การจัดสรรงานแบบซุปเปอร์ไปป์ไลน์ของการแลกเปลี่ยนแบบออล-ทู-ออล เพอร์ซันนัลไลซ์ที่ดีที่สุดบนเครือข่ายการเชื่อมต่อแบบมัลติสเตจพลัส ที่สามารถแบ่งกลุ่มย่อยได้

Super-pipeline Scheduling of Optimal All-to-All Personalized Exchange on Partitionable Multistage Interconnection Networks+

รสลิน เพตะกร

Roselin Petagon

Super-pipeline Scheduling of Optimal All-to-All Personalized Exchange on Partitionable Multistage Interconnection Networks+

ROSELIN PETAGON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE
DOCTOR OF PHILOSOPHY PROGRAM IN COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2017

KMITL-2017-SC-D-002-001

คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ "การจัดสรรงานแบบซุปเปอร์ไปป์ไลน์ของการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ที่ดีที่สุดบนเครือข่ายการเชื่อมต่อแบบมัลติสเตจพลัสที่สามารถแบ่งกลุ่มย่อยได้"
(SUPER-PIPELINE SCHEDULING OF OPTIMAL ALL-TO-ALL PERSONALIZED EXCHANGE ON PARTITIONABLE MULTISTGE INTERCONNECTION NETWORKS[+])

ชื่อนักศึกษา นางสาวรสลิน เพตะกร
รหัสประจำตัว 54650501
ปริญญา ปรัชญาดุษฎีบัณฑิต (สาขาวิทยาการคอมพิวเตอร์)
ภาควิชา วิทยาการคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์ รศ.ดร.จีรพร วีระพันธุ์

คณะกรรมการสอบวิทยานิพนธ์	ลายมือชื่อ
ผศ.ดร.ศรัณย์ อินทโกสุม ประธานกรรมการ	
ผศ.ดร.วรางคณา กิ้มปาน อาจารย์บัณฑิตประจำ (ในสาขาวิชาที่เกี่ยวข้อง)	
ผศ.ดร.สายชล ใจเย็น อาจารย์บัณฑิตประจำ (ในสาขาวิชาที่เกี่ยวข้อง)	
ดร.เฉลิมศักดิ์ เลิศวงศ์เสถียร ผู้ทรงคุณวุฒิจากภายนอกสถาบันฯ	
รศ.ดร.จีรพร วีระพันธุ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์	

วัน/ เดือน/ ปี ที่สอบ 6 มกราคม พ.ศ. 2560 เวลา 14.00 - 16.00 น.
สถานที่สอบ ณ ห้อง 305 อาคารพระจอมเกล้า

คณะวิทยาศาสตร์รับรองแล้ว

(รองศาสตราจารย์ ดร.ดุษณี ธนะบริพัฒน์)
คณบดีคณะวิทยาศาสตร์
วันที่ 28 **เดือน** ม.ค. **พ.ศ.** 60

หัวข้อวิทยานิพนธ์	การจัดสรรงานแบบซุปเปอร์ไปป์ไลน์ของการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ที่ดีที่สุดบนเครือข่ายการเชื่อมต่อแบบมัลติสเตจพลัสที่สามารถแบ่งกลุ่มย่อยได้
ชื่อนักศึกษา	นางสาวรสลิน เพตะกร
รหัสประจำตัว	54650501
ปริญญา	ปรัชญาดุษฎีบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา	วิทยาการคอมพิวเตอร์
พ.ศ.	2560
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รองศาสตราจารย์ ดร. จีรพร วีระพันธุ์

บทคัดย่อ

ในระบบคอมพิวเตอร์แบบขนาน ฟังก์ชันการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ (All-to-All Personalized Exchange : ATAPE) เป็นฟังก์ชันหนึ่งที่ได้รับความสนใจอย่างมากสำหรับการประมวลผลงานขนาดใหญ่ๆ ทางวิทยาศาสตร์และวิศวกรรมศาสตร์ เพราะเป็นการติดต่อสื่อสารเพื่อแลกเปลี่ยนข้อมูลระหว่างหน่วยประมวลผลที่ดีที่สุดหรือเร็วที่สุด และสามารถนำไปประยุกต์ใช้ได้อย่างกว้างขวางเพื่อให้ได้ผลลัพธ์ในเวลาที่เร็วที่สุดด้วย เช่น การทรานส์โพสเมตริกซ์แบบขนาน การแปลงฟูรีเยร์อย่างเร็วแบบขนาน การกระจายข้อมูลแบบออล-ทู-ออล เป็นต้น โดยในอดีตจนถึงปัจจุบันได้มีการศึกษาค้นคว้ามากมายเกี่ยวกับการติดต่อสื่อสารแบบ ATAPE ทั้งบนเครือข่ายแบบสแตติค (เช่น ไฮเปอร์คิวบ์ ตาข่ายหรือกริด) และบนเครือข่ายแบบไดนามิคโดยเฉพาะเครือข่ายการเชื่อมต่อแบบมัลติสเตจ (Multistage Interconnection Networks : MINs) (อาทิ เบสไลน์ บัตเตอร์ฟลาย) แต่การติดต่อสื่อสารแบบ ATAPE บนเครือข่ายแบบ MINs ยังไม่เสถียรเพราะขั้นตอนวิธีส่วนใหญ่ขึ้นกับแบบของการเชื่อมต่อภายในสวิทช์สำหรับเครือข่าย MIN แต่ละชนิด ทำให้การฝัง ATAPE ดังกล่าวลงบนชิปทำได้ยาก ดังนั้นวิทยานิพนธ์ฉบับนี้ จึงนำเสนอฟังก์ชัน ATAPE แบบใหม่สองชนิด คือฟังก์ชันแบบคงที่และฟังก์ชันแบบปรับเปลี่ยนได้ โดยทั้งสองฟังก์ชันสามารถฝังลงบนชิปของเครือข่ายแบบ $MINs^+$ ที่มีการปรับพอร์ตให้เป็นเครือข่าย MINs ที่สมบูรณ์ และบนครอสส์บาร์ของ $MINs^+$ (Crossbar of $MINs^+$: $CMINs^+$) ที่เร็วขึ้น และในการศึกษาวิจัยต่อยอดในวิทยานิพนธ์จะเน้นที่ 1. การออกแบบระบบเครือข่าย $MINs^+$ ใหม่ให้มีประสิทธิภาพมากยิ่งขึ้น คือสามารถแบ่งเป็นกลุ่มย่อยๆ ได้ (x x x Crossbar of Partitionable MINs : $PMINs^+$ ขนาด N) เพื่อประมวลผลงานหลายๆ งาน (ขนาด $N'' = N/2^t$) ได้พร้อมๆ กัน โดยการใช้ I/O สวิทช์ควบคุมแบบครอสส์ (I/O Cross-Control Switches) และ 2. เสนอระบบย่อย $MINs^+$ แบบสมบูรณ์ที่ใช้สวิทช์ขนาด dxd พร้อมทั้ง 3. เสนอการฝังฟังก์ชัน ATAPE ที่ดีที่สุดแบบใหม่ลงบนชิปชื่อว่า ขั้นตอนวิธีแบบ “วดี” (VA-DE : Valuable ATAPE with Dynamic Embedding) เพื่อให้สามารถประยุกต์ใช้กับทุกระบบย่อยได้อย่างเต็มประสิทธิภาพ ด้วยหลักการของซุปเปอร์ไปป์ไลน์ประกอบกับการจัดสรรงานที่ดีที่สุด (ที่จัดสรรได้ครบทั้ง 3 ปัจจัยสำคัญ) คือสามารถจัดลำดับงานย่อยให้เหมาะกับกลุ่มย่อยในเวลาที่เหมาะสม (Right Task, Right Section, Right Time) ในการวัดประสิทธิภาพด้วยการศึกษาทดลองบนระบบจำลอง $PMINs^+$ (ขนาด $N \le 32768$ และ $x \le 16$) พบว่า ค่าอัตราเร็วเพิ่มขึ้น (Speedup) และปริมาณงานที่ประมวลผลได้ (Throughput) มีค่าเพิ่มขึ้นอย่างมีนัยสำคัญ เมื่อเปรียบเทียบกับค่าดังกล่าวบน $CMINs^+$ ที่ไม่สามารถแบ่งเป็นกลุ่มย่อยได้

คำสำคัญ : การแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ที่ดีที่สุด เครือข่ายการเชื่อมต่อแบบมัลติสเตจพลัสที่สามารถแบ่งกลุ่มย่อยได้ I/O สวิทช์ควบคุมแบบครอสส์สำหรับฟังก์ชัน ATAPE ที่ปรับแบ่งบนกลุ่มย่อยได้ การก้าวกระโดดสองต่อของตัวควบคุมในการจัดสรรงานที่ดีที่สุดบน $PMINs^+$ แบบซุปเปอร์ไปป์ไลน์ที่สามารถจัดสรรลำดับงานให้เหมาะกับกลุ่มในเวลาที่เหมาะสม

Thesis Title	Super-pipeline Scheduling of Optimal All-to-All Personalized Exchange on Partitionable Multistage Interconnection Networks^{+}
Student Name	Miss Roselin Petagon
Student ID	54650501
Degree	Doctor of Philosophy (Computer Science)
Department	Computer Science
Year	2017
Thesis Advisor	Associate Professor Dr. Jeeraporn Werapun

Abstract

In parallel computer systems, the optimal ATAPE (all-to-all personalized exchange) is a popular processing for science and engineering since it is a key solution to achieve the optimal communication in parallel matrix transposition, parallel fast-Fourier-transformation, all-to-all broadcasting, etc. The ATAPE has been extensively studied on static networks (i.e., hypercube, mesh, etc.) and dynamic MINs (multistage interconnection networks i.e., baseline, butterfly, etc.). However on the dynamic networks, existing ATAPE strategies rely on switch patterns on each type of the MINs and cannot be embedded on-chip easily. Therefore, this thesis proposes the static and dynamic ATAPE functions in the first part to be embedded on-chip for all types of the complete MINs^{+} with port-adjusting. The second part of this thesis focuses on 1. the new designing of the $x \times x$ crossbar of partitionable MINs^{+} (PMINs^{+}) by using the I/O cross-control switches, 2. the fully self-routable MINs^{+} with $d \times d$ switches for PMINs^{+}, and 3. The optimal VA-DE (valuable ATAPE with dynamic embedding) algorithm, an efficient partitionable ATAPE, with the triple-right scheduling (right-task, right-section, right-time) for multiple ATAPE-tasks of flexible sizes $N''=N/2^{t}$ (where $t \leq \log_2 x$) and the efficient subsystem-utilization. In experiments, efficient speedup and outperformed throughput were strongly confirmed on various simulated PMINs^{+} ($N \leq 32768$ and $x \leq 16$), compared to the regular CMINs^{+} (the $x \times x$ crossbar of MINs^{+}).

Keywords : Optimal ATAPE (All-to-All Personalized Exchange); Partitionable multistage interconnection networks^{+} (PMINs^{+}); I/O cross-control switches for ATAPE partitioning; Double jumping of controls for the triple-right scheduling (right task, right section, right time) on PMINs^{+}.

กิตติกรรมประกาศ

วิทยานิพนธ์นี้มีอาจจะสำเร็จลุล่วงไปได้ด้วยดี หากมิได้รับคำแนะนำ คำชี้แจง ความรู้ และความเอาใจใส่เป็นอย่างดียิ่งจาก รศ.ดร.จีรพร วีระพันธุ์ ผู้เป็นอาจารย์ที่ปรึกษา ซึ่งท่านได้สละเวลาให้กับข้าพเจ้าอย่างเต็มที่ จึงใคร่ขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ ผศ.ดร.ศรัณย์ อินทโกสุม ผศ.ดร. วรางคณา กิ้มปาน ดร. สายชล ใจเย็น และ ดร.เฉลิมศักดิ์ เลิศวงศ์เสถียร คณะกรรมการสอบวิทยานิพนธ์ ที่กรุณาให้คำแนะนำตลอดจนข้อชี้แนะจนในที่สุดทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้

ขอขอบพระคุณมหาวิทยาลัยราชภัฏเชียงใหม่ ที่สนันสนุนให้ได้เรียนในระดับที่ได้ตั้งใจ อีกทั้งยังได้ดูแลเรื่องค่าใช้จ่ายต่างๆระหว่างศึกษาเป็นอย่างดีอีกด้วย

ขอขอบพระคุณสำนักงานคณะกรรมการการอุดมศึกษา ที่สนับสนุนทุนการศึกษา

ขอขอบพระคุณบิดา ที่สนันสนุนให้ได้เรียนในระดับที่ได้ตั้งใจ และเป็นกำลังใจในระหว่างการศึกษาเป็นอย่างดียิ่ง

ขอขอบพระคุณสามี และลูกสาวสำหรับกำลังใจ แรงสนับสนุน และส่งเสริมในทุกๆ ด้าน

ขอขอบพระคุณพี่ๆ เพื่อนๆ น้องๆ ที่ได้เรียน ได้ศึกษางานวิจัยร่วมกัน ช่วยเป็นกำลังใจส่งเสริม สนับสนุนในทุกๆ ด้าน

ขอขอบพระคุณบรรณาธิการวารสารวิชาการระดับนานาชาติ Parallel and Distributed Computing (JPDC) ผู้อ่านงานวิจัยทุกท่าน และผู้ที่เกี่ยวข้องที่ได้ให้ข้อคิดเห็น วิพากษ์วิจารณ์ เปิดโอกาสและให้การยอมรับแนวคิดให้ได้รับการตีพิมพ์ในวารสาร

สำหรับคุณงามความดีและประโยชน์อันใดที่เกิดขึ้นจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดา มารดา อาจารย์ทุกท่านซึ่งเป็นที่เคารพรักยิ่ง ตลอดจนญาติพี่น้อง และเพื่อนๆทุกคน

รสลิน เพตะกร

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

คำย่อ/สัญลักษณ์

ATAPE	All-to-All Personalized Exchange
*p*ATAPE	Partitionable All-to-All Personalized Exchange
VA-DE	Valuable ATAPE with Dynamic Embedding
LS	Latin-Square
ISC	Inter Stage Connection
GSEN	Generalized Shuffle Exchange Network
MIN	Multistage Interconnection Network
MIN^+	Multistage Interconnection Network Plus
$CMINs^+$	a Crossbar of Multistage Interconnection Networks Plus
$PMINs^+$	a Crossbar of Partitionable Multistage Interconnection Networks Plus
XOR	Exclusive OR Operation
MSb	Most Significant bit
LSb	Least Significant bit

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ (All-to-All Personalized Exchange : ATAPE) เป็นการติดต่อสื่อสารระหว่างหน่วยประมวลผลชนิดหนึ่งในระบบคอมพิวเตอร์แบบขนาน ที่ทุกๆ หน่วยประมวลผล (N) สามารถส่งข้อความเฉพาะไปยังทุกๆ หน่วยประมวลผลอื่นๆ ได้พร้อมๆ กัน (Parallel Point-to-Point Communication) เป็นจำนวน N รอบได้อย่างมีประสิทธิภาพ ด้วยความซับซ้อนด้านเวลาดีที่สุดหรือเร็วที่สุดเป็น $O(N+\log_2 N)$ ดังนั้นแอพพลิเคชันที่สามารถประยุกต์ใช้ ATAPE ในการติดต่อสื่อสารก็จะมีความซับซ้อนด้านเวลาดีที่สุดและประมวลผลได้เร็วที่สุดด้วย เช่น การทรานส์โพสเมตริกซ์แบบขนาน (Parallel Matrix Transposition) การแปลงฟูรีเยร์อย่างเร็วแบบขนาน (Parallel Fast Fourier Transformation) การกระจายข้อมูลแบบออล-ทู-ออล (All-to-All Broadcasting) เป็นต้น ตัวอย่างเช่น ความซับซ้อนด้านเวลาของการกระจายข้อมูลแบบออล-ทู-ออล คือ $O(N \log_2 N)$ ในกรณีที่ไม่ได้ประยุกต์ใช้ ATAPE แต่เมื่อดำเนินการผ่านการติดต่อสื่อสารแบบ ATAPE จะได้ความซับซ้อนด้านเวลาดีที่สุดคือ $O(N+\log_2 N)$

จากอดีตจนถึงปัจจุบันได้มีการศึกษาค้นคว้าเกี่ยวกับการติดต่อสื่อสารแบบ ATAPE เป็นจำนวนมากทั้งบนเครือข่ายแบบสแตติค (Static Interconnection Networks) เช่น เครือข่ายแบบไฮเปอร์คิวบ์ (Hypercube Network) เครือข่ายแบบตาข่ายหรือกริด (Mesh Network) เครือข่ายแบบทอรัส (Torus Network) เป็นต้น [11-14] และในปัจจุบันการศึกษา ATAPE ส่วนใหญ่มุ่งไปที่เครือข่ายแบบไดนามิค (Dynamic Interconnection Networks) โดยเน้นที่เครือข่ายการเชื่อมต่อแบบมัลติสเตจ (Multistage Interconnection Networks หรือ MINs) [3, 5, 6, 17] เพราะสามารถสร้างด้วยราคาที่ไม่แพงมากนักในเครือข่ายขนาดใหญ่ๆ เช่น เครือข่ายแบบโอเมก้า (Omega Network) เครือข่ายแบบฟลิป (Flip Network) เครือข่ายแบบคิวบ์ (Cube Network) เครือข่ายแบบเบสไลน์ (Baseline Network) เครือข่ายแบบบัตเตอร์ฟลาย (Butterfly Network) เป็นต้น

ในยุคแรกๆ การติดต่อสื่อสาร ATAPE บนเครือข่ายแบบสแตติคส่วนใหญ่ สามารถใช้ฟังก์ชัน XOR (Exclusive OR Operation) ในการคำนวณหาปลายทาง $D = S \oplus C$ [12] ได้อย่างมีประสิทธิภาพโดยเฉพาะบนเครือข่ายแบบไฮเปอร์คิวบ์ สำหรับทุกค่าของต้นทาง $S = 0, 1, 2, \ldots, N-1$ และค่าควบคุมในแต่ละรอบ $C = 0, 1, 2, \ldots, N-1$ (เป็นจำนวน N รอบ) แต่การติดต่อสื่อสารระหว่างหน่วยประมวลผลแต่ละคู่ของไฮเปอร์คิวบ์อาจใช้เวลาไม่เท่ากัน (Inconsistent Latency) เพราะหน่วยประมวลผลบางคู่ที่อยู่ติดกัน (Adjacent Nodes) จะใช้เวลาน้อย แต่บางคู่ที่ไม่ได้อยู่ติดกันจะใช้เวลามากกว่า ต่อมาในปี ค.ศ. 2013 ฟังก์ชัน ATAPE แบบลำดับชั้น $h^D = (h^S + h^P) \bmod N^l$ [11] ถูกนำเสนอสำหรับเครือข่ายดราก้อนฟลาย (Dragonfly Network) ซึ่งเป็นเครือข่ายสแตติคแบบลำดับชั้น (Static Hierarchical Network) ที่ประกอบด้วยสวิทช์ N^l ชั้น และเพิ่มสวิทช์ในแต่ละชั้นเพื่อให้ทุกคู่สามารถติดต่อสื่อสารด้วยเวลาเท่ากัน แต่การใช้สวิทช์ดังกล่าวจะทำให้การสร้างเครือข่ายมีค่าใช้จ่ายสูงโดยเฉพาะเมื่อเครือข่ายมีขนาดใหญ่ๆ

ในปัจจุบันการติดต่อสื่อสารแบบ ATAPE ที่ดีที่สุด ถูกนำเสนอบนเครือข่ายแบบ MINs [1, 2, 4, 7, 8, 15] ได้อย่างมีประสิทธิภาพ แต่การติดต่อสื่อสารแบบ ATAPE บนเครือข่ายแบบ MINs [3, 5, 6, 16, 17] ยังไม่เสถียร เพราะขั้นตอนวิธีส่วนใหญ่ขึ้นกับแบบของการเชื่อมต่อภายในสวิตช์บนสเตจ

(Switch-Stage Pattern) สำหรับเครือข่าย MIN แต่ละชนิด โดยเฉพาะเครือข่ายแบบ MINs ที่เป็นรีเคอร์ซีฟ (Recursive MINs) เช่น เครือข่ายแบบเบสไลน์ (Baseline Network) เครือข่ายแบบบัตเตอร์ฟลาย (Butterfly Network) เครือข่ายแบบบันยาน (Banyan Network) เป็นต้น ทำให้การฝัง ATAPE ลงบนชิปด้วยวิธีต่างๆ ที่นำเสนอไว้แล้วในอดีตทำได้ยากมาก

ดังนั้นเพื่อแก้ปัญหาดังกล่าว (ความยุ่งยากในการฝังฟังก์ชัน ATAPE ลงบนชิปของเครือข่ายแบบ MINs) วิทยานิพนธ์ฉบับนี้ จึงนำเสนอผลการศึกษาวิจัยเกี่ยวกับการฝังฟังก์ชัน ATAPE ลงบนชิปเพื่อการประมวลผลที่เร็วยิ่งขึ้นสำหรับระบบคอมพิวเตอร์แบบขนานในอนาคต (Next Generation of Parallel Computer Systems) โดยแบ่งการศึกษาออกเป็น 2 ส่วนใหญ่ๆ คือ

ส่วนแรกนำเสนอฟังก์ชัน ATAPE แบบใหม่สองชนิด (นำเสนอในบทที่ 3) คือฟังก์ชันแบบคงที่ (Static ATAPE) และฟังก์ชันแบบปรับเปลี่ยนได้ (Dynamic ATAPE) โดยทั้งสองฟังก์ชันสามารถฝังลงบนชิป (On-chip Embedding) ของเครือข่ายแบบ MINs^{+} ที่มีการปรับพอร์ตให้เป็นเครือข่าย MINs ที่สมบูรณ์ ได้แก่ เครือข่ายแบบเบสไลน์พลัส (Baseline^{+} Network) เครือข่ายแบบบัตเตอร์ฟลายพลัส (Butterfly^{+} Network) เครือข่ายแบบบันยานพลัส (Banyan^{+} Network)) เป็นต้น

ส่วนที่สองเป็นการศึกษาวิจัยต่อยอดจากส่วนแรก (นำเสนอในบทที่ 4) โดยจะเน้นทั้งการออกแบบปรับปรุงเครือข่าย MINs^{+} ใหม่ และการฝัง ATAPE ที่มีประสิทธิภาพยิ่งขึ้นบนเครือข่ายดังนี้

1. การออกแบบระบบเครือข่าย MINs^{+} ใหม่ให้มีประสิทธิภาพมากยิ่งขึ้น คือสามารถแบ่งเป็นกลุ่มย่อยๆ ได้ (an $x \times x$ Crossbar of Partitionable MINs หรือ PMINs^{+} (ขนาด N) เพื่อประมวลผลงานหลายๆ งาน (ขนาด $N" = N/2^{t}$) ได้พร้อมๆ กัน โดยการใช้ I/O สวิทช์ควบคุมแบบครอสส์ (I/O Cross-Control Switches)

2. เสนอระบบย่อย MINs^{+} แบบสมบูรณ์ที่ใช้สวิทช์ขนาด dxd (Complete Self-Routable d-nary MINs^{+}) เนื่องจากการศึกษาวิจัยในอดีตและการศึกษาในส่วนแรกของวิทยานิพนธ์ เน้นการแก้ปัญหา ATAPE บนเครือข่ายแบบ MINs ที่ใช้สวิทช์ขนาด 2x2 เป็นหลัก และ

3. เสนอการฝังฟังก์ชัน ATAPE ที่ดีที่สุดแบบใหม่ลงบนชิป ชื่อว่าขั้นตอนวิธี "วดี" (VA-DE : Valuable ATAPE with Dynamic Embedding) เพื่อให้สามารถประยุกต์ใช้กับทุกระบบย่อยได้อย่างเต็มประสิทธิภาพ (Optimal Time and Space) ด้วยหลักการของซุปเปอร์ไปป์ไลน์ (Super-pipelining Technique) ประกอบกับการจัดสรรงานที่ดีที่สุดที่จัดสรรได้ครบทั้ง 3 ปัจจัยสำคัญ (Triple-Right Scheduling) คือสามารถจัดลำดับงานย่อยให้เหมาะกับกลุ่มย่อยในเวลาที่เหมาะสม (Right task, Right section, Right time) หรือจัดสรรงานให้ ถูกคน (Right man or Right task) ถูกที่ (Right place or Right section) ถูกเวลา (Right time)

ฟังก์ชัน ATAPE แบบใหม่ (VA-DE) ที่เสนอในส่วนนี้ จะเน้นที่ความสามารถก้าวกระโดดสองต่อสำหรับการประมวลผลแต่ละชิ้นงานบนเครือข่ายแบบ PMINs^{+} คือสำหรับกลุ่มย่อย y (ที่ค่าของ y = 0, 1, 2, ..., x-1) จะประยุกต์ใช้ฟังก์ชันคำนวณปลายทาง $D^{y} = S^{y} \oplus C_{l}^{y}$ จากต้นทาง S^{y} = 0, 1, 2, ..., N-1 ที่ใช้ค่าควบคุมหลัก $C_{l}^{y} = (yN/x + l) \bmod N$) เพื่อให้สามารถก้าวกระโดดไปยังกลุ่มย่อย y ถัดไป) และค่าควบคุมรอง $c^{z} = zN"/x$ (สำหรับก้าวกระโดดไปยังชิ้นงานย่อย z ถัดไปที่อยู่ภายในกลุ่มเดียวกัน) เมื่อ l = 0, 1, 2, ..., N/x และ z = 0, 1, 2, ..., 2^{t}-1 และสุดท้ายในการวัดประสิทธิภาพด้วยการศึกษาทดลองบนระบบจำลอง PMINs^{+} (ขนาด $N \le 32768$ และ $x \le 16$) พบว่า ค่าอัตราเร็วเพิ่มขึ้น (Speedup) และปริมาณงานที่ประมวลผลได้ (Throughput) มีค่าเพิ่มขึ้นอย่างมีนัยสำคัญ เมื่อเปรียบเทียบกับค่าดังกล่าวบนครอสส์บาร์ของ MINs^{+} (Crossbar of MINs^{+}: CMINs^{+}) ที่ไม่สามารถแบ่งเป็นกลุ่มย่อยได้

1.2 วัตถุประสงค์ของงานวิจัย

วิทยานิพนธ์นี้ มีวัตถุประสงค์เพื่อศึกษาและออกแบบฟังก์ชัน ATAPE ให้สามารถฝังลงบนชิปของเครือข่ายแบบ MINs$^+$ และเครือข่ายแบบ PMIN$^+$ ที่สามารถแบ่งเป็นกลุ่มย่อยได้ ดังนี้

1) เพื่อสามารถฝังฟังก์ชัน ATAPE ลงบนชิปของเครือข่ายแบบ MINs$^+$ ที่มีการปรับพอร์ตให้เป็นเครือข่ายแบบ MINs ที่สมบูรณ์ โดยฟังก์ชันที่ฝังจะประกอบด้วยฟังก์ชันแบบคงที่ ($D = S$ XOR (C + *order*) mod N) และฟังก์ชันแบบปรับเปลี่ยนได้ ($D = \rho[(S + C + order) \bmod N]$)

2) เพื่อออกแบบระบบเครือข่ายแบบ PMINs$^+$ (Partitionable MINs$^+$) ที่สามารถแบ่งเป็นกลุ่มย่อยๆ ได้ (Partitionable MINs$^+$: PMINs$^+$) เพื่อให้สามารถทำการประมวลผลงานหลายๆ งาน ได้พร้อมๆ กัน โดยการใช้ I/O สวิทช์ควบคุมแบบครอสส์ (I/O Cross-Control Switches)

3) เพื่อสามารถฝังฟังก์ชัน *p*ATAPE แบบใหม่ลงบนชิปของเครือข่ายแบบ PMINs$^+$ เพื่อให้สามารถนำไปประยุกต์ใช้กับทุกระบบย่อยได้อย่างเต็มประสิทธิภาพด้วยหลักการของซุปเปอร์ไปป์ไลน์

1.3 ขอบเขตของงานวิจัย

1) ศึกษาฟังก์ชัน ATAPE แบบฝังบนชิปของเครือข่ายแบบ MINs$^+$ ได้แก่ เครือข่ายแบบเบสไลน์พลัส (Baseline$^+$ Network) เครือข่ายแบบบัตเตอร์ฟลายพลัส (Butterfly$^+$ Network) เครือข่ายแบบบันยานพลัส (Banyan$^+$ Network) เป็นต้น และบนเครือข่ายแบบ CMINs$^+$

2) ศึกษาการแบ่งเป็นกลุ่มย่อยๆ ของระบบเครือขาย PMINs$^+$ ใหม่ ที่ใช้สวิทช์ขนาด *dxd* เพื่อให้สามารถทำการประมวลผลงานหลายๆ งาน ได้พร้อมๆ กัน

3) ศึกษาการฝังฟังก์ชัน ATAPE แบบใหม่ลงบนชิปของเครือข่าย PMINs$^+$ เพื่อให้สามารถนำไปประยุกต์ใช้กับทุกระบบย่อยได้อย่างเต็มประสิทธิภาพด้วยหลักการของซุปเปอร์ไปป์ไลน์

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้ฟังก์ชัน ATAPE แบบฝังบนชิปของเครือข่ายแบบ MINs$^+$ ที่สมบูรณ์ และบนเครือข่ายแบบ CMINs$^+$ โดยสามารถประมวลผลได้เร็วขึ้นพร้อมทั้งสามารถจัดเส้นทางจากต้นทางไปยังปลายทางได้แบบสมบูรณ์

2) ได้ระบบเครือข่าย PMINs$^+$ ใหม่ที่มีประสิทธิภาพยิ่งขึ้น คือรองรับการใช้สวิทช์ขนาด *dxd* และสามารถแบ่งเป็นกลุ่มย่อย เพื่อให้สามารถทำการประมวลผลงานหลายๆ งาน ได้พร้อมๆ กัน

3) ได้ฟังก์ชัน ATAPE แบบใหม่ที่ฝังลงบนชิป ชื่อว่าขั้นตอนวิธี “วดี” (VA-DE : Valuable ATAPE with Dynamic Embedding) เพื่อให้สามารถนำไปประยุกต์ใช้กับทุกระบบย่อยบนเครือข่าย PMINs$^+$ ได้อย่างเต็มประสิทธิภาพด้วยหลักการของซุปเปอร์ไปป์ไลน์ ที่ซึ่งจะสามารถจัดสรรงานที่ดีที่สุด ที่จัดสรรได้ครบทั้ง 3 ปัจจัยสำคัญ (Right Task, Right Section, Right Time) คือสามารถจัดลำดับงานย่อยให้เหมาะกับกลุ่มย่อยในเวลาที่เหมาะสม

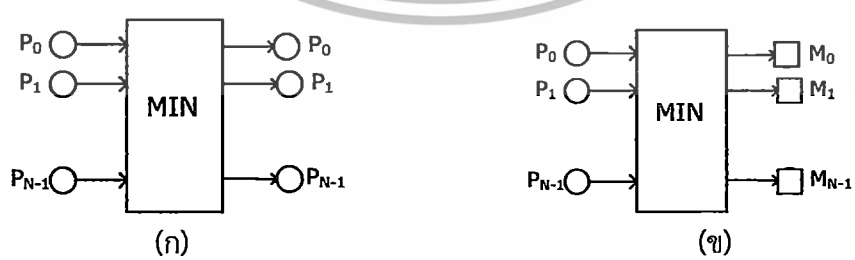
4) ได้ต้นแบบของระบบคอมพิวเตอร์แบบขนานในอนาคต ที่ประยุกต์ใช้เครือข่ายแบบ PMINs$^+$ พร้อมฟังก์ชัน ATAPE ที่ยืดหยุ่นที่สามารถฝังลงบนชิป เพื่อการติดต่อสื่อสารแบบ ATAPE ที่เร็วที่สุด

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

เนื้อหาในบทนี้จะกล่าวถึงทฤษฎีที่เกี่ยวข้องกับชนิดของเครือข่ายการเชื่อมต่อสำหรับระบบคอมพิวเตอร์แบบขนานซึ่งสามารถแบ่งออกเป็น 2 ชนิดใหญ่ๆ คือ เครือข่ายการเชื่อมต่อแบบสแตติค และเครือข่ายการเชื่อมต่อแบบไดนามิค โดยที่เครือข่ายแบบสแตติคเป็นเครือข่ายที่มีการเชื่อมต่อโดยตรงระหว่างหน่วยประมวลผลซึ่งจะมีความล่าช้าในการติดต่อสื่อสาร (Communication Latency) ระหว่างแต่ละคู่ของหน่วยประมวลผลไม่เท่ากัน เช่น เครือข่ายแบบตาข่าย (Meshes) ไฮเปอร์คิวบ์ (Hypercubes) และ ทอรัส (Torus) แต่สำหรับเครือข่ายการเชื่อมต่อแบบไดนามิคเป็นรูปแบบเครือข่ายที่มีความยืดหยุ่นมากกว่าพร้อมทั้งสามารถปรับเปลี่ยนรูปแบบการติดต่อสื่อสารได้หลายรูปแบบ ซึ่งสามารถแบ่งเป็นประเภทย่อยๆ ได้ 3 กลุ่มตามความซับซ้อนของฟังก์ชันการเชื่อมต่อและสวิทช์ เช่น เครือข่ายที่ใช้เส้นทางแบบดิจิทัล (Digital Bus Networks) เครือข่ายการเชื่อมต่อแบบมัลติสเตจ (Multistage Interconnection Networks: MINs) ที่สามารถจะกำหนดการเชื่อมต่อใหม่ได้โดยการปรับการเชื่อมต่อภายในของสวิทช์ เช่น เครือข่ายแบบโอเมก้า (Omega Network) เครือข่ายแบบฟลิบ (Flip Network) เครือข่ายแบบคิวบ์ (Cube Network) เป็นต้น และเครือข่ายที่ใช้สวิทช์แบบครอสส์บาร์ (Crossbar Switch Networks) จะใช้สวิทช์ที่ซับซ้อนและยืดหยุ่นมากขึ้นแต่การสร้างเครือข่ายชนิดนี้มีข้อจำกัดที่ต้นทุนสูงต่อหน่วยของสวิทช์ทำให้โดยรวมของระบบเครือข่ายแบบขนานชนิดนี้สูงตามไปด้วยแต่มีข้อดีคือมีความรวดเร็วในการติดต่อสื่อสารสูงมาก จากนั้นจะกล่าวถึงงานวิจัยที่เกี่ยวข้องกับการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ (All-to-All Personalized Exchange : ATAPE) บนเครือข่ายการเชื่อมต่อแบบมัลติสเตจ

2.1 ระบบเครือข่ายการเชื่อมต่อแบบมัลติสเตจ

รูปแบบเครือข่ายที่มีความยืดหยุ่น เช่น เครือข่ายการเชื่อมต่อแบบมัลติสเตจสามารถประยุกต์ใช้ได้เหมาะสมทั้งในคอมพิวเตอร์แบบเอ็มไอเอ็มดี (Multiple Instruction and Multiple Data : MIMD) และ เอสไอเอ็มดี (Single Instruction and Multiple Data : SIMD) ระบบเครือข่ายแบบนี้จะมีการใช้ในการเชื่อมต่อระหว่างหน่วยประมวลผลกับหน่วยประมวลผล และ หน่วยประมวลผลกับหน่วยความจำดังแสดงในรูปที่ 2.1



รูปที่ 2.1 ระบบเครือข่ายการเชื่อมต่อแบบมัลติสเตจ (ก) การเชื่อมต่อระหว่างหน่วยประมวลผลกับหน่วยประมวลผล (ข) การเชื่อมต่อระหว่างหน่วยประมวลผลกับหน่วยความจำ

