รายงานวิจัยเรื่อง

ระบบการประมวลภาพดิจิตอลโดยใช้ทรานสพิวเตอร์

โดย ผศ. ดร. กิตติ ไพฑูรย์วัฒนกิจ
นาย ไพลิน บุญเดช
นาย สุวัฒน์ ศรีธนรัตน์
นส.จันทร์เพ็ญ เจริญยัง

โครงการนี้ได้รับการสนับสนุนจากสำนักงาน คณะกรรมการวิจัยแห่งชาติ
ประจำปี ๒๕๓๖

ระบบประมวลภาพดิจิตอลโดยใช้ทรานสพิวเตอร์
(Transputer based image analysis system)

ปัญหาที่สำคัญเกี่ยวกับการประมวลภาพดิจิตอล(Digital image processing) ก็คือเวลาที่ใช้ในการประมวลผล (Processing times) ค่อนข้างจะช้ามากถ้าหากขนาดของภาพมีขนาดใหญ่ ในการวิจัยครั้งนี้ได้ใช้วิธีเพิ่มความเร็ว (Speedup) โดยใช้การประมวลผลแบบขนาน (Parallel processing) โดยอาศัยทรานสพิวเตอร์เป็นตัวประมวลผล เขียนโปรแกรมด้วยภาษาออคคาม (OCCAM) ซึ่งเป็นภาษาที่ใช้ควบคุมและจัดการ เกี่ยวกับส่งผ่านข้อมูลระหว่างทรานสพิวเตอร์แต่ละตัวได้อย่างมีประสิทธิภาพ และเนื่องจากโปรแกรมย่อยเดิมที่เขียนด้วยภาษาซี (C) สามารถนำมาแก้ไขใหม่โดยเปลี่ยนเป็นภาษาออคคามได้ง่าย เพราะโครงสร้างของภาษาออคคามเป็นแบบภาษาระดับสูง เพื่อเป็นการสะดวกจึงทำการสร้างเป็นโปรแกรมย่อย เพื่อสามารถนำไปใช้กับการประมวลภาพดิจิตอลที่มีความเร็วสูงต่อไป

Transputer based image analysis system

The major problem of real time digital image processing is time consuming. The greater the image size, the slower the process. The solutionis to speed up the process by parallel processing using the array of transputers. The current transputer development system is equipped with high level language OCCAM which controls the communication between each transputer throughits high speed 4-serial channel. The research's aim is to study and create the digital image processing procedures, which is based on the transputers, to provide the user for analysis the required image within a constrained time.

1. บทนำ

คอมพิวเตอร์ที่ประมวลผลแบบลำดับ (Sequential computer) จะกระทำคำสั่ง (Run) ได้ทีละคำสั่งในเวลาขณะใดขณะหนึ่ง วิธีเพิ่มความเร็วในการประมวลผลของคอมพิวเตอร์แบบลำดับ โดยลดเวลาในการกระทำแต่ละคำสั่งลง โดยการเพิ่มขีดความสามารถของโปรเซสเซอร์ มีข้อจำกัดในด้านเทคโนโลยีการผลิตชิปและผลกระทบจากการใช้สัญญาณนาฬิกาความถี่สูง จึงมีวิธีการเพิ่มความเร็วอีกวิธีหนึ่ง คือลดจำนวนครั้งในการทำคำสั่งลง โดยเพิ่มจำนวนโปรเซสเซอร์เข้าไปหลายๆตัว (ระบบมัลติโปรเซสเซอร์: Multiprocessor) ช่วยกันประมวลผลหลายๆคำสั่งพร้อมๆกัน จึงทำให้ประมวลผลได้เร็วกว่าคอมพิวเตอร์แบบลำดับ เรียกวิธีประมวลผลวิธีนี้ว่าการ ประมวลผลแบบขนาน

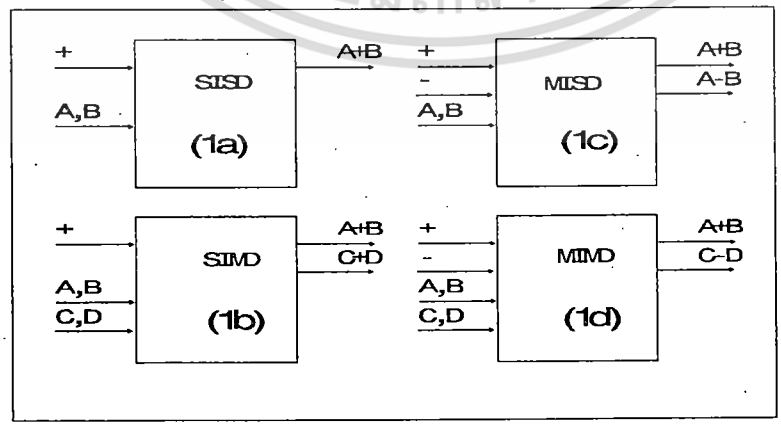
1.1 การจัดโครงสร้างสถาปัตยกรรมของคอมพิวเตอร์[14] สามารถแบ่งเป็นวิธีต่างๆได้ดังนี้

1.1.1 SISD (Single Instruction Single Data) เป็นคอมพิวเตอร์ที่กระทำคำสั่ง ได้ทีละหนึ่งคำสั่ง และกระทำบนข้อมูลชุดเดียวมีโครงสร้างแสดงดังรูป 1a คอมพิวเตอร์แบบลำดับจัดอยู่ในกลุ่มนี้

1.1.2 SIMD (Single Instruction Multiple Data) เป็นคอมพิวเตอร์ที่กระทำคำสั่งทีละหนึ่งคำสั่งบนข้อมูลหลายๆชุดพร้อมๆกัน คอมพิวเตอร์ขนานชนิดนี้ถูกเรียกว่า เวกเตอร์คอมพิวเตอร์ (Vecter Computer) หรือ อะเรย์ คอมพิวเตอร์ (Array Computer) มีโครงสร้างแสดงดังรูป 1b

1.1.3 MISD (Multiple Instruction Single Data) เป็นคอมพิวเตอร์ที่กระทำ คำสั่งหลายๆคำสั่งได้พร้อมกันบนข้อมูลชุดเดียวกันมีโครงสร้างแสดงดังรูป 1c มีชื่อเรียกอีกอย่างหนึ่งว่าไปป์ไลน์คอมพิวเตอร์ (Pipeline Computer)

1.1.4 MIMD (Multiple Instruction Multiple Data) เป็นคอมพิวเตอร์ที่กระทำคำสั่งหลายๆคำสั่งบนข้อมูลหลายๆชุดพร้อมๆกัน มีโครงสร้างแสดงดังรูป 1d ในแต่ละโปรเซสเซอร์อาจจะมีการติดต่อเชื่อมโยงกันในระหว่างกระทำคำสั่งหรือไม่ก็ได้



รูป 1: โครงสร้างคอมพิวเตอร์ชนิดต่างๆ

เวลาที่ใช้ในการประมวลผลแบบลำดับหาได้จากสมการที่ (1)

$$\text{เวลาในการประมวลผลแบบลำดับ} = (\sum x_i) * t_i \qquad (1)$$

$(\sum x_i)$ คือจำนวนครั้งของคำสั่งที่ต้องกระทำทั้งหมด

t_i คือเวลาในการกระทำแต่ละคำสั่ง

เวลาที่ใช้ในการประมวลผลแบบขนานหาได้จากสมการที่ (2)

$$\text{เวลาในการประมวลผลแบบขนาน} = \left(\frac{(\sum x_i)}{p_i}\right) * t_i \qquad (2)$$

$(\sum x_i)$ คือจำนวนครั้งของคำสั่งที่ต้องกระทำทั้งหมด

p_i คือจำนวนคำสั่งที่ทำขนานกันได้

t_i คือเวลาในการกระทำแต่ละคำสั่ง

1.2 การพิจารณาเลือกใช้วิธีการประมวลผลแบบลำดับหรือแบบขนานจะพิจารณาจากคุณสมบัติ 2 ประการคือ

1.2.1 ขีดความสามารถ (Speedup) โดยวัดจากความเร็วที่เพิ่มขึ้นเทียบอัตราส่วนกับความเร็วของการประมวลผลแบบลำดับ ดังแสดงในสมการที่ (3)

$$\text{ขีดความสามารถ} = \frac{\text{เวลาในการประมวลผลแบบลำดับ}}{\text{เวลาในการประมวลผลแบบขนาน}} \qquad (3)$$

1.2.2 ประสิทธิภาพการทำงาน (Performance) โดยวัดว่ามีการใช้โปรเซสเซอร์ให้เกิดประโยชน์สูงสุดเพียงไร หาได้จากสมการที่ (4)

$$\text{ประสิทธิภาพ} = \frac{\text{เวลาในการประมวลผลแบบลำดับ}}{(\text{เวลาในการประมวลผลแบบขนาน} * \text{จำนวนโปรเซสเซอร์ในระบบ})} \qquad (4)$$

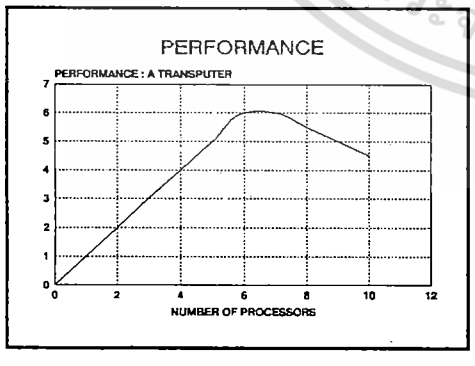
ความพยายามในการนำไมโครโปรเซสเซอร์แบบทั่ว ๆ ไปมาต่อเชื่อมกันเพื่อร่วมกันประมวลผลแบบขนานโดยเชื่อมกันทางระบบบัส (BUS LINK) ประสบปัญหา หลายประการเช่น

- กลไกจัดการเชื่อมโยงยังไม่ดีพอ
- ความสามารถในการสื่อสารระหว่างโปรเซสเซอร์ยังไม่มีประสิทธิภาพ
- ความเร็วในการติดต่อมีข้อจำกัดที่บัส ทำให้เกิดปัญหาการแย่งใช้บัส (เกิด Bottle neck บนบัส)

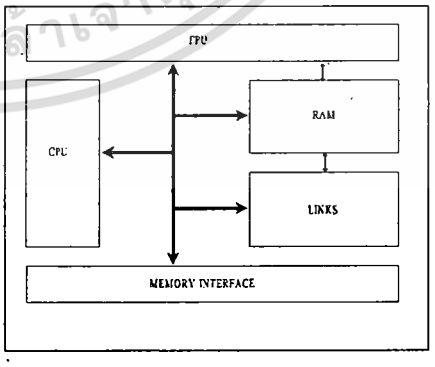
เมื่อเพิ่มจำนวนโปรเซสเซอร์ไปจนถึงขีดจำกัดของบัส ทำให้ขีดความสามารถและประสิทธิภาพรวมของระบบตกลง ดังแสดงในกราฟรูปที่ 2. จึงได้มีการคิดค้นโปรเซสเซอร์ชนิดพิเศษ สำหรับต่อเป็นระบบมัลติโปรเซสเซอร์ เพื่อประมวลผลแบบขนานโดยเฉพาะ โดยบริษัทอินมอส (INMOS Ltd.) และให้ชื่อว่า "ทรานสพิวเตอร์" พร้อมกันนี้ได้พัฒนาภาษาโปรแกรมสำหรับการประมวลผลแบบขนานใช้ควบคู่กัน และให้ชื่อว่า "ภาษาออคคาม" ซึ่งในปัจจุบันภาษาออคคามได้พัฒนามาเป็นภาษาออคคาม2 (OCCAM2)

2 ทรานสพิวเตอร์ [1,2,3,4,11]

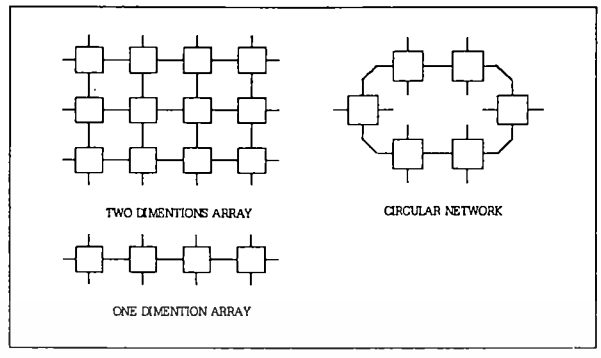
ทรานสพิวเตอร์ถูกคิดค้นและออกแบบให้เป็นโปรเซสเซอร์ที่สมบูรณ์แบบในชิปเดียว ภายในประกอบด้วยหน่วยประมวลผลกลาง (CPU), หน่วยความจำ (Memory), หน่วยคำนวนทางคณิตศาสตร์ (FP:Floating Point Unit) และหน่วยควบคุมการติดต่อสื่อสารหรือหน่วยเชื่อมโยง (Link) มีโครงสร้างแสดงดังรูปที่ 3 สถาปัตยกรรมของทรานสพิวเตอร์ที่ผลิตออกจำหน่ายมีทั้งสถาปัตยกรรมขนาด 16 บิท เช่นเบอร์ T212, และ 32 บิทเช่นเบอร์ T800, T805, และ T9000 เป็นต้น ทรานสพิวเตอร์หลาย ๆ ตัวสามารถนำมาต่อเชื่อมโยงกันเป็นเน็ทเวิร์ค (Network) ในรูปแบบต่างๆเช่น อะเรย์ 1 มิติ, อะเรย์ 2 มิติ, วงกลม ฯลฯ เพื่อให้ประมวลผลแบบขนานดังตัวอย่างแสดงในรูปที่ 4 ขีดความสามารถของระบบทรานสพิวเตอร์เน็ทเวิร์คจะยังคงเป็นเชิงเส้นและประสิทธิภาพรวมของระบบจะไม่ตกลงมาก แม้จะเพิ่มจำนวนทรานสพิวเตอร์เข้าไปในเน็ทเวิร์คมาก ๆ ก็ตาม ดังแสดงในกราฟรูปที่ 5a และ 5.b ตามลำดับ รายละเอียดโครงสร้างสถาปัตยกรรมและกลไก (Mechanism) ทั้งหมด ที่ทำให้ทรานสพิวเตอร์ต่อเชื่อมโยงเป็นเน็ทเวิร์คประมวลผลแบบขนานได้มีรายละเอียดทั้งหมดในเอกสารอ้างอิง. [12,13] เฉพาะรายละเอียดที่สำคัญของทรานสพิวเตอร์มีดังนี้



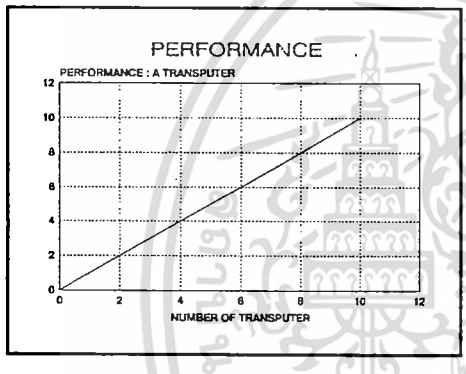
รูปที่ 2 ขีดความสามารถคอมพิวเตอร์ระบบบัส



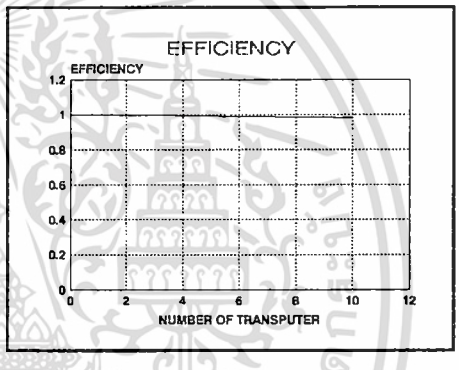
รูปที่ 3 โครงสร้างพื้นฐานของทรานสพิวเตอร์



