ของมัน

- ฟิลเตอร์แบบความถี่ต่ำผ่าน (Low - pass Filter)

สัญญาณรบกวนในระดับสีเทาของภาพมักจะมีส่วน ของสัญญาณความถี่สูง ของการแปลงฟูเรียร์อย่างมากซึ่งจะทำให้เกิดการเบลอร์ของภาพ ดังนั้นจึงต้องหาวิธีการกำจัดเฉพาะช่วงความถี่สูงที่ไม่ต้องการออกไป จากทฤษฎีการทำคอนโวลูชั่นจะได้ว่า

$$G(u,v) = H(u,v)\, F(u,v)$$

เมื่อ F(u,v) คือ การแปลงฟูเรียร์ของภาพที่ต้องการให้เรียบ ปัญหาก็คือ การเลือกทรานส์เฟอร์ฟังก์ชั่น H(u,v) ที่ให้ได้ผลลัพธ์ G(u,v) โดยที่สามารถกำหนดจุดช่วงที่ความถี่สูงของ F(u,v) ออกไปได้ การแปลงอินเวอร์สลาปาส (Inverse Laplace) ของ G(u,v)

$$g(x,y) = L^{-1}\,[\, H(u,v)\, F(u,v)\,]$$

จะได้ว่า g(x,y) เรียบขึ้น เพราะว่า ส่วนประกอบความถี่สูงไม่สามารถผ่านออกไปได้ แต่ช่วงความถี่ต่ำจะสามารถผ่านไปได้ โดยปราศจากการลดทอนของสัญญาณซึ่งวิธีนี้เรียกว่า ฟิลเตอร์ แบบความถี่ต่ำผ่าน ฟังก์ชั่น H(u,v) ถูกเรียกว่า ทรานส์เฟอร์ฟังก์ชั่นของฟิลเตอร์ (Filter Transfer Function) ซึ่งในการคำนวณจะพิจารณาเฉพาะ H(u,v) เท่านั้น ฟิลเตอร์แบบความถี่ต่ำผ่านนี้มีอยู่หลายชนิดด้วยกัน โดยที่แต่ละฟิลเตอร์จำทำหน้าที่กรองความถี่สูงออกไปให้เหลือเฉพาะความถี่ต่ำเท่านั้น

บทที่ 2
การกระจายทางคณิตศาสตร์

2.1 ระบบตัวเลข

ก่อนอื่นจะกล่าวถึงพื้นฐานของเลขฐานสองก่อน เนื่องจากในระบบเลขฐานสิบนั้นมีเลขอยู่สิบค่าที่เป็นฐานคือ 0 ถึง 9 สัญลักษณ์ + และ - แทน บวก และ ลบ ตามลำดับ และจุดในเลขฐานสิบใช้เป็นตัวแยกระหว่างตัวเลขจำนวนเต็มและเศษส่วน แต่ในดิจิตอลฮาร์ดแวร์มีเลขอยู่สองค่าที่เป็นฐานคือ 0 และ 1 ซึ่งเลขทั้งสองค่าดังกล่าวใช้เป็นสัญลักษณ์แทนทั้ง +, - และจุด ถ้าแทนเลขไบนารี่ของตัวเลขจำนวนเต็มที่ไม่มีเครื่องหมายซึ่งเป็นสัญลักษณ์เลขไบนารี่จำนวนเลขไบนารี่ B bit (X^{B-1}, X^{B-2},..., X^0) สามารถแปลงเป็นเลขฐานสิบด้วยสูตร

$$X = \sum_{J=0}^{B-1} X^J 2^J$$

โดย X = 0 หรือ 1 ดังนั้นเลขไบนารี่ 11011 แทนเป็นเลขฐานสิบได้ 26 ดังนี้ $0X2^0 + 1X2^1 + 0X2^2 + 1X2^3 + 1X2^4 = 2+8+16 = 26$ ซึ่งคล้ายระบบเลขฐานสิบ นั่นคือ $26 = 2X10^1 + 6X10^0$ จากสมการข้างบนค่าตัวเลขจำนวนเต็มที่มากที่สุดที่ถูกแทนโดยเลขฐานสอง B bit คือ $2^B - 1$ ดังนั้น 10, 16, หรือ 32 บิท สามารถถูกแทนด้วยเลขจำนวนเต็ม 1023,65,435 หรือ 4,150,998,095 ตามลำดับ ค่าที่อธิบายข้างบนสามารถเปลี่ยนแปลงเป็นเลขเศษส่วนได้ดังตัวอย่างต่อไปนี้ สมมุติ 16 บิท ถูกแทนด้วยเลขฐานสิบและทุกจำนวนถูกสเกลด้วย $2^{-5} = 0.03125$ ดังนั้นค่าที่ มากที่สุดที่สามารถแทนได้คือ $(2^{16} - 1) 2^{-5} = 2047$ และค่าที่น้อยที่สุดคือ 0.03125 มันเปรียบเสมือนกับว่ามีจุดในตำแหน่งที่ 5 จาก Least Significant Bit (LSB) และคำนวณส่วนที่ เป็นเศษส่วนในสมการข้างบนโดยการใช้ค่า 2 ยกกำลังด้วยค่าลบ ดังนั้นในกรณีนี้เลขไบนารี่ 0101100010110110 แทน $0X10^{10} + 1X10^9 + 0X10^8 + 1X10^7 + 1X10^6 + 0X10^5 + 0X2^4 + 1X2^3 + 1X2^2 + 0X2^1 + 1X2^0 + 1X2^{-1} + 0X2^{-2} + 1X2^{-3} + 1X2^{-4} + 0X2^{-5}$ หรือเท่ากับ 709.6875

ในขบวนการดิจิตอล (เหมือนในคอมพิวเตอร์) ทุกจำนวนมีขนาดน้อยกว่าหรือเท่ากับหนึ่งซึ่งสามารถทำได้โดยสมมุติว่าจำนวนบิทมีจุดอยู่ทางซ้ายของ Most Significant Bit (MSB) หรือทุกจำนวนถูกสเกลด้วย 2^{-B} ซึ่งทำให้เลขไบนารี่มากที่สุด 1111...1, $(2^B - 1)/2^{-B} = 1 - 2^{-B}$ ซึ่งน้อยกว่า 1

- Sign-Magnitude Notation

ใน Sign-Magnitude Notation เลขที่มีค่า absolute เท่ากันจะแทนด้วยเลขไบนารี่ที่เหมือนกันแต่จะมีบิทพิเศษเรียกว่า บิทเครื่องหมาย (Sign Bit) รวมเข้าไปทางซ้ายของ MSB เพื่อทำให้เกิดความแตกต่างระหว่างค่าบวกกับลบ ตัวอย่างเช่น

+ 0.828125 = 0 110101

- 0.828125 = 1 110101

ดังนั้นเลขไบนารี่ B บิทใน Sign-Magnitude Notation ซึ่งสมมุติว่ามีจุดอยู่ระหว่าง Sign Bit และ MSB มีค่าอยู่ระหว่าง $-(1 - 2^{-B+1}) \leq X \leq (1 - 2^{-B+1})$ ทำให้จำนวนค่าบวกมีจำนวนเท่ากับจำนวนค่าลบ

และค่า 0 แทนได้ทั้ง + 0 หรือ - 0

- 2'S Complement Notation

ใน 2'S Complement Notation จำนวนบวกเหมือนกับ Sign-Magnitude Notation ซึ่งบิทแรกเป็นบิทเครื่องหมาย จำนวนลบแทนโดยการเอาค่า absolute ลบออกจาก 2 และเพราะว่าค่าของ absolute ต้องน้อยกว่า 1 ดังนั้นบิทแรกจะเป็น 1 เสมอเปรียบเทียบด้วยตัวอย่างค่า 2'S Complement ของ - 0.375 หาได้โดย 2 - 0.375 = 1.625 ซึ่งค่าไบนารี่คือ 1.101 ในการแปลงเลข 2'S Complement โดยการสมมุติว่าบิทแรก (sign bit) ถูกคูณด้วย -1 ดังนั้นถ้า X ถูกแทนที่ด้วยเลขไบนารี่ B บิท ($X^0, X^1, ...X^{B-1}$) ใน 2's complement notation , ค่าของ X ในระบบเลขฐานสิบ หาได้จาก

$$X = -X^0 + \sum_{j=1}^{B-1} X^j 2^{-j}$$

ดังนั้นตามสมการเลขไบนารี่ 2'S Complement 1101 แทนเป็นเลขระบบฐานสิบ

$$-1 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} = -1 + 0.5 + 0.125 = - 0.375$$

จากสมการการแปลง 2's complement จำนวนที่มีค่ามากที่สุด คือ $1-2^{-B+1}$ (0111...1) และจำนวนที่มีค่าน้อยที่สุด คือ -1 (1000...0) นอกจากนั้น ค่า 0 แทนด้วยค่าเดียวคือ (000...0), และมีค่าที่เป็นลบมากกว่าค่าที่เป็นบวกอยู่หนึ่งจำนวนคือ (-1)

2.2 ทฤษฎีการกระจายทางคณิตศาสตร์

การกระจายทางคณิตศาสตร์ (Distributed Arithmetic)[1,2,4] หรือเรียกย่อ ๆ ว่า DA ใช้กันมากในการจัดรูปแบบคณิตศาสตร์ โดยเฉพาะจะปรากฏอยู่ในงานการประมวลผลสัญญาณดิจิตอล โดยจัดสมการให้อยู่ในรูปแบบการคำนวณแบบบิท เพื่อให้สมการเปลี่ยนเป็นวงจรดิจิตอลได้ โดยเฉพาะความเร็วในการคูณ ซึ่งจะออกแบบระบบการคูณแบบเปิดตารางค่าสัมประสิทธิ์ ที่ถูกเก็บไว้ในหน่วยความจำแบบอ่านได้อย่างเดียว การประยุกต์ใช้การกระจายทางคณิตศาสตร์ (DA) นี้ จะแปลงสมการผลบวกของผลคูณให้กระทำการคำนวณแบบบิท ซึ่งจะทำให้หาผลลัพธ์ได้เร็วอุปกรณ์สำคัญที่ใช้ใน DA มีชิฟท์รีจีสเตอร์ (Shift Register) เพื่อเลื่อนข้อมูลจากเวิร์ดเป็นบิท รอม (Read only Memory : ROM) สำหรับเก็บค่าสัมประสิทธิ์ ตัวรวมข้อมูล (Accumulator) เพื่อรวมข้อมูลจากรอมเมื่อเปรียบเทียบอุปกรณ์รอบข้างระหว่าง DA กับ Standard CPU อย่างเช่นตระกูล TMS320 ก็มีลักษณะคล้ายกัน คือมี DAC (Digital to Analog Converter), ADC (Analog to Digital Converter) และระบบสัญญาณนาฬิกาเช่นกัน แต่ DA จะไม่มี CPU ประมวลสัญญาณเรื่องของความเร็ว DA จะเร็วกว่าเพราะว่า ถ้าสมมุติให้คำนวณหาค่า $Y = A_1X_1 + A_2X_2 + A_3X_3 + A_4X_4$ TMS32010 จะใช้เวลาประมาณ 640ns เพราะคำสั่งคูณ 16x16 บิทใช้เวลา 160ns ส่วน DA ถ้าใช้รอมความเร็วสูงและข้อมูลเข้ามีขนาด 16 บิทต่อ 1 เวิร์ด ส่งเข้าขาแอดเดรสของรอมแบบ 1 BAAT [3] DA

ใช้สัญญาณนาฬิกาประมวลผล 16 ลูก, 2 BAAT สัญญาณนาฬิกา 8 ลูก, 8 BAAT ใช้สัญญาณนาฬิกา 2 ลูก แต่ถ้าเป็น 16 BAAT จะใช้สัญญาณนาฬิกา 1 ลูกเท่านั้น Standard CPU สามารถที่จะประมวลสัญญาณได้หลายลักษณะรูปแบบขึ้นอยู่กับซอร์ฟแวร์หรือโปรแกรมที่เขียนให้หน่วยประมวลผลกลางทำงาน แต่ DA เมื่อออกแบบฮาร์ดแวร์ และค่าสัมประสิทธิ์ที่เก็บไว้ในรอมแล้วจะใช้กับระบบที่ออกแบบไว้ได้ระบบเดียวเท่านั้น ไม่สามารถควบคุมทางโปรแกรมได้ เพราะอาศัยการทำงาน ของฮาร์ดแวร์เป็นหลัก DA เป็นกรรมวิธีทางคณิตศาสตร์ที่จัดรูปแบบการคำนวณที่ส่วนใหญ่อยู่ในรูปผลบวกของผลคูณ (Sum of Products) ของเลขฐานสิบให้กระจายออกเป็นบิทหรือในรูปเลขฐานสองเพื่อให้การกระทำคณิตศาสตร์เปลี่ยนไปในรูปของวงจรดิจิตอลได้ วงจรดังกล่าวอาจเป็นวงจรซึ่งใช้แทนทรานส์เฟอร์ฟังก์ชั่น (Transfer function) ทางดิจิตอลของดิจิตอลฟิลเตอร์ (Digital filtering) แบบต่าง ๆ ที่มีลักษณะค่าสัญญาณเชิงอินพุทและหรือเอาท์พุทคูณกับค่าคงที่ การใช้ DA ในการประมวลผลจะมีประสิทธิภาพมากโดยเฉพาะสมการแบบผลบวกของผลคูณของสัญญาณดิจิตอลที่เป็นอินพุทกับค่าสัมประสิทธิ์ของสมการซึ่งจะได้กล่าวถึงรายละเอียดต่อไป

จุดประสงค์ของ DA คือ เปลี่ยนสมการคณิตศาสตร์ที่ส่วนใหญ่อยู่ในรูปของผลบวกของผลคูณเป็นวงจรดิจิตอลให้หาผลลัพธ์ออกมาได้อย่างรวดเร็วเมื่อเทียบกับไมโครโปรเซสเซอร์มาตรฐานในงาน ประมวลผลสัญญาณดิจิตอลที่มีสัญญาณนาฬิกาเท่ากันเพราะว่า DA ใช้หลักการเปิดตาราง (Look-up Tables) คูณค่าสัญญาณอินพุทกับค่าสัมประสิทธิ์ที่ถูกเก็บไว้ในหน่วยความจำ ลองพิจารณาตัวอย่างผลบวกของผลคูณต่อไปนี้

$$Y = \sum_{k=1}^{K} A_k \cdot X_k \qquad (2.1)$$

สมการที่ (2.1) ให้ A_k เป็นค่าสัมประสิทธิ์ X_k เป็นข้อมูลอินพุทซึ่ง X_k เป็น ตัวเลขคอมพลีเมนท์ของสอง (2's complement) และให้ Absolute ของ X_k <1 ดังนั้นแสดงค่าของ X_k ในเทอมของบิทได้

$$X_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} \cdot 2^{-n} \qquad (2.2)$$

เมื่อ b_{kn} = บิทต่าง ๆ ของ Xk เวิร์ดมีค่าเป็น 0 หรือ 1

b_{k0} = บิทที่แสดงเครื่องหมาย

b_{kN-1} = บิทที่มีนัยสำคัญต่ำสุด (LSB)

N = จำนวนบิทต่อข้อมูลอินพุท

เมื่อแทนค่า X_k จากสมการที่ (2.2) ลงในสมการที่ (2.1) ได้เป็นสมการที่ (2.3a) แสดงค่า Y ในเทอมของบิทของ X_k ดังนี้

$$Y = \sum_{k=1}^{K} A_k \left(-b_{k0} + \sum_{n=1}^{N-1} b_{kn} \cdot 2^{-n} \right) \qquad (2.3a)$$

สมการที่ (2.3a) สับเปลี่ยนจัดลำดับการบวกใหม่ให้เป็นกลุ่มย่อยจะได้

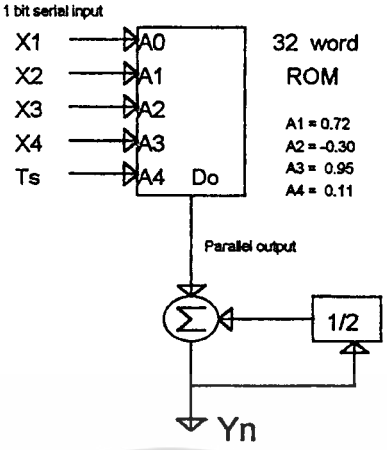
$$Y = \sum_{n=1}^{N-1} \left(\sum_{k=1}^{K} A_k b_{kn} \right) 2^{-n} + \sum_{k=1}^{K} A_k (-b_{k0}) \qquad (2.3b)$$

$$\begin{aligned} Y = & -2^{0}(b_{10}A_1 + b_{20}A_2 + b_{30}A_3 + \ldots\ldots + b_{k0}A_k) \\ & + 2^{-1}(b_{11}A_1 + b_{21}A_2 + b_{31}A_3 + \ldots\ldots + b_{k1}A_k) \\ & + 2^{-2}(b_{12}A_1 + b_{22}A_2 + b_{32}A_3 + \ldots\ldots + b_{k2}A_k) \\ & + 2^{-3}(b_{13}A_1 + b_{23}A_2 + b_{33}A_3 + \ldots\ldots + b_{k3}A_k) \\ & + 2^{-4}(b_{14}A_1 + b_{24}A_2 + b_{34}A_3 + \ldots\ldots + b_{k4}A_k) \\ & + 2^{-(N-1)}(b_{1,N-1}A_1 + b_{2,N-1}A_2 + \ldots\ldots + b_{k,N-1}A_k) \end{aligned} \qquad (2.3c)$$

สมการที่ (2.3c) เป็นการกระจายทางคณิตศาสตร์ของค่าสัมประสิทธิ์กับบิทต่าง ๆ ของข้อมูลอินพุท ด้วยเหตุนี้สมการที่ (2.3c) จึงเรียกว่า Distributed Arithmetic (DA) พิจารณาวงเล็บใหญ่ของสมการที่ (2.3b) คือ

$$Y' = \sum_{k=1}^{K} A_k \cdot b_{kn} \qquad (2.3d)$$

เพราะว่า b_{kn} จะมีค่าเป็น 0 หรือ 1 เท่านั้น ดังนั้นผลลัพธ์สมการ (2.3d) จะมีค่าที่เป็นไปได้ทั้งหมด 2^K ค่า ซึ่งค่าต่าง ๆ เหล่านี้ถูกคำนวณแล้วเก็บไว้ในหน่วยความจำแบบอ่านได้อย่างเดียว (ROM) ข้อมูลบิทของอินพุทถูกใช้เป็นแอดเดรส (Address) ของรอมได้โดยตรง ผลลัพธ์ถูกส่งไปเก็บไว้ในตัวรวมข้อมูล (Accumulator) หลังจากที่มีการส่งข้อมูลอินพุทที่ทำเป็นแอดเดรสจำนวน N รอบ ก็จะได้ผลลัพธ์ Y ออกมา



รูปที่ 2.1 โครงสร้าง DA แบบ 1 BAAT ของสมการที่ (2.1)[3]

ตารางที่ 2.1 ข้อมูลที่บรรจุในรอม[3]

1<= n < N-1						n = 0					
Input Code					32-word	Input Code					32-word
Ts	b1n	b2n	b3n	b4n	Memory Content	Ts	b1n	b2n	b3n	b4n	Memory Content
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	A4 = 0.11	1	0	0	0	1	-A4 = - 0.11
0	0	0	1	0	A3 = 0.95	1	0	0	1	0	-A3 = -0.95
0	0	0	1	1	A3+A4 = 1.06	1	0	0	1	1	-(A3+A4) = -1.06
0	0	1	0	0	A2 = -0.30	1	0	1	0	0	-A2 = +0.30
0	0	1	0	1	A2+A4 = -0.19	1	0	1	0	1	-(A2+A4) = +0.19
0	0	1	1	0	A2+A3 = 0.65	1	0	1	1	0	-(A2+A3) = -0.65
0	0	1	1	1	A2+A3+A4 = 0.75	1	0	1	1	1	-(A2+A3+A4) = -0.75
0	1	0	0	0	A1 = 0.72	1	1	0	0	0	-A1 = -0.72
0	1	0	0	1	A1+A4 = 0.83	1	1	0	0	1	-(A1+A4) = -0.83
0	1	0	1	0	A1+A3 = 1.67	1	1	0	1	0	-(A1+A3) = -1.67
0	1	0	1	1	A1+A3+A4 = 1.78	1	1	0	1	1	-(A1+A3+A4) = -1.78
0	1	1	0	0	A1+A2 = 0.42	1	1	1	0	0	-(A1+A2) = -0.42
0	1	1	0	1	A1+A2+A4 = 0.53	1	1	1	0	1	-(A1+A2+A4) = -0.53
0	1	1	1	0	A1+A2+A3 = 1.37	1	1	1	1	0	-(A1+A2+A3) = -1.37
0	1	1	1	1	A1+A2+A3+A4 = 1.48	1	1	1	1	1	-(A1+A2+A3+A4) = -1.48

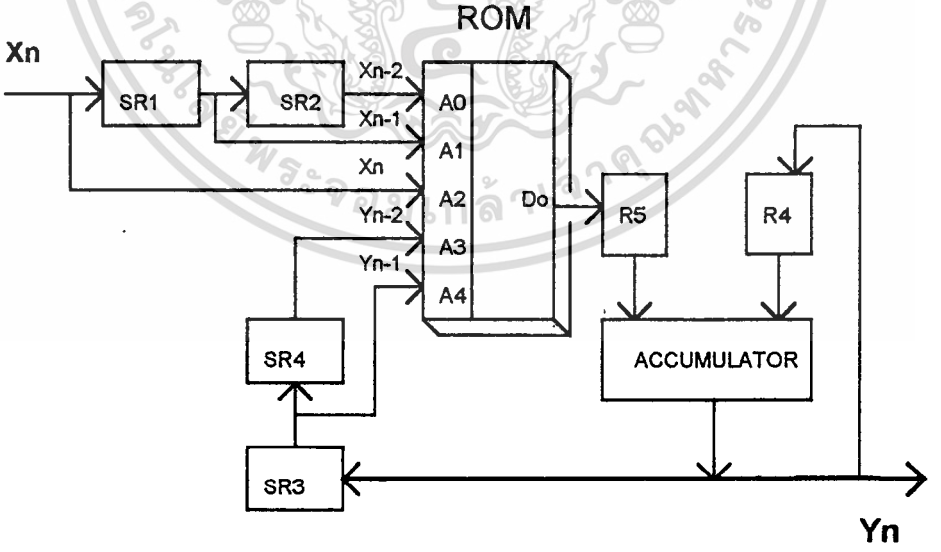
2.3 โครงสร้างของวงจรกรองอันดับสองโดยวิธีการกระจายทางคณิตศาสตร์[3,5]

การกรองความถี่ดิจิตอลแบบไบควอทดราติก (Biquadratic) มีทรานส์เฟอร์ฟังก์ชั่นดังนี้

$$H(z) = \frac{\sum_{k=0}^{2} A_k Z^{-k}}{1 + \sum_{k=1}^{2} B_k Z^{-k}} \qquad .....(2.4)$$

$$\frac{Y(z)}{X(z)} = \frac{A_0 + A_1 Z^{-1} + A_2 Z^{-2}}{1 + B_1 Z^{-1} + B_2 Z^{-2}} \qquad .....(2.5)$$

ซึ่งโพล (Pole) ถูกกำหนดโดย B_1 และ B_2 ส่วนซีโร่ (Zero) A_0, A_1 และ A_2 เป็นตัวกำหนดแสดงในรูปของตัวแปรเวลา $Y_n = [A_0 A_1 A_2 B_1 B_2]^T [X_n X_{n-1} X_{n-2} Y_{n-1} Y_{n-2}]$ ค่าสัมประสิทธิ์ คือ $[A_0\ A_1\ A_2\ B_1\ B_2]^T$ และข้อมูลอินพุทคือ $[X_n X_{n-1} X_{n-2} Y_{n-1} Y_{n-1}]$ ใช้ DA เปลี่ยนเป็นวงจรการกระจายทางคณิตศาสตร์ข้อมูลเข้าแบบอนุกรมครั้งละหนึ่งบิท (bit at a time) ได้ดังรูปที่ 2.2



รูปที่ 2.2 โครงสร้างวงจรกรองสัญญาณเชิงเลขอันดับสองแบบอนุกรม (1BAAT)

$$Y_n = A_0X_n + A_1X_{n-1} + A_2X_{n-2} + B_1Y_{n-1} + B_2Y_{n-2} \qquad .....(2.5a)$$

$$\begin{aligned} Y_n &= X_n(A_0) \\ &+ X_{n-1}(A_1) \\ &+ X_{n-2}(A_2) \\ &+ Y_{n-1}(B_1) \\ &+ Y_{n-2}(B_2) \end{aligned}$$

กระจาย X_n และ Y_n โดยการกระจายทางคณิตศาสตร์

$$\begin{aligned} &(X_n^s + X_n^7 + X_n^6 + X_n^5 + X_n^4 + X_n^3 + X_n^2 + X_n^1 + X_n^0)(A_0) \\ &(X_{n-1}^s + X_{n-1}^7 + X_{n-1}^6 + X_{n-1}^5 + X_{n-1}^4 + X_{n-1}^3 + X_{n-1}^2 + X_{n-1}^1 + X_{n-1}^0)(A_1) \\ &(X_{n-2}^s + X_{n-2}^7 + X_{n-2}^6 + X_{n-2}^5 + X_{n-2}^4 + X_{n-2}^3 + X_{n-2}^2 + X_{n-2}^1 + X_{n-2}^0)(A_2) \\ &(Y_{n-1}^s + Y_{n-1}^7 + Y_{n-1}^6 + Y_{n-1}^5 + Y_{n-1}^4 + Y_{n-1}^3 + Y_{n-1}^2 + Y_{n-1}^1 + Y_{n-1}^0)(B_1) \\ &(Y_{n-2}^s + Y_{n-2}^7 + Y_{n-2}^6 + Y_{n-2}^5 + Y_{n-2}^4 + Y_{n-2}^3 + Y_{n-2}^2 + Y_{n-2}^1 + Y_{n-2}^0)(B_2) \end{aligned} \qquad (2.5b)$$

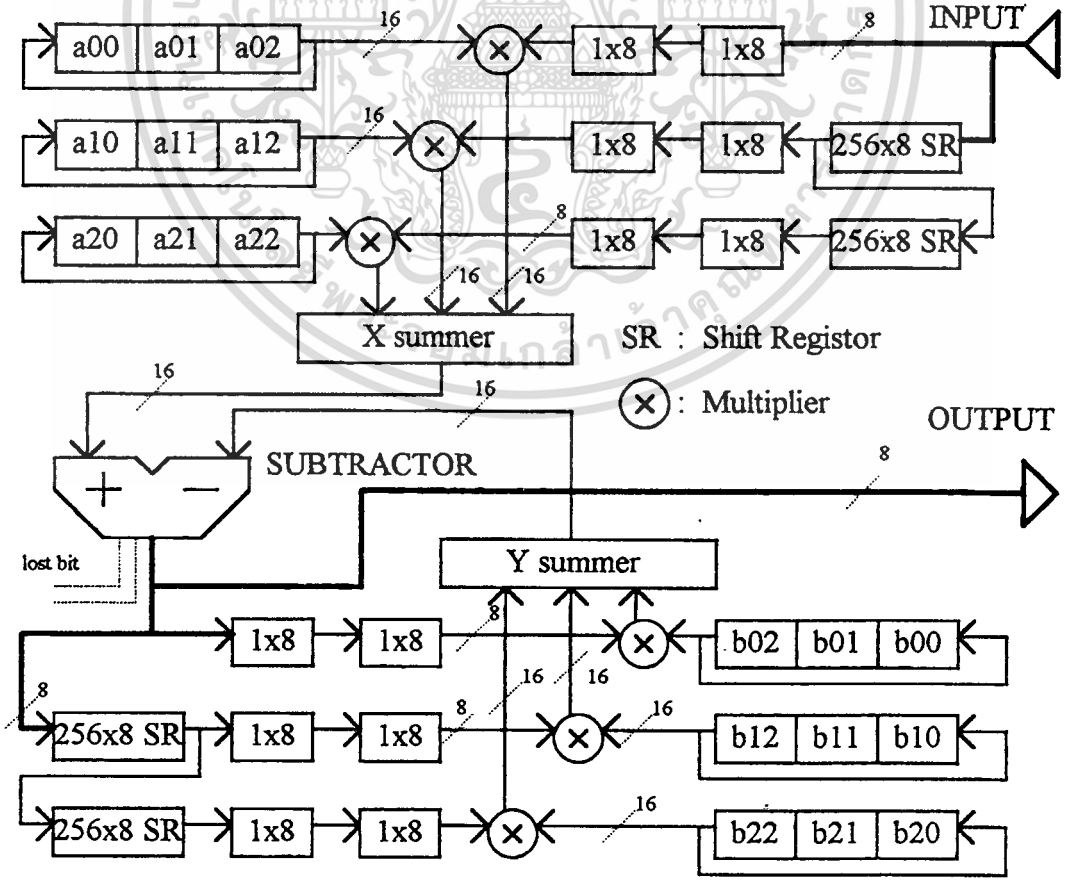
สมการที่ 2.5a และ 2.5b แสดงการใช้วิธีการกระจายทางคณิตศาสตร์กับการกรองสัญญาณ แล้วทำการสลับการบวกใหม่โดยอ้างถึงการกระจายตามสมการที่ (2.3c)

จากรูปที่ 2.2 ใช้ IC ตระกูล TTL. สร้างวงจรกรองอันดับสอง ข้อมูลเข้า X_n จะเข้าแบบอนุกรม เลื่อนเข้าไปยังตัวเลื่อนข้อมูล (shift register) SR1, SR2 ขนาด 8 บิทได้ข้อมูล X_n, X_{n-1}, X_{n-2} และลำดับสัญญาณออก Y_n ถูกส่งไปให้ R4 พร้อมเลื่อนขวาหนึ่งบิทเพื่อหารสอง ก่อนถูกส่งไปให้ ALU (Arithmetic Logic Unit) ซึ่งเป็นตัวรวมข้อมูลขนาด 8 บิท ที่มาจาก R4 กับ R5 ได้ข้อมูลส่งให้ SR3 แล้วทำการเลื่อนไปยัง SR4 สัญญาณออกของ SR3, SR4 ถูกแยกเป็นสัญญาณ Y_{n-1}, Y_{n-2} จากวงจรรูปที่ 2.2 SR1, SR2, SR4 เป็นตัวเลื่อนข้อมูลแบบเข้าอนุกรมออกอนุกรม (SISO) SR3 เป็นเข้าขนานออกอนุกรม (PISO) ส่วน ROM คือ IC TTL เบอร์ SN7488A มีขนาด 32x8 บิท access time เท่ากับ 30 ns (สูงสุด 50 ns) โดย ALU ใช้ 74181 ทำเป็นตัวบวกขนาด 8 บิท ซึ่งใช้เวลาบวกเท่ากับ 30 ns ดังนั้นถ้าใช้สัญญาณนาฬิกา 20MHz (50 ns) คำนวณหาค่า Y_n ต้องการสัญญาณนาฬิกา 9 คาบเวลาที่ใช้เท่ากับ 50ns x 9 (450ns) โดยสรุปแล้ววงจรกรองอันดับสองนี้ทำงานที่ความถี่ 20MHz ใช้ IC TTL 9 ตัวในการสร้าง โดยใช้กำลังงานประมาณ 4W และ Y_n ขนาด 8 บิทมีความเร็ว 20/9=2.2 MHz ถ้าเพิ่มความละเอียดให้มีขนาด 16 บิท ฮาร์ดแวร์จะมีขนาดเป็น 2 เท่าความเร็วประมาณ 1.17MHz

2.4 โครงสร้างของวงจรกรอง 2 มิติโดยวิธีการกระจายทางคณิตศาสตร์

ตัวกรอง 2 มิติเชิงเลข มีประโยชน์ประยุกต์ใช้ โดยเฉพาะเป็นตัวเพิ่มคุณภาพหรือลดสัญญาณรบกวนของสัญญาณในลักษณะ 2 มิติของระบบการประมวลสัญญาณเชิงเลข เช่น สัญญาณภาพรังสีเอ็กซ์ สัญญาณภาพจากดาวเทียม เป็นต้น สัญญาณเหล่านี้จะต้องพิจารณาเป็นลักษณะช่วง ๆ สัญญาณซึ่งส่งมาเป็นลำดับสามารถเขียนในรูปของฟังก์ชั่นได้โดยมีตัวแปรจำนวนเต็ม 2 ตัว เช่น เป็นตัวแสดงลำดับของสัญญาณซึ่งถูกกำหนดให้ค่าทุกค่าเป็นจำนวนเต็ม การประยุกต์ DA กับวงจรกรองสัญญาณสองมิติสมการความแตกต่างเชิงเส้น (linear differential equation) ของวงจรกรองสัญญาณสองมิติแบบป้อนกลับ (two-dimensional second order recursive digital filter)[6] โดยทั่วไปอยู่ในรูปของสมการที่ 2.7[10] และมีโครงสร้างของตัวประมวลผลดังรูปที่ 2.3

$$Y_{m,n} = \sum_{k=0}^{2}\sum_{l=0}^{2} a_{k,l} X_{m-k,n-l} - \sum_{i=0}^{2}\sum_{j=0}^{2} b_{i,j} Y_{m-i,n-j} \qquad \text{.....(2.7)}$$



รูปที่ 2.3 โครงสร้างตัวประมวลสัญญาณ 2 มิติโดยทั่วไป

$X_{m,n}$ และ $Y_{m,n}$ เป็นสัญญาณเข้าและสัญญาณออกตามลำดับ $a_{k,l}$ และ $b_{i,j}$ เป็นค่าสัมประสิทธิ์ของการกรองกำหนดให้สัญญาณทุกค่าอยู่ในช่วง +1, -1 ของเลขคอมพลีเมนท์ของ 2 (2's Complement) ความละเอียดขนาด B บิท รวมบิทเครื่องหมายอีก 1 บิท กระจายเป็นเลขฐานสองโดยสมการที่ (2.8,2.9)

$$X_{m-k,n-l} = \sum_{s=1}^{B-1} X^{s}_{m-k,n-l} 2^{-s} - X^{0}_{m-k,n-l} \qquad (2.8)$$

และ

$$Y_{m-i,n-j} = \sum_{s=1}^{B-1} Y^{s}_{m-i,n-j} 2^{-s} - Y^{0}_{m-i,n-j} \qquad (2.9)$$

แทนค่าสมการที่ (2.8), (2.9) ลงสมการที่ (2.7) จะได้สมการที่ (2.10)[8]

$$Y_{m-i,n-j} = \sum_{s=1}^{B-1}\left(\sum_{k=0}^{2}\sum_{l=0}^{2}(a_{k,l}X^{s}_{m-k,n-l})2^{-s}\right) - \left(\sum_{k=0}^{2}\sum_{l=0}^{2}(a_{k,l}X^{0}_{m-k,n-l}\right)$$
$$- \sum_{s=1}^{B-1}\left(\sum_{i=0}^{2}\sum_{j=0}^{2}(b_{i,j}Y^{s}_{m-i,n-j})2^{-s}\right) + \left(\sum_{i=0}^{2}\sum_{j=0}^{2}(b_{i,j}Y^{0}_{m-i,n-j}\right)$$

$$Y_{m,n} = \sum_{s=1}^{B-1} F_1^{s}(\bullet)2^{-s} - F_1^{0} - \sum_{s=1}^{B-1} F_2^{s}(\bullet)2^{-s} + F_2^{0}(\bullet) \qquad (2.10)$$

การประมวลข้อมูลภาพเป็นการทำให้ข้อมูลภาพ หรือภาพใหม่ออกมาตามวัตถุประสงค์ต่าง ๆ ไม่ว่าจะเป็นการตกแต่งภาพให้ชัดขึ้น การเน้นขอบของภาพให้คมหรืออาจจะเป็นการตรวจหาขอบภาพ เป็นต้น ซึ่งวิธีการต่าง ๆ เหล่านี้มีวิธีการแบบเดียวทั้งสิ้นขึ้นอยู่กับว่าจะกำหนดอิมพัลส์เรสพอนส์ (impulse response) ให้มีค่าเป็นเท่าใดขนาดเท่าไร โดยใช้แผ่นข้อมูล (mask) วางทาบไปบนข้อมูลภาพ แล้วเคลื่อนที่ไปในตำแหน่งต่าง ๆ ของภาพทีละจุด ในแต่ละครั้งข้อมูลทั้งหมดที่อยู่ในบริเวณนี้ที่ตำแหน่งเดียวกันจะถูกนำมาคูณกันผลลัพธ์ที่ได้จะนำมารวมกันเพื่อหาขนาดแล้วเก็บเป็นข้อมูลใหม่ แผ่นข้อมูลจะมีค่าไม่เหมือนกันแล้วแต่ความต้องการดังตัวอย่างต่อไปนี้

1/16

1	2	1
2	4	2
1	2	1

(smooth filter)

0	-1	0
-1	5	-1
0	-1	0

(high-pass filter)