ตารางที่ 2.1 คุณสมบัติพื้นฐานของมัลติสเตจ

ขนาดของสวิทช์ (Switch Sizes)	$a = b$ นิยมใช้
จำนวนโหนด (Processors)	N = จำนวนต้นทาง (Source) และ ปลายทาง (Destination)
จำนวนสวิทช์ต่อสเตจ	$s = N/b$
จำนวนสเตจ (Stages)	$n = \log_b N$

โครงสร้างโดยทั่วไปของเครือข่ายการเชื่อมต่อแบบมัลติสเตจ ประกอบด้วยองค์ประกอบต่างๆ ดังต่อไปนี้ คือ

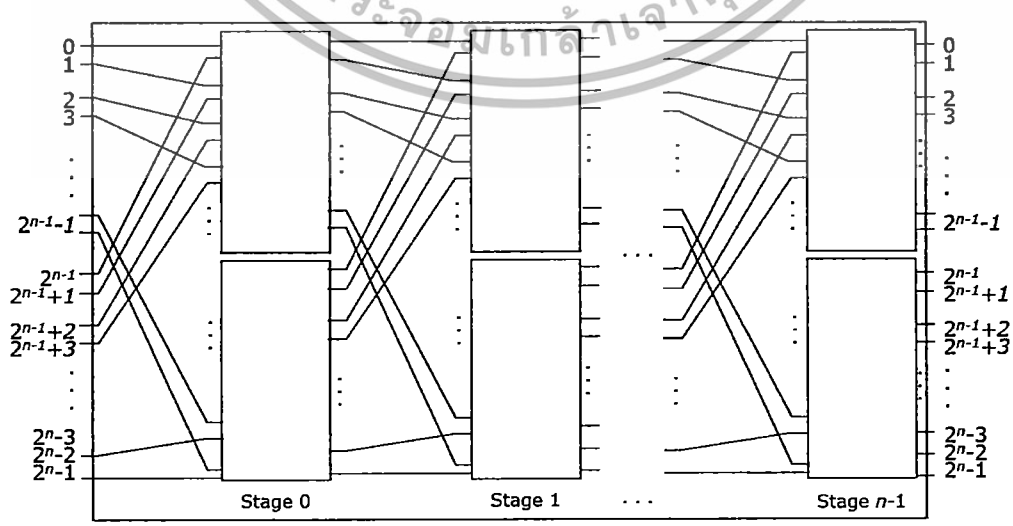
2.1.1 สวิทช์

โมดูลของสวิทช์ (Switch-Box Module) ขนาด $a \times b$ ซึ่งมี a เป็นส่วนนำข้อมูลเข้า และ b เป็นส่วนนำข้อมูลออก โดยทั่วไปนิยมใช้ขนาดของสวิทช์เท่ากันคือ $a = b = d = 2^k$ เมื่อ $k \geq 1$ เช่น ขนาด 2x2 4x4 และ 8x8 เป็นต้น โดยการเชื่อมต่อภายในสวิทช์สามารถปรับเปลี่ยนได้หลายแบบ เช่น การปรับแบบสับเปลี่ยนหรือไขว้ (Cross : X) คือการปรับช่องทางภายในของสวิทช์ประกอบด้วยช่องทาง 0-1 และช่องทาง 1-0 และการปรับแบบตรง (Straight : —) คือการปรับช่องทางภายในของสวิทช์ ประกอบด้วยช่องทาง 0-0 และช่องทาง 1-1 ดังแสดงในรูปที่ 2.2 และอื่นๆ โดยกำหนดให้มีคุณสมบัติพื้นฐานของมัลติสเตจ ดังตารางที่ 2.1

รูปที่ 2.2 แสดงลักษณะการเชื่อมต่อภายในสวิทช์ขนาด 2X2 แบบ 1-1

2.1.2 สเตจ

สมมุติให้ ขนาดของสวิทช์ (Switch Size) = a (= b) ถ้าขนาดของส่วนนำข้อมูลเข้า = $N = 2^n$ ดังนั้น จำนวนของสเตจ (Stages) ขนาด $n = \log_b N$ สเตจ จำนวนของสวิทช์ต่อสเตจขนาด = N/b ยกตัวอย่างดังแสดงในรูปที่ 2.3

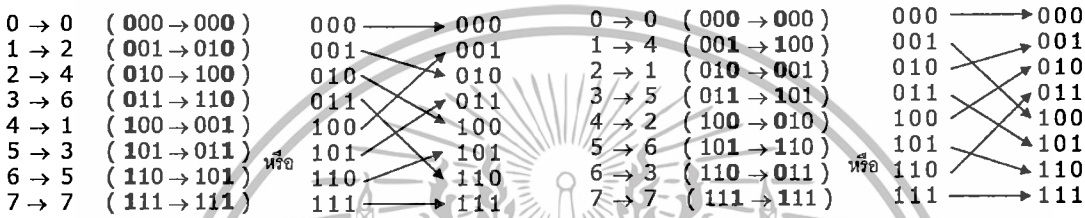


รูปที่ 2.3 แสดงตัวอย่างการเชื่อมต่อของสวิทช์ขนาด 2X2 ($N = 2^n$)

2.1.3 รูปแบบการเชื่อมต่อระหว่างสเตจ

แบบการเชื่อมต่อระหว่างสเตจ (Inter Stage Connection: ISC) บนเครือข่าย MIN เป็นรูปแบบการเชื่อมต่อแบบใดแบบหนึ่งที่ต้องถูกกำหนดไว้ก่อน เพื่อให้สามารถแบ่งเครือข่าย MIN เป็นรูปแบบต่างๆ ได้ตาม ISC เช่น เครือข่ายแบบโอเมก้า เครือข่ายแบบฟลิบ เป็นต้น โดยสามารถยกตัวอย่างรูปแบบ ISC ที่นิยมใช้ 2 แบบดังนี้

1) แบบเฟอร์เฟคชับเฟิล (Perfect Shuffle) คือการเลื่อนซ้าย 1 บิตแล้ววนกลับ (1-bit Circular-Left-Shift) ดังรูปที่ 2.4 (ก)
2) แบบอินเวอร์สเฟอร์เฟคชับเฟิล (Inverse Perfect Shuffle) คือ การเลื่อนขวา 1 บิตแล้ววนกลับ (1-bit Circular-Right-Shift) ดังรูปที่ 2.4 (ข)



(ก) (ข)

รูปที่ 2.4 แสดงตัวอย่างของ ISC (ก) แบบเฟอร์เฟคชับเฟิล (ข) แบบอินเวอร์สเฟอร์เฟคชับเฟิลของเครือข่ายที่มีขนาด N=8 (= 2^3)

2.1.4 ประเภทของเครือข่ายการเชื่อมต่อแบบมัลติสเตจ

เครือข่ายแบบ MIN ที่เชื่อมต่อด้วยการใช้สวิทช์ขนาด 2x2 ใช้ความสัมพันธ์แบบ ISC เชื่อมระหว่างสเตจแบบต่างๆ ได้กำหนดให้รูปแบบการเชื่อมต่อเครือข่ายมีจำนวนหน่วยประมวลผล $N = 2^n$

จำนวนสเตจ $= n = \log_2 N$ สเตจ

จำนวนสวิทช์ต่อสเตจจำนวน $= N/2$ สวิทช์

ดังนั้นการจัดเส้นทางภายใน (Self-Routing) สามารถกำหนดสวิทช์เพื่อจัดเส้นทางระหว่าง

ต้นทางที่บิต s_j (S: $s_{n-1}..s_j..s_1s_0$) และ

ปลายทางของบิต d_j (D: $d_{n-1}..d_j..d_1d_0$) ที่สเตจ i (โดยที่ $0 \le i < n$)

ถ้า $s_j \neq d_j$ ดังนั้นกำหนดให้ในสเตจที่ i สามารถปรับสวิทช์แบบสับเปลี่ยน (เช่น การสับเปลี่ยนช่องทางภายในสวิทช์เป็น 0-1 หรือ 1-0) และ

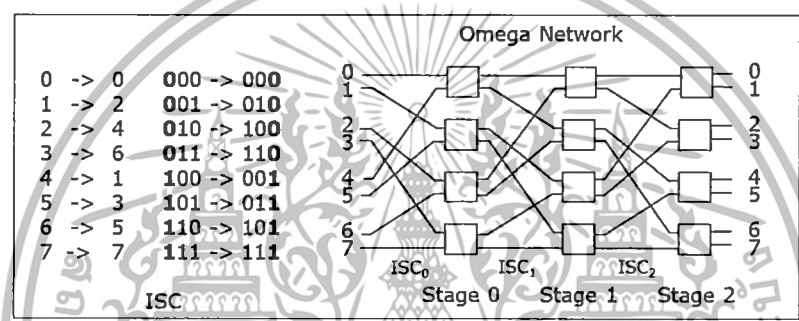
ถ้า $s_j = d_j$ ดังนั้นกำหนดให้ในสเตจที่ i สามารถปรับสวิทช์ภายในแบบตรง (เช่น การปรับช่องทางภายในสวิทช์เป็น 0-0 หรือ 1-1)

ดังนั้นในแต่ละรูปแบบของ ISC จึงจะสามารถสร้างเครือข่ายการเชื่อมต่อพื้นฐานได้ 5 แบบ ดังต่อไปนี้

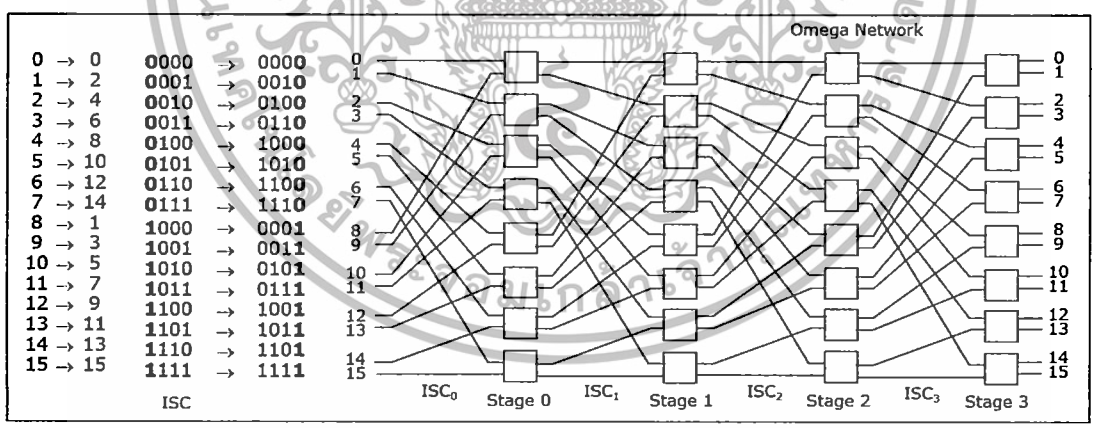
1) เครือข่ายแบบโอเมก้า (Omega Network)
2) เครือข่ายแบบฟลิบ (Flip Network)
3) เครือข่ายแบบคิวบ์ (Cube Network)
4) เครือข่ายแบบเบสไลน์ (Baseline Network)
5) เครือข่ายแบบอาร์ (R-Network)

2.1.4.1 เครือข่ายแบบโอเมก้า

เครือข่ายแบบโอเมก้า (Omega หรือ Shuffle Exchange Network) [4] จะใช้รูปแบบการเชื่อมต่อระหว่างสเตจทุกๆ สเตจแบบเพอร์เฟคชัฟเฟิลเป็นการเลื่อนซ้าย 1 บิตแล้ววนกลับ ทำให้บิตที่มีค่ามากสุด (Most Significant bit : MSb) ถูกเลื่อนไปอยู่ในตำแหน่งบิตที่มีค่าน้อยสุด (Least Significant bit : LSb) เช่น บิตที่ b_{n-1} ที่อยู่ในตำแหน่งบิตที่มีค่ามากที่สุดจะถูกเลื่อนออกไปทางซ้ายแล้ววนกลับไปอยู่ในตำแหน่งที่มีค่าน้อยสุดดังนี้ ($\boldsymbol{b_{n-1}}\, b_{n-2}...b_2b_1b_0 \longrightarrow b_{n-2}...b_2b_1b_0\boldsymbol{b_{n-1}}$) ดังแสดงตัวอย่างเครือข่ายแบบโอเมก้าในรูปที่ 2.5 (ก) ที่มีจำนวนหน่วยประมวลผล $N = 8 = 2^3$ ใช้สวิทช์ขนาด 2x2 ดังนั้น จำนวนสเตจ = $\log_2 N = 3$ สเตจ และจำนวนของสวิทช์ต่อสเตจ = $N/2 = 4$ สวิทช์ต่อสเตจ ดังนั้นทุกๆ ISC ใช้แบบการเลื่อนซ้าย 1 บิตแล้ววนกลับของทั้งหมด 3 บิต ดังนี้ ($\boldsymbol{b_2}b_1b_0 \longrightarrow b_1b_0\boldsymbol{b_2}$) สำหรับในรูปที่ 2.5 (ข) มีจำนวนหน่วยประมวลผล $N = 16 = 2^4$ ใช้ ISC แบบการเลื่อนซ้าย 1 บิตแล้ววนกลับของทั้งหมด 4 บิต เช่น ($\boldsymbol{b_3}b_2b_1b_0 \longrightarrow b_2b_1b_0\boldsymbol{b_3}$)



(ก)

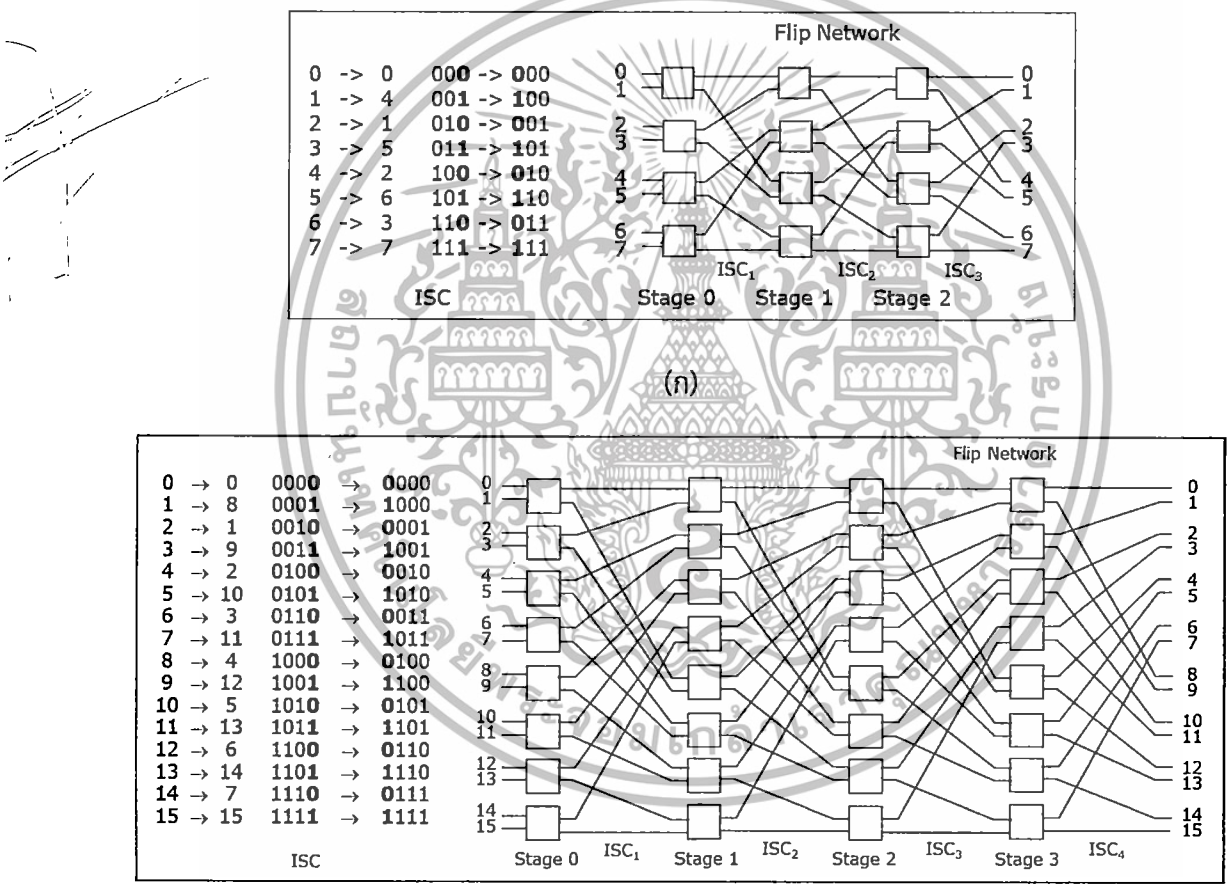


(ข)

รูปที่ 2.5 เครือข่ายแบบโอเมก้า (ก) N=8 และ (ข) N=16

2.1.4.2 เครือข่ายแบบฟลิบ

เครือข่ายแบบฟลิบ (Flip Network) [1] จะใช้รูปแบบการเชื่อมต่อระหว่างสเตจทุกๆ สเตจ แบบอินเวอร์สเพอร์เฟคชับเฟิลเป็นการเลื่อนขวา 1 บิตแล้ววนกลับ ทำให้บิตที่มีค่าน้อยสุด (Least Significant Bit) ถูกเลื่อนไปอยู่ในตำแหน่งบิตที่มีค่ามากสุด (Most Significant Bit) ดังนี้ ($b_{n-1}b_{n-2}...b_2b_1\mathbf{b_0} \longrightarrow \mathbf{b_0}b_{n-1}b_{n-2}...b_2b_1$) ยกตัวอย่างเครือข่ายแบบฟลิบที่ใช้สวิทช์ขนาด 2x2 ในรูปที่ 2.6 (ก) ซึ่งมีจำนวนหน่วยประมวลผล $N = 8 = 2^3$ ดังนั้น จำนวนสเตจ = $\log_2 N = 3$ สเตจ และจำนวนของสวิทช์ต่อสเตจ = $N/2 = 4$ ดังนั้นทุกๆ ISC ใช้แบบการเลื่อนขวา 1 บิตแล้ววนกลับของทั้งหมด 3 บิต ดังนี้ ($b_2b_1\mathbf{b_0} \longrightarrow \mathbf{b_0}b_2b_1$) สำหรับในรูปที่ 2.6 (ข) มีจำนวนหน่วยประมวลผล $N = 16 = 2^4$ ใช้ ISC แบบการเลื่อนขวา 1 บิตแล้ววนกลับของทั้งหมด 4 บิต เช่น ($b_3b_2b_1\mathbf{b_0} \longrightarrow \mathbf{b_0}b_3b_2b_1$)



รูปที่ 2.6 เครือข่ายแบบฟลิบ (ก) N=8 และ (ข) N=16

2.1.4.3 เครือข่ายแบบคิวบ์

เครือข่ายแบบคิวบ์ (Cube Network) [8] จะใช้รูปแบบการเชื่อมต่อระหว่างสเตจ ระหว่างสเตจที่ i-1 และสเตจที่ i คือ ISC_1, ISC_2, ... , ISC_{n-1} โดยที่ ISC_i (i = 1, 2, 3, ... , n-1) จะใช้รูปแบบการสลับหนึ่งคู่ของตำแหน่งบิตที่น้อยสุดและมากที่สุดภายในของ i+1บิต โดยที่อาจจะเป็นภายใน 2, 3, ... , n บิต ดังนี้

ISC_1: $b_{n-1}b_{n-2}...b_2(\mathbf{b_1b_0}) \longrightarrow b_{n-1}b_{n-2}...b_2(\mathbf{b_0b_1})$

ISC_2: $b_{n-1}b_{n-2}...(\mathbf{b_2}b_1\mathbf{b_0}) \longrightarrow b_{n-1}b_{n-2}...(\mathbf{b_0}b_1\mathbf{b_2})$

...

ISC_{n-1}: $(\mathbf{b_{n-1}}b_{n-2}...b_2b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_{n-2}...b_2b_1\mathbf{b_{n-1}})$

ส่วน ISC_n เป็นแบบอินเวอร์สเฟอร์เฟคชั่นเพิล ยกตัวอย่างเครือข่ายแบบคิวบ์ในรูปที่ 2.7 (ก) มีจำนวนหน่วยประมวลผล N = 8 มี 3 รูปแบบการเชื่อมต่อดังนี้ ISC_1: $b_2(\mathbf{b_1b_0}) \longrightarrow b_2(\mathbf{b_0b_1})$ ISC_2: $(\mathbf{b_2}b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_1\mathbf{b_2})$ และ ISC_3: $(b_2b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_2b_1)$ สำหรับในรูปที่ 2.7 (ข) มี 4 รูปแบบการเชื่อมต่อดังนี้ ดังนี้

ISC_1: $b_3b_2(\mathbf{b_1b_0}) \longrightarrow b_3b_2(\mathbf{b_0b_1})$

ISC_2: $b_3(\mathbf{b_2}b_1\mathbf{b_0}) \longrightarrow b_3(\mathbf{b_0}b_1\mathbf{b_2})$

ISC_3: $(\mathbf{b_3}b_2b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_2b_1\mathbf{b_3})$ และ

ISC_4: $(b_3b_2b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_3b_2b_1)$

2.1.4.4 เครือข่ายแบบเบสไลน์

เครือข่ายแบบเบสไลน์ (Baseline Network) [15] มีรูปแบบการเชื่อมต่อระหว่างสเตจที่ i-1 และสเตจ i ประกอบด้วย ISC_1, ISC_2, ... , ISC_{n-1} โดยที่ ISC_i ระหว่างสเตจที่ i-1 และสเตจที่ i เป็นการเลื่อนขวา 1 บิตแบบวนรอบภายใน n-(i-1) บิต โดยที่กำหนดให้ i = 1, 2, 3, ... , n-1 ดังนี้

ISC_1: $(b_{n-1}b_{n-2}...b_2b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_{n-1}b_{n-2}...b_2b_1)$

ISC_2: $b_{n-1}(b_{n-2}...b_2b_1\mathbf{b_0}) \longrightarrow b_{n-1}(\mathbf{b_0}\ b_{n-2}...b_2b_1)$

...

ISC_{n-1}: $b_{n-1}b_{n-2}...b_2(b_1\mathbf{b_0}) \longrightarrow b_{n-1}b_{n-2}...b_2(\mathbf{b_0}b_1)$

ยกตัวอย่างเครือข่ายแบบเบสไลน์ในรูปที่ 2.8 (ก) มีจำนวนหน่วยประมวลผล N = 8 จำนวนสเตจ n = 3 สเตจ จึงมีแบบการเชื่อมต่อ 2 แบบดังนี้

ISC_1: $(b_2b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_2b_1)$ และ

ISC_2: $b_2(b_1\mathbf{b_0}) \longrightarrow b_2(\mathbf{b_0}b_1)$

สำหรับในรูปที่ 2.8 (ข) มีจำนวนหน่วยประมวลผล N = 16 จำนวนสเตจ n = 4 สเตจ ดังนั้นมีรูปแบบการเชื่อมต่อ 3 แบบดังนี้

ISC_1: $(b_3b_2b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_3b_2b_1)$

ISC_2: $b_3(b_2b_1\mathbf{b_0}) \longrightarrow b_3(\mathbf{b_0}b_2b_1)$ และ

ISC_3: $b_3b_2(b_1\mathbf{b_0}) \longrightarrow b_3b_2(\mathbf{b_0}b_1)$

2.1.4.5 เครือข่ายแบบอาร์

เครือข่ายแบบอาร์ (R-Network) [7] มีรูปแบบการเชื่อมต่อระหว่างสเตจที่ i-1 และสเตจ i ประกอบด้วย ISC_1, ISC_2, ... , ISC_{n-1} โดยที่ ISC_i ระหว่างสเตจที่ i-1 และสเตจที่ i เป็นการเลื่อนขวา 1 บิตแบบวนรอบภายใน n-(i-1) บิต โดยที่กำหนดให้ i = 1, 2, 3, ... , n-1 และสำหรับรูปแบบการเชื่อมต่อระหว่างสเตจในเตจที่ n (ISC_n) ใช้การกลับบิตในทุกๆ ตำแหน่ง ดังนี้

ISC_1: $(b_{n-1}b_{n-2}...b_2b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_{n-1}b_{n-2}...b_2b_1)$

ISC_2: $b_{n-1}(b_{n-2}...b_2b_1\mathbf{b_0}) \longrightarrow b_{n-1}(\mathbf{b_0}\ b_{n-2}...b_2b_1)$

...

ISC_{n-1}: $b_{n-1}b_{n-2}...b_2(b_1\mathbf{b_0}) \longrightarrow b_{n-1}b_{n-2}...b_2(\mathbf{b_0}b_1)$

ISC_n: $(b_{n-1}b_{n-2}...b_2b_1b_0) \longrightarrow (b_0b_1b_2...\ b_{n-2}\ b_{n-1})$

ยกตัวอย่างเครือข่ายแบบอาร์ในรูปที่ 2.9 (ก) มีจำนวนหน่วยประมวลผล N = 8 จำนวนสเตจ n = 3 สเตจ จึงมีแบบการเชื่อมต่อ 3 แบบดังนี้

ISC_1: $(b_2b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_2b_1)$

ISC_2: $b_2(b_1\mathbf{b_0}) \longrightarrow b_2(\mathbf{b_0}b_1)$ และ

ISC_3: $(b_2b_1b_0) \longrightarrow (b_0b_1b_2)$

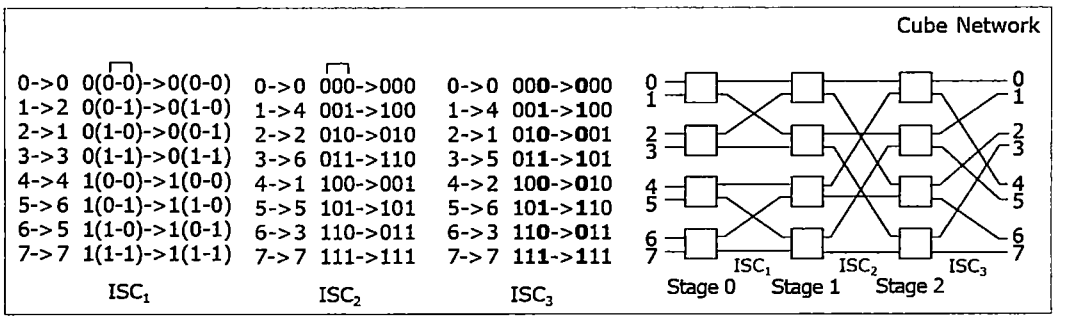
สำหรับในรูปที่ 2.9 (ข) มีจำนวนหน่วยประมวลผล N = 16 จำนวนสเตจ n = 4 สเตจ จึงมีแบบการเชื่อมต่อ 4 แบบดังนี้

ISC_1: $(b_3b_2b_1\mathbf{b_0}) \longrightarrow (\mathbf{b_0}b_3b_2b_1)$

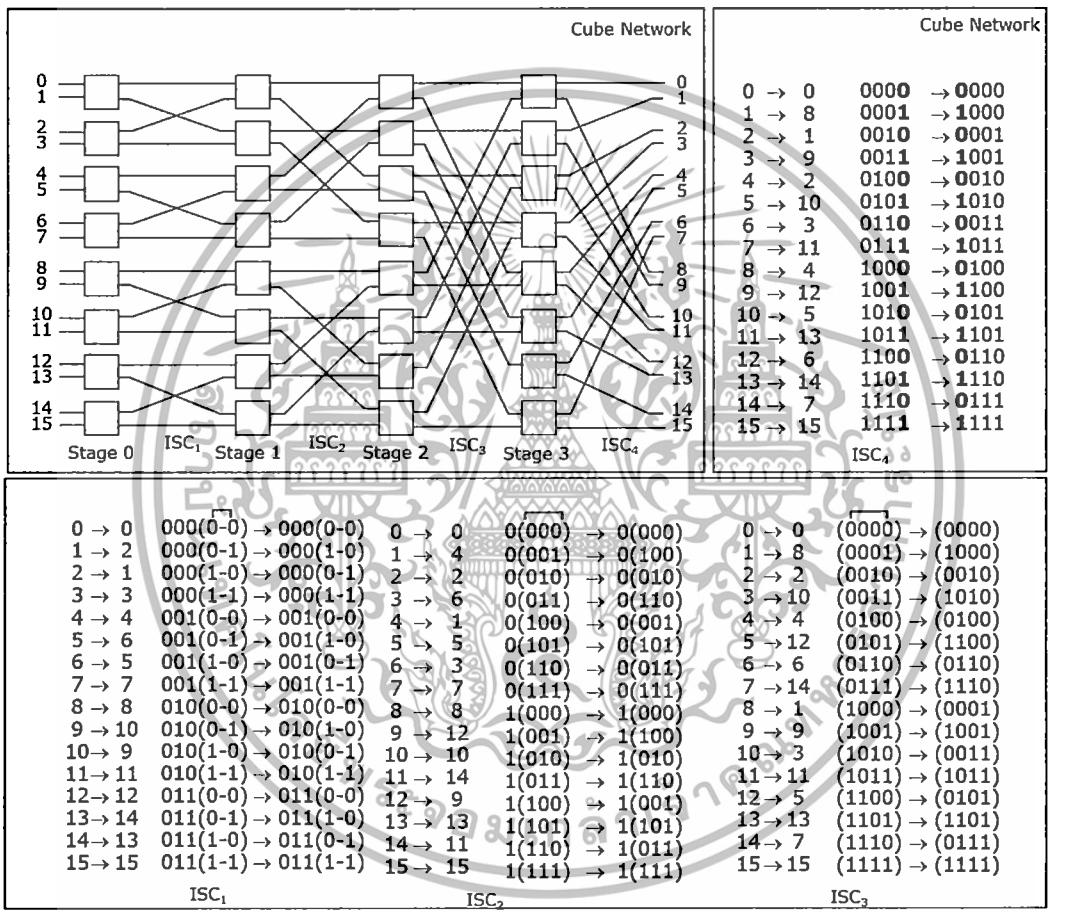
ISC_2: $b_3(b_2b_1\mathbf{b_0}) \longrightarrow b_3(\mathbf{b_0}b_2b_1)$

ISC_3: $b_3b_2(b_1\mathbf{b_0}) \longrightarrow b_3b_2(\mathbf{b_0}b_1)$ และ

ISC_4: $(b_3b_2b_1b_0) \longrightarrow (b_0b_3b_2b_1)$

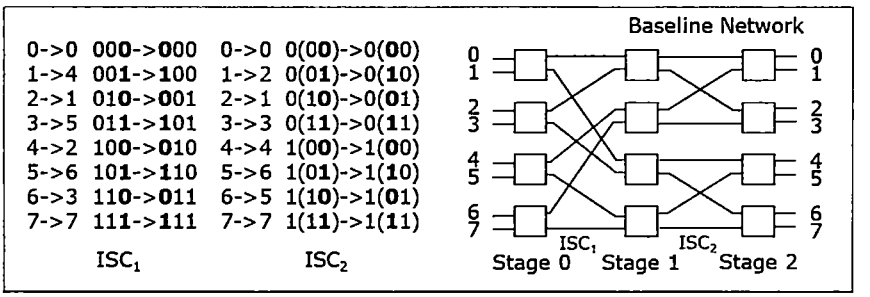


(ก)

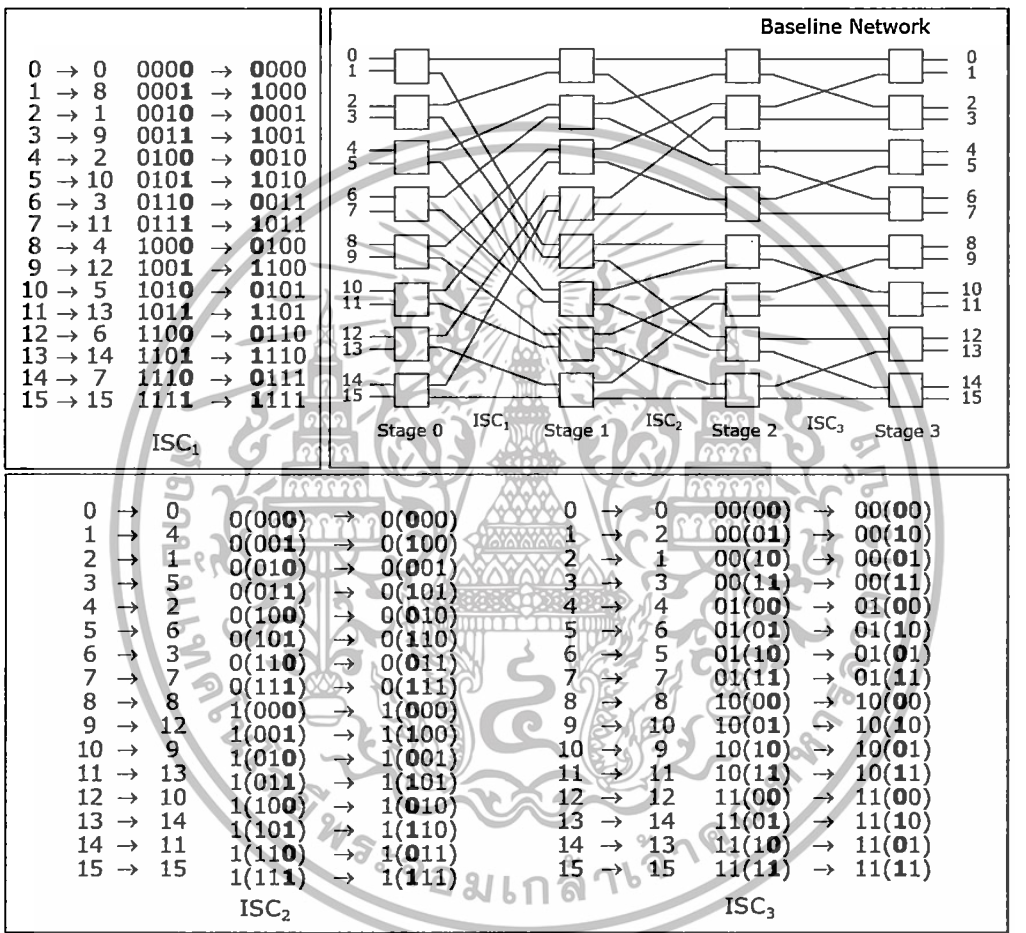


(ข)

รูปที่ 2.7 เครือข่ายแบบคิวบ์ (ก) N=8 และ (ข) N=16

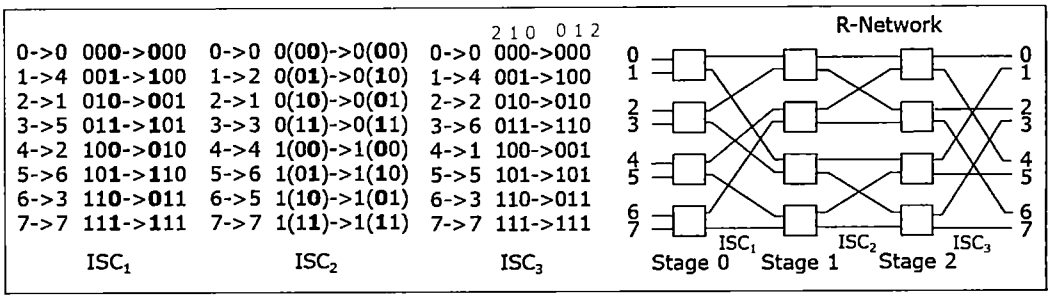


(ก)