รูปที่ 4 ตัวอย่างรูปแบบการจัดทรานสพิวเตอร์เน็ทเวิร์ค



รูปที่ 5a กราฟแสดงขีดความสามารถ-ของทรานสพิวเตอร์เน็ทเวิร์ค



รูปที่ 5b กราฟแสดงประสิทธิภาพของ-ทรานสพิวเตอร์เน็ทเวิร์ค

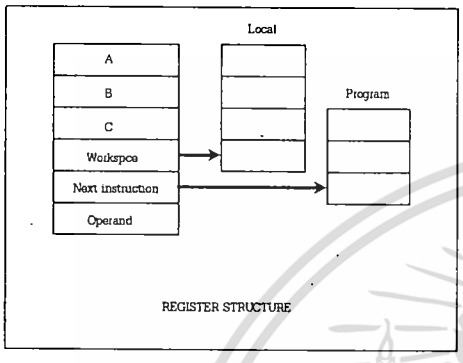
2.1 หน่วยประมวลผล

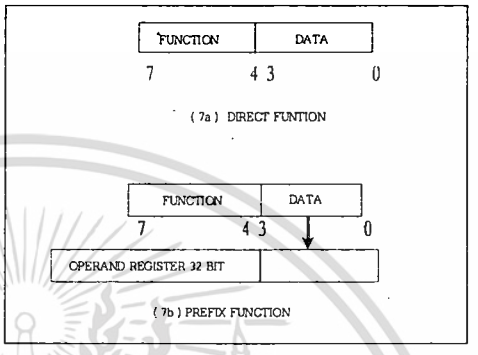
หน่วยประมวลผลของทรานสพิวเตอร์เป็นโครงสร้างสถาปัตยกรรมแบบ RISC (Reduce Instruction Set Computer) ทำงานแบบสแตกแมชชีน (Stack machine) ทั้งหน่วยประมวลผลกลางและหน่วยคำนวณทางคณิตศาสตร์มีโครงสร้างเหมือนกัน แต่ชุดคำสั่งต่างกัน คือหน่วยประมวลผลกลางมีชุดคำสั่งเหมือนซีพียูแบบ RISC ทั่วไป แต่หน่วยคำนวณทางคณิตศาสตร์มีชุดคำสั่งในการคำนวณฟังก์ชันทางคณิตศาสตร์ที่สลับซับซ้อน หน่วยประมวลผลกลางของทรานสพิวเตอร์มีรายละเอียดต่างๆดังนี้

2.1.1 รีจีสเตอร์

รีจีสเตอร์ของหน่วยประมวลผลประกอบด้วยรีจีสเตอร์ขนาด 32 บิทจำนวน 6 ตัว รีจีสเตอร์ 3 ตัวแรกได้แก่ A,B,C วางทับซ้อนกันเป็นแสตกคำนวณและประเมินค่า (Evaluation Stack) ทำหน้าที่โหลด(Load), เก็บคืน (Store) ตัวถูกดำเนินการ (Operand), และกระทำตามคำสั่งผลลัพธ์ของการกระทำคำสั่งจะเก็บไว้ชั้นบนสุดของสแตกส่วนรีจีสเตอร์ตัวชี้ (Pointer Register) อีกสองตัวคือ รีจิสเตอร์ชี้ข้อมูล (WorkSpace Pointer) ทำหน้าที่ชี้ตำแหน่งตัวแปรแบบท้องถิ่น (Local Variable) และรีจิสเตอร์ชี้คำสั่งถัดไป

(Next Instruction Pointer) ทำหน้าที่ชี้คำสั่งที่จะถูกกระทำของโปรแกรม คล้ายกับตัวนับคำสั่งโปรแกรม (Program Counter) ของซีพียูทั่วไป รีจีสเตอร์ตัวสุดท้ายคือรีจิสเตอร์ตัวถูกกระทำ (Operand Register) ทำหน้าที่เก็บค่าตัวถูกกระทำ โครงสร้างรีจิสเตอร์แสดงในรูปที่ 6





รูปที่ 6 โครงสร้างรีจิสเตอร์ของทรานสพิวเตอร์ รูปที่ 7 รูปแบบชุดคำสั่งของทรานสพิวเตอร์

2.1.2 ชุดคำสั่ง

ทรานสพิวเตอร์ถูกออกแบบมาให้โปรแกรมโดยภาษาระดับสูง ชุดคำสั่งจึงออกแบบให้ง่ายต่อการแปลภาษา (Compile) และมีประสิทธิภาพเพียงพอต่อการเขียนโปรแกรม รูปแบบของคำสั่ง (Format) จึงสั้นง่ายเหมือนกันทุกคำสั่ง ทำให้การถอดรหัส (Decode) คำสั่งด้วยไมโครโค้ด (Microcode) ง่าย ใช้เวลาในการกระทำคำสั่งน้อย ชุดคำสั่งทั้งหมดของทรานสพิวเตอร์[10] มีลักษณะที่สำคัญคือ คำสั่งทุกคำสั่งประกอบด้วย หน้าที่ (Function) และข้อมูล (Data) ไบท์คำสั่งมีขนาด 8 บิท แบ่งออกเป็น 2 ส่วนคือ สี่บิทแรกเป็นฟังก์ชัน และ สี่บิทหลังเป็นตัวถูกดำเนินการหรือข้อมูลดังแสดงในรูปที่ 7a ชุดคำสั่งแบ่งเป็นกลุ่มดังนี้

2.1.2.1 กลุ่มคำสั่งตรง (Direct Function)

เป็นคำสั่งไบท์เดียวมี 16 ฟังก์ชัน กระทำกับข้อมูลที่มีค่าอยู่ระหว่าง 0-15 กลุ่มคำสั่งในกลุ่มนี้เป็นกลุ่มคำสั่งพื้นฐานที่ใช้งานบ่อยที่สุดกลุ่มคำสั่งเหล่านี้ได้แก่ LOAD, STORE, JUMP, ADD และCALL

2.1.2.2 ปรีฟิกซ์ ฟังก์ชัน (Prefix Function)

ทำหน้าที่ขยายตัวถูกดำเนินการของคำสั่งตรง ซึ่งเดิมกระทำได้กับตัวถูกดำเนินการเพียงขนาด 4 บิท ให้สามารถกระทำคำสั่งกับตัวถูกดำเนินการเป็นขนาด 32 บิท ทั้งค่าบวกและค่าลบ วิธีการขยายทำโดยย้ายส่วนข้อมูลจากไบท์คำสั่งลงมาเก็บที่รีจิสเตอร์ตัวถูกดำเนินการ และเลื่อนบิทไปทางซ้าย ทีละ 4 บิท ค่าตัวถูกดำเนินการจริง คือค่ารีจิสเตอร์ตัวถูกดำเนินการ 32 บิทกระทำลอจิก OR กับตัวถูกดำเนินการของไบท์คำสั่งอีก 4 บิทดังแสดงในรูป 7b ส่วนคำสั่ง Negative Prefix จะทำค่าให้เป็นลบโดยการทำ 2'Complement ก่อนการเลื่อนเสมอ

2.1.2.3 กลุ่มคำสั่งทางอ้อม (Indirect Function)

นอกจากจะขยายตัวถูกดำเนินการได้แล้ว ฟังก์ชันคำสั่งก็ขยายได้เช่นกัน โดยวิธีเดียวกันกับการขยายตัวถูกดำเนินการ จากเดิมที่มีคำสั่งตรงเพียง 16 ฟังก์ชันซึ่งไม่เพียงพอต่อการใช้งาน จึงมีวิธีการขยายฟังก์ชันเพิ่มขึ้นอีก เนื่องจากการเลื่อนบิท 1 ครั้ง ครั้งละ 4 บิท ขยายคำสั่งได้ถึง 512 ฟังก์ชัน (ขยายทางบวกได้ 255, และขยายทางลบได้ -256 รวมได้ 512 คำสั่ง) ซึ่งเกินพอสำหรับทุกคำสั่งในทรานสพิวเตอร์ กลุ่มคำสั่งที่เป็นกลุ่มขยายฟังก์ชันได้แก่กลุ่มคำสั่งที่ถูกเรียกใช้งานน้อยลงมาเป็น ลำดับรองจากกลุ่มคำสั่งตรง

2.2 การกระทำคำสั่ง (Execute)

2.2.1 การกระทำคำสั่งแบบลำดับ (Sequential Execute)

กระทำคำสั่งทีละหนึ่งคำสั่งเรียงตามลำดับดังตัวอย่างที่ 1 เป็นการแสดงการทำคำสั่งโปรแกรมบวกค่าตัวแปร เขียนโดยภาษาออคคาม และแปลเป็นภาษาแอสเซมบลีของทรานสพิวเตอร์

ตัวอย่างที่ (1) การทำคำสั่งแบบลำดับของทรานสพิวเตอร์

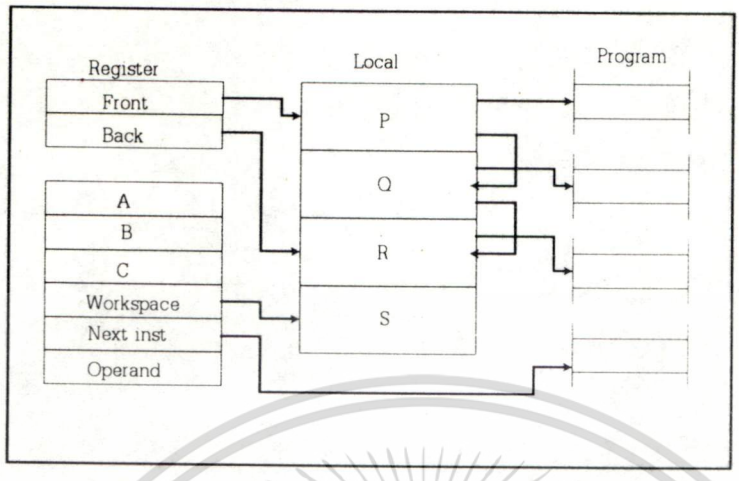
ภาษาออคคาม	---แปล-->	ภาษาแอสเซมบลี	
INT x,y,z:		ldl x	;ขั้นที่ 1
SEQ -- ให้ทำคำสั่งแบบลำดับ		ldl y	;ขั้นที่ 2
z := x + y		add	;ขั้นที่ 3
		st z	;ขั้นที่ 4

ไดอะแกรมการปฏิบัติคำสั่ง

ขั้นที่ 1 ---------->	ขั้นที่ 2 ---------->	ขั้นที่ 3 ---------->	ขั้นที่ 4
load x into A	load y into A	add A,B	store A to Z
push old-A into B	push old-A into B		pop A from B
push old-B into C	push old-B into C		pop B from C
old-C lose	old-C lose		C undefine

2.2.2 การกระทำคำสั่งแบบขนาน (Parallel Execute)

จากการกระทำคำสั่งแบบลำดับตามตัวอย่างที่ 1 เมื่อให้ขั้นตอนหรือโปรเซส (Process) เข้ามาปฏิบัติทีละคำสั่งแล้วสลับเอาโปรเซสใหม่เข้ามาปฏิบัติสลับไปมา โปรเซสหลายๆโปรเซสจะถูกปฏิบัติไปพร้อมๆกันจึงเรียกวิธีปฏิบัติโปรเซสแบบนี้ว่าการปฏิบัติโปรเซสแบบขนาน (Concurrency Process) โปรเซสหลายโปรเซสปฏิบัติแบบขนานบนทรานสพิวเตอร์ตัวเดียวได้ โดยมีกลไกสนับสนุนการทำงานคือ ลิงค์ลิส(Link list) และคิว(Queue) โดยใช้ หน่วยความจำแบบท้องถิ่น (Local Memory) เก็บข้อมูลลิสร่วมกับรีจิสเตอร์ชี้หัวคิว (Front Register) และรีจิสเตอร์ชี้ท้ายคิว (Back Register) ดังแสดงในรูปที่ 8



รูปที่ 8 กลไกในการกระทำคำสั่งแบบขนาน

ซีพียูมีคำสั่งพิเศษในการควบคุมโปรเซสคือ

Start Process , End Process

โปรเซสมีสถานะอยู่ 2 สถานะคือ

สถานะทำงาน (Active) คือขณะ

-กำลังปฏิบัติอยู่

-รอปฏิบัติอยู่ในคิวรอ

สถานะไม่ทำงาน (Inactive) คือขณะ

-รอเงื่อนไขอินพุท

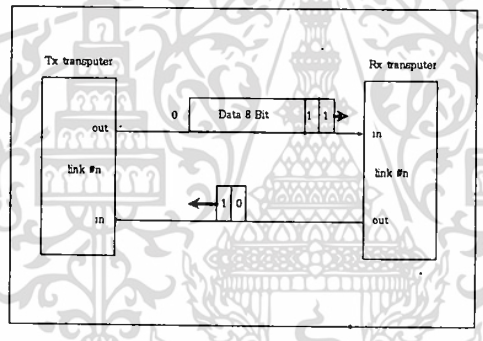
-รอเงื่อนไขเอาท์พุท

-รอเงื่อนไขไทม์เมอร์

เมื่อซีพียูปฏิบัติโปรเซสแบบขนาน คำสั่ง Start process จะสร้างโปรเซสใหม่ต่อ ท้ายลิงค์ลิสในคิวรอ (Waiting Queue) จากนั้นเริ่มเรียกโปรเซสเข้ามาปฏิบัติทีละหนึ่งคำสั่ง จบแล้วสลับเอาโปรเซสต่อไปเข้ามาปฏิบัติใหม่อีกหนึ่งคำสั่งสลับกันไปโปรเซสละคำสั่งจนจบท้ายคิว แล้วกลับไปปฏิบัติโปรเซสที่ต้นคิวใหม่ทีละหนึ่งคำสั่งจนจบคิว จึงดูเหมือนโปรเซสปฏิบัติขนานกันไป กรณีที่โปรเซสใดมีการรอเงื่อนไข อินพุท, เอาท์พุท หรือไทม์เมอร์ โปรเซสนั้นจะถูกผลักออกจากคิวรอ (Deschedule) ให้อยู่ในสภาวะไม่ทำงานจนกว่าจะได้เงื่อนไขที่ถูกต้องจึงจะถูกเรียกเข้ามารอในคิวรอ (Reschedule) เพื่อรอปฏิบัติต่อไป อนึ่งโปรเซสในภาษาออคคามมีการจัดลำดับความสำคัญ 2 ระดับคือลำดับความสำคัญสูง (Hi-Priority Process) และลำดับความสำคัญต่ำ (Low-Priority Process) โดยโปรเซสที่มีความสำคัญต่ำ จะทำงานได้เมื่อไม่มีโปรเซสที่มีลำดับความสำคัญสูงทำงานอยู่ก่อน และจะเข้าไปปฏิบัติได้เป็นช่วงๆในคาบเวลาแคบๆเท่านั้น (Period Timeslice มีค่า 1 ms.)