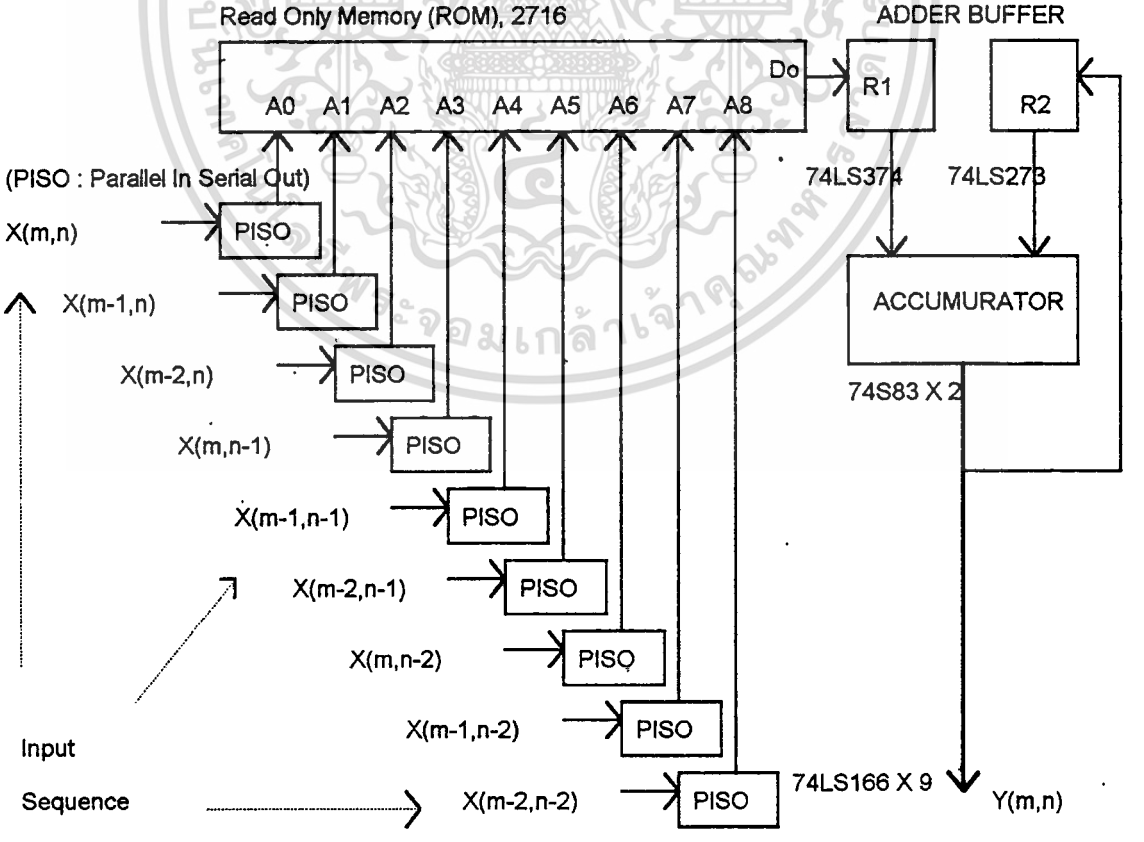
-1	0	1
-1	0	1
-1	0	1

(vertical edge detector)

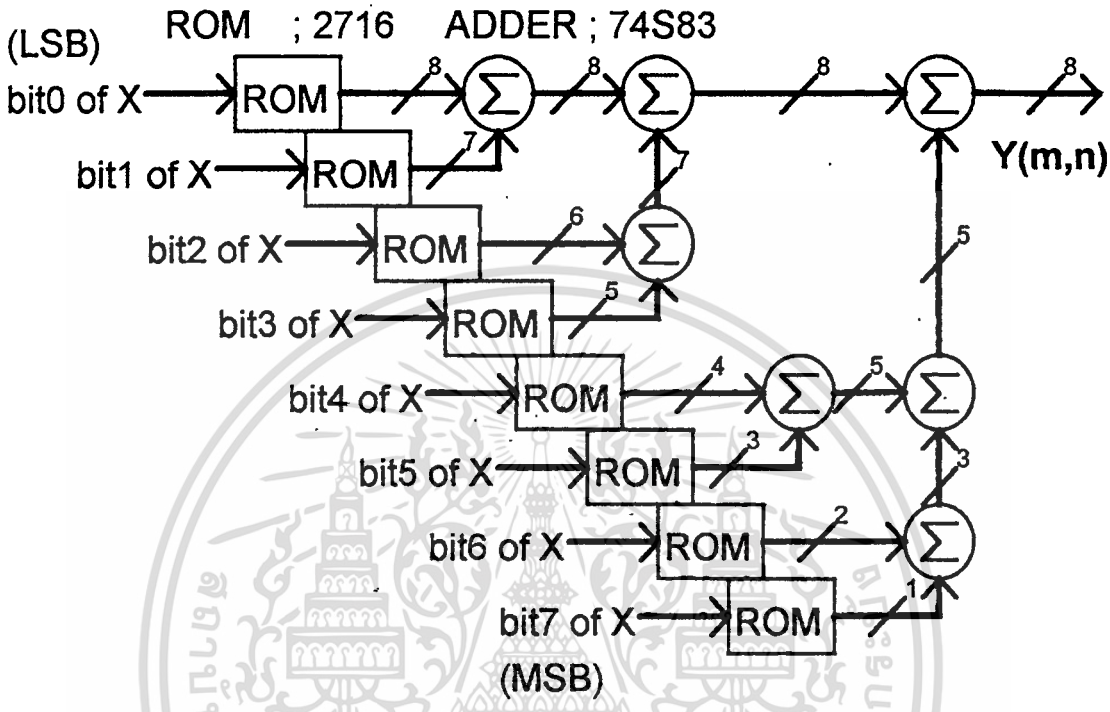
1	1	1
0	0	0
-1	-1	-1

(horizontal edge detector)

การประมวลผลข้อมูลภาพโดยวิธีการกระจายทางคณิตศาสตร์ใช้วิธีนำลำดับสัญญาณ X(m,n) มาจัดลำดับใหม่ดังที่กล่าวมาแล้ว ลำดับสัญญาณทั้ง 9 ค่า ส่งมาหน่วยประมวลผล โดยการเปิดตารางข้อมูล (look-up table) ซึ่งเก็บแผ่นข้อมูลแบบต่าง ๆ ไม่ว่าจะเป็น low-pass, high-pass หรือ edge detector ลำดับข้อมูลที่ออกมาจากหน่วยความจำจะถูกส่งตัวรวมข้อมูล เพื่อให้ได้ลำดับข้อมูลออก Y(m,n) ต่อไป จากรูปที่ 2.4 แสดงถึงวิธีการส่งลำดับ X(m,n) แบบอนุกรม เมื่อส่งครบ 8 บิทแล้วถึงจะได้ Y(m,n) 1 ไบท์ การส่งลำดับ X(m,n) นอกจากจะเป็นแบบอนุกรมแล้ว ยังสามารถส่งแบบขนานได้อีกด้วย ทั้งนี้ก็เพื่อต้องการให้ความเร็วในการประมวลผลสูงขึ้น จากรูปที่ 2.5 จะแยกบิทแต่ละบิทของ สัญญาณเข้าทั้ง 9 แล้วส่งเข้าหน่วยความจำ แต่ละตัวข้อมูลที่ออกจากหน่วยความจำจะถูกส่งไปรวมกันทั้งหมดแล้วได้ลำดับข้อมูล Y(m,n) ออกมา จะเห็นว่าการใช้วิธีสัญญาณเข้าแบบขนานนี้จะประมวลผลได้เร็วกว่าแบบอนุกรมถึง 8 เท่า เมื่อใช้ความถี่ของสัญญาณนาฬิกาเท่ากัน การทำงานจะไม่ซับซ้อนเหมือนกับแบบอนุกรม แต่ว่าจำเป็นจะต้องใช้หน่วยความจำถึง 8 ชุดตัวรวมข้อมูลอีก 7 ชุด ดังนั้นแบบขนานจะสิ้นเปลืองทั้งอุปกรณ์และกำลังงานมากกว่า จะให้ผลดีด้านความเร็ว



รูปที่ 2.4 โครงสร้างวงจรกรองสัญญาณเชิงเลขสองมิติแบบ 1 BAAT (Bit At A Time)[3]



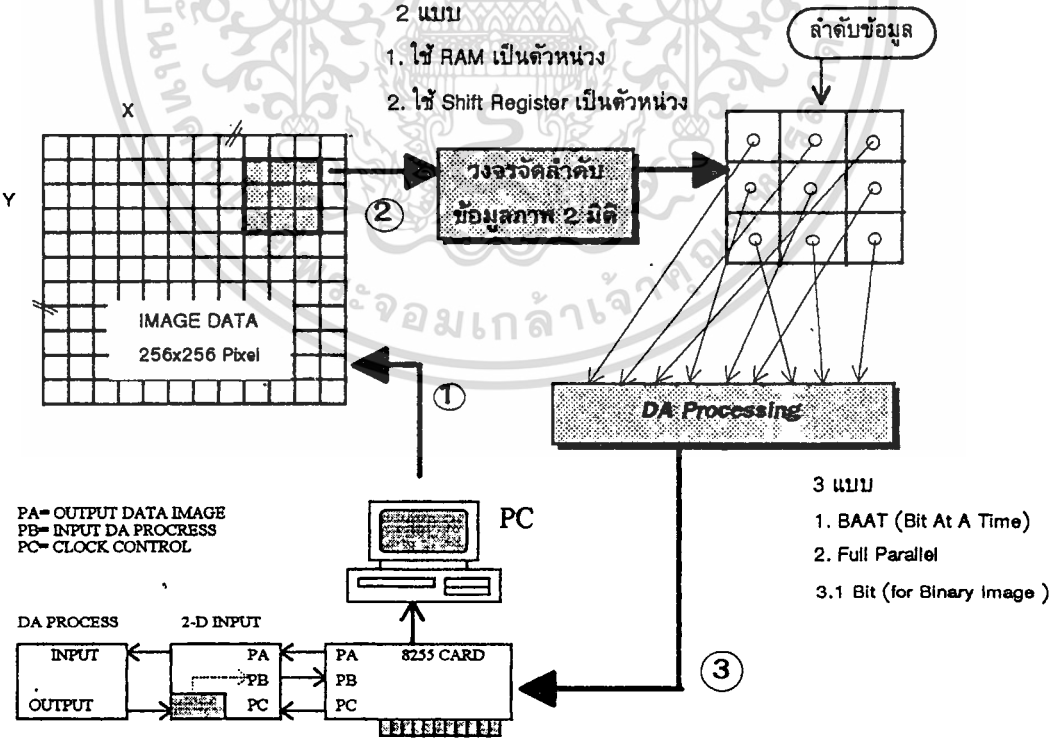
รูปที่ 2.5 โครงสร้างวงจรกรองสัญญาณ 2 มิติ แบบขนาน (DA full parallel)

บทที่ 3
การออกแบบ

3.1 การติดต่อตัวประมวลผลกับคอมพิวเตอร์

ในการวิจัยการกระจายทางคณิตศาสตร์ใช้คอมพิวเตอร์ส่วนบุคคล (Personal Computer) เป็นตัวจำลองการทำงานของตัวรับข้อมูลภาพ เช่น กล้องวีดีโอ สแกนเนอร์ เป็นต้น โดยคอมพิวเตอร์จะทำหน้าที่ส่งข้อมูลภาพขนาด 256 x 256 ไปยังวงจรประมวลผลภาพ โดยผ่านอุปกรณ์เชื่อมต่อ (Interface Card) ที่ทำหน้าที่ส่งสัญญาณข้อมูลขนาด 8 บิท และสัญญาณควบคุมต่าง ๆ เช่น ควบคุมวงจรหน่วงสัญญาณ สัญญาณนาฬิกา เป็นต้น ซึ่งจะอธิบายโครงสร้าง โดยทั่วไปของอุปกรณ์ดังต่อไปนี้

จากรูปที่ 3.1 แสดงอุปกรณ์และขั้นตอนหลักในการประมวลข้อมูลภาพโดย คอมพิวเตอร์จะส่งลำดับข้อมูลภาพ 256 ระดับสีเทาขนาด 256x256 จุดผ่านไอโอพอท(I/O Port) 8255 โดยกำหนดให้พอร์ทเอ ใช้ทำหน้าที่เป็นเอาท์พุทขนาด 8 บิท ทำหน้าที่ส่งข้อมูลภาพ พอร์ทบี ใช้ทำหน้าที่เป็นอินพุทขนาด 8 บิท รับข้อมูลที่ผ่านวงจรจัดลำดับข้อมูลภาพเพื่อสร้างลำดับข้อมูลให้เป็น 2 มิติแล้วส่งข้อมูลดังกล่าวไปตัวประมวลผลโดยวิธีการกระจายทางคณิตศาสตร์ได้เป็นข้อมูลภาพใหม่ พอร์ทซี ใช้ทำหน้าที่เป็นเอาท์พุทส่งสัญญาณไปควบคุมอุปกรณ์ต่าง ๆ บนวงจรการกระจายทางคณิตศาสตร์ ซึ่งสัญญาณควบคุมดังกล่าวนี้จะกล่าวรายละเอียดในเรื่องของการออกแบบวงจรหน่วงสัญญาณและการทำงานของ การกระจายทางคณิตศาสตร์แบบ 1 บิท ต่อไป

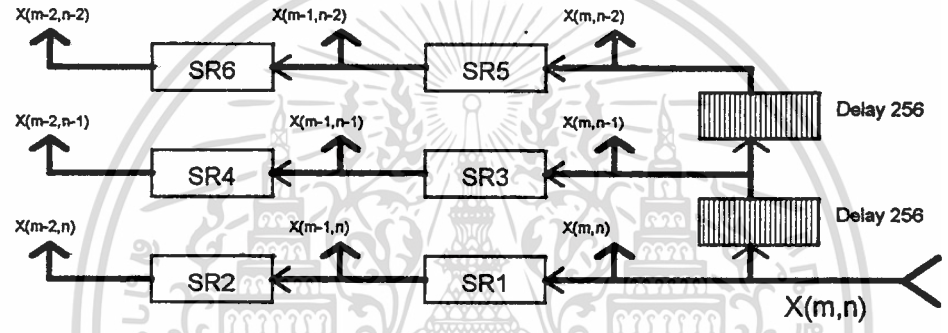


รูปที่ 3.1 ขั้นตอนการส่งข้อมูลเพื่อไปประมวลภาพโดย DA

ส่วนโปรแกรมควบคุมการทำงาน ส่งข้อมูลภาพ ส่งสัญญาณควบคุม ดูรายละเอียดได้จากหัวข้อโปรแกรมที่ใช้ในการทดลองท้ายบท และรายละเอียดของวงจรแสดงไว้ที่บทวงจรที่ใช้ในการทดลอง

3.2 การออกแบบวงจรหน่วงเวลาข้อมูลภาพโดยใช้แรม

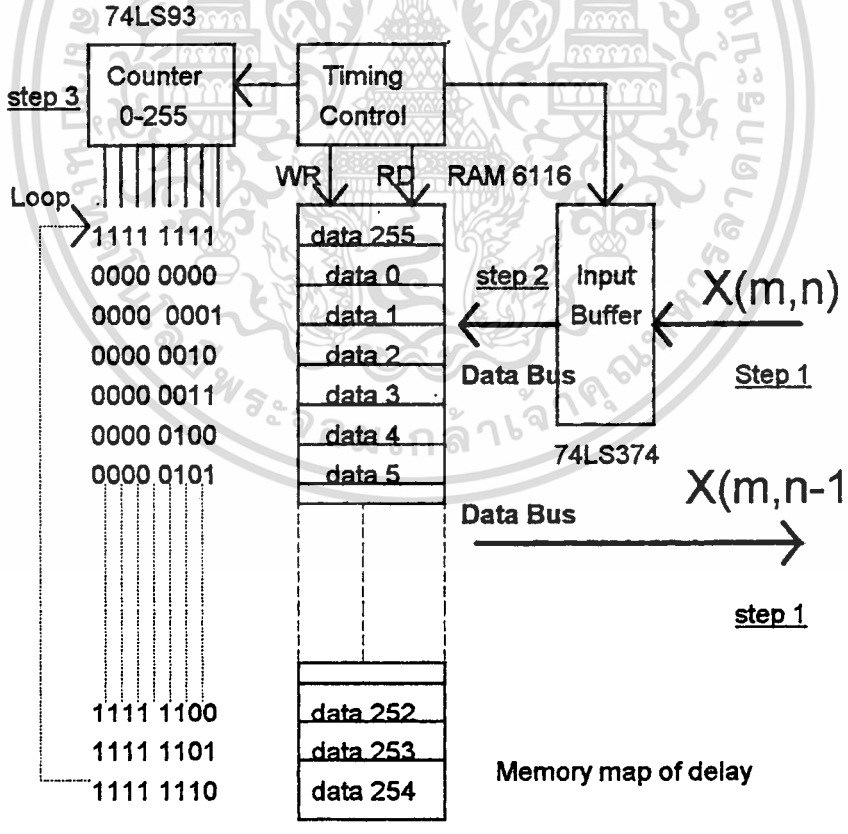
ลำดับข้อมูลภาพที่ส่งเข้ามานี้ จำเป็นจะต้องจัดลำดับสัญญาณข้อมูลเข้าให้ถูกต้องก่อนนำไปประมวลผลนั่นคือต้องสร้างลำดับข้อมูล X(m,n), X(m-1,n), X(m-2,n), X(m,n-1), X(m-1,n-1), X(m-2,n-1), X(m,n-2), X(m-1,n-2), X(m-2,n-2) ส่งไปคูณกับแผ่นข้อมูล (mask) ขนาด 3x3 ที่มีคุณสมบัติต่าง ๆ เช่น highpass filter, lowpass filter , edge detector เป็นต้น กำหนดให้ n เป็นค่าชี้ข้อมูลแถว (row) ส่วน m ชี้ข้อมูลหลัก (column) สมมุติว่าลำดับข้อมูลภาพมีขนาด 256x256 ตัวหน่วงข้อมูลต้องมีค่าเท่ากับ 256x2 ดังนั้นสามารถใช้ตัวเลื่อนข้อมูล (shift register) ต่ออนุกรมกับจำนวน 512 ตัว ใช้ตัวเลื่อนข้อมูลภาพตามแนวนอนอีก 6 ตัว ถ้าเราใช้วงจรรวมชนิด TTL เบอร์ 74S374 นำมาต่อตามโครงสร้างดังรูปที่ 3.2



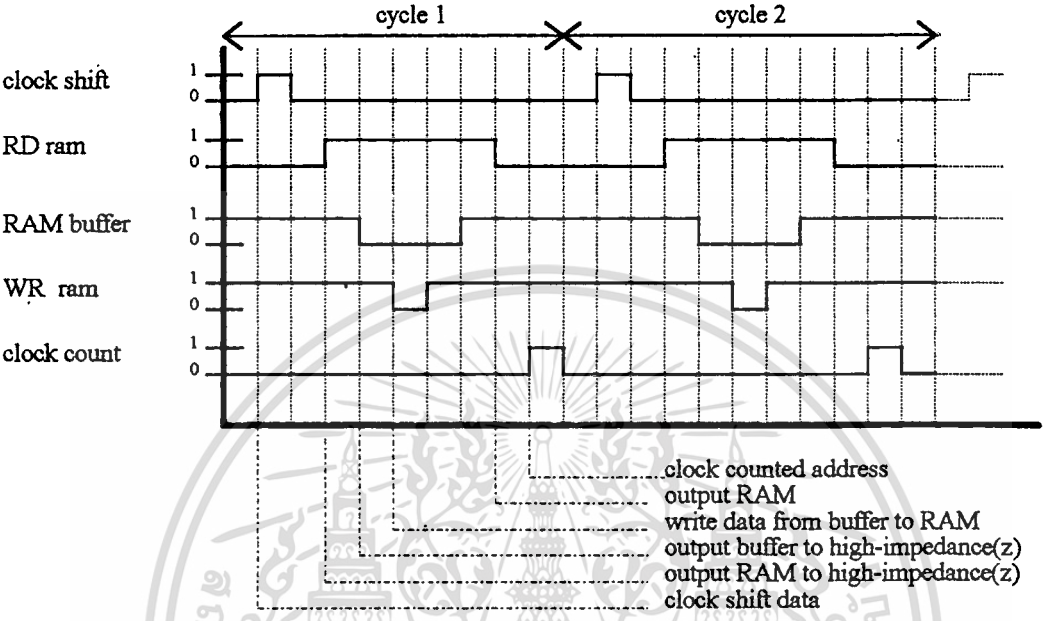
รูปที่ 3.2 โครงสร้างการจัดลำดับข้อมูล 2 มิติขนาด 3x3 สำหรับภาพ 256x256 จุด

จากรูปที่ 3.2 วงจรมีขนาดใหญ่ เพราะว่าใช้วงจรรวมอย่างน้อย 518 ตัว แถมยังมีความต้องการกำลัง (power consumtion) สูงด้วย การแก้ทางหนึ่งของการลดขนาดและกำลังงานของวงจรก็คือต้องลดขนาดของวงจรหน่วงข้อมูลที่ทำหน้าที่จัดลำดับข้อมูลแนวตั้งที่มีขนาด 256 หน่วยจำนวน 2 ชุด โดยการใช้หน่วยความจำชนิดอ่านและเขียนได้ (RAM) เป็นตัวเลื่อนข้อมูล ที่ถูกควบคุมแอดเดรส (address) โดยวงจรนับ (counter) ขนาด 8 บิท ซึ่งจะแสดงการทำงานของชุดหน่วงข้อมูลที่มีวงจรนับและหน่วยความจำดังต่อไปนี้ จากรูปที่ 3.2 และ 3.3 สมมุติว่าวงจรนับมีค่าเริ่มต้นที่ 0000 0000 ข้อมูลเข้า X(m,n) จังหวะแรกในการเลื่อนข้อมูล X(m,n) เลื่อนผ่าน SR1 จะได้สัญญาณ X(m-1,n) ในขณะเดียวกัน X(m,n) ส่งเข้าไปเก็บใน input buffer ด้วยขณะที่จะอ่านข้อมูลจากหน่วยความจำตำแหน่งที่ 0000 0000 ออกไปให้ SR3 ทำการเลื่อนให้ได้สัญญาณ X(m-1,n-1) ก่อนที่ X(m,n) ต่อไปจะถูกส่งเข้ามา timing Control จะควบคุมให้ input buffer เขียนข้อมูล X(m,n) ชุดแรกลงหน่วยความจำตำแหน่งที่ 0000 0000 ต่อจากนั้นวงจรนับจะเพิ่มขึ้นเป็น 0000 0001 แล้วแรมเขียนข้อมูลส่งไป output เพื่อเตรียมรอเลื่อนข้อมูล X(m,n-1) เป็นชุดต่อไป จังหวะที่สองในการเลื่อนข้อมูล X(m,n) ชุดที่สองส่งมาที่อินพุทของ SR1 ขณะเดียวกับ X(m,n) ชุดแรกก็เลื่อนผ่าน SR1 เพื่อกำเนิดสัญญาณ X(m-1,n) ส่วน X(m-1,n) ชุดแรกจะผ่าน SR2 ได้สัญญาณ X(m-1,n) ด้วย วงจรหน่วงสัญญาณก็จะทำงานเหมือนเดิมโดยอ่านข้อมูลออกและเขียนข้อมูลเข้าแล้ววงจรนับเพิ่มขึ้นเป็น 0000 0010 กระทำซ้ำอย่างนี้ไปตลอดจนวงจรนับมีค่าถึง 255 ใหม่อีกหนึ่งรอบข้อมูลก็สามารถถูกหน่วงไปได้ 256 ข้อมูล

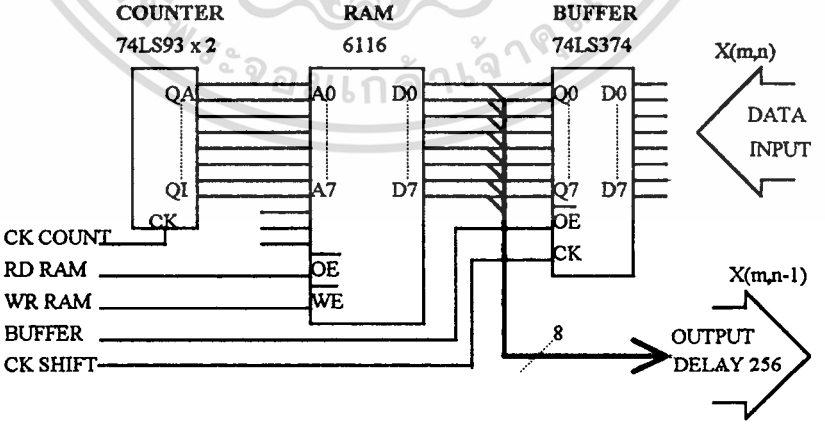
จากการสร้างหน่วยความจำ และวงจรนับเข้ามาแทนตัวหน่วงนี้ ทำให้วงจรจัดลำดับข้อมูลมีขนาดเล็กลงมากทั้งยังประหยัดกำลังงานอีกด้วย แต่ว่าจำเป็นต้องสร้างสัญญาณควบคุมหน่วยความจำและวงจรนับให้มีการทำงานได้เหมือนตัวหน่วงก็จะทำให้ การทำงานของวงจรทั้งหมดช้าลงไปด้วย เพราะต้องมีการควบคุมเขียนอ่านหน่วยความจำและวงจรนับด้วย แต่ในขณะที่ใช้ตัวเลื่อนต่ออนุกรมกัน 512 ตัวจะทำงานเร็วกว่ามากเพราะใช้สัญญาณนาฬิกา 1 ลูกต่อลำดับสัญญาณข้อมูล 1 ชุด นอกจากนี้ การทำงานของวงจรนับไม่จำเป็นต้องเริ่มจาก 0 ก็ได้ เพราะว่าเมื่อเริ่มต้นที่ตำแหน่งใดของหน่วยความจำจะถือว่าตำแหน่งนั้นเป็นตำแหน่งเริ่มต้นและวงจรนับจะนับเพิ่มไป 256 ครั้งก็จะกลับมาที่ตำแหน่งเดิม ดังนั้นจึงไม่จำเป็นต้องมีสัญญาณไปรีเซ็ท (reset) วงจรนับในช่วงเริ่มต้นของการนับก็ได้ ส่วนลำดับสัญญาณออกทั้ง 9 จะถูกส่งไปวงจรประมวลผลอีกที วงจรประมวลผลก็ใช้วิธีการของการกระจายทางคณิตศาสตร์ซึ่งจะกล่าวถึงการออกแบบ และการทำงานในหัวข้อต่อไป



รูปที่ 3.3 แสดงตำแหน่งหน่วยความจำที่นำมาใช้เป็นตัวหน่วงขนาด 256 ข้อมูล



รูปที่ 3.4 Timing Diagram ของสัญญาณควบคุมวงจรหน่วงข้อมูลโดยใช้แรม (RAM)

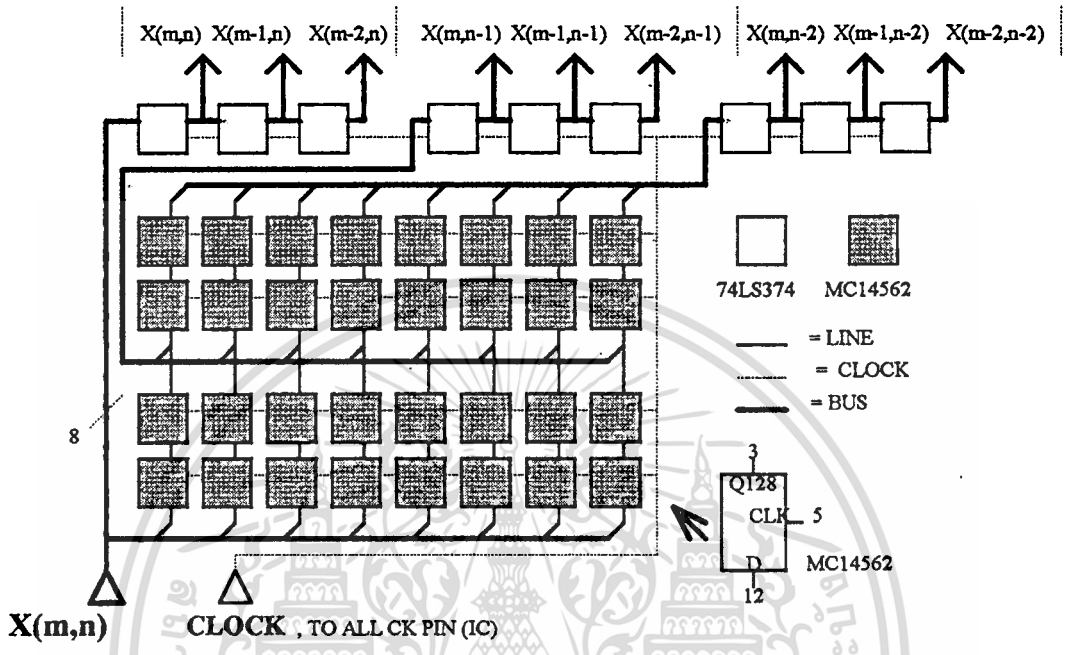


รูปที่ 3.5 วงจรหน่วงข้อมูลโดยใช้แรม (RAM)

จากรูปที่ 3.4 แสดง Timing Diagram ที่ไปควบคุมวงจรรูปที่ 3.5 ของสัญญาณควบคุมวงจรหน่วงข้อมูลโดยใช้แรม (RAM) โดยมี Clock shift ทำหน้าที่ส่งสัญญาณเลื่อนข้อมูลภาพเข้าไปยังตัวเลื่อนข้อมูล (Shift Register) ในขณะเดียวกันข้อมูลที่อยู่ในแรมก็จะถูกส่งออกไปด้วย เพราะว่าถูกควบคุมโดยสัญญาณ RD RAM ต่อจากนั้น RAM buffer ซึ่งทำหน้าที่ส่งสัญญาณไปควบคุมตัวพักข้อมูลให้ส่งข้อมูลที่อยู่ในตัวพักข้อมูลไปเขียนลงแรมโดยใช้สัญญาณ WR RAM ข้อมูลก็จะถูกเลื่อนจากตัวพักข้อมูลไปเก็บไว้ในแรมเรียบร้อย เพื่อให้ข้อมูลดังกล่าวเลื่อนไปตามแอดเดรสต่าง ๆ จำนวน 256 ดังนั้นข้อมูลจึงถูกหน่วงไปได้ 256 ข้อมูลเช่นกัน โดยสัญญาณควบคุม Clock count จะไปส่งให้วงจรนับ 74LS93 ซึ่งต่อร่วมกันทำงาน 2 ตัว เป็นวงจรนับจำนวน 8 บิท ในการที่ข้อมูลจะหน่วงไปได้ 256 นั้นจะต้องใช้ cycle ทั้งหมด 256 cycle แต่รูปที่ 3.4 นั้น แสดงให้ดูเพียง 2 cycle เท่านั้น ส่วนความเร็วในการทำงานจะขึ้นอยู่กับ ความเร็วของคอมพิวเตอร์ที่ส่งมาจากอุปกรณ์เชื่อมต่อที่กล่าวในหัวข้อที่ 3.1 และโปรแกรมควบคุมการทำงาน งานวิจัยนี้ใช้โปรแกรมคอมพิวเตอร์ภาษาซีเป็นตัวส่งข้อมูลและสัญญาณควบคุม จึงทำให้ความเร็วของการหน่วงข้อมูลภาพโดยใช้แรมเป็นไปค่อนข้างช้า ผู้ทำวิจัยคิดว่าถ้าใช้โปรแกรมคอมพิวเตอร์ภาษาแอสเซมบลี้ (Assembly Language) ก็จะทำให้ความเร็วในการทำงานดีขึ้น แต่ทั้งนี้เพื่อความสะดวกในการพัฒนาจึงจำเป็นต้องใช้โปรแกรมคอมพิวเตอร์ภาษาซี[9]

3.3 การออกแบบวงจรหน่วงข้อมูลภาพความเร็วสูง

โดยปกติแล้ววงจรหน่วงข้อมูลภาพโดยใช้แรมจะทำงานช้า แต่เรามีวิธีการทำให้ความเร็วในการเลื่อนเร็วขึ้นโดยใช้ตัวเลื่อนข้อมูลมาต่ออนุกรมกันตามโครงสร้างในรูปที่ 3.2 สมมุติว่าใช้วงจรรวมชนิด TTL เบอร์ 74LS374 มาต่ออนุกรม จะต้องใช้ไอซีทั้งหมด 256 + 256 + 6 = 518 ตัว ซึ่งนอกจากจะใช้กระแสประมาณ 14 แอมป์ (27 mA x 518) แล้วยังต้องใช้พื้นที่ไม่ต่ำกว่า 200 ตารางนิ้ว (ประมาณ 1.2 ตารางฟุต) จะเห็นได้ว่าสิ้นเปลืองทั้งกำลังงานและพื้นที่ ผู้ทำวิจัยขอเสนอวงจรรวมชนิด CMOS เบอร์ MC14562B ทำหน้าที่เป็นตัวเลื่อนขนาด 128 บิทนำมาต่อขนานกัน 8 ตัวต่อหนึ่งชุดแล้วนำมาต่ออนุกรมกันสองชุดก็จะได้ตัวหน่วงข้อมูลขนาด 256 ไบท์ จากรูปที่ 3.6 แสดงโครงสร้างของตัวหน่วงข้อมูลภาพและการจัดลำดับข้อมูลภาพก่อนถูกส่งไปวงจรประมาลผลโดยวิธีการกระจายทางคณิตศาสตร์ วงจรจัดลำดับข้อมูลภาพโดยใช้ MC14562B ที่ใช้ในการทดลองนี้ จะทำงานได้เร็วกว่าการใช้แรมเป็นตัวหน่วงข้อมูลมาก เพราะว่าใช้สัญญาณนาฬิกาเพียง 1 ไซเคิล (cycle) เท่านั้นเมื่อเทียบกับการใช้แรมต้องใช้สัญญาณนาฬิกาถึง 10 ไซเคิล ถึงแม้ว่า MC14562B จะใช้สัญญาณนาฬิกา 1 ไซเคิลแต่ก็มีข้อจำกัดในเรื่องความถี่สูงซึ่งทำงานได้ความเร็วสูงสุดประมาณ 1.5 MHz เมื่อเทียบกับ 74LS374 ทำงานที่ความถี่สูงสุด 50 MHz[13]แต่มีข้อเสียในเรื่องของขนาดและกำลังเท่านั้น ถ้าเราสามารถพัฒนาวงจรจัดลำดับข้อมูลในรูปของวงจรรวมขนาดใหญ่มาก (VLSI) ก็จะทำให้การทำงานมีประสิทธิภาพสูง



รูปที่ 3.6 โครงสร้างตัวจัดลำดับข้อมูลภาพโดยใช้ตัวเลื่อนข้อมูล MC14562B

3.3 การออกแบบข้อมูลลงหน่วยความจำ

การออกแบบข้อมูลที่จะเก็บไว้ในหน่วยความจำเป็นสิ่งที่ไม่ยากนักสำหรับการประมวลข้อมูลภาพ เนื่องมาจากว่าไม่ต้องนำความถี่ในการสุ่มสัญญาณมาเป็นเงื่อนไขเพราะว่าข้อมูลภาพในที่นี้เป็นสัญญาณที่อยู่ในรูปแบบสัญญาณเชิงเลขอยู่แล้ว ไม่เหมือนกับการออกแบบข้อมูลจากค่าสัมประสิทธิ์ของวงจรกรองสัญญาณ 1 มิติซึ่งต้องเปลี่ยนสัญญาณเชิงอุปมานก่อนการประมวลผลให้เป็นสัญญาณเชิงเลข ดังนั้นต้องคำนึงถึงจังหวะการทำงานของวงจรด้วย เมื่อเราเข้าใจหลักการของการกระจายทางคณิตศาสตร์ก็จะทำให้ทราบว่าต้องใช้ขนาดหน่วยความจำเท่าใดกรณีของการประมวลสัญญาณ 2 มิติที่กล่าวถึงนี้มีลำดับข้อมูลเข้า 9 ชุดดังนั้นต้องใช้รอมขนาด 512xB ให้ B แทนจำนวนบิท จากนั้นนำเอาค่าสัมประสิทธิ์จากแผ่นข้อมูลมารวมกันแบบไม่ซ้ำได้ทั้งหมด 512 ค่า (2^9) จากตารางที่ 3.1 สมมุติให้มีค่า A, B, C, D, E, F, G, H, I เป็นค่าที่อยู่ในแผ่นข้อมูลขนาด 3x3 นำค่าทั้ง 9 ค่านี้ มาบวกกันโดยใช้ระบบเลขฐานสองเป็นตัวกำหนดการบวกได้ผลลัพธ์ Y คือค่าที่ต้องบรรจุลงในรอม ถ้าค่า Y เป็นลบต้องเปลี่ยนเป็นเลขคอมพลีเมนท์ของสอง (2's Complement) ก่อน แสดงการสร้างข้อมูลในตารางที่ 3.1