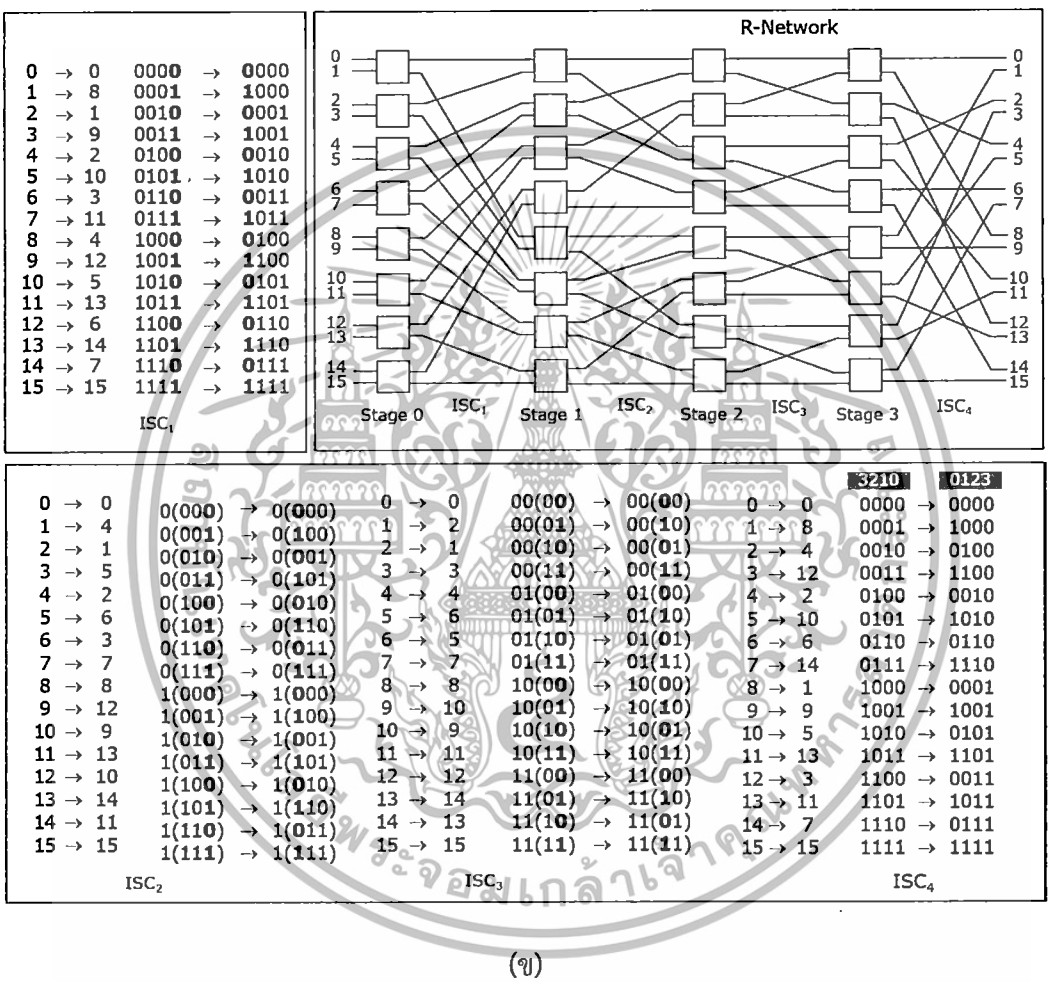


(ข)

รูปที่ 2.8 เครือข่ายแบบเบสไลน์ (ก) N=8 และ (ข) N=16



(ก)



(ข)

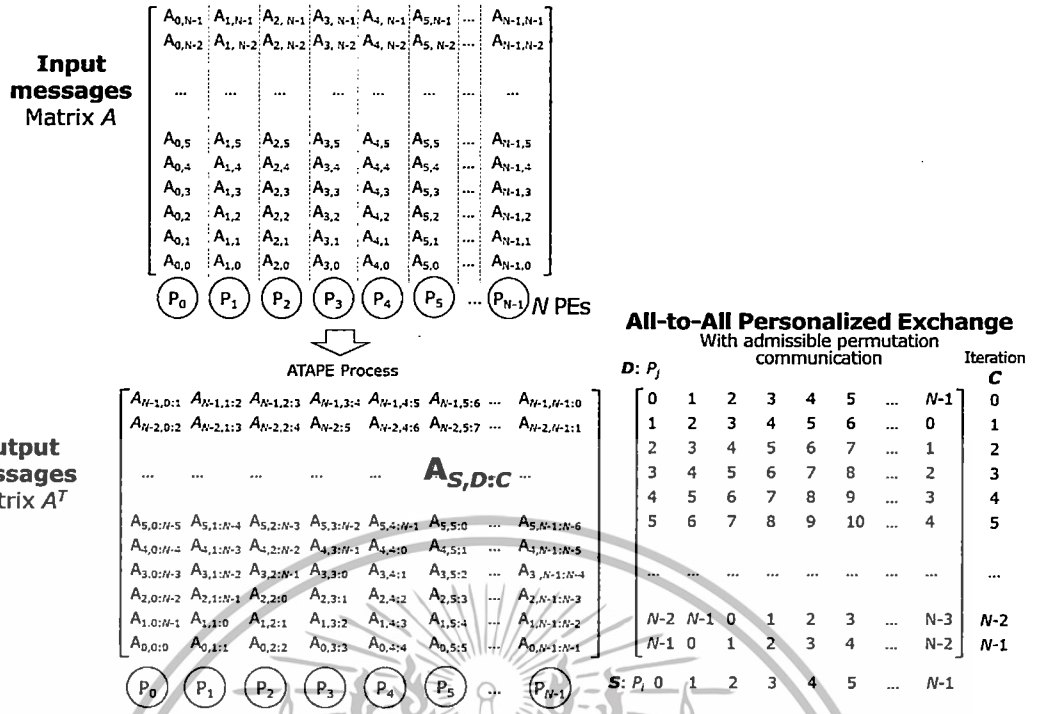
รูปที่ 2.9 เครือข่ายแบบอาร์ (ก) N=8 และ (ข) N=16

2.2 ขั้นตอนวิธีของการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์

การประมวลผลในระบบคอมพิวเตอร์แบบขนานและกระจายข้อมูลมีวิธีการติดต่อสื่อสารกันระหว่างหน่วยประมวลผลได้หลายรูปแบบ ซึ่งการติดต่อสื่อสารกันแบบขนานกับหน่วยประมวลผลอื่นนั้นอาจจะเป็นการติดต่อกันระหว่างหนึ่งหน่วยประมวลผล (One-to-One) หรือหนึ่งหน่วยประมวลผลกับอีกหลายๆ หน่วยประมวลผล (One-to-Many) หรือหนึ่งหน่วยประมวลผลกับอีกทุกๆ หน่วยประมวลผล (One-to-All) หรือในทุกๆ หน่วยประมวลผลกับอีกทุกๆ หน่วยประมวลผล (All-to-All) ซึ่งการที่จะทำการส่งหรือการรับข้อความจากทุกๆ หน่วยประมวลผลใดๆ สามารถแบ่งการติดต่อสื่อสารตามชนิดของข้อมูลที่ทำการส่งได้ 2 ประเภทคือ

- การกระจายข้อมูลแบบออล-ทู-ออล (All-to-All Broadcast : ATAB) คือทุกๆ หน่วยประมวลผล N หน่วยสามารถส่งข้อความเดียวกันไปยังทุกๆ หน่วยประมวลผลอื่นๆ

- การแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ (All-to-All Personalized Exchange : ATAPE) คือการที่หน่วยประมวลผลทุกๆ หน่วยประมวลสามารถส่งข้อความเฉพาะไปยังหน่วยประมวลผลอื่นๆ ได้ครบทุกๆ หน่วยประมวลผล ซึ่งแอพพลิเคชั่นที่สามารถใช้การติดต่อสื่อสารแบบ ATAPE ได้อย่างมีประสิทธิภาพคือการทรานสโพสเมตริกซ์แบบขนาน การแปลงฟูเรียร์แบบขนานและอื่นๆ ดังนั้นเพื่อให้เข้าใจการติดต่อสื่อสารแบบ ATAPE ได้ดียิ่งขึ้น ในรูปที่ 2.10 แสดงตัวอย่างการติดต่อสื่อสารแบบ ATAPE ในการทรานสโพสเมตริกซ์แบบขนานเป็นจำนวน N รอบของการติดต่อสื่อสารแบบขนานแบบจุดต่อจุด (Parallel Point-to-Point) เพื่อทำการส่งข้อมูลเฉพาะจากทุกๆ N ต้นทาง (Sources : S) ไปยังทุกๆ N ปลายทาง (Destinations : D) โดยใช้ฟังก์ชัน $D = (S+C) \bmod N$ ซึ่งผลลัพธ์ของการประมวลผลแบบ ATAPE แทนด้วยเมตริกซ์ทรานสโพสขนาด NxN หรือผลลัพธ์ในตารางลาตินของปลายทาง สำหรับการทรานสโพสของเมตริกซ์ขนาด NxN จะใช้หน่วยประมวลผลจำนวน N หน่วย เพื่อทรานสโพสเมตริกซ์ใน N รอบ ($C = 0$ ถึง N-1) เริ่มต้นโดยทำการแบ่งเมตริกส์เพื่อนำข้อมูลเข้า (ตามแนวคอลัมน์) เป็น N ส่วน ให้แต่ละส่วนเก็บอยู่ในหน่วยประมวลผล $P_i (i = 0, 1, 2, 3, ..., N-1)$ เช่น หน่วยประมวลผล P_0 จะเก็บเมตริกซ์นำข้อมูลเข้าส่วนที่ 0 (คอลัมน์ 0) ประกอบด้วย [0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], ..., [0, N–1] สำหรับหน่วยประมวลผล P_1- P_{N-1} จะเก็บเมตริกซ์ส่วนที่ 1 จนถึงส่วนที่ N-1 ตามลำดับ และผลลัพธ์ของการทรานสโพสเมตริกซ์ A ขนาด NxN คือ เมตริกซ์ A^T เช่น $A^T[i, j] = A[j, i]$ โดยที่ $0 \le i, j < N$ ในการติดต่อสื่อสารแบบ ATAPE ในรูปที่ 2.10 หน่วยประมวลผล P_0 - P_{N-1} จะทำการส่งหรือรับค่าทุกๆ ค่าในเมตริกซ์ A (N ค่าต่อรอบ) โดยแต่ละหน่วยประมวลผลจะทำการส่งหรือรับค่าตามแบบที่มีอยู่ของการติดต่อสื่อสารของวิธีเรียงสับเปลี่ยนที่เป็นไปได้ (Admissible Permutation Communication) เป็นจำนวน N-1 รอบของการรับส่งข้อมูล ดังแสดงในแถวที่ 1 ถึงแถวที่ N-1 สำหรับแถวที่ 0 คือค่าแนวทะแยงของเมตริกซ์ A และเมตริกซ์ A^T ซึ่งค่าจะมีอยู่แล้วใน P_0 - P_{N-1} ในรอบแรก ต่อไปในรอบที่ 2 หน่วยประมวลผล P_0 จนถึง P_{N-1} จะทำการส่งหรือรับค่าในเมตริกซ์ A จำนวน N ค่าที่ประกอบด้วย $[S, D]$ = [0, 1] [1, 2] ... $[j, j+1]$ ... [N-1, 0] จากทุกๆ หน่วยประมวลผลต้นทาง P_s ไปยังหน่วยประมวลผลปลายทาง P_d เมื่อครบรอบสุดท้ายทุกๆ ค่าของเมตริกซ์ A^T จะทำการทรานสโพสเสร็จเรียบร้อย โดยแอพพลิเคชั่นของการติดต่อสื่อสารแบบ ATAPE จะสามารถส่งค่าเฉพาะของเมตริกซ์ในทุกๆ หน่วยประมวลผลไปยังทุกๆ หน่วยประมวลผลอื่นๆ ได้ด้วยเวลา $O(N+\log_2 N)$ ซึ่งถ้าเป็นวิธีการเดิมจะส่งข้อมูลด้วยเวลา $O(N\log_2 N)$



รูปที่ 2.10 ตัวอย่างของการติดต่อสื่อสารแบบ ATAPE ด้วยฟังก์ชัน $D = (S+C) \bmod N$ เพื่อทรานสโพสเมตริกซ์ขนาด NxN โดยใช้ N หน่วยประมวลผล

ขั้นตอนวิธีแบบ ATAPE ได้รับความสนใจศึกษาอย่างแพร่หลายทั้งบนเครือข่ายแบบสแตติคและไดนามิคสำหรับบนเครือข่ายแบบสแตติคในปี ค.ศ. 1991 [12] นำเสนอ ATAPE ที่ใช้ตัวดำเนินการแบบ XOR ด้วยฟังก์ชัน $D = S \oplus C$ สำหรับจัดเส้นทางภายในบนเครือข่ายแบบสแตติค เช่น เครือข่ายไฮเปอร์คิวบ์ เครือข่ายแบบตาข่ายหรือกริด เป็นต้น อย่างไรก็ตามฟังก์ชันนี้แต่ละคู่ของหน่วยประมวลผลติดต่อสื่อสารด้วยเวลาไม่เท่ากัน (Inconsistent Latency) และไม่สามารถนำไปใช้ได้โดยตรงบนเครือข่ายแบบ MIN โดยเฉพาะอย่างยิ่งเครือข่ายแบบ MINs ที่เป็นรีเคอร์ซีฟ (Recursive MINs) เช่น เครือข่ายแบบเบสไลน์ เครือข่ายแบบบัตเตอร์ฟลาย และเครือข่ายแบบบันยาน ต่อมาในปี ค.ศ. 2013 [11] จึงได้มีการนำเสนอ ATAPE บนเครือข่ายดราก้อนฟลาย (Dragonfly Network) ซึ่งเป็นเครือข่ายสแตติคแบบลำดับชั้น (Static Hierarchical Network) ที่ประกอบด้วยสวิทช์ N^l ชั้น และเพิ่มสวิทช์ในแต่ละชั้นเพื่อให้ทุกคู่สามารถติดต่อสื่อสารด้วยเวลาเท่ากัน แต่การใช้สวิทช์ดังกล่าวจะทำให้การสร้างเครือข่ายมีค่าใช้จ่ายสูงโดยเฉพาะเมื่อเครือข่ายมีขนาดใหญ่ๆ

เครือข่ายแบบ MINs มีข้อดีคือ สามารถปรับรูปแบบการติดต่อสื่อสารได้ตามความต้องการของโปรแกรมในขณะประมวลผลด้วยแบบของการเชื่อมต่อภายในสวิทช์ และหน่วยประมวลผลทุกคู่ใช้เวลาติดต่อสื่อสารเท่ากัน ดังนั้นสังเกตว่าขั้นตอนวิธี ATAPE เดิมบนเครือข่ายแบบ MINs เดิมนั้นจะเน้นการจัดเส้นทางโดยใช้ค่าควบคุมสวิทช์เป็นหลัก เช่น เป็นการกำหนดสวิทช์จากค่าของต้นทาง (S) ตามด้วยการจัดเส้นทางด้วยแบบของสวิตช์บนสเตจ (Switch-Stage Pattern) หรืออาจจัดเส้นทางด้วยปลายทาง (D) เป็นหลัก คือเริ่มจากต้นทาง (S) ใดๆ แล้วต่อจากนั้นจะเป็นการกำหนดสวิทช์ตามค่าของปลายทาง (D-Control) เท่านั้น โดยการกำหนดค่าแบบต่างๆ จะใช้ค่าจากตารางลาตินของวิธีเรียงสับเปลี่ยนที่เป็นไปได้ในการกำหนดค่าให้แต่ละเครือข่ายแบบ MIN เรียกเทคนิคค่าควบคุมแบบนี้ว่า วิธีแก้ปัญหาการจัดเส้นทางแบบครึ่งเดียว (Half-Routing Solutions) ซึ่งจะเห็นว่าวิธีการนี้จะไม่

สามารถจัดเส้นทางด้วยต้นทาง (S) และปลายทาง (D) ทั้งสองค่าคู่กันตลอดเส้นทางผ่านสวิทช์ และสเตจบนเครือข่ายแบบ MINs ที่เป็นรีเคอร์ซีฟได้

หัวข้อนี้ได้เสนอผลงานวิจัยที่เกี่ยวข้องคือ ขั้นตอนวิธีแบบ ATAPE เดิมที่มีการศึกษาบนเครือข่ายการเชื่อมต่อแบบมัลติสเตจ (Multistage Interconnection Network : MINs) ในปี ค.ศ. 2000 - 2010ดังต่อไปนี้

2.2.1 การแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์บนเครือข่ายการเชื่อมต่อแบบมัลติสเตจ โดย Yang และ Wang

ในปี ค.ศ. 2000 Yang และ Wang [16] (Optimal All-to-All Personalized Exchange in Self-Routable Multistage Networks) ได้เสนอขั้นตอนวิธีแบบ ATAPE ที่ดีที่สุดบนเครือข่ายแบบ MIN ที่มีความยืดหยุ่นซึ่งใช้สวิทช์ขนาด 2x2 โดยมุ่งเน้นไปที่เครือข่ายแบบชัฟเฟิลเอคเชนจ์ (Shuffle Exchange Network) หรือ เครือข่ายแบบโอเมก้า (Omega Network) เครือข่ายแบบบันยาน (Banyan Network) และ เครือข่ายแบบเบสไลน์ (Baseline Network) ซึ่งจะใช้วิธีการสร้างตารางลาติน (Latin-Square : LS) พิเศษที่มีทุกๆ วิธีเรียงสับเปลี่ยน (Admissible Permutations) สำหรับแต่ละเครือข่ายเพื่อจัดเส้นทางภายในโดยใช้ค่าของวิธีเรียงสับเปลี่ยนในตารางลาตินกำหนดค่าภายในของแต่ละสวิทช์ในแต่ละสเตจนั่นคือการเชื่อมต่อภายในสวิตช์ตามแบบของค่าควบคุมในแต่ละสเตจ (Stage Switch-Setting Control) สำหรับเครือข่ายแบบ MINs แต่ละประเภท โดยเฉพาะอย่างยิ่งเครือข่ายแบบ MINs ที่เป็นรีเคอร์ซีฟ เช่น เครือข่ายแบบบันยาน เครือข่ายแบบเบสไลน์ เป็นต้น ซึ่งต้องการตารางลาตินที่มีค่าเฉพาะและแตกต่างในเครือข่ายแบบ MINs แต่ละประเภท ดังนั้นค่าควบคุม N ค่าของวิธีเรียงสับเปลี่ยนที่แตกต่างกันในแต่ละแถวของตารางลาตินหรือวิธีเรียงสับเปลี่ยน ρ (Permutation ρ) จะถูกใช้เป็นค่าสำหรับควบคุมของแต่ละสเตจเพื่อตั้งค่าภายในของสวิทช์ เพื่อทำการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์จากต้นทาง (Source : S) ไปยังปลายทาง (Destinations : D) โดยใช้การจัดเส้นทางภายในด้วยปลายทาง (D-Control Routing) เป็นต้น ซึ่งในงานวิจัยนี้ได้สร้างตารางลาตินสำหรับการติดต่อสื่อสารของ ATAPE โดยการจัดเส้นทางด้วยปลายทางโดยเน้นให้สามารถประมวลผลได้บนเครือข่ายที่เป็นรีเคอร์ซีฟ ดังต่อไปนี้

1.ตัวอย่าง ATAPE บนเครือข่ายแบบเบสไลน์ ดังรูปที่ 2.11 (ก) กำหนดให้ค่า N = 8 สำหรับทุกๆ วิธีเรียงสับเปลี่ยนจำนวน N ค่า คือ $\rho 1$ – $\rho 8$ ในตารางลาตินที่ 1 (L_1) ซึ่งใช้แทนการเชื่อมต่อระหว่างหน่วยประมวลผล (Processer: P) ดังต่อไปนี้

แถวที่ 1 ในตารางของวิธีเรียงสับเปลี่ยนคือ ρ_1 = (0 4 2 6 1 5 3 7) ใช้แทนการเชื่อมต่อระหว่างหน่วยประมวลผล ดังต่อไปนี้ $P_0 \longrightarrow P_0$ $P_1 \longrightarrow P_4$ $P_2 \longrightarrow P_2$ $P_3 \longrightarrow P_6$ $P_4 \longrightarrow P_1$ $P_5 \longrightarrow P_5$ $P_6 \longrightarrow P_3$ และ $P_7 \longrightarrow P_7$

แถวที่ 2 ในตารางของวิธีเรียงสับเปลี่ยนคือ ρ_2 = (1 5 3 7 0 4 2 6) ใช้แทนการเชื่อมต่อระหว่างหน่วยประมวลผล P ดังต่อไปนี้ $P_0 \longrightarrow P_1$ $P_1 \longrightarrow P_5$ $P_2 \longrightarrow P_3$ $P_3 \longrightarrow P_7$ $P_4 \longrightarrow P_0$ $P_5 \longrightarrow P_4$ $P_6 \longrightarrow P_2$ และ $P_7 \longrightarrow P_6$

แถวที่ 3 ในตารางของวิธีเรียงสับเปลี่ยนคือ ρ_3 = (3 7 1 5 2 6 0 4) ใช้แทนการเชื่อมต่อระหว่างหน่วยประมวลผล P ดังต่อไปนี้ $P_0 \longrightarrow P_3$ $P_1 \longrightarrow P_7$ $P_2 \longrightarrow P_1$ $P_3 \longrightarrow P_5$ $P_4 \longrightarrow P_2$ $P_5 \longrightarrow P_6$ $P_6 \longrightarrow P_0$ และ $P_7 \longrightarrow P_4$

...

แถวที่ 8 ในตารางของวิธีเรียงสับเปลี่ยนคือ ρ_8 = (4 0 6 2 5 1 7 3) ใช้แทนการเชื่อมต่อระหว่างหน่วยประมวลผล P ดังต่อไปนี้ $P_0 \longrightarrow P_4$ $P_1 \longrightarrow P_0$ $P_2 \longrightarrow P_6$ $P_3 \longrightarrow P_2$ $P_4 \longrightarrow P_5$ $P_5 \longrightarrow P_1$ $P_6 \longrightarrow P_7$ และ $P_7 \longrightarrow P_3$

2. <u>ตัวอย่าง ATAPE บนเครือข่ายแบบบันยาน</u> ดังรูปที่ 2.11 (ข) กำหนดให้ใช้หน่วยประมวลผล N = 8 สำหรับทุกๆ วิธีเรียงสับเปลี่ยนจำนวน N ค่า คือ $\rho 1 - \rho 8$ ในตารางลาตินที่ 2 (L_2) ซึ่งใช้แทนการเชื่อมต่อระหว่างหน่วยประมวลผล (Processer: P) ดังต่อไปนี้

แถวที่ 1 ในตารางของวิธีเรียงสับเปลี่ยนคือ ρ_1 = (0 2 4 6 1 3 5 7) ใช้แทนการเชื่อมต่อระหว่างหน่วยประมวลผล ดังต่อไปนี้ $P_0 \longrightarrow P_0$ $P_1 \longrightarrow P_2$ $P_2 \longrightarrow P_4$ $P_3 \longrightarrow P_6$ $P_4 \longrightarrow P_1$ $P_5 \longrightarrow P_3$ $P_6 \longrightarrow P_5$ และ $P_7 \longrightarrow P_7$

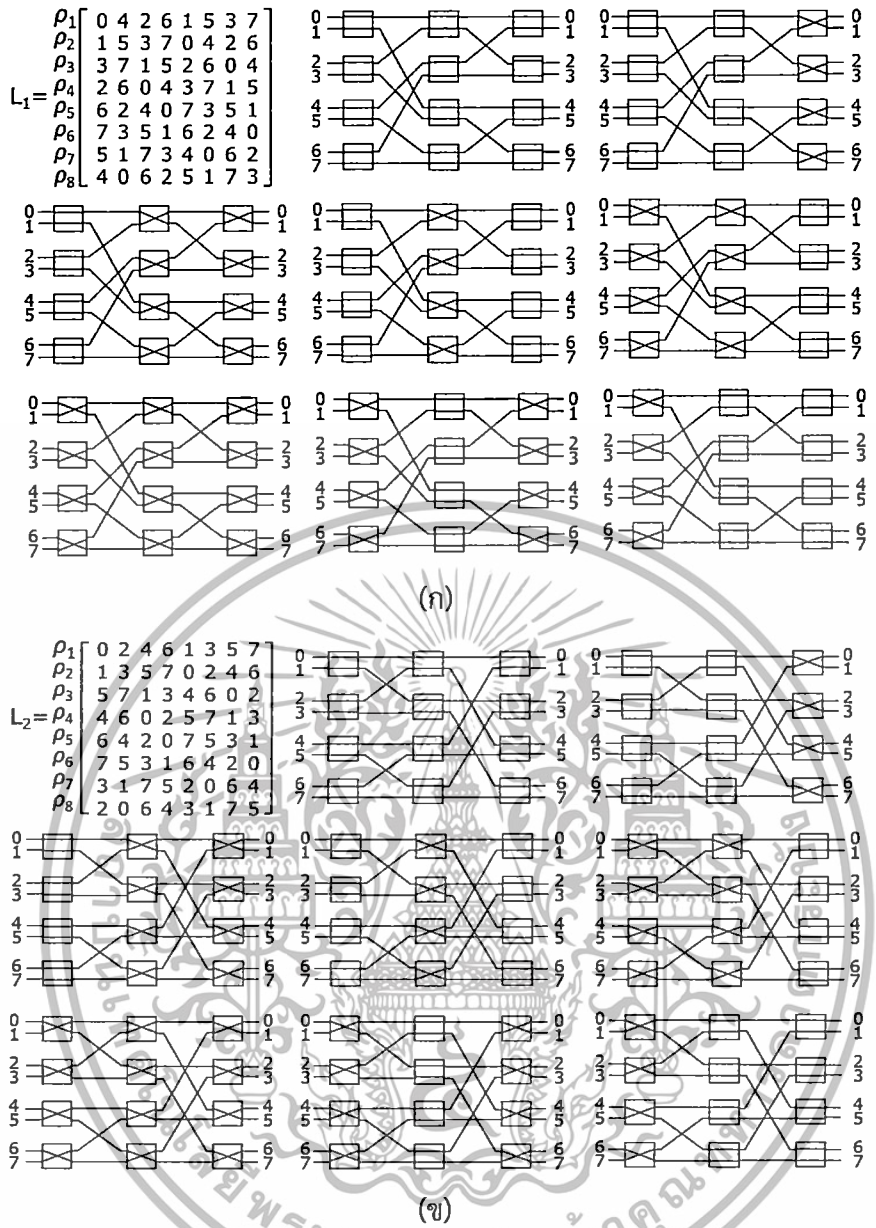
แถวที่ 2 ในตารางของวิธีเรียงสับเปลี่ยนคือ ρ_2 = (1 3 5 7 0 2 4 6) ใช้แทนการเชื่อมต่อระหว่างหน่วยประมวลผล P ดังต่อไปนี้ $P_0 \longrightarrow P_1$ $P_1 \longrightarrow P_3$ $P_2 \longrightarrow P_5$ $P_3 \longrightarrow P_7$ $P_4 \longrightarrow P_0$ $P_5 \longrightarrow P_2$ $P_6 \longrightarrow P_4$ และ $P_7 \longrightarrow P_6$

แถวที่ 3 ในตารางของวิธีเรียงสับเปลี่ยนคือ ρ_3 = (5 7 1 3 4 6 0 2) ใช้แทนการเชื่อมต่อระหว่างหน่วยประมวลผล P ดังต่อไปนี้ $P_0 \longrightarrow P_5$ $P_1 \longrightarrow P_7$ $P_2 \longrightarrow P_1$ $P_3 \longrightarrow P_3$ $P_4 \longrightarrow P_4$ $P_5 \longrightarrow P_6$ $P_6 \longrightarrow P_0$ และ $P_7 \longrightarrow P_2$

...

แถวที่ 8 ในตารางของวิธีเรียงสับเปลี่ยนคือ ρ_8 = (2 0 6 4 3 1 7 5) ใช้แทนการเชื่อมต่อระหว่างหน่วยประมวลผล P ดังต่อไปนี้ $P_0 \longrightarrow P_2$ $P_1 \longrightarrow P_0$ $P_2 \longrightarrow P_6$ $P_3 \longrightarrow P_4$ $P_4 \longrightarrow P_3$ $P_5 \longrightarrow P_1$ $P_6 \longrightarrow P_7$ และ $P_7 \longrightarrow P_5$

อย่างไรก็ตามวิธีการของขั้นตอนวิธี ATAPE แบบนี้ต้องสร้างตารางลาตินของวิธีเรียงสับเปลี่ยน แยกเฉพาะสำหรับเครือข่ายแบบ MINs ที่แตกต่างกัน ซึ่งจะจัดเส้นทางโดยใช้เทคนิคค่าควบคุมแบบสเตจ (Stage-Control Routing) โดยที่การติดต่อสื่อสารแบบ ATAPE ทุกคู่ของหน่วยประมวลผลจะสามารถติดต่อสื่อสารด้วยเวลาที่เท่ากัน เช่น ขั้นตอนวิธีแบบ ATAPE ที่เหมาะสมจะเตรียมการหน่วงเวลาผ่านสเตจ (Stage Delay) สำหรับการติดต่อสื่อสารผ่าน log_2N สเตจด้วยความซับซ้อนด้านเวลาคือ $O(N+\log_2 N)$ และมีความยืดหยุ่นดีกว่าการติดต่อสื่อสารแบบ ATAPE บนเครือข่ายแบบสแตติค (เช่น ไฮเปอร์คิวบ์ (Hypercubes) ตาข่ายหรือกริด (Meshes)) ซึ่งจะมีการหน่วงเวลาไม่เท่ากันด้วยความซับซ้อนด้านเวลาคือ $O(N\log_2 N)$ แต่ปัญหาที่พบในการจัดเส้นทางบนเครือข่ายการเชื่อมต่อแบบรีเคอร์ซีฟ (คือ เครือข่ายแบบเบสไลน์ และเครือข่ายแบบบันยาน) จะมีการเชื่อมต่อเพื่อทำการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์โดยใช้การจัดเส้นทางด้วยปลายทาง (D) เป็นหลัก และสังเกตว่าขั้นตอนวิธีแบบ ATAPE ของทั้งสองเครือข่ายไม่สามารถจัดเส้นทางจากต้นทางไปยังปลายทาง ((S, D) Self-Routing) ด้วยค่าไบนารี่ควบคุมของทั้ง S และ D ตลอดเส้นทางได้ แต่สามารถใช้เฉพาะค่าปลายทาง (D) ในแต่ละแถวของวิธีเรียงสับเปลี่ยนในตารางลาตินเฉพาะเครือข่ายเพื่อกำหนดสวิทช์แต่ละสวิทช์บนสเตจ (ผ่านจำนวน $n = \log_2 N$ สเตจ)

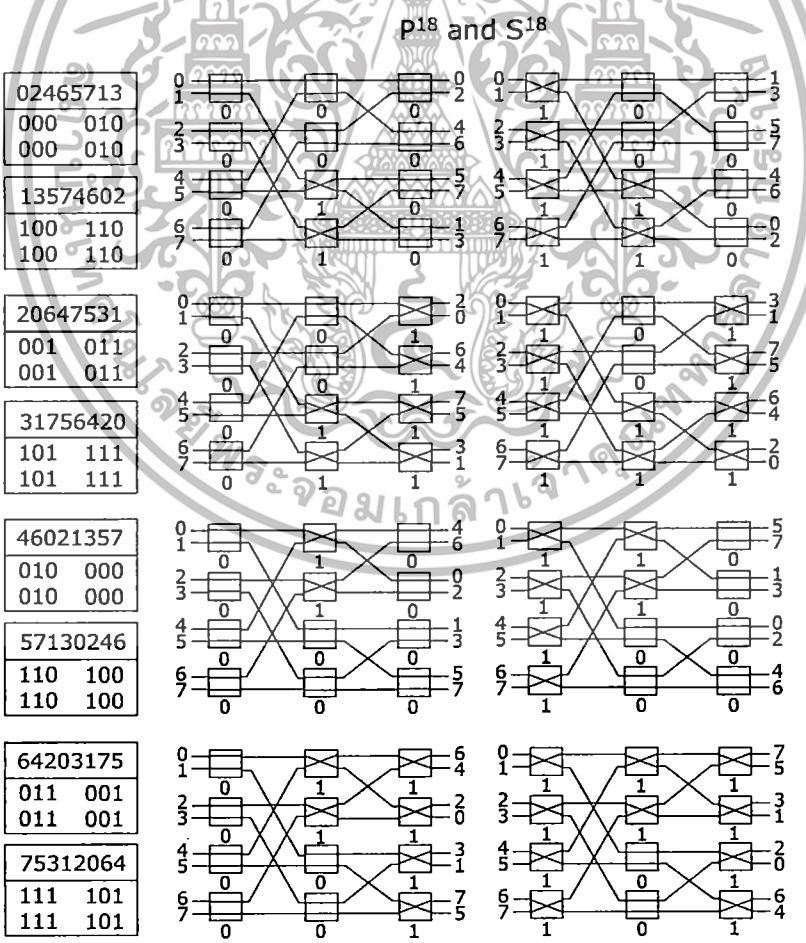


รูปที่ 2.11 การกำหนดสวิทช์ในแต่ละสเตจของ ATAPE [16] (ก) บนเครือข่ายแบบเบสไลน์ และ (ข) บนเครือข่ายแบบบันยาน (N = 8)

2.2.2 การติดต่อสื่อสารแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ บนเครือข่ายการเชื่อมต่อแบบมัลติสเตจ โดย Massini

ในปี ค.ศ. 2003 Massini [6] (All-to-All Personalized Communication on Multistage Interconnection Networks) ได้กล่าวว่าขั้นตอนวิธีของ Yang [16] เป็นขั้นตอนวิธีที่ขึ้นอยู่กับการควบคุมสวิทช์ทั้งหมดบนสเตจ (Stage-Control Routing) ของแต่ละชนิดของเครือข่ายนั่นคือต้องใช้ตารางลาตินเฉพาะสำหรับแต่ละประเภทของเครือข่ายแบบ MINs ดังนั้น Massini จึงเสนอขั้นตอนวิธีใหม่ที่ไม่ขึ้นอยู่กับแต่ละโครงสร้างเครือข่ายโดยแบ่งวิธีเรียงสับเปลี่ยนที่เป็นไปได้ทั้งหมดออกเป็น 2 ส่วนเรียกว่า วิธีเรียงสับเปลี่ยนที่เป็นไปได้ที่แบ่งแบบ 2 ทาง (2-Way Partitioning of Admissible Permutations : P) ประกอบด้วย ค่าของปลายทางที่ต้องการส่งข้อมูล และค่าไบนารี่สำหรับ