2.3 ลิงค์เชื่อมโยง

เป็นกลไกทางฮาร์ดแวร์เพื่อเชื่อมต่อทรานสพิวเตอร์หลายๆตัวเข้าด้วยกัน ลิงค์เชื่อมโยงเป็นสายสัญญาณชนิดทิศทางเดียว 2 เส้น เส้นหนึ่งทำหน้าที่ส่ง (Output) อีกเส้นหนึ่งทำหน้าที่รับ (Input) แต่ละเส้นภาษาออคคามจะกำหนดเป็นช่องสัญญาณ (Channel) ข้อมูลที่รับส่งจะส่งทีละไบท์มีโปรโตคอล (Protocol) พิเศษทำการตรวจสอบผลการรับส่ง (Handshaking) รูปแบบของชุดข้อมูล (Package) ใน 1 ไบท์ ประกอบด้วยบิทเริ่มต้นค่า '1' (Start bit) ตามด้วยบิทตามค่า '1' (Follow bit) และต่อด้วยข้อมูลขนาด 8 บิท จบด้วยบิทจบค่า '0' (Stop bit) จะส่งออกทางช่องเอาท์พุท สัญญาณตอบรับ (Acknowledge) ผู้รับจะส่งให้ผู้ส่งเพื่อบอกว่าได้รับข้อมูลแล้วมีรูปแบบเป็นขนาด 2 บิทบิทแรกค่า '1' บิทที่สองค่า'0' สัญญาณตอบรับจะถูกสร้างทันทีที่ได้รับสัญญาณที่ส่งมา เมื่อผู้ส่งได้รับสัญญาณตอบรับก็จะส่งข้อมูลไบท์ต่อไปทันทีโดยไม่ต้องเสียเวลาหยุดรอ ดังแสดงในไดอะแกรมรูปที่ 9



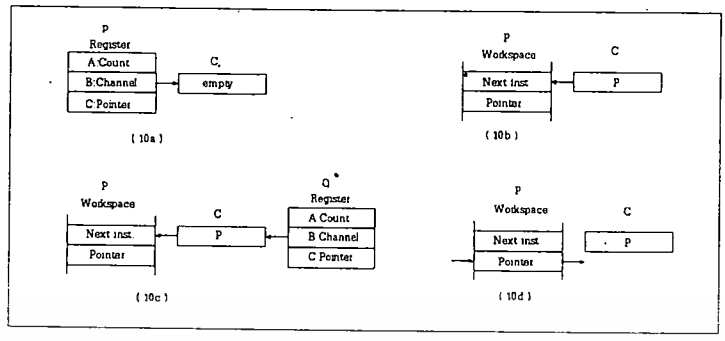
รูปที่ 9 ไดอะแกรมการรับส่งข้อมูลระหว่างทรานสพิวเตอร์

2.4 การติดต่อระหว่างโปรเซส

การติดต่อระหว่างโปรเซสแต่ละโปรเซสเป็นการส่งผ่านข้อมูลให้กัน การติดต่อทำได้ 2 ลักษณะคือ

2.4.1 ติดต่อระหว่างโปรเซสบนทรานสพิวเตอร์ตัวเดียวกัน

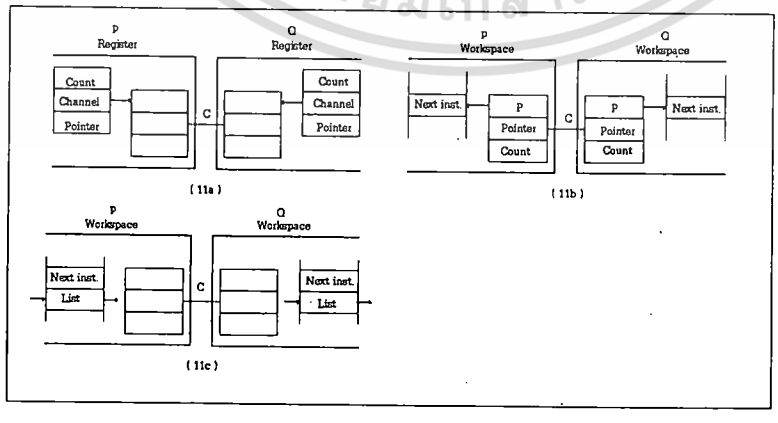
การติดต่อระหว่างทรานสพิวเตอร์ตัวเดียว มีวิธีการกระทำโดยใช้หน่วยความจำขนาด 1 ไบท์ถูกสมมุติให้เป็นช่องสัญญาณ สมมุติว่าโปรเซส P จะส่งข้อความไปให้โปรเซส Q โดยผ่านช่องสัญญาณ C เมื่อทำคำสั่งเอาท์พุท ซีพียูจะสั่งให้สแตกเก็บค่าต่างๆคือ A เก็บค่าจำนวนข้อมูล, B เก็บตำแหน่งของช่องสัญญาณ, C เก็บตำแหน่งคำสั่งถัดไปของโปรเซส P ไว้แล้ว โปรเซส P จะถูกทำให้เป็นสภาวะไม่ทำงาน เมื่อโปรเซส Q ต้องการรับข้อมูลจากโปรเซส P โดยทำคำสั่งอินพุทจะจัดสแตกให้เป็นเช่นเดียวกับโปรเซส P รีจิสเตอร์ B จะชี้ไปที่ C จากนั้นขั้นตอนการคัดลอก (Copy) จะเริ่มขึ้นจนจบโปรเซส P และ Q จะถูกเรียกกลับมากระทำคำสั่งต่อไป (สภาวะ Active) พร้อมกับรีเซท (Reset) ช่องสัญญาณ ให้เป็นว่างใหม่ (Empty) ดังแสดงในไดอะแกรมรูป 10



รูปที่ 10 ไดอะแกรมการรับส่งข้อมูลบนทรานสพิวเตอร์ตัวเดียวกัน

2.4.2 การติดต่อระหว่างโปรเซสบนทรานสพิวเตอร์ 2 ตัว

ทรานสพิวเตอร์จะมีช่องสัญญาณ 4 ชุดมีชื่อเรียกว่า ลิงค์ (Link) 0,1,2,3 ตามลำดับ ลิงค์แต่ละลิงค์ของทรานสพิวเตอร์จะมีรีจิสเตอร์ 3 ตัวคอยควบคุมการรับ/ส่งข้อมูลโดยรีจิสเตอร์ 0 เก็บตำแหน่งเริ่มต้นของข้อมูล, รีจิสเตอร์ 1 เก็บตำแหน่งคำสั่งถัดไปของโปรเซส, รีจิสเตอร์ 2 เก็บจำนวนข้อมูล สมมุติว่าโปรเซส P ทำคำสั่งเอาท์พุท ค่าจำนวนข้อมูล, ตัวชี้ตำแหน่งคำสั่ง, ตัวชี้ตำแหน่งข้อมูลบนสแตก จะโอนถ่ายการควบคุมไปให้รีจิสเตอร์ควบคุมการติดต่อ และโปรเซสจะถูกทำให้เป็นสภาวะไม่ทำงาน เมื่อมีอีกโปรเซสหนึ่งซึ่งปฏิบัติอยู่บนทรานสพิวเตอร์อีกตัวหนึ่งแต่ต่อเชื่อมโยงลิงค์เข้าด้วยกันชื่อโปรเซส Q ทำคำสั่งอินพุท รับข้อมูลจากโปรเซส Q โปรเซส Q จะโอนถ่ายการควบคุมไปยังรีจิสเตอร์ควบคุมเช่นเดียวกับโปรเซส P ทรานสพิวเตอร์ทั้ง 2 จะติดต่อรับส่งข้อความกันจนเสร็จ จึงโอนถ่ายการควบคุมกลับไปยังแสตกเพื่อให้โปรเซส P และ Q ถูกเรียกขึ้นมาทำงานต่อไปได้ จะเห็นว่าการที่โปรเซสจะปฏิบัติคำสั่งต่อไปได้ต้องให้การติดต่อจบสิ้นสมบูรณ์ก่อนเสมอดังแสดงในไดอะแกรมรูป 11



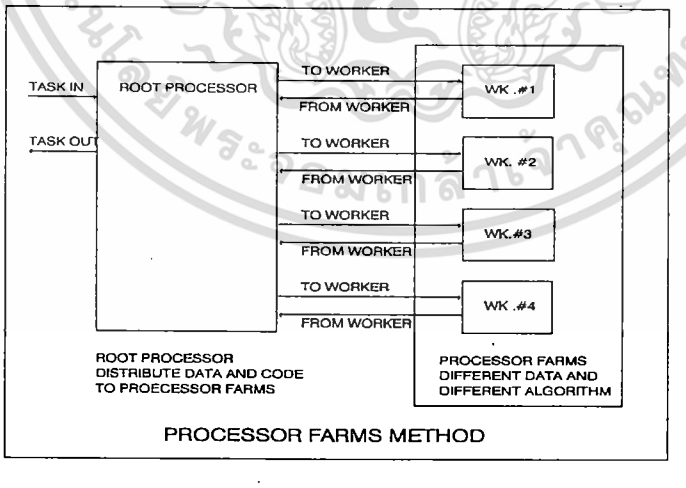
รูปที่ 11 ไดอะแกรมการรับส่งข้อมูลระหว่างทรานสพิวเตอร์ 2 ตัว

3 การเขียนโปรแกรมแบบขนาน (Parallel Programming)

หลักสำคัญของการเขียนโปรแกรมแบบขนาน[14] คือการแบ่งกระจายงาน (Distribution) ให้ สัมพันธ์กับโปรเซสเซอร์เน็ทเวิร์ค โดยให้งาน (Task) แก่โปรเซสเซอร์ทุกตัวในอัตราที่เท่ากัน ต้องไม่มีโปรเซสเซอร์ใดๆ ว่างงาน (Idle) ในขณะที่ตัวอื่นๆทำงานการจบงานต้องจบพร้อมกันทุกโปรเซสเซอร์ ในส่วนของอัลกอริทึมจะต้องพิจารณาว่ามีส่วนใดบ้างของอัลกอริทึมที่สามารถแยกกันกระทำคำสั่งได้อย่างอิสระไม่มีผลกระทบต่อการทำงานแบบขนานกัน งานที่เหมาะสมกับการประมวลผลแบบขนานคืองานที่มีข้อมูลขนาดใหญ่แบ่งส่วนข้อมูลเป็นส่วนย่อยๆได้เช่นงานประมวลผลภาพ (Image Processing), ระบบปัญญาประดิษฐ์ (AI Machine) เป็นต้น วิธีการแบ่งกระจายงานทำได้หลายวิธีขึ้นอยู่กับลักษณะปัญหาที่จะนำมาประมวลผล แบ่งเป็นวิธีต่าง ๆ ได้ดังนี้

3.1 โปรเซสเซอร์ฟาร์ม (Processor Farms)

เป็นวิธีการแบ่งกระจายงานทั้งหมดออกเป็นงานย่อย ๆ ภายในเน็ทเวิร์คโปรเซสเซอร์ประกอบด้วยโปรเซสเซอร์หลัก (Supervisory Processor หรือ Root Processor) ทำหน้าที่แบ่งกระจายงานให้กับโปรเซสเซอร์ทำงาน (Worker Processor) เมื่อโปรเซสเซอร์ทำงานประมวลผลเสร็จ จะส่งผลงานคืนให้โปรเซสเซอร์หลักดังแสดงในรูปที่ 12 ข้อด้อยของการแบ่งกระจายงานวิธีนี้คือโปรเซสเซอร์หลักหนึ่งตัวควบคุมโปรเซสเซอร์ย่อยได้เพียง 4 ตัว (จากคุณสมบัติของทรานสพิวเตอร์ที่มีลิงค์ 4 ลิงค์) เมื่อนำทรานสพิวเตอร์ตัวที่ 5,6,7,... มาต่อเพิ่มต้องจัดแบ่งระบบการควบคุมใหม่ให้เป็นลำดับชั้นแบบต้นไม้ (Tree) ทำให้ไม่สะดวกเมื่อมีการเปลี่ยนแปลงขนาดของโปรเซสเซอร์ในเน็ทเวิร์ค ในการวิจัยครั้งนี้ใช้ทรานสพิวเตอร์เพียง 3 ตัว จึงใช้วิธีการแบ่งกระจายงานแบบโปรเซสเซอร์ฟาร์ม



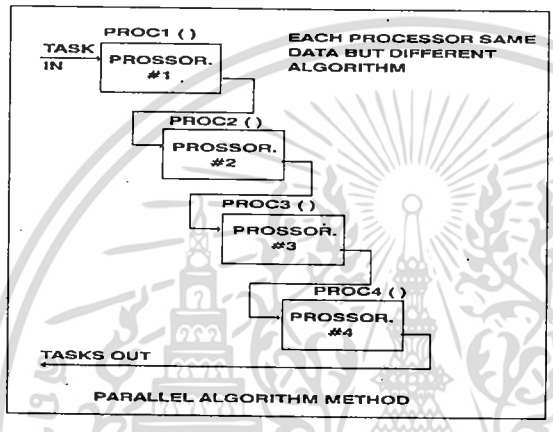
รูปที่ 12 วิธีแบ่งกระจายงานแบบโปรเซสเซอร์ฟาร์ม

3.2 แบบขนานอัลกอริทึม (Algorithm - Parallel)

เป็นวิธีการแบ่งกระจายงานโดยการแบ่งที่รหัสโปรแกรม(Code) คือโปรแกรมหลักจะถูกแบ่งออกเป็นโปรแกรมย่อยๆ(Procedure) แล้วส่งโปรแกรมย่อยไปปฏิบัติงานบนโปรเซสเซอร์แต่ละตัว ข้อมูลที่จะ

ประมวลผลจะถูกส่งไปประมวลผล ที่โปรเซสเซอร์ตัวแรก เสร็จแล้วไปประมวลผลที่โปรเซสเซอร์ตัวที่ 2,3,... จนจบ เรียกอีกอย่างหนึ่งว่าการประมวลผลแบบไปป์ไลน์ (Pipe Line Processing) ดังแสดงในรูปที่ 13

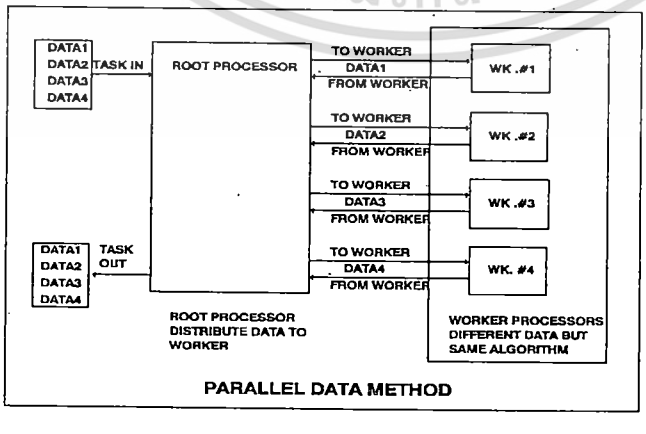
การแบ่งกระจายงานวิธีนี้เห็นได้ชัดว่า รหัสโปรแกรมของโปรเซสเซอร์แต่ละตัวไม่เหมือนกัน แต่ข้อมูลที่ประมวลผลเป็นข้อมูลชุดเดียวกัน จึงเหมาะกับงานที่มีส่วนย่อย (Routine) การประมวลผลหลายๆส่วนย่อย ข้อมูลทุกส่วนจะถูกส่งไปประมวลผลผ่านทุกส่วนย่อย ตัวอย่างเช่นการทำคำสั่งภายในของซีพียูแบบไปป์ไลน์ เป็นต้น



รูปที่ 13 วิธีแบ่งกระจายงานแบบขนานอัลกอริทึม

3.3 การขนานแบบเรขาคณิต หรือขนานข้อมูล (Data Parallel:Geometric Parallelism)