ตารางที่ 3.1 แสดงการสร้างข้อมูลที่เก็บไว้ในรอม

Binary Control Sumation									Output Data for ROM
R	S	T	U	V	W	X	Y	Z	Q=R.A+S.B+T.C+U.D+V.E+W.F+X.G+Y.H+Z.I
0	0	0	0	0	0	0	0	0	Q=0
0	0	0	0	0	0	0	0	1	Q=I
0	0	0	0	0	0	0	1	0	Q=H
0	0	0	0	0	0	0	1	1	Q=H+I
0	0	0	0	0	0	1	0	0	Q=G
:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
0	0	1	0	1	0	1	0	1	Q=C+E+G+I
0	0	1	0	1	0	1	1	0	Q=C+E+G+H
:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
1	1	1	1	1	1	1	0	0	Q=A+B+C+D+E+F+G
1	1	1	1	1	1	1	0	1	Q=A+B+C+D+E+F+G+I
1	1	1	1	1	1	1	1	0	Q=A+B+C+D+E+F+G+H
1	1	1	1	1	1	1	1	1	Q=A+B+C+D+E+F+G+H+I

MASK

โปรแกรมสร้างข้อมูลใส่ในรอมสำหรับวงจรการกระจายทางคณิตศาสตร์

```
10 REM create data for ROM
20 CLEAR
40 INPUT "Filename data to ROM :",FIL$
45 OPEN FIL$ FOR OUTPUT AS #1
110 INPUT "Data for position   X1 : ",     A
120 INPUT "Data for position   X2 : ",     B
130 INPUT "Data for position   X3 : ",     C
140 INPUT "Data for position   X4 : ",     D
150 INPUT "Data for position   X5 : ",     E
160 INPUT "Data for position   X6 : ",     F
170 INPUT "Data for position   X7 : ",     G
180 INPUT "Data for position   X8 : ",     H
190 INPUT "Data for position   X9 : ",     I
210 FOR R = 0 TO 1
220 FOR S = 0 TO 1
230 FOR T = 0 TO 1
```

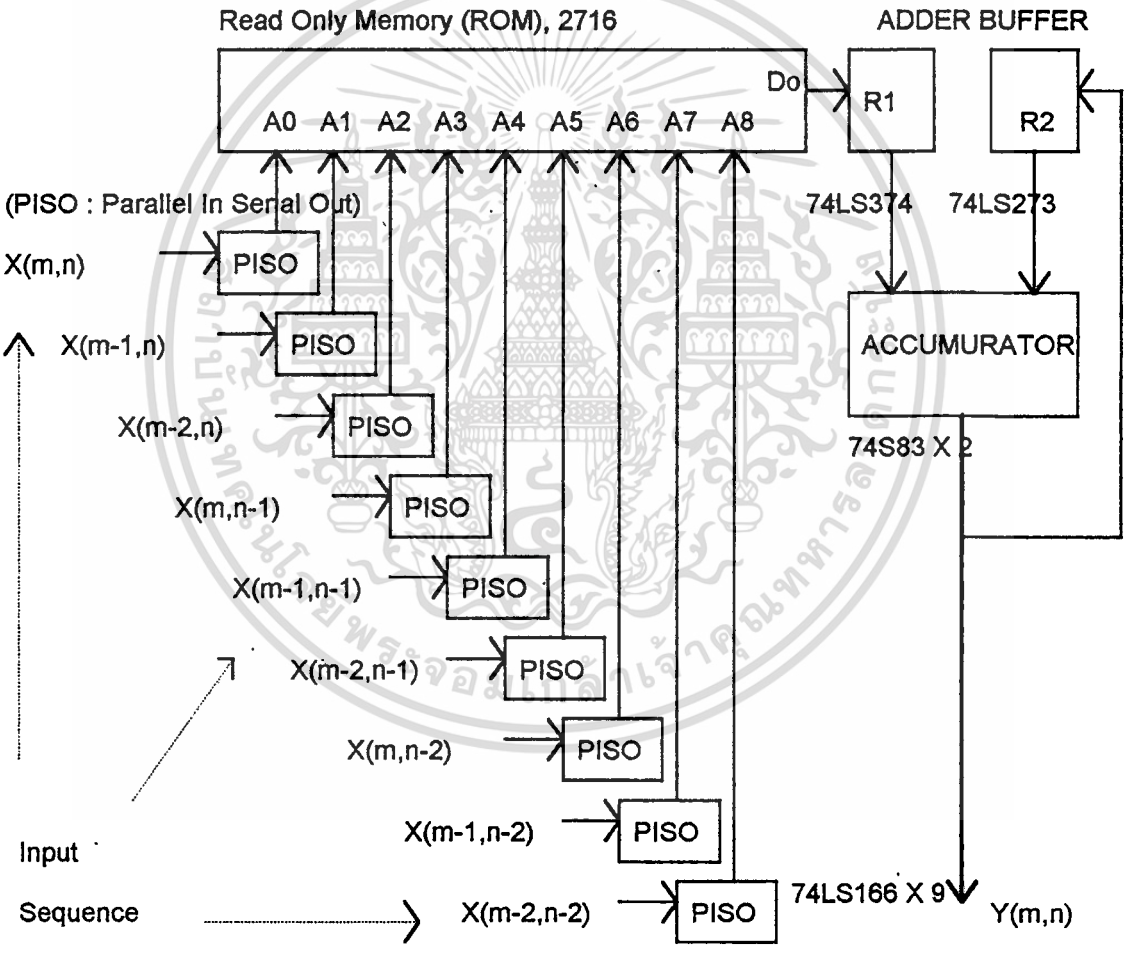
โปรแกรมสร้างข้อมูลใส่ในรอมสำหรับวงจรการกระจายทางคณิตศาสตร์ (ต่อ)

```
240 FOR U = 0 TO 1
250 FOR V = 0 TO 1
260 FOR W = 0 TO 1
270 FOR X  = 0 TO 1
280 FOR Y  = 0 TO 1
290 FOR Z  = 0 TO 1
300 Y1 = (A*R) + (B*S) + (C*T)
310 Y2 = (D*U) + (E*V) + (F*W)
320 Y3 = (G*X) + (H*Y) + (I*Z)
330 YT = Y1 + Y2 + Y3
335 GOSUB 700
360 PRINT#1, CHR$(YT);
410 NEXT          R
420 NEXT          S
430 NEXT          T
440 NEXT          U
450 NEXT          V
460 NEXT          W
470 NEXT          X
480 NEXT          Y
490 NEXT          Z
500 CLOSE#1
600 END
700 REM sub  2'S Complement
705 IF YT > -1 THEN RETURN
710 YT = 256 + YT
750 RETURN
800 END
```

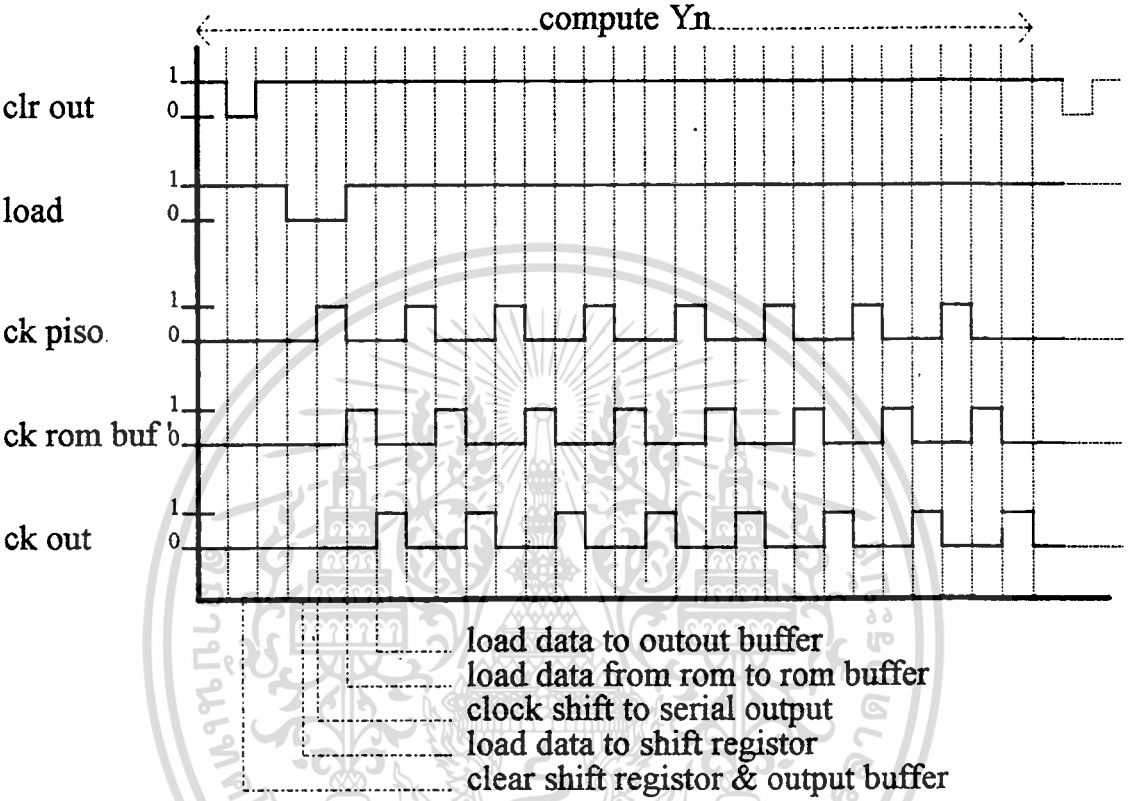
จากโปรแกรมข้างบนเป็นโปรแกรมสร้างข้อมูลที่จะเก็บเข้ารอม โดยขั้นแรกจะรับข้อมูลที่เป็นสัมประสิทธิ์ของตัวกรองกำหนดให้เป็นค่า A, B, C, D, E, F, G, H, I ตามลำดับต่อจากนั้นจะทำการบวกแบบไม่ซ้ำกันโดยใช้วิธีการนับเลขฐานสองในขณะที่จะตรวจสอบค่าลบด้วย ถ้าค่าที่คำนวณออกมาเป็นลบจะทำการส่งค่านั้นไปยังโปรแกรมย่อยทำให้เป็นเลขคอมพลีเมนท์ เมื่อทำการบวกแบบไม่ซ้ำกันครบ 512 ค่า แล้วโปรแกรมจะเก็บค่าที่คำนวณได้ทั้งหมดลงแฟ้มข้อมูล (file) ที่กำหนดแล้วเขียนข้อมูลลงรอมโดยเครื่องเขียนรอม แต่งานวิจัยนี้ใช้แผงวงจรคอมพิวเตอร์วงจรเดี่ยว (Single Board) รุ่น CP32 ร่วมกับ ET-BASIC 180 ของบริษัท ETT ทำการทดลอง เพราะว่าพัฒนาโปรแกรมจำลองการทำงานและส่งข้อมูลให้วงจรการกระจายทางคณิตศาสตร์ได้คล่องตัวกว่า

3.4 การออกแบบวงจรการกระจายทางคณิตศาสตร์

การประมวลผลข้อมูลภาพโดยวิธีการกระจายทางคณิตศาสตร์ใช้วิธีนำลำดับสัญญาณ X(m,n) มาจัดลำดับใหม่ดังที่กล่าวมาแล้ว ลำดับสัญญาณทั้ง 9 ค่า ส่งมาหน่วยประมวลผลโดยการเปิดตารางข้อมูล (look-up table) ลำดับข้อมูลที่ออกมาจากหน่วยความจำจะถูกส่งตัวรวมข้อมูลเพื่อให้ได้ลำดับข้อมูลออก Y(m,n) ต่อไป จากรูปที่ 3.7 แสดงถึงวิธีการส่งลำดับ X(m,n) แบบอนุกรม เมื่อส่งครบ 8 บิทแล้วถึงจะได้ Y(m,n) 1 ไบท์

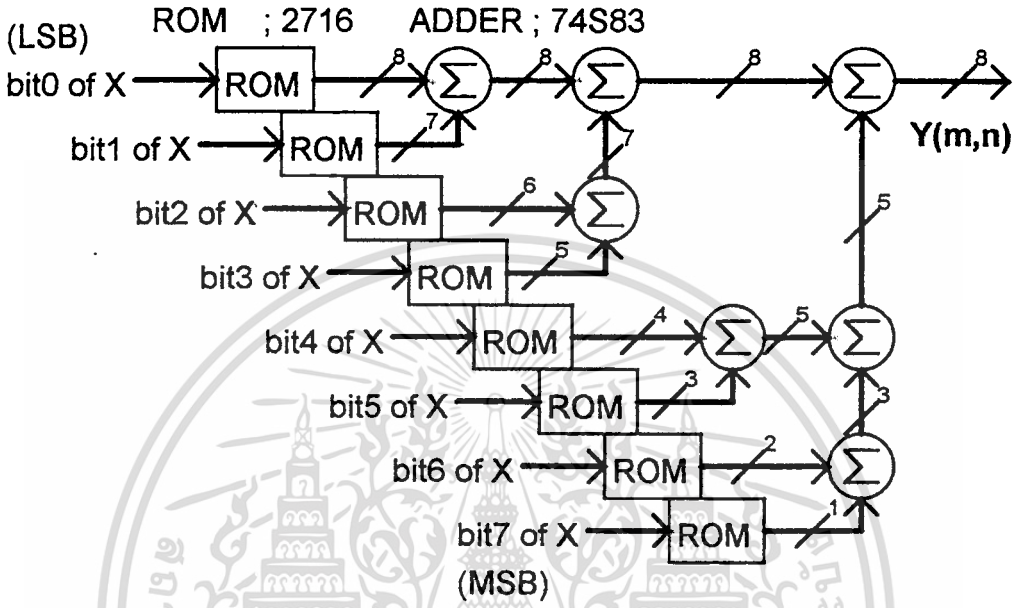


รูปที่ 3.7 โครงสร้างวงจรกรองสัญญาณเชิงเลขสองมิติแบบ 1 BAAT (Bit At A Time)[3]



รูปที่ 3.8 Time Diagram สัญญาณควบคุมจังหวะการทำงานของวงจรรูปที่ 3.7

จากรูปที่ 3.8 สัญญาณควบคุมการทำงานของวงจรการกระจายทางคณิตศาสตร์แบบอนุกรม จังหวะแรกสัญญาณ clr out ทำการเคลียร์ (clear) ข้อมูลออกที่ค้างอยู่ จังหวะที่สองสัญญาณ load ทำหน้าที่ให้ตัวเลื่อนข้อมูลเข้าขนานออกอนุกรม (Parallel In Serial Out : PISO) นำข้อมูลขนาด 8 บิททั้ง 9 ลำดับข้อมูล (ที่ส่งมาจากวงจรจัดลำดับข้อมูล 2 มิติ) เข้าไปเก็บไว้ในตัวเลื่อนข้อมูลในขณะที่ข้อมูลบิทแรกจะถูกเลื่อนออกมาด้วยเพื่อส่งไปเปิดตารางข้อมูลที่อยู่ในรอม จังหวะที่สามสัญญาณ ck rom buf' ทำหน้าที่ส่งข้อมูลจากรอมไปตัวรวมข้อมูล ส่วนสัญญาณ ck out ทำให้ผลลัพธ์จากตัวรวมข้อมูลส่งออกมาที่ตัวรวมข้อมูลในขณะที่จะทำการเลื่อนซ้าย 1 บิทตามทฤษฎีการกระจายทางคณิตศาสตร์ จังหวะที่สี่จะกลับไปวนรอบจังหวะที่สองอีกเจ็ดรอบ (ทำให้ครบจำนวน 8 บิท) ยกเว้นจะไม่มีสัญญาณ load เพราะจะโหลดขณะเลื่อน 1 ครั้งและเลื่อนต่ออีก 7 ครั้ง เมื่อรอบการทำงานครบ 1 ไซเคิลก็จะได้สัญญาณออก Y_n มีข้อสังเกตุอย่างหนึ่งว่าต้องมี buffer ที่รอมและตัวรวมข้อมูลเพราะจังหวะในการรวมข้อมูลจากรอมกับข้อมูลออกจะต้องรักษาสภาพข้อมูลจริงก่อนที่จะมีการเปลี่ยนแปลงมิฉะนั้นข้อมูล Y_n จะผิดไปจากทฤษฎี



รูปที่ 3.9 โครงสร้างวงจรกรองสัญญาณ 2 มิติ แบบขนาน (DA full parallel)

การส่งลำดับ X(m,n) นอกจากจะเป็นแบบอนุกรมแล้ว ยังสามารถส่งแบบขนานได้อีกด้วยทั้งนี้ก็เพื่อต้องการให้ความเร็วในการประมวลผลสูงขึ้น จากรูปที่ 3.9 จะแยกบิทแต่ละบิทของสัญญาณเข้าทั้ง 9แล้วส่งเข้าหน่วยความจำแต่ละตัวข้อมูลที่ออกจากหน่วยความจำจะถูกส่งไปรวมกันทั้งหมดแล้วได้ลำดับข้อมูล Y(m,n) ออกมา จะเห็นว่าการใช้วิธีสัญญาณเข้าแบบขนานนี้จะประมวลผลได้เร็วกว่าแบบอนุกรมถึง 8 เท่า เมื่อใช้ความถี่ของสัญญาณนาฬิกาเท่ากันการทำงานจะไม่ซับซ้อนเหมือนกับแบบอนุกรมแต่ว่าจำเป็นจะต้องใช้หน่วยความจำถึง 8 ชุด ตัวรวมข้อมูลอีก 7 ชุด ดังนั้นแบบขนานจะสิ้นเปลืองทั้งอุปกรณ์และกำลังงานมากกว่าจะให้ผลดีด้านความเร็วเพราะใช้สัญญาณนาฬิกา 1 ลูก

อย่างไรก็ตามถ้าอุปกรณ์ที่นำมาใช้สามารถทำงานได้กรณีใช้ความถี่สูงมาก ก็จะยิ่งเป็นผลดีกับการประมวลผลทั้ง 2 แบบเพราะถ้าเพิ่มความเร็วสัญญาณนาฬิกาใช้สูงมากอุปกรณ์หน่วยความจำโดยเฉพาะจำเป็นต้องใช้แบบพิเศษเพราะปกติรอมจะใช้กับความถี่ที่ไม่สูงมากนัก เช่น รอมของบริษัทเทคซัสอินสทรูเมนท์เบอร์ TMS27C292-3(2Kx8) เป็นรอมพิเศษที่มีค่า access time ต่ำถึง 35ns[12] ส่วนตัวรวมข้อมูลอาจจะใช้ F100181 (74S181) ซึ่งมี propagation delay time โดยเฉลี่ยประมาณ 5ns[13]

บทที่ 4
การประยุกต์ใช้งาน

4.1 การกระจายทางคณิตศาสตร์กับการกรองสัญญาณภาพ

ตัวกรองสัญญาณภาพมีประโยชน์ประยุกต์ใช้ โดยเฉพาะเป็นตัวเพิ่มคุณภาพ หรือ ลดสัญญาณรบกวนของสัญญาณในลักษณะ 2 มิติของระบบการประมวลสัญญาณเชิงเลขสัญญาณเหล่านี้จะต้องพิจารณาเป็นลักษณะช่วง ๆ สัญญาณซึ่งส่งมาเป็นลำดับ สามารถเขียนในรูปของฟังก์ชั่นได้โดยมีตัวแปรจำนวนเต็ม 2 ตัวซึ่งถูกกำหนดให้ค่าทุกค่าเป็นจำนวนเต็ม การประยุกต์ DA กับวงจรกรองสัญญาณภาพสมการของวงจรกรองโดยทั่วไปอยู่ในรูป

$$Y_{m,n} = \sum_{k=0}^{2}\sum_{l=0}^{2} a_{k,l} X_{m-k,n-l} \qquad (4.1)$$

$X_{m,n}$ และ $Y_{m,n}$ เป็นสัญญาณเข้าและสัญญาณออกตามลำดับ $a_{k,l}$ สัมประสิทธิ์ของการกรองกำหนดให้สัญญาณทุกค่าอยู่ในช่วง +128 , -128 ของเลขคอมพลีเมนท์ของ 2 (2's Complement) ความละเอียดขนาด 7 บิท รวมบิทเครื่องหมายอีก 1 บิท กระจายเป็นเลขฐานสองโดยสมการที่ 4.2

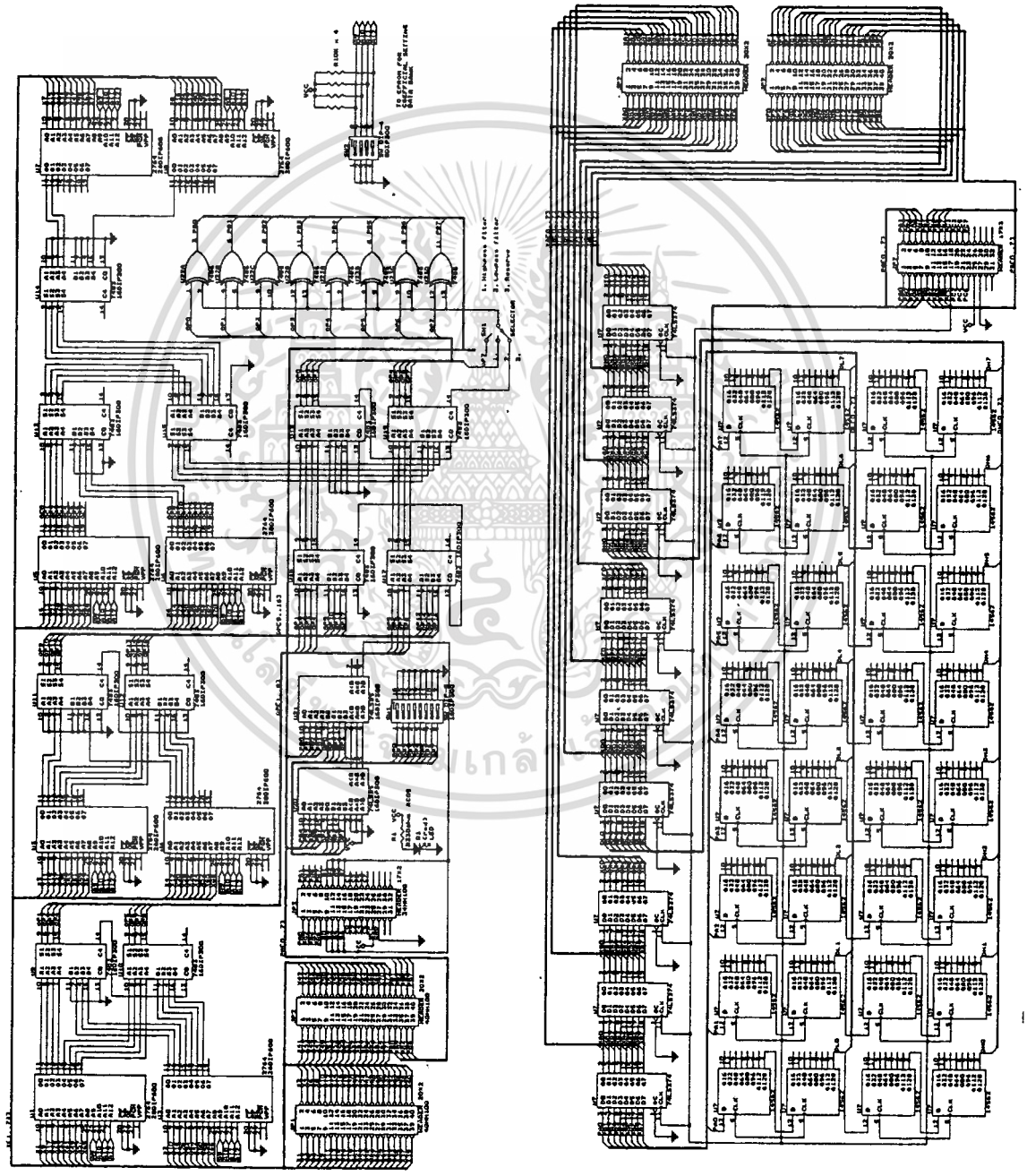
$$X_{m-k,n-l} = \sum_{s=1}^{7} X^{s}_{m-k,n-l} 2^{s} - X^{8}_{m-k,n-l} \qquad (4.2)$$

แทนค่าสมการที่ 4.2 ลงสมการที่ 4.1 จะได้สมการที่ 4.3

$$Y_{m-i,n-j} = \sum_{s=1}^{7}\left(\sum_{k=0}^{2}\sum_{l=0}^{2}(a_{k,l} X^{s}_{m-k,n-l})2^{s}\right) - \left(\sum_{k=0}^{2}\sum_{l=0}^{2}(a_{k,l} X^{s}_{m-k,n-l})\right) \qquad (4.3)$$

การออกแบบข้อมูลที่จะเก็บไว้ในหน่วยความจำกรณีของการประมวลสัญญาณ 2 มิติที่กล่าวถึงนี้มีลำดับข้อมูลเข้า 9 ชุด ดังนั้นต้องใช้รอมขนาด 512 x 8 จากนั้นนำเอาค่าสัมประสิทธิ์จากแผ่นข้อมูลมารวมกันแบบไม่ซ้ำได้ทั้งหมด 512 ค่า (2^9) สมมุติให้มีค่า 1, -2, 1, -2, 5, -2, 1, -2, 1 เป็นค่าที่อยู่ในแผ่นข้อมูลขนาด 3x3 นำค่าทั้ง 9 ค่านี้ มาบวกกันโดยใช้ระบบเลขฐานสองเป็นตัวกำหนดการบวก Y คือค่าที่ต้องบรรจุลงในรอม ถ้าค่า Y เป็นลบต้องเปลี่ยนเป็นเลขคอมพลีเมนท์ของสอง (2's complement) ก่อน แสดงการสร้างข้อมูลในบทที่ 3.3

การกระจายทางคณิตศาสตร์กับการกรองสัญญาณภาพต่อไปนี้ จะกล่าวถึงการประยุกต์ใช้งานการกระจายทางคณิตศาสตร์กับข้อมูลภาพ ที่ส่งมาจากคอมพิวเตอร์ซึ่งจำลองการทำงานแทนกล้อง



รูปที่ 4.1 วงจรกรองสัญญาณ 2 มิติ แบบขนาน (DA full parallel)

ถ่ายภาพและอุปกรณ์รับภาพแบบต่าง ๆ โดยปกติแล้วถ้าข้อมูลภาพไม่สมบูรณ์ เช่น ภาพไม่คมชัด (Blure), มีสัญญาณรบกวน (Noise) ในระบบการประมวลผลภาพ (Image Processing) มีวิธีการทำให้ภาพชัดเจนมากขึ้น (Enhance) ส่วนของงานวิจัยในบทที่ 4.1 จะประยุกต์กับการกรองความถี่สูงผ่าน (Highpass Filtering) และการกรองความถี่ต่ำผ่าน (Lowpass Filtering) โดยการประมวลภาพต่าง ๆ จะใช้วงจรดังรูปที่ 4.1 ทำการทดลองซึ่งไอซี MC14562 ทำหน้าที่เป็นตัวเลื่อนขนาด 128 บิท (128-BIT STATIC SHIFT REGISTER) ดังนั้นจะต้องใช้ 32 ตัว เพื่อหน่วงข้อมูลขนาด 8 บิท จำนวน 256 ข้อมูล 2 ครั้ง และผ่านไอซี 74LS374 หน่วงข้อมูลอีก 1 บิท ซึ่งการทำงานจะเป็นไปตามโครงสร้างจัดลำดับข้อมูลภาพรูปที่ 3.6 ดังนั้นจะได้ลำดับข้อมูล X(m,n) จนถึง X(m-2,n-2) (9 ลำดับข้อมูล) ส่งผ่านลำดับข้อมูลทั้ง 9 ไปยังคอนเน็คเตอร์ขนาด 20X2 จำนวน 2 ตัว ส่วนคอนเน็คเตอร์ขนาด 17X2 ตำแหน่ง PA[0..7] เป็นพอร์ท A จาก 8255 การ์ดส่งลำดับข้อมูลเข้ามาจัดลำดับข้อมูลใหม่เพื่อให้ได้ลำดับข้อมูลแบบ 2 มิติ โดยใช้ PC0 แทนสัญญาณนาฬิกาที่ป้อนให้ขา CLK ของ IC MC14562 และ 74LS374 ทุกตัว ลำดับข้อมูล 2 มิติดังกล่าวจะถูกส่งไปยังตัวประมวลผลโดย DA สังเกตได้ว่าลำดับข้อมูลทั้ง 9 ค่าจะถูกแยกบิทออกจากกันทั้งหมดโดยบิท 0 ของข้อมูลทั้ง 9 จะถูกส่งไปยังรอม U1 ในทำนองเดียวกันจนถึงบิท 7 ของข้อมูลทั้ง 9 จะถูกส่งไปยังรอม U8 ข้อมูลจากรอมทุกตัวจะนำมารวมกันโดยตัวบวก (ADDER) 74LS83 เป็นตัวบวกขนาด 4 บิท การประมวลผลจะเป็นไปตามโครงสร้างวงจรกรองสัญญาณสองมิติแบบขนานดังรูปที่ 3.9 ส่วนการเลือกค่าสัมประสิทธิ์จะกำหนดโดยดิปสวิตท์เป็นตัวเลือกแอดเดรสของรอม เอาท์พุทที่ได้จากการประมวลผลจะถูกส่งไปให้คอนเน็คเตอร์ขนาด 20X2 (JP2) โดยใช้ชื่อบัส PB[0..7] เพื่อส่งกลับไปยังพอร์ท B ของ 8255 ให้คอมพิวเตอร์ทำการจัดเก็บข้อมูลภาพที่ผ่านการประมวลผลไว้ในฮาร์ทดิสก์

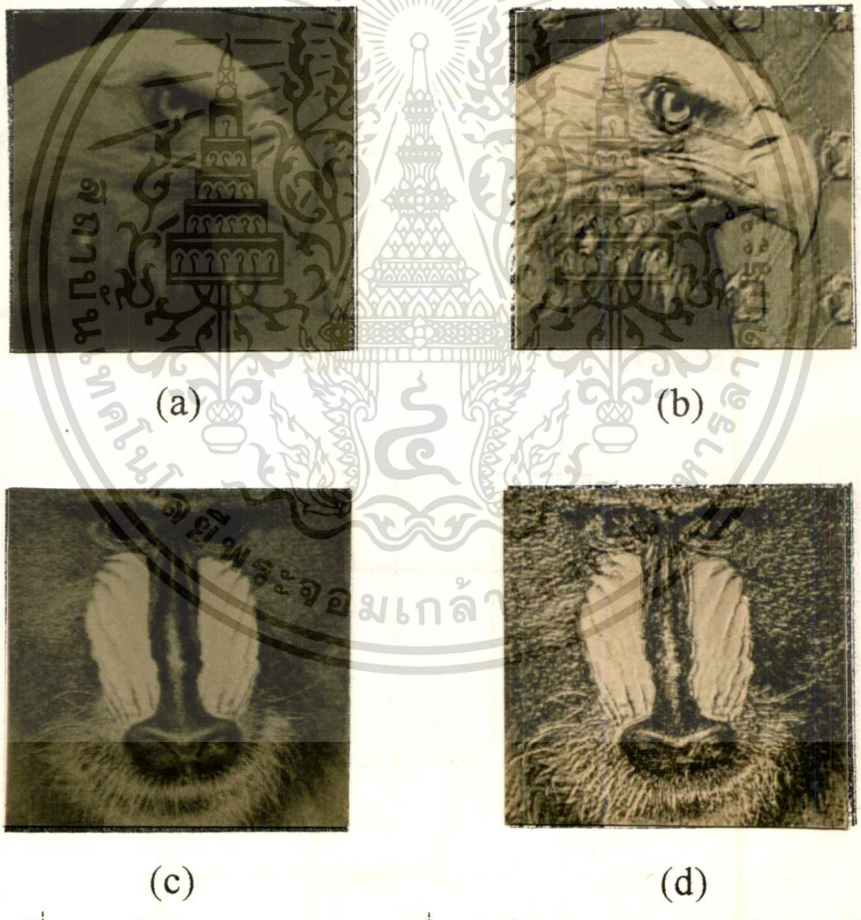
4.1.1 การกระจายทางคณิตศาสตร์กับการกรองความถี่สูงผ่าน

ตัวกรองความถี่สูงผ่าน(ความหมายโดยทั่วไปในการประมวลผลภาพ เป็นการเน้นส่วนที่เป็นความถี่สูงหรือระดับความแตกต่างความเข้มของแสงมาก ขณะที่ลดส่วนที่เป็นความถี่ต่ำหรือระดับความแตกต่างความเข้มของแสงน้อย เพราะว่าขอบ (Edge) หรือรายละเอียดของภาพเป็นส่วนของความถี่สูงดังนั้นการกรองความถี่สูงผ่านจะเป็นการเพิ่มความแตกต่างของแสงแต่ละที่ (Local Contrast) และ ความคมชัด (Sharp) ของภาพ ตัวอย่างผลตอบสนองทางอิมพัลส์ (Impulse Response) ของตัวกรองความถี่สูงผ่านที่ใช้ปรับความแตกต่างของแสงให้ชัดเจนขึ้นแสดงดังรูปที่ 4.2[10]

0	-1	0
-1	5	-1
0	-1	0

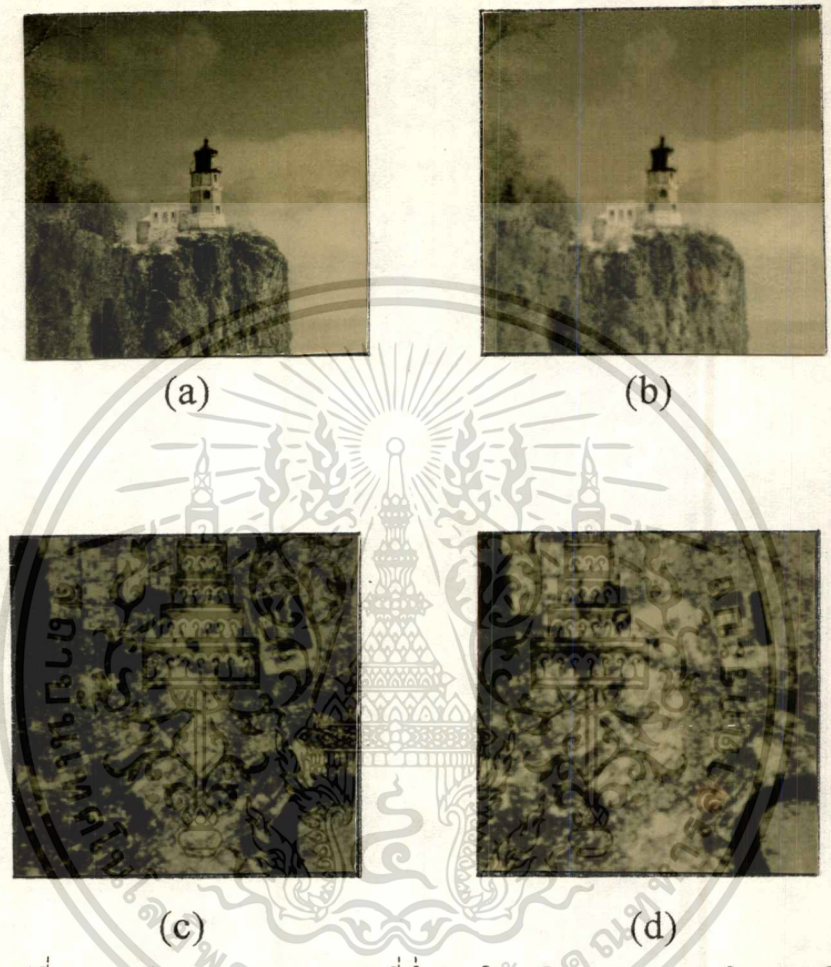
รูปที่ 4.2 ผลตอบสนองทางอิมพัลส์ของตัวกรองความถี่สูงผ่าน

คุณสมบัติหนึ่งของทุกตัวกรองในรูปที่ 4.2 คือผลรวมทุก ๆ แอมพลิทูด (Amplitude) ของผลตอบสนองทางอิมพัลส์ทุก ๆ ตัวเท่ากับ 1 ดังนั้นผลตอบสนองทางความถี่ของตัวกรองคือ 1 ที่ $\omega 1 = \omega 2 = 0$ และจะให้ส่วนที่เป็น DC (ไม่มีการเปลี่ยนแปลงของระดับสัญญาณ) ผ่านค่า ๆ นั้นออกไป จากคุณสมบัตินี้มีผลกระทบกับค่าเฉลี่ยของความเข้มของแสง (Average Intensity) ที่สงวนเอาไว้ไม่ให้เปลี่ยนแปลงในภาพต้นฉบับ นอกจากนี้ผลลัพธ์ความเข้มของแสงที่ประมวลผลออกมาจะไม่อยู่ในช่วง 0 ถึง 255 เสมอไป ถ้าความเข้มของแสงบางจุด (Pixel) ในการประมวลผลภาพมีค่าเกินช่วง เราสามารถตัดออกจากช่วง 0 และ 255 หรือทำการกำหนดอัตราส่วนใหม่ (Rescale) ของความเข้มทุกจุดให้อยู่ในช่วง 0 ถึง 255



รูปที่ 4.3 ตัวอย่างของการกรองความถี่สูงผ่านโดยการกระจายทางคณิตศาสตร์
(a,c) ภาพที่นำมาทดลองขนาด 256 x 256 (b,d) ภาพกรองความถี่สูงผ่าน