กำหนดการเชื่อมต่อภายในของแต่ละสวิทช์ตามแบบของวิธีเรียงสับเปลี่ยนที่เป็นไปได้ทั้งหมดที่แบ่งแบบ 2 ทางโดยออกแบบขั้นตอนวิธีมีการให้ทุกเครือข่ายใช้ตารางลาตินร่วมกันได้ ดังนั้นขั้นตอนวิธีแบบ ATAPE แบบนี้จึงไม่ขึ้นอยู่กับชนิดของเครือข่าย โดยขั้นตอนวิธีได้สร้างตารางลาตินที่มีการแบ่งวิธีเรียงสับเปลี่ยนแบบ 2 ทางเพื่อตั้งค่าสวิทช์ผ่าน $\log_2 N$ สเตจ ในการแบ่งวิธีเรียงสับเปลี่ยนดังกล่าวจะเตรียม P^l จำนวน $2^{(N/2)\log N} / N = N^{(N/2)-1}$ ชุด โดยที่ $l = 0, 1, 2, 3, ..., N^{(N/2)-1} - 1$ และชุดของทุกๆ ค่าไบนารี่ที่เป็นไปได้ ของแต่ละวิธีเรียงสับเปลี่ยนจะใช้สำหรับปรับเส้นทางในเครือข่ายโดยการเชื่อมต่อภายในสวิทช์สำหรับการจัดเส้นทางโดยใช้ค่าควบคุมที่ตัวสวิทช์ (Switch-Control Routing) เช่น ถ้าสถานะของสวิทช์เป็น 0 คือการเชื่อมต่อสวิทช์แบบตรง (—) หรือช่องทางเข้า/ออก 0-0 และช่องทางเข้า/ออก 1-1 และถ้าสถานะของสวิทช์เป็น 1 คือการเชื่อมต่อสวิทช์แบบไขว้ (×) หรือช่องทางเข้า/ออก 0-1 และช่องทางเข้า/ออก 1-0 สำหรับสวิทช์ขนาด 2x2 ตัวอย่างดังรูปที่ 2.12 แสดงตัวอย่างของวิธีเรียงสับเปลี่ยนที่เป็นไปได้ทั้งหมดแบ่งแบบ 2 ทางบนเครือข่ายบัตเตอร์ฟลาย โดยที่ $N = 8$ ใช้ $l = 18$ สำหรับ P^l และ S^l แต่ในการประยุกต์ใช้ ATAPE แบบนี้ยังคงพบปัญหาเช่นเดียวกับการศึกษาของ Yang และ Wang [16] คือยังคงไม่สามารถจัดเส้นทางได้จากต้นทางไปยังปลายทางด้วยค่าไบนารี่ของทั้ง S และ D ได้ตลอดเส้นทาง ((*S*, *D*) Self-Routing) บนเครือข่ายแบบ MINs ที่เป็นรีเคอร์ซีฟ (เครือข่ายบัตเตอร์ฟลาย เครือข่ายเบสไลน์ และเครือข่ายบันยาน) เช่นกัน



รูปที่ 2.12 ตัวอย่างวิธีเรียงสับเปลี่ยนที่เป็นไปได้ทั้งหมดแบ่งแบบ 2 ทางของการติดต่อสื่อสารแบบ ATAPE โดยที่ P^{18} และ S^{18} [6] บนเครือข่ายแบบบัตเตอร์ฟลาย (N=8 และ l=18)

2.2.3 การแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ในสวิทช์ขนาด *dxd* ของเครือข่ายการเชื่อมต่อแบบมัลติสเตจ โดย Liu และคณะ

ในปี ค.ศ. 2007 Liu และคณะ [5] (Optimal All-to-All Personalized Exchange in *d*-nary Banyan Multistage Interconnection Networks) ได้เสนอขั้นตอนวิธีแบบ ATAPE แบบใหม่สำหรับเครือข่ายแบบ MINs ที่ใช้สวิทช์ขนาด *dxd* ดังนั้นแต่ละเครือข่ายแบบ MINs จะมีจำนวนสเตจ $\log_d N$ สเตจ โดยขั้นตอนวิธีนี้จะใช้ตัวดำเนินการแบบ *d*-XOR สำหรับในแต่ละสวิทช์จะใช้เลขฐาน *d* แทนแบบของการกำหนดค่าในแต่ละสวิทช์ โดยกำหนดให้ *x* และ *y* เป็นช่องทางเข้าและช่องทางออกของแต่ละสวิทช์ขนาด *dxd* โดยที่ *x* และ *y* = (0, 1, 2, 3,..., *d*-1) ในการตั้งค่าของสวิทช์ขนาด *dxd* ใช้ชุดของ *d* แบบ (patterns: *P*) ประกอบด้วยเซตของแบบ *P* = {เลื่อนซ้าย 0 บิต เลื่อนซ้าย 1 บิต เลื่อนซ้าย 2 บิต เลื่อนซ้าย 3 เลื่อนซ้าย 4 บิต ... เลื่อนซ้าย *d*-1 บิต} ตามแบบของสวิทช์ที่มีการสร้างขึ้นโดยตัวดำเนินการของ *d*-XOR โดยภายในของแต่ละสวิทช์จะถูกคำนวณโดยฟังก์ชั่นดังนี้ $y = x \oplus_d p = (x + p) \bmod d$ สำหรับแต่ละ $(N/d) \times \log_d N$ สวิทช์ ตัวอย่างแบบของสวิทช์ 4 แบบ โดยกำหนดให้ $d = 4$ และ $p = 0, 1, 2, 3$ ดังนั้นสวิทช์แรกกำหนดให้ค่าภายในสวิทช์เลื่อนซ้าย 0 บิต ($y = x$) สวิทช์สองกำหนดให้ค่าภายในสวิทช์เลื่อนซ้าย 1 บิต ($y = (x+1)\%4$) สวิทช์สามกำหนดให้ค่าภายในสวิทช์เลื่อนซ้าย 2 บิต ($y = (x+2)\%4$) สวิทช์สี่กำหนดให้ค่าภายในสวิทช์เลื่อนซ้าย 3 บิต ($y = (x+3)\%4$) ดังในรูปที่ 2.13 (ก) สำหรับรูปที่ 2.13 (ข) แสดงวิธีเรียงสับเปลี่ยนจำนวน 16 วิธี สำหรับ $N = 16$ และจำนวนสเตจ $= \log_4 16 = 2$ สเตจ ดังนั้นใช้ค่าไบนารี่ 2 บิตแทนเลขฐาน 4 สำหรับจัดเส้นทางที่แตกต่างจากหน่วยประมวลผลต้นทาง *S* ไปยังหน่วยประมวลปลายทาง *D* ดังนี้

ในรอบที่ 0 ($= 00_4$) ดังนั้นวิธีเรียงสับเปลี่ยนของตารางลาตินขนาด 16x16 คือ [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15] สำหรับเชื่อมต่อระหว่างหน่วยประมวลผลต้นทางไปยังปลายทางดังนี้ $P_0 \longrightarrow P_0$ $P_1 \longrightarrow P_1$ $P_2 \longrightarrow P_2$ $P_3 \longrightarrow P_3$ $P_4 \longrightarrow P_4$ $P_5 \longrightarrow P_5$ ... และ $P_{15} \longrightarrow P_{15}$

ในรอบที่ 1 ($= 01_4$) ดังนั้นวิธีเรียงสับเปลี่ยนของตารางลาตินขนาด 16x16 คือ [1 2 3 0 5 6 7 4 9 10 11 8 13 14 15 12] สำหรับเชื่อมต่อระหว่างหน่วยประมวลผลต้นทางไปยังปลายทางดังนี้ $P_0 \longrightarrow P_1$ $P_1 \longrightarrow P_2$ $P_2 \longrightarrow P_3$ $P_3 \longrightarrow P_0$ $P_4 \longrightarrow P_5$ $P_5 \longrightarrow P_6$ ... และ $P_{15} \longrightarrow P_{12}$

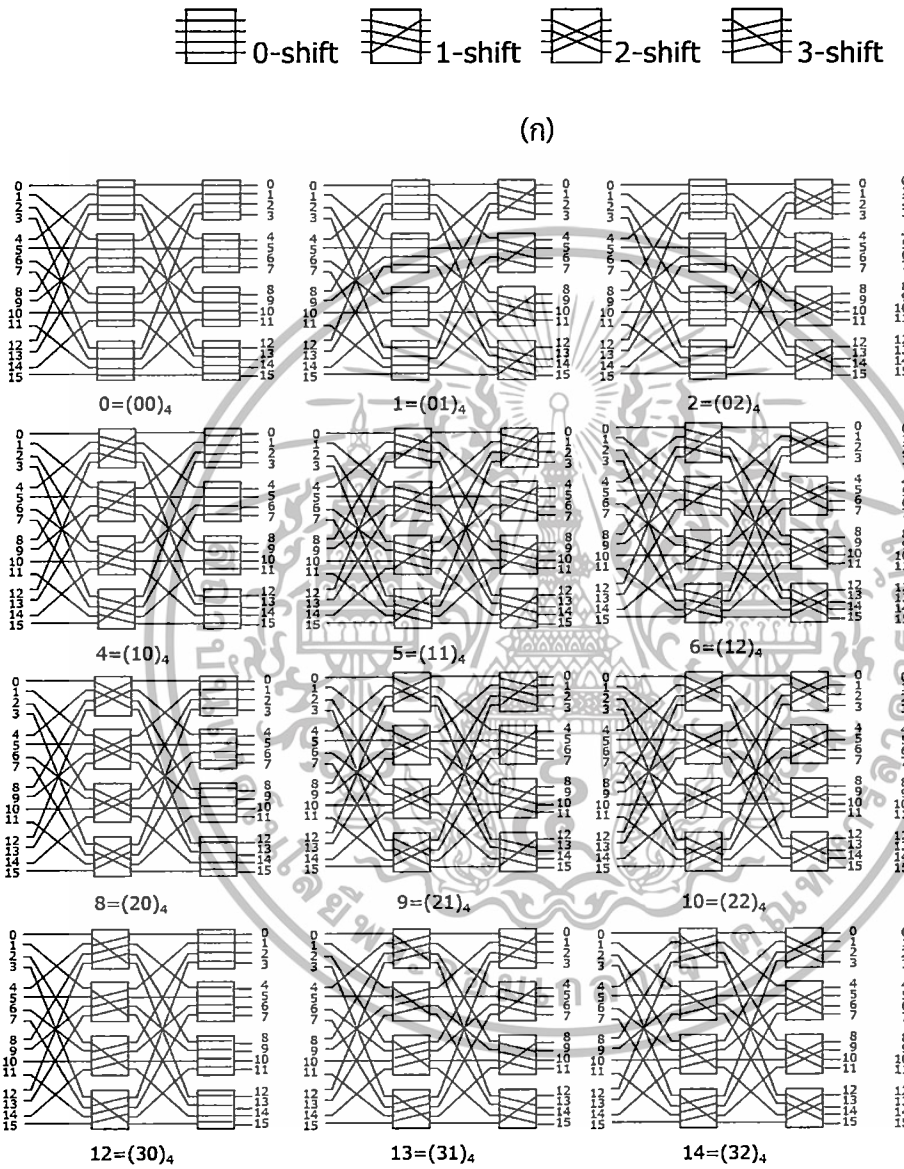
ในรอบที่ 2 ($= 02_4$) ดังนั้นวิธีเรียงสับเปลี่ยนของตารางลาตินขนาด 16x16 คือ [2 3 0 1 6 7 4 5 10 11 8 9 14 15 12 13] สำหรับเชื่อมต่อระหว่างหน่วยประมวลผลต้นทางไปยังปลายทางดังนี้ $P_0 \longrightarrow P_2$ $P_1 \longrightarrow P_3$ $P_2 \longrightarrow P_0$ $P_3 \longrightarrow P_1$ $P_4 \longrightarrow P_6$ $P_5 \longrightarrow P_7$ ... และ $P_{15} \longrightarrow P_{13}$

...

ในรอบที่ 14 ($= 32_4$) ดังนั้นวิธีเรียงสับเปลี่ยนของตารางลาตินขนาด 16x16 คือ [14 15 12 13 2 3 0 1 6 7 4 5 10 11 8 9] สำหรับเชื่อมต่อระหว่างหน่วยประมวลผลต้นทางไปยังปลายทางดังนี้ $P_0 \longrightarrow P_{14}$ $P_1 \longrightarrow P_{15}$ $P_2 \longrightarrow P_{12}$ $P_3 \longrightarrow P_{13}$ $P_4 \longrightarrow P_2$ $P_5 \longrightarrow P_3$ ... และ $P_{15} \longrightarrow P_9$

ในรอบที่ 15 ($= 33_4$) ดังนั้นวิธีเรียงสับเปลี่ยนของตารางลาตินขนาด 16x16 คือ [15 12 13 14 3 0 1 2 7 4 5 6 11 8 9 10] สำหรับเชื่อมต่อระหว่างหน่วยประมวลผลต้นทางไปยังปลายทางดังนี้ $P_0 \longrightarrow P_{15}$ $P_1 \longrightarrow P_{12}$ $P_2 \longrightarrow P_{13}$ $P_3 \longrightarrow P_{14}$ $P_4 \longrightarrow P_3$ $P_5 \longrightarrow P_0$ ... และ $P_{15} \longrightarrow P_{10}$

ถึงแม้ว่าจะเป็นการติดต่อสื่อสาร ATAPE แบบใหม่สำหรับสวิทช์ขนาด *dxd* ก็ตามแต่เมื่อกลับมาใช้สวิทช์ขนาด 2 x 2 บนเครือข่ายแบบ MINs เช่นเครือข่ายแบบโอเมก้า เป็นต้น แล้ววิธีเรียงสับเปลี่ยนของการติดต่อสื่อสารแบบ ATAPE แบบนี้ก็จะใช้วิธีการจัดเส้นทางด้วยค่าควบคุมแบบสเตจเหมือนกันกับวิธีของ Yang และ Wang [16]



รูปที่ 2.13 (ก) ตัวอย่างการเชื่อมต่อสวิทช์ขนาด 4x4 จำนวน 4 แบบด้วยค่าควบคุมแบบสเตจ และ (ข) การจัดเส้นทางภายในด้วยวิธีเรียงสับเปลี่ยนของการติดต่อสื่อสารแบบ ATAPE บนเครือข่ายแบบ MIN (N = 16, d = 2) จำนวน 16 วิธี [5]

2.2.4 การแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์บนเครือข่ายแบบชับเฟิลเอคเซนจ์ทั่วไป โดย Chou และ Chen

ในปี ค.ศ. 2010 Chou และ Chen [3] (All-to-All Personalized Exchange in Generalized Shuffle-Exchange Networks) ได้เสนอขั้นตอนวิธีแบบ ATAPE บนเครือข่ายแบบชับเฟิลเอคเซนจ์ ทั่วไป (Generalized Shuffle Exchange Networks : GSENs) โดยเน้นศึกษาเฉพาะที่ $N \neq 2^n$ สำหรับเครือข่ายแบบชับเฟิลเอคเซนจ์ หรือโอเมก้า (เมื่อ $N = 2^n$) ในงานวิจัยนี้ได้กำหนดให้สเตจมีขนาด = $\lceil \log_2 N \rceil$ สเตจ แต่ละสเตจมี $N/2$ สวิทช์ และใช้ ISC แบบเฟอร์เฟคชับเฟิล (Perfect Shuffle) ในกรณีทั่วไปซึ่งสามารถแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์โดยใช้เทคนิคค่าควบคุมแบบสลับสเตจ (Alternating Stage Control : A) กำหนดให้ $A = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots + a_2 2^2 + a_1 2^1 + a_0 2^0$ โดยที่ $0 \leq A < 2^n$ สำหรับการตั้งค่าสวิทช์จะสลับระหว่าง ตรง (—) หรือ 0 และไขว้ (X) หรือ 1 โดยค่าควบคุมบิต a_ℓ ดังต่อไปนี้ สำหรับ $N \equiv 2 \pmod 4$ กำหนดให้ a_ℓ แทนค่าควบคุมสำหรับตั้งค่าสวิทช์ที่สเตจ n-1-ℓ ดังต่อไปนี้

$a_\ell = 0$ หมายถึงการตั้งค่าสลับกันในแต่ละสวิทช์เป็น 0, 1, 0, 1, 0, 1, ...

$a_\ell = 1$ หมายถึงการตั้งค่าสลับกันในแต่ละสวิทช์เป็น 1, 0, 1, 0, 1, 0, ...

ดังนั้นการตั้งค่าสลับ N ค่า ($A_0, A_1, A_2, \ldots, A_{N-1}$) จะถูกคำนวณด้วยสูตรดังนี้ $A_k = k \oplus \lfloor k/2 \rfloor$ โดยที่ $k = 0, 1, 2, 3, \ldots, N$-1 เช่น ตัวอย่างเครือข่ายแบบ GSEN กรณี $N = 10$ ดังนั้น ค่าควบคุมการตั้งค่าแบบสลับสเตจจำนวน 10 ค่า ดังนี้ $A_0 = 0$, $A_1 = 1$, $A_2 = 3$, $A_3 = 2$, $A_4 = 6$, $A_5 = 7$, $A_6 = 9$, $A_7 = 10$, $A_8 = 12$ และ $A_9 = 13$ เพื่อใช้สำหรับตั้งค่าเครือข่ายในการส่งข้อมูลแบบออล-ทู-ออลเพอร์ซันนัลไลซ์จำนวน 10 รอบ โดยใช้ 4 บิต สำหรับ ($a_3a_2a_1a_0$) ตั้งค่าสวิทช์ผ่านสเตจจำนวน 4 สเตจ ดังรูปที่ 2.14 ยกตัวอย่างเช่น

ในรอบที่ 0 ค่าสลับคือ $A_0 = 0 = 0000_2$ ดังนั้นในแต่ละสเตจใช้ค่าควบคุมแต่ละบิตของการสลับค่า ดังนี้ สเตจ 0 ถึงสเตจ 3 ใช้บิต 0_2 สำหรับสลับในแต่ละสวิทช์ระหว่าง 0 1 0 1 0 (- x – x : ตรง ไขว้ ตรง ไขว้)

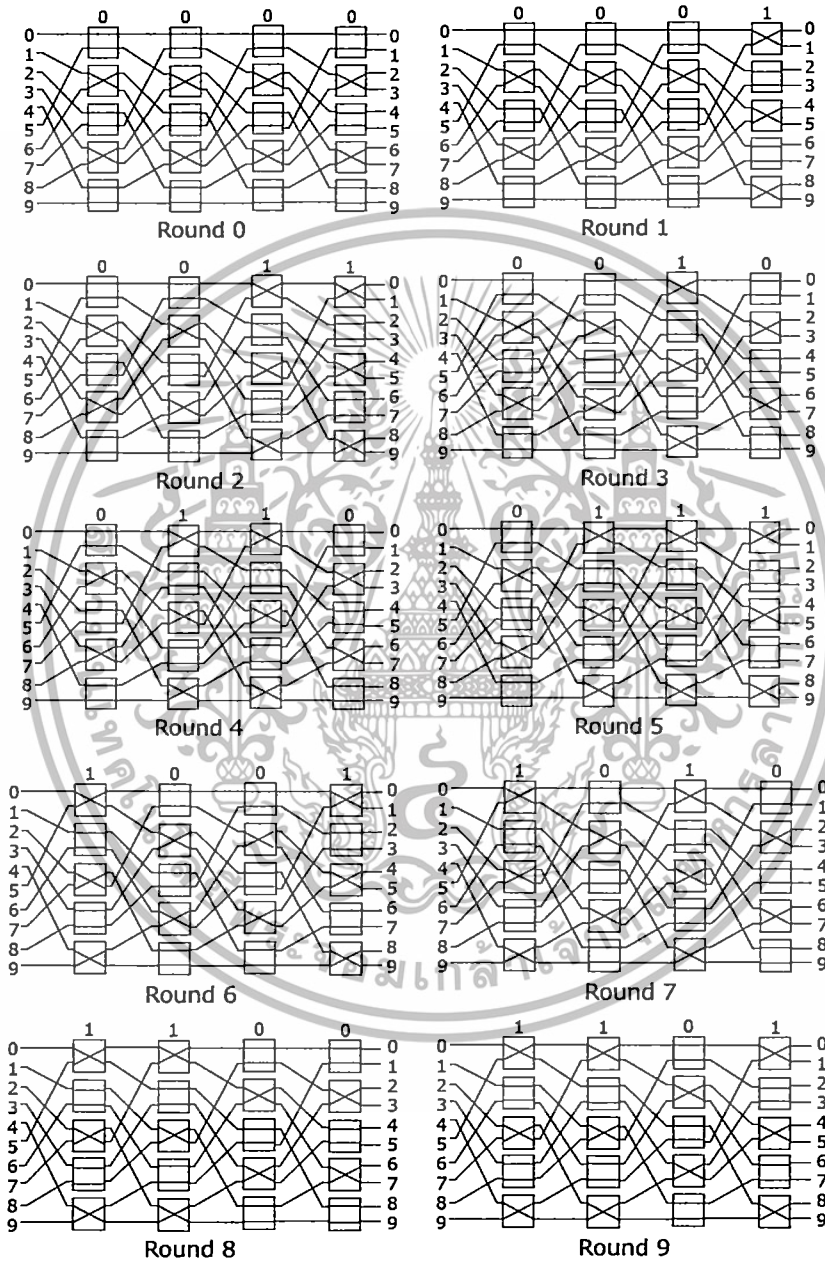
ในรอบที่ 1 ค่าสลับคือ $A_1 = 1 = 0001_2$ ดังนั้นในแต่ละสเตจใช้ค่าควบคุมแต่ละบิตของการสลับค่า ดังนี้ สเตจ 0 สเตจ 1 และสเตจ 2 ใช้บิต 0_2 สำหรับสลับในแต่ละสวิทช์ระหว่าง 0 1 0 1 0 (- x – x : ตรง ไขว้ ตรง ไขว้) และสเตจ 3 ใช้บิต 1_2 สำหรับสลับในแต่ละสวิทช์ระหว่าง 1 0 1 0 1 (x - x - x : ไขว้ ตรง ไขว้ ตรง)

ในรอบที่ 2 ค่าสลับคือ $A_2 = 3 = 0011_2$ ดังนั้นในแต่ละสเตจใช้ค่าควบคุมแต่ละบิตของการสลับค่า ดังนี้ สเตจ 0 และสเตจ 1 ใช้บิต 0_2 สำหรับสลับในแต่ละสวิทช์ระหว่าง 0 1 0 1 0 (- x – x : ตรง ไขว้ ตรง ไขว้) สเตจ 2 และสเตจ 3 ใช้บิต 1_2 สำหรับสลับในแต่ละสวิทช์ระหว่าง 1 0 1 0 1 (x - x - x : ไขว้ ตรง ไขว้ ตรง)

ในรอบที่ 3 ค่าสลับคือ $A_3 = 2 = 0010_2$ ดังนั้นในแต่ละสเตจใช้ค่าควบคุมแต่ละบิตของการสลับค่า ดังนี้ สเตจ 0 สเตจ 1 และสเตจ 3ใช้บิต 0_2 สำหรับสลับในแต่ละสวิทช์ระหว่าง 0 1 0 1 0 (- x – x : ตรง ไขว้ ตรง ไขว้) และสเตจ 2 ใช้บิต 1_2 สำหรับสลับในแต่ละสวิทช์ระหว่าง 1 0 1 0 1 (x - x - x : ไขว้ ตรง ไขว้ ตรง)

...

ในรอบที่ 9 ค่าสลับคือ $A_9 = 13 = 1101_2$ ดังนั้นในแต่ละสเตจใช้ค่าควบคุมแต่ละบิตของการสลับค่า ดังนี้ สเตจ 0 สเตจ 1 และสเตจ 3 ใช้บิต 1_2 สำหรับสลับในแต่ละสวิทช์ระหว่าง 1 0 1 0 1 (x - x - x : ไขว้ ตรง ไขว้ ตรง) และสเตจ 2 ใช้บิต 0_2 สำหรับสลับในแต่ละสวิทช์ระหว่าง 0 1 0 1 0 (- x – x : ตรง ไขว้ ตรง ไขว้)



รูปที่ 2.14 ตัวอย่างของขั้นตอนวิธี ATAPE บนเครือข่ายแบบ GSENs [3] สำหรับ N=10

บทที่ 3
การแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ที่ดีที่สุด แบบฝังลงบนชิปบนเครือข่ายการเชื่อมต่อแบบมัลติสเตจพลัส

ปัญหาของการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ (All-to-All Personalized Exchange : ATAPE) ได้มีการศึกษามากมายในหลายๆ รูปแบบของเครือข่าย (Network Topologies) เมื่อเร็วๆ นี้ได้มีการศึกษาแอพลิเคชั่น ATAPE ที่เน้นศึกษาบนเครือข่ายการเชื่อมต่อแบบมัลติสเตจ (Multistage Interconnection Networks : MINs) ซึ่งจะมีความยืดหยุ่นมากกว่าเครือข่ายแบบสแตติค (Static Networks) โดยการเชื่อมต่อบนเครือข่ายแบบไดนามิคสามารถทำการติดต่อสื่อสารระหว่างหน่วยประมวลผลคู่ใดๆ ด้วยการหน่วงเวลาผ่านเพียง $\log_2 N$ สเตจ ดังนั้นวิทยานิพนธ์นี้นำเสนอการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ที่ดีที่สุดแบบฝังลงบนชิปบนเครือข่ายการเชื่อมต่อแบบมัลติสเตจพลัส (On-Chip Embedding the optimal All-to-All Penalized Exchange on MINs$^+$) [9] โดยเครือข่ายแบบ MINs เดิมประกอบด้วย เครือข่ายแบบโอเมก้า (Omega Network) เครือข่ายแบบฟลิบ (Flip Network) เครือข่ายแบบคิวบ์ (Cube Network) เครือข่ายแบบบันยาน (Banyan Network) เครือข่ายแบบบัตเตอร์ฟลาย (Butterfly Network) และเครือข่ายแบบเบสไลน์ (Baseline Network) ส่วนเครือข่ายแบบ MINs$^+$ ที่เสนอขึ้นใหม่ในวิทยานิพนธ์นี้ เป็นเครือข่ายแบบ MINs ที่สมบูรณ์ประกอบด้วย เครือข่ายแบบบันยานพลัส (Banyan$^+$ Network) เครือข่ายแบบบัตเตอร์ฟลายพลัส (Butterfly$^+$ Network) เครือข่ายแบบเบสไลน์พลัส (Baseline$^+$ Network) และเครือข่ายแบบอินเวอร์สเบสไลน์พลัส (Inverst Baseline$^+$ Network) เป็นต้น

ก่อนอื่นในตารางที่ 3.1 จะแสดงการเปรียบเทียบระหว่างลักษณะสำคัญของการติดต่อสื่อสาร ATAPE ด้วยวิธีการเดิม เช่น การจัดเส้นทางภายในที่มีความยืดหยุ่นซึ่งปรับเปลี่ยนตามผลลัพธ์ในตารางลาติน เป็นต้น ซึ่งแต่ละขั้นตอนวิธีมีวิธีการแก้ปัญหาของ ATAPE แบบเดิมดังนี้ สำหรับสวิทช์ขนาด 2x2 วิธีการของ Yang และ Wang [16] ในปี ค.ศ. 2000 เสนอ ATAPE ที่เป็นต้นฉบับบนเครือข่ายแบบ MINs แต่ต้องทำการสร้างตารางลาตินก่อนสำหรับแต่ละชนิดของเครือข่ายแบบ MIN ส่วนการศึกษาของ Massini [6] ในปี ค.ศ. 2003 เสนอการสร้างตารางลาตินเดียวสำหรับ ATAPE เพื่อใช้ค่าในตารางกำหนดค่าให้กับแต่ละสวิทช์บนทุกๆเครือข่ายแบบ MINs ต่อมาในปี ค.ศ. 2007 Liu และคณะ [5] ได้เสนอ ATAPE บนเครือข่ายแบบ MINs ที่ใช้สวิทช์ขนาด dxd ในการจัดเส้นทางภายในเครือข่ายแบบ MINs ซึ่งจะใช้การกำหนดสวิทช์ด้วยเลขฐาน d ดังนั้นวิธีการ ATAPE แบบนี้จึงต้องการฟังก์ชันเฉพาะบนสวิทช์ขนาดต่างๆ บน $N/d \times \log_d N$ สวิทช์ซึ่งมีราคาแพงกว่าสวิทช์ทั่วไป หลังจากนั้นในปี ค.ศ. 2010 Chou และ Chen [3] จึงเสนอฟังก์ชันเฉพาะสำหรับเครือข่ายแบบชับเฟิลเอคเชนจ์ทั่วไป (Generalized Shuffle Exchange Networks : GSENs) สำหรับเครือข่ายที่มีขนาด $N \neq 2^n$ เพราะรองรับค่า N ที่เป็นค่าเฉพาะจึงไม่สามารถปรับวิธีการแบบเดิมๆ มาประยุกต์ใช้ได้โดยตรง แต่สำหรับเครือข่ายแบบ GSENs ที่มีขนาด N ใหญ่ๆ นอกจากสร้างเครือข่ายยุ่งยากแล้วการปรับวิธีการ ATAPE ก็ยุ่งยากอีกด้วย อย่างไรก็ตามทุกๆ วิธีของ ATAPE ดังกล่าวจะใช้แบบการจัดเส้นทางผ่านสวิทช์เป็นหลักโดยใช้แบบของการเชื่อมต่อภายในสวิทช์ที่มีการสลับสัญญาณผ่านได้ตามค่าควบคุมสวิทช์จากต้นทาง (Sources : S) ไปยังปลายทาง (Destinations : D) ซึ่งจะใช้

การจัดเส้นทางได้ทั้งการกำหนดค่าผ่านสเตจหรือสวิทช์ตามแบบของสวิตช์บนสเตจ (Switch-Stage Pattern) เทคนิควิธีแบบนี้เรียกว่า วิธีแก้ปัญหาที่ควบคุมการจัดเส้นทางแบบครึ่งเดียว (Half-Routing Solution) เพราะบนเครือข่ายแบบ MINs ที่เป็นแบบรีเคอร์ซีฟ มีการเชื่อมต่อ ISC ที่มีความซับซ้อนมากกว่าจึงไม่สามารถจัดเส้นทางจากต้นทางไปยังปลายทางได้โดยตรงตามค่าไบนารี่ของ S และ D ตลอดเส้นทาง (S, D Self-Routing)

ตารางที่ 3.1 ตารางเปรียบเทียบประโยชน์และขอบเขตของ ATAPE วิธีการเดิมทั้งบนเครือข่ายแบบสแตติค และไดนามิค (MINs) และวิธีการใหม่บนเครือข่ายแบบ MINs^{+}

วิธีการค้นหาเส้นทางของ ATAPE	เครือข่ายแบบสแตติค	เครือข่ายแบบไดนามิค					
	ไฮเปอร์คิวบ์ ดาก้อนฟลาย	MINs			GSENs	MINs^{+}	CMINs^{+}
		$O(N+\log N)$					
	1991 [12] 2013 [11]	2000 [16]	2003 [6]	2007 [5]	2010 [3]	ฟังก์ชัน ATAPE แบบใหม่ $D = S \oplus (C+order)_{\bmod N}$ $D=\rho[(S+C+order)_{\bmod N}]$	
ฟังก์ชัน ATAPE สำหรับการจัดเส้นทางภายในบนเครือข่ายแบบสแตติค	✓						
วิธีแก้ปัญหาของการจัดเส้นทางภายในแบบ ATAPE บนเครือข่ายแบบ MINs		half	half	half	half	full	full
○ การจัดเส้นทางโดยใช้ค่าควบคุมแบบสวิทช์ (Switch Control)			✓		✓	✓	✓
○ การจัดเส้นทางโดยใช้ค่าควบคุมแบบสเตจ (Stage Control)		✓		✓	✓	✓	✓
○ การจัดเส้นทางโดยใช้ค่าควบคุมแบบปลายทาง (D Control)		✓	✓	✓		✓	✓
○ การจัดเส้นทางภายในด้วยต้นทางและปลายทาง ((S, D) Control)						✓	✓
ตารางลาตินของปลายทาง							
○ ต้องการตารางลาตินเฉพาะสำหรับแต่ละเครือข่ายแบบ MIN		✓	✓	✓	✓		
○ สามารถใช้ตารางลาตินมาตรฐานทั่วไปสำหรับทุก ๆ เครือข่ายแบบ MINs^{+}						✓	✓
เครือข่ายที่ใช้สวิทช์ขนาด 2x2		✓	✓	✓	✓	✓	✓
เครือข่ายที่ใช้สวิทช์ขนาด $d \times d$				✓		✓	✓
ขนาดของเครือข่ายกรณีพิเศษ $N \neq 2^n$					✓		

ในวิทยานิพนธ์นี้จึงได้เสนอเครือข่ายสมบูรณ์แบบ MINs (Completely Self-Routable MINs) เรียกว่าเครือข่ายแบบ MINs$^+$ การจัดเส้นทางแบบเต็ม (Full Self-Routing) สำหรับการติดต่อสื่อสารแบบ ATAPE ที่สามารถฝังลงบนชิปบนเครือข่ายแบบ MINs$^+$ ซึ่งประกอบด้วยเครือข่ายแบบโอเมก้า (หรือชัฟเฟิลเอคเชนจ์) เครือข่ายแบบฟลิบ เครือข่ายแบบคิวบ์ เครือข่ายแบบบันยานพลัส เครือข่ายแบบบัตเตอร์ฟลายพลัส เครือข่ายแบบเบสไลน์พลัส และเครือข่ายแบบอินเวอร์สเบสไลน์พลัส พร้อมทั้งเสนอฟังก์ชัน $D = S$ XOR $(C + order)$ mod N สำหรับการจัดเส้นทางการสลับสัญญาณผ่านสวิทช์ได้อย่างอิสระ 3 วิธีประกอบด้วย

1. การจัดเส้นทางด้วยค่าควบคุมหลักของสเตจ (Stage-Control Routing)
2. การจัดเส้นทางด้วยค่าควบคุมหลักของสวิทช์ (Switch-Control Routing) และ
3. การจัดเส้นทางด้วยต้นทางและปลายทาง ((S, D) Self-Routing)

และในงานวิจัยนี้ค่าควบคุมหลัก N รอบจะสามารถกำหนดให้เพิ่มขึ้นได้อย่างต่อเนื่องกันโดยใช้ตัวนับค่าบนชิป (On-Chip Counter) คือ $C = 0, 1, 2, \ldots, N-1$ ด้วยเวลาที่เร็วขึ้นกว่าเดิมภายใต้การทำงานบนชิปคือ $O(N+\log_2 N)$ โดยเฉพาะอย่างยิ่งได้เสนอฟังก์ชันทางเลือก f-in-1 แบบปรับเปลี่ยนได้ (f-in-1 Dynamic Function) โดยโปรแกรมใช้งาน คือ $D = \rho[(S + C + order) \bmod N]$ ซึ่งขณะเวลาทำงานสามารถกำหนดค่าได้ยืดหยุ่นจากหนึ่งใน ATAPE ฟังก์ชันได้ fN ฟังก์ชันตามความต้องการของผู้ใช้ ส่วนการประยุกต์ใช้ฟังก์ชัน ATAPE สามารถถูกฝังลงบนชิปบนเครือข่ายแบบ MINs$^+$ ที่หลากหลาย เช่น สวิทช์ขนาด dxd และครอสส์บาร์ของเครือข่ายแบบ MINs$^+$ (Crossbar of MINs$^+$: CMINs$^+$) นอกจากนั้นสำหรับประยุกต์ใช้และการวัดประสิทธิภาพได้นำเสนอผลการทดลองของการนำขั้นตอนวิธี ATAPE ไปประยุกต์ใช้บนเครือข่ายแบบ MINs$^+$ และ CMINs$^+$ และได้นำเสนอการวัดประสิทธิภาพของแอพพลิเคชั่น ATAPE สำหรับคำนวณเมตริกซ์ทรานสโพส (A^T) แบบขนานขนาด $N \times N$ บนเครือข่ายแบบ MINs$^+$ ซึ่งทำให้ความเร็วในการส่งข้อมูลเร็วขึ้นด้วยเวลา $O(N+\log_2 N)$ เทียบกับวิธี A^T ปกติด้วยเวลา $O(N\log_2 N)$

เนื้อหาในบทที่ 3 นี้จะมีการแบ่งออกเป็น 4 ส่วนดังต่อไปนี้ ในหัวข้อที่ 3.1 เสนอ MINs$^+$ และวิธีการจัดเส้นทางภายในแบบสมบูรณ์ของ MINs$^+$ สำหรับขั้นตอนวิธี ATAPE แบบฝังลงบนชิปที่ดีที่สุดบนเครือข่ายแบบ MINs$^+$ ในหัวข้อที่ 3.2 ส่วนหัวข้อที่ 3.3 แสดงขั้นตอนวิธี ATAPE แบบฝังลงบนชิปที่ดีที่สุดบน CMINs$^+$ และหัวข้อที่ 3.4 แอพพลิเคชั่นของ ATAPE แบบฝังลงบนชิป