เป็นวิธีการแบ่งกระจายงาน แบบตรงข้ามกับแบบขนานอัลกอริทึม คือข้อมูลที่จะประมวลผลถูกแบ่งออกเป็นข้อมูลย่อยๆ แล้วแยกไปประมวลผลที่โปรเซสเซอร์แต่ละตัวรหัสโปรแกรมที่ ปฏิบัติงานบนโปรเซสเซอร์แต่ละตัวเป็นรหัสเดียวกัน ดังแสดงในรูปที่ 14 วิธีการแบ่งกระจายงานวิธีนี้เหมาะกับงานที่มีโครงสร้างข้อมูลขนาดใหญ่แต่ฟังก์ชันที่ประมวลผลกับข้อมูลทั้งหมด มีฟังก์ชันเดียว เช่นงานประมวลผลภาพ



รูปที่ 14 วิธีแบ่งกระจายงานแบบขนานเรขาคณิต หรือแบบขนานข้อมูล

ส่วนภาษาที่ใช้เขียนโปรแกรมแบบขนานมีหลายภาษาเช่น ภาษาเอดา(ADA), ภาษาออคคาม เป็นต้น นอกจากนี้ยังมีการแก้ไขภาษาโปรแกรมแบบลำดับให้สามารถประมวลผลแบบขนานได้ เช่น ภาษาซีแบบขนาน (Parallel C) , ภาษาฟอร์แทรน (FORTRAN), ภาษาปาสคาล (PASCAL) เป็นต้น

4 ภาษาออคคาม

ภาษาออคคาม[15,16,17] เป็นภาษาระดับสูง มีโครงสร้างและข้อกำหนดของภาษาคล้ายกับภาษาระดับสูงที่ประมวลผลแบบลำดับทั่วไป ข้อพิเศษที่เพิ่มเข้ามาในภาษาออคคาม ได้แก่

- คำสั่งควบคุมวิธีการประมวลผลได้แก่ SEQ (Sequential), PAR (Parallel), ALT (Alternate)
- คำสั่งควบคุมการติดต่อผ่านลิงค์สื่อสารได้แก่ ? (Input), ! (Output)

โปรแกรมย่อยหรือฟังก์ชันของภาษาออคคาม ขณะที่ปฏิบัติการอยู่จะถูกเรียกว่าโปรเซส (Process) นอกจากนั้นการปฏิบัติงานโปรแกรมภาษาออคคาม ต้องจัดสรรฮาร์ดแวร์(Hardware allocate) ให้กับโปรเซสในภาษาออคคามก่อน โดยการเขียนโมเดลการติดตั้ง (Configuration Model) หรือการจัด Configuration รายละเอียดการเขียนโปรแกรมภาษาออคคามที่เป็นข้อพิเศษเพิ่มเติมจากภาษาอื่นๆมีดังนี้

4.1 การส่งผ่านข้อมูลระหว่างโปรเซสในภาษาออคคาม

ในโปรแกรมคอมพิวเตอร์แบบลำดับ เช่น ภาษาซี (C), ภาษาปาสคาล (Pascal) การส่งผ่านข้อมูลให้โปรแกรมย่อย ส่งได้ทั้งวิธี (1) ส่งตัวข้อมูลโดยตรง (Pass by value) และ (2) ส่งทางอ้อมโดยผ่านตัวชี้ (Pass by reference หรือ Pointer) ในภาษาออคคามจะใช้วิธีส่งแบบส่งข้อมูลโดยตรงเท่านั้น หากโปรเซสสองโปรเซสปฏิบัติการบนทรานสพิวเตอร์คนละตัว การติดต่อกันระหว่างโปรเซสจะติดต่อผ่านช่องติดต่อซึ่งมีข้อตกลง (Protocol) กำหนดไว้ล่วงหน้าว่าจะใช้ติดต่อข้อมูลชนิดใด

การประกาศตัวบอกช่องติดต่อ [15] (Declaring a Channel)

- กรณีที่ใช้ติดต่อข้อมูลพื้นฐาน ใช้คำบอกดังนี้

รูปแบบ CHAN OF data.type channel.name:

ตัวอย่าง (2) CHAN OF BYTE screen:

(3) CHAN OF BYTE screen, key:

ตัวอย่างที่ (2) แสดงการประกาศชื่อช่องติดต่อ Screen ใช้รับ/ส่งข้อมูลชนิด BYTE ส่วนตัวอย่างที่ (3) เป็นการประกาศช่องติดต่อหลายช่องพร้อมกัน และวิธีการใช้ช่องติดต่อ[8] กระทำได้โดยวิธีการดังตัวอย่างที่ (4), (5)

ตัวอย่างที่ (4) การผ่านค่าโดยใช้ช่องติดต่อรับ/ส่งข้อมูล

```
--PROCESS on A. Transputer
PROC Screen.Process (CHAN OF BYTE screen)
    screen ? screen.char   --receive screen character from screen channel
   write.to.crt (screen.char)
```

ตัวอย่างที่ (5)

```
-- Process in B.Transputer
PROC Keyboard.Process (CHAN OF BYTE screen)
  key.data := key.Press ( )
  screen ! key.data  -- send key.data to screen channel
```

จากตัวอย่างที่ (4),(5) โปรเซสในทรานสพิวเตอร์ B ทำหน้าที่รับข้อมูลจากแป้นพิมพ์ (Key board) แล้วส่งไปให้โปรเซส B โดยผ่านที่ช่องติดต่อชื่อ screen ขณะเดียวกันทรานสพิวเตอร์ A รับข้อมูลมาแล้ว จะเขียนข้อมูลนั้นออกทางจอภาพ (CRT.) แม้ว่าโปรเซสทั้ง 2 จะปฏิบัติการบนทรานสพิวเตอร์คนละตัวก็สามารถติดต่อกันได้กรณีต้องการส่งข้อมูลหลายชนิดในช่องสัญญาณเดียว กระทำได้โดยการประกาศชื่อโปรโตคอลให้เป็นชนิดแวเรียนต์ (Variant protocol) ดังตัวอย่างที่ (6),(7)

ตัวอย่างที่ (6) การใช้งานโตคอล ชนิดข้อมูลหลายชุด

-การประกาศ PROTOCOL COMPLEX IS REAL64;REAL64:

-การใช้งาน

```
PROC math.test (CHAN OF COMPLEX items)
  real.part : = 0.125(REAL)
  img.part : = 0.25(REAL)
  -- valve (0.125+j0.25)
   items ! real.part ;img.part --send 0.125+j0.25 to items chanel
```

จากตัวอย่างที่ (6) โปรโตคอลชื่อ Complex เป็นข้อตกลงให้รับ/ส่งข้อมูลชนิด REAL64 จำนวน 2 ชุด และประกาศ ช่องติดต่อ ชื่อ items ใช้รับ/ส่ง ข้อมูลที่มีโปรโตคอลแบบ Complex

ตัวอย่าง (7) การใช้งานแวเรียนต์โปรโตคอลที่ซับซ้อนยิ่งขึ้น

-การประกาศ

```
PROTOCOL Image.part
  CASE
    threshold; BYTE
```

```
        image; [90] BYTE
        halt
```

-การใช้งาน

```
--PROCESS in root.transputer
    PROC root.tsp (CHAN OF image.part to.worker)
      to.worker ! threshold; threshold.edge
      to.worker ! array ; image.array
      to worker ! halt

    --PROCESS in worker. transputer
    PROC edge.detect (CHAN OF image.part to.worker)
      to.warker ? CASE
        threshold ; threshold.edge
          ...do whatever
        array ; image.array
          ..do find.edge (image.array; threshold.edge)
        halt
          .. stop process
```

จากตัวอย่างที่ (7) จะเห็นว่าการส่งผ่านข้อมูลที่มีชนิดแตกต่างกัน โดยใช้ช่องติดต่อช่องเดียวกันมีวิธีคล้ายกับการผ่านค่าตัวแปรชนิดโครงสร้าง (structure) ในภาษา C นั่นเองโดยประกาศชื่อแทก (Tag) ของข้อมูลก่อน แล้วตามด้วยชนิดของข้อมูลที่ต้องการเช่น threshold เป็นแทกที่ใช้รับ/ส่งข้อมูลชนิด BYTE, แทก array ใช้รับ/ข้อมูลชนิดอะเรย์

โปรโตคอลชนิดสุดท้ายคือ โปรโตคอลที่ไม่มีข้อกำหนดแน่นอน (Anarchic Protocal) ข้อมูลจะถูกส่ง ที่ละไบท์เรียงกันไปโดยไม่มีการตรวจสอบ โปรโตคอลชนิดนี้ใช้ส่งข้อมูลให้อุปกรณ์ภาย นอกบางประเภท เช่น เครื่องพิมพ์ (Printer) , เครื่องเทอร์มินอล (Terminal) บางประเภท เป็นต้น

-การประกาศ

```
        CHAN OF ANY printer:
```

-การใช้งาน

```
        printer ! data.to.print
```

จากตัวอย่างที่กล่าวมาทั้งหมด จะเห็นว่าการติดต่อระหว่างโปรเซสสองโปรเซสกระทำได้ง่ายเช่นเดียวกับการผ่านค่าให้ฟังก์ชันของภาษาระดับสูงทั่วไป นอกจากจะรับส่งข้อมูลชนิดพื้นฐานแล้ว ยังสามารถรับส่งข้อมูลชนิดโครงสร้างได้เช่นกัน

4.2 การติดตั้งโปรแกรมบนทรานสพิวเตอร์ฮาร์ดแวร์

โปรแกรมภาษาออคคามสามารถปฏิบัติบนทั้งทรานสพิวเตอร์ตัวเดียว หรือ ทรานสพิวเตอร์เน็ทเวิร์คโดยที่สภาพการทำงานของโปรแกรม (Program Logical) ไม่เปลี่ยนแปลง จึงทำให้ภาษาออคคามสามารถย้ายไปปฏิบัติบนทรานสพิวเตอร์ฮาร์ดแวร์ที่จัดรูปแบบต่างๆ กันได้เพียงแต่ปรับการติดตั้งโปรแกรมให้เข้ากับกายภาพของฮาร์ดแวร์นั้นๆ (Physical Hardware) รายละเอียดทุกขั้นตอนอธิบายในเอกสารอ้างอิง [13,14] ส่วนรายละเอียดที่สำคัญมีดังนี้

4.2.1 วิธีการติดตั้งโปรแกรมบนทรานสพิวเตอร์ตัวเดียว (Single Transputer)

วิธีนี้กระทำเหมือนโปรแกรมแบบลำดับทั่วไป ตัวอย่างเช่นหากต้องการให้โปรเซสปฏิบัติแบบลำดับก็ใช้คำบอก SEQ หากต้องการให้ปฏิบัติแบบขนาน ก็ทำได้โดยใช้คำบอก PAR ดังตัวอย่างที่ (8.1)

ตัวอย่างที่ 8.1 การใช้คำสั่งควบคุมให้ภาษาออคคามปฏิบัติแบบขนาน

```
PAR  --ทั้งสามฟังก์ชัน ออคคามถือว่าเป็นโปรเซส และให้ทั้ง 3 โปรเซสปฏิบัติแบบขนาน
  terminal.process ()      -- ทุกโปรเซสมีลำดับความสำคัญเท่ากัน
  editer.process  ()
  printer.process ()
```

ถ้าต้องการให้จัดลำดับความสำคัญ ให้กับบางโปรเซสกระทำได้โดย ใช้คำบอก PRI PAR (Priority Parallel) ดังตัวอย่างที่ (8.2) โปรเซสที่มีลำดับความสำคัญสูงกว่าจะถูกปฏิบัติก่อน และสามารถขัดจังหวะโปรเซส ที่มีลำดับความสำคัญต่ำกว่าได้

ตัวอย่างที่ 8.2 การใช้คำสั่งควบคุมให้ภาษาออคคามปฏิบัติแบบขนานพร้อมกับจัดลำดับความสำคัญ

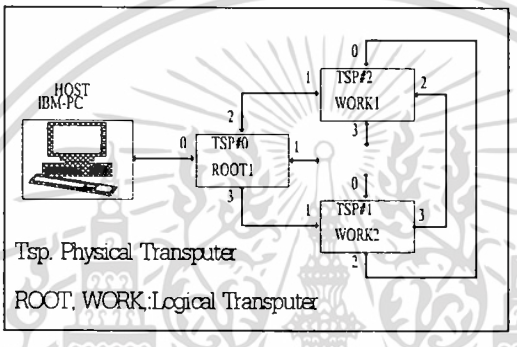
```
PRI PAR -- ให้ปฏิบัติโปรเซสแบบขนานพร้อมกับจัดลำดับความสำคัญ
  terminal.process ()      --ลำดับความสำคัญมากที่สุด
  editer.process  ()       --ลำดับความสำคัญปานกลาง
  printer.process ()       --ลำดับความสำคัญต่ำที่สุด
```

4.2.2 การติดตั้งโปรแกรมบนทรานสพิวเตอร์เน็ทเวิร์ค (Transputer Network)

กระทำได้โดยการเขียนโมเดลการติดตั้ง (Configuration Model) [13] ตามรูปแบบที่ภาษาออคคามได้กำหนดไว้ให้เข้ากับเน็ทเวิร์คที่มีอยู่ มีวิธีการดังนี้

4.2.2.1 ประกาศลักษณะของฮาร์ดแวร์ (Hardware Description)

โดยการเขียนรายละเอียดของฮาร์ดแวร์ที่ต่ออยู่ดังตัวอย่างเช่น มีทรานสพิวเตอร์ในเน็ทเวิร์ค 3 ตัว ประกอบด้วยทรานสพิวเตอร์หลัก (Root หรือ Master Processor) ต่ออยู่กับเครื่องไมโครคอมพิว-เตอร์ IBM-PC เป็น เทอร์มินอล และมีทรานสพิวเตอร์ย่อย (Sub หรือWorker Processor) 2 ตัว ดังไดอะแกรมรูป 15 โดยทรานสพิวเตอร์หลักทำหน้าที่ 2 อย่างคือ แบ่งงานให้กับทรานสพิวเตอร์ย่อย และควบคุมการติดต่อกับเครื่องเทอร์มินอล ส่วนทรานสพิวเตอร์ทำงานทำหน้าที่รับงานจากทรานสพิวเตอร์หลัก แล้วประมวลผลจนเสร็จจึงส่งผลคืนให้ทรานสพิวเตอร์หลักต่อไป



รูปที่ 15 แสดงการต่อฮาร์ดแวร์ทรานสพิวเตอร์เน็ทเวิร์ค

4.2.2.2 การแทนที่ความสัมพันธ์ระหว่างข้อมูล (Mapping)

เพื่อจัดสรรโปรเซสเซอร์ทางกายภาพ ให้กับโปรเซสเซอร์ทางจินตภาพ (Logical) กระทำโดยเขียนชื่อของโปรเซสเซอร์ทางจินตภาพให้ Mapping ลงบนโปรเซสเซอร์ทางจินตภาพโดยที่จำนวนโปรเซสเซอร์ทางจินตภาพหลายๆตัวสามารถ Mapping ลงบนโปรเซสเซอร์ทางกายภาพตัวเดียวกันได้

4.2.2.3 การประกาศลักษณะของซอฟต์แวร์ (Software Description)

เพื่อจัดสรรโปรเซสทางกายภาพและช่องสัญญาณติดต่อทางกายภาพ ให้กับโปรเซสทางจินตภาพและช่องสัญญาณติดต่อทางจินตภาพ โดยการประกาศเช่นเดียวกับการ Mapping ฮาร์ดแวร์

ตัวอย่างที่ 9 แสดงขั้นตอนการเขียนโมเดลการติดตั้งภาษาออคคาม

```
-- ส่วนประกาศลักษณะของฮาร์ดแวร์ (Hardware Description)
-- ส่วนประกาศตัวแปร
VAL Kbyte                IS 1024:
VAL Mbyte                IS Kbyte*Kbyte:
VAL number.of.transputers    IS 3:
VAL type.of.transputers      IS "T805":
[number.of.transputers] NODE transputers:
```