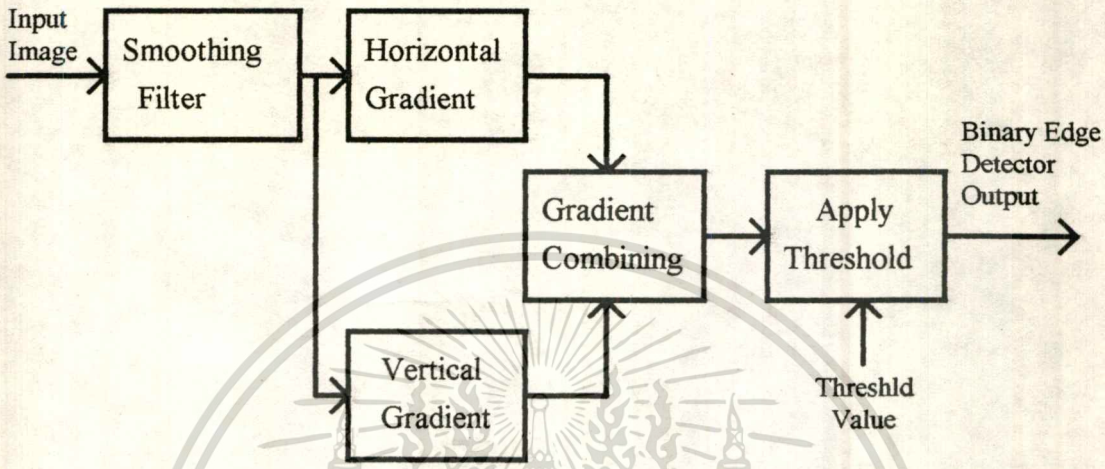
จากรูปที่ 4.3 เป็นผลของการกรองความถี่สูงผ่านโดยรูปที่ 4.3(a),(c) เป็นภาพต้นแบบที่นำมาทดลองมีขนาด 256 x 256 และรูปที่ 4.3 (b),(d) แสดงภาพซึ่งเป็นผลลัพธ์ของการกรองความถี่สูงผ่าน โดยใช้ตัวกรอง



รูปที่ 4.5 ตัวอย่างการกรองความถี่ต่ำผ่านโดยการกระจายทางคณิตศาสตร์
(a,c) ภาพที่นำมาทดลองขนาด 256 x 256 (b,d) ภาพการกรองความถี่ต่ำผ่าน

4.2 การกระจายทางคณิตศาสตร์กับ Sobel Edge Detector

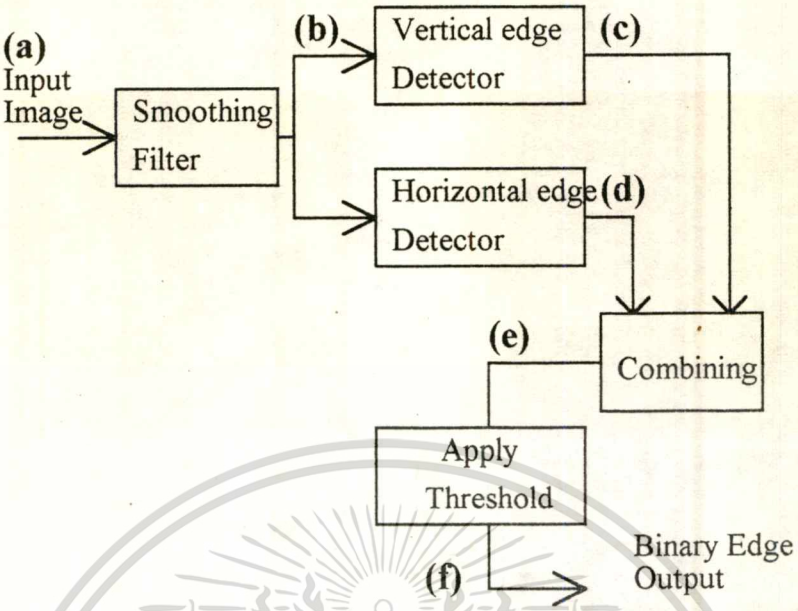
Sobel edge detector[10] เป็นกระบวนการทำให้ข้อมูลภาพในส่วนที่เป็นเส้นขอบรูป (contour) ของวัตถุในภาพมีลักษณะเด่นชัดขึ้นโดยจะไม่คำนึงข้อมูลภาพในส่วนอื่นมากนัก วิธีการทำได้โดยใช้แผ่นข้อมูล (mask) 3 ชุด ที่มีขนาด 3x3 จุด และมีข้อมูลดังนี้ {1, 2, 1, 2, 4, 2, 1, 2, 1} ทำหน้าที่ smooth filter , {1, 1, 1, 0, 0, 0, -1, -1, -1} ทำหน้าที่ horizontal edge detector, {-1, 0, 1, -1, 0, 1, -1, 0, 1} ทำหน้าที่ vertical edge detector แผ่นข้อมูลทั้ง 3 ชุดนี้จะวางทาบไปบนข้อมูลภาพแล้วเคลื่อนที่ไปในตำแหน่งต่าง ๆ ของภาพทีละจุด ในแต่ละครั้งข้อมูลทั้งหมดที่อยู่ในบริเวณนี้ที่ตำแหน่งเดียวกันจะถูกนำมาคูณกันผลลัพธ์ที่ได้ทั้ง 9 ค่าจะนำมารวมกันเพื่อคำนวณหาขนาด (absolute value) แล้วเก็บเป็นข้อมูลใหม่ ณ จุดกึ่งกลาง แสดงการทำงาน edge detector ดังรูปที่ 4.6



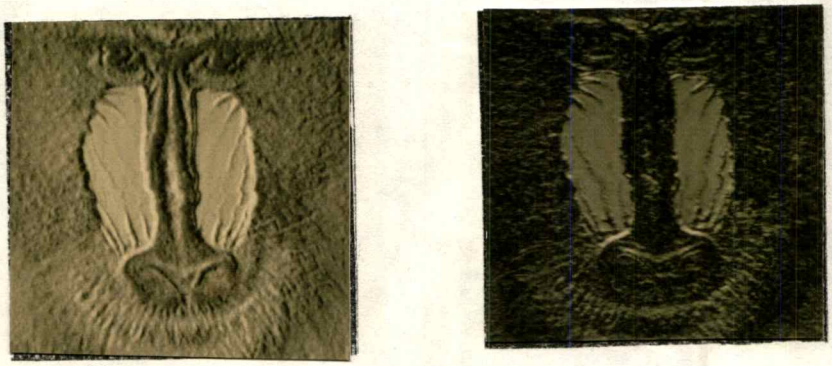
รูปที่ 4.6 การทำงานของ Sobel edge detection

การทดลอง edge detect กับ DA นี้ ต้องแปลงแผ่นข้อมูลทั้ง 3 ชุดตามวิธีบทที่ 2 จากนั้นส่งข้อมูลภาพผ่าน DA ฮาร์ดแวร์ 3 รอบ ที่ทำเช่นนี้เพราะว่าใช้แผ่นข้อมูล 3 ชุด แต่ฮาร์ดแวร์ประมวลผลได้ทีละ 1 ข้อมูลภาพเท่านั้นจากนั้นนำข้อมูลภาพที่เป็น ver-edge และ hor-edge มารวมกันทางซอล์ฟแวร์รวมถึงการปรับค่า threshold ด้วย อย่างไรก็ตามเราสามารถที่จะเพิ่ม DA ฮาร์ดแวร์ 3 ชุด, วงจรรวมข้อมูลระหว่าง ver-edge กับ hor-edge และ วงจรเปรียบเทียบ (comparator)เพื่อกำหนดค่า threshold ก็จะทำให้การประมวลผลของ DA กับ edge detector สะดวกและเร็วยิ่งขึ้น ผลที่ได้จากการทดลองนี้ถึงแม้ว่าจะใช้ซอล์ฟแวร์บ้าง แต่ก็เป็นเพียงเรื่องการรวมข้อมูลการแสดงผลบนจอคอมพิวเตอร์ไม่ได้มีการใช้ซอล์ฟแวร์ตกแต่ง หรือ filter ข้อมูลภาพแบบใด ๆ ทั้งสิ้น

จากรูปที่ 4.7 เป็นโครงสร้างการทำงานหาขอบภาพโดย Sobel edge detector ตัวอักษรที่กำกับอยู่ระหว่างบล๊อก จะกำหนดให้เป็นข้อมูลภาพดังรูปที่ 4.8(a)-4.8(f) โดยรูปที่ 4.8(a) เป็นข้อมูลภาพที่นำมาทดลองผ่านตัวกรองแบบราบเรียบประมวลผลโดยวงจร DA full parallel ได้ผลเป็นภาพ 4.8(b) ส่งประมวลผลอีก 2 ครั้งสำหรับหาขอบแนวตั้งและแนวนอน ได้ภาพ 4.8(c) และ 4.8(d) ตามลำดับ นำภาพทั้งสองมารวมกันทางซอร์ฟแวร์ได้ภาพ 4.8(e) แล้วทำการกำหนดค่าเทรสโชลด์จะได้ขอบของภาพสองระดับดังรูปที่ 4.8(f)



รูปที่ 4.7 โครงสร้างการทำงานหาขอบภาพโดย Sobel edge detector



รูปที่ 4.8 ตัวอย่างของ sobel edge detector

(e)

(f)

รูปที่ 4.8 (ต่อ) ตัวอย่างของ Sobel edge detector

(a) ข้อมูลภาพที่นำมาวิเคราะห์

(b) ข้อมูลภาพที่ผ่านตัวกรองความถี่ต่ำผ่าน

(c) ภาพที่ผ่านตัวกรอง Vertical Sobel edge detector

(d) ภาพที่ผ่านตัวกรอง Horizontal Sobel edge detector

(e) ภาพที่รวมแนวตั้งและแนวนอนเข้าด้วยกัน

(f) ขอบของภาพสองระดับที่กำหนดค่าเทรสโชลด์เท่ากับ 80

จากรูปที่ 4.9(a) เป็นข้อมูลภาพที่นำมาทดลองประมวลผลข้อมูล ภาพถูกประมวลทำให้เป็นภาพราบเรียบ (smiooth) โดยใช้ตัวกรองความถี่ต่ำผ่าน 1/16 { 1, 2, 1, 2, 4, 2, 1, 2, 1} แล้วส่งข้อมูลจากคอมพิวเตอร์ผ่านวงจรจัดลำดับข้อมูลให้เป็นลักษณะ 2 มิติ วงจรการกระจายทางคณิตศาสตร์จะประมวลภาพอีก 2 ครั้งคือ หาขอบภาพทางแนวตั้งและหาขอบภาพทางแนวนอน คล้ายกับรูปที่ 4.8(c) และ 4.8(d) ตามลำดับ ส่วนรูปที่ 4.9(b) เป็นผลลัพธ์ของขอบรูปภาพทั้งหมดโดยกำหนดค่าเทรสโชลด์เท่ากับ 65 รูปที่ 4.9 ไม่ได้แสดงรายละเอียดของภาพแต่ละขั้นตอน แต่วิธีการทดลองเหมือนกับรูปที่ 4.8

(a)

(b)

รูปที่ 4.9 (a) ข้อมูลภาพที่นำมาวิเคราะห์ (b) ผลของ Sobel edge detetor

4.3 การกระจายทางคณิตศาสตร์กับ Binary image edge detector

ภาพสองระดับ (Binary image) เป็นภาพที่มาจากภาพระดับสีเทา (gray scale) 256 ระดับที่ผ่านการตัดระดับสีเทาให้เป็นสองระดับ (0, 1) โดยการกำหนดค่าเทรสโชลด์ (Threshold) ภาพที่ได้ขึ้นมาใหม่จะมีค่า '0' = มืด และ '1' = สว่าง ซึ่งเป็นภาพจะมีเฉพาะส่วนที่สนใจเท่านั้น ส่วนที่เป็นพื้นด้านหลังจะเป็นสีเรียบสีเดียว และอีกส่วนหนึ่งคือวัตถุจริง ๆ ที่ต้องการวิเคราะห์โดยมีสีที่แตกต่างไปจากสีพื้น ภาพสองระดับจะไม่เน้นรายละเอียดของรูปภาพแต่จะเน้นเฉพาะรูปร่างของภาพทำให้มีประโยชน์ในด้านของประหยัดที่เก็บข้อมูล การวิเคราะห์หาพื้นที่ หาเส้นขอบ และการเข้ารหัส เป็นต้น เพื่อให้คอมพิวเตอร์ตรวจสอบชนิดของวัตถุได้ การทดลองนี้จะแตกต่างกับการประมวลผลภาพที่กล่าวมาแล้วเนื่องจากว่าภาพที่จะนำมาประมวลในบทนี้ข้อมูลแต่ละจุดมีขนาด 1 บิท ดังนั้นถ้าเรายอ้นกับไปถึงทฤษฎี และโครงสร้างของการกระจายทางคณิตศาสตร์จะทำให้เราทราบว่า วงจรหน่วงสัญญาณและวงจรรวมข้อมูลจะมีขนาดเล็กลงมาก พิจารณาสมการการกรองสัญญาณภาพต่อไปนี้

$$Y_{m,n} = \sum_{m=0}^{2}\sum_{n=0}^{2} M_{m-1,n-1} X_{m-1,n-1} \tag{4.4}$$

จากทฤษฎีบทที่ 2 ทำการกระจาย $X_{m-1,n-1}$

$$X_{m-k,n-l} = \sum_{s=0}^{7} X_{m-k,n-l}^{s} 2^{s} \tag{4.5}$$

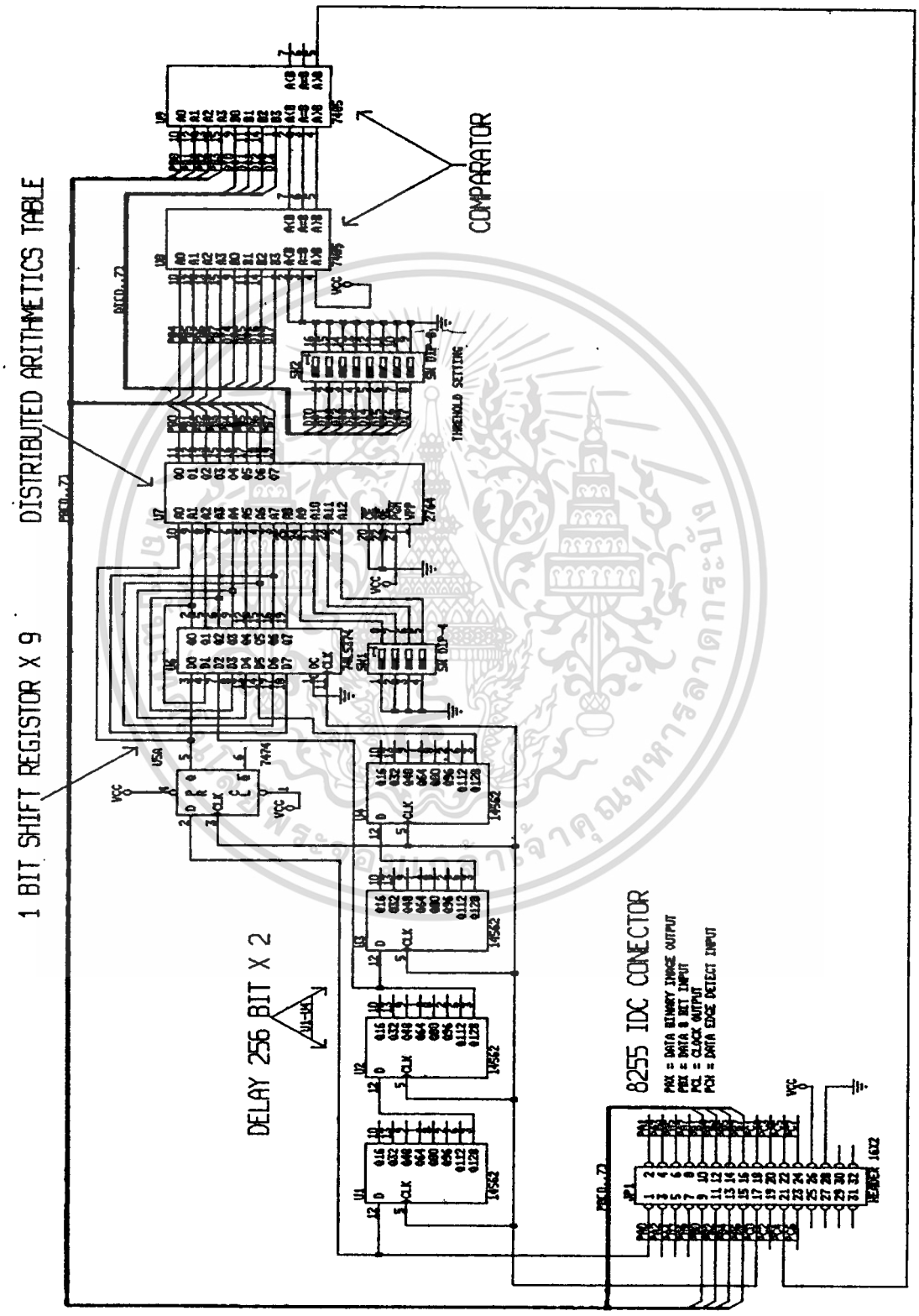
แต่เนื่องจากว่า $X_{m-1,n-1}$ มีขนาด 1 บิท ค่าของมันจึงเป็น 0 หรือ 1 เท่านั้น ดังนั้น

$$Y_{m,n} = 2^{0}\sum_{m=0}^{2}\sum_{n=0}^{2} M_{m-1,n-1} X_{m-1,n-1}^{0} \tag{4.6}$$

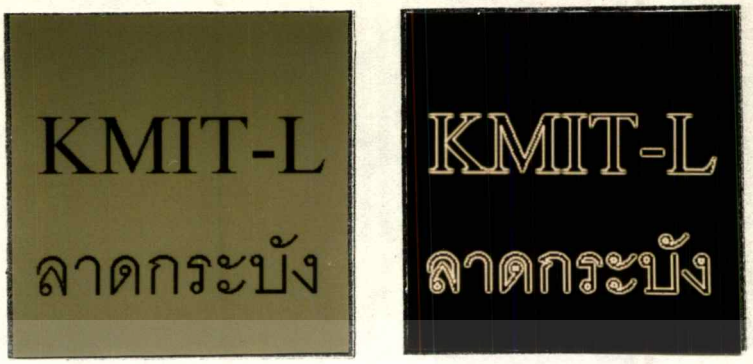
จากการทดลองใช้การกรองแบบลาปลาซเซียล (Laplacian Filtering)[10] ซึ่งมีแผ่นข้อมูล (mask) คือ

-1	-1	-1
-1	8	-1
-1	-1	-1

รูปที่ 4.10 ตัวกรองแบบ Laplacian edge detector



รูปที่ 4.11 วงจรการกระจายทางคณิตศาสตร์กับข้อมูลขนาด 1 บิท

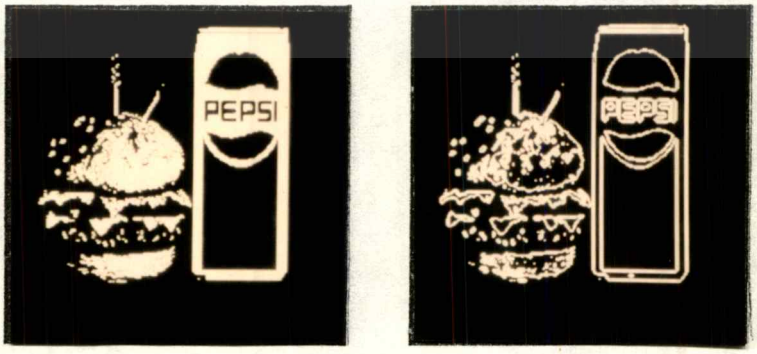


(a) (b)

รูปที่ 4.12 ตัวอย่างของ Laplacian edge detector
(a)รูปภาพที่นำมาทดลอง (b) ผลของลาปลาซเชียล

(a) (b)

รูปที่ 4.13 ตัวอย่างของ Laplacian edge detector
(a)รูปภาพที่นำมาทดลอง (b) ผลของลาปลาซเชียล



(a) (b)

รูปที่ 4.14 ตัวอย่างของ Laplacian edge detector
(a)รูปภาพที่นำมาทดลอง (b) ผลของลาปลาซเชียล

บทที่ 5
แนวทางพัฒนา

5.1 การพัฒนาการกระจายทางคณิตศาสตร์กับวงจรกรองอันดับสูง

จากที่เราได้ทราบกันแล้วว่า การคูณในคอมพิวเตอร์จะใช้เวลามากทำให้การทำงานไม่สามารถจะเป็นเวลาจริงได้ วิธีค้นหาจากรอมหรือวิธีรอมเปลี่ยนการคูณเป็นการบวกและเลื่อน แนวความคิดของวิธีนี้สามารถแสดงให้เห็นได้อย่างดีในการพัฒนากับการกรองแบบ Second-order Infinite Impulse Response (IIR)[11] ซึ่งความสัมพันธ์ ระหว่างอินพุทกับเอาท์พุท เป็นดังนี้

$$Y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} - b_1 Y_{n-1} - b_2 Y_{n-2} \qquad (5.1)$$

จากสมการที่ 5.1 ให้ X_n และ Y_n คือ สัญญาณแบบอันดับของอินพุทและเอาท์พุทตามลำดับ ส่วน a_0, a_1, a_2, b_1 และ b_2 เป็นสัมประสิทธิ์ของฟิลเตอร์

การสร้างวงจรกรองสัญญาณเชิงเลขอันดับ 6 (A Sixth Order Digital Filter Implememtation)

การประยุกต์กับวงจรกรองสัญญาณอันดับ 6 โดยใช้วงจรกรองอันดับ 2 จำนวน 3 ชุดมาแคสเคด (Cascade) กันมีรูปแบบดังสมการที่ 5.2a,5.2b และ 5.2c

$$V_n = a_{0,1} X_n + a_{1,1} X_{n-1} + a_{2,1} X_{n-2} - b_{1,1} V_{n-1} - b_{2,1} V_{n-2} \qquad (5.2a)$$

$$Z_n = a_{0,2} V_n + a_{1,2} V_{n-1} + a_{2,2} V_{n-2} - b_{1,2} Z_{n-1} - b_{2,2} Z_{n-2} \qquad (5.2b)$$

$$Y_n = a_{0,3} Z_n + a_{1,3} Z_{n-1} + a_{2,3} Z_{n-2} - b_{1,3} Y_{n-1} - b_{2,3} Y_{n-2} \qquad (5.2c)$$

เมื่อ X_n ลำดับสัญญาณเข้า (input sequence) ขณะที่ V_n , Z_n คือสัญญาณออกของส่วนแรกและส่วนที่ 2 ตามลำดับ Y_n เป็นสัญญาณออก (output sequence) โดยมี $\{a_{ij}\}$, $\{b_{ij}\}$ เป็นค่าสัมประสิทธิ์ของวงจรกรอง (j คือส่วนต่าง ๆ ของวงจรกรองอันดับ 2) ดังนั้นจึงตัดตารางของอันดับ 2 $F(X^K_n, X^K_{n-1}, X^K_{n-2}, Y^K_{n-1}, Y^K_{n-2})$ ถึง 3 ชุด โดยแต่ละชุดจะให้ 32 ค่า $(=2^5)$ โดยสมการที่ 5.3

$$F(X^K_n, X^K_{n-1}, X^K_{n-2}, Y^K_{n-1}, Y^K_{n-2}) = a_0 X^K_n + a_1 X^K_{n-1} + a_2 X^K_{n-2} - b_1 Y^K_{n-1} - b_2 Y^K_{n-2}) \qquad (5.3)$$

ค่า F ทั้ง 32 ค่าจะถูกเก็บไว้ในหน่วยความจำ 32 ตำแหน่ง ซึ่งสอดคล้องกับตำแหน่งที่ชี้โดยชิฟท์รีจิสเตอร์ X_n, X_{n-1}, X_{n-2}, Y_{n-1}, Y_{n-2} บิทขวาสุดโดยที่บิทขวาสุดของชิฟท์รีจิสเตอร์เหล่านี้จะเป็นตัวชี้แอดเดรสของค่า F ในตารางหน่วยความจำ จากนั้นหน่วยคำนวณก็จะนำค่า F ในตำแหน่งที่ถูกชี้โดยบิทขวาสุดของชิฟท์รีจิสเตอร์มาคำนวณดังสมการ (5.4)

$$Y_N = \sum_{K=1}^{B-1} 2^{-K} F(X_n^K, X_{n-1}^K, X_{n-2}^K, Y_{n-1}^K, Y_{n-2}^K) - F(X_n^0, X_{n-1}^0, X_{n-2}^0, Y_{n-1}^0, Y_{n-2}^0) \quad (5.4)$$

คำนวณไปทีละบิทจนครบ 16 บิท (สำหรับรีจิสเตอร์ 16 บิท) จึงจะได้ค่า Y_n หนึ่งครั้งแต่ถ้าเป็นแบบอันดับที่ 6 การคำนวณจะเป็นไปตามสมการ 5.5a,5.5b และ 5.5c

$$V_n = a_{0,1}X_n + a_{1,1}X_{n-1} + a_{2,1}X_{n-2} - b_{1,1}V_{n-1} - b_{2,1}V_{n-2} = F(X_n^K, X_{n-1}^K, X_{n-2}^K, V_{n-1}^K, V_{n-2}^K) \quad (5.5a)$$

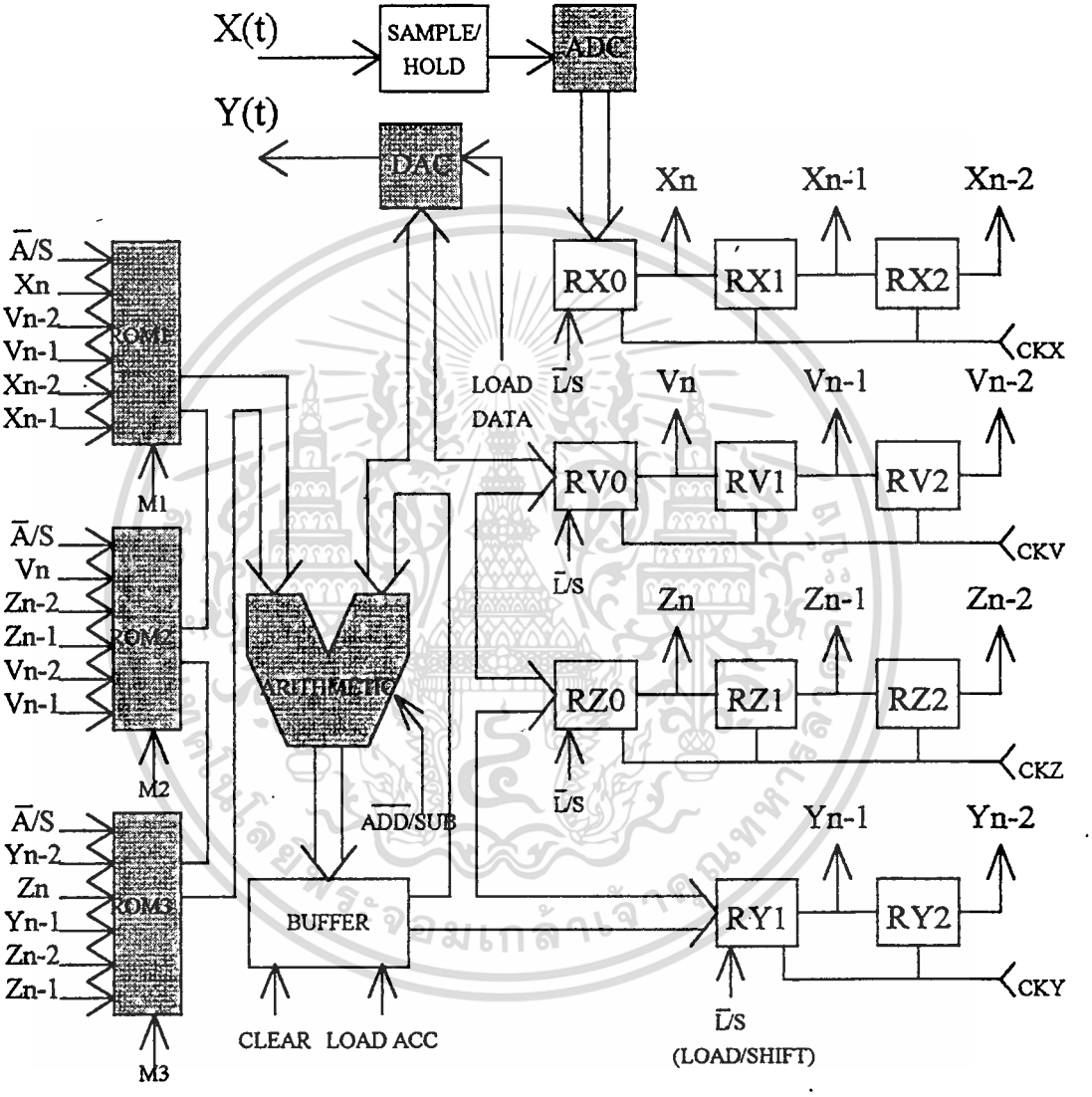
$$Z_n = a_{0,2}V_n + a_{1,2}V_{n-1} + a_{2,2}V_{n-2} - b_{1,2}Z_{n-1} - b_{2,2}Z_{n-2} = F(X_n^K, X_{n-1}^K, X_{n-2}^K, Z_{n-1}^K, Z_{n-2}^K) \quad (5.5b)$$

$$Y_n = a_{0,3}Z_n + a_{1,3}Z_{n-1} + a_{2,3}Z_{n-2} - b_{1,3}Y_{n-1} - b_{2,3}Y_{n-2} = F(X_n^K, X_{n-1}^K, X_{n-2}^K, Y_{n-1}^K, Y_{n-2}^K) \quad (5.5c)$$

จะเห็นได้ว่าถ้าสัมประสิทธิ์เปลี่ยนไป จะทำให้ลักษณะของการกรองเปลี่ยนไปด้วย เราสามารถกำหนดคุณลักษณะของฟิลเตอร์แล้วป้อนเข้าไปในโปรแกรมก็จะได้ค่าสัมสิทธ์ซึ่งเป็นเลขประกอบของสอง (2's Complement) เมื่อได้ค่าสัมประสิทธิ์แล้วก็สามารถป้อนเข้าไปเก็บไว้ใน ROM โดยผ่านไมโครคอมพิวเตอร์วงจรเดี่ยว (Single Board) หรือ เครื่องลอกแบบรอม (copy EPROM) ซึ่งสามารถป้อนเป็นเลขฐาน 16 และพัฒนาได้ง่าย และมีโครงสร้างของตัวกรองดังรูปที่ 5.1

รายละเอียดของสัญญาณควบคุมวงจรกรอง

สัญญาณ	หน้าที่
CKX (Clock register X control)	สัญญาณนาฬิกาของ RX0, RX1 และ RX2
CKV (Clock register X control)	สัญญาณนาฬิกาของ RV0, RV1 และ RV2
CKZ (Clock register X control)	สัญญาณนาฬิกาของ RZ0, RZ1 และ RZ2
CKY (Clock register X control)	สัญญาณนาฬิกาของ RY0, RY1 และ RY2
L/S (Load Shift)	สัญญาณควบคุมตัวเลื่อนข้อมูลให้อ่านและเลื่อนข้อมูล
CLEAR	เมื่อเป็น 0 จะเคลียร์ข้อมูลจากแอกคิวมูเลเตอร์
M1 (Select ROM 1)	กำหนดการทำงานของรอม 1
M2 (Select ROM 2)	กำหนดการทำงานของรอม 2
M3 (Select ROM 3)	กำหนดการทำงานของรอม 3



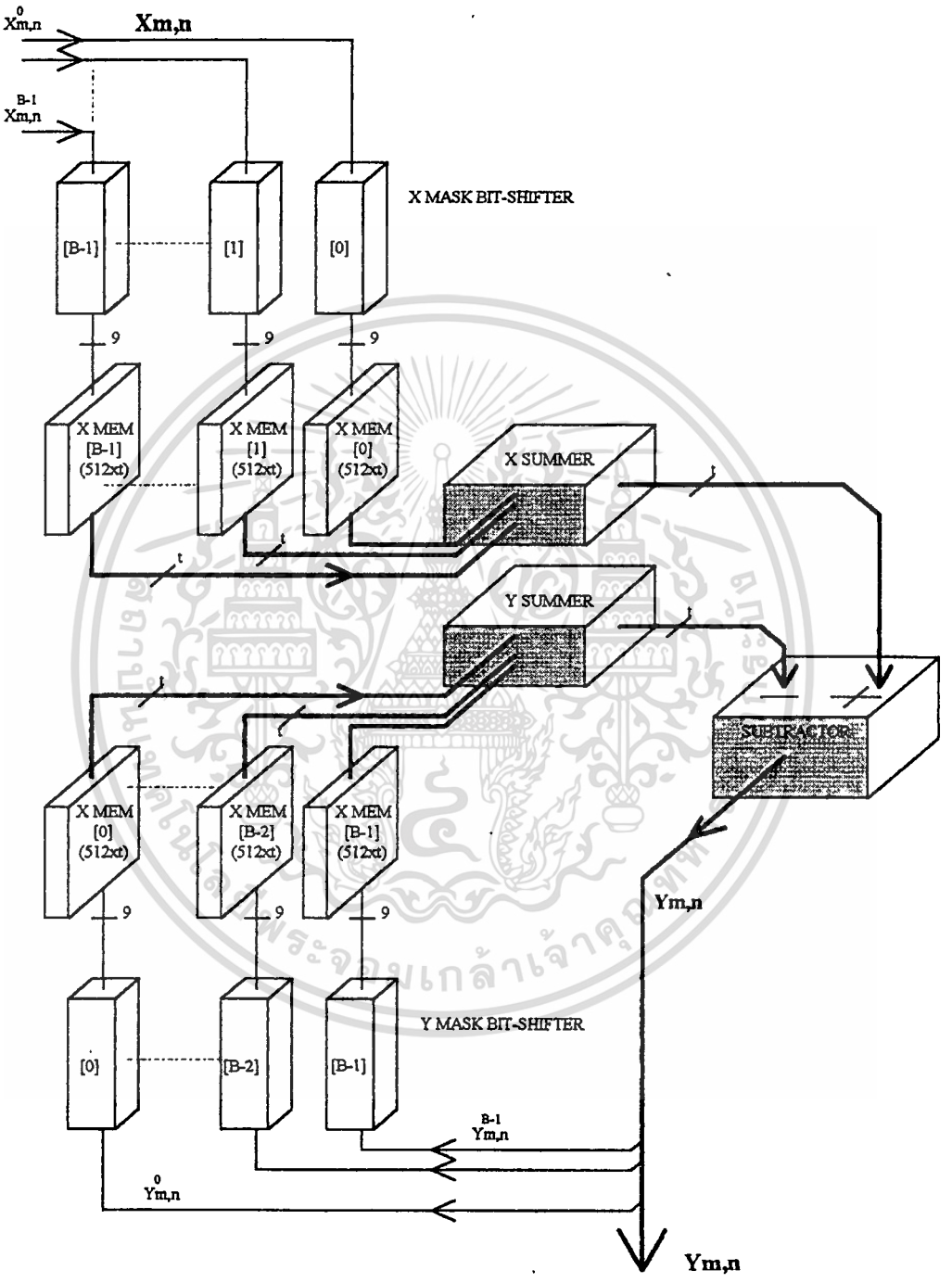
รูปที่ 5.1 โครงสร้างของวงจรกรองสัญญาณเชิงเลขอันดับ 6

5.2 การพัฒนาการกระจายทางคณิตศาสตร์กับตัวกรองสองมิติแบบป้อนกลับ

วงจรการกระจายทางคณิตศาสตร์ที่ทำวิจัยขึ้นมานี้ยังมีจุดที่ต้องพัฒนาให้ประมวลผลข้อมูลได้ละเอียดมากขึ้น จากทฤษฎีการกรองสัญญาณภาพการประมวลผลบางครั้งค่าที่ได้ไม่อยู่ในช่วง 0 ถึง 255 เสมอไปทำให้วงจรการกระจายทางคณิตศาสตร์ประมวลผลออกมาผิดพลาดไปจากทฤษฎี ตัวอย่างเช่น ตัวกรองความถี่ต่ำผ่าน มีค่าสัมประสิทธิ์ที่เป็นค่าบวกทั้งหมดเมื่อนำไปคูณกับข้อมูลภาพที่มีค่ามาก ก็จะทำให้ได้ผลลัพธ์ออกมาเกิน 255 หรือกรณีตัวกรองความถี่สูงผ่านค่าสัมประสิทธิ์มีทั้งค่าที่เป็นลบและบวก เมื่อนำไปคูณกับข้อมูลภาพที่ทำให้ได้ผลลัพธ์ติดลบ วงจรการกระจายทางคณิตศาสตร์จะประมวลผลออกมาผิดพลาด ดังนั้นถ้าเราจัดโครงสร้างของวงจรให้สามารถประมวลผลข้อมูลขนาด 12 บิท หรือมากกว่า การประมวลผลจะถูกต้องมากขึ้น จากวงจรการกระจายทางคณิตศาสตร์จะเห็นได้ว่ารอบตัวแรกใช้เอาท์พุทขนาด 8 บิท ตัวที่สอง 7 บิท จนถึงรอบตัวสุดท้ายใช้เพียงบิทเดียว ดังนั้นถ้าเราเอาเอาท์พุทที่ไม่ได้ใช้ของรอบตัวที่สองถึงตัวสุดท้ายมาใช้งานด้วยนั้นหมายความว่าเราจำเป็นจะต้องจัดตัวรวมข้อมูลให้มีโครงสร้างขนาด 16 บิท ส่วนผลลัพธ์ที่ออกมาจากการรวมข้อมูลก็ให้ทำการปรับอัตราให้ลดลง (scale down) ตามความเหมาะสมของข้อมูล นอกจากนี้ถ้าสมการการกรองสัญญาณสองมิติเป็นแบบป้อนกลับเราสามารถพัฒนาวงจรการกระจายทางคณิตศาสตร์เพิ่มมาอีกหนึ่งชุดสำหรับการจัดลำดับข้อมูลและประมวลผลข้อมูลของสัญญาณออกซึ่งจะแสดงโครงสร้างดังรูปที่ 5.2