3.1 เครือข่ายแบบ MINs$^+$ และวิธีการจัดเส้นทางภายในแบบสมบูรณ์ของ MINs$^+$

เนื่องจากเครือข่ายแบบ MINs ที่เป็นรีเคอร์ซีฟที่มีอยู่เดิม เช่น เครือข่ายแบบบันยาน เครือข่ายแบบบัตเตอร์ฟลาย และเครือข่ายแบบเบสไลน์ จะมีวิธีการเชื่อมต่อระหว่างสเตจ (Inter-Stage Connections : ISCs) ที่ซับซ้อน เป็นการยากที่จะใช้ฟังก์ชัน $D = S \oplus C$ ในระดับบิตเพราะผลลัพธ์ที่ได้จากฟังก์ชันไม่สามารถนำมาใช้บนเครือข่ายแบบ MINs ที่เป็นรีเคอร์ซีฟได้โดยตรง ดังนั้นเพื่อที่จะสามารถใช้ฟังก์ชันนี้ในระดับบิตได้ วิทยานิพนธ์นี้จึงได้มีการปรับ ISC เพิ่มให้สมบูรณ์หรือเพิ่มเทคนิคการปรับพอร์ตบนเครือข่ายแบบ MINs ที่เป็นรีเคอร์ซีฟแต่ละแบบเป็นเครือข่ายแบบ MINs ชนิดใหม่เรียกว่า MINs$^+$ ดังนั้นเครือข่ายแบบ MINs$^+$ สามารถจัดเส้นทางได้สมบูรณ์ (Fully Routable MINs$^+$) รูปที่ 3.1 แสดงตัวอย่างของเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด 2x2 ที่ได้ทำการปรับพอร์ตของแต่ละเครือข่ายแบบ MINs$^+$ ดังนี้

- เครือข่ายแบบเบสไลน์พลัส (Baseline$^+$ Network) ได้ปรับพอร์ตที่ ISC_n คือการกลับตำแหน่งที่อยู่จำนวน n บิต โดยให้บิตที่มีนัยสำคัญสูงกลับไปอยู่ในตำแหน่งของบิตที่มีนัยสำคัญต่ำและบิตที่มีนัยสำคัญต่ำกลับไปอยู่ในตำแหน่งของบิตที่มีนัยสำคัญสูงด้วย เช่น ($b_{n-1}b_{n-2}...b_2b_1b_0 \longrightarrow b_0b_1b_2...\ b_{n-2}b_{n-1}$)
- เครือข่ายแบบอินเวอร์สเบสไลน์พลัส (Inverse Baseline$^+$ Network) ได้ปรับพอร์ตที่ ISC_0 คือการกลับตำแหน่งที่อยู่จำนวน n บิตเช่นกันกับ ISC_n ของเครือข่ายแบบเบสไลน์พลัส
- เครือข่ายแบบบันยานพลัส (Banyan$^+$ Network) ได้ปรับพอร์ตที่ ISC_n ด้วยการเลื่อนขวา 1 บิตแล้ววนกลับทำให้บิตที่อยู่ตำแหน่งน้อยสุด (Least Significant bit : LSb) ถูกเลื่อนไปอยู่ในตำแหน่งที่มากสุด (Most Significant bit : MSb)
- เครือข่ายแบบบัตเตอร์ฟลายพลัส (Butterfly$^+$ Network) ได้ปรับพอร์ตที่ ISC_0 คือการเลื่อนซ้าย 1 บิตแล้ววนกลับทำให้บิตที่อยู่ตำแหน่งมากที่สุด (MSb) ถูกเลื่อนไปอยู่ในตำแหน่งที่น้อยสุด (LSb)

สำหรับการตรวจสอบความถูกต้องของการจัดเส้นทางภายในของเครือข่ายแบบเบสไลน์พลัส จากต้นทาง (S) หรือ ข้อมูลนำเข้า ($I = (i_{n-1}i_{n-2} \dots i_2i_1i_0)$) ไปยังปลายทาง ($D$) หรือ ข้อมูลออก ($J = (j_{n-1}j_{n-2} \dots j_2j_1j_0)$) ในแต่ละสเตจมีการเชื่อมต่อ $ISC_1 - ISC_n$ พร้อมการปรับสวิทซ์ดังต่อไปนี้

sw-ISC_1: $(i_{n-1}i_{n-2} \dots i_2i_1i_0) \longrightarrow (j_0i_{n-1}i_{n-2} \dots i_2i_1)$

sw-ISC_2: $j_0\ (i_{n-1}i_{n-2} \dots i_2i_1) \longrightarrow j_0\ (j_1i_{n-1}i_{n-2} \dots i_2)$

...

sw-ISC_{n-1}: $j_0j_1j_2j_3...j_{n-3}(i_{n-1}i_{n-2}) \longrightarrow j_0j_1j_2j_3...j_{n-3}(j_{n-2}i_{n-1})$

[sw-ISC_n: $(j_0j_1j_2j_3...j_{n-3}j_{n-2}i_{n-1}) \longrightarrow (j_{n-1}j_{n-2}...j_3j_2j_1j_0)$]

ความถูกต้องของสำหรับเครือข่ายแบบบันยานพลัสจะสามารถการจัดเส้นทางภายในของการเชื่อมต่อระหว่างสเตจได้สมบูรณ์โดยมีการเชื่อมต่อระหว่างสเตจของ $ISC_1 - ISC_n$ พร้อมการปรับสวิทซ์ได้ดังนี้

sw-ISC_1: $i_{n-1}i_{n-2} \dots i_4i_3i_2\ (i_1i_0) \longrightarrow i_{n-1}i_{n-2} \dots i_3i_2\ (j_0i_1)$

sw-ISC_2: $i_{n-1}i_{n-2} \dots i_4i_3\ (i_2j_0i_1) \longrightarrow i_{n-1}i_{n-2} \dots i_3\ (j_1j_0i_2)$

...

sw-ISC_{n-1}: $(i_{n-1}j_{n-3}...j_2j_1j_0i_{n-2}) \longrightarrow (j_{n-2}j_{n-3} \dots j_2j_1j_0i_{n-1})$

[sw-ISC_n: $j_{n-2}j_{n-3} \dots j_3j_2j_1j_0i_{n-1} \longrightarrow j_{n-1}j_{n-2}j_{n-3} \dots j_2j_1j_0]$

ในการตรวจสอบความถูกต้องของเครือข่ายแบบบัตเตอร์ฟลายพลัสจะแสดงความสามารถจัดเส้นทางภายในของการเชื่อมต่อระหว่างสเตจได้สมบูรณ์โดยมีการเชื่อมต่อระหว่างสเตจของ $ISC_0 - ISC_{n-1}$ พร้อมการปรับสวิทซ์ดังต่อไปนี้

[ISC_0-sw: $i_{n-1}i_{n-2}i_{n-3} \dots i_3i_2i_1i_0 \longrightarrow i_{n-2}i_{n-3}i_{n-4} \dots i_2i_1i_0j_{n-1}$]

ISC_1-sw: $(i_{n-2}i_{n-3} \dots i_2i_1i_0j_{n-1}) \longrightarrow (j_{n-1}i_{n-3}i_{n-4} \dots i_2i_1i_0j_{n-2})$

ISC_2-sw: $j_{n-1}(i_{n-3}i_{n-4}...i_1i_0j_{n-2}) \longrightarrow j_{n-1}(j_{n-2}i_{n-4} \dots i_2i_1i_0j_{n-3})$

...

ISC_{n-2}-sw: $j_{n-1}j_{n-2}\ldots j_4j_3(\mathbf{i_1}i_0j_2) \longrightarrow j_{n-1}j_{n-2}\ldots j_4j_3(j_2i_0\mathbf{j_1})$

ISC_{n-1}-sw: $j_{n-1}j_{n-2}\ldots j_4j_3j_2(\mathbf{i_0}j_1) \longrightarrow j_{n-1}j_{n-2}\ldots j_4j_3j_2(j_1\mathbf{j_0})$

และสุดท้ายสำหรับเครือข่ายแบบอินเวอร์สเบสไลน์พลัสจะแสดงความสามารถจัดเส้นทางภายในของการเชื่อมต่อระหว่างสเตจได้สมบูรณ์และถูกต้องจากการเชื่อมต่อระหว่างสเตจของ $ISC_0 - ISC_{n-1}$ พร้อมการปรับสวิทช์ดังต่อไปนี้

[ISC_0-sw: $i_{n-1}i_{n-2}i_{n-3}\ldots i_3i_2i_1i_0 \longrightarrow i_0i_1i_2i_3\ldots i_{n-2}j_{n-1}$]

ISC_1-sw: $i_0i_1i_2i_3\ldots i_{n-4}i_{n-3}(\mathbf{i_{n-2}}j_{n-1}) \longrightarrow j_0i_1i_2i_3\ldots i_{n-4}i_{n-3}(j_{n-1}\mathbf{j_{n-2}})$

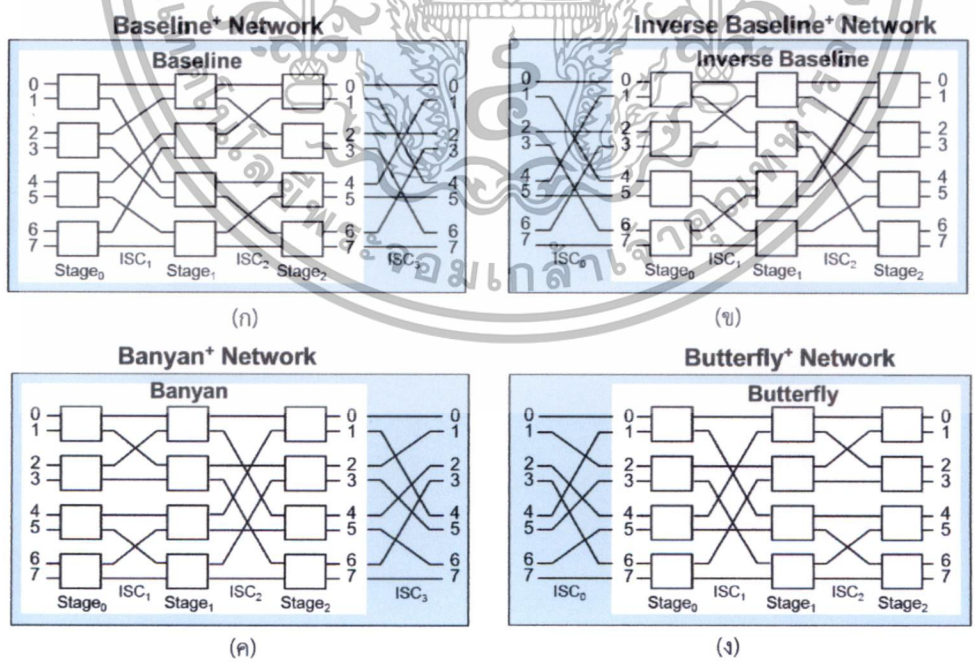
ISC_2-sw: $i_0i_1i_2i_3\ldots i_{n-4}(\mathbf{i_{n-3}}j_{n-1}j_{n-2}) \longrightarrow i_0i_1i_2i_3\ldots i_{n-4}(j_{n-1}j_{n-2}\mathbf{j_{n-3}})$

...

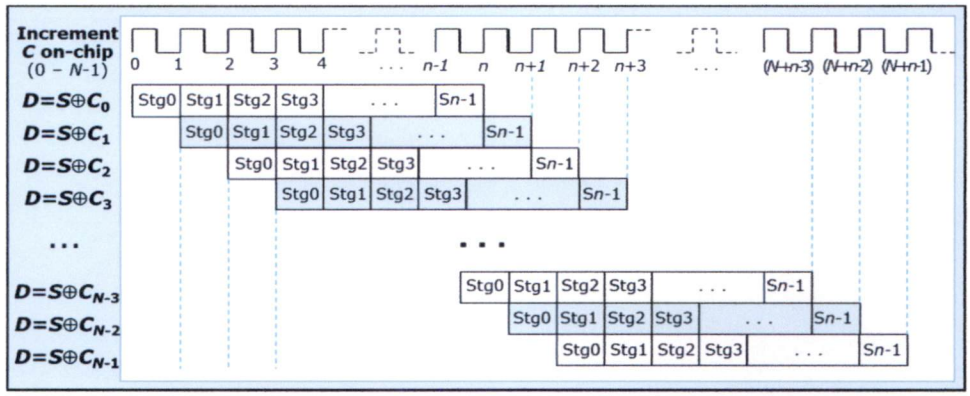
ISC_{n-2}-sw: $i_0(\mathbf{i_1}j_{n-1}j_{n-2}\ldots j_5j_4j_3j_2) \longrightarrow i_0(j_{n-1}j_{n-2}\ldots j_5j_4j_3j_2j_1)$

ISC_{n-1}-sw: $(\mathbf{i_0}j_{n-1}j_{n-2}\ldots j_5j_4j_3j_2j_1) \longrightarrow (j_{n-1}j_{n-2}\ldots j_5j_4j_3j_2j_1\mathbf{j_0})$

นอกจากนั้นได้ทดสอบความถูกต้องสำหรับการจัดเส้นทางบนเครือข่ายแบบ MINs$^+$ ขนาด N ต่างๆ ด้วยโปรแกรม (กำหนดให้มีหน่วยประมวลผลจำนวน $N = 16$ ถึง $N = 2^{20}$ หน่วยประมวลผล) โดยถ้าสวิทช์มีขนาด $d \times d$ ($d = 2^k$) จะใช้ตัวดำเนินการแบบบิตจำนวน k บิตสำหรับการปรับพอร์ตด้วย ISC_0 หรือ ISC_n บนเครือข่ายแบบ MINs$^+$ ที่มีการจัดเส้นทางภายในได้สมบูรณ์ (แสดงตัวอย่างดังรูปที่ 3.5 - 3.6)



รูปที่ 3.1 เครือข่ายแบบ MINs$^+$ (N=8) ประกอบด้วย เครือข่ายแบบ (ก) เบสไลน์พลัส (ข) อินเวอร์สเบสไลน์พลัส (ค) บันยานพลัส และ (ง) บัตเตอร์ฟลายพลัส



รูปที่ 3.2 แผนภาพแสดงช่วงเวลาของการติดต่อสื่อสาร ATAPE โดยใช้เทคนิคมัลติสเตจไปป์ไลน์โดยที่ตัวนับค่าสามารถเพิ่มขึ้นได้อัตโนมัติด้วยค่าควบคุมบนชิป (C_0, C_1, ... , C_{N-1})

3.2 ขั้นตอนวิธี ATAPE แบบฝังลงบนชิปบนเครือข่ายแบบ MINs$^+$

ขั้นตอนวิธี ATAPE แบบฝังลงบนชิปบนเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด d x d จะสามารถประมวลผลได้เร็วขึ้นตามความเร็วของสัญญาณนาฬิกาด้วย 2 ฟังก์ชันในขั้นตอนวิธีที่ 3.1 ซึ่งทำให้ปลายทาง (D) สามารถรับข้อมูลเฉพาะจากหน่วยประมวลผลต้นทาง (S) ในรอบที่ $r = 0$ จนถึง N-1 โดยใช้ตัวนับค่าบนชิป (Counter) สำหรับค่าควบคุม (Control) แต่ละบิต เมื่อ $C = r$ ดังนั้นรีจีสเตอร์ที่เป็นตัวนับค่าบนชิปจะเพิ่มค่าอัตโนมัติซึ่งจะเร็วกว่าการดึงคำสั่งเพิ่มค่า C จากหน่วยความจำ ในการนำขั้นตอนวิธี ATAPE ไปใช้บนเครือข่ายแบบ MINs$^+$ ซึ่งการเชื่อมต่อระหว่างต้นทางและปลายทาง (S, D Connections) จะใช้การเพิ่มค่าจากตัวนับ C ได้อย่างมีประสิทธิภาพใน N รอบที่ต่อเนื่องกัน นอกจากนั้นยังสามารถใช้ขั้นตอนวิธี ATAPE แบบฝังลงบนชิปได้เต็มประสิทธิภาพสำหรับการติดต่อสื่อสาร ATAPE บนเครือข่ายแบบ MINs$^+$ โดยใช้เทคนิคแบบมัลติสเตจไปป์ไลน์ที่สามารถเริ่มคำสั่งถัดไปได้ทันทีเมื่อผ่านไปแล้วหนึ่งสเตจ (ไม่ต้องรอให้เสร็จ n สเตจของแต่ละคำสั่งหรือแต่ละค่าของ C) ด้วยความซับซ้อนด้านเวลาที่เหมาะสมยิ่งขึ้น $O(N+\log_2 N)$ ได้พิสูจน์ในทฤษฎีบทที่ 3.1

ขั้นตอนวิธีที่ 3.1 ขั้นตอนวิธีการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์แบบฝังลงบนชิป (ATAPE Embedding) บนเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด d x d

1. **Incrementing the on-chip counter control** C *(=0 to N-1)*
   (incorporate with multistage pipelining, ดังรูปที่ 3.2)
2. **In parallel, all processors** S *(0 ≤ S < N)* ***do***
3. $D = S \oplus (C+order) \bmod N$ (default)
   or $D = \rho[(S+C+order) \bmod N]$ (option)
4. all processors S prepare and send their personalized massages to processors D (k-bit control tag per switch per stage.)
5. ***end*** **parallel processors**
6. ***end*** **the on-chip counter** C

Note: For routing the message, the control tag F (or B) = $S \oplus (C + order) \bmod N$ may be used as a header with the k-bit control, where $F = (f_{n-1}f_{n-2} \ldots f_1f_0)_2$ or $B = (b_{n-1}\, b_{n-2} \ldots b_1b_0)_2$ represents the destination processor D. Then, for the next stage, k bits of the remaining bits (in F or B) is discarded by each successive switch to use the first k-bit element in the control tag across $m = \log_d N$ stages, such as forward MS k-bit for omega and butterfly$^+$ networks and backward LS k-bit for other MINs$^+$ (i.e., flip, cube, banyan$^+$, and baseline$^+$ networks).

<u>ทฤษฎีบทที่ 3.1</u> ขั้นตอนวิธี ATAPE แบบฝังลงบนชิป ด้วยความซับซ้อนของเวลาแบบดีที่สุดคือ $O(N+\log_2 N)$

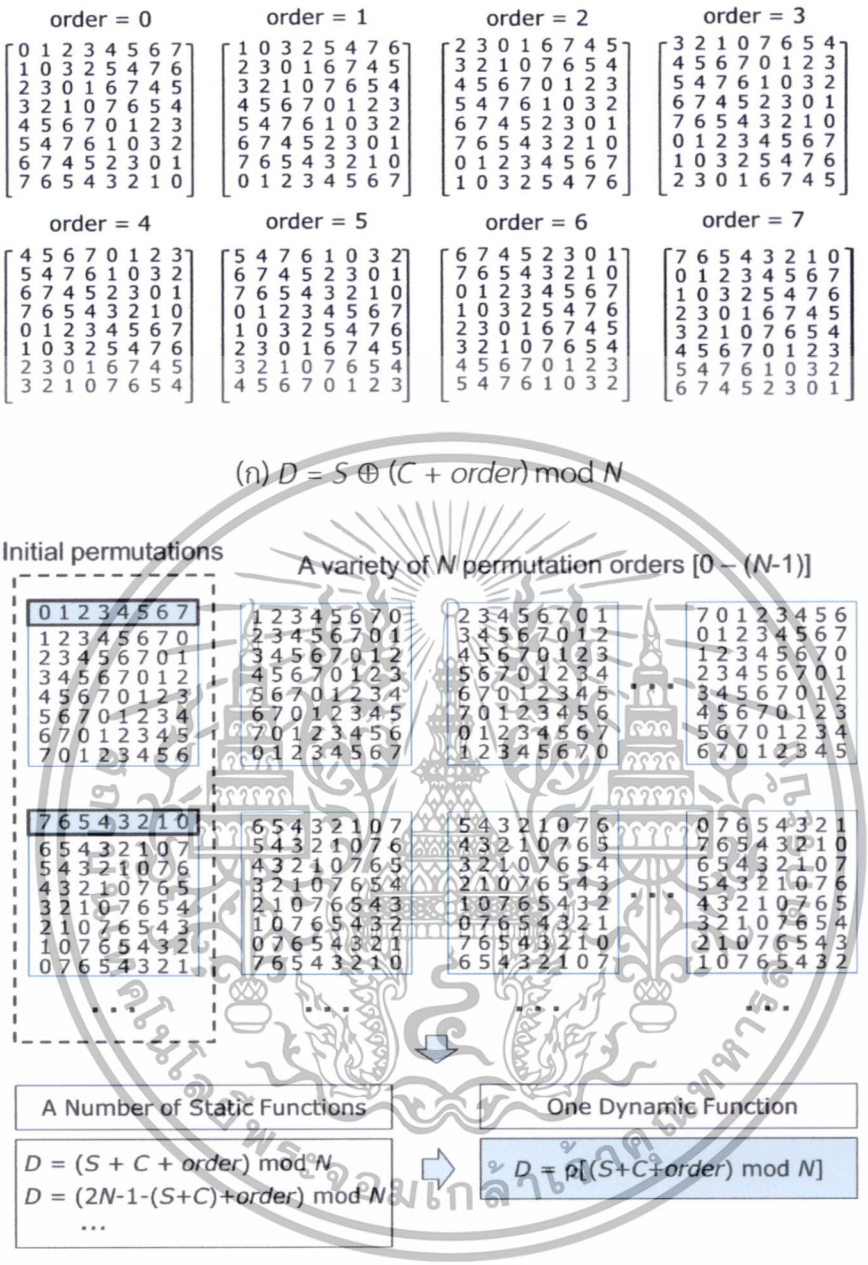
พิสูจน์ การรับข้อมูลเฉพาะข้อมูลแรกในรอบแรกของแต่ละ N หน่วยประมวลผล โดยเริ่มที่ตัวนับของค่าควบคุมหลัก $C = 0$ ผ่าน $n = \log_2 N$ สเตจในเวลา n หน่วยเวลา (สมมุติว่า 1 สัญญาณนาฬิกา (Clock) ต่อหนึ่งหน่วยเวลาสำหรับการประมวลผลบนชิป) และรับข้อมูลเฉพาะที่เหลือทั้งหมดในเวลา N-1 สัญญาณนาฬิกา (1 สัญญาณนาฬิกาต่อรอบ) โดยใช้เทคนิคแบบมัลติสเตจไปป์ไลน์เพื่อให้แต่ละสเตจทำงานต่อเนื่องกันดังตัวอย่างรูปที่ 3.2 ในการประมวลผล N รอบต่อเนื่องกันของ $C_0, C_1, C_2, \ldots, C_{N-1}$ ด้วยเทคนิคมัลติสเตจไปป์ไลน์ ซึ่งเป็นการที่ในรอบถัดไป (C_{i+1}) สามารถเริ่มทำการประมวลผลได้ในสเตจ 0 เมื่อรอบก่อน (C_i) ประมวลผลผ่านสเตจ 0 แล้ว โดยไม่ต้องรอให้ C_i ประมวลผลเสร็จผ่านทุกสเตจ และทำต่อเนื่องในทำนองเดียวกันไปในสเตจอื่นๆ ถัดไป โดยใช้ค่าที่เพิ่มขึ้นของตัวนับค่าที่อยู่ในรีจีสเตอร์ทำให้ได้ค่าตัวนับ C ที่จะมีประสิทธิภาพมากกว่าและเร็วกว่า วิธีการประมวลผลแบบนี้เรียกว่า เทคนิคแบบมัลติเสตไปป์ไลน์แบบเต็มประสิทธิภาพ (Fully Multistage Pipelining) เพราะใช้รีจีสเตอร์เป็นหลักในการประมวลผลการเพิ่มค่าตัวนับ C ที่จะมีประสิทธิภาพที่เร็วกว่าการดึง N คำสั่งหรือ N ค่าของ C จากหน่วยความจำของขั้นตอนวิธี ATAPE แบบเดิม นั่นคือสมมุติว่าเวลาในการเข้าถึงข้อมูลในหน่วยความจำ แต่ละครั้งเท่ากับค่า c ซึ่งโดยปกติทั่วไปจะใช้เวลามากกว่า 1 สัญญาณนาฬิกา ดังนั้นขั้นตอนวิธี ATAPE แบบนี้สามารถประมวลผลด้วยความซับซ้อนด้านเวลาที่เหมาะสม $O(N+\log_2 N)$ นั่นคือทุกๆ ข้อมูลเฉพาะสามารถส่งได้ครบ N รอบภายใน $N+n$-1 สัญญาณนาฬิกา (เช่นใช้ n สัญญาณนาฬิกาในการประมวลผลรอบแรกที่ทำการส่งข้อมูลผ่าน n สเตจบวกกับ N-1 สัญญาณนาฬิกาสำหรับการประมวลผล N-1 รอบที่เหลือ) แต่ขั้นตอนวิธี ATAPE แบบเดิมจะประมวลผลเสร็จในเวลามากกว่า เท่ากับ $cN+n$-1 สัญญาณนาฬิกา □

3.2.1 วิธีเรียงสับเปลี่ยนที่เป็นไปได้ในตารางลาติน

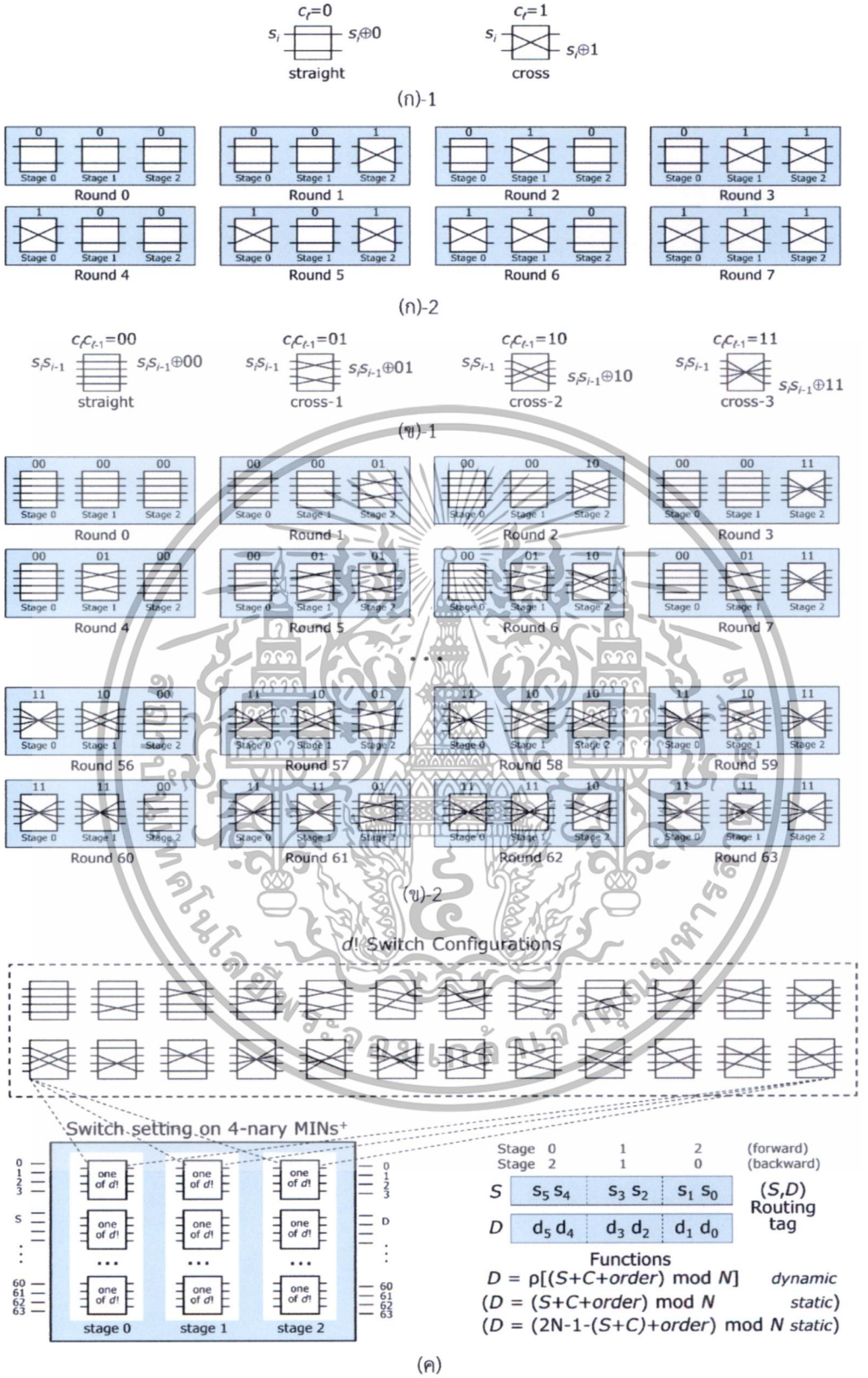
โดยทั่วไปฟังก์ชัน ATAPE หนึ่งฟังก์ชันสามารถสร้างตารางลาติน (Latin-Square : LS) ของวิธีเรียงสับเปลี่ยนที่เป็นไปได้เพียงตารางลาตินเดียวเท่านั้นซึ่งทำให้ไม่ยืดหยุ่น ดังนั้นเพื่อที่จะสามารถจัดเส้นทางจากต้นทางไปยังปลายทางได้ยืดหยุ่นมากยิ่งขึ้น (เช่น สามารถเลือกค่าเริ่มต้นที่ต้องการได้) โดยที่ไม่จำเป็นต้องมีการสร้างตารางลาตินไว้ก่อนที่จะทำการจัดเส้นทางจึงได้เสนอความยืดหยุ่นของการเตรียมตารางลาตินไว้หลากหลายด้วยฟังก์ชัน ATAPE แบบฝังลงบนชิป 2 ฟังก์ชันดังนี้

1) ฟังก์ชันแบบคงที่ (Static ATAPE Function) $D = S \oplus (C+order) \bmod N$ ซึ่งเตรียมลำดับที่ยืดหยุ่น (Order) N ลำดับสำหรับเริ่มต้นส่งข้อมูลเฉพาะได้ N ทางเลือก
2) ฟังก์ชันแบบปรับเปลี่ยนได้ (Dynamic ATAPE Function) $D=\rho[(S+C+order) \bmod N]$ ซึ่งสามารถเปลี่ยนแปลงได้ยืดหยุ่นตามความต้องการของผู้ใช้โดยผู้ใช้สามารถโปรแกรมได้ระหว่างใช้งาน

สำหรับฟังก์ชัน ATAPE แบบฝังลงบนชิปนั้นในทุกๆ การเชื่อมต่อระหว่างต้นทางและปลายทางสามารถเลือกลำดับใดๆ ก่อนที่จะทำการส่งข้อมูลได้อัตโนมัติ รูปที่ 3.3 (ก) แสดงผลลัพธ์ของปลายทาง ($D = S \oplus (C+order) \bmod N$) กำหนดให้เครือข่าย $N = 8$ ดังนั้นมีการสลับลำดับ 8 ลำดับในตารางลาตินขนาด NxN โดยที่แต่ละตารางลาตินสามารถนำไปใช้ได้ในทุกๆ ชนิดของเครือข่ายแบบ MINs$^+$ ดังตัวอย่างในรูปที่ 3.9



รูปที่ 3.3 ผลลัพธ์ในตารางลาตินหรือวิธีเรียงสับเปลี่ยน (Permutations) ของการใช้ฟังก์ชัน ATAPE แบบฝังลงบนชิปสำหรับการจัดเส้นทางภายในระหว่างต้นทางและปลายทาง (N=8) โดยที่ *Orders* = 0 ถึง N-1



รูปที่ 3.4 การกำหนดค่าเครือข่าย N ค่า (ก) ถ้า c_t = 0 หรือ 1 (แบบของสวิทช์ขนาด 2x2 จำนวน 2 แบบ) สำหรับ N=8 (ข) ถ้า c_tc_{t-1}=00 ถึง 11 (แบบของสวิทช์ขนาด 4x4 จำนวน 4 แบบ) สำหรับ N=64 และ (ค) ทุกๆ แบบที่เป็นไปได้ของสวิทช์จำนวน $d!$ แบบ (d=4)

นอกจากนั้นในการติดต่อสื่อสาร ATAPE ที่มีความยืดหยุ่นมากยิ่งขึ้น จึงเสนอฟังก์ชัน ATAPE แบบปรับเปลี่ยนได้ $D = \rho[(S+C+order) \bmod N]$ ที่อนุญาติให้ผู้ใช้สามารถโปรแกรมได้โดยการกำหนดค่าวิธีเรียงสับเปลี่ยนเริ่มต้นที่แตกต่างเพื่อส่งข้อมูลเฉพาะ โดยฟังก์ชันเชิงทฤษฎี f-in-1 ได้เตรียมวิธีเรียงสับเปลี่ยนเริ่มต้น f วิธี ($\rho[i]$, i = 0, 1,..., N-1) ของตารางลาติน fN ตาราง รูปที่ 3.3 (ข) แสดงผลลัพธ์ของตารางลาตินสำหรับกำหนดวิธีเรียงสับเปลี่ยนเริ่มต้น 2 วิธีของฟังก์ชันแบบปรับเปลี่ยนได้ $D = \rho[(S+C+order) \bmod N]$ โดยยกตัวอย่างฟังก์ชันแบบคงที่ 2 ฟังก์ชันดังนี้

1) $\rho[0..N-1] = (0, 1, 2, \ldots, N-1)$ ของ $D = (S+C+order) \bmod N$

2) $\rho[0..N-1] = (N-1, N-2, \ldots, 2, 1, 0)$ ของ $D = (2N-1-(S+C)+order) \bmod N$

ซึ่งฟังก์ชัน ATAPE ที่มีความยืดหยุ่นดังกล่าวจะใช้การตั้งค่าแบบสวิทซ์ในทุกๆ สวิทซ์หลากหลายเป็นจำนวน $d!$ ค่า (แสดงในรูปที่ 3.4 (ค)) เทียบกับที่กำหนดได้อย่างชัดเจน การเชื่อมต่อภายในสวิทซ์โดยใช้ตัวดำเนินการแบบ XOR ซึ่งมีแบบของการเชื่อมต่อภายในสวิทซ์ d แบบ (แสดงในรูปที่ 3.4 (ก-ข))

3.2.2 วิธีเรียงสับเปลี่ยนที่เป็นไปได้ของขั้นตอนวิธี ATAPE โดยใช้การตั้งค่าเครื่องข่าย N ค่าบนเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทซ์ขนาด $d \times d$

สำหรับการใช้ฟังก์ชัน $D = S \oplus (C + order) \bmod N$ จะสามารถจัดเส้นทางภายในสำหรับส่งข้อมูลแบบ ATAPE บนเครือข่ายแบบ MINs$^+$ ได้ 3 วิธีดังนี้

1) การจัดเส้นทางแบบค่าควบคุมหลักของสเตจ (Stage-Control Routing)
2) การจัดเส้นทางแบบค่าควบคุมหลักของสวิทซ์ (Switch-Control Routing)
3) การจัดเส้นทางภายในได้สมบูรณ์จากต้นทางไปยังปลายทาง (Full (S, D) Self-Routing)