- ส่วนประกาศลักษณะของฮาร์ดแวร์เน็ทเวิร์ค

```
ARC hostlink:
NETWORK   --ประกาศลักษณะเน็ทเวิร์ค
DO
  DO i=0 FOR number.of.transputers -- บอกชนิดและจำนวนของทรานสพิวเตอร์
    SET transputers[i] (type ,memsize :=type.of.transputers,1*Mbyte)
-- ส่วนกำหนดการเชื่อมต่อลิงค์เส้นต่างๆ
CONNECT transputers[0][link][0]  TO HOST WITH hostlink
CONNECT transputers[0][link][2]  TO transputers[1][link][1]
CONNECT transputers[0][link][3]  TO transputers[2][link][1]
CONNECT transputers[1][link][2]  TO transputers[2][link][3]
CONNECT transputers[1][link][0]  TO transputers[2][link][2]
-- ส่วนMappingฮาร์ดแวร์ทางจินตภาพลงบนฮาร์ดแวร์ทางกายภาพ
VAL number.of.work.p     IS 2:
VAL root.processor.id     IS 0:
NODE root.processor:
[number.of.work.p]NODE work.p:
MAPPING
  DO
    MAP root.processor ONTO transputers[root.processor.id]
    MAP work.p[0] ONTO transputers[1]
    MAP work.p[1] ONTO transputers[2]
:
-- ส่วนกำหนดลักษณะของซอฟต์แวร์ (Software Description)
#INCLUDE "hostio.inc"
#INCLUDE "tspi.inc"
#USE "tsp_mn.lku"
#USE "tsp_sa.lku"
#USE "tsp_sb.lku"
```

```
CONFIG
-- กำหนดช่องสัญญาณทางจินตภาพลงบนช่องสัญญาณทางกายภาพ
CHAN OF SP fs,ts:
PLACE fs,ts ON hostlink:
CHAN OF MASSEGE rta1,a1rt,rta2,a2rt,a1a2,a2a1:  -- ส่วนนี้ Mapping อัตโนมัติ
-- กำหนดให้โปรเซสให้กับโปรเซสเซอร์แต่ละตัว และให้ทำงานแบบขนาน
PLACED PAR
  PROCESSOR root.processor  -- งานของโปรเซสเซอร์หลัก
    master.tsp (fs,ts,rta1,a1rt,rta2,a2rt)
  PROCESSOR work.p [0]      -- งานของโปรเซสเซอร์ย่อยตัวที่ 1
    sub.trial.a (rta1,a1rt,a1a2,a2a1)
  PROCESSOR work.p [1]      -- งานของโปรเซสเซอร์ย่อยตัวที่ 2
    sub.trial.b (rta2,a2rt,a1a2,a2a1)
```

4.3 ข้อกำหนดพิเศษของภาษาออคคามในการโปรแกรมแบบขนาน

ข้อกำหนดพิเศษสำหรับโปรแกรมภาษาออคคาม[15] ที่สำคัญคือการใช้ตัวแปรร่วม (Sharing Variable) ภาษาออคคามได้ตั้งกฎการใช้ตัวแปรไว้ดังนี้

1 ตัวแปรที่เป็น อะเรย์ (Array) จะถูกใช้ในโปรเซสเดียวเท่านั้น ยกเว้นว่าการใช้อะเรย์นั้นรู้ค่าตัวชี้ (Index of Array)ในขณะแปล และต้องไม่เกิดการเขียนทับซ้อนด้วย

2 ตัวแปรอะเรย์ จะต้องไม่เกิดการเขียนทับซ้อน (Overlab) ในตำแหน่งเดียวกัน

3 ไม่กำหนดค่าให้ตัวแปร หรือ อะเรย์ ตัวเดียวกันหลายๆ โปรเซส

4 ตัวแปรชนิด แชนเนล เป็นแบบทิศทางเดียว ด้านหนึ่งใช้ส่งอีกด้านหนึ่งต้องใช้รับ หรือทำในทิศทางตรงข้ามกันได้ แต่ให้เป็นส่ง หรือ รับ อย่างเดียวพร้อมกันไม่ได้

ตัวอย่างที่ 10 วิธีการใช้ตัวแปรที่ถูกและผิด

ผิด	ถูก
PAR	PAR
a := x -- กำหนดค่าซ้ำ	a1 := x
a := y	a2 := y
array [1] := x -- เขียนทับซ้อน	array [1] := x
array [1] := y -- บนอะเรย์	array [2] := y

5 การประมวลผลภาพแบบขนาน

การแทนภาพบนคอมพิวเตอร์[8] ทำได้หลายวิธีด้วยกันตัวอย่างเช่นการแทนโดยอะเรย์ จุดภาพแต่ละจุดถูกแทนโดยข้อมูล 1 ส่วน (Element) ซึ่งอาจจะใช้ขนาด 1 ไบท์หรือมากกว่าขึ้นอยู่ กับ ความ ต้องการความสมบูรณ์ของข่าวสาร (Information) ของภาพเพียงไร ในระบบการประมวลผลภาพชนิดเกรย์สเกล (Gray scale) ข้อมูลภาพแต่ละจุดจะถูกแทนโดยข้อมูลขนาด 1 ไบท์ ค่าที่ถูกเก็บในข้อมูลคือค่าความสว่าง เริ่มจาก 0 (มืด) และค่าความสว่างสูงสุดคือ 255 ส่วนค่าอื่นๆ จะลดหลั่นกันตามอัตรา ส่วนขนาดความกว้างของภาพในการทดลองของบทความนี้ใช้ขนาด 256*256 จุดในการวิจัยครั้งนี้ได้ทดสอบการประมวลผลภาพ 2 หัวข้อคือ

1 การกำจัดสัญญาณรบกวนของภาพโดยการกรองสัญญาณรบกวนออกจากภาพ

2 การค้นหาขอบภาพ

5.1 การกำจัดสัญญาณรบกวนภาพ

สัญญาณรบกวนของภาพกรณีภาพชนิดเกรย์สเกล คือจุดภาพที่มีความเด่นชัดผิดปกติหรือผิดแปลกออกไปจากจุดอื่นๆในบริเวณใกล้เคียง ซึ่งสัญญาณรบกวนอาจเป็นจุดขาวในบริเวณรอบๆ เป็นจุดดำหรืออาจเป็นจุดดำในขณะที่บริเวณรอบๆเป็นจุดขาวก็ได้สัญญาณรบกวนของภาพเป็นอุปสรรคอย่างหนึ่งในการประมวลผลภาพ เพราะจะทำให้การประมวลผลภาพบางกรรมวิธีเกิดความผิดพลาดได้ การกำจัดสัญญาณรบกวน[1,8,9]จึงมีความจำเป็นมาก วิธีกำจัดคือการทำให้จุดภาพที่เด่นชัดผิดปกติหายไป หรือลดระดับความเด่นชัดลงมาให้กลมกลืนกับจุดภาพในบริเวณใกล้เคียง

วิธีการกำจัดสัญญาณรบกวนภาพมีหลายวิธี เทคนิควิธีที่ใช้ในการวิจัยครั้งนี้คือ การนำค่าระดับความสว่างของจุดภาพในย่านใกล้เคียงมาเฉลี่ย โดยวิธีหาค่าเฉลี่ย 2 วิธีเปรียบเทียบกันคือ (1) ค่าเฉลี่ยเรขาคณิต(Geometric Mean) (2) ค่าเฉลี่ยค่ากลาง (Median) ในการหาค่าระดับความสว่างของ ภาพทุก ๆ จุด จะนำกลุ่มภาพในย่านใกล้เคียงรอบ ๆ จุดนั้นมาพิจารณาขนาดที่ใช้พิจารณาคือ 3*3 จุดดังแสดงในรูปที่ 16 .สมการของจุดภาพใหม่จากการหาค่าเฉลี่ยค่ากลางคือ สมการที่ (5) และสมการที่ได้จากการหาค่าเฉลี่ยเรขาคณิตคือสมการที่ (6)

$$g(m,n) = \text{Median}\, f(m-p,n-q) \qquad (5)$$

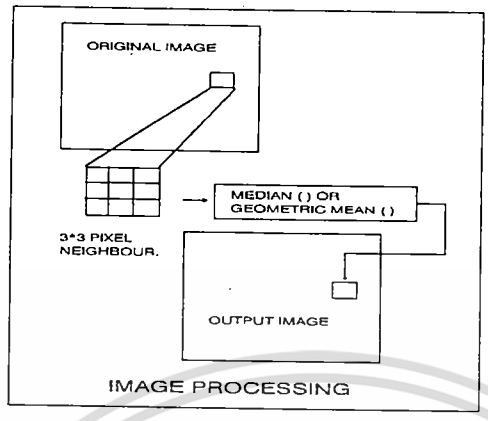
$g(m,n)$ คือระดับความสว่างของจุดภาพเอาท์พุท

$f(m-p,n-q)$ คือระดับความสว่างของจุดภาพย่านใกล้เคียงทั้ง 9 จุด

$$g(m,n) = \frac{10}{n}\log x_i \qquad (6)$$

n คือจำนวนจุดภาพที่นำมาเฉลี่ย (9 จุด)

x_i คือระดับความสว่างของจุดภาพย่านใกล้เคียงทั้ง 9 จุด



รูปที่ 16 แสดงวิธีการกำจัดสัญญาณรบกวนภาพโดยการกรอง

เหตุผลของการเลือกใช้วิธีการหาค่าเฉลี่ยทั้งสองวิธี เพราะต้องการเปรียบเทียบขีดความสามารถในการประมวลผลของโปรเซสเซอร์ กรณีที่ (1) คำนวนฟังก์ชันคณิตศาสตร์ที่ซับซ้อน (ฟังก์ชันลอกการิทึม, ยกกำลัง ในวิธีเฉลี่ยเรขาคณิต) และ (2) กรณีคำนวนแบบฟังก์ชันง่ายๆ (ฟังก์ชันการเรียงข้อมูล ในวิธีเฉลี่ยค่ากลาง)

5.2 การหาขอบภาพ (Edge datection)

ขอบภาพ[8] คือจุดภาพส่วนที่มีการเปลี่ยนแปลงระดับความเข้มอย่างรวดเร็ว เช่น การเปลี่ยนแปลงจากดำเป็นขาวหรือขาวเป็นดำเป็นต้น ซึ่งขอบภาพนี้มีคุณสมบัติเฉพาะของแต่ละภาพ ในการประมวลผลภาพบางกรรมวิธีอาจใช้แค่ขอบภาพแทนภาพทั้งหมดได้ จะทำให้ประหยัดพื้นที่หน่วยความจำและประมวลผลได้รวดเร็วขึ้น ภาพชนิดที่มีความเข้ม 2 ระดับ (Binary Image) ตัวอย่างขอบภาพเช่นจุดภาพสีดำที่มีสีขาวอยู่ล้อมรอบอย่างน้อย 1 จุด สมการของขอบภาพชนิดความเข้ม 2 ระดับคือสมการที่ 7

$$g(m,n) = \left[u(m,n) \oplus u(m \pm 1)\right].OR.\left[u(m,n) \oplus u(m,n \pm 1)\right] \qquad (7)$$

(m,n) จุดภาพที่พิจารณา

g(m,n) จุดที่เป็นขอบภาพ

u(m,n) จุดภาพต้นฉบับ

$\oplus$ เอ็กซ์คลูสซิฟออร์ (Exclusive OR)

กรณีภาพที่มีความเข้มต่อเนื่องกัน (Continuous image) เช่นภาพเกรย์สเกล การหาขอบภาพกระทำโดยวิธีต่างๆดังต่อไปนี้

5.2.1 วิธีวัดแกรเดียน[8] เป็นวิธีการหาขอบภาพ โดยการวัดค่าอนุพันธ์ (Derivative) ความเข้มของแสงสว่างในทิศทางใดทิศทางหนึ่ง ดังรูปที่ 17 ส่วนสมการที่ใช้คำนวนได้จากสมการที่ 8

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial r}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} = f_x \cos\theta + f_y \sin\theta \qquad (8)$$

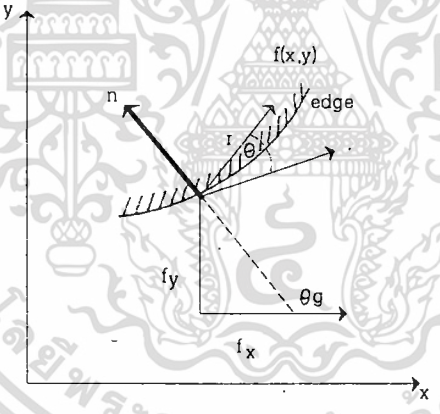
ค่าสูงสุดของ $\frac{\partial f}{\partial r}$ จะเกิดขึ้นเมื่อ $\left(\frac{\partial}{\partial \theta}\right)\left(\frac{\partial f}{\partial r}\right) = 0$

และจะส่งผลให้

$$f_x \sin\theta_g + f_y \sin\theta_g = 0 \;\text{-->}\; \theta_g = \tan^{-1}\left(\frac{f_y}{f_x}\right) \qquad (9)$$

$$\left(\frac{\partial f}{\partial r}\right)_{max} = \sqrt{f_x^{\,2} + f_y^{\,2}} \qquad (10)$$

θ มุมของแกรเดียน



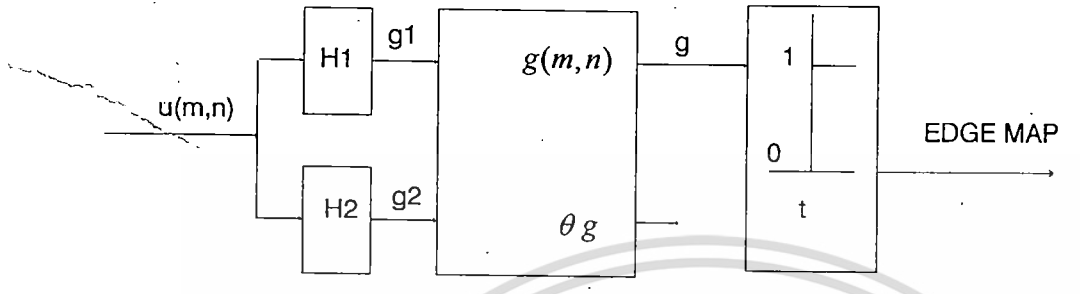
รูปที่ 17 แสดงแกรเดียนของฟังก์ชัน f (x,y) ในทิศทาง r

วิธีวัดแกรเดียนของภาพในคอมพิวเตอร์กระทำโดยใช้ตัวกระทำ (Operators) ที่เป็นกรอบรูปสี่เหลี่ยมจัสตุรัส (Mask หรือ Kernel) ขนาดต่างๆ กัน ตัวกระทำชนิดกรอบทำงานโดย ให้กรอบดังกล่าวเลื่อนกวาดไปทุก ๆ จุดภาพ ผลลัพท์ที่ได้คือจุดภาพต้นฉบับ ถูกกระทำโดยตัวกระทำชนิดกรอบดังสมการที่ (11) (สมการสหสัมพันธ์ หรือ Correlation equation)

$$\langle U, H \rangle_{m,n} = \sum_i \sum_j h(i,j)u(i+m, j+n) = u(m,n) \otimes h(-m,-n) \qquad (11)$$