จากรูปที่ 5.2 เป็นโครงสร้างตัวประมวลสัญญาณภาพ โดย X-MEM เป็นตัวเก็บค่าสัมประสิทธิ์ของการประมวลผลสัญญาณเข้ามีหน่วยความจำขนาด 512 x t (t แทนจำนวนบิท) จำนวน 8 ตัว สัญญาณเข้า $X_{m,n}$ จะถูกแยกบิทออกเป็นจำนวน 8 บิท ส่งผ่านตัวจัดลำดับข้อมูลสองมิติ X MASK BIT-SHIFTER ได้สัญญาณแบบสองมิติมีขนาด 9 บิท จำนวน 8 ชุด ส่งไปเปิดตารางข้อมูลเพื่อนำไปรวมกันโดยตัวรวมข้อมูล X SUMMER ในขณะที่สัญญาณ $Y_{m,n}$ ถูกจัดลำดับสัญญาณไปเปิดตารางข้อมูลและรวมข้อมูลเหมือนกับกรณีของ $X_{m,n}$ เอาท์พุทที่ได้จาก X SUMMER และ Y SUMMER จะถูกมารวมกันอีกทีโดยตัว SUBTRACTOR จะได้ผลลัพธออกมาเป็นสัญญาณ $Y_{m,n}$

การสร้างวงจรกรองสัญญาณเชิงเลข โดยการกระจายทางคณิตศาสตร์นี้ทำให้วงจรกรองสัญญาณประมวลสัญญาณด้วยความเร็วสูง อีกทั้งสามารถนำเอาอัลกอริทึมของวงจรกรองแบบต่าง ๆ ทำเป็นฮาร์ดแวร์ การสร้างฮาร์ดแวร์สามารถออกแบบได้สองลักษณะ แบบแรกเป็นการประมวลผลสัญญาณเข้าอนุกรม แบบที่สองประมวลสัญญาณเข้าแบบขนาน การกระจายทางคณิตศาสตร์นี้สามารถประยุกต์กับอัลกอริทึม ได้อีกมาก เช่น ฟาสท์ฟูเรียทรานส์ฟอร์ม (Fast Fourier Transform, FFT) ดิสครีทฟูเรียทรานส์ฟอร์ม (Discrete Fourier Transform, DFT) วงจรกรองอันดับสี่หรือสูงกว่านั้น การประมวลสัญญาณภาพแบบต่างๆ



รูปที่ 5.2 โครงสร้างตัวกรองสองมิติแบบป้อนกลับโดยวิธีการกระจายทางคณิตศาสตร์[8]

การสร้างวงจรกรองด้วยวิธีการกระจายทางคณิตศาสตร์นี้ หลักสำคัญจะต้องหาค่าสัมประสิทธิ์จากทรานส์เฟอร์ฟังก์ชั่นให้ได้ก่อนแล้วนำค่าสัมประสิทธิ์นี้ไปเก็บไว้ในหน่วยความจำ แล้วสัญญาณเข้าจะเข้าไปเปิดตารางหน่วยความจำออกมาประมวลผล ดังนั้นการออกแบบวงจรด้วยวิธีนี้สัญญาณออกจะถูกต้องขึ้นอยู่กับค่าสัมประสิทธิ์ และการทำงานของฮาร์ดแวร์เป็นหลัก เกี่ยวกับฮาร์ดแวร์เรื่องการบวกลบโดยคิดเครื่องหมาย อาจจำเป็นจะต้องเพิ่มวงจรการบวกโดยคิดเครื่องหมายและเปลี่ยนการประมวลผลจาก 8 บิท เป็น 16 บิทเพราะว่า การประมวลผลบางครั้งค่าที่ได้มีค่าเกิน 255 ถึงแม้ว่าข้อมูลจะมีค่าตั้งแต่ 0 - 255 เช่น low-pass filter มีค่าสัมประสิทธิ์ที่ทำให้การประมวลผลได้ค่า output เกิน 255 และต้องนำผลลัพธ์นั้นมาหารด้วย 16 อีกครั้งหนึ่งก่อนที่จะเป็นข้อมูลภาพแต่ละ pixel นั้น กรณีให้ DA ทำงานวิเคราะห์หาขอบภาพ Sobel edge detector จำเป็นต้องมีวงจรเพิ่มเติมอื่นอีกคือ วงจรตั้งค่า threshold , วงจร absolute value ดังนั้นผลการ วิเคราะห์หาขอบรูปภาพที่นำมาแสดงนี้ไม่ได้เป็นผลลัพธ์มาจากการประมวลผลจาก DA อย่างเดียว แต่ถูก นำมาแก้ไขข้อมูลทางซอฟแวร์ คือรวม vertical edge กับ horizontal เข้าด้วยกัน เป็นต้น วงจรการกระจายทางคณิตศาสตร์ที่สร้างขึ้นมาควรเป็นระบบที่พัฒนาทำงานกับเครื่องไมโครคอมพิวเตอร์เพื่อความสะดวกในการวิจัยและพัฒนาจากฮาร์ดแวร์ที่สร้างขึ้นเราสามารถ ที่จะพัฒนาวงจรรวมแบบ VLSI (very large scale integration) เพื่อทำให้ขนาดและกินกำลังงานน้อยลง[7] ระบบนี้ยังสามารถนำไปพัฒนากับ algorithm ต่าง ๆ ที่เกี่ยวข้องกับการประมวลสัญญาณเชิงเลขได้ในอนาคต

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

สรุปผลการวิจัยและข้อเสนอแนะ

การประมวลภาพโดยวิธีการกระจายทางคณิตศาสตร์กับตัวกรองแบบต่าง ๆ ในการทดลองนั้น ผลลัพธ์ที่ได้คือ ภาพที่เห็นความคมมากขึ้นสำหรับการกรองความถี่สูงผ่าน ภาพมัว (Blur) เป็นผลมาจากตัวกรองความถี่ต่ำผ่าน ส่วนภาพที่เป็นขอบใช้การหาขอบภาพแบบ Sobel edge detector และ Laplacian ภาพที่กล่าวมาทั้งหมดนี้ผ่านวงจรการกระจายทางคณิตศาสตร์แบบขนานโดยใช้ตัวจัดลำดับข้อมูลภาพที่มีไอซี MC14562 เป็นตัวหน่วงข้อมูล ใช้สัญญาณนาฬิกาจำนวน 65536 ลูก ดังนั้นความเร็วในการประมวลผลต่อ 1 ภาพ จึงขึ้นอยู่กับความเร็วของคอมพิวเตอร์ ถ้าคอมพิวเตอร์ส่งสัญญาณนาฬิกาที่ความถี่ 1 MHz เวลาที่ใช้ในการประมวลผลใช้เวลาประมาณ 0.07 วินาที แต่การทดลองในงานวิจัยนี้ใช้เครื่องคอมพิวเตอร์รุ่น 486DX2-66 เวลาประมวลผลใช้เวลาประมาณ 6 วินาที เนื่องจากเวลาการทำงานของโปรแกรมส่วนใหญ่ใช้ไปกับการอ่านและเขียนข้อมูลที่ฮาร์ทดิสก์ การประมวลผลรวมจึงใช้เวลามาก ผลลัพธ์ที่ได้จากการประมวลผลโดยวงจรการกระจายทางคณิตศาสตร์ในแต่ละจุดภาพนั้น บางครั้งข้อมูลไม่อยู่ในช่วง 0 ถึง 255 จึงทำให้เกิดการผิดพลาดที่จุดภาพนั้นขึ้นมาได้ อาจจำเป็นจะต้องสร้างวงจรตรวจสอบข้อมูลเพิ่มเข้าไปใช้งานเฉพาะกับการประมวลผลนั้น ๆ

วงจรการกระจายทางคณิตศาสตร์ที่วิจัยขึ้นมานี้ยังมีขนาดใหญ่โตอยู่พอสมควรจึงใช้กำลังงานมากบางครั้งการประมวลผลภาพอาจจะผิดพลาดอันเนื่องมาจากความร้อน และความถี่ของสัญญาณนาฬิกาสูง ดังนั้นถ้าผู้ศึกษาวิจัยทางด้านนี้ทำการออกแบบในรูปของวงจรขนาดใหญ่ (VLSI)[7] ก็จะทำให้ใช้กำลังงานและมีขนาดน้อยลง

ตารางเปรียบเทียบเวลาประมวลผล

หลักการ / ขั้นตอนประมวลภาพ	เวลาที่ใช้แต่ละขั้นตอน (วินาที)	
	ซอร์ฟแวร์	Distributed Arithmetics
1. อ่านข้อมูลภาพจากฮาร์ทดิสก์	2	2
2. ประมวลข้อมูลภาพ 256x256 จุด	6	1
3. แสดงภาพผลลัพธ์บนมอนิเตอร์	1	1
4. เก็บข้อมูลภาพลงฮาร์ทดิสก์	2	2

หมายเหตุ ทั้งนี้ความเร็วในการประมวลผล ขึ้นอยู่กับเทคนิคของโปรแกรมด้วย

ผลงานวิจัยที่ได้รับการตีพิมพ์

1. สมจินต์ ทองปลิว, กอบชัย เดชหาญ. " การสร้างวงจรกรองสัญญาณเชิงเลข 2 มิติโดย Distributed Arithmetics ," วิศวสาร ลาดกระบัง , คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีที่ 11 ฉบับที่ 1 (มิถุนายน, 2537), 1-6.
2. K. Dejhan, F. Cheevasuvit, S. Thongplew, S. Oraintara and N. Arjrith, " A Sobel edge detection digital filter structure and its Distributed arithmetic implementation ," Proc. of The 15th Asian Conference on Remote Sensing, Bangalore, India, Novenber 17-23, 1994.

เอกสารอ้างอิง

[1] A. Peled and B. Liu, " A new hardware relization of digital filter," IEEE Trans. on A.S.S.P., Vol. ASSP-22, pp. 456-462, December 1974.

[2] A. Peled and B. Liu, " Digital Signal Processing , " John Wiley & Sons, pp. 212-226, 1976.

[3] S. A. White, " Application of Distributed Arithmetic , " IEEE ASSP magazine, Vol. 6, No.3, pp. 4-9, July 1989.

[4] G. K. Ma, F. J. Taylor, "Multiplier Policies for Digital Signal Processing ," IEEE ASSP magazine, Vol. 7, No.1, pp. 6-20, January 1990.

[5] C. F. N. Cowan, Stewart G. Smith and J. H. Elliott, " A Digital Adaptive Filter Using a Memory Accumulator Architecture ," IEEE Trans. on A.S.S.P., Vol. ASSP-31, June 1983.

[6] C. S. Burrus, " Digital Filter Structure Described by Distributed Arithmetic ," IEEE on circuit and systems, Vol. CAS-24, No. 12, December 1977.

[7] S. Zohar: " A VLSI Implementation of a Digital Filter Based on Distributed Arithmetic ," IEEE ASSP, Vol. 37, No. 1, January 1989.

[8] H. Jaggernauth and A.N. Venetsanopoulos, " Real-Time Image Processing Through Distributed Arithmetic ," IEEE Trans. on Circuit and Systems, Vol. 1, pp. 394-397, May 1983.

[9] P. M. Embree and B. Kimble, "C Language Algorithm for Digital Signal Processing ," Prentice Hall, pp. 393-400, 1991.

[10] J. S. Lim, " Two-Dimensional Signal and Image Processing ," Prentice Hall, pp. 476-487,1990.

[11] D. F. Elliott, "Handbook of Digital Signal Processing ," Academic press, pp. 964-772, 1987.

[12] K. S. Lin, " Texas Instruments Digital Signal Processing ," Prentice Hall, pp. (F)2-(F)9, 1988.

[13] C. E. Sporck," National Semiconductor F100K ECL Logic Databook ," pp.(3)199 -(3)205, 1990.

หน้าที่ของโปรแกรมที่ใช้ในงานวิจัยการกระจายทางคณิตศาสตร์

โปรแกรมที่ใช้ในงานวิจัยนี้มีทั้งหมด 9 โปรแกรม ใช้ภาษา BASIC 1 โปรแกรม และอีก 8 โปรแกรม ใช้ภาษา C ผู้ทำวิจัยขอสรุปการทำงานและใช้งานทั้ง 9 โปรแกรมดังต่อไปนี้

1.CRROM.BAS

2.RAMSHDA.C

3.RAMFLLDA.C

4.FULLPA2.C

5.ED181.C

6.TSHOLD.C

7.BINDIS.C

8.IMGTOBMP.C

9.BMPTOIMG.C

1. โปรแกรม CRROM.BAS

โปรแกรมทำหน้าที่สร้างข้อมูลตามทฤษฎีการกระจายทางคณิตศาสตร์ที่จะเก็บไว้ในรอม โปรแกรมทำ งานบนแผนวงจรไมโครคอมพิวเตอร์วงจรเดี่ยว (Single board) รุ่น CP32 ซึ่งติดวงจรรอมโดยใช้แรม (Ram pack) เพื่อไปใช้ในการทดลอง หรือติดตั้งวงจร ET-EPP เป็นตัวเขียนข้อมูลลงรอมหรือ Ram pack การทำงานของโปรแกรมจะรับข้อมูลของตัวเลข 9 ค่า ตามชนิดของการกรองแบบต่าง ๆส่วนเอาท์พุทจะถูกส่งไปเขียนใส่รอม

2. โปรแกรม RAMSHBA.C

โปรแกรมใช้กับวงจรการกระจายทางคณิตศาสตร์ที่ใช้ตัวจัดข้อมูลแบบแรมเป็นตัวหน่วงข้อมูลกับวงจรประมวลผลแบบอนุกรม โปรแกรมจะรับชื่อแฟ้มข้อมูลภาพขนาด 256 x 256 แล้วส่งข้อมูลภาพออกไปทีละจุดพร้อมกับส่งสัญญาณควบคุมให้วงจรจัดลำดับข้อมูลที่ใช้แรมและสัญญาณควบคุมวงจรประมวลผลแบบอนุกรม แล้วรับข้อมูลเอาท์พุทเขียนลงฮาร์ทดิสก์ในชื่อแฟ้มข้อมูลที่กำหนด

3. โปรแกรม RAMFLLDA.C

โปรแกรมใช้กับวงจรการกระจายทางคณิตศาสตร์ที่ใช้ตัวจัดลำดับข้อมูลแบบแรมเป็นตัวหน่วงข้อมูลกับวงจรประมวลผลแบบขนาน โปรแกรมจะรับชื่อแฟ้มข้อมูลภาพขนาด 256 x 256 แล้วส่งข้อมูลภาพออกไปทีละจุดขณะที่จะส่งสัญญาณควบคุมให้กับวงจรจัดลำดับข้อมูลที่ใช้แรม เอาท์พุทที่จะจากการประมวลผลจะถูกเขียนลงฮาร์ทดิสก์ในชื่อแฟ้มข้อมูลที่กำหนด

4. โปรแกรม FULLPA2.C

โปรแกรมใช้กับวงจรการกระจายทางคณิตศาสตร์ที่ใช้ตัวจัดลำดับข้อมูลแบบ MC14562 เป็นตัวหน่วง ข้อมูลกับวงจรประมวลผลแบบขนาน โปรแกรมจะรับชื่อแฟ้มข้อมูลขนาด 256 x 256 แล้วส่งข้อมูลภาพ ออกไปทีละจุดพร้อมกับส่งสัญญาณนาฬิกาไปเลื่อนข้อมูลจำนวน 1 ลูก เอาท์พุทที่ได้จากการประมวลผล จะถูกอ่านจากพอร์ท PA2 แล้วเขียนข้อมูลลงฮาร์ทดิสก์ในชื่อแฟ้ม

5.โปรแกรม ED181.C

โปรแกรมใช้งานกับวงจรการกระจายทางคณิตศาสตร์แบบ 1 บิทกับการหาขอบภาพโดย Laplacian filter โดยเฉพาะ โปรแกรมจะรับชื่อแฟ้มข้อมูลภาพไบนารีขนาด 256 x 256 แล้วส่งข้อมูลภาพออกไปทีละจุด พร้อมกับส่งสัญญาณนาฬิกาไปเลื่อนข้อมูล การทำงานเหมือนกับโปรแกรม FULLPA2.C จะแตกต่างกันตรงที่การใช้พอร์ทของ 8255

6. โปรแกรม TSHOLD.C

ทำภาพระดับสีเทา 256 ระดับ ไปเป็นภาพสองระดับ (Binary image) โปรแกรมจะรับชื่อแฟ้มข้อมูลที่จะทำการแปลงและค่าตัวเลขที่จะกำหนดให้เป็นจุดตัดระดับ (Threshold value) ข้อมูลภาพใหม่จะถูกเก็บไว้ในแฟ้มข้อมูลที่กำหนดมีค่า 0 กับ 1 เท่านั้น

7. โปรแกรม IMGTOBMP.C

โปรแกรมทำหน้าที่แปลงข้อมูลภาพในลักษณะของข้อมูลปกติให้อยู่ในโครงสร้างของแฟ้มข้อมูลแบบบิทแมบ (Bit map) ซึ่งมีประโยชน์ในเรื่องของสามารถนำไปแสดงผลในโปรแกรมกราฟฟิคอื่น ๆ ได้

8. โปรแกรม BMPTOIMG.C

แปลงโครงสร้างแฟ้มข้อมูลที่ทำหน้าที่ตรงข้ามกับโปรแกรม IMGTOBMP.C

CRROM.BAS

```
10 REM   CRROM.BAS create data for ROM
20 CLEAR
40 INPUT "Filename data to ROM :",FIL$
45 OPEN FIL$ FOR OUTPUT AS #1
110 INPUT "Data for position   X1 : ",   A
120 INPUT "Data for position   X2 : ",   B
130 INPUT "Data for position   X3 : ",   C
140 INPUT "Data for position   X4 : ",   D
150 INPUT "Data for position   X5 : ",   E
160 INPUT "Data for position   X6 : ",   F
170 INPUT "Data for position   X7 : ",   G
180 INPUT "Data for position   X8 : ",   H
190 INPUT "Data for position   X9 : ",   I
210 FOR R = 0 TO 1
220 FOR S = 0 TO 1
230 FOR T = 0 TO 1
240 FOR U = 0 TO 1
250 FOR V = 0 TO 1
260 FOR W = 0 TO 1
270 FOR X  = 0 TO 1
280 FOR Y  = 0 TO 1
290 FOR Z  = 0 TO 1
300 Y1 = (A*R) + (B*S) + (C*T)
310 Y2 = (D*U) + (E*V) + (F*W)
320 Y3 = (G*X) + (H*Y) + (I*Z)
330 YT = Y1 + Y2 + Y3
335 GOSUB 700
360 PRINT#1, CHR$(YT);
410 NEXT          R
420 NEXT          S
430 NEXT          T
440 NEXT          U
450 NEXT          V
460 NEXT          W
470 NEXT          X
480 NEXT          Y
490 NEXT          Z
500 CLOSE#1
600 END
700 REM sub  2'complement
705 IF YT > -1 THEN RETURN
710 YT = 256 + YT
750 RETURN
800 END
```

RAMSHDA.C

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>

int PA1=0x0300,PB1=0x0301,PC1=0x0302,PCONT_1=0x0303;
int PA2=0x0304,PB2=0x0305,PC2=0x0306,PCONT_2=0x0307;

void initport() {
  outportb(PCONT_1,0x82);
  outportb(PCONT_2,0x92);
  outportb(PB1,0x0ff);
  outportb(PA2,0x0ff);
}
/*-------------------------------------------------------------------------
*/
infile(argc,argv)
int argc;
char *argv[];
{
  FILE *fopen(),*fp_in,*fp_out;
  char fn_in[12],fn_out[12],*s;
  unsigned char input_line[256];
  register row,column,i;
  int scale;

  if(argc != 3)
   {
   printf("\nEnter name of input file: ");
   scanf("%s",fn_in);
   printf("Enter name of output file: ");
   scanf("%s",fn_out);
   printf("\nScale Down input =  ");
   scanf("%d",&scale);
   }
  else
   {
   strcpy(fn_out, argv[2]);
   strcpy(fn_in, argv[1]);
   }
  fp_out = fopen(fn_out,"w+b");
  fp_in  = fopen(fn_in,"rb");

  if(fp_in == NULL)
   {   printf("No input file %s\n",fn_in);
   exit(1);
   }
 printf("\n\n\n.....Please wait !.....");


 /* Read free(not write to disk) 1 pixel  */
  fread(&input_line[0],sizeof(input_line[0]),1,fp_in);
 outportb(PA1,input_line[0]/scale);
```

```
clockshift();

/* Read free (not write to disk) 256 pixel */
fread(&input_line[0],sizeof(input_line[0]),256,fp_in);
  for (i=0;i<256;i++)
  {
  outportb(PA1,input_line[i]/scale);
  clockshift();
  }

/* DA process  */
for (row=0;row<256;row++)
{
  fread(&input_line[0],sizeof(input_line[0]),256,fp_in);
    for(column=0;column<256;column++)
    {
    outportb(PA1,input_line[column]/scale);
    clockshift();
    clocksimage();
    input_line[column] = inportb(PB1);
    }
  fwrite(&input_line[0],sizeof(input_line[0]),256,fp_out);
}

  printf("\nFile %s filtering \n",fn_in);
  printf("Output in file: %s\n",fn_out);
  fclose(fp_in);
  fclose(fp_out);
  exit(1);
}

clockshift()
{
  outportb(PC1,0x0f0);
  outportb(PC1,0x0f1);
  outportb(PC1,0x0f0);
  outportb(PC1,0x0f8);
  outportb(PC1,0x0d8);
  outportb(PC1,0x0c8);
  outportb(PC1,0x0d8);
  outportb(PC1,0x0f8);
  outportb(PC1,0x0f0);
  outportb(PC1,0x0f4);
  outportb(PC1,0x0f0);
}
/*-------------------------------------------------------------------------*/
clocksimage()
{
  outportb(PC2,0x0d2);
  outportb(PC2,0x012);
  outportb(PC2,0x0d2);
  outportb(PC2,0x0c2);
  outportb(PC2,0x0c6);
  outportb(PC2,0x0da);
  outportb(PC2,0x0f2);
  outportb(PC2,0x0d6);
  outportb(PC2,0x0da);
```

```
  outportb(PC2,0x0f2);
  outportb(PC2,0x0d6);
  outportb(PC2,0x0da);
  outportb(PC2,0x0f2);
  outportb(PC2,0x0d6);
  outportb(PC2,0x0da);
  outportb(PC2,0x0f2);
  outportb(PC2,0x0d6);
  outportb(PC2,0x0da);
  outportb(PC2,0x0f2);
  outportb(PC2,0x0d6);
  outportb(PC2,0x0da);
  outportb(PC2,0x0f2);
  outportb(PC2,0x0d6);
  outportb(PC2,0x0da);
  outportb(PC2,0x0f2);
  outportb(PC2,0x0d6);
  outportb(PC2,0x0da);
  outportb(PC2,0x0f2);
}
/*-------------------------------------------------------------------*/
clearbuf()
{
 int c,zero=0;
 printf("Clear buffer shift_DA ");
 for (c=0;c<521;c++){
  outportb(PA1,zero);
  clockshift();
  }
 printf("\n");
}
/*-------------------------------------------------------------------*/
main()
{
 clrscr();
 initport();
 clearbuf();
 infile();
 getche();
}
/*-------------------------------------------------------------------*/
```

RAMFLLDA.C

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
int PA1=0x0300,PB1=0x0301,PC1=0x0302,PCONT_1=0x0303;
int PA2=0x0304,PB2=0x0305,PC2=0x0306,PCONT_2=0x0307;
void initport()
{
  outportb(PCONT_1,0x82);
  outportb(PCONT_2,0x92);
  outportb(PB1,0x0ff);
  outportb(PA2,0x0ff);
}
/*-------------------------------------------------------------------*/
infile(argc,argv)
int argc;
char *argv[]; {
  FILE *fopen(),*fp_in,*fp_out;
  char fn_in[12],fn_out[12],*s;
  unsigned char input_line[256];
  register row,column,i;
  int scale;
  if(argc != 3)
   {
   printf("\nEnter name of input file: ");
   scanf("%s",fn_in);
   printf("Enter name of output file: ");
   scanf("%s",fn_out);
   printf("\nScale Down input =  ");
   scanf("%d",&scale);
   }
  else
   {   strcpy(fn_out, argv[2]);
   strcpy(fn_in, argv[1]);
   }
  fp_out = fopen(fn_out,"w+b");
  fp_in  = fopen(fn_in,"rb");

  if(fp_in == NULL)
   {
   printf("No input file %s\n",fn_in);
   exit(1);
   }
 printf("\n\n\n.....Please wait !.....");

 /* Read free(not write to disk) 1 pixel  */
  fread(&input_line[0],sizeof(input_line[0]),1,fp_in);
outportb(PA1,input_line[0]/scale);
 clockshift();
 /* Read free (not write to disk) 256 pixel */
 fread(&input_line[0],sizeof(input_line[0]),256,fp_in);
   for (i=0;i<256;i++)
   {
   outportb(PA1,input_line[i]/scale);
   clockshift();
   }
```

```
/* DA process  */
for (row=0;row<256;row++)
{
  fread(&input_line[0],sizeof(input_line[0]),256,fp_in);
    for(column=0;column<256;column++)
    {
    outportb(PA1,input_line[column]/scale);
    clockshift();
    input_line[column] = inportb(PB1);
    }
  fwrite(&input_line[0],sizeof(input_line[0]),256,fp_out);
}
 printf("\nFile %s filtering \n",fn_in);
 printf("Output in file: %s\n",fn_out);
 fclose(fp_in);
 fclose(fp_out);
 exit(1);
}

clockshift()
{
 outportb(PC1,0x0f0);
 outportb(PC1,0x0f1);
 outportb(PC1,0x0f0);
 outportb(PC1,0x0f8);
 outportb(PC1,0x0d8);
 outportb(PC1,0x0c8);
 outportb(PC1,0x0d8);
 outportb(PC1,0x0f8);
 outportb(PC1,0x0f0);
 outportb(PC1,0x0f4);
 outportb(PC1,0x0f0);
}
/*-------------------------------------------------------------------*/
clearbuf()
{
 int c,zero=0;
 printf("Clear buffer shift_DA ");
 for (c=0;c<521;c++){
  outportb(PA1,zero);
  clockshift();
  }
 printf("\n");
}
/*-------------------------------------------------------------------*/
main()
{
 clrscr();
 initport();
 clearbuf();
 infile();
 getche();
}
/*-------------------------------------------------------------------*/
```

FULLPA2.C

```
/* fullpa2.c send data out to PA2 */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>

int PA1=0x0300,PB1=0x0301,PC1=0x0302,PCONT_1=0x0303;
int PA2=0x0304,PB2=0x0305,PC2=0x0306,PCONT_2=0x0307;

void initport()
{
  outportb(PCONT_1,0x82);
  outportb(PCONT_2,0x92);
  outportb(PB1,0x0ff);
  outportb(PA2,0x0ff);
}
/*_____________________________________________________________*/
infile(argc,argv)
int argc;
char *argv[];
{
  FILE *fopen(),*fp_in,*fp_out;
  char fn_in[12],fn_out[12],*s;
  unsigned char input_line[256];
  register row,column,i;
  int scale;

  if(argc != 3)
   {
   printf("\nEnter name of input file: ");
   scanf("%s",fn_in);
   printf("Enter name of output file: ");
   scanf("%s",fn_out);
   printf("\nScale Down input =  ");
   scanf("%d",&scale);
   }
  else
   {
   strcpy(fn_out, argv[2]);
   strcpy(fn_in, argv[1]);
   }
  fp_out = fopen(fn_out,"w+b");
  fp_in  = fopen(fn_in,"rb");

  if(fp_in == NULL)
   {
   printf("No input file %s\n",fn_in);
   exit(1);
   }
 printf("\n\n\n.....Please wait !.....");


 /* Read free(not write to disk) 1 pixel  */
  fread(&input_line[0],sizeof(input_line[0]),1,fp_in);
```

```
outportb(PA1,input_line[0]/scale);
clockshift();

/* Read free (not write to disk) 256 pixel */
fread(&input_line[0],sizeof(input_line[0]),256,fp_in);
  for (i=0;i<256;i++)
  {
  outportb(PA1,input_line[i]/scale);
  clockshift();
  }

/* DA process  */
for (row=0;row<256;row++)
{
  fread(&input_line[0],sizeof(input_line[0]),256,fp_in);
    for(column=0;column<256;column++)
    {
    outportb(PA1,input_line[column]/scale);
    clockshift();
    input_line[column] = inportb(PA2);
    }
  fwrite(&input_line[0],sizeof(input_line[0]),256,fp_out);
}

  printf("\nFile %s filtering \n",fn_in);
  printf("Output in file: %s\n",fn_out);
  fclose(fp_in);
  fclose(fp_out);
  exit(1);
}

clockshift()
{
  outportb(PC1,0x000);
  outportb(PC1,0x0ff);
  outportb(PC1,0x000);
}
/*-------------------------------------------------------------------*/
clearbuf()
{
 int c,zero=0;
 printf("Clear buffer shift_DA ");
 for (c=0;c<521;c++){
  outportb(PA1,zero);
  clockshift();
  }
 printf("\n");
}
/*-------------------------------------------------------------------*/
main()
{
 clrscr();
 initport();
 clearbuf();
 infile();
 getche();
}
/*-------------------------------------------------------------------*/
```

ED181.C

```
/* ed181.c send data output to PA2 */

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>

int PA2=0x0304,PB2=0x0305,PC2=0x0306,PCONT_2=0x0307;

void initport()
{
  outportb(PCONT_2,0x082);
  /*  outportb(PB2,0x0ff);  */
}
/*_____________________________________________________________*/
infile(argc,argv)
int argc;
char *argv[];
{
  FILE *fopen(),*fp_in,*fp_out;
  char fn_in[12],fn_out[12],*s;
  unsigned char input_line[256];
  register row,column,i;
  int scale;

  if(argc != 3)
   {
   printf("\n\n**  Laplacian Edge Detector (-1,8-1)  **");
   printf("\n\nEnter name of input file: ");
   scanf("%s",fn_in);
   printf("Enter name of output file: ");
   scanf("%s",fn_out);
   printf("\nScale Down input =  ");
   scanf("%d",&scale);
   }
  else
   {
   strcpy(fn_out, argv[2]);
   strcpy(fn_in, argv[1]);
   }
  fp_out = fopen(fn_out,"w+b");
  fp_in  = fopen(fn_in,"rb");

  if(fp_in == NULL)
   {
   printf("No input file %s\n",fn_in);
   exit(1);
   }
 printf("\n\n\n.....Please wait !.....");


 /* Read free(not write to disk) 1 pixel  */
  fread(&input_line[0],sizeof(input_line[0]),1,fp_in);
 outportb(PA2,input_line[0]/scale);
 clockshift();
```

```
 /* Read free (not write to disk) 256 pixel */
 fread(&input_line[0],sizeof(input_line[0]),256,fp_in);
   for (i=0;i<256;i++)
   {
   outportb(PA2,input_line[i]/scale);
   clockshift();
   }

 /* DA process  */
 for (row=0;row<256;row++)
 {
   fread(&input_line[0],sizeof(input_line[0]),256,fp_in);
     for(column=0;column<256;column++)
     {
     outportb(PA2,input_line[column]/scale);
     clockshift();
     input_line[column] = inportb(PB2);
     }
   fwrite(&input_line[0],sizeof(input_line[0]),256,fp_out);
 }

   printf("\nFile %s filtering \n",fn_in);
   printf("Output in file: %s\n",fn_out);
   fclose(fp_in);
   fclose(fp_out);
   exit(1);
}

clockshift()
{
   outportb(PC2,0x000);
   outportb(PC2,0x0ff);
   outportb(PC2,0x000);
}
/*-------------------------------------------------------------------*/
clearbuf()
{
 int c,zero=0;
 printf("Clear buffer shift_DA ");
 for (c=0;c<521;c++){
   outportb(PA2,zero);
   clockshift();
   }
 printf("\n");
}
/*-------------------------------------------------------------------*/
main()
{
 clrscr();
 initport();
 clearbuf();
 infile();
 getche();
}
/*-------------------------------------------------------------------*/
```

TSHOLD.C

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <graphics.h>
FILE *input_prt1,*input_prt2;
unsigned char input_line1[256];
void display(int row,int z,int threshold)
 {
    register column;
    int color;
    for (column=0;column<256;column++)
     if(input_line1[column] > threshold )
{
input_line1[column] = 1;
putpixel(column+z,row+1,7);
}      else
{
input_line1[column] = 0;
putpixel(column+z,row+1,0);
}
    fwrite((char *)input_line1,sizeof(char),256,input_prt2);
};
main()
 {
  int graphdriver = DETECT,graphmode;
  register row,column;
  char filename1[15],filename2[15];
  int threshold;
  clrscr();
  printf("Enter INPUT FILE  2 GRAY      >: ");
  gets(filename1);
  printf("Enter OUTPUT FILE 0-1 BINARY  >: ");
  gets(filename2);
  printf("\n\nEnter Threshold =  ");
  scanf("%d",&threshold);

  initgraph(&graphdriver,&graphmode,"");
  rectangle(29,80,286,337);
  /* Open Input File */
  input_prt1 = fopen(filename1,"rb");
  input_prt2=  fopen(filename2,"w+b");
  if( input_prt1 == (FILE *)NULL)
   {
     printf("Cannot open %s\n",filename1);
     exit(1);
   }
   /* Read lines from input file  */
    for (row=0;row<256;row++)
     {
fread((char *)input_line1,sizeof(char),256,input_prt1);
display(row+80,30,threshold);
     };
  getche();
  close(input_prt1);
  close(input_prt2);
  closegraph();
 }
```

IMGTOBMP.C

```
/*-------------------------------------------------------------
PROGRAM ->CHANGE IMAGE TO BMP FILE...
START WRITTEN->7 JUNE 1993...
LAST UPDATE ->8 JUNE 1993...
INSTITUTE ->KMIT'L...
COMMENT  ->...have output (*.bmp) that add to original file...
-------------------------------------------------------------*/
#include   <stdio.h>
#include   <dos.h>
#include   <string.h>
#include   <conio.h>
#include   <stdlib.h>

#define  pixel2bytes(n)  ((n+7)/8)

int       XPIX,YPIX;
int       t_width, t_height;

typedef struct  {
  char id[2];
long filesize;
  int reserved[2];
  long headersize;
  long infoSize;
  long width;
  long depth;
  int biPlanes;
  int bits;
  long biCompression;
  long biSizeImage;
  long biXPelsPerMeter;
  long biYPelsPerMeter;
  long biClrUsed;
  long biClrImportant;
  }BMPHEAD;

main()
{
int x,y,bytes;
char    *n;
FILE    *infile, *outfile; static char   name[80];
static unsigned char  image[1024];
BMPHEAD   bmp;

 clrscr();

 printf("\nCONVERT...IMAGE TO BMP...VERSION 2.0 (structure HEAD)");
 printf("\nBY..ARMOTE SOMBOONKAEW...");

 printf("\nxpix -> ");scanf("%d",&XPIX);
 printf("ypix -> ");  scanf("%d",&YPIX);
 /*for THE odd XPIX*/
 bytes=XPIX;
 if(bytes & 0x0003) {
bytes |= 0x0003;
 ++bytes;
 }
```

```
memset((char *)&bmp,0,(long)sizeof(BMPHEAD));
memcpy(bmp.id,"BM",2);

bmp.headersize=1078L;

bmp.width=(long)XPIX;
bmp.depth=(long)YPIX;
bmp.filesize=bmp.headersize+(long)bytes*(long)bmp.depth;
bmp.infoSize=0x28L;
bmp.bits=8;
bmp.biPlanes=1;
bmp.biCompression=0L;
bmp.biSizeImage=bmp.filesize-bmp.headersize;
bmp.biXPelsPerMeter=0L;
bmp.biYPelsPerMeter=0L;
bmp.biClrUsed=0L;
bmp.biClrImportant=0L;

printf("\nHeader Complete!");
do {
   textcolor(LIGHTGREEN);
   cprintf("\n\rInput from  -> ");  scanf("%s",&name);
   printf("Output  to  -> (*.bmp)...");
   printf("\nReading IMG file [%s]",name);
   infile = fopen(name,"rb");
   n=strchr(name,'.'); if(n) strcpy(n,".bmp"); else  strcat(name,".bmp");
   printf("\nSaving BMP file [%s]\n",name);
   outfile = fopen(name,"wb");
   if((infile&&outfile) == NULL) {
   printf(" Open file error..\n");
   exit(1);
   }
   /*write HEADER*/
   if(fwrite((char *)&bmp,1,(long)sizeof(BMPHEAD),outfile) !=
sizeof(BMPHEAD)) {
   printf("fwrite HEADER error");
   exit(1);
   }
   /*write PALETTE*/
   for(x=0;x<256;++x){
   fputc(x,outfile);/*B*/
   fputc(x,outfile);/*G*/
  fputc(x,outfile);/*R*/
   fputc(0,outfile);
   }
   /*write IMAGE*/
   memset((unsigned char *)&image,0,bytes);
   for(y=0;y<bmp.depth;++y){
   printf("p");
   fread(image,1,XPIX,infile);
   fwrite(image,1,bytes,outfile);
   }
   printf("\nClosing both file");
   fclose(infile);
   fclose(outfile);
 } while(1);
}
```

BMPTOIMG.C