การจัดเส้นทางแบบแรกจะสามารถทำได้เช่นเดียวกับวิธีการเดิมของ Yang และ Wang [16] คือจะสามารถจัดเส้นทางโดยใช้วิธีการแบบค่าควบคุมแบบสเตจในแต่ละสวิทซ์บนเครือข่ายแบบ MINs$^+$ ซึ่งการตั้งค่าเครือข่ายสามารถจัดเส้นทางด้วยวิธีเรียงสับเปลี่ยนผ่าน ได้ในรอบเดียว(Single-Pass Permutation) โดยที่จะไม่เกิดการขัดแย้งกันภายในสวิทซ์ (Non-Conflict Switch) ในการคำนวณเส้นทางของตารางลาตินจะใช้บิตค่าควบคุมแบบสเตจเป็นค่าไบนารี่จำนวน n บิต ($c_{n-1}\, c_{n-2}\, c_{n-3} \ldots c_1\, c_0)_2$ และเพื่อให้ง่ายในการคำนวณและอ้างอิงจะใช้เลขฐานสิบคือ C โดยที่ $0 \le C < 2^n$ สำหรับวิธีเรียงสับเปลี่ยนแบบส่งรอบเดียวผ่าน (Single-Pass Permutation) จะทำการจับคู่วิธีเรียงสับเปลี่ยนแบบหนึ่งต่อหนึ่ง $\rho = (a_0\, a_1 \ldots a_i \ldots a_{N-1})$ ระหว่างต้นทาง S และปลายทาง D หรือ $\rho(S)$ ของเครือข่ายแบบ MINs$^+$ โดยที่ $\rho(S) = D \in \{0, 1, \ldots, N-1\}$ สำหรับทุกค่า S = 0, 1, ... , N-1

ส่วนการจัดเส้นทางภายในจากต้นทางไปยังปลายทาง เพื่อให้เชื่อมต่อระหว่างต้นทางและปลายทางเป็นการจัดเส้นทางเพื่อส่งข้อมูลจากหน่วยประมวลผล S ไปยังหน่วยประมวลผล D บนเครือข่ายแบบ MINs$^+$ เช่นการจัดเส้นทางการเชื่อมต่อโดยตลอดเส้นทางผ่าน n สเตจสามารถกำหนดโดยลำดับของค่าควบคุมบิตของ S และ D จำนวน n บิต เพื่อใช้ปรับสวิทซ์ที่ต่อเนื่องกัน กำหนดให้ $\oplus$ แทนตัวดำเนินการแบบ XOR แบบบิตดังนี้ $0 \oplus 0 = 0$ $0 \oplus 1 = 1$ $1 \oplus 0 = 1$ $1 \oplus 1 = 0$) สำหรับต้นทาง $S = (s_{n-1}s_{n-2} \ldots s_2s_1s_0)_2$ และค่าควบคุมหลัก $C = (c_{n-1}\, c_{n-2} \ldots c_2c_1c_0)_2$ ดังนั้น $D = (d_{n-1}d_{n-2} \ldots d_2d_1d_0)_2 = S \oplus C = (s_{n-1} \oplus c_{n-1}\ s_{n-2} \oplus c_{n-2} \ldots s_1 \oplus c_1\ s_0 \oplus c_0)_2$ และใช้ตัวนับของค่าควบคุมหลัก C โดยการเพิ่มขึ้นอัตโนมัติจาก 0 จนถึง N-1 สำหรับค่า ที่แตกต่างใดๆ จำนวน N ค่าบนเครือข่ายแบบ MINs$^+$

เมื่อต้องการใช้ค่าควบคุมหลักของสเตจถ้ากำหนดให้ R(N) แทนจำนวนรอบทั้งหมดที่ต้องการกำหนดค่าให้กับเครือข่ายเพื่อติดต่อสื่อสารกันแบบออล-ทู-ออลบนเครือข่ายแบบ MINs$^+$ ดังพิสูจน์ใน

ทฤษฎีบทที่ 3.2-3.3 กำหนดให้รอบ r (โดยที่ $0 \le r < N$) แทนรอบในการส่งทุกๆ ข้อมูลเฉพาะ N ข้อมูลจาก N ต้นทางไปยัง N ปลายทางโดยใช้ตัวดำเนินการแบบบิตของค่าควบคุมหลัก $C = (c_{n-1}c_{n-2} \dots c_1c_0)_2$ เพื่อกำหนดค่าสวิทช์ในแต่ละสเตจโดยที่ทุกๆ ค่าไบนารี่ของค่าควบคุมที่แตกต่าง N ค่าที่สามารถเพิ่มขึ้นโดยใช้ตัวนับค่าที่อยู่ภายในชิป (On-Chip Counter) ประกอบด้วย $000\dots00_2$, $000\dots01_2$, $000\dots10_2$, ... , $111\dots10_2$, $111\dots11_2$

ในหัวข้อนี้จะเน้นที่การกำหนดให้วิธีเรียงสับเปลี่ยนของตารางลาตินทั่วไปสามารถใช้ฟังก์ชัน $D = S \oplus C$ ได้อย่างเต็มประสิทธิภาพ โดยที่ค่าควบคุม N ค่าจะได้วิธีเรียงสับเปลี่ยน N วิธีที่เพิ่มขึ้นด้วยตัวนับ C= 0 (000_2), 1 (001_2), 2 (010_2), ... , N-1 (111_2) สำหรับ N = 8 ดังภาพที่ 3.3 (ก) ซึ่งใช้ *order* = 0 โดยทั่วไปฟังก์ชันแบบฝังจะไม่ยืดหยุ่นและไม่สามารถแก้ไขได้ ดังนั้นจึงได้เสนอฟังก์ชันแบบคงที่ (Static ATAPE) $D = S \oplus (C + order) \bmod N$ ซึ่งจะเตรียมลำดับ (*order*) ที่ยืดหยุ่นมากกว่าของวิธีเรียงสับเปลี่ยนในตารางลาติน (ดังภาพที่ 3.3 (ก) *order* 0-7) ที่สามารถโปรแกรมปรับค่าได้ระหว่างใช้งานซึ่งลำดับมีค่าอยู่ระหว่าง 0 จนถึง N-1 เมื่อกำหนด *order* = 0 จะหมายถึง $D = S \oplus C$ (ในการสร้างวิธีเรียงสับเปลี่ยนของตารางลาตินทั่วไป) ส่วน *order* = 1 จะหมายถึงการสลับลำดับ ในแถวแรกของตารางลาตินใหม่คือเริ่มต้นแถวที่สองของตารางลาตินทั่วไป และวนสลับลำดับไปเรื่อยๆ ยกตัวอย่างเช่น ตารางลาตินกับ *order* = 2 จะเริ่มในแถวแรกของการเชื่อมต่อจากต้นทางไปยังปลายทางโดยวิธีเรียงสับเปลี่ยน [2 3 0 1 6 7 4 5] คือ (0,2), (1,3), (2,0), (3,1), (4,6), (5,7), (6,4) และ (7,5) ได้โดยอัตโนมัติ

หมายเหตุ แม้ว่าตารางลาตินจำนวน N ตารางจะเตรียมลำดับที่ยืดหยุ่น N ลำดับ (แต่ละลำดับจะใช้ค่าควบคุมที่เหมือนกันต่อเนื่องกันไป N ค่าโดยที่ C = 0, 1, 2, ..., N-1) แต่จะมีเพียงตารางลาติน (LS) หนึ่งตารางจะถูกประยุกต์ใช้ในขณะเวลาหนึ่งๆ บนเครือข่ายแบบ MINs$^+$ โดยไม่ต้องทำการสร้างตารางลาตินไว้ก่อน (LS-Preprocessing)

ทฤษฎีบทที่ 3.2 ในแต่ละเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด dxd กำหนดให้จำนวนของการกำหนดค่าเครือข่ายเพื่อการติดต่อสื่อสารแบบ ATAPE โดยใช้ค่าควบคุมใดๆ คือ $R(N)=N=2^n$

พิสูจน์ การกำหนดค่าเครือข่ายเพื่อการติดต่อสื่อสารแบบ ATAPE โดยใช้ค่าควบคุมหลัก $C = (c_{n-1}c_{n-2} \dots c_tc_{t+1} \dots c_1c_0)_2$ N ค่า (C = 0, 1, 2, ..., N-1) ในทุกๆ การจัดเส้นทางที่แตกต่าง N เส้นทางจะเริ่มจากทุกๆ หน่วยประมวลผลต้นทาง (S = 0, 1, 2, 3, ... , N-1) ส่งข้อมูลไปยังหน่วยประมวลผลปลายทาง (D) ด้วยวิธีเรียงสับเปลี่ยน $\rho = (a_0\ a_1 \dots a_i \dots a_{N-1})$ โดยที่ผลลัพธ์ของตัวดำเนินการแบบ XOR แบบบิตของฟังก์ชัน $D = S \oplus C$ ในการกำหนดค่าเครือข่ายแต่ละค่าสำหรับ ATAPE (ทุกๆ การเชื่อมต่อระหว่างต้นทางและปลายทางจะถูกกำหนดตามค่าควบคุมหลัก C (โดยที่ $0 \le C < 2^n$)) เพื่อส่งข้อมูลเฉพาะ N ข้อมูลไปยัง N หน่วยประมวลผลที่เกี่ยวข้องในการสร้างวิธีเรียงสับเปลี่ยนแบบส่งผ่านรอบเดียว (Single-Pass Permutations) ของปลายทาง (D) ในตารางลาตินทั่วไปจำนวน N วิธี เพื่อจัดเส้นทางภายในโดยใช้ตัวนับ n บิตคือ $C = (c_{n-1}c_{n-2} \dots c_1c_0)_2$ ในทุกๆ วิธีเรียงสับเปลี่ยนที่แตกต่างของค่าควบคุมหลักจำนวน N วิธี (เช่น 0000...00, 0000 ... 01, 0000 ... 10, 0000 ... 11, ... , 1111 ... 10, และ 1111 ... 11) ตัวอย่างในรูปที่ 3.4 (ก-ข) แสดงแบบของการกำหนดค่าเครือข่าย N ค่าเมื่อ N = 16 และ d = 2 (ค่าควบคุม ($C = c_2c_1c_0$) 3 บิตถูกนำไปใช้โดยแบ่งทีละ 1 บิตสำหรับในแต่ละสวิทช์ (ขนาด 2×2) ของแต่ละสเตจ) และเมื่อ N = 64 และ d = 4 (ค่าควบคุม ($C = c_5c_4c_3c_2c_1c_0$) 4 บิตถูกนำไปใช้โดยแบ่งทีละ 2 บิตสำหรับในแต่ละ สวิทช์ (ขนาด

4X4) ของแต่ละสเตจ) ดังนั้นการกำหนดค่าเครือข่ายในการติดต่อสื่อสาร ATAPE แต่ละค่าจะใช้หนึ่งค่าควบคุมต่อหนึ่งรอบโดยที่ $r = C$ เพื่อส่งข้อมูลเฉพาะในแบบขนาน N ข้อมูลจาก N หน่วยประมวลผลต้นทาง (S) ไปยังหน่วยประมวลผลปลายทาง (D) สำหรับทุกๆ ต้นทาง (S = 0, 1, 2, 3, ..., N-1) และใช้ค่าควบคุมหลัก C ของวิธีเรียงสับเปลี่ยนเรียงลำดับจากรอบแรกแถวแรกในตารางลาตินคือ $D = S \oplus$ 0000...00 รอบถัดไปในตารางลาตินแถวถัดไป $D = S \oplus$ 0000...01 และรอบอื่นๆ ในตารางลาตินแถวอื่นๆ เรียงตามลำดับ โดยวิธีเรียงสับเปลี่ยนที่แตกต่างกันอื่นๆ ($D = S \oplus C$) นั่นคือผลลัพธ์ของตารางลาติน (ในรูปที่ 3.3) จะใช้แทนการกำหนดค่าเครือข่าย N ค่าของทุกๆ การเชื่อมต่อจากต้นทางไปยังปลายทางสำหรับวิธีการส่งข้อมูลแบบ ATAPE ท้ายที่สุดแล้วการกำหนดค่าเครือข่ายแต่ละค่าสามารถนำวิธีเรียงสับเปลี่ยนที่เป็นไปได้ที่แตกต่างกัน N ค่าไปใช้ได้สมบูรณ์บนทุกๆ เครือข่ายแบบ MINs$^+$ สำหรับการประยุกต์ใช้ลำดับอื่นๆ (*order* = 1 ถึง N-1) แสดงในรูปที่ 3.3 (ก) จากฟังก์ชัน $D = S \oplus (C + order)$ mod N ส่วนกรณีฟังก์ชันปรับเปลี่ยนได้แบบ f-in-1 คือฟังก์ชัน $D = \rho$ [$(S + C + order)$ mod N] ที่เตรียมตารางลาตินอื่นๆ fN ตาราง (รูปที่ 3.3 (ข)) ที่อนุญาตให้ผู้ใช้โปรแกรมกำหนดวิธีเรียงสับเปลี่ยนเริ่มต้นของฟังก์ชัน ATAPE ได้ โดยกรณีนี้แบบของสวิทช์ $d!$ แบบแสดงในรูปที่ 3.4 (ค) เป็นการกำหนดค่าเครือข่ายที่ยืดหยุ่นโดยใช้ฟังก์ชัน ATAPE แบบปรับเปลี่ยนได้ตามความต้องการของผู้ใช้ โดยวิธีเรียงสับเปลี่ยนในแต่ละรอบของฟังก์ชัน ATAPE N รอบคือส่วนหนึ่งของวิธีเรียงสับเปลี่ยนที่เป็นไปได้บนแต่ละ MIN$^+$ จำนวน $(d!)^{N/d \times \log_d N}$ วิธีเพื่อให้สามารถติดต่อสื่อสารแบบจุดต่อจุดพร้อมๆ กัน (Parallel Point-to-Point) เมื่อแต่ละเส้นทางการเชื่อมต่อแบบ ATAPE แต่ละรอบทำได้ด้วยการกำหนดค่า k บิตของต้นทางและปลายทาง (S-D) ในแต่ละสวิทช์ขนาด dxd สามารถจับคู่กันได้แบบหนึ่งต่อหนึ่ง (One-to-One Mapping) ตลอดเส้นทาง ดังนั้นในทุกๆ ปลายทาง (Ds) ที่แตกต่างสามารถใช้ค่าควบคุมหลัก C และลำดับ (*order*) เดียวกันโดยที่ไม่เกิดสวิทช์ที่ขัดแย้งกัน สำหรับ $0 \le Ss < N$ ได้พร้อมๆ กัน □

<u>ทฤษฎีบทที่ 3.3</u> สำหรับเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด d x d ($N = 2^n$, $d = 2^k$ และ $m = \log_d N$) ในการกำหนดค่าเครือข่ายในแต่ละค่าจะสามารถเพิ่มขึ้นได้ด้วยตัวนับบนชิป $C = (c_{n-1}c_{n-2} \ldots c_1c_0)_2$ และในต้นทาง S (= 0, 1, 2,..., N-1) จะใช้ k บิตสำหรับการกำหนดค่าควบคุมภายในของแต่ละสวิทช์ในแต่ละสเตจเพื่อทำการเชื่อมต่อไปยังปลายทาง D โดยสามารถเลือกใช้ 2 ฟังก์ชันคือ (3.1) หรือ (3.2)

$$F \text{ (หรือ } B) = S \oplus (C + order) \bmod N \qquad \text{(default)} \quad \text{— (3.1)}$$

หรือ $$F \text{ (หรือ } B) = \rho[(S + C + order) \bmod N] \qquad \text{(option)} \quad \text{— (3.2)}$$

โดยที่ F (Forward Control Tag) หมายถึงค่าควบคุมแบบไปข้างหน้า และ B (Backward Control Tag) หมายถึงค่าควบคุมแบบย้อนกลับ

C (On-Chip Control) หมายถึงค่าควบคุมบนชิป = 0, 1, 2,..., 2^n-1 และ $0 \le order < 2^n$

พิสูจน์ การจัดเส้นทางภายในของเครือข่ายแบบ MINs$^+$ จะสามารถจัดเส้นทางตามกฎ 2 ข้อดังนี้

1) การจัดเส้นทางโดยเริ่มใช้บิตที่มีนัยสำคัญสูงสุดก่อน (Most Significant Bit: MSb) สำหรับการนำไปใช้บนเครือข่ายการจัดเส้นทางแบบไปข้างหน้า (Forward Routing Networks) เช่นเครือข่ายแบบโอเมก้า เครือข่ายแบบบัตเตอร์ฟลายพลัสเป็นต้น เรียกว่าค่าควบคุมแบบไปข้างหน้า (Forward Control Tag)

2) การจัดเส้นทางโดยเริ่มใช้บิตที่มีนัยสำคัญต่ำสุดก่อน (Least Significant Bit: LSb) สำหรับการนำไปใช้บนเครือข่ายการจัดเส้นทางแบบย้อนกลับ (Backward Routing Networks) เช่นเครือข่ายแบบฟลิบ เครือข่ายแบบบันยานพลัส และเครือข่ายแบบเบสไลน์พลัส เรียกว่าค่าควบคุมแบบย้อนกลับ (Backward Control tag)

ดังนั้นค่าควบคุมแบบไปข้างหน้า (ในรูปที่ 3.5 (ก) สำหรับสวิทช์ขนาด 4 x 4) สเตจที่ 0 จะเริ่มใช้บิตที่มีนัยสำคัญสูง 2 บิตของค่าควบคุม C ($c_{n-1}c_{n-2}$) สเตจถัดไปก็ใช้บิตต่อไปทีละ 2 บิตไปเรื่อยๆ จนกระทั่งสเตจที่ m-1 จะใช้บิตที่มีนัยสำคัญต่ำ 2 บิตของค่าควบคุม C (c_1c_0) นั่นคือใช้ตำแหน่งบิตที่มีนัยสำคัญสูงไปยังตำแหน่งบิตที่มีนัยสำคัญต่ำในสเตจที่ 0 จนถึงสเตจที่ m-1 จำนวน $m=\log_d N$ สเตจ ในทางตรงข้ามเครือข่ายการจัดเส้นทางแบบย้อนกลับ (ในรูปที่ 3.5 (ข) สำหรับสวิทช์ขนาด 4 × 4) สเตจที่ 0 จะใช้ 2 บิตที่มีนัยสำคัญต่ำของค่าควบคุม C (c_1c_0) ก่อนทีละ 2 บิตไปเรื่อยๆ จนกระทั่งถึงสเตจที่ m-1 จึงจะใช้ 2 บิตที่มีนัยสำคัญสูงของค่าควบคุม C ($c_{n-1}c_{n-2}$) นั่นคือใช้ตำแหน่งของบิตที่มีนัยสำคัญน้อยไปยังตำแหน่งบิตที่มีนัยสำคัญสูงสำหรับสเตจที่ 0 จนถึงสเตจที่ m-1 โดยเฉพาะอย่างยิ่งถ้า $order = 0$ สำหรับ $D = S \oplus C$ จะใช้ค่าควบคุมแบบไปข้างหน้า $F = S \oplus C$ หรือค่าควบคุมแบบย้อนกลับ $B = S \oplus C$ ตามวิธีเรียงสับเปลี่ยนในตารางลาตินของปลายทาง ($D = S \oplus (C + order) \bmod N$ หรือ $D = \rho[(S+C+order) \bmod N]$) สำหรับฟังก์ชัน ATAPE บนเครือข่ายแบบ MINs$^+$ ($N=2^n$, $d=2^k$) ส่วนในกรณีของสวิทช์ขนาด $d \times d$ จะทำการแบ่ง k-บิตต่อสวิทช์ในแต่ละสเตจของต้นทางและปลายทาง (ของ S และ D) ผ่านสวิทช์ในแต่ละสเตจจำนวน $\log_d N$ สเตจ □

3.2.3 การจัดเส้นทางโดยใช้ค่าควบคุมแบบสเตจจำนวน k บิตบนเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด $d \times d$

ในการกำหนดค่าเครือข่ายโดยใช้สวิทช์ขนาด $d \times d$ กำหนดให้ C เป็นค่าควบคุมสำหรับแต่ละสเตจ (Stage-Control) ด้วยค่าเลขฐานสองดังนี้ $(c_{n-1}c_{n-2} \ldots c_tc_{t-1} \ldots c_3c_2\, c_1c_0)_2$ โดยที่ $0 \le C < 2^n$ ดังนั้นการกำหนดค่าเครือข่าย N ค่าโดยฟังก์ชัน ATAPE แบบแรกด้วยตัวดำเนินการแบบ XOR ($\oplus$) จะสามารถทำได้ด้วยการกำหนดสวิทช์ในแต่ละสเตจโดยใช้แบบของสวิทช์จำนวน d แบบ เมื่อกำหนดให้ $c_t\, c_{t-1}$ ใช้แทนค่าควบคุมจำนวน 2 บิตจาก $C = (c_{n-1}c_{n-2}\, c_{n-3}\, c_{n-4} \ldots c_tc_{t-1} \ldots c_1c_0)_2$ จากสเตจที่ 0 ไปยังสเตจที่ m-1 ของสวิทช์ขนาด 4x4 (d=4, k=2) และการกำหนดค่าควบคุมแบบสเตจสำหรับเชื่อมต่อระหว่างต้นทางและปลายทาง (ดังรูปที่ 3.4 (ข)) สามารถใช้ค่าควบคุมจำนวน 2 บิตในแต่ละสเตจดังนี้

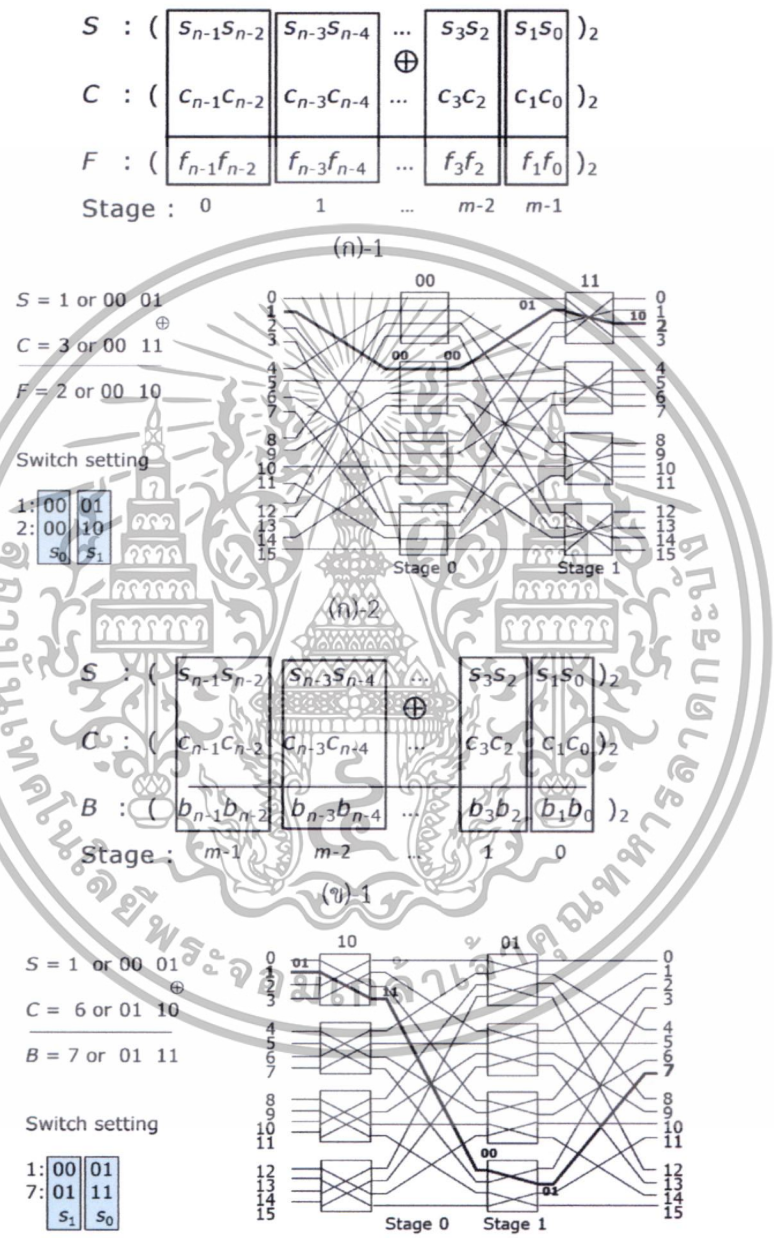
$c_t\, c_{t-1}$ = 00 หมายถึงการเชื่อมต่อภายในสวิทช์แบบตรงทุกๆ พอร์ต (Straight) ในสเตจนั้นๆ

$c_t\, c_{t-1}$ = 01 หมายถึงการเชื่อมต่อภายในสวิทช์แบบครอสส์แบบ 1 พอร์ต (Cross-1) ในสเตจนั้นๆ

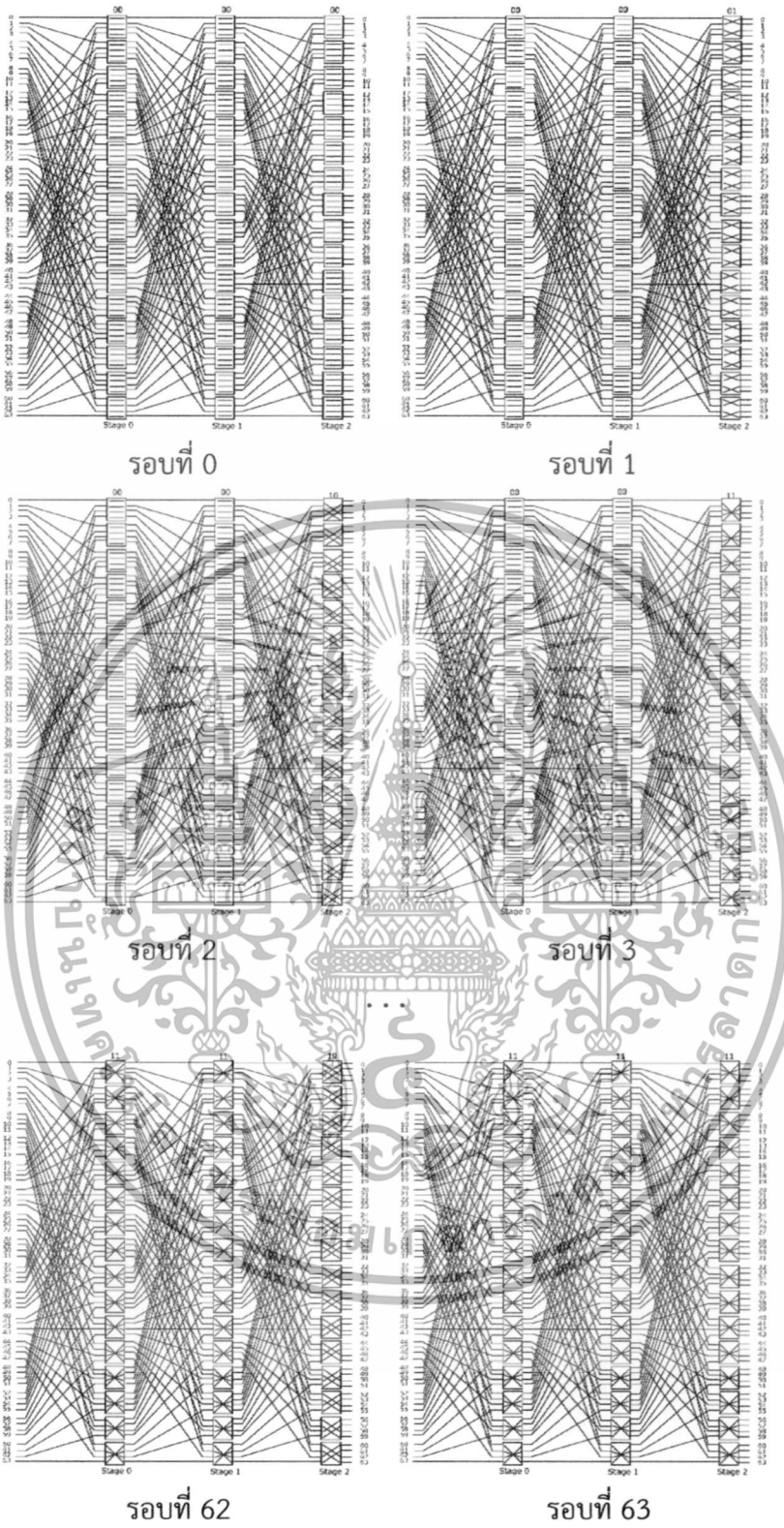
$c_t\, c_{t-1}$ = 10 หมายถึงการเชื่อมต่อภายในสวิทช์แบบครอสส์แบบ 2 พอร์ต (Cross-2) ในสเตจนั้นๆ

$c_t\, c_{t-1}$ = 11 หมายถึงการเชื่อมต่อภายในสวิทช์แบบครอสส์แบบ 3 พอร์ต (Cross-3) ในสเตจนั้นๆ

การจัดเส้นทางโดยใช้วิธีการค่าควบคุมแบบสเตจจำนวน k บิตดังกล่าวสามารถนำไปใช้ได้ทั้งบนเครือข่ายแบบไปข้างหน้าและเครือข่ายแบบย้อนกลับผ่านสเตจจำนวน $\log_d N$ สเตจดังตัวอย่างในรูปที่ 3.4 (ข) การกำหนดค่าภายในของสวิทช์จำนวน N ค่า (N=64, n=6, m=3, d=4 และ k=2) และรูปที่ 3.6 (การติดต่อสื่อสาร ATAPE บนเครือข่ายแบบ MINs$^+$ เช่นเครือข่ายแบบโอเมก้า (ก) และเครือข่ายแบบฟลิบ (ข))

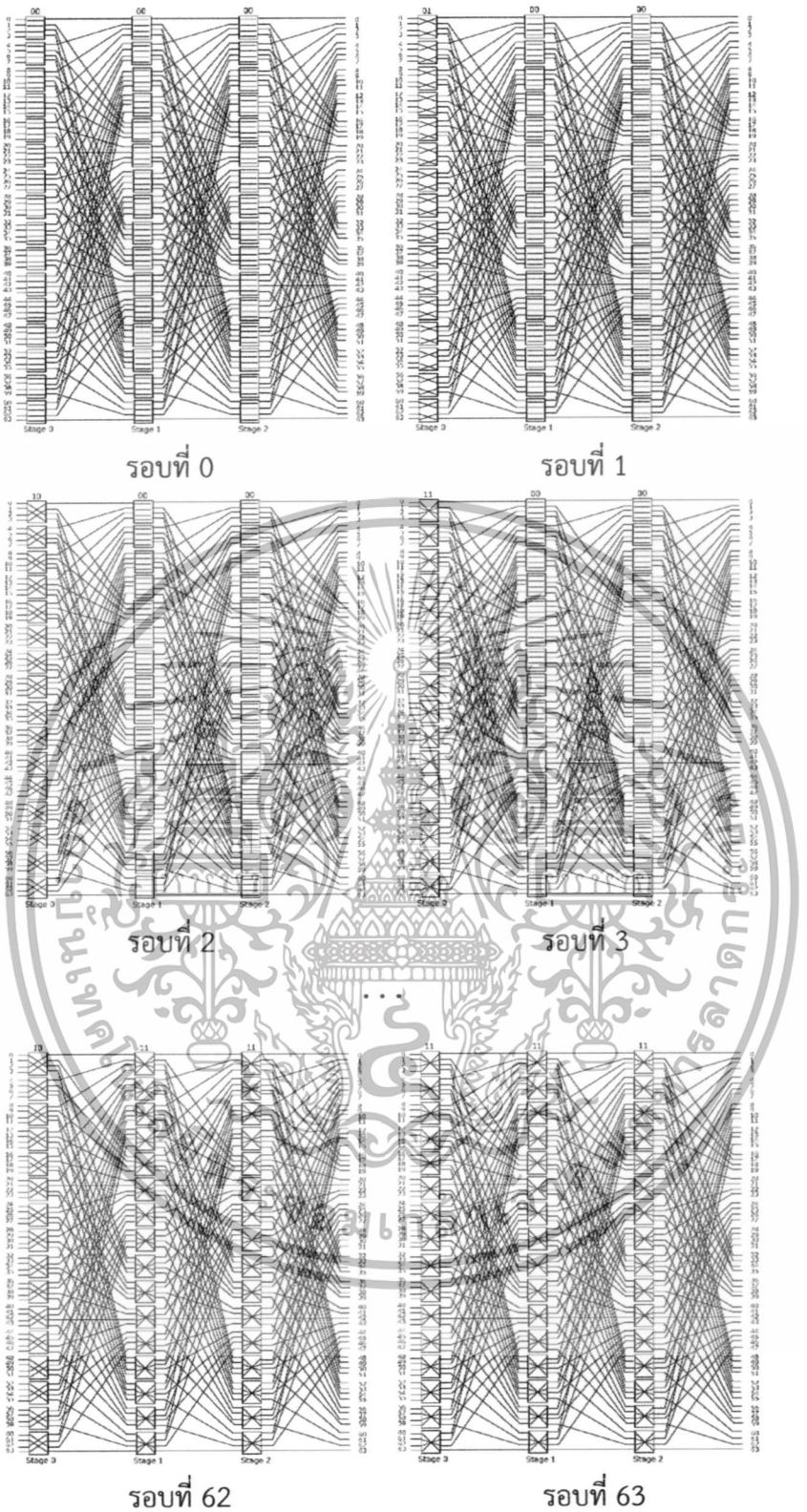


รูปที่ 3.5 ค่าควบคุมแบบบิต $C = (c_{n-1}c_{n-2}c_{n-3}...c_1...c_1c_0)_2$ สำหรับ m สเตจ (k บิตต่อสวิทช์ต่อสเตจ) บนเครือข่ายแบบ MINs$^+$ (N = 16, d = 4, k = 2) (ก) เครือข่ายการจัดเส้นทางแบบไปข้างหน้า (Forward MINs$^+$) (ข) เครือข่ายการจัดเส้นทางแบบย้อนกลับ (Backward MINs$^+$)



(ก)

รูปที่ 3.6 ผลลัพธ์ของการติดต่อสื่อสาร ATAPE บนเครือข่ายแบบ MINs^{+} ที่ใช้สวิทช์ขนาด 4 x 4 (N = 64, d = 4) (ก) เครือข่ายแบบโอเมก้าเป็นเครือข่ายการจัดเส้นทางแบบไปข้างหน้า และ (ข) เครือข่ายแบบฟลิบเป็นเครือข่ายการจัดเส้นทางแบบย้อนกลับ



(ข)

รูปที่ 3.6 (ต่อ)