H กรอบสี่เหลี่ยมจัสตุรัส ขนาด PXP

U ภาพต้นฉบับ



รูปที่ 18 แสดงวิธีการหาขอบภาพโดยใช้ตัวกระทำชนิดแกรเดียน

ขนาดของแกรเดียน (Gradient vector magnitude) และทิศทาง (Direction) หาได้จากสมการ 12 และ 13 โดยรูปที่ 18 แสดงวิธีหาขอบภาพโดยการวัดแกรเดียน

$$g(m,n) = \sqrt{g_1^2(m,n) + g_2^2(m,n)} \qquad (12)$$

$$\theta g(m,n) = \tan^{-1} \frac{g_2(m,n)}{g_1(m,n)} \qquad (13)$$

g1 ขนาดของแกรเดียนในทิศทาง x

g2 ขนาดของแกรเดียนในทิศทาง y

θg ทิศทางของแกรเดียน

ในการประมวลผลด้วยคอมพิวเตอร์ ขนาดของแกรเดียนอาจหาได้จากสมการ (14) ซึ่งคำนวณโดยคอมพิวเตอร์ได้สะดวกกว่าสมการที่ (12)

$$g(m,n) = |g_1(m,n)| + |g_2(m,n)| \qquad (14)$$

หลังจากได้ขนาดของแกรเดียนมาแล้ว ขั้นตอนต่อไปคือ ตัดสินว่าจะให้เป็นขอบภาพเมื่อแกรเดียนมีขนาดเท่าไร การทำขั้นตอนนี้ทำโดยกำหนดค่าเทรสโฮสด์ (Threshold) ขึ้นมาเพื่อเปรียบเทียบกับแกรเดียนที่วัดได้ แล้วตัดสินให้เป็นขอบภาพหรือไม่ได้ทันที จากสมการที่ (15), (16)

$$\varepsilon(m,n) = \begin{cases} 1, (m,n) \subset Ig \\ 0, \text{หรือค่าอื่นๆ} \end{cases} \quad (15)$$

$$Ig = \{(m,n); g(m,n) > threshold\} \quad (16)$$

ค่าเทรสโฮลด์ กำหนดได้จากฮิสโตแกรม (Histogram) ของแกรเดียนทั้งหมด แล้วเลือกใช้ที่ 5-10 เปอร์เซนต์ของจุดภาพที่มีแกรเดียนสูงสุด วิธีหาขอบภาพจากการวัดแกรเดียนด้มีผู้นำไปคิดค้นเป็นทฤษฎีขึ้นมาหลายทฤษฎี [8,9] เช่น ทฤษฎีของพรีเวต (Prewet), โซเบล (Sobel), ไอโซทรอปิค (Isotropic) เป็นต้น ทั้ง 3 ทฤษฎี จะมีการค่าสัมประสิทธิ์รวมของกรอบเท่ากับศูนย์ ส่วนตัวอย่างกรอบที่เป็นตัวกระทำ ตามทฤษฎีต่าง ๆ แสดงในตารางที่ (1) การหาขอบภาพโดยวิธีวัดแกรเดียนเหมาะกับภาพที่มีโครงขอบภาพปรากฏชัดเจน บางกรณีภาพที่มีการเปลี่ยนแปลงความเข้มอย่างราบเรียบจะไม่ปรากฏขอบภาพชัดเจนนัก วิธีหาขอบภาพวิธีนี้จะหาขอบไม่พบ เกิดความผิดพลาดได้ง่าย ต่อไปจะกล่าวถึงวิธีหาขอบภาพที่ให้ความผิดพลาดน้อยลงคือ ตัวกระทำมาร์ร์ และ ตัวกระทำกาบอร์ ซึ่ง 2 วิธีนี้มีทฤษฎีการคำนวณหาที่มาสลับซับซ้อน จึงขอกล่าวถึงเฉพาะวิธีประยุกต์ใช้งาน ส่วนรายละเอียดศึกษาได้จากเอกสารอ้างอิง

	H1	H2
PREWITT	-1 0 1 -1 0 1 -1 0 1	-1 -1 -1 0 0 0 1 1 1
SOBEL	-1 0 1 -2 0 2 -1 0 1	-1 -2 -1 0 0 0 1 2 1
ISOTROPIC	-1 0 1 $-\sqrt{2}$ 0 $\sqrt{2}$ -1 0 1	-1 $-\sqrt{2}$ -1 0 0 0 1 $\sqrt{2}$ 1

ตารางที่ (1) แสดงกรอบตัวกระทำของทฤษฎีการหาขอบภาพวิธีต่าง ๆ

5.2.2 ตัวกระทำมาร์ร์ (Marr's Operator)

D. Marr เป็นผู้คิดค้นทฤษฎีนี้[7,9] เพื่อให้การตรวจจับการเปลี่ยนแปลงความเข้มของแสง ในขอบเขตที่กว้างกว่าวิธีวัดแกรเดียน โดยการใช้ตัวกระทำหลายตัวทำงานร่วมกันดังสมการที่ 17

$$\nabla^2 G * I \quad (17)$$

I เป็นภาพที่ต้องการหาขอบ, ∇^2 เป็นตัวกระทำลาปาเชียล (Laplacian) ทำหน้าที่กรองเอาเฉพาะส่วนที่มีการเปลี่ยนแปลงความเข้มของแสงเพื่อหาขอบ, G เป็นตัวกระทำการแพร่กระจายแบบเกาซ์เซียน 2 มิติ (Two-dimensional Gaussian distribution) ทำหน้าที่ปรับย่านการเปลี่ยนแปลงความเข้มให้เรียบขึ้นและช่วยลดสัญญาณรบกวน สมการสมบูรณ์ของตัวกระทำมาร์คือ สมการที่ 18 ขอบภาพที่ได้คือ ผลลัพท์ของสมการที่ได้คำตอบเป็นศูนย์ดังสมการที่ 19 ส่วนกรอบที่ใช้เป็นตัวกระทำของมาร์ร์ โดยใช้ค่า σ = 4 แสดงในตารางที่ 2

$$\nabla^2 G(r) = \left(-1/\pi\sigma^4\right)\left[1 - r^2/2\sigma^2\right]\exp\left(-r^2/2\sigma^2\right) \qquad (18)$$

$$g(m,n) = \begin{cases} 1, \nabla^2 G * I = 0 \\ 0, \text{กรณีอื่นๆ} \end{cases} \qquad (19)$$

I คือรัศมี $\sqrt{(m^2+n^2)}$

σ คือส่วนเบี่ยงเบนมาตรฐาน

5.2.3 ตัวกระทำกาบอร์ (Gabor operator)

เป็นตัวกระทำกรองสัญญานชนิดหนึ่งใช้หลักการมอดูเลท (Modulate) ผลคูณระหว่างเกาซ์เซียนกับฟังก์ชันซายน์ (Sinusoidal) [5,6] สมการของตัวกระทำกาบอร์คือสมการที่ 20 ขอบภาพที่ได้คือผลลัพท์ของสมการที่ให้ค่ามากกว่าค่าเทรสโฮลด์เช่นเดียวกับวิธีของโซเบลดังสมการที่ 15, 16 ตารางกรอบของตัวกระทำกาบอร์เป็นขนาด 11*11 โดยใช้ $\sigma = 4$ และ ω = 9/10 แสดงดังตารางที่ 3

$$h(m,n) = \exp\left[-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma^2}\right)\right]\sin(\omega n) \qquad (20)$$

โดน $\omega = 2\pi f$ และ f คือความถี่ของสัญญาณซายน์ที่มอดดูเลท

0.028	0.027	0.024	0.021	0.017	0.016	0.017	0.021	0.024	0.027	0.028
0.027	0.023	0.016	0.008	0.002	0.000	0.002	0.008	0.016	0.023	0.027
0.024	0.016	0.004	-0.008	-0.017	-0.021	-0.017	-0.008	0.004	0.016	0.024
0.021	0.008	-0.008	-0.024	-0.037	-0.041	-0.037	-0.024	-0.008	0.008	0.021
0.017	0.002	-0.017	-0.037	-0.051	-0.057	-0.051	-0.037	-0.017	0.002	0.017
0.016	0.000	-0.021	-0.041	-0.057	-0.062	-0.057	-0.041	-0.021	0.000	0.016
0.017	0.002	-0.017	-0.037	-0.051	-0.057	-0.051	-0.037	-0.017	0.002	0.017
0.021	0.008	-0.008	-0.024	-0.037	-0.041	-0.037	-0.024	-0.008	0.008	0.021
0.024	0.016	0.004	-0.008	-0.017	-0.021	-0.017	-0.008	0.004	0.016	0.024
0.027	0.023	0.016	0.008	0.002	0.000	0.002	0.008	0.016	0.023	0.027
0.028	0.027	0.024	0.021	0.017	0.016	0.017	0.021	0.024	0.027	0.028

ตารางที่ 2 กรอบที่ใช้เป็นตัวกระทำมาร์ $\nabla^2 G(r)$ ขนาด 11*11 โดยใช้ $\sigma = 4$

0.000	0.000	0.000	-0.003	-0.011	-0.018	-0.014	-0.005	-0.001	0.000	0.000
0.000	0.000	-0.005	-0.030	-0.080	-0.103	-0.064	-0.017	0.000	0.000	0.000
0.000	-0.003	-0.030	-0.132	-0.280	-0.288	-0.125	0.000	0.017	0.000	0.001
-0.001	-0.011	-0.080	-0.280	-0.475	-0.339	0.000	0.125	0.064	0.010	0.001
-0.002	-0.018	-0.103	-0.288	-0.339	0.000	0.339	0.288	0.103	0.010	0.002
-0.001	-0.014	-0.064	-0.125	0.000	0.339	0.475	0.280	0.080	0.010	0.001
-0.001	-0.005	-0.017	0.000	0.125	0.288	0.280	0.132	0.030	0.000	0.000
0.000	-0.001	0.000	0.017	0.064	0.103	0.080	0.030	0.005	0.000	0.000
0.000	0.000	0.001	0.005	0.014	0.018	0.011	0.003	0.000	0.000	0.000
0.000	0.000	0.000	0.001	0.001	0.002	0.001	0.000	0.000	0.000	0.000

ตารางที่ 3 กรอบที่ใช้เป็นตัวกระทำกาบอร์ h(m,n) ขนาด 11*11 โดยใช้ $\sigma = 4$ และ ω = 9/10

6 ผลการทดสอบ

การทดสอบผลในการวิจัยนี้ได้แบ่งหัวข้อการทดสอบเป็น 2 หัวข้อคือการทดสอบผลลัพท์ของภาพที่ได้จากการประมวลผล และทดสอบความสามารถในด้านความเร็ว, ประสิทธิภาพในการประมวลผล

6.1 ผลการทดสอบการประมวลผลภาพ

6.1.1 ผลการทดสอบการกำจัดสัญญาณรบกวน โดยรูปที่ใช้ทดสอบเป็นรูปที่มีสัญญาณ รบกวนเป็นจุดๆ แบบสุ่ม (Random) ดังรูปที่ 19 ได้ผลลัพท์จากวิธีการเฉลี่ยแบบหาค่ากลางดังรูปที่ 20 และผลลัพท์จากวิธีการเฉลี่ยแบบเรขาคณิตดังรูปที่ 21 ซึ่งจะสังเกตุเห็นว่าสัญญาณรบกวนแบบจุดที่เกิดขึ้นนั้นหายไป แต่ยังมีรอยด่างเกิดขึ้นบริเวณรอบๆจุดรบกวนจุดเดิมอยู่ สาเหตุเนื่องจากการเฉลี่ยค่าไปจากสัญญาณรบกวนนั่นเอง

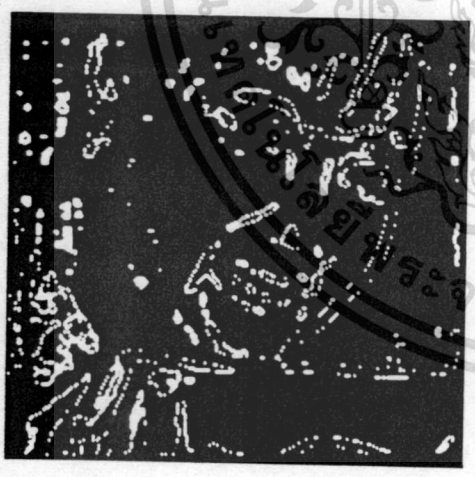
รูปที่ 19 รูปต้นแบบที่มีสัญญาณรบกวน

รูปที่ 20 รูปที่ได้จากการกำจัดสัญญาณรบกวน โดยการกรองแบบค่าเฉลี่ยค่ากลาง

รูปที่ 21 รูปที่ได้จากการกำจัดสัญญาณรบกวน โดยการกรองแบบค่าเฉลี่ยเรขาคณิต

รูปที่ 22 ภาพต้นฉบับที่ไม่ปรากฏขอบชัดเจน

รูปที่ 23 ขอบภาพที่ได้จากวิธีโซเบล
โดยค่าเทรสโฮลด์ = 7

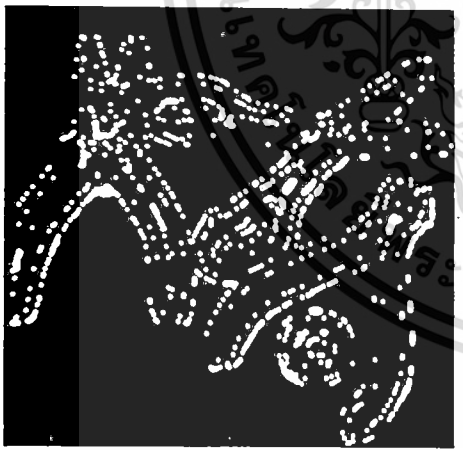


รูปที่ 24 ขอบภาพที่ได้จากวิธีมาร์
โดยค่า σ = 4

รูปที่ 25 ขอบภาพที่ได้จากวิธีกาบอร์
โดยค่าเทรสโฮลด์ =15 , σ = 4

รูปที่ 26 ภาพต้นฉบับที่ปรากฏขอบชัดเจน

รูปที่ 27 ขอบภาพที่ได้จากวิธีโซเบล
โดยค่าเทรสโฮลด์ = 7



รูปที่ 28 ขอบภาพที่ได้จากวิธีมาร์
โดยค่า σ = 4

รูปที่ 29 ขอบภาพที่ได้จากวิธีกาบอร์
ค่าเทรสโฮลด์ = 15 , σ = 4

6.1.2 ผลการทดสอบการหาขอบภาพในการทดลองใช้ภาพเกรย์สเกลขนาด 256*256 จุดจำนวน 2 ภาพ แบ่งเป็นภาพที่มีขอบปรากฏชัดเจน (รูปที่ 22) และภาพที่ไม่ปรากฏขอบชัดเจน (รูปที่ 26) นำมาหาขอบภาพตามวิธีต่างๆ 3 วิธี โดยวิธีโซเบลใช้กรอบตัวกระทำขนาด 3*3 จากตารางที่ 1, วิธีมาร์ใช้กรอบ ตัวกระทำขนาด 11*11 จากตาราง ที่ 2 และวิธีกาบอร์ใช้กรอบตัวกระทำขนาด 11*11 จากตารางที่ 3 ได้ผลการทดลองดังนี้

-โดยรูปที่ 22, 23, 24, 25 คือรูปต้นฉบับภาพที่ไม่ปรากฏขอบชัดเจน ผ่านกระบวนการหาขอบภาพตามวิธีของโซเบลใช้ค่าเทรสโฮลด์เท่ากับ 7, วิธีของมาร์ และวิธีของกาบอร์ใช้ค่าเทรสโฮลด์เท่ากับ 15 ค่าส่วนเบี่ยงเบนมาตรฐานเท่ากับ 4, ค่า = 9/10 ตามลำดับ ผลการทดสอบกับภาพที่ไม่ปรากฏขอบชัดเจนวิธีของมาร์ และ กาบอร์ สามารถหาขอบได้ถูกต้องกว่า