```
/*-------------------------------------------------------------
PROGRAM ->CHANGE BMP TO IMGAGE FILE...
START WRITTEN->7 JUNE 1993...
LAST UPDATE->8 JUNE 1993...
INSTITUTE->KMIT'L...
COMMENT->...have output (*.img) that add to original file...
-------------------------------------------------------------*/
#include  <stdio.h>
#include  <dos.h>
#include  <string.h>
#include  <conio.h>
#include  <stdlib.h>

#define pixel2bytes(n) ((n+7)/8)

int     XPIX,YPIX;
int      t_width, t_height;

typedef struct  {
  char id[2];
  long filesize;
  int reserved[2];
  long headersize;
  long infoSize;
  long width;
  long depth;
  int biPlanes;
  int bits;
  long biCompression;
  long biSizeImage;
  long biXPelsPerMeter;
  long biYPelsPerMeter;
long biClrUsed;
  long biClrImportant;
  }BMPHEAD;

main()
{
int    x,y,bytes;
char    *n;
FILE    *infile, *outfile;
static char   name[80];
static unsigned char  image[1024]; BMPHEAD   bmp;

 clrscr();

 printf("\nCONVERT...BMP TO IMAGE...VERSION 2.0 (structure HEAD)");
 printf("\nBY..ARMOTE SOMBOONKAEW...");

 do {
    printf("\n\r");
    textcolor(LIGHTGREEN);
    cprintf("Input from  -> ");  scanf("%s",&name);
    printf("Output  to  -> (*.img)...");
```

```
    printf("\nReading BMP file [%s]",name);

    infile = fopen(name,"rb");
    n=strchr(name,'.'); if(n) strcpy(n,".img"); else  strcat(name,".img");

    printf("\nSaving IMG file [%s]",name);
    outfile = fopen(name,"wb");
    if((infile&&outfile) == NULL) {
    printf(" Open file error..\n");
    exit(1);
    }

    /*read HEADER*/
    memset((char *)&bmp,0,(long)sizeof(BMPHEAD));
    if(fread((char *)&bmp,1,(long)sizeof(BMPHEAD),infile) !=
sizeof(BMPHEAD)) {
    printf("fread HEADER error");
    exit(1);
    }

    printf("\nRead Header Complete!
(xsize=%ld,ysize=%ld)\n",bmp.width,bmp.depth);

    /*for THE odd XPIX*/
    XPIX=bytes=bmp.width;
    if(bytes & 0x0003) {
    bytes |= 0x0003;
    ++bytes;
    }