3.3 การติดต่อสื่อสาร ATAPE แบบฝังลงบนชิปที่ดีที่สุดบนเครือข่ายแบบ CMINs$^+$

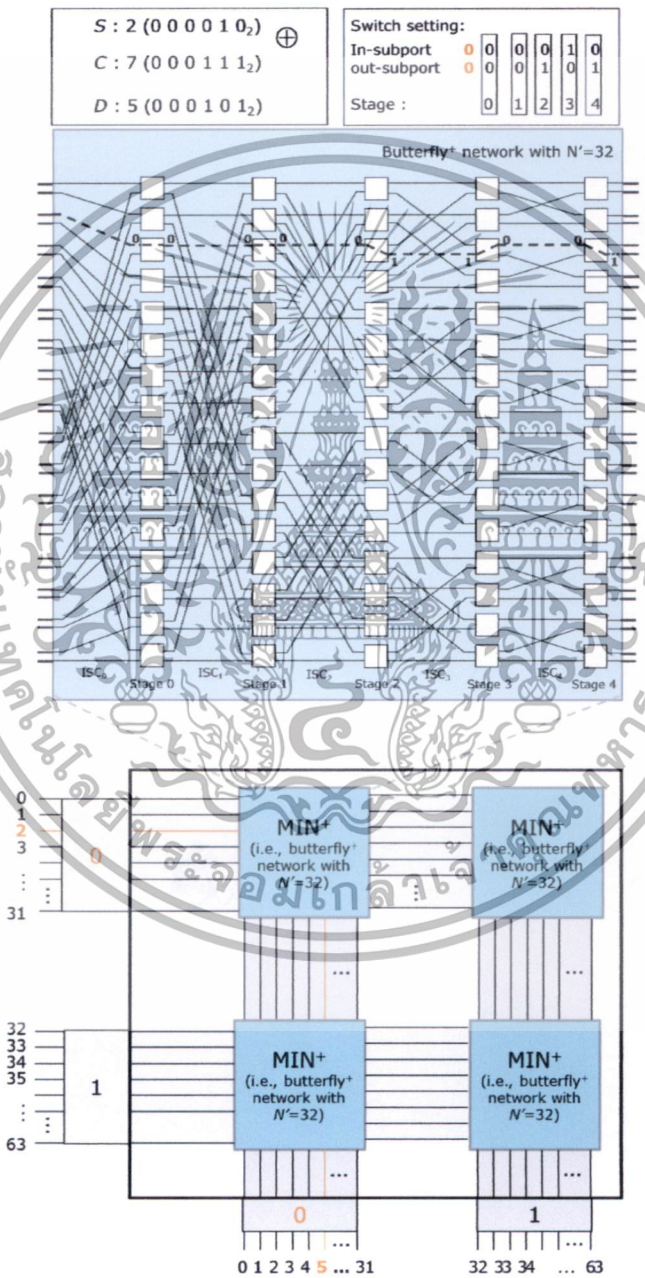
การที่จะเพิ่มประสิทธิภาพของการติดต่อสื่อสารแบบ ATAPE ได้เร็วยิ่งขึ้นนั้นสามารถทำได้ด้วยการออกแบบหรือปรับปรุงเครือข่ายให้เป็นเครือข่ายที่มีการติดต่อสื่อสารได้เร็วขึ้นกว่าเดิมนั่นเอง หัวข้อนี้จึงได้เสนอเครือข่ายแบบใหม่อีกชนิดที่เรียกว่า ครอสส์บาร์ของเครือข่ายแบบ MINs$^+$ (a Crossbar of MINs$^+$: CMINs$^+$) ที่สามารถติดต่อสื่อสารได้เร็วขึ้นและค่าใช้จ่ายในการสร้างไม่แพงมากเครือข่ายแบบ CMINs$^+$ เป็นการรวมข้อดีของเครือข่ายหลักๆ 2 ชนิดคือ เครือข่ายแบบครอสส์บาร์ (Crossbar Network) ซึ่งเป็นเครือข่ายที่มีการติดต่อสื่อสารเร็วแต่ราคาในการสร้างแพง และเครือข่ายแบบ MINs$^+$ ที่มีราคาถูกกว่าแต่ติดต่อสื่อสารได้ช้ากว่า โดยเครือข่ายแบบครอสส์บาร์เป็นเครือข่ายแบบส่งผ่านข้อมูลได้เพียงผ่านแค่เพียงสเตจเดียวเท่านั้นเพราะมีแบนด์วิสสูง (เป็นเครือข่ายที่สามารถเชื่อมต่อได้แบบ 1-1 ในการติดต่อสื่อสารระหว่างหน่วยประมวลผลได้โดยตรง) แต่เป็นเครือข่ายแบบไดนามิคที่มีราคาแพง (ต้องการจุดตัด (Cross-Points) ระหว่างสวิทช์จำนวน $N \times N$ จุด เพื่อกำหนดค่าของการเชื่อมต่อระหว่างต้นทางและปลายทางจำนวน N คู่ ที่ผ่านการหน่วงเวลาแค่เพียงสเตจเดียว) ในทางตรงกันข้ามเครือข่ายแบบ MINs$^+$ มีราคาที่ถูกกว่า เมื่อใช้สวิทช์ขนาด 2 x 2 ($N = 2^n$) จะใช้สวิทช์ทั้งหมดจำนวน $nN/2$ สวิทช์ แต่เครือข่ายแบบ MINs$^+$ จะมีการติดต่อสื่อสารระหว่างหน่วยประมวลผลที่นานกว่านั่นคือต้องมีการหน่วงเวลาผ่านสเตจจำนวน $\log_2 N$ สเตจ ดังนั้นในแต่ละชนิดของเครือข่ายแบบ MINs$^+$ จะมีสเตจจำนวน $n = \log_2 N$ สเตจ ($N/2$ สวิทช์ต่อสเตจ) และการเชื่อมต่อระหว่างสเตจ (Inter-Stage Connection: ISCs) จำนวน n ชุด ส่วนการออกแบบ CMINs$^+$ ที่เสนอขึ้นใหม่เพื่อเชื่อมต่อระหว่างต้นทางและปลายทาง (เช่น หน่วยประมวลผลไปยังหน่วยประมวลผล หน่วยประมวลผลไปยังโมดูลของหน่วยความจำ เป็นต้น) จะมีจุดตัดของสวิทช์จำนวน $x \times x$ จุดตัด ($x = 2^b < N$) โดยจุดตัดจะแทนที่ด้วยเครือข่ายย่อยแบบ MIN$^+$ โดยในเครือข่ายแบบ CMINs$^+$ จะมีเครือข่ายย่อยแบบ MIN$^+$ อย่างไรก็ตามต้นทุนในการสร้างและเวลาในการติดต่อสื่อสารในเครือข่ายแบบ CMINs$^+$ จะขึ้นอยู่กับจำนวนดีกรีของครอสส์บาร์ที่ถูกนำมาใช้ ดังแสดงในตารางที่ 3.2 (กรณี $N = 64$)

ตารางที่ 3.2 ต้นทุนและจำนวนสเตจที่แสดงการหน่วงเวลาที่ลดลงของเครือข่ายแบบ CMINs$^+$ (N=64)

ดีกรีของครอสส์บาร์	ขนาดของระบบย่อยของเครือข่ายแบบ MIN$^+$	จำนวนสเตจที่ลดลง (การหน่วงเวลาที่ลดลง)	จำนวนสวิทช์ที่ใช้ใน CMINs$^+$ (ต้นทุน)
16x16	4	4 สเตจ	1,024 สวิทช์
8x8	8	3 สเตจ	768 สวิทช์
4x4	16	2 สเตจ	512 สวิทช์
2x2	32	1 สเตจ	320 สวิทช์

ตารางที่ 3.2 แสดงดีกรีที่แตกต่าง ($x \times x$) บนเครือข่ายแบบ CMINs$^+$ (x = 2, 4, 8 และ 16) เพื่อจะเปรียบเทียบต้นทุนและการหน่วงเวลาผ่านสเตจ (เช่น ต้นทุนขึ้นกับจำนวนสวิทช์ และการหน่วงเวลาขึ้นกับจำนวนสเตจ) โดยที่ถ้าดีกรีมีจำนวนมากกว่าการติดต่อสื่อสารจะเร็วกว่าแต่ราคาแพงกว่า สำหรับกรณีของเครือข่ายแบบ MINs$^+$ แบบเดิมที่ N = 64 ต้องการการหน่วงเวลา $\log_2 N$ = 6 สเตจและสวิทช์จำนวน 192 สวิทช์ แต่ในเครือข่ายแบบ CMINs$^+$ ที่มีขนาด 2 x 2 ก็จะสามารถส่งข้อมูลได้เร็วกว่า MINs$^+$ แต่ละประเภทในการติดต่อสื่อสารระหว่างหน่วยประมวลผล N หน่วย เพราะใน CMINs$^+$ แต่ละจุดตัดจะแทนด้วยเครือข่ายย่อยแบบ MIN$^+$ โดยที่ $N'=N/x=32$ และ $\log_2 N'=5$

สเตจ ดังนั้นทำให้สามารถลดสเตจได้ 1 สเตจในการเชื่อมต่อระหว่างต้นทางและปลายทาง และในระบบทั้งหมดจะใช้สวิทช์จำนวน $N'/2 \times \log_2 N' = 320$ สวิทช์ ดังนั้นในการแลกเปลี่ยนข้อมูลในแต่ละคู่ของ N หน่วยประมวลผลจะใช้เวลาส่งผ่านสเตจที่ลดลงจำนวน 1 สเตจ และในทำนองเดียวกัน CMINs$^+$ ที่มีขนาด 4 x 4 แต่ละจุดตัดคือเครือข่ายย่อยแบบ MIN$^+$ โดยที่ $N'=16$ และ $\log_2 N'= 4$ สเตจและการเชื่อมต่อระหว่างต้นทางและปลายทางสามารถลดสเตจลงได้จำนวน 2 สเตจ โดยทั่วไป CMINs$^+$ ที่มีขนาด $x \times x$ ($x = 2^b$, $N=2^n$, $N'=2^{n-b}$) จะสามารถสร้างเครือข่ายโดยลดสเตจได้จำนวน $n-b$ สเตจ (หรือ b สเตจที่ลดลง) โดยใช้สวิทช์ทั้งหมดจำนวน $x^2(n-b)(N/x)/2$ สวิทช์



รูปที่ 3.7 ตัวอย่างของ CMINs$^+$ (N = 64 และ ดีกรีของครอสส์บาร์ = 2 x 2) กับแต่ละจุดตัดที่แทนด้วยเครือข่ายแบบบัสเตอร์ฟลายพลัส (N'=32) และการจัดเส้นทางจากต้นทาง ($S = 2_{10}$) ไปยังปลายทาง ($D = 5_{10}$) โดยใช้ค่าควบคุมหลัก $C=7_{10}$ (หรือ 000111_2)

สำหรับเครือข่ายแบบ CMINs$^+$ การจัดเส้นทางภายในเป็นแบบสมบูรณ์ขึ้นกับชนิดของเครือข่ายย่อยแบบ MINs$^+$ โดยในแต่ละประเภทของระบบย่อย MINs$^+$ สามารถเป็นเครือข่ายแบบโอเมก้า เครือข่ายแบบฟลิบ เครือข่ายแบบคิวบ์หรือเครือข่ายแบบบันยานพลัส เครือข่ายแบบบัตเตอร์ฟลายพลัส เครือข่ายแบบเบสไลน์พลัสและเครือข่ายแบบอินเวอร์สเบสไลน์พลัส ในรูปที่ 3.7 แสดงตัวอย่างเครือข่ายแบบ CMINs$^+$ ขนาด N = 64 ของครอสส์บาร์ของเครือข่ายย่อยแบบบัสเตอร์ฟลายพลัส ($N' = N/x$ = 32) โดยที่ดีกรีของครอสส์บาร์ = 2 x 2 ที่มีการเชื่อมต่อระหว่างหน่วยประมวลผล 64 หน่วยประมวลผลผ่าน 5 สเตจในระบบย่อยของเครือข่ายแบบ MINs$^+$

เครือข่ายแบบ CMINs$^+$ มีองค์ประกอบเป็นเครือข่ายย่อย MINs$^+$ จึงสามารถฝังฟังก์ชัน ATAPE ที่นำเสนอในหัวข้อที่ 3.2 โดยถ้าใช้ฟังก์ชัน $D = S \oplus (C + order) \bmod N$ โดยที่ $S = (s_{n-1}s_{n-2} \ldots s_1s_0)$ และ $D = (d_{n-1}d_{n-2} \ldots d_1d_0)$ ในกรณีของ D และ S ขนาด n บิตจะใช้บิตที่มีนัยสำคัญสูง (MSb) สำหรับระบุค่าครอสส์บาร์ขนาด $x \times x$ และตำแหน่งบิตที่เหลือจะใช้สำหรับระบบย่อยของเครือข่ายแบบ MINs$^+$ โดยผ่านสเตจจำนวน $n\text{-}b = \log_2 N/x$ สเตจ เมื่อกำหนดให้ $x \times x = 2^b \times 2^b$ แทนแถวและคอลัมน์ของเครือข่ายแบบ CMINs$^+$ โดยบิตที่มีนัยสำคัญสูงจำนวน b บิตของ $S = (s_{n-1}..s_{n-b})$ จะใช้สำหรับเลือกแถว (ของพอร์ตเข้า) และบิตดังกล่าวจำนวน b บิตของ $D = (d_{n-1}..d_{n-b})$ จะใช้สำหรับเลือกคอลัมน์ (ของพอร์ตออก) สำหรับ $n\text{-}b$ บิตที่เหลือจะใช้ในการจัดเส้นทางภายในของแต่ละระบบย่อยแบบ MIN$^+$

รูปที่ 3.7 แสดงตัวอย่างของการจัดเส้นทางภายในของเครือข่ายแบบ CMINs$^+$ ที่มีขนาด 2 x 2 (N=64, x=2 และ N'=32) ในวิธีเรียงสับเปลี่ยนแบบส่งผ่านเพียงรอบเดียว (Single-Pass Permutation) ได้กำหนดให้มีการจับคู่แบบ 1 ต่อ 1 ของค่า $\rho = (a_0\ a_1 \ldots a_S \ldots a_{N-1})$ ระหว่างต้นทาง (S) และปลายทาง $D = \rho(S)$ ของเครือข่าย กำหนดให้ $\rho(S) = \epsilon \{0, 1, 2, 3, \ldots, N\text{-}1\}$ โดยที่ S = 0, 1, 2, 3, ... , N-1 ยกตัวอย่างเช่น ρ = (7 6 5 4 3 2 1 0 15 14 13 12 11 10 ... 56) ระบุการจับคู่ระหว่าง S = 64 ค่ากับ D =64 ค่า โดยใช้ 6 บิตของค่าควบคุมหลักของสเตจคือ 000111_2 โดยที่ 1 บิตที่มีนัยสำคัญสูงของ S ใช้สำหรับเลือกแถวและ 1 บิตที่มีนัยสำคัญสูงของ D ใช้สำหรับเลือกคอลัมน์ และที่เหลือ 5 บิตใช้สำหรับจัดเส้นทางภายในของระบบ MINs$^+$ ย่อย โดยการแบ่งใช้ทีละ 1 บิตต่อ 1 สวิทช์ต่อ 1 สเตจสำหรับเชื่อมต่อภายในสวิทช์ทุกๆ สวิทช์ได้อย่างอิสระโดยใช้ C ค่าเดียวกัน (เช่น S = 0 เชื่อมไปยัง D = 7, S = 1 เชื่อมไปยัง D = 6, S = 2 เชื่อมไปยัง D = 5, ... , และ S = 63 เชื่อมไปยัง D = 56) สำหรับ N หน่วยประมวลผลจะสามารถกำหนดค่าการเชื่อมต่อระหว่างต้นทาง (S) และปลายทาง (D) ในแบบขนาน เช่นรายละเอียดของการเชื่อมต่อระหว่างต้นทาง S = 2 และปลายทาง D = 5 ด้วยค่าควบคุมหลัก C = 7 (หรือ $000111)_2$ จะได้ปลายทาง $D = S \oplus (C+0)_{\bmod 16}$ = 5 (หรือ $000101)_2$ โดยขั้นแรกการเลือกระบบ MINs$^+$ ย่อยก่อนโดยใช้ 1 บิต s_5 (= 0) ไปเลือกแถวที่ 0 (ในพอร์ตเข้าของครอสส์บาร์) และ1 บิต d_5 (= 0) ไปเลือกคอลัมน์ที่ 0 (ในพอร์ตออกของครอสส์บาร์) และขั้นสอง 5 บิตที่เหลือจะถูกนำไปใช้สำหรับการจัดเส้นทางภายในของระบบย่อยของ MIN$^+$ ดังนั้นในการเชื่อมต่อระหว่างต้นทาง S = 2 และปลายทาง D = 5 บนเครือข่ายแบบบัตเตอร์ฟลายพลัสที่ N'=32 จะดำเนินการผ่าน 5 สวิทช์ ผ่านสเตจจำนวน 5 สเตจดังนี้ พอร์ตเข้า 0 ของ S ไปยังพอร์ตออก 0 ของ D ที่สเตจ 0 และสเตจ 1 ถัดไปพอร์ตเข้า 0 ของ S ไปยังพอร์ตออก 1 ของ D ที่สเตจ 2 ถัดไปพอร์ตเข้า 1 ของ S ไปยังพอร์ตออก 0 ของ D ที่สเตจ 3 และพอร์ตเข้า 0 ของ S ไปยังพอร์ตออก 1 ของ D ที่สเตจ 4

3.4 แอพพลิเคชั่นของขั้นตอนวิธี ATAPE แบบฝังลงบนชิป

หัวข้อนี้จะแสดงผลการทดลองของการประยุกต์ใช้ขั้นตอนวิธี ATAPE ในการจัดเส้นทางภายในที่หลากหลายของ ของเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด $d \times d$ โดยหัวข้อที่ 3.4.1 แสดงตัวอย่างเครือข่ายขนาด $N = 8$, $d = 2$ และขั้นตอนวิธี ATAPE สามารถกำหนดแบบการตั้งค่าในเครือข่ายได้ 8 แบบ ผ่านสเตจจำนวน $n = \log_2 N = 3$ สเตจ ซึ่งในรอบที่ 0 ($C = 0\ 0\ 0_2$) ไปจนกระทั่งรอบที่ 7 ($C = 1\ 1\ 1_2$) โดยใช้บิตค่าควบคุมหลัก 1 บิตในแต่ละสวิทช์ของสเตจนั้นๆ (จำนวน $N/2$ สวิทช์) บนเครือข่ายแบบ MINs$^+$ (รูปที่ 3.9) ในหัวข้อที่ 3.4.2 (รูปที่ 3.10) เสนอการประยุกต์ใช้ขั้นตอนวิธี ATAPE บนเครือข่ายแบบ CMINs$^+$ ที่มีขนาด $N = 16$, $x = 2$ และหัวข้อสุดท้ายเสนอแอพพลิเคชันของขั้นตอนวิธี ATAPE ที่มีประสิทธิภาพในการทรานสโพสเมตริกส์แบบขนาน (รูปที่ 3.11) และผลการทดลองในหัวข้อที่ 3.4.3

3.4.1 การประยุกต์ใช้ขั้นตอนวิธี ATAPE บนเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด $d \times d$

การนำขั้นตอนวิธี ATAPE ไปประยุกต์ใช้บนทุกๆ ประเภทของเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด $d \times d$ ซึ่งประกอบไปด้วย 1) เครือข่ายการจัดเส้นทางแบบไปข้างหน้า (เช่น เครือข่ายแบบโอเมก้า เครือข่ายแบบบัตเตอร์ฟลายพลัส และเครือข่ายแบบอินเวอร์สเบสไลน์พลัส) และ 2) เครือข่ายการจัดเส้นทางแบบย้อนกลับ (เช่นเครือข่ายแบบฟลิป เครือข่ายแบบคิวบ์หรือบันยานพลัส และเครือข่ายแบบเบสไลน์พลัส) โดยผลลัพธ์ของการประยุกต์ใช้ขั้นตอนวิธี ATAPE บนเครือข่ายการจัดเส้นทางแบบไปข้างหน้าโดยใช้สวิทช์ขนาด $d \times d$ ($d=2$) เพื่อกำหนดค่าเครือข่าย N ค่าดังรูปที่ 3.9 (ก) โดยใช้ค่าควบคุมหลัก $C = (c_{n-1}c_{n-2}c_{n-3} \ldots c_1c_0)_2$ สำหรับจัดเส้นทางภายในโดยการแบ่ง 1 บิต (บนสเตจ 0, 1, 2, ..., n-1) จากบิตที่มีนัยสำคัญสูงไปยังบิตที่มีนัยสำคัญต่ำ เช่นในรอบที่ 2 หน่วยประมวลผล N หน่วยประมวลผลสามารถกำหนดการเชื่อมต่อระหว่างต้นทางและปลายทางแบบขนานได้ดังนี้ $P_0 \longrightarrow P_2\ P_1 \longrightarrow P_3\ P_2 \longrightarrow P_0\ P_3 \longrightarrow P_1\ P_4 \longrightarrow P_6\ P_5 \longrightarrow P_7\ P_6 \longrightarrow P_4\ P_7 \longrightarrow P_5$ โดยเฉพาะในกรณีของ $S = 1$ (หรือ $001)_2$ และ $C = 2$ (หรือ $010)_2$ จะเตรียมการกำหนดค่าภายในสวิทช์สำหรับหน่วยประมวลผลปลายทาง $D = S \oplus (C + order) \bmod N = 3$ (หรือ $011)_2$ โดยที่ในกรณีนี้กำหนดให้ $order = 0$ (เช่น สเตจที่ 0 ทำการเชื่อมต่อกับพอร์ตภายในพอร์ตเข้า 0 ไปยังพอร์ตออก 0 สเตจที่ 1 เชื่อมพอร์ตภายในพอร์ตเข้า 0 ไปยังพอร์ตออก 1 และสเตจที่ 2 เชื่อมพอร์ตภายในพอร์ตเข้า 1 ไปยังพอร์ตออก 1)

ส่วนเครือข่ายการจัดเส้นทางแบบย้อนกลับ ($d = 2$) จะสามารถจัดเส้นทางการเชื่อมต่อจากต้นทางไปยังปลายทางแบบขนานได้เหมือนกับเครือข่ายการจัดเส้นทางแบบไปข้างหน้า ยกเว้นในกรณีของ (n บิต) ค่าควบคุมหลัก (C) จะใช้บิตที่มีนัยสำคัญต่ำไปยังบิตที่มีนัยสำคัญสูง ดังรูปที่ 3.9 (ข) (เช่นการเชื่อมต่อระหว่าง (1, 3) ของรอบที่ $r = 2$ สามารถกำหนดค่าสวิทช์ทุกๆ สวิทช์ ดังนี้พอร์ตภายในพอร์ตเข้า 1 ไปยังพอร์ตออก 1 ที่สเตจ 0 พอร์ตภายในพอร์ตเข้า 0 ไปยังพอร์ตออก 1 ที่สเตจ 1 และพอร์ตภายในพอร์ตเข้า 0 ไปยังพอร์ตออก 0 ที่สเตจ 2)

ส่วนสุดท้ายรูปที่ 3.9 (ค) แสดงการจัดเส้นทางของเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด 4 x 4 กำหนดให้ $N = 16$ และ $d = 4$ <u>หมายเหตุ</u> สำหรับเครือข่ายประเภทอื่นๆ ของเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด $d \times d$ การประยุกต์ใช้ฟังก์ชัน ATAPE ทั้งสองแบบได้ทำการตรวจสอบความถูกต้องสมบูรณ์โดยการเขียนโปรแกรมทดสอบบนการจำลองเครือข่ายแบบ MIN$^+$ ที่มีขนาดจาก $N = 16$ ถึง $N = 2^{20}$

3.4.2 การประยุกต์ใช้ขั้นตอนวิธี ATAPE บนเครือข่ายแบบ CMINs$^+$ ที่เร็วขึ้น

การนำขั้นตอนวิธี ATAPE ไปประยุกต์ใช้บนเครือข่ายแบบ CMINs$^+$ ที่มีขนาด $N = 16 = 2^4$, $x = 2$ และ $d = 2$ ด้วยระบบย่อยของเครือข่ายแบบ MIN$^+$ ($N' = N/x = 8$) ดังนั้นทุกๆ การเชื่อมต่อระหว่างต้นทางและปลายทางจะใช้ 1 บิตที่อยู่ในตำแหน่งที่มีนัยสำคัญสูงของต้นทาง $S = (s_{n-1}s_{n-2}... s_1s_0)$ และปลายทาง $D = (d_{n-1}d_{n-2}... d_1d_0)$ สำหรับเลือกแถวและคอลัมน์ของครอสส์บาร์ ดังนั้น n-1 บิตที่เหลือจะใช้สำหรับจัดเส้นทางภายในของเครือข่ายแบบ MIN$^+$ ผ่าน n-1 สเตจ ผลลัพธ์ในการประยุกต์ใช้ขั้นตอนวิธี ATAPE บนครอสส์บาร์ของเครือข่ายการจัดเส้นทางแบบไปข้างหน้าหรือย้อนกลับสำหรับการกำหนดค่าเครือข่ายจำนวน N ค่าแสดงในรูปที่ 3.10 (N=16) คือครอสส์บาร์ของเครือข่ายแบบโอเมก้าและเครือข่ายแบบบันยานพลัส สำหรับตัวอย่างแรกการจัดเส้นทางภายในของครอสส์บาร์ของเครือข่ายแบบ MINs$^+$ แบ่งเป็น 2 ส่วนคือ บิตที่มีนัยสำคัญสูง 1 บิต (หรือตำแหน่งบิตที่สูง) ใช้สำหรับเลือกครอสส์บาร์ (เช่น การเลือกครอสส์บาร์ในแถวที่ 0 (s_3=0) และคอลัมน์ที่ 0 (d_3=0)) และ 3 บิตที่เหลือจะใช้ในการเชื่อมต่อภายในสวิทช์ผ่านการจัดเส้นทางภายในระบบย่อยของเครือข่ายแบบ MINs$^+$ ในรูปที่ 3.10 (ก) เสนอผลลัพธ์ของขั้นตอนวิธี ATAPE บนครอสส์บาร์ขนาด 2x2 ของเครือข่ายแบบโอเมก้า (N=16, x=2 และ $N' = N/x = 8$) โดยที่ครึ่งแรก (N/2) ของวิธีเรียงสับเปลี่ยนในตารางลาตินจะทำการประมวลผลบนจุดตัดของระบบย่อย (0, 0) และ (1, 1) ส่วนอีกครึ่งหลัง (N/2) ของวิธีเรียงสับเปลี่ยนในตารางลาตินจะทำการประมวลผลบนจุดตัดของระบบย่อย (0, 1) และ (1, 0) ตามผลลัพธ์ของ $D = S \oplus C$ ยกตัวอย่างเช่นรอบที่ r = 1 (หรือ 001)$_2$ ทุกๆ การเชื่อมต่อ ($P_0 \longrightarrow P_1$, $P_1 \longrightarrow P_0$, $P_2 \longrightarrow P_3$, $P_3 \longrightarrow P_2$, $P_4 \longrightarrow P_5$, ..., $P_{15} \longrightarrow P_{14}$) จะทำการเชื่อมต่อสวิทช์ผ่านพอร์ตภายในแบบต่อตรง (-) คือ 0-0 หรือ 1-1 ที่สเตจ 0 และ 1 และเชื่อมต่อพอร์ตภายในแบบสับเปลี่ยน 0-1 และ 1-0 ที่สเตจ 2 ส่วนในกรณีอื่นๆ ($r \neq 1$) จะมีการกำหนดผ่านพอร์ตภายในของสเตจอื่นๆ ได้ในทำนองเดียวกัน ดังรูปที่ 3.10 (ข) แสดงเครือข่ายการจัดเส้นทางแบบย้อนกลับคือครอสส์บาร์ขนาด 2x2 ของเครือข่ายแบบบันยานพลัส (N = 16, x = 2 และ $N' = N/x$ = 8)

<u>หมายเหตุ</u> ในการประยุกต์ใช้ดีกรีของครอสส์บาร์ขนาดอื่นๆ (x) และ N ได้มีการตรวจสอบความถูกต้องโดยการเขียนโปรแกรมทดสอบ

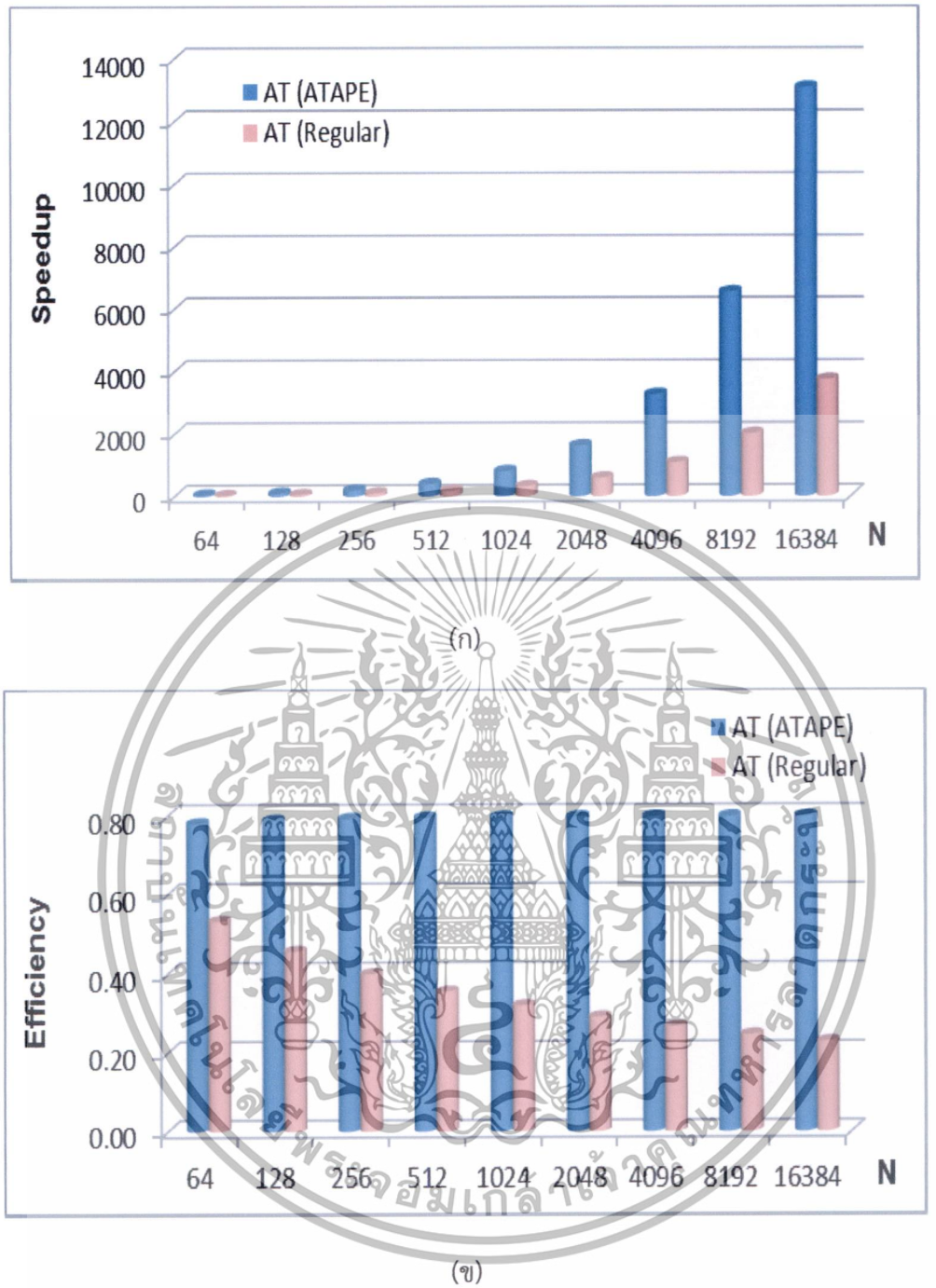
3.4.3 การประมวลผลขั้นตอนวิธี ATAPE สำหรับการทรานสโพสเมตริกซ์ที่เร็วขึ้น

แอพลิเคชั่นที่ประยุกต์ใช้การติดต่อสื่อสาร ATAPE ได้อย่างมีประสิทธิภาพคือการทรานสโพสเมตริกซ์แบบขนานบน N หน่วยประมวลผลของเครือข่ายแบบ MINs$^+$ ด้วยเวลาที่เหมาะสมคือ $O(N+\log_2 N)$ โดยปกติการทำทรานสโพส (A^T) แบบลำดับนั้นจะใช้เวลา = $O(N^2)$ และการทำทรานสโพสแบบขนานบนเครือข่ายทอรัส (Torus Networks) ขนาด NxN = $O(N)$ และบนเครือข่ายไฮเปอร์คิวบ์ (Hypercube Networks) ขนาด $N = 2^n$ ใช้เวลา = $O(N\log_2 N)$

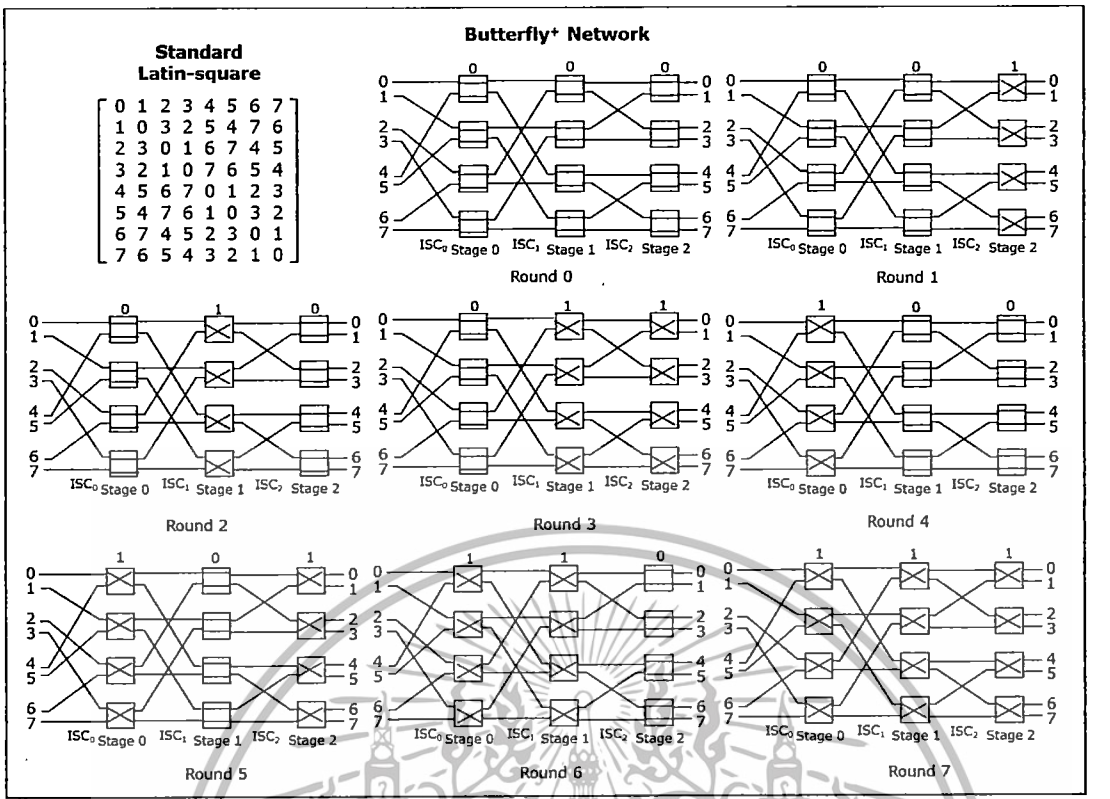
การวัดประสิทธิภาพของขั้นตอนวิธี ATAPE ได้ทำการทดลองโดยการศึกษาแบบจำลองเครือข่าย MINs$^+$ เพื่อเปรียบเทียบอัตราเร็วที่เพิ่มขึ้น (Speedup $S_P = T_1/T_P$) และประสิทธิภาพ

(Efficiency $E_P = S_P/P$) ของการทรานสโพสเมตริกซ์โดยใช้ขั้นตอนวิธี ATAPE (A^T_{ATAPE}) เปรียบเทียบกับการทรานสโพสเมตริกทั่วไปบนเครือข่ายแบบ MINs$^+$ โดยที่กำหนดให้ T_1 หมายถึงเวลาของการประมวลผล A^T โดยหนึ่งหน่วยประมวลผล (Processing Element : PE) และ T_P หมายถึงเวลาของการประมวลผล A^T แบบขนานโดยใช้ P หน่วยประมวลผล ในการจำลองระบบเครือข่ายกำหนดให้มีพารามิเตอร์ดังต่อไปนี้ ช่วงเวลาของการติดต่อสื่อสาร (Clock Period of Communication Time : p_{comm}) คือ 1.25 เท่าของช่วงเวลาของการคำนวณ (Clock Period of Computing Time : p_{comp}) และ p_{comp} คือ 4 เท่าของช่วงเวลาของการหน่วงผ่านสเตจ (Clock Period of Delay Time through a Stage : p_{delay}) รูปที่ 3.8 แสดงประสิทธิภาพของการประมวลผล A^T โดยใช้ขั้นตอนวิธี ATAPE ที่สามารถแสดงการประมวลผลในกรณีที่ $P=N$ โดยที่ค่า Speedup เข้าใกล้ P และค่า Efficentcy เข้าใกล้ 1 สำหรับเครือข่ายทุกๆ แบบขนาด N (= 64 ถึง 16,384 หน่วยประมวลผล) ตามการติดต่อสื่อสาร ATAPE ที่เหมาะสมที่สุดผ่านมัลติสเตจไปป์ไลน์ (Multistage Pipelining) โดยประสิทธิภาพของการประมวลผล A^T ด้วยวิธีทั่วไปจะลดลงเมื่อหน่วยประมวลผล N เพิ่มขึ้น (เพราะเวลาที่ใช้ในการติดต่อสื่อสารเริ่มมีผลกระทบ) แต่ขั้นตอนวิธี ATAPE ที่เสนอใหม่บนเครือข่ายแบบ MINs$^+$ จะไม่มีผลกระทบมากนักเมื่อมีจำนวน N เพิ่มมากขึ้น