-และรูปที่ 26, 27, 28, 29 คือผลการทดสอบกับภาพที่มีขอบปรากฏชัดเจน ภายใต้เงื่อนไขเดียวกันกับภาพต้นฉบับที่ไม่ปรากฏขอบชัดเจนตามลำดับ ผลการทดสอบกับภาพที่มีขอบชัดเจนทั้งสามวิธีสามารถหาขอบได้ถูกต้อง และขอบที่ได้จากวิธีของกาบอร์มีขนาดเส้นหนากว่าเช่นเดียวกัน

6.2 ผลการวัดและเปรียบเทียบการประมวลผลแบบขนาน

การทดสอบความเร็วและประสิทธิภาพของการประมวลผลใช้ฮาร์ดแวร์ประกอบด้วยเครื่องไมโครคอมพิวเตอร์ IBM PC-AT ซีพียู 80386SX-16 หน่วยความจำ 2 เมกกะไบท์ ทำหน้าที่เป็นเทอร์มินอล (Terminal) ของระบบ ส่วนเน็ทเวิร์คของทรานสพิวเตอร์ ประกอบด้วยแผงวงจรหลัก (Mother Board) ของทรานสพิวเตอร์เน็ทเวิร์ค รุ่นTBX03 และโมดูลทรานสพิวเตอร์ 3 โมดูล ติดตั้งบนแผงวงจรหลัก แต่ละโมดูลประกอบด้วย ทรานสพิวเตอร์เบอร์ T805, ความเร็ว 25 เมกกะเฮริตส์ หน่วยความจำ 1 เมกกะไบท์ ทั้ง 3 โมดูลเชื่อมต่อกัน ด้วยลิงค์สื่อสาร จัดเป็นเน็ทเวิร์คดังรูปที่ 15. ทรานสพิวเตอร์ตัวที่ 1 เป็นโปรเซสเซอร์หลัก นอกจากทำหน้าที่ประมวลผลทั่วไปแล้วยังทำหน้าที่เชื่อมต่อกับบัสของเครื่องเทอร์มินอลด้วย ส่วนทรานสพิวเตอร์ตัวที่ 2 และตัวที่ 3 เป็นโปรเซสเซอร์ใช้งานทั่วไป

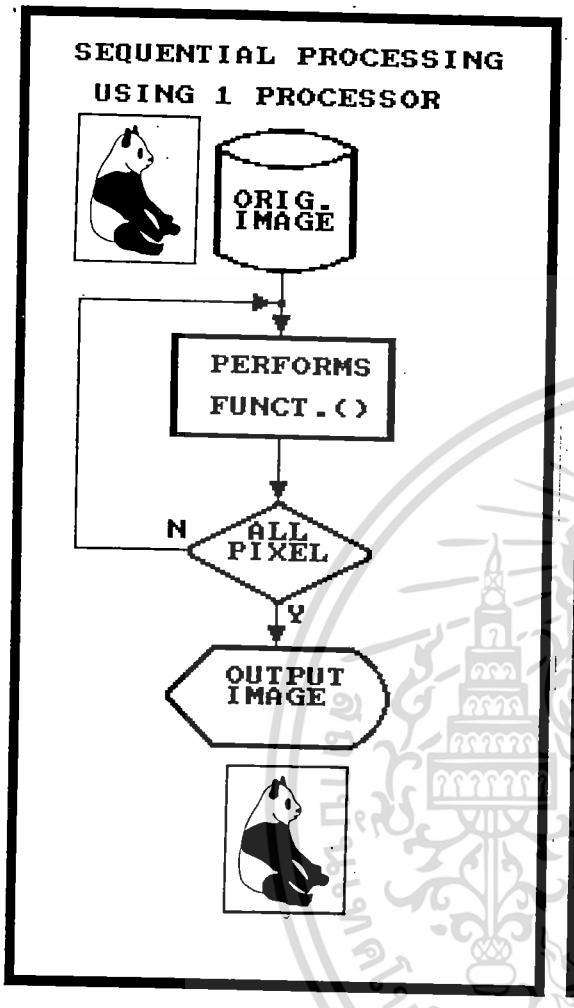
การวัดขีดความสามารถและประสิทธิภาพการประมวลผล โดยจัดอัลกอริทึมเป็น 4 วิธีดังนี้

1 อัลกอริทึมแบบลำดับที่ไม่มีการแบ่งงาน ใช้ทรานสพิวเตอร์ตัวเดียวประมวลผลดังแสดงในโฟลว์ชาร์ตรูปที่ 30

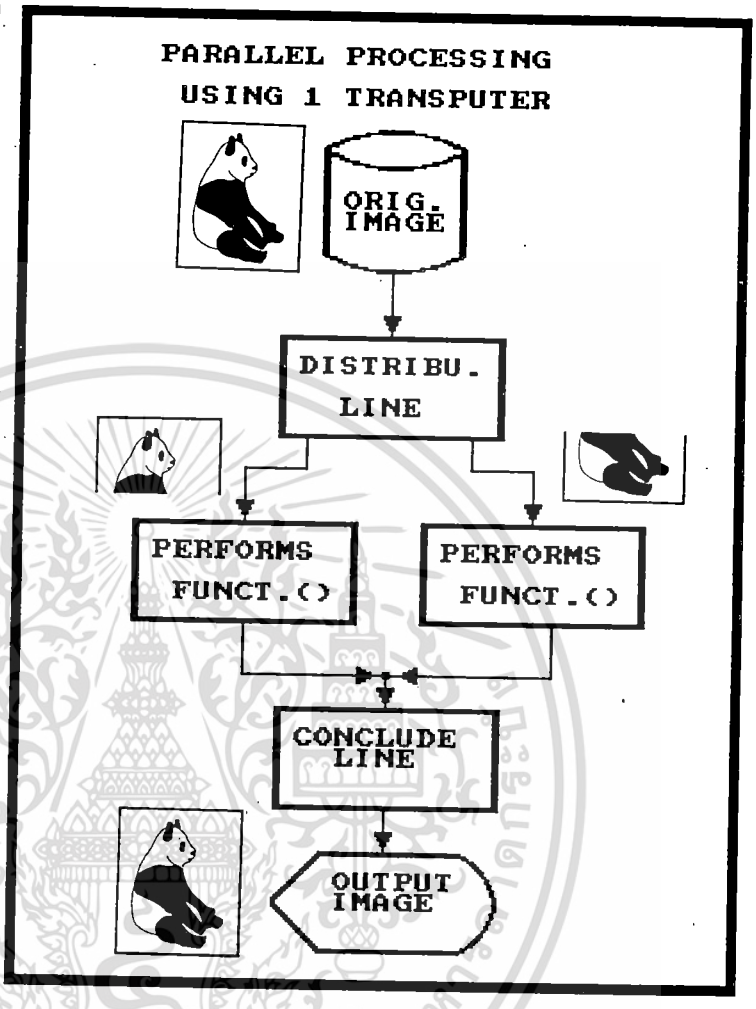
2. อัลกอริทึมแบบขนาน ใช้ทรานสพิวเตอร์ตัวเดียวจำลองให้ทำงานแบบขนานโดยแบ่งภาพแนวนอนเป็น 2 ส่วนคือส่วนบน (เส้นที่ 0-127) และส่วนล่างคือเส้นที่ (128-256) ดังแสดงในโฟลว์ชาร์ตรูปที่ 31

3 อัลกอริทึมแบบขนาน ใช้ทรานสพิวเตอร์ 2 ตัว แบ่งงานเช่นเดียวกับอัลกอริทึมในข้อ 2 ดังแสดงในโฟลว์ชาร์ตรูปที่ 32

4 อัลกอริทึมแบบขนานใช้ทรานสพิวเตอร์ 3 ตัว แบ่งในลักษณะเดียวกัน แต่แบ่งเป็น 3 ส่วน คือ ส่วนบน (เส้นที่ 0-84), ส่วนกลาง (เส้นที่ 85-169) และส่วนล่าง (เส้นที่ 170-255) ดังแสดงในโฟลว์ชาร์ตรูปที่ 33

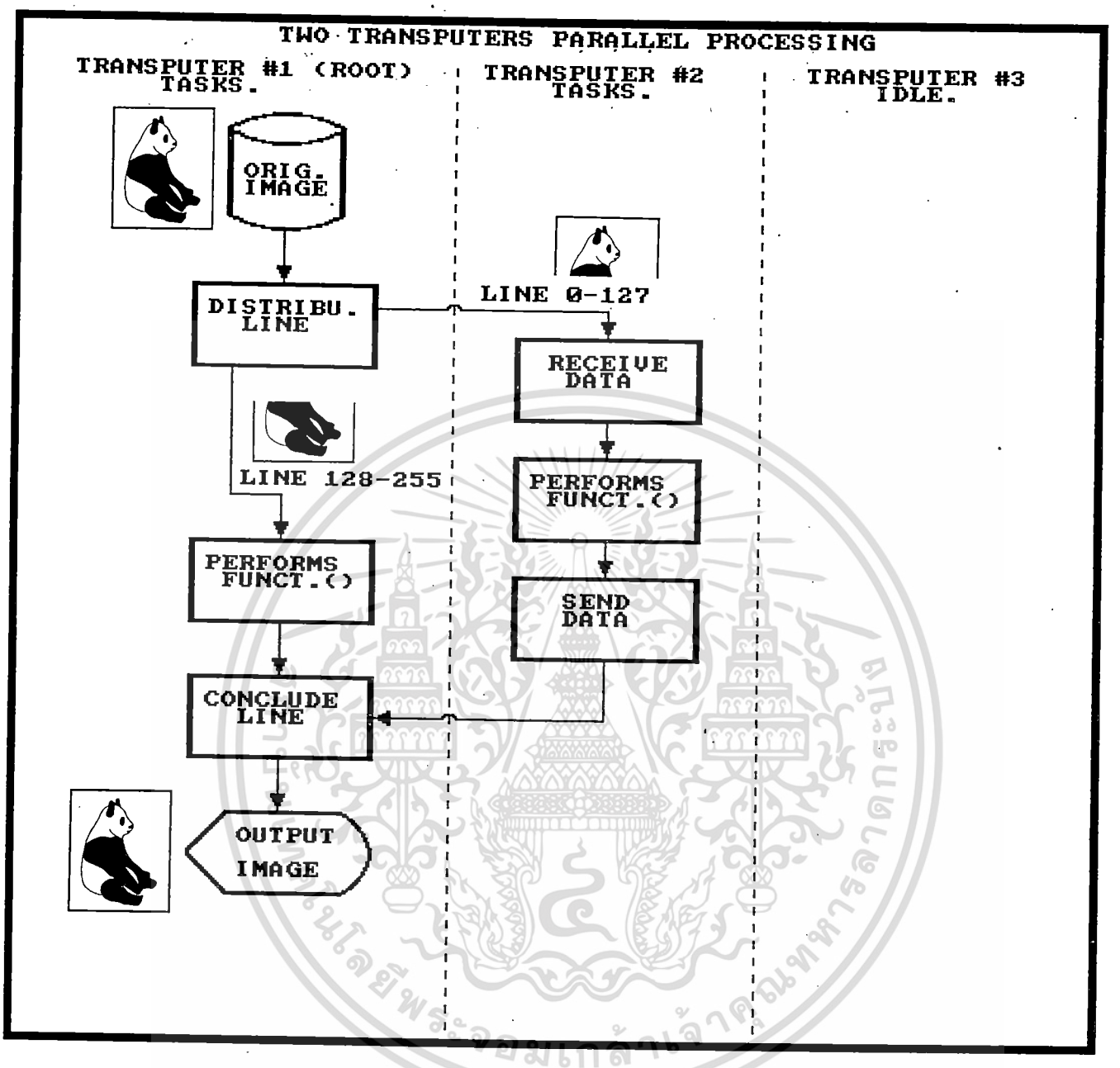


รูปที่ 30 โฟล์วชาร์ตการประมวลผลภาพแบบลำดับ



รูปที่ 31 โฟล์วชาร์ตการประมวลผลภาพแบบขนานใช้ทรานสพิวเตอร์ 1 ตัว

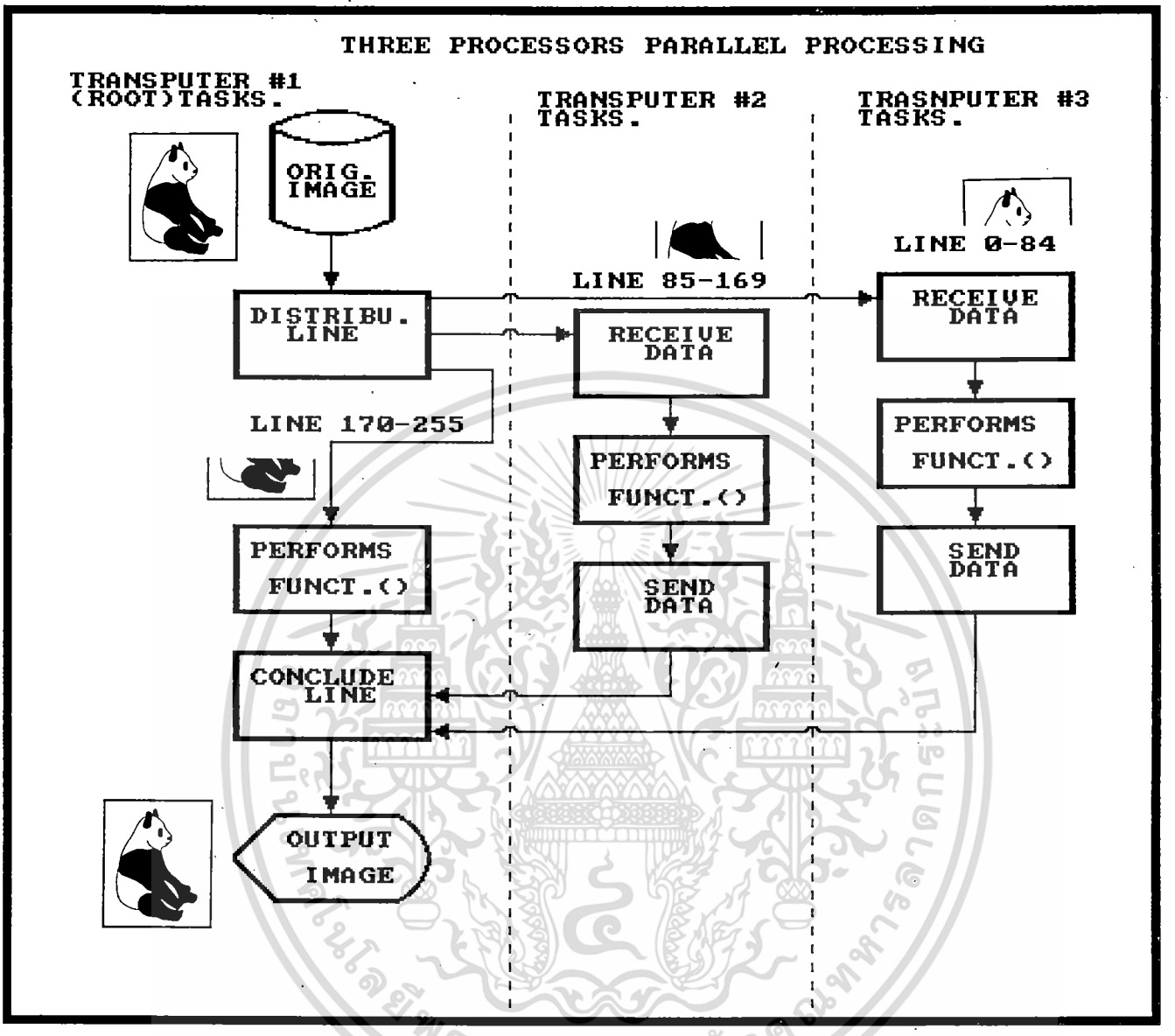
ผลการทดสอบการประมวลผลการกำจัดสัญญาณรบกวนของภาพ คาบเวลาที่ใช้ในการประมวลผลแสดงดังตารางที่ 4 ส่วนประสิทธิภาพในการประมวลผลแสดงดังตารางที่ 5 กราฟเปรียบเทียบความเร็วที่เพิ่มขึ้นแสดงดังรูปที่ 34 และกราฟเปรียบเทียบประสิทธิภาพของอัลกอริทึมแสดงดังรูปที่ 35 ส่วนการประมวลผลในการหาขอบภาพใช้อัลกอริทึมทดสอบเพียง 2 กรณีคืออัลกอริทึมแบบลำดับในข้อ 1 และอัลกอริทึมแบบขนานดังข้อ 4 ได้ผลการทดสอบดังตารางที่ 6 กราฟเปรียบเทียบคาบเวลาในการประมวลผลแสดงดังรูปที่ 36