    /*write IMAGE*/
    fseek(infile,bmp.headersize,SEEK_SET);
    memset((unsigned char *)&image,0,bytes);
    for(y=0;y<bmp.depth;++y){
    printf("p");
    fread(image,1,bytes,infile);
    fwrite(image,1,XPIX,outfile);
    }
    printf("\nClosing both file");
    fclose(infile);
    fclose(outfile);
 }while(1);
}
```

วงจรที่ใช้ในการทดลอง

1. วงจรเชื่อมต่อกับคอมพิวเตอร์ (8255 INTERFACE CARD)

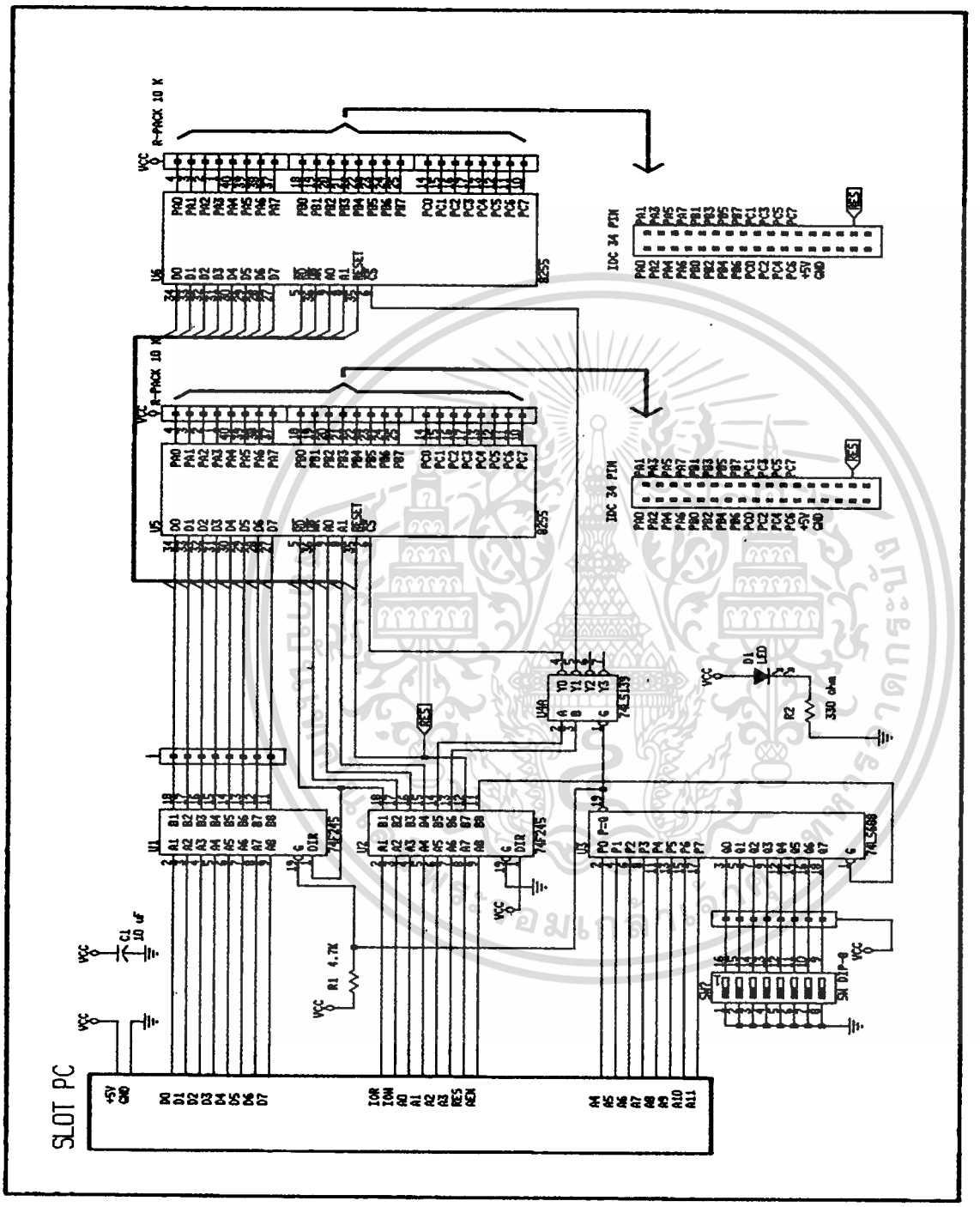
2. วงจรจัดลำดับสัญญาณ 2 มิติโดยใช้แรมเป็นตัวหน่วงข้อมูล
   - ลายแผ่นวงจรพิมพ์ด้านล่างและด้านอุปกรณ์
   - ตำแหน่งการวางอุปกรณ์

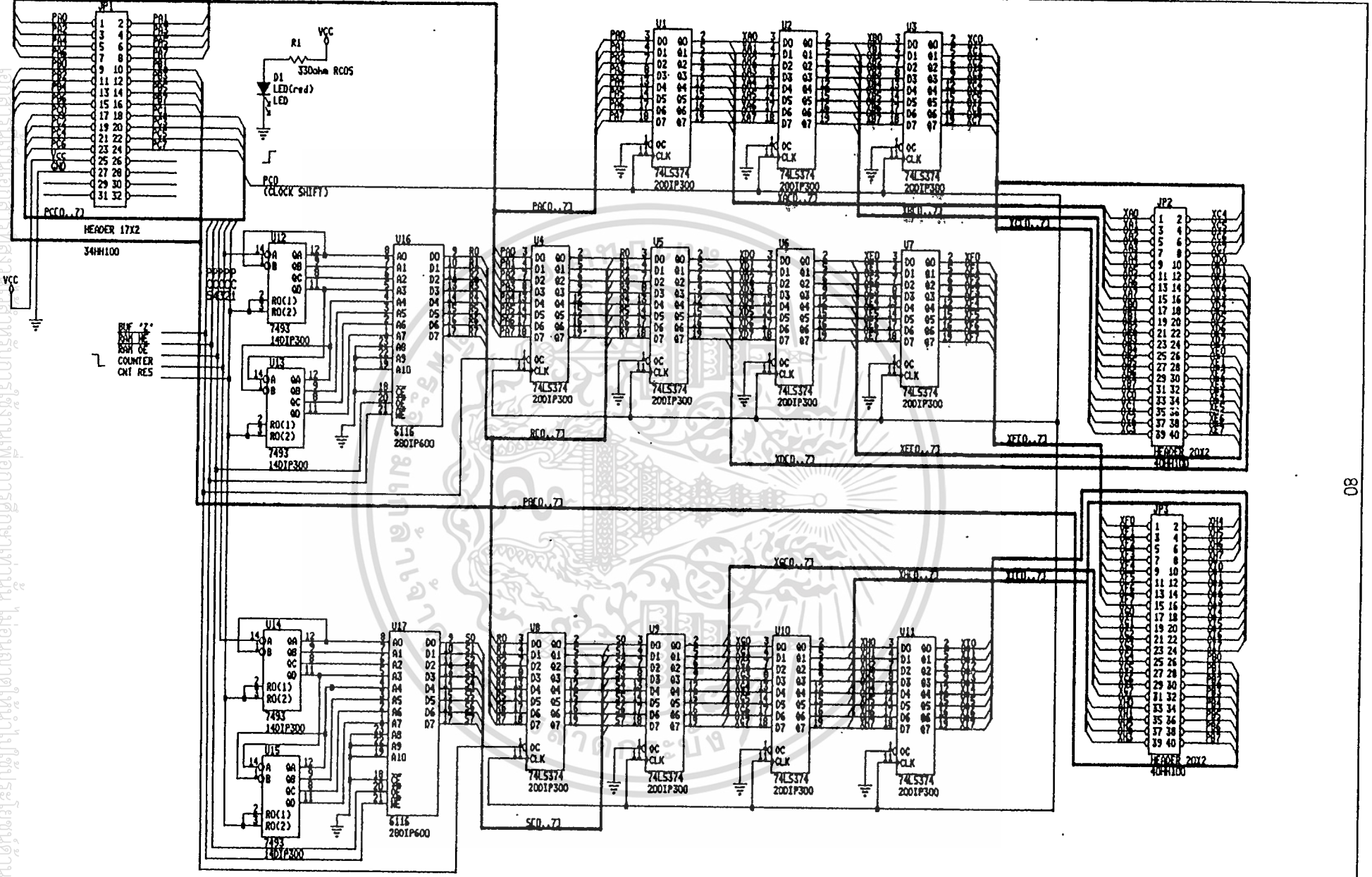
3. วงจรจัดลำดับสัญญาณ 2 มิติโดยใช้ตัวเลื่อนข้อมูลเป็นตัวหน่วงข้อมูล
   - ลายแผ่นวงจรพิมพ์ด้านอุปกรณ์
   - ลายแผ่นวงจรพิมพ์ด้านล่าง
   - ตำแหน่งการวางอุปกรณ์

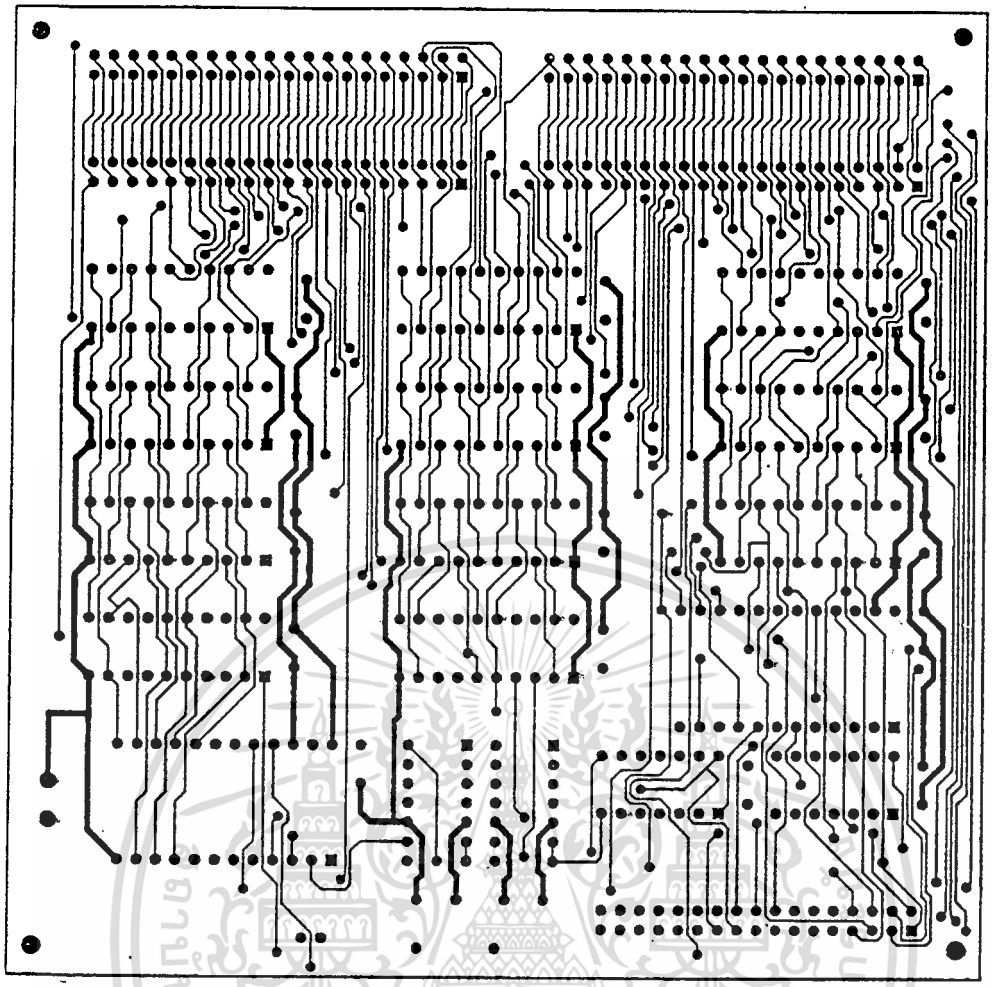
4. วงจรการกระจายทางคณิตศาสตร์แบบอนุกรม (1BAAT : 1 Bit At A Time)
   - ลายแผ่นวงจรพิมพ์ด้านล่างและด้านอุปกรณ์
   - ตำแหน่งการวางอุปกรณ์

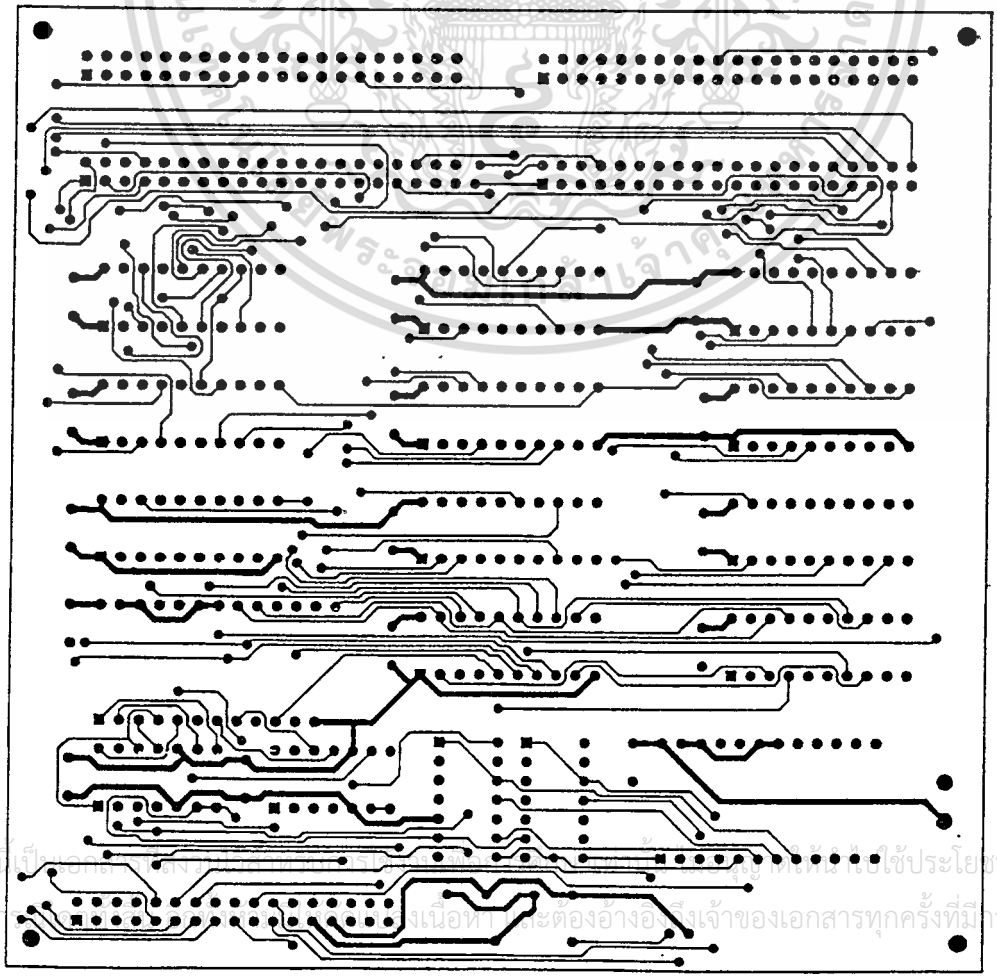
5. วงจรการกระจายทางคณิตศาสตร์แบบขนาน (DA Full Parallel)
   - ลายแผ่นวงจรพิมพ์ด้านอุปกรณ์และด้านล่าง
   - ตำแหน่งการวางอุปกรณ์

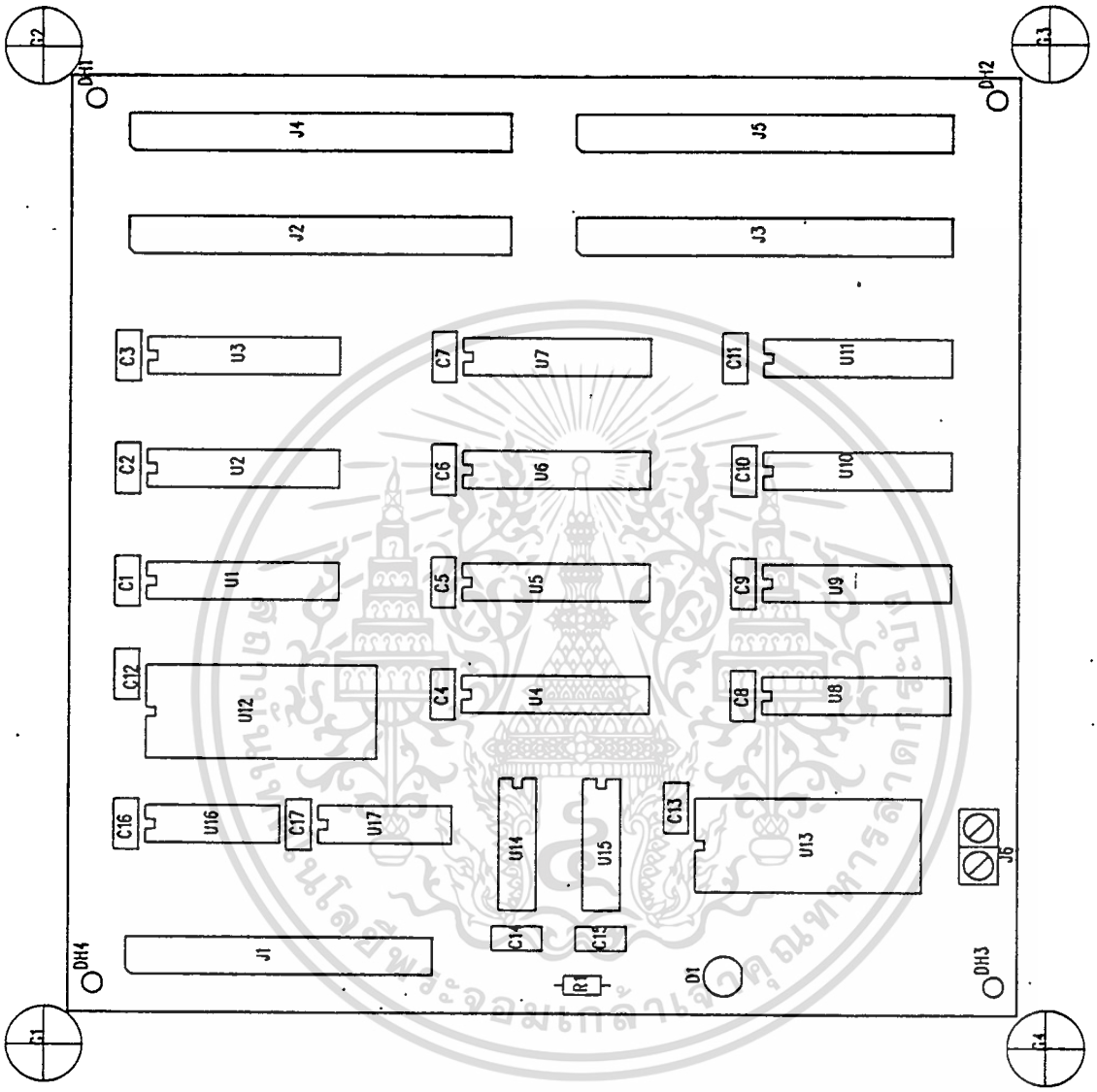
6. วงจรการกระจายทางคณิตศาสตร์ขนาด 1 บิท (Laplacian Filtering)

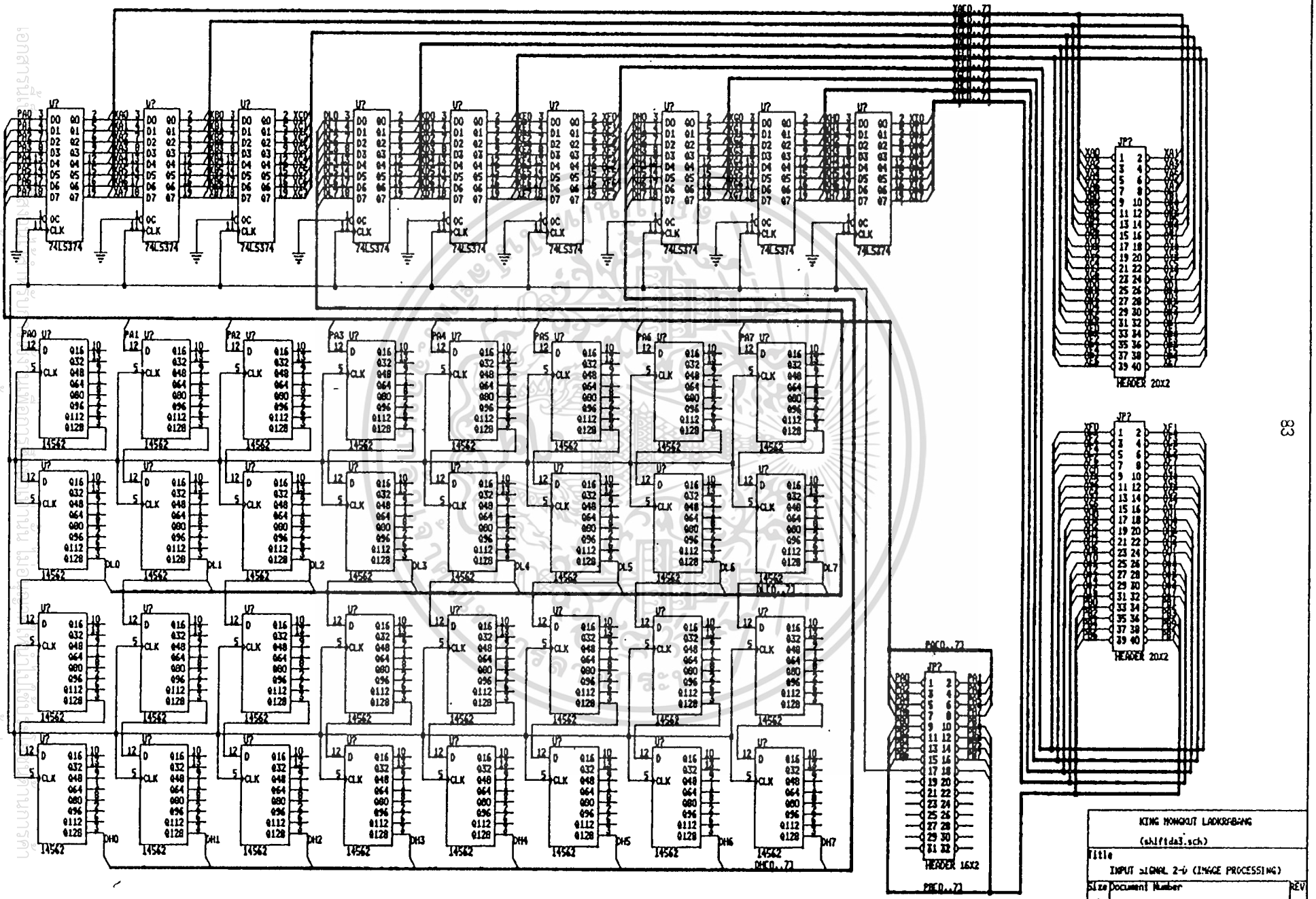


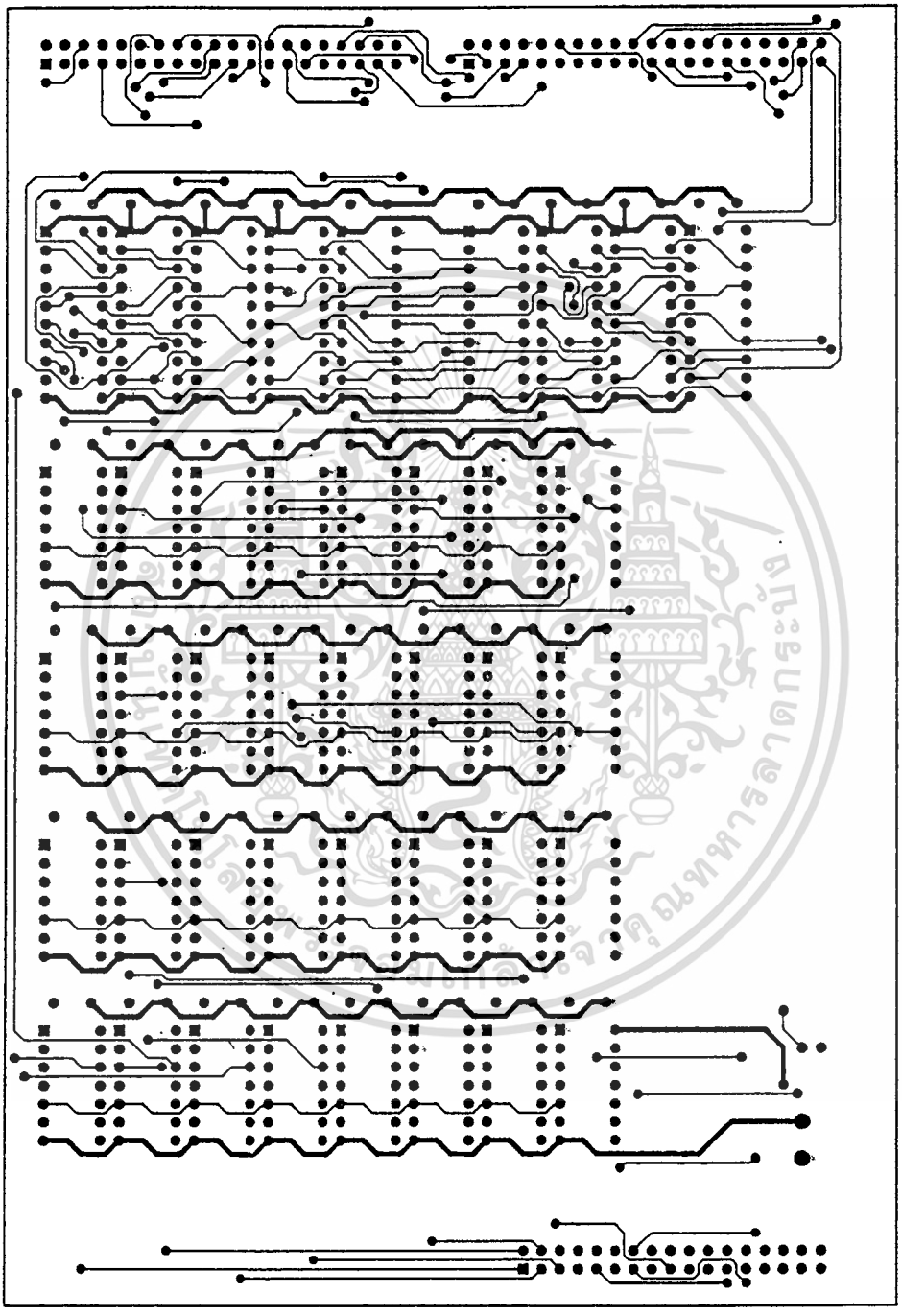

HEADER 17X2
34H100
R1
330ohm RC05
D1
LED(red)
LED
PCO
(CLOCK SHIFT)
PCO..72
VCC
BUF "2"
RAM WE
RAM OE
COUNTER
CNT RES
7493
140IP300
6116
280IP600
74LS374
20OIP300
PAO..72
RO..72
XDO..72
XEO..72
PBO..72
XGO..72
XHO..72
XIO..72
SO..72
HEADER 20X2
40H100
KING MONGKUTI LADKRABANG
SIGNAL INPUT 2-0 (IMAGE PROCESSOR)
Size
Document Number
REV
A
1
1

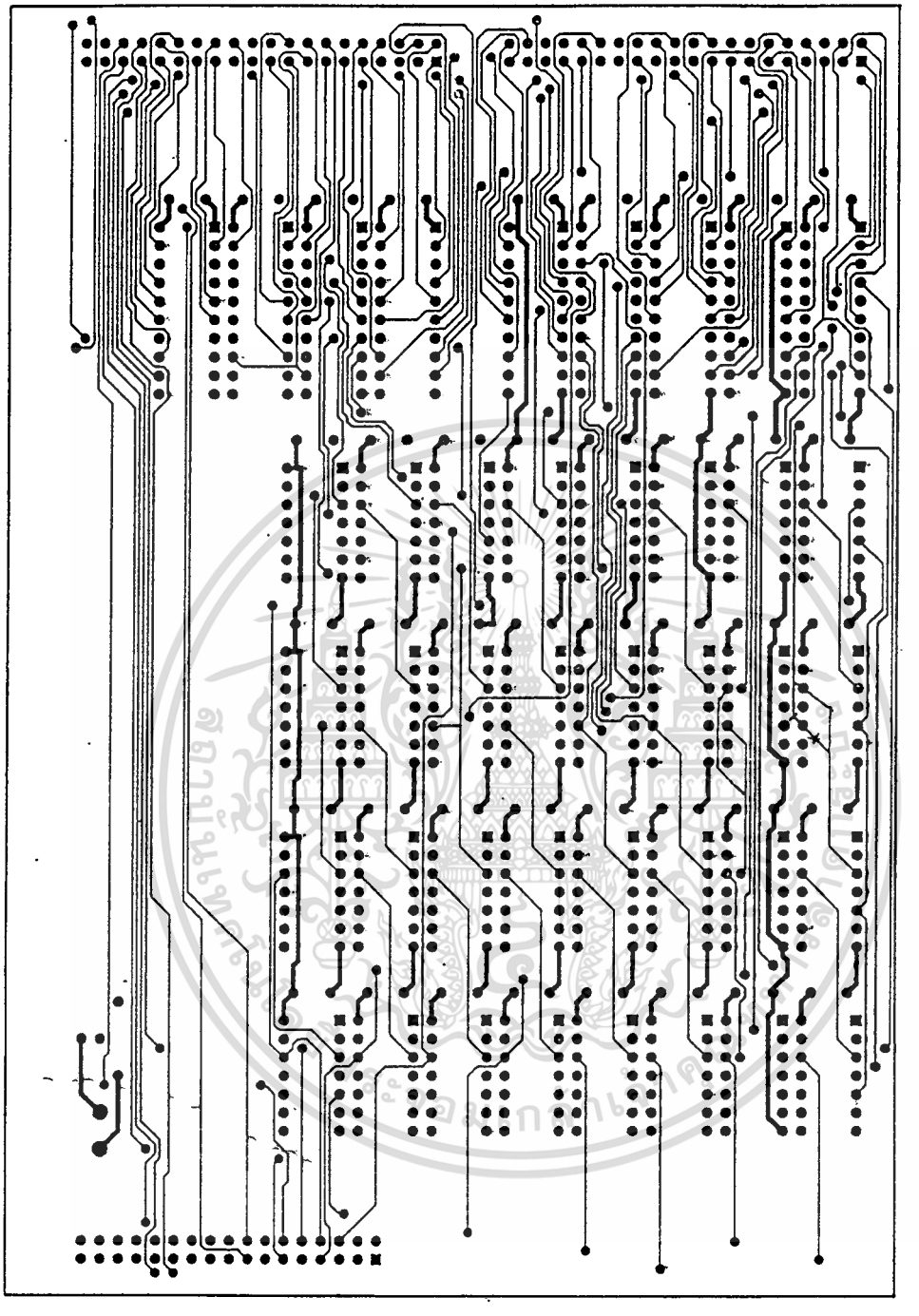


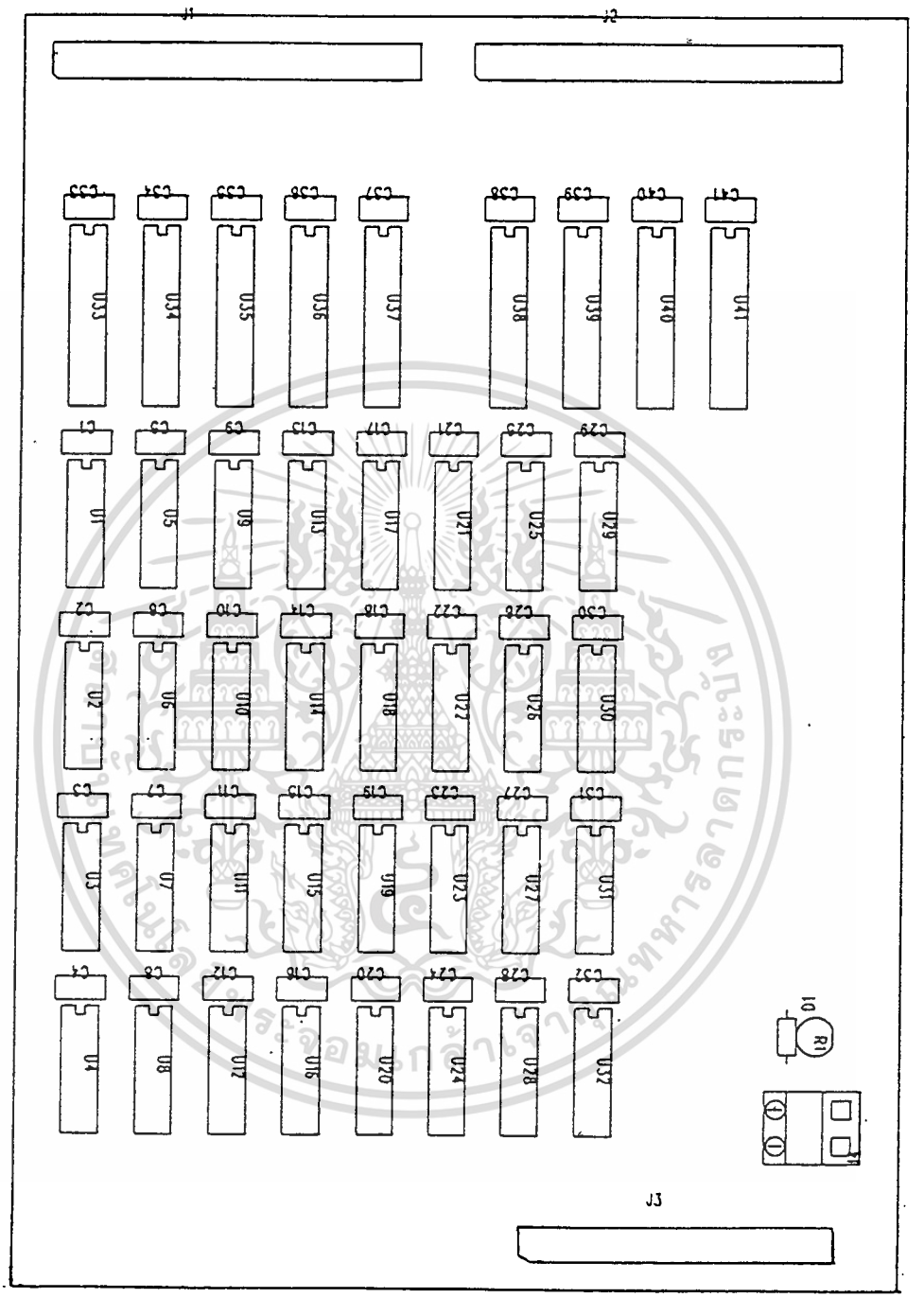


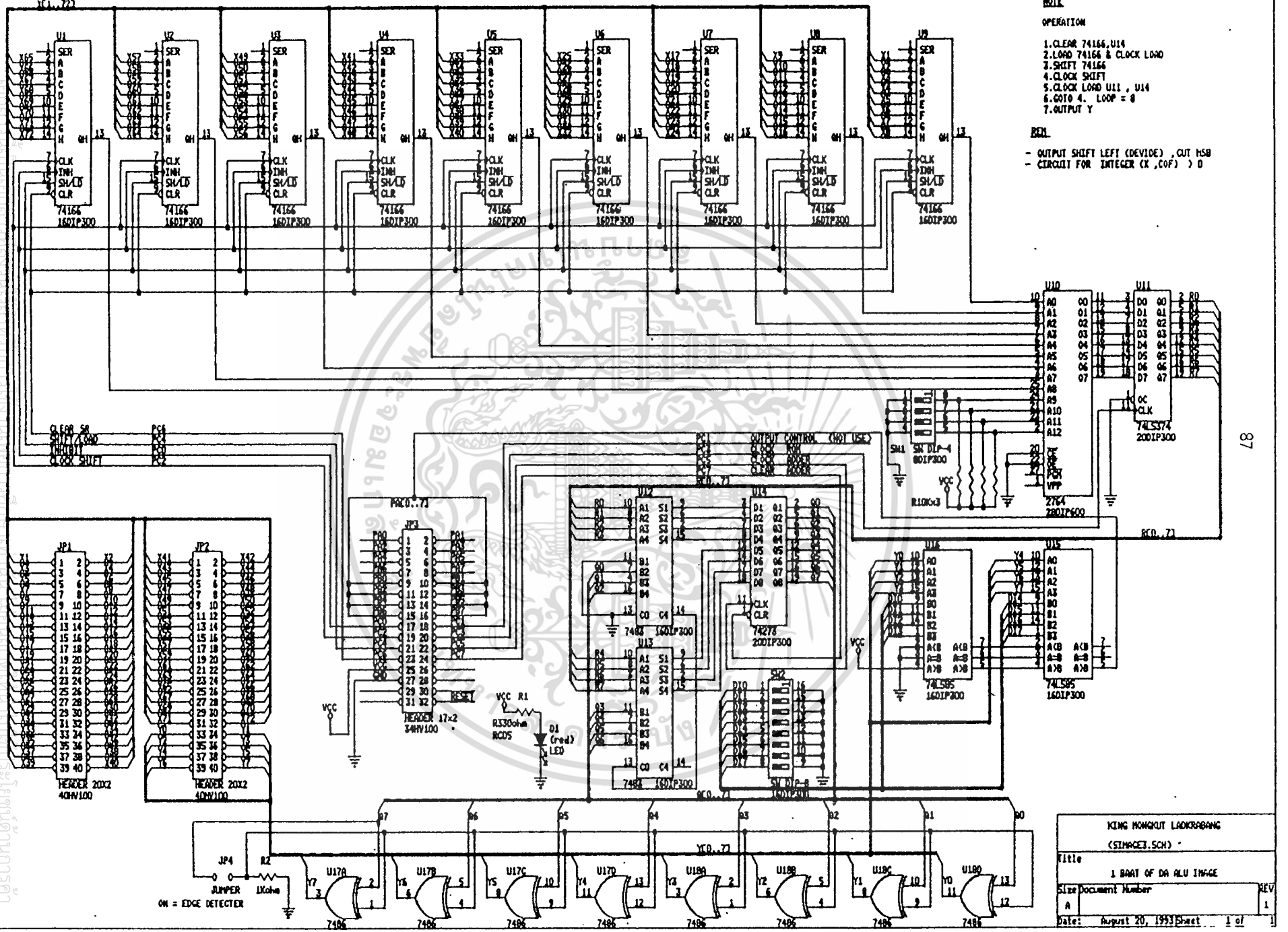


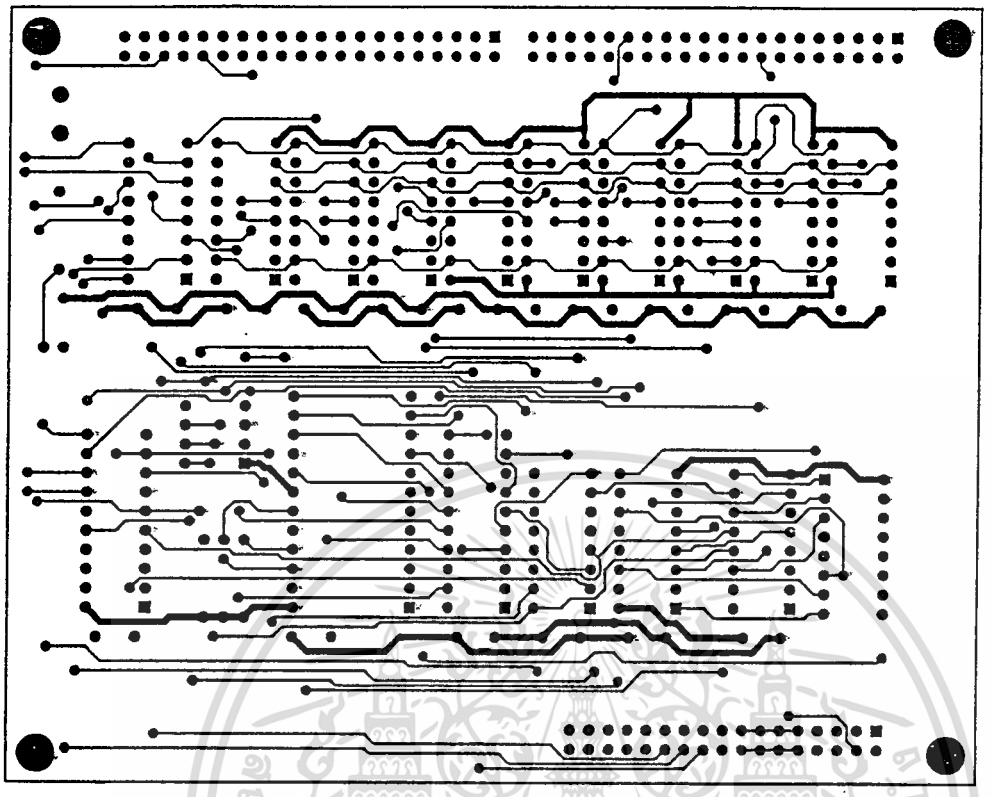

74LS374
14562
HEADER 20X2
HEADER 16X2
KING MONGKUT LADKRABANG
(shiftda3.sch)
Title
INPUT SIGNAL 2-D (IMAGE PROCESSING)
Size
Document Number
REV
A
1
1

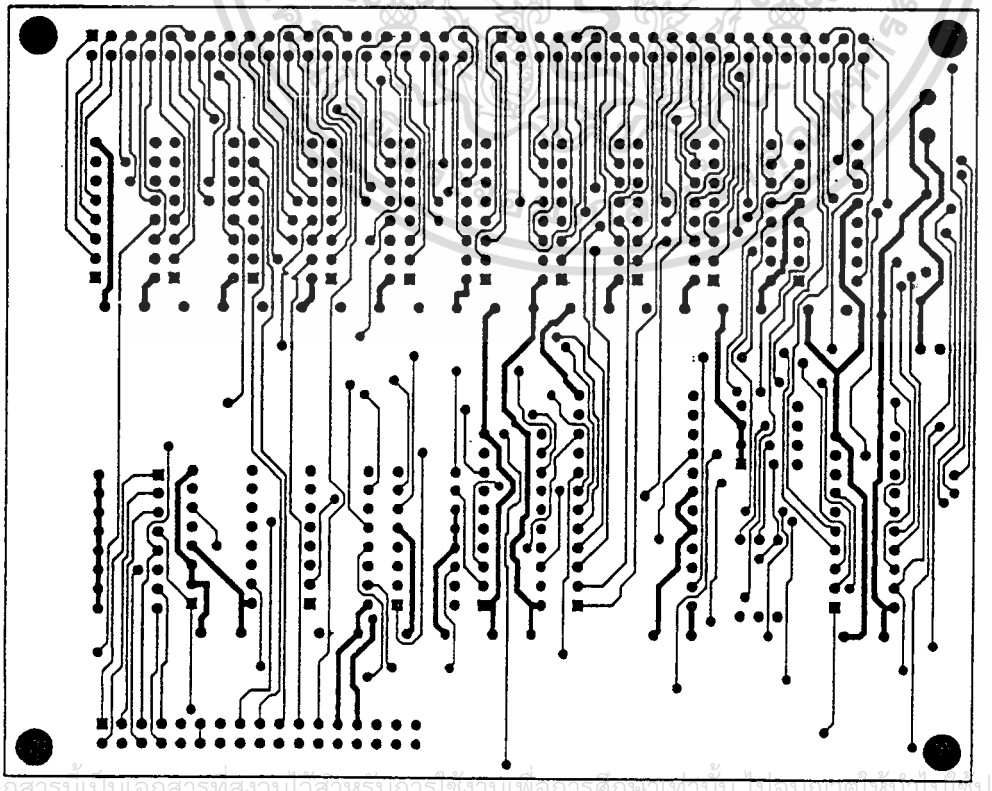


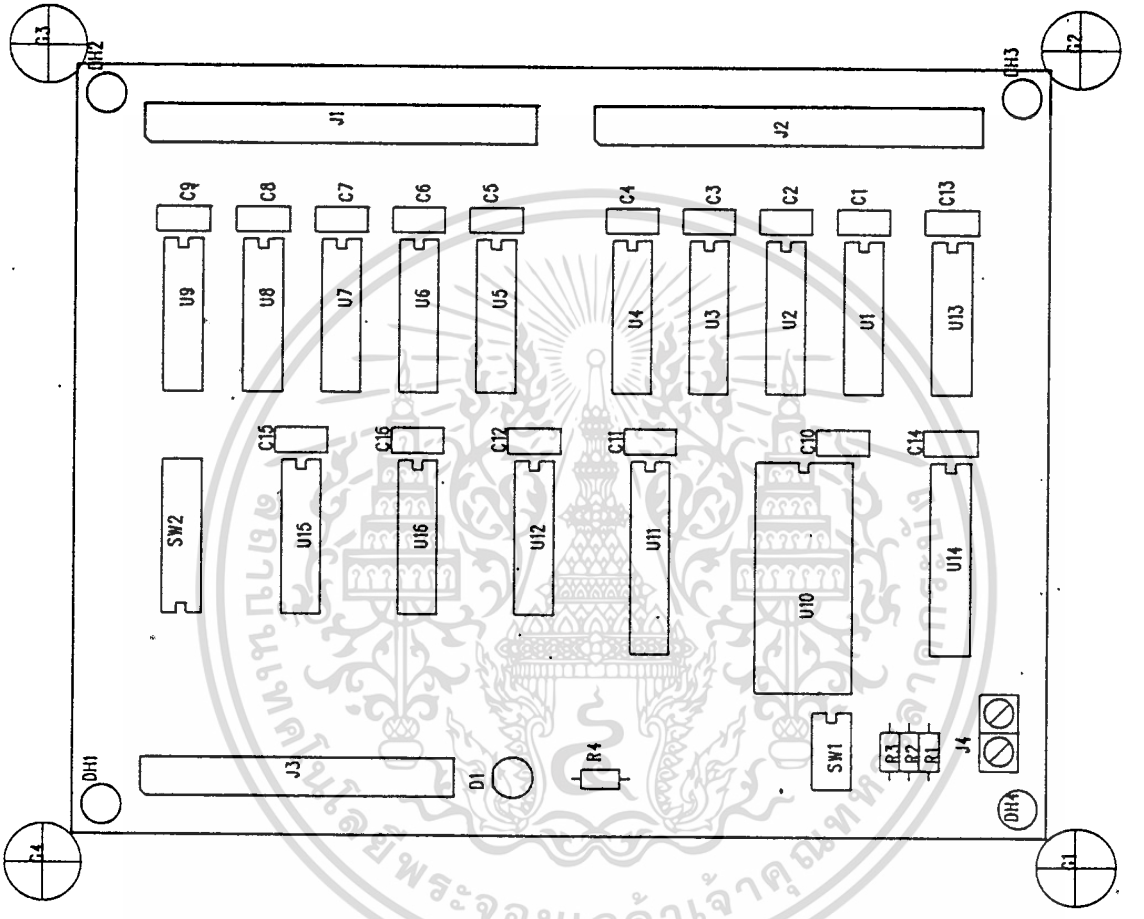


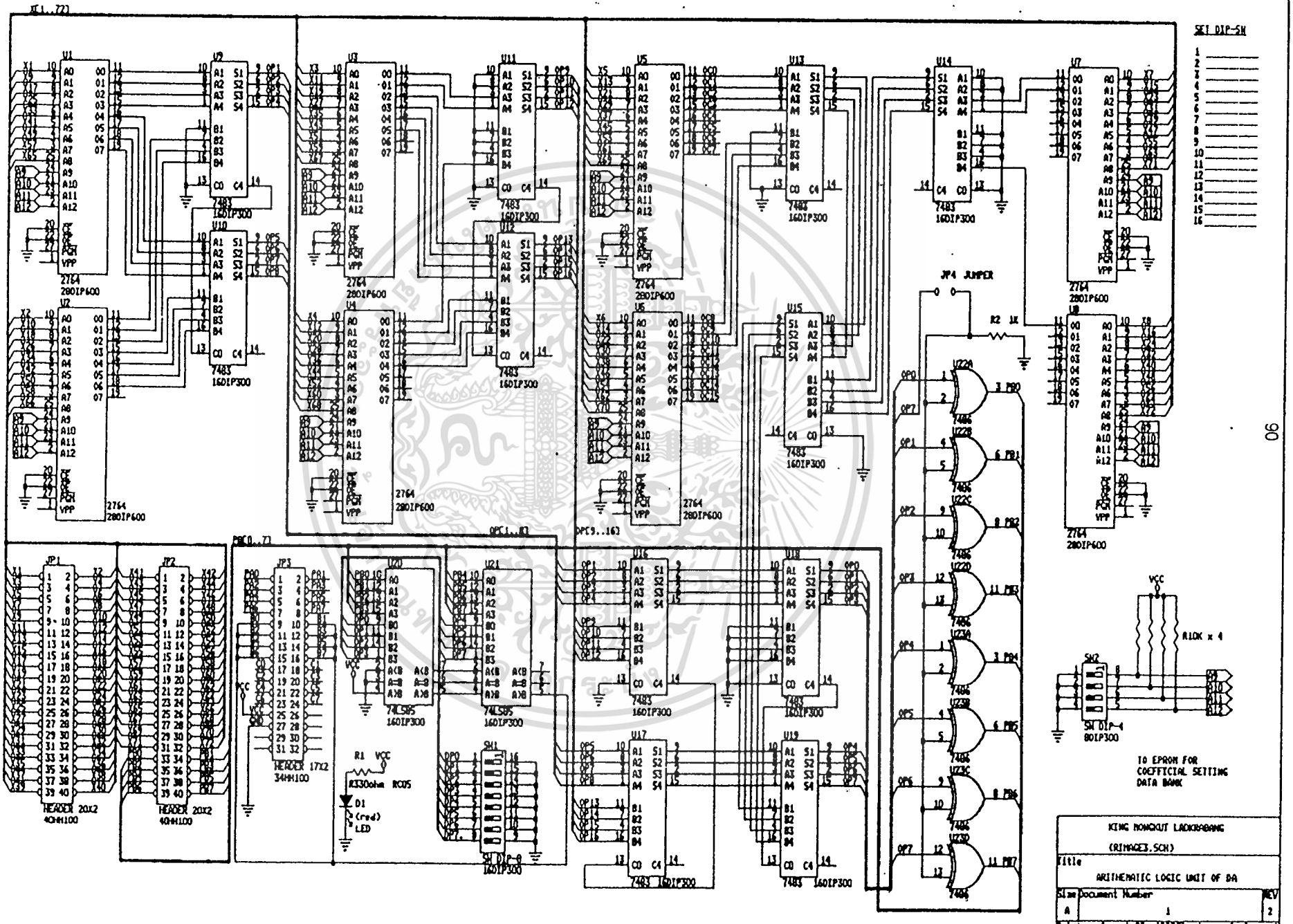


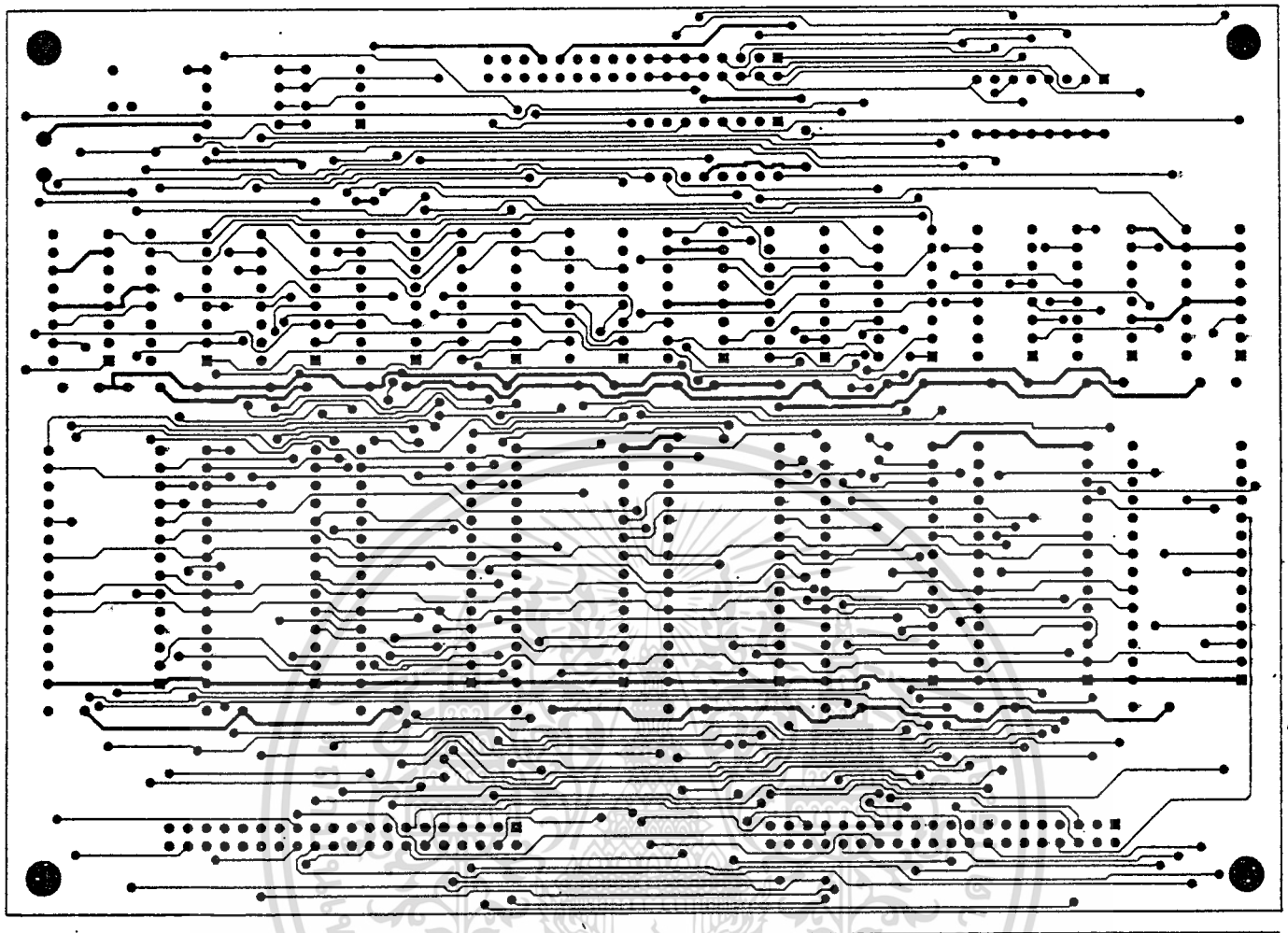

OPERATION
1.CLEAR 74166,U14
2.LOAD 74166 & CLOCK LOAD
3.SHIFT 74166
4.CLOCK SHIFT
5.CLOCK LOAD U11 , U14
6.GOTO 4. LOOP = 8
7.OUTPUT Y
REM
- OUTPUT SHIFT LEFT (DEVICE) ,OUT MSB
- CIRCUIT FOR INTEGER (X ,COF)) 0
74166 16DIP300
74LS374 20DIP300
2764 28DIP600
74273 20DIP300
74LS85 16DIP300
74LS86 14DIP300
SW DIP-4 8DIP300
SW DIP-8 16DIP300
RES 10K x 3
RES 330ohm
LED
HEADER 17x2 34HW100
HEADER 20X2 40HW100
JUMPER
1Kohm
ON = EDGE DETECTER
OUTPUT CONTROL (NOT USE)
CLEAR
CLOCK SHIFT
KING MONGKUT LADKRABANG
(SIMAGE3.SCH)
1 BAAT OF DA ALU IMAGE
Size A
Document Number
REV 1
Date: August 20, 1993
Sheet 1 of 1

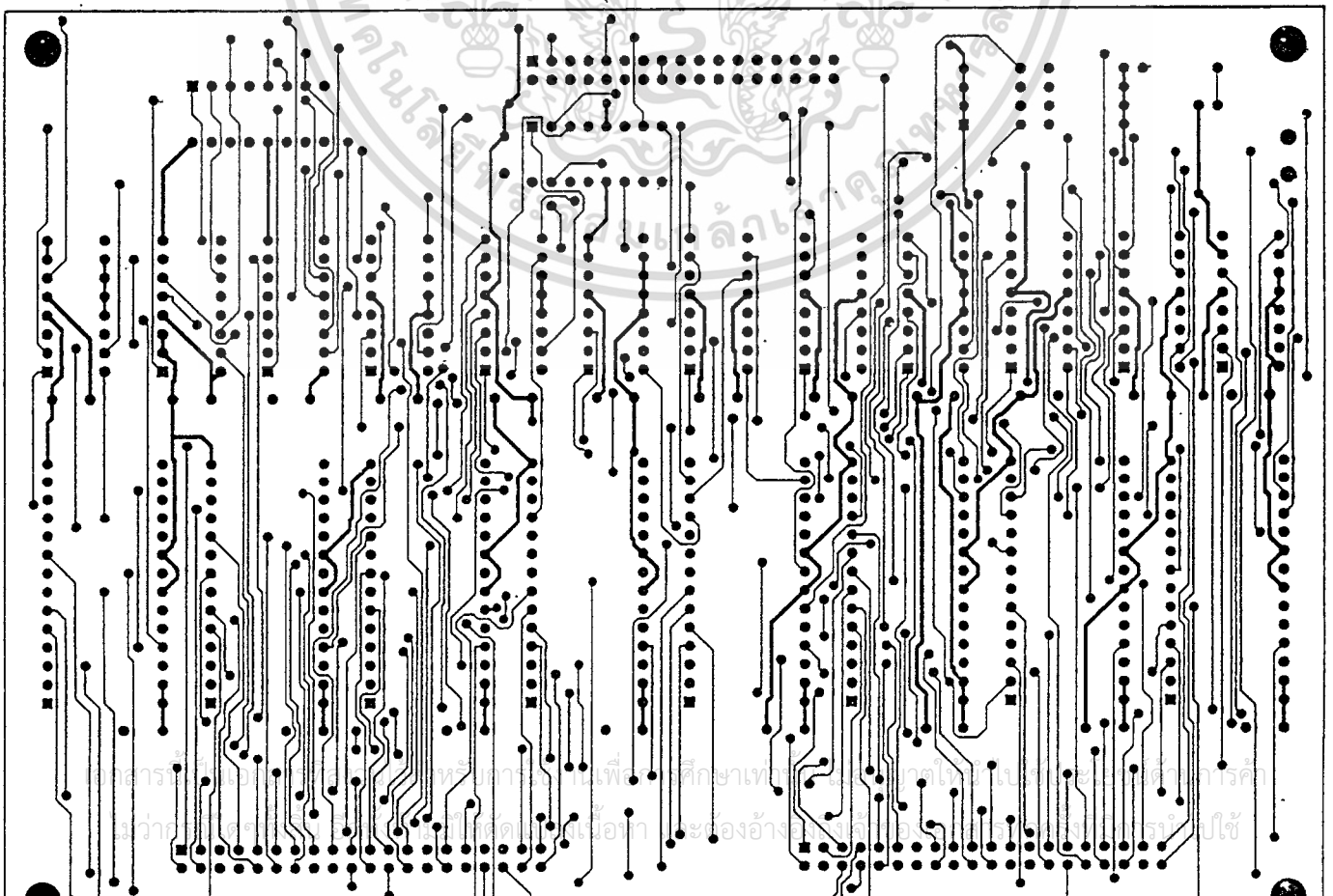


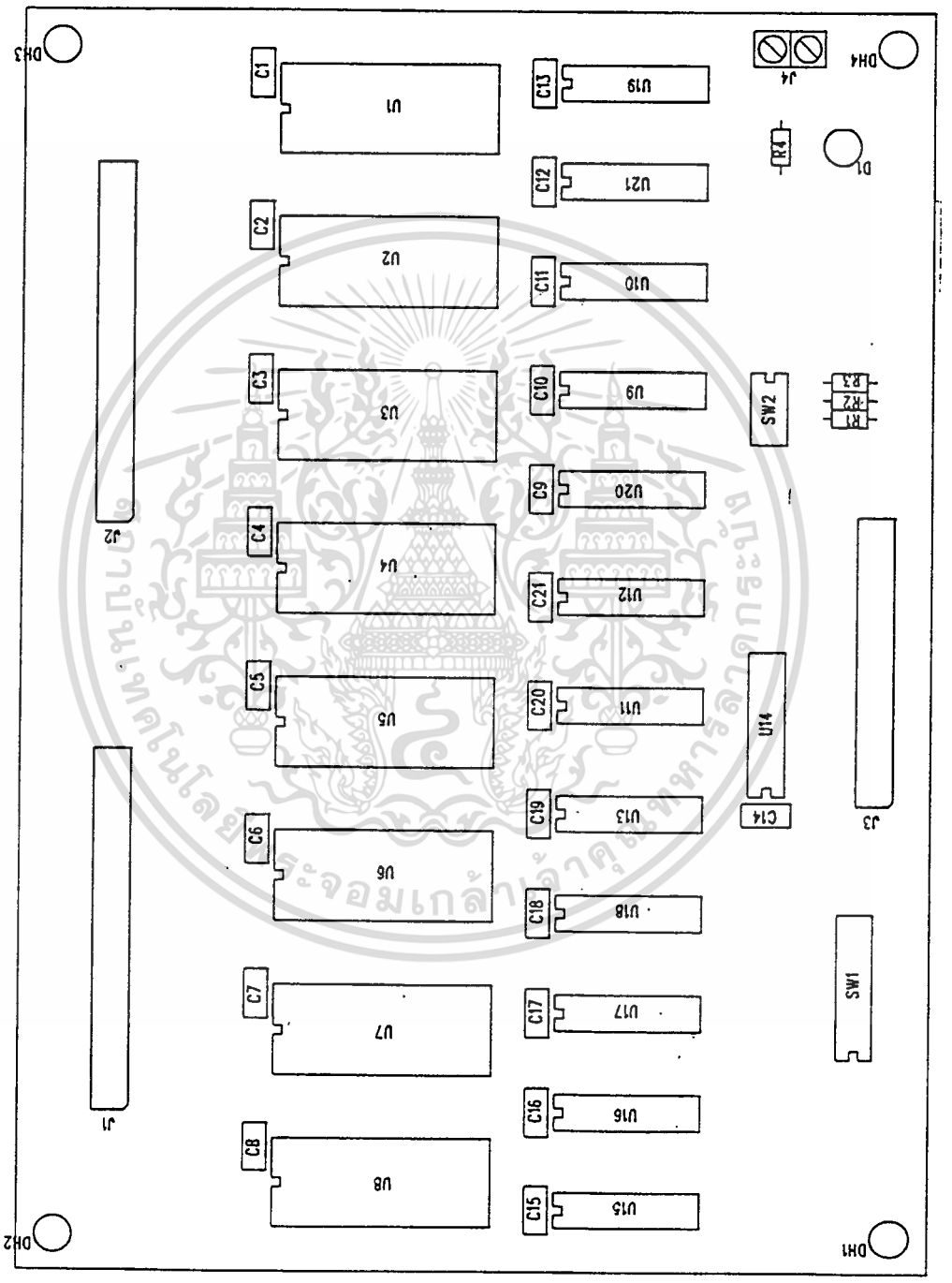


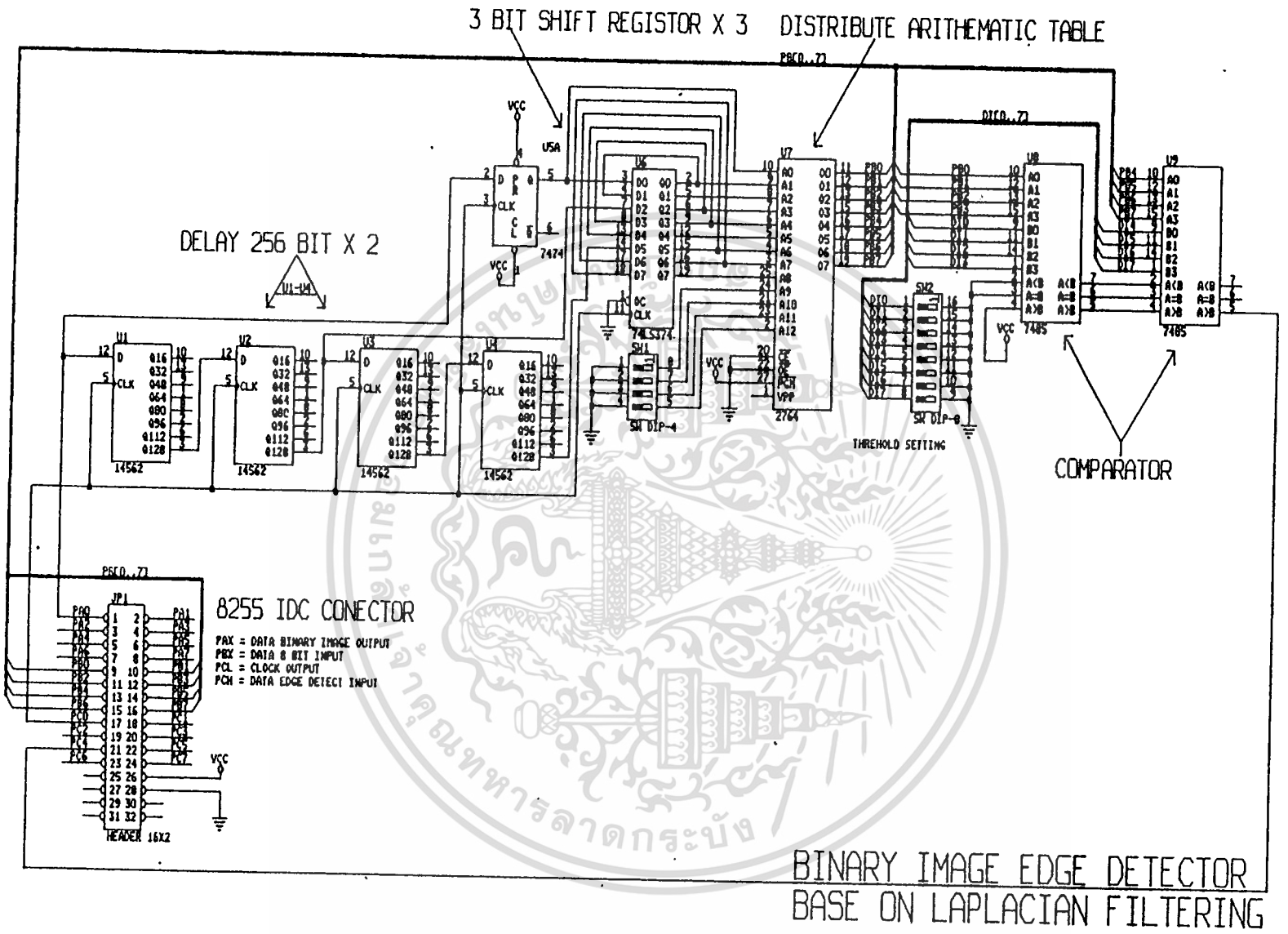



SW1 DIP-SW
JP4 JUMPER
R2 1K
RIOK x 4
SW DIP-4
8DIP300
TO EPROM FOR
COEFFICIAL SETTING
DATA BANK
HEADER 20X2
HEADER 17X2
R330ohm RC05
D1
(red)
LED
74LS85
16DIP300
2764
28DIP600
7483
16DIP300
7486
KING MONGKUT LADKRABANG
(RIMAGE3.SCH)
Title
ARITHEMATIC LOGIC UNIT OF BA
Size
Document Number
REV
A
1
2









KING MONGKUT LADKRABANG	
Title	BINARY IMAGE EDGE DETECTOR
Size: A	Document Number: ED1BIT2.SCH
Rev: 1	
Date: September 3, 1993	Sheet 1 of 1

ภาคผนวก

ภาคผนวก

MOTOROLA

MC14562B

128-BIT STATIC SHIFT REGISTER

The MC14562B is a 128-bit static shift register constructed with MOS P-channel and N-channel enhancement mode devices in a single monolithic structure. Data is clocked in and out of the shift register on the positive edge of the clock input. Data outputs are available every 16 bits, from 16 through bit 128. This complementary MOS shift register is primarily used where low power dissipation and/or high noise immunity is desired.

- Diode Protection on All Inputs
- Noise Immunity = 45% of V_{DD} typical
- Single Supply Operation – Positive or Negative
- Fully Static Operation
- Exceedingly Slow Input Transition Rates May Be Applied to the Clock Input
- 5.6 MHz Operation @ V_{DD} = 10 Vdc
- Cascadable to Provide Longer Shift Register Lengths – 1.5 MHz Operation @ V_{DD} = 10 Vdc
- Supply Voltage Range = 3.0 Vdc to 18 Vdc
- Capable of Driving Two Low-power TTL Loads, One Low-power Schottky TTL Load or Two HTL Loads Over the Rated Temperature Range

CMOS LSI

(LOW-POWER COMPLEMENTARY MOS)

128-BIT STATIC SHIFT REGISTER.

L SUFFIX
CERAMIC PACKAGE
CASE 632

P SUFFIX
PLASTIC PACKAGE
CASE 646

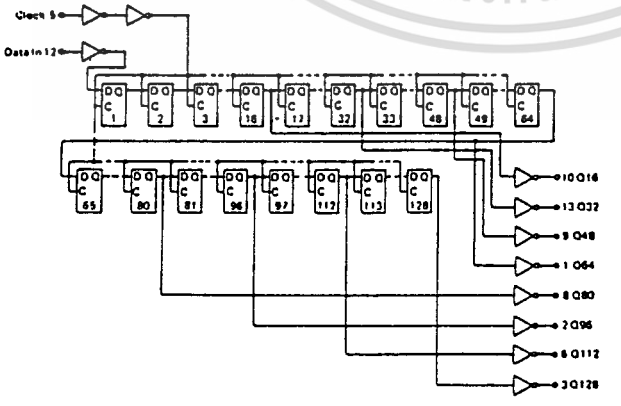
ORDERING INFORMATION

MC14XXXB — Suffix Denotes
- L Ceramic Package
- P Plastic Package
- A Extended Operating Temperature Range
- C Limited Operating Temperature Range

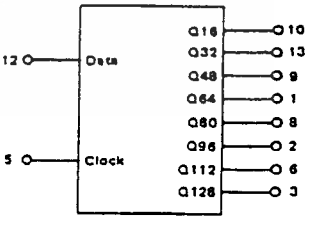
MAXIMUM RATINGS (Voltages referenced to V_{SS})

Rating	Symbol	Value	Unit
DC Supply Voltage	V_{DD}	-0.5 to +18	Vdc
Input Voltage, All Inputs	V_{in}	-0.5 to V_{DD} + 0.5	Vdc
DC Current Drain per Pin	I	10	mAdc
Operating Temperature Range – AL Device CL/CP Device	T_A	-55 to +125 -40 to +85	°C
Storage Temperature Range	T_{stg}	-65 to +150	°C

LOGIC DIAGRAM



BLOCK DIAGRAM



Pins 4 and 11 not used.

V_{DD} = Pin 14
V_{SS} = Pin 7

ELECTRICAL CHARACTERISTICS

Characteristic	Symbol	V_{DD} Vdc	T_{low}* Min	T_{low}* Max	25°C Min	25°C Typ	25°C Max	T_{high}* Min	T_{high}* Max	Unit
Output Voltage "0" Level V_{in} = V_{DD} or 0	V_{OL}	5.0	–	0.05	–	0	0.05	–	0.05	Vdc
		10	–	0.05	–	0	0.05	–	0.05	
		15	–	0.05	–	0	0.05	–	0.05	
"1" Level V_{in} = 0 or V_{DD}	V_{OH}	5.0	4.95	–	4.95	5.0	–	4.95	–	Vdc
		10	9.95	–	9.95	10	–	9.95	–	
		15	14.95	–	14.95	15	–	14.95	–	
Input Voltage# "0" Level	V_{IL}									Vdc
(V_O = 4.5 or 0.5 Vdc)		5.0	–	1.5	–	2.25	1.5	–	1.5	
(V_O = 9.0 or 1.0 Vdc)		10	–	3.0	–	4.50	3.0	–	3.0	
(V_O = 13.5 or 1.5 Vdc)		15	–	4.0	–	6.75	4.0	–	4.0	
"1" Level	V_{IH}									Vdc
(V_O = 0.5 or 4.5 Vdc)		5.0	3.5	–	3.5	2.75	–	3.5	–	
(V_O = 1.0 or 9.0 Vdc)		10	7.0	–	7.0	5.50	–	7.0	–	
(V_O = 1.5 or 13.5 Vdc)		15	11.0	–	11.0	8.25	–	11.0	–	
Output Drive Current (AL Device)	I_{OH}									mAdc
(V_{OH} = 2.5 Vdc) Source		5.0	–1.2	–	–1.0	–1.7	–	–0.7	–	
(V_{OH} = 4.6 Vdc)		5.0	–0.25	–	–0.2	–0.36	–	–0.14	–	
(V_{OH} = 9.5 Vdc)		10	–0.62	–	–0.5	–0.9	–	–0.35	–	
(V_{OH} = 13.5 Vdc)		15	–1.8	–	–1.5	–3.5	–	–1.1	–	
(V_{OL} = 0.4 Vdc) Sink	I_{OL}	5.0	0.64	–	0.51	0.88	–	0.36	–	mAdc
(V_{OL} = 0.5 Vdc)		10	1.6	–	1.3	2.25	–	0.9	–	
(V_{OL} = 1.5 Vdc)		15	4.2	–	3.4	8.8	–	2.4	–	
Output Drive Current (CL/CP Device)	I_{OH}									mAdc
(V_{OH} = 2.5 Vdc) Source		5.0	–1.0	–	–0.8	–1.7	–	–0.6	–	
(V_{OH} = 4.6 Vdc)		5.0	–0.2	–	–0.16	–0.36	–	–0.12	–	
(V_{OH} = 9.5 Vdc)		10	–0.5	–	–0.4	–0.9	–	–0.3	–	
(V_{OH} = 13.5 Vdc)		15	–1.4	–	–1.2	–3.5	–	–1.0	–	
(V_{OL} = 0.4 Vdc) Sink	I_{OL}	5.0	0.52	–	0.44	0.88	–	0.36	–	mAdc
(V_{OL} = 0.5 Vdc)		10	1.3	–	1.1	2.25	–	0.9	–	
(V_{OL} = 1.5 Vdc)		15	3.6	–	3.0	8.8	–	2.4	–	
Input Current (AL Device)	I_{in}	15	–	±0.1	–	±0.00001	±0.1	–	±1.0	μAdc
Input Current (CL/CP Device)	I_{in}	15	–	±0.3	–	±0.00001	±0.3	–	±1.0	μAdc
Input Capacitance (V_{in} = 0)	C_{in}	–	–	–	–	5.0	7.5	–	–	pF
Quiescent Current (AL Device) (Per Package)	I_{DD}	5.0	–	5.0	–	0.010	5.0	–	150	μAdc
		10	–	10	–	0.020	10	–	300	
		15	–	20	–	0.030	20	–	600	
Quiescent Current (CL/CP Device) (Per Package)	I_{DD}	5.0	–	50	–	0.010	50	–	375	μAdc
		10	–	100	–	0.020	100	–	750	
		15	–	200	–	0.030	200	–	1500	
Total Supply Current**†. (Dynamic plus Quiescent, Per Package) (C_L = 50 pF on all outputs, all buffers switching)	I_T	5.0	I_T = (1.94 μA/kHz) f + I_{DD}							μAdc
		10	I_T = (3.81 μA/kHz) f + I_{DD}							
		15	I_T = (5.62 μA/kHz) f + I_{DD}							

*T_{low} = -55°C for AL Device, -40°C for CL/CP Device.
T_{high} = +125°C for AL Device, +85°C for CL/CP Device.
#Noise immunity specified for worst-case input combination.
Noise Margin for both "1" and "0" level = 1.0 Vdc min @ V_{DD} = 5.0 Vdc
2.0 Vdc min @ V_{DD} = 10 Vdc
2.5 Vdc min @ V_{DD} = 15 Vdc

†To calculate total supply current at loads other than 50 pF:
$I_T(C_L) = I_T(50\ pF) + 4 \times 10^{-3} (C_L - 50) V_{DD} f$
where: I_T is in μA (per package), C_L in pF, V_{DD} in Vdc, and f in kHz is input frequency.
**The formulas given are for the typical characteristics only at 25°C.

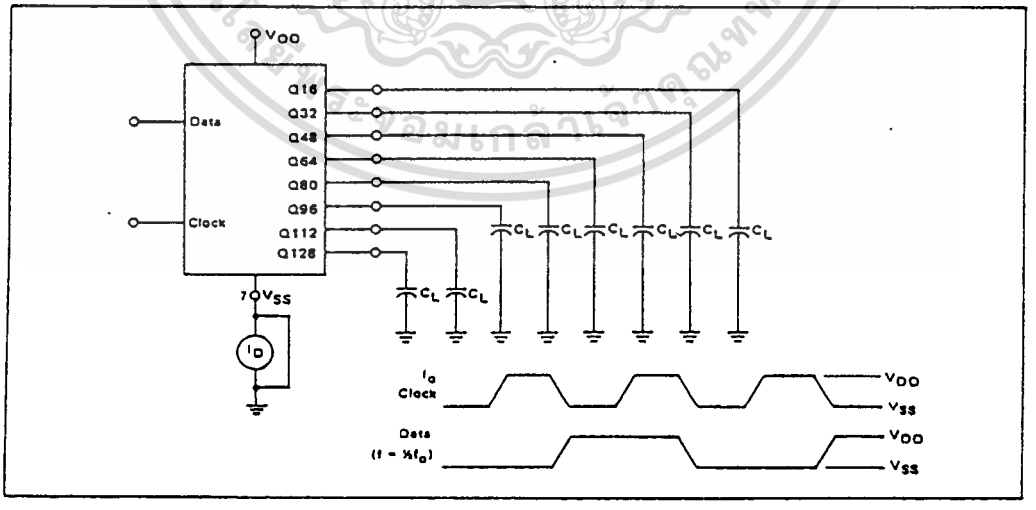
This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range $V_{SS} \le (V_{in}$ or $V_{out}) \le V_{DD}$.
Unused inputs must always be tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

SWITCHING CHARACTERISTICS* (C_L = 50 pF, T_A = 25°C)

Characteristic	Symbol	V_{DD}	Min	Typ	Max	Unit
Output Rise Time	t_{TLH}					ns
t_{TLH} = (3.0 ns/pF) C_L + 30 ns		5.0	–	180	360	
t_{TLH} = (1.5 ns/pF) C_L + 15 ns		10	–	90	180	
t_{TLH} = (1.1 ns/pF) C_L + 10 ns		15	–	65	130	
Output Fall Time	t_{THL}					ns
t_{THL} = (1.5 ns/pF) C_L + 25 ns		5.0	–	100	200	
t_{THL} = (0.75 ns/pF) C_L + 12.5 ns		10	–	50	100	
t_{THL} = (0.55 ns/pF) C_L + 9.5 ns		15	–	40	80	
Propagation Delay Time	t_{PLH}, t_{PHL}					ns
Clock to Q						
t_{PLH}, t_{PHL} = (1.7 ns/pF) C_L + 515 ns		5.0	–	600	1200	
t_{PLH}, t_{PHL} = (0.66 ns/pF) C_L + 217 ns		10	–	250	500	
t_{PLH}, t_{PHL} = (0.5 ns/pF) C_L + 145 ns		15	–	170	340	
Clock Pulse Width	t_{WH}	5.0	600	300	–	ns
(50% Duty Cycle)		10	220	110	–	
		15	150	75	–	
Clock Pulse Frequency	f_{cl}	5.0	–	1.9	1.1	MHz
		10	–	5.6	3.0	
		15	–	8.0	4.0	
Data to Clock Setup Time	$t_{su(1)}$	5.0	-20	-170	–	ns
		10	-10	-64	–	
		15	0	-60	–	
	$t_{su(0)}$	5.0	-20	-91	–	ns
		10	-10	-58	–	
		15	0	-48	–	
Data to Clock Hold Time	$t_{h(1)}$	5.0	350	263	–	ns
		10	165	109	–	
		15	155	100	–	
	$t_{h(0)}$	5.0	350	267	–	ns
		10	200	140	–	
		15	140	93	–	

*The formula given is for the typical characteristics only.

FIGURE 1 – POWER DISSIPATION TEST CIRCUIT AND WAVEFORMS



5483/7483 Single 4-Bit Binary Full Adder (Look Ahead Carry)

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF
T.I.											SN54LS83A	J①			W①	SN5483A	J①			W①					
											SN74LS83A	J①N①				SN7483A	J①N①								
FAIRCHILD											FM54LS83/FM93LS83	D①			F①	FM5483/FM9383	D①			F①					
											FC74LS83/FC93LS83	D①	P①		F①	FC7483/FC9383	D①	P①							
MOTOROLA																MC5483	L①								
											SN74LS83		P①			MC7483	L①	P①							
N. S. C.											DM54LS83A		J①			DM5483	J①			W①					
											DM74LS83A		J①			DM7483	J①N①								
PHILIPS																									
											N74LS83		J①			FJH211/7483		J①							
SIGNETICS																S5483	F①	B①		W①					
											N74LS83		A①			N7483	F①	B①							
SIEMENS																FLH241		J①							
FUJITSU																									
HITACHI											HD74LS83		P①			HD7483A/HD2535	J①	P①							
MITSUBISHI											M74LS83		P①			M53283		P①							
NEC																μPB230	D①								
TOSHIBA																									

Electrical Characteristics SN54LS83A/SN74LS83A

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7V	Operating free-air temperature range	SN54LS	−55°C to	125°C
Input voltage	7V		SN74LS	0°C to	70°C
Interemitter voltage (see Note 1)	5.5V	Storage temperature range		−65°C to	150°C

recommended operating conditions

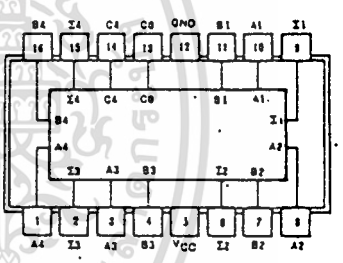
	SN54LS83A			SN74LS83A			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply Voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range

PARAMETER*			TEST CONDITIONS†		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage				2			V
V_{IL}	Low-level input voltage						0.8	V
V_I	Input clamp voltage		V_{CC} = MIN, I_I = −18mA				−1.5	V
V_{OH}	High-level output voltage		V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8V, I_{OH} = MAX		2.7	3.4		V
V_{OL}	Low-level output voltage		V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8V, I_{OL} = 4mA			0.25	0.4	V
I_I	Input current at maximum input voltage	Any A or B	V_{CC} = MAX, V_I = 7V				0.2	mA
		C0					0.1	
I_{IH}	High-level input current	Any A or B	V_{CC} = MAX, V_I = 2.7V				40	μA
		C0					20	
I_{IL}	Low-level input current	Any A or B	V_{CC} = MAX, V_I = 0.4V				−0.8	mA
		C0					−0.4	
I_{OS}	Short-circuit output current*		V_{CC} = MAX		−20		−100	mA
I_{CC}	Supply current		V_{CC} = MAX, Outputs open	All B low, other inputs at 4.5V				mA
				All inputs at 4.5V SN74 LS		19	34	
t_{PLH}	from C0 to output Any Σ		V_{CC} = 5V, C_L = 15pF, T_A = 25°C, R_L = 2kΩ			16	24	ns
t_{PHL}						15	24	
t_{PLH}	from A_i or B_i to output Σ_i					15	24	ns
t_{PHL}						15	24	
t_{PLH}	from C0 to output C4					11	17	ns
t_{PHL}						15	22	
t_{PLH}	from A_i or B_i to output C4					11	17	ns
t_{PHL}						12	17	

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
‡ All typical values are at V_{CC} = 5V, T_A = 25°C.
* Only one output should be shorted at a time.
• t_{PLH} = propagation delay time, low-to-high-level output
t_{PHL} = propagation delay time, high-to-low-level output

Pin Assignment (Top View)



positive logic: see function table

Function Table

'83A, 'LS83 (see Note 2)

INPUT					OUTPUT					
					WHEN C0 = L			WHEN C0 = H		
					Σ1 = L		C2 = L	Σ1 = H		C2 = H
A1 (A3)	B1 (B3)	A2 (A4)	B2 (B4)		Σ1 (Σ3)	Σ2 (Σ4)	C2 (C4)	Σ1 (Σ3)	Σ2 (Σ4)	C2 (C4)
L	L	L	L		L	L	L	H	L	L
H	L	L	L		H	L	L	L	H	L
L	H	L	L		H	L	L	H	H	L
H	H	L	L		L	H	L	H	H	L
L	L	H	L		H	H	L	H	H	L
H	L	H	L		H	H	L	L	L	H
L	H	H	L		H	H	L	H	L	H
H	H	H	L		L	L	H	H	L	H
L	L	L	H		L	L	H	H	H	L
H	L	L	H		H	L	H	L	H	H
L	H	L	H		H	H	L	L	L	H
H	H	L	H		L	L	H	L	L	H
L	L	H	H		L	L	H	H	L	H
H	L	H	H		H	H	L	L	H	H
L	H	H	H		H	H	L	H	H	H
H	H	H	H		L	H	H	H	H	H

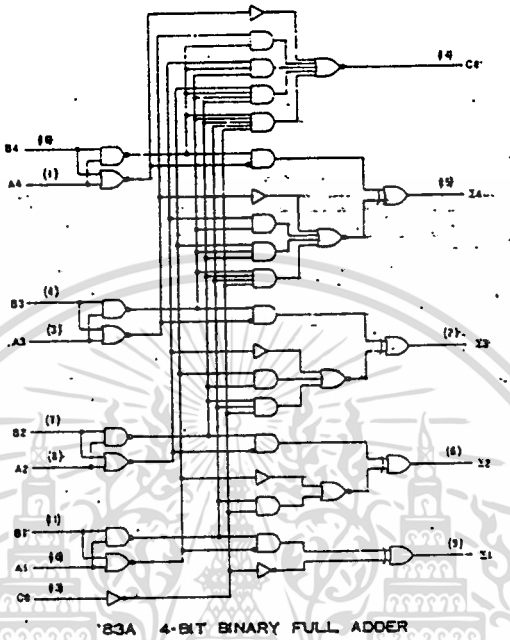
H = high level, L = low level

NOTES: 1. This is the voltage between two emitters of a multiple-emitter transistor. For the 83A, this rating applies between the following pairs: A1 and B1, A2 and B2, A3 and B3, A4 and B4.
2. Input conditions at A1, B1, A2, B2, and C0 are used to determine outputs Σ1 and Σ2 and the value of the internal carry C2. The values at C2, A3, B3, A4, and B4 are then used to determine outputs Σ3, Σ4, and C4.

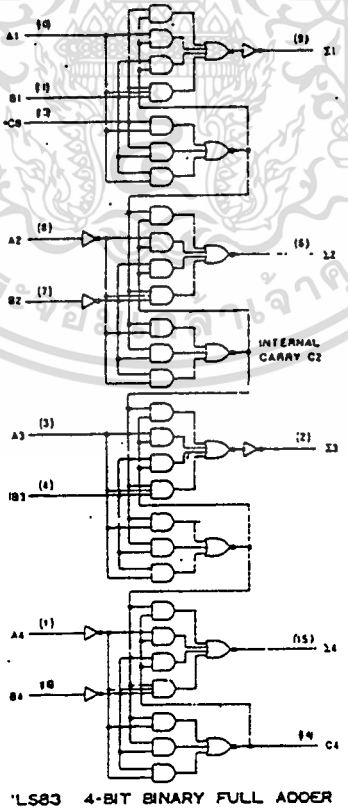
—CONTINUED ON NEXT PAGE

5483/7483(CONTINUED)

Functional Block Diagrams



'83A 4-BIT BINARY FULL ADDER



'LS83 4-BIT BINARY FULL ADDER

National Semiconductor

F100181
4-Bit Binary/BCD Arithmetic Logic Unit

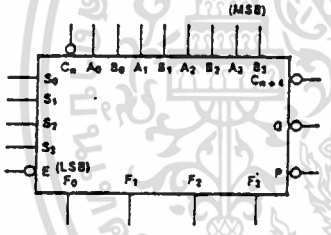
General Description

The F100181 performs eight logic operations and eight arithmetic operations on a pair of 4-bit words. The operating mode is determined by signals applied to the Select (S_n) inputs, as shown in the Function Select table. In addition to performing binary arithmetic, the circuit contains the necessary correction logic to perform BCD addition and subtraction. Output latches are provided to reduce overall package count and increase system operating speed. When the latches are not required, leaving the Enable ($\overline{E}$) input LOW makes the latches transparent.

The circuit uses internal lookahead carry to minimize delay to the F_n outputs and to the ripple Carry output, $\overline{C}_{n+4}$. Group Carry Lookahead Propagate ($\overline{P}$) and Generate ($\overline{G}$) outputs are also provided, which are independent of the Carry input $\overline{C}_n$. The $\overline{P}$ output goes LOW when a plus operation produces fifteen (nine for BCD) or when a minus operation produces zero. Similarly, $\overline{G}$ goes LOW when the sum of A and B is greater than fifteen (nine for BCD) in a plus mode, or when their difference is greater than zero in a minus mode. All inputs have 50 kΩ pull-down resistors.

Ordering Code: See Section 8

Logic Symbol

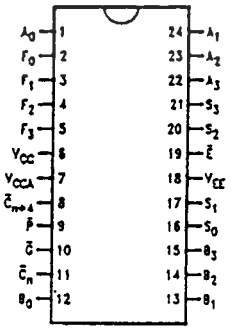


TL/F/9873–4

Pin Names	Description
A_0–A_3	Word A Operand Inputs
B_0–B_3	Word B Operand Inputs
$\overline{C}_n$	Carry Input (Active LOW)
S_0–S_3	Function Select Inputs
$\overline{E}$	Latch Enable Input (Active LOW)
$\overline{P}$	Carry Lookahead Propagate Output (Active LOW)
$\overline{G}$	Carry Lookahead Generate Output (Active LOW)
$\overline{C}_{n+4}$	Carry Output
F_0–F_3	Function Outputs

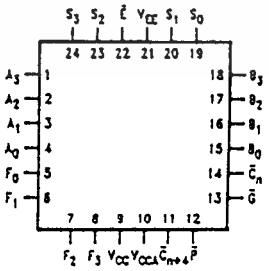
Connection Diagrams

24-Pin DIP



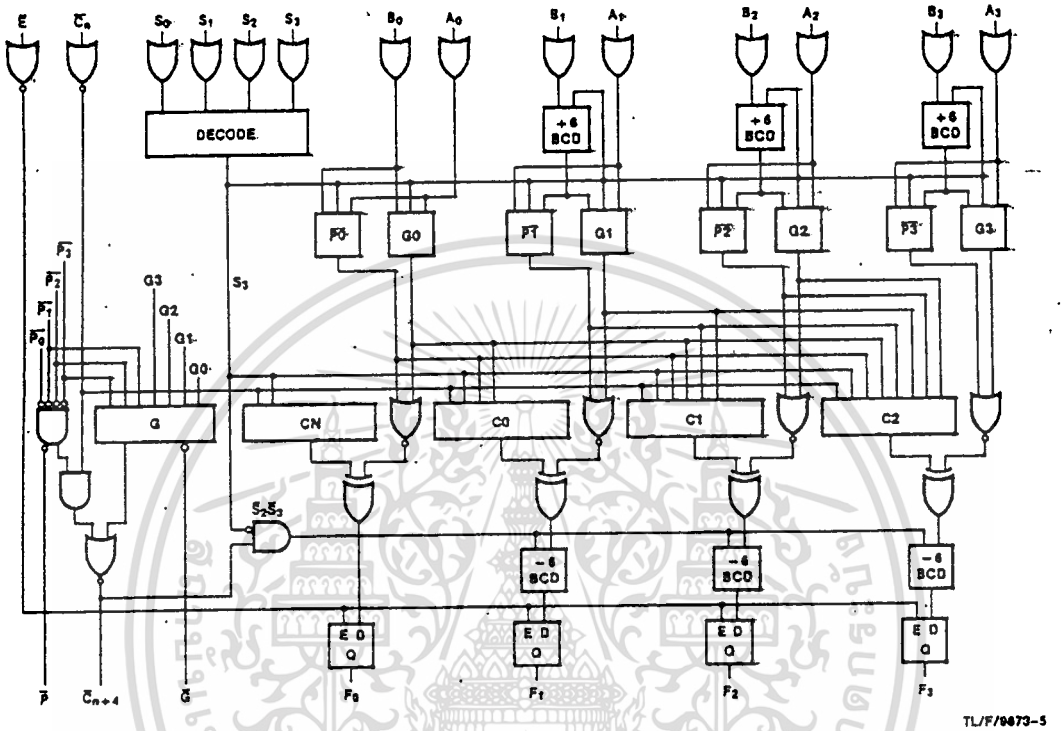
TL/F/9873–1

24-Pin Quad Cerpak



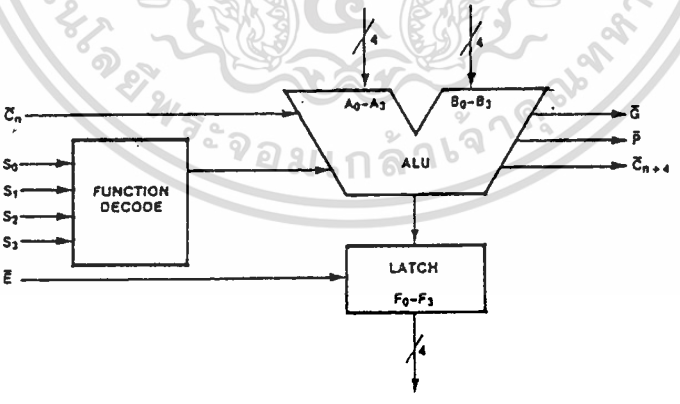
TL/F/9873–2

Logic Diagram



TL/F/9673–5

Block Diagram



TL/F/9673–6

Functional Description

There are two modes of operation: Arithmetic and Logic. The S_3 input controls these two modes:

S_3 = LOW for Arithmetic mode

S_3 = HIGH for Logic mode

The arithmetic mode includes decimal and binary arithmetic operations. S_2 is the control input with S_3 = LOW,

S_2 = LOW for Decimal Arithmetic (BCD)

S_2 = HIGH for Binary Arithmetic.

DECIMAL ARITHMETIC OPERATION

Addition

F = A plus B plus C_n. Arguments A and B are directly applied to the inputs. The circuit automatically performs the "+6" and "−6" logic correction internally.

Subtraction

F = A minus B plus C_n. Arguments A and B are directly applied to the inputs. The circuit automatically takes the nines complement of B and adds "+6". A "−6" adjustment is made if the subtraction algorithm calls for it. If there is a carry out, the result is a positive number. With no carry out, the result is a negative number expressed in its nines complement form. Therefore, to perform a subtraction with results in the tens complement form, an initial carry should be forced into the lowest order bit, i.e., set $\overline{C}_n$ = LOW.

(tens complement of B) = (nines complement of B) + 1

F = B minus A plus C_n. Operation is similar to and results are the same as F = A minus B plus C_n.

BINARY ARITHMETIC OPERATION

Addition

F = A minus B plus C_n. Arguments A and B are directly applied to the inputs.

Subtraction

F = A minus B plus C_n. Arguments A and B are directly applied to the inputs. The circuit automatically takes the ones complement of B (by inverting B internally). If there is a carry out the result is a positive number. With no carry out, the result is a negative number expressed in its ones complement form. Therefore, to perform a subtraction with results in the twos complement form, an initial carry should forced into the lowest order bit, i.e., set $\overline{C}_n$ = LOW.

(twos complement of B) = (ones complement of B) + 1

F = B minus A plus C_n. Operation is similar and results are the same as F = A minus B plus C_n.

Function Table

S_3	S_2	S_1	S_0	F_n Function	G_n (n = 0 to 3)	P_n (n = 0 to 3)	Outputs		
					Internal Signals		$\overline{C}_{n+4}$	$\overline{G}$	$\overline{P}$
L	L	L	L	F_n = A plus B plus C_n (BCD)	A_nD_n	$A_n + D_n$	$\overline{C}_{n+4}$	$\overline{G}$	$\overline{P}$
L	L	L	H	F_n = A minus B plus C_n (BCD)	$A_n\overline{B}_n$	$A_n + \overline{B}_n$	$\overline{C}_{n+4}$	$\overline{G}$	$\overline{P}$
L	L	H	L	F_n = B minus A plus C_n (BCD)	$\overline{A}_nB_n$	$\overline{A}_n + B_n$	$\overline{C}_{n+4}$	$\overline{G}$	$\overline{P}$
L	L	H	H	F_n = 0 minus B plus C_n (BCD)	L	$\overline{B}_n$	$\overline{C}_{n+4}$	$\overline{G}$	H
L	H	L	L	F_n = A plus B plus C_n (Binary)	A_nB_n	$A_n + B_n$	$\overline{C}_{n+4}$	$\overline{G}$	$\overline{P}$
L	H	L	H	F_n = A minus B plus C_n (Binary)	$A_n\overline{B}_n$	$A_n + \overline{B}_n$	$\overline{C}_{n+4}$	$\overline{G}$	$\overline{P}$
L	H	H	L	F_n = B minus A plus C_n (Binary)	$\overline{A}_nB_n$	$\overline{A}_n + B_n$	$\overline{C}_{n+4}$	$\overline{G}$	$\overline{P}$
L	H	H	H	F_n = 0 minus B plus C_n (Binary)	L	$\overline{B}_n$	$\overline{C}_{n+4}$	$\overline{G}$	H
H	L	L	L	$F_n = A_nB_n + \overline{A}_n\overline{B}_n$	A_nB_n	$A_n + B_n$	$\overline{C}_{n+4}$	$\overline{G}$	$\overline{P}$
H	L	L	H	$F_n = A_n\overline{B}_n + \overline{A}_nB_n$	$A_n\overline{B}_n$	$A_n + \overline{B}_n$	$\overline{C}_{n+4}$	$\overline{G}$	$\overline{P}$
H	L	H	L	$F_n = A_n + B_n$	A_n	$\overline{B}_n$	$\overline{C}_{n+4}$	$\overline{G}_x$	$\overline{P}$
H	L	H	H	$F_n = A_n$	A_n	H	$\overline{C}_{n+4}$	$\overline{G}$	L
H	H	L	L	$F_n = \overline{B}_n$	L	$\overline{B}_n$	$\overline{C}_{n+4}$	H	$\overline{P}$
H	H	L	H	$F_n = B_n$	L	$\overline{B}_n$	$\overline{C}_{n+4}$	H	$\overline{P}$
H	H	H	L	$F_n = A_nB_n$	L	$\overline{A}_n + \overline{B}_n$	$\overline{C}_{n+4}$	H	$\overline{P}$
H	H	H	H	F_n = LOW	L	H	$\overline{C}_n$	H	L

H = HIGH Voltage Level

L = LOW Voltage Level

$$\overline{P} = \overline{P}_0 + \overline{P}_1 + \overline{P}_2 + \overline{P}_3$$

$$\overline{G} = \overline{G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0}$$

$$\overline{C}_{n+4} = \overline{G} \bullet (\overline{P} + \overline{C}_n)$$

Arithmetic Operations

$$F_n = \overline{G_n + \overline{P}_n} \oplus C_i \quad I = 0 \text{ to } 3$$

Logic Operations

$$F_n = G_n + \overline{P}_n$$

Internal Equations for Carry Lookahead

(i = 0, 1, 2, 3)

$$C_0 = C_n + S_3$$

$$C_1 = G_0 + P_0C_n + S_3$$

$$C_2 = G_1 + P_1G_0 + P_1P_0C_n + S_3$$

$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_n + S_3$$

Internal Equations for +6 Logic

$$D_0 = B_0$$

$$D_1 = \overline{B}_1$$

$$D_2 = B_1B_2 + \overline{B}_1\overline{B}_2$$

$$D_3 = B_1 + B_2 + B_3$$

$$\overline{G}_x = \overline{G_3P_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0}$$

Absolute Maximum Ratings

Above which the useful life may be impaired. (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Storage Temperature	−65°C to +150°C
Maximum Junction Temperature (T_J)	+150°C
Case Temperature under Bias (T_C)	0°C to +85°C
V_{EE} Pin Potential to Ground Pin	−7.0V to +0.5V
Input Voltage (DC)	V_{EE} to +0.5V
Output Current (DC Output HIGH)	−50 mA
Operating Range (Note 2)	−5.7V to −4.2V

DC Electrical Characteristics

V_{EE} = −4.5V, V_{CC} = V_{CCA} = GND, T_C = 0°C to +85°C (Note 3)

Symbol	Parameter	Min.	Typ	Max.	Units	Conditions (Note 4)	
V_{OH}	Output HIGH Voltage	−1025	−955	−880	mV	V_{IN} = V_{IH} (Max) or V_{IL} (Min)	Loading with 50Ω to −2.0V
V_{OL}	Output LOW Voltage	−1810	−1705	−1620			
V_{OHC}	Output HIGH Voltage	−1035			mV	V_{IN} = V_{IH} (Min) or V_{IL} (Max)	Loading with 50Ω to −2.0V
V_{OLC}	Output LOW Voltage			−1610			
V_{IH}	Input HIGH Voltage	−1165		−880	mV	Guaranteed HIGH Signal for All Inputs	
V_{IL}	Input LOW Voltage	−1810		−1475	mV	Guaranteed LOW Signal for All Inputs	
I_{IL}	Input LOW Current	0.50			µA	V_{IN} = V_{IL} (Min)	

DC Electrical Characteristics

V_{EE} = −4.2V, V_{CC} = V_{CCA} = GND, T_C = 0°C to +85°C (Note 3)

Symbol	Parameter	Min	Typ	Max	Units	Conditions (Note 4)	
V_{OH}	Output HIGH Voltage	−1020		−870	mV	V_{IN} = V_{IH} (Max) or V_{IL} (Min)	Loading with 50Ω to −2.0V
V_{OL}	Output LOW Voltage	−1810		−1605			
V_{OHC}	Output HIGH Voltage	−1030			mV	V_{IN} = V_{IH} (Min) or V_{IL} (Max)	Loading with 50Ω to −2.0V
V_{OLC}	Output LOW Voltage			−1595			
V_{IH}	Input HIGH Voltage	−1150		−870	mV	Guaranteed HIGH Signal for All Inputs	
V_{IL}	Input LOW Voltage	−1810		−1475	mV	Guaranteed LOW Signal for All Inputs	
I_{IL}	Input LOW Current	0.50			µA	V_{IN} = V_{IL} (Min)	

DC Electrical Characteristics

V_{EE} = −4.8V, V_{CC} = V_{CCA} = GND, T_C = 0°C to +85°C (Note 3)

Symbol	Parameter	Min	Typ	Max	Units	Conditions (Note 4)	
V_{OH}	Output HIGH Voltage	−1035		−880	mV	V_{IN} = V_{IH} (Max) or V_{IL} (Min)	Loading with 50Ω to −2.0V
V_{OL}	Output LOW Voltage	−1830		−1620			
V_{OHC}	Output HIGH Voltage	−1045			mV	V_{IN} = V_{IH} (Min) or V_{IL} (Max)	Loading with 50Ω to −2.0V
V_{OLC}	Output LOW Voltage			−1610			
V_{IH}	Input HIGH Voltage	−1165		−880	mV	Guaranteed HIGH Signal for All Inputs	
V_{IL}	Input LOW Voltage	−1830		−1490	mV	Guaranteed LOW Signal for All Inputs	
I_{IL}	Input LOW Current	0.50			µA	V_{IN} = V_{IL} (Min)	

Note 1: Absolute maximum ratings are those values beyond which the device may be damaged or have its useful life impaired. Functional operation under these conditions is not implied.

Note 2: Parametric values specified at −4.2V to −4.8V.

Note 3: The specified limits represent the "worst case" value for the parameter. Since these "worst case" values normally occur at the temperature extremes, additional noise immunity and guard banding can be achieved by decreasing the allowable system operating ranges.

Note 4: Conditions for testing shown in the tables are chosen to guarantee operation under "worst case" conditions.

DC Electrical Characteristics

V_{EE} = −4.2V to −4.8V unless otherwise specified, V_{CC} = V_{CCA} = GND, T_C = 0°C to +85°C

Symbol	Parameter	Min	Typ	Max	Units	Conditions
I_{IH}	Input HIGH Current S_n, $\overline{E}$ All Others			350 250	µA	V_{IN} = $V_{IH\ (Max)}$
I_{EE}	Power Supply Current	−300	−210	−130	mA	Inputs Open

Ceramic Dual-In-Line Package AC Electrical Characteristics

V_{EE} = −4.2V to −4.8V, V_{CC} = V_{CCA} = GND

Symbol	Parameter	T_C = 0°C		T_C = +25°C		T_C = +85°C		Units	Conditions
		Min	Max	Min	Max	Min	Max		
t_{PLH} t_{PHL}	Propagation Delay A_n, B_n to F_n	2.00	6.90	2.10	6.80	2.30	7.40	ns	*Figures 1* and *2*
t_{PLH} t_{PHL}	Propagation Delay A_n, B_n to $\overline{P}$, $\overline{G}$	1.40	4.70	1.40	4.40	1.40	4.70	ns	
t_{PLH} t_{PHL}	Propagation Delay A_n, B_n to $\overline{C}_{n+4}$	2.00	6.50	2.00	6.50	2.10	6.80	ns	
t_{PLH} t_{PHL}	Propagation Delay $\overline{C}_n$ to F_n	1.60	5.10	1.60	5.20	1.60	5.50	ns	*Figures 1* and *2*
t_{PLH} t_{PHL}	Propagation Delay $\overline{C}_n$ to $\overline{C}_{n+4}$	1.30	3.00	1.40	3.00	1.40	3.10	ns	
t_{PLH} t_{PHL}	Propagation Delay S_n to F_n	1.40	8.80	1.50	8.60	1.50	9.00	ns	*Figures 1* and *2*
t_{PLH} t_{PHL}	Propagation Delay S_n to $\overline{P}$, $\overline{G}$	1.70	7.40	2.00	5.90	2.00	6.50	ns	
t_{PLH} t_{PHL}	Propagation Delay S_n to $\overline{C}_{n+4}$	2.70	10.10	2.80	8.50	2.90	8.70	ns	
t_{PLH} t_{PHL}	Propagation Delay $\overline{E}$ to F_n	1.00	3.40	0.90	3.60	1.10	3.80	ns	*Figures 1* and *2*
t_{TLH} t_{THL}	Transition Time 20% to 80%, 80% to 20%	0.45	2.70	0.45	2.60	0.45	2.70	ns	*Figures 1* and *2*
t_s	Setup Time A_n, B_n S_n $\overline{C}_n$	 7.60 8.70 4.80		 7.60 8.50 5.00		 8.10 9.60 5.30		ns	*Figure 3*
t_h	Hold Time A_n, B_n S_n $\overline{C}_n$	 0.10 0.60 0.60		 0.10 0.60 0.60		 0.10 0.60 0.60		ns	
$t_{pw(L)}$	Pulse Width LOW $\overline{E}$	2.00		2.00		2.00		ns	*Figure 2*

ประวัติผู้เขียน

ชื่อผู้เขียน	นายสมจินต์ ทองปลิว
วันเดือนปีเกิด	23 กันยายน พ.ศ. 2512
สถานที่เกิด	จังหวัดกรุงเทพฯ
วุฒิการศึกษาระดับปริญญาตรี	อุตสาหกรรมศาสตรบัณฑิต สาขาวัดและควบคุมทางอุตสาหกรรม
สถานที่สำเร็จการศึกษา	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2534
ผลงานทางวิชาการที่ได้รับการตีพิมพ์	เรื่องการสร้างวงจรกรองสัญญาณเชิงเลขสองมิติ โดย Distributed Arithmetics. วิศวสารลาดกระบัง
ประสบการณ์การทำงาน	วิศวกร บริษัท อิริคสัน ประเทศไทยจำกัด หัวหน้าแผนกวิศวกรรม บริษัท ซาซายี อินสทรูเมนท์ ประเทศไทยจำกัด
อาชีพปัจจุบัน	ผู้จัดการ บริษัท ที. เท็ค เอ็นจิเนียริ่ง จำกัด