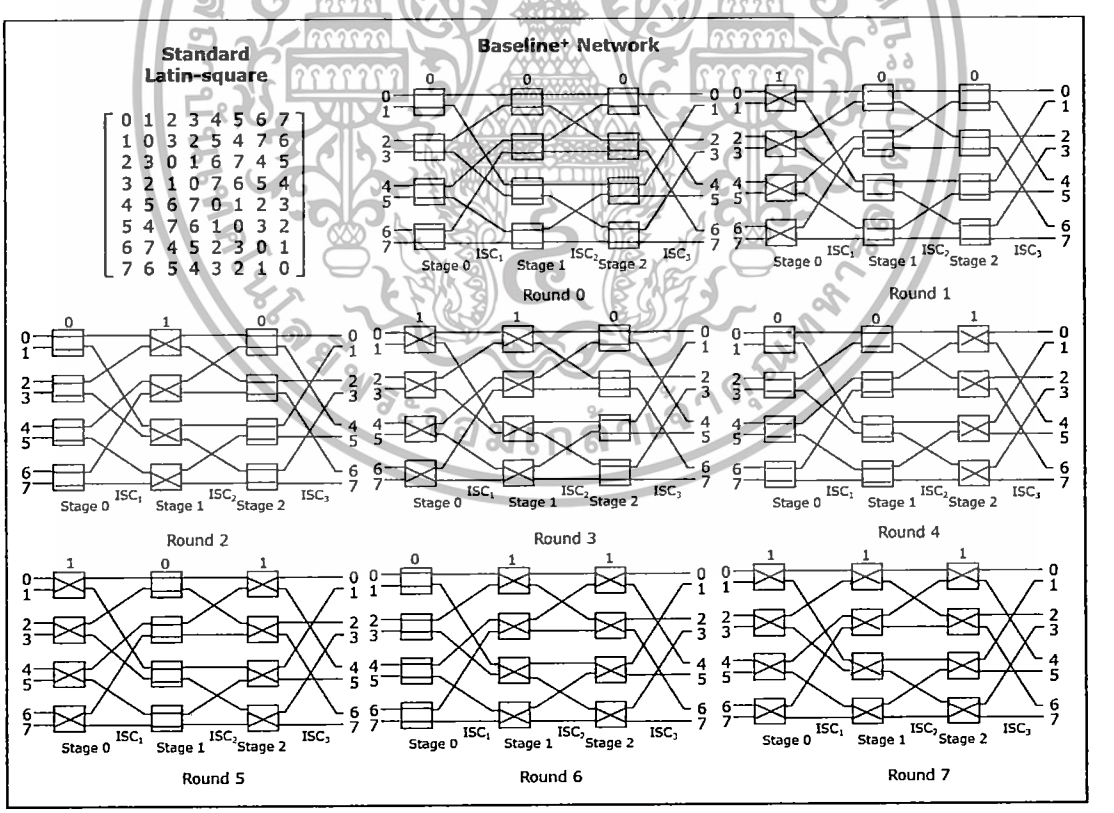
และสุดท้ายนี้ ในรูปที่ 3.11 แสดงผลการประยุกต์ใช้ ATAPE และตอบคำถามที่ว่า “การติดต่อสื่อสาร ATAPE ทำงานได้ผลอย่างมีประสิทธิภาพและถูกต้องได้อย่างไร” จึงแสดงผลลัพธ์จากการประยุกต์ใช้ในขั้นตอนวิธี ATAPE เพื่อทำประมวลผลการทรานสโพส A^T (N = 8) ทีละขั้นตอนบนเครือข่ายแบบบัตเตอร์ฟลายพลัส โดยการประมวลผล A^T ดังกล่าวจะถูกคำนวณเป็นจำนวน N รอบ (ยกเว้นรอบที่ 0) ด้วยฟังก์ชัน $D = S \oplus C$ และค่าควบคุมหลัก C = 1 2 3 ... N-1 เริ่มต้นด้วยการที่เมตริกส์ A จะถูกแบ่งเป็น N ส่วน และกำหนดให้แต่ละ N ส่วน ($A_{i,0}, A_{i,1}, A_{i,2}, \ldots, A_{i,7}$) เก็บในแต่ละหน่วยประมวลผล P_i โดยที่ i = 0 1 2 3 ... และ N-1 เมื่อทำการส่งหรือรับ $A_{S,D}$ (ข้อมูลเฉพาะของเมตริกซ์ A) จากต้นทาง (S) ไปยังทุกๆ ปลายทาง (D) เป็นจำนวน N-1 รอบผลลัพธ์ของการประมวลผล A^T จะปรากฏใน N หน่วยประมวลผล เช่นในรอบที่ 1 ผลลัพธ์ $A_{S,D}$ ของวิธีเรียงสับเปลี่ยน [1 0 3 2 5 4 7 6] คือ $A_{0,1}$ (จาก P_0 ไปยัง P_1), $A_{1,0}$ (จาก P_1 ไปยัง P_0), $A_{2,3}$ (จาก P_2 ไปยัง P_3), $A_{3,2}$ (จาก P_3 ไปยัง P_2), $A_{4,5}$ (จาก P_4 ไปยัง P_5), $A_{5,4}$ (จาก P_5 ไปยัง P_4), $A_{6,7}$ (จาก P_6 ไปยัง P_7), $A_{7,6}$ (จาก P_7 ไปยัง P_6) และใช้วิธีเรียงสับเปลี่ยนถัดไปเรื่อยๆ ในรอบที่ 2 จนกระทั่งถึงรอบที่ 7 หรือรอบสุดท้ายผลลัพธ์ $A_{S,D}$ ของวิธีเรียงสับเปลี่ยน [7 6 5 4 3 2 1 0] จะส่ง N ข้อมูลชุดสุดท้าย ($A_{0,7}, A_{1,6}, A_{2,5}, A_{3,4}, A_{4,3}, A_{5,2}, A_{6,1}$ และ $A_{7,0}$) จากทุกๆ หน่วยประมวลผล P_S ไปยัง P_D



รูปที่ 3.8 การวัดประสิทธิภาพในการทำทรานสโพสแบบ ATAPE (A^T_{ATAPE}) เปรียบเทียบกับการทำทรานสโพสแบบทั่วไป (A^T)

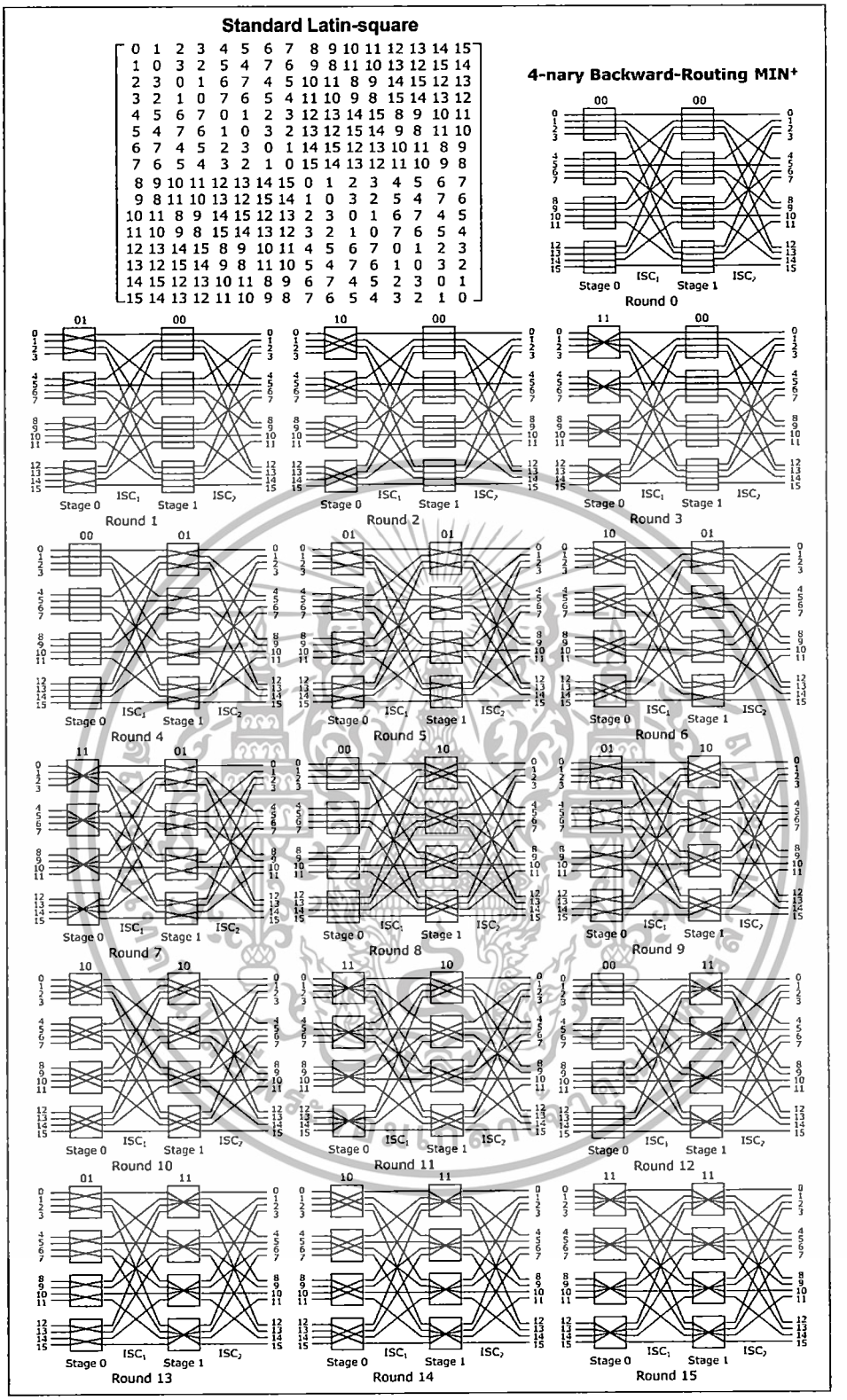


(ก)



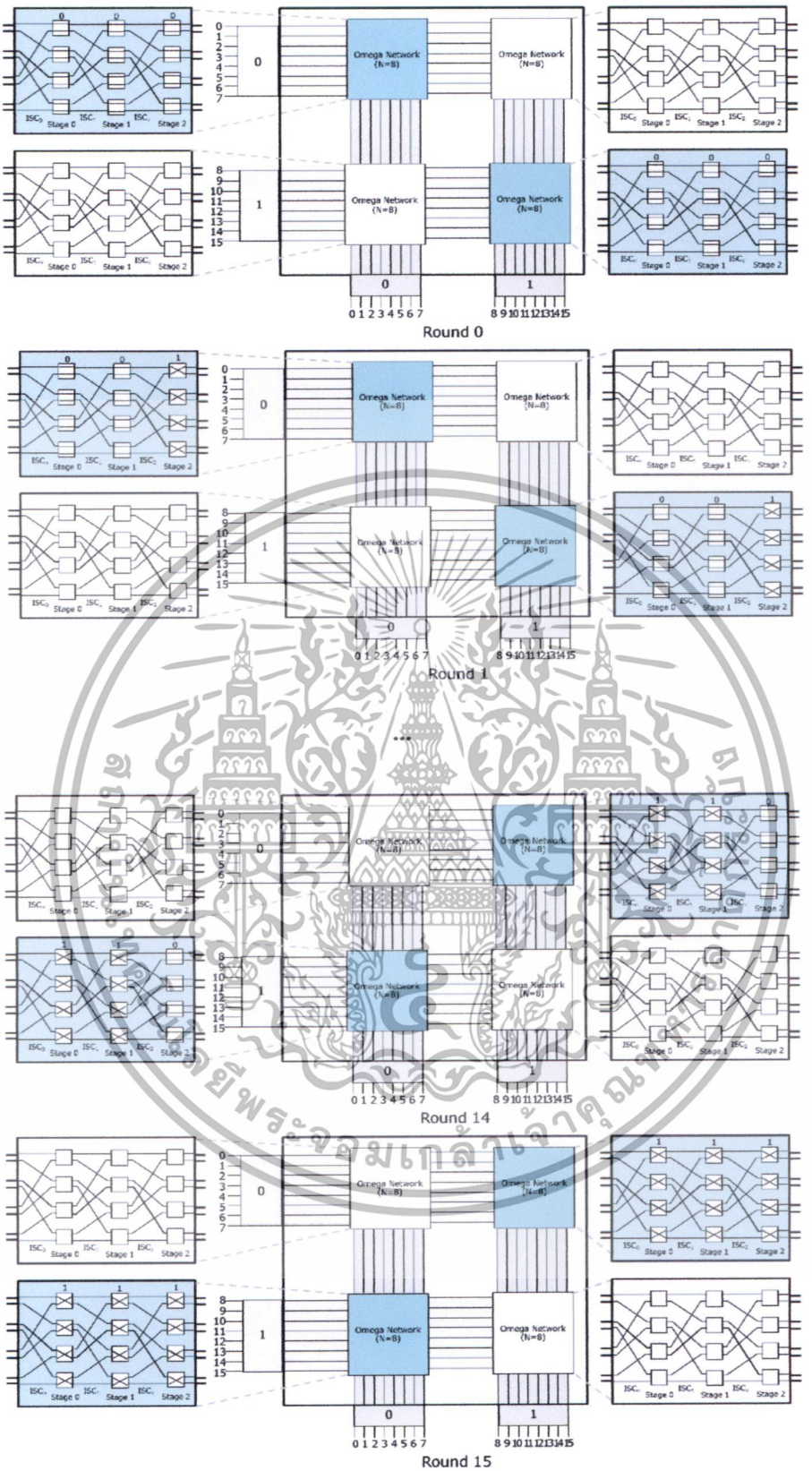
(ข)

รูปที่ 3.9 ตัวอย่างของผลลัพธ์ของการติดต่อสื่อสาร ATAPE บนเครือข่ายแบบ MINs+ ประกอบด้วย (ก) เครือข่ายแบบบัตเตอร์ฟลายพลัส (N=8, d=2) (ข) เครือข่ายแบบเบสไลน์พลัส (N=8, d=2) และ (ค) เครือข่ายแบบ MINs+ ที่ใช้สวิทช์ขนาด $d \times d$ (N=16, d=4)



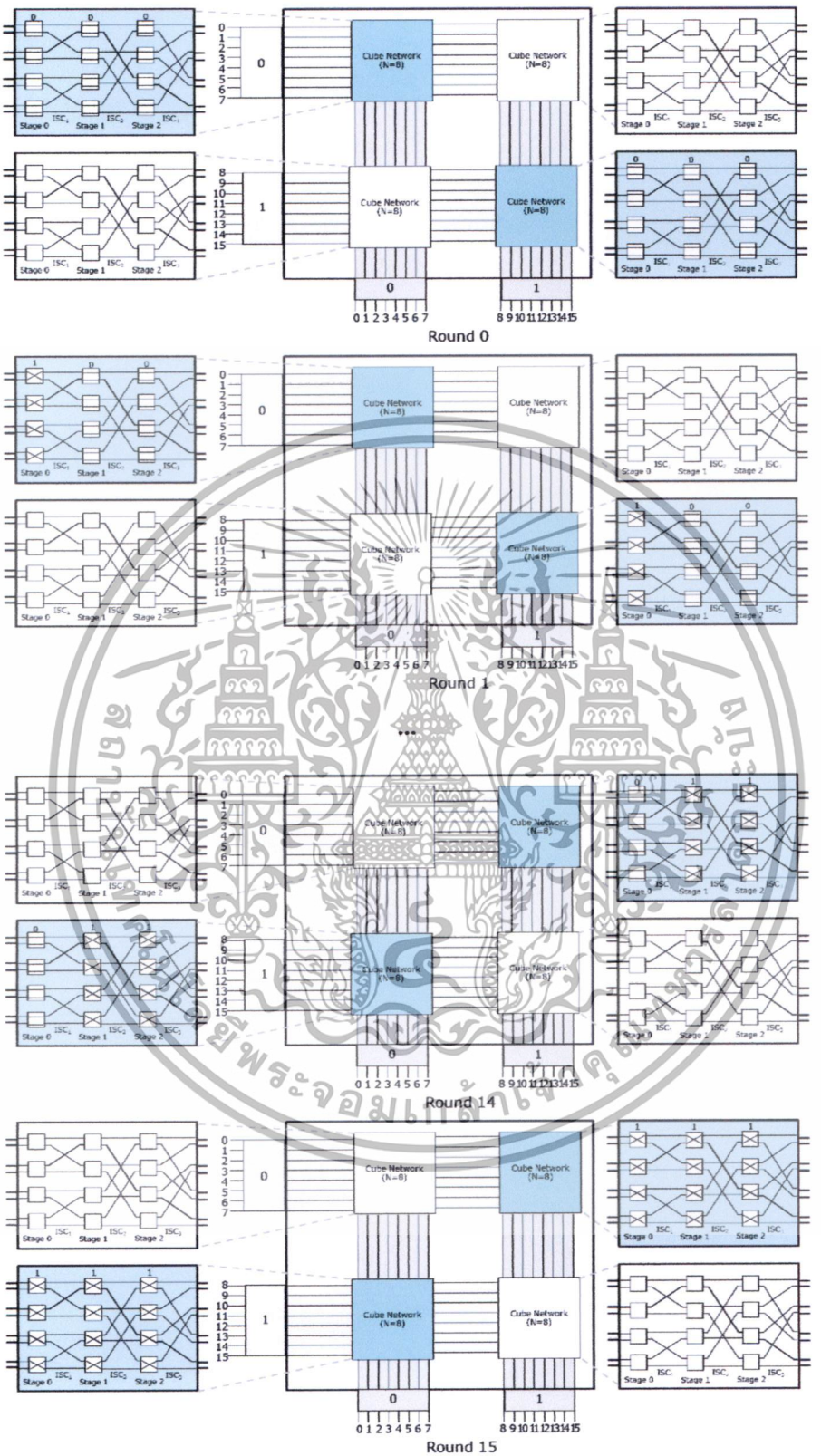
(ค)

รูปที่ 3.9 (ต่อ)



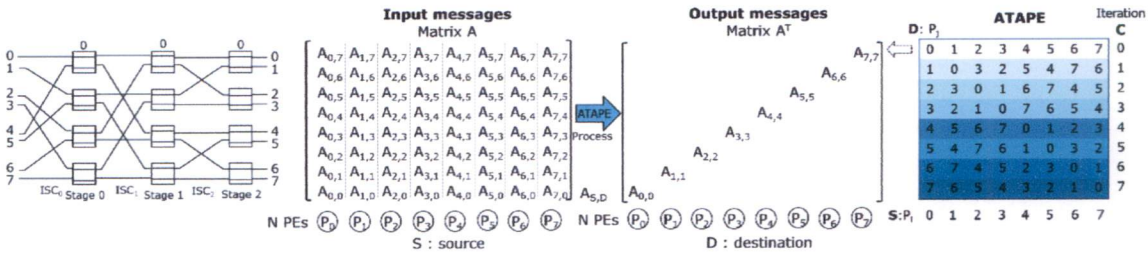
(ก)

รูปที่ 3.10 ตัวอย่างของผลลัพธ์ของการติดต่อสื่อสาร ATAPE บน CMINs^{+} ประกอบด้วย (ก) ครอสส์บาร์ของเครือข่ายแบบโอเมก้า และ (ข) ครอสส์บาร์ของเครือข่ายแบบบันยานพลัส (N=16, x=2 และ d=2) และดีกรีของครอสส์บาร์ = 2 x 2

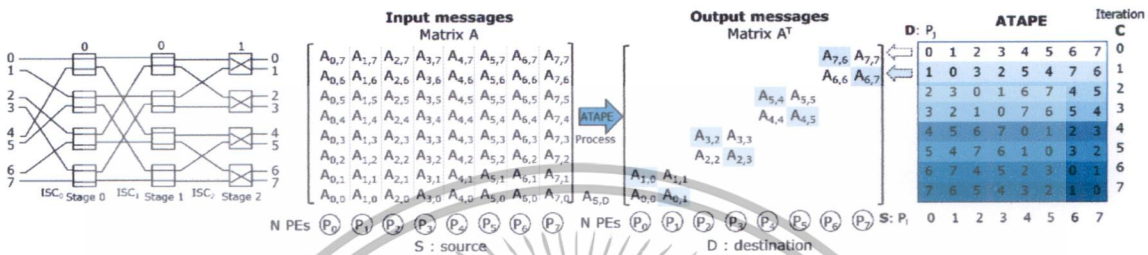


(ข)

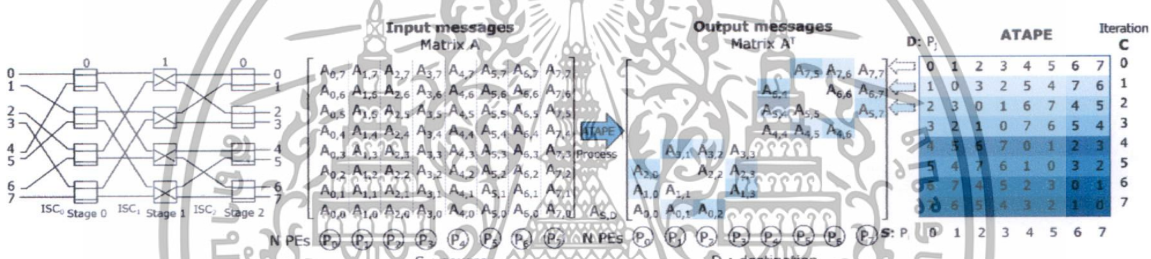
รูปที่ 3.10 (ต่อ)



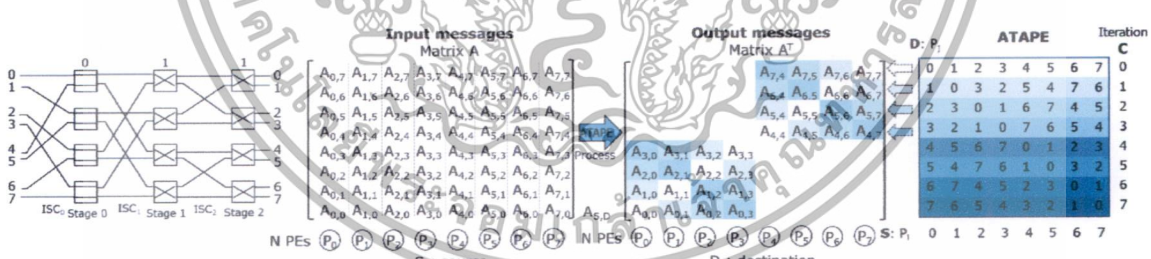
Round 0



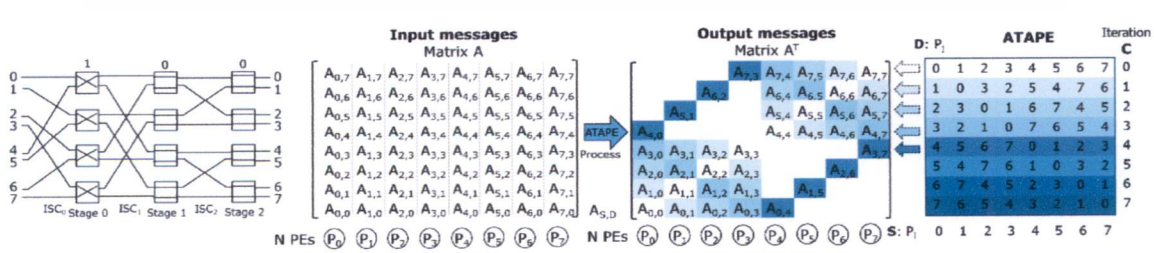
Round 1



Round 2

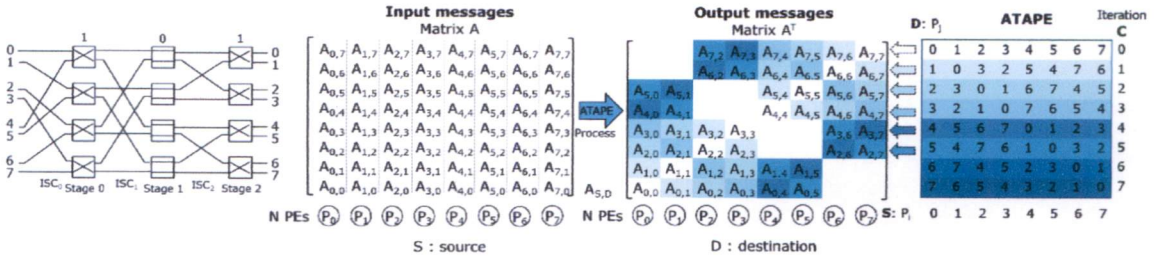


Round 3

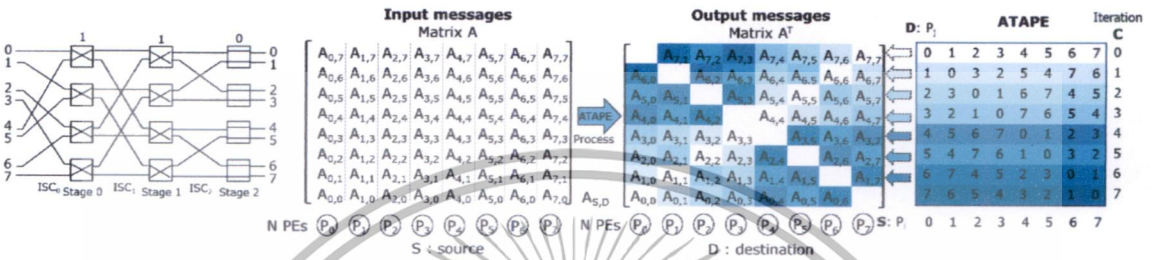


Round 4

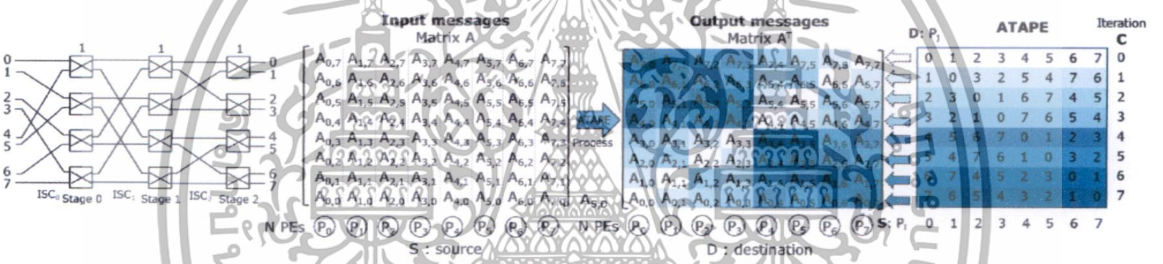
รูปที่ 3.11 ตัวอย่างของผลลัพธ์การทรานสโพสเมตริกซ์แบบขนาน (N = 8 รอบ) ด้วยการติดต่อสื่อสาร ATAPE $D = S \oplus C$ โดยที่ N = 8 และ C = 0, 1, 2, 3, ..., 7



Round 5



Round 6



Round 7

รูปที่ 3.11 (ต่อ)

บทที่ 4
การจัดสรรงานแบบซุปเปอร์ไปป์ไลน์ของการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ที่ดีที่สุดบนเครือข่ายการเชื่อมต่อแบบมัลติสเตจพลัสที่สามารถแบ่งกลุ่มย่อยได้

ในบทนี้จะนำเสนอการจัดสรรงานแบบซุปเปอร์ไปป์ไลน์ของการแลกเปลี่ยนแบบออล-ทู-ออลเพอร์ซันนัลไลซ์ (ATAPE) แบบดีที่สุดบนเครือข่ายการเชื่อมต่อแบบมัลติสเตจพลัสที่สามารถแบ่งกลุ่มย่อยได้ (VA-DE: Valuable ATAPE with Dynamic Embedding and Super-pipeline Scheduling on Partitionable Multistage Interconnection Networks^{+}) [10] ถึงแม้ว่าการติดต่อสื่อสารแบบ ATAPE บนเครือข่ายแบบ CMINs^{+} (Crossbar of MINs^{+}) แบบเดิมจะเร็วขึ้นแต่การสร้างเครือข่ายมีราคาค่อนข้างแพง แล้วเครือข่ายยังถูกใช้เพื่อประมวลผลของงานเพียง $1/x$ ส่วนของเครือข่ายทั้งหมดเท่านั้น ซึ่งทำให้เครือข่ายบางส่วนไม่ถูกใช้งานนั่นคือใช้งานเครือข่ายได้ไม่เต็มประสิทธิภาพไม่ถึง 100% ของเครือข่ายทั้งหมดไม่เหมือนกับการติดต่อสื่อสารแบบ ATAPE บนเครือข่ายแบบ MINs^{+} เดิมที่ใช้เครือข่ายทั้งเครือข่าย นอกจากนั้นคุณสมบัติที่สำคัญอีกข้อหนึ่งก็คือเมื่อทำการขยายเครือข่ายแล้วจะต้องสามารถประมวลผลได้หลายๆ งานพร้อมๆ กันในเวลาเดียวกัน (โดยที่งานอาจมีขนาดแตกต่างกันได้ $\leq N$) ซึ่งวิธีการของฟังก์ชัน ATAPE แบบเดิมบนเครือข่ายแบบ MINs [3, 5, 6, 16, 17] เครือข่ายแบบ MINs^{+} [9] และเครือข่ายแบบ CMINs^{+} [9] สามารถประมวลผลงานแบบ ATAPE ได้เพียงแค่งานเดียว ณ เวลาหนึ่งๆ เท่านั้น

ตารางที่ 4.1 ประโยชน์ของฟังก์ชัน ATAPE แบบใหม่บนเครือข่ายแบบ MINs^{+} ที่สามารถแบ่งได้

การแบ่งฟังก์ชัน ATAPE และการจัดสรรงาน	MINs [1, 4, 8,15]	MINs^{+} [9]	x^2 Crossbar of MINs^{+} [9]	x^2 Crossbar of partitionable MINs^{+}
	$O(N+\log_2 N)$		$O(N+\log_2 N/x)$	$O(N/x+\log_2 N/x)$
	Switch-pattern ATAPE	ATAPE [9] $D = S \oplus (C + \text{order}) \% N$ $D = \rho\,[(S+C+\text{order}) \% N]$		New pATAPE $D^y = S^y \oplus C_i^y$ $C_i^y = (yN/x + l) \bmod N$
ความล่าช้าของการติดต่อสื่อสาร (Communication Delay)	n-stage	n-stage	n'-stage $< n$	n'-stage $< n$
เครือข่ายที่สามารถแบ่งได้	-	-	-	✓
ฟังก์ชัน ATAPE ที่สามารถแบ่งได้	-	-	-	✓
การจัดสรรงาน (Scheduling Tasks)	one	one	one	many
การใช้งานระบบ (System Utilization)	100%	100%	$(1/x)$100%	100%
CPI (สัญญาณนาฬิกาต่อคำสั่ง)	1	1	1	$1/x$

ตารางที่ 4.1 แสดงประโยชน์ของ ATAPE เดิมบนเครือข่ายแบบ MINs$^+$ เพื่อให้เห็นถึงข้อจำกัด (เช่น ณ เวลาหนึ่งๆ งานเพียงหนึ่งงานถูกประมวลผลเท่านั้น สามารถใช้งานเครือข่ายย่อยๆ MINs$^+$ ได้น้อยกว่า 100% ของเครือข่ายแบบ CMINs$^+$ เป็นต้น (การศึกษาวิจัยนี้จึงได้เสนอฟังก์ชัน ATAPE แบบฝังบนชิปแบบใหม่บนเครือข่ายแบบ PMINs$^+$ คือฟังก์ชัน $D^y = S^y \oplus C_i^y$ (โดยที่ C_i^y = ($yN/x + l$) mod N กำหนดให้ $0 \le y < x$ และ $0 \le l < N$) ซึ่งจะสามารถใช้ระบบได้ 100% โดยที่ *p*ATAPE แบบใหม่จะใช้เทคนิคซุปเปอร์ไปป์ไลน์เพื่อให้สามารถใช้งานระบบได้พร้อมๆ กันหลายๆ งานของขนาดที่มีความยืดหยุ่นคือ N'' (= N, $N/2$, $N/4$, ..., $N/2^t$, ..., N/x) ในการจัดสรรงานที่ดีที่สุด (ที่จัดสรรได้ครบทั้ง 3 ปัจจัย) คือสามารถจัดลำดับงานย่อยให้เหมาะกับกลุ่มย่อยในเวลาที่เหมาะสม (Right task, Right section, Right time) ผลลัพธ์ของ *p*ATAPE บนเครือข่ายแบบ PMINs$^+$ จะสามารถส่งข้อมูล ได้เร็วกว่าแบบ *c*ATAPE บนเครือข่ายแบบ CMINs$^+$ [9] ซึ่งมีค่า CPI = $1/x$ จำนวนสัญญาณนาฬิกาต่อคำสั่ง (Clock per Instruction : CPI) ด้วยการทำงานของ xN หน่วยประมวลผลเสมือน (PEs) ภายใต้ N หน่วยประมวลผลที่มีอยู่จริง

เนื้อหาในบทที่ 4 นี้จะมีการแบ่งออกเป็น 3 ส่วนดังต่อไปนี้ ในหัวข้อที่ 4.1 เสนอสถาปัตยกรรมของ PMIN$^+$ ที่สามารถแบ่งได้ ในหัวข้อที่ 4.2 เสนอฟังก์ชัน *p*ATAPE แบบฝังบนชิปที่ยืดหยุ่นบนเครือข่ายแบบ PMINs$^+$ และสำหรับหัวข้อที่ 4.3 แสดงประสิทธิภาพของการทำงานแบบ ATAPE หลายๆ งานบน x x x ครอสส์บาร์ของ PMINs$^+$ ที่สามารถแบ่งกลุ่มย่อยได้ (Crossbar of Partitionable MINs$^+$)

4.1 สถาปัตยกรรมของเครือข่ายแบบ MIN$^+$ ที่สามารถแบ่งกลุ่มย่อยได้

ในหัวข้อนี้นำเสนอการเพิ่มประสิทธิภาพของ MIN$^+$ ที่สามารถแบ่งเป็นกลุ่มย่อยๆ ได้ (Partitionable MIN$^+$: PMIN$^+$) โดยได้ปรับปรุงจากครอสส์บาร์ของ MINs$^+$ (CMINs$^+$) [9] ซึ่งใช้ I/O สวิทช์ควบคุมแบบครอสส์ (I/O Cross-Control Switches) ร่วมกับการจัดสรรงานที่ดีที่สุด(ที่จัดสรรได้ครบทั้ง 3 ปัจจัย) คือสามารถจัดลำดับงานย่อยให้เหมาะกับกลุ่มย่อยในเวลาที่เหมาะสม ในหัวข้อที่ 4.1.1 เสนอการปรับพอร์ตของเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด d x d ในหัวข้อที่ 4.1.2 เสนอ x x x ครอสส์บาร์ของ PMINs$^+$ ($N = 2^n$ และ $x = 2^b$) ซึ่งได้เพิ่ม I/O สวิทช์ควบคุมแบบครอสส์เพื่อให้สามารถควบคุมสัญญาณเข้าหรือออกระบบย