รูปที่ 32 โฟล์ชาร์ตการประมวลผลภาพแบบขนานใช้ทรานสพิวเตอร์ 2 ตัว

วิธีประมวลผล อัลกอริทึม	ความเร็วในการประมวลผล (SEC.)			
	แบบอนุกรม(1)	แบบขนาน(2)	แบบขนาน(3)	แบบขนาน(4)
เฉลี่ยค่ากลาง	13.34	13.34	6.67	4.54
เฉลี่ยเรขาคณิต	39.07	40.03	19.66	13.26

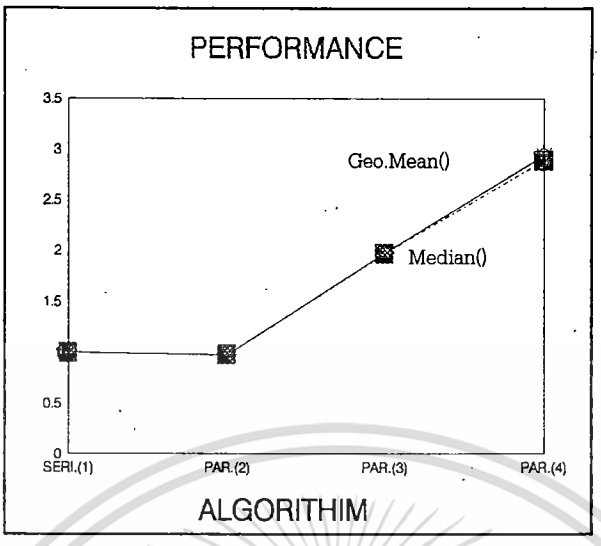
ตารางที่ 4 ผลการเปรียบเทียบคาบเวลาที่ใช้ในการประมวลผลกำจัดสัญญาณรบกวนของภาพ



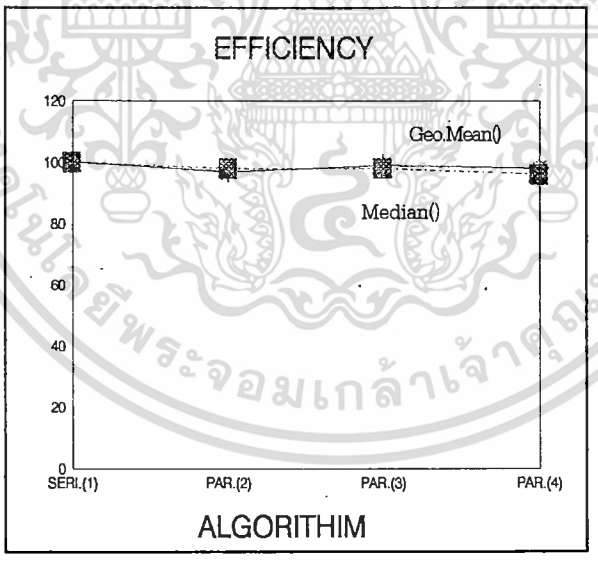
รูปที่ 33 โฟล์ชาร์ตการประมวลผลภาพแบบขนานใช้ทรานสพิวเตอร์ 3 ตัว

วิธีประมวลผล อัลกอริทึม	ความเร็วเทียบกับแบบลำดับ(เท่า) / ประสิทธิภาพ(%)			
	แบบอนุกรม(1)	แบบขนาน(2)	แบบขนาน(3)	แบบขนาน(4)
เฉลี่ยค่ากลาง	1/100	.98/98	1.97/98	2.89/96
เฉลี่ยเรขาคณิต	1/100	.97/97	1.98/99	2.94/98

ตารางที่ 5 ผลการเปรียบเทียบความเร็วเพิ่มขึ้นและประสิทธิภาพของอัลกอริทึมแบบลำดับและขนาน



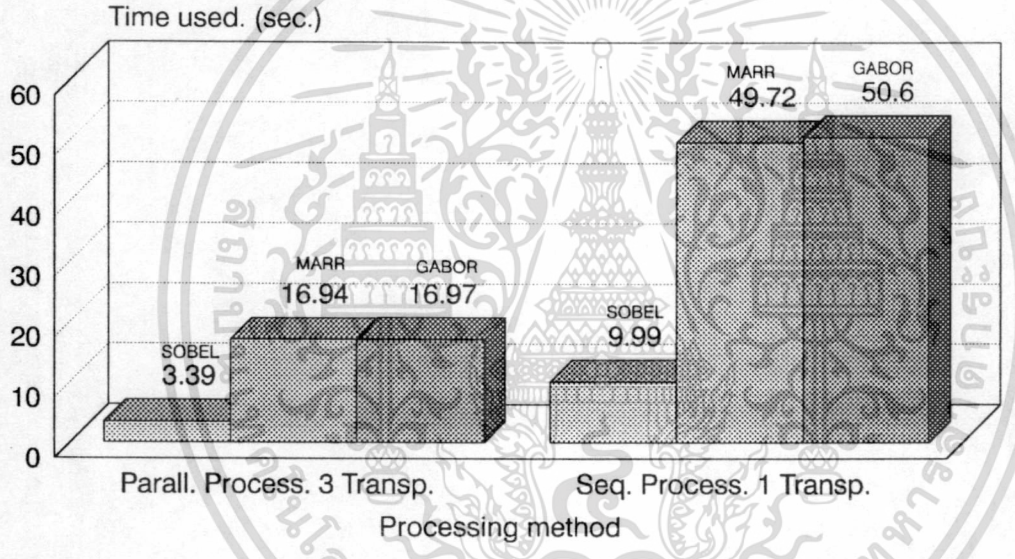
รูปที่ 34 กราฟเปรียบเทียบความเร็วระหว่างอัลกอริทึมแบบลำดับกับแบบขนาน



รูปที่ 35 กราฟเปรียบเทียบประสิทธิภาพระหว่างอัลกอริทึมแบบลำดับกับแบบขนาน

วิธีประมวลผล	ความเร็วในการประมวลผล (sec.)			
	แบบอนุกรม	แบบขนาน	เร็วขึ้น (เท่า)	ประสิทธิภาพ%
โซเบล	9.99	3.39	2.93	98.2
มาร์	49.72	16.94	2.98	97.8
กาบอร์	50.60	16.97	2.94	99.3

ตาราง 6 ผลการเปรียบเทียบเวลาในการประมวลผลการหาขอบภาพ



รูป 36 กราฟแสดงผลการเปรียบเทียบเวลาในการประมวลผลการหาขอบภาพ

7 สรุป

ในส่วนของการประมวลผลภาพ สรุปได้ว่าการกำจัดสัญญาณรบกวนของภาพโดยการเฉลี่ยความสว่างของจุดภาพ โดยวิธีการเฉลี่ยแบบหาค่ากลางหรือวิธีเฉลี่ยแบบเรขาคณิต ทั้งสองวิธีสามารถกำจัดสัญญาณรบกวนได้ดี ส่วนการหาขอบภาพนั้นโดยวิธีของโซเบลสามารถใช้กับภาพที่มีขอบปรากฏชัดเจนหรือภาพที่มีการเปลี่ยน แปลงความเข้มแบบทันทีทันใดได้ดี ส่วนการหาขอบภาพด้วยวิธีมาร์และกาบอร์สามารถใช้กับภาพที่มีการ เปลี่ยนแปลงความเข้มแบบราบเรียบได้ดีกว่าโซเบล แต่เส้นขอบที่ได้หนากว่าต้องนำไปผ่านการทำให้ขอบบางลง (Thinning) ก่อนที่จะใช้งานต่อไป

อนึ่งความเร็วที่ใช้ในการประมวลผล หากใช้วิธีประมวลผลแบบขนานจะทำให้ประมวลผล ได้เร็วกว่าแบบลำดับ ยิ่งมีจำนวนชิปทรานสพิวเตอร์มากก็ยิ่งเร็วมากขึ้นเป็นอัตราส่วนกัน แต่ในทางปฏิบัติยังมีสาเหตุหนึ่งที่ทำให้ การประมวลผลไม่ได้ประสิทธิภาพเต็ม 100 เปอร์เซนต์ คือเวลาที่สูญเสียไปในการติดต่อรับส่งข้อมูลกันระหว่างโปรเซสซึ่งทั้งหมดสรุปได้ดังนี้

7.1 ขีดความสามารถของการประมวลผลแบบขนานบนทรานสพิวเตอร์เน็ทเวิร์ค จะขึ้นอยู่กับจำนวนทรานสพิวเตอร์ที่นำมาต่อในเน็ทเวิร์ค โดยขีดความสามารถของระบบจะเพิ่มเป็นเชิงเส้นเทียบกับจำนวนทราสพิวเตอร์

7.2 ประสิทธิภาพของการประมวลผล ของระบบจะลดลงเพียงเล็กน้อยเมื่อเพิ่มจำ นวนทรานสพิวเตอร์เข้าไป ทั้งนี้ประสิทธิภาพที่ลดลงจะสูญเสียไปกับเวลาที่ใช้ในการติดต่อระหว่างทรานสพิวเตอร์ และเวลาที่ใช้ในการสลับโปรเซสของซีพียูในทรานสพิวเตอร์ตลอดถึงวิธีการแบ่งงานและการจัดอัลกอริทึมของโปรแกรมด้วย

7.3 การย้ายโปรแกรมไปประมวลผลบนทรานสพิวเตอร์เน็ทเวิร์คที่มีการจัด Configuration ต่าง ๆ กัน ไม่มีผลต่อลอจิกการทำงานของโปรแกรม

7.4 วิธีการประมวลผลแบบขนานบนทรานสพิวเตอร์กระทำโดย เริ่มจากเขียนอัลกอริทึมแบบขนานขึ้นมาแล้วนำไปเขียนเป็นโปรแกรมภาษาออคคาม และเขียนโมเดลการติดตั้งให้เข้ากับฮาร์ดแวร์ที่มีจากนั้นนำ โปรแกรมภาษาออคคามไปติดตั้งบนทรานสพิวเตอร์เน็ทเวิร์คเพื่อปฏิบัติงานที่ต้องการต่อไป

อย่างไรก็ตามการประมวลผลแบบขนานไม่สามารถแก้ปัญหาได้ทุกชนิดปัญหา ยังมีปัญหาบางชนิดที่ไม่เหมาะสมกับอัลกอริทึมแบบขนาน งานเหล่านี้คืองานขนาดเล็กๆ ไม่ซับซ้อนใช้อัลกอริทึมแบบลำดับจะเหมาะสมกว่า งานอีกลักษณะหนึ่งที่ไม่เหมาะสมคือ อัลกอริทึมเวียนเกิด (Recursive) เพราะมีปัญหาในการใช้ตัวแปรร่วมและการรอผลการกระทำคำสั่งก่อนหน้าคำสั่งปัจจุบันเสมออย่างไรก็ตามงานประมวลผลภาพถือว่าเป็นงานที่เหมาะกับวิธีประมวลผลแบบขนานมาก เพราะสามารถจัดการโปรแกรมแบบขนานได้ง่าย และในการวิจัยครั้งนี้พบว่าสามารถเพิ่มความเร็วได้เป็นเชิงเส้น และให้ประสิทธิภาพได้ถึงมากกว่า 90 เปอร์เซนต์

8 เอกสารอ้างอิง

[1] กิตติ ไพทูรย์วัฒนกิจ, ไพลิน บุญเดช, การประมวลผลแบบขนานโดยใช้ทรานสพิวเตอร์, วารสารคอมพิวเตอร์ สคพท., ปีที่ 10, ฉบับที่ 103 กันยา-ตุลา 36, หน้า 35-45.

[2] กิตติ ไพทูรย์วัฒนกิจ, ไพลิน บุญเดช, การประมวลผลแบบขนานสำหรับการจำลองแอนนีลลิ่งบนทรานสพิวเตอร์, วารสารคอมพิวเตอร์ สคพท., ปีที่ 10 , ฉบับที่ 107, พฤษภา-มิถุนา 37, หน้า 29-35.

[3] ยืน ภู่วรวรรณ, แฟ้มไมโครโปรเซสเซอร์-ทรานสพิวเตอร์ ตอน 1, วารสารเซมิคอนดัคเตอร์อิเล็กทรอนิกส์, ฉบับที่118, หน้า146-152, 2535

[4] ยืน ภู่วรวรรณ, แฟ้มไมโครโปรเซสเซอร์-ทรานสพิวเตอร์ ตอนจบ, วารสารเซมิคอนดัคเตอร์อิเล็กทรอนิกส์, ฉบับที่119, หน้า120-125, 2535

[5] R. Mehrota, K. R. Namudur and N. Ranganathan, IEEE Pattern Recognition, Vol..25, No. 12, pp. 1479-1494, 1992.

[6] Jie Yao, IEEE Image Processing, Vol. 2, No. 2, pp. 152-159, April 1993.

[7] D. Marr and E. Hildreth, Proc. R. Soc. Lond., B 207, pp. 187-217, 1980.

[8] Anil K. Jan, Fundamentals of Digital Image Processing, U.K. : Prentice-Hall International Ltd., 1989.

[9] Wayne Niblack, An Introduction to Digital Image Processing, U.K. : Prentice-Hall International Ltd., 1986.

[10] INMOS Limited, Transputer Instruction set, Hemel Hempstead, (U.K.): Prentice Hall International Ltd., 1988.

[11] INMOS Limited, Transputer Reference Manual, Hemel Hempstead, (U.K.): Prentice Hall Internatinoal Ltd., 1988.

[12] INMOS Limited, Transputer Technical Note, Hemel Hempstead, (U.K.): Prentice Hall Internatinoal Ltd., 1988.

[13] INMOS Limited, IMS D7205 IBM/NEC PC OCCAM2 Toolset user manual, Bristol, (U.K.): INMOS Ltd., 1991.

[14] Ronald S. Cok, Parallel Programs for the Transputer, Newjersey, (U.S.A.): Prentice-Hall Inc., 1991.

[15] INMOS Limited, OCCAM2 Reference Manual, Hemel Hempstead, (U.K.): Prentice Hall International Ltd., 1988.

[16] Jones G. and M. Goldsmith, Programming in OCCAM2, Hemel Hempstead, (U.K.): Prentice Hall International Ltd., 1988.

[17] Pountain, Dick and David May, A Tutorial Introduction to OCCAM Programming, (U.K.): Blackwell Scientific Publications Professional Books, 1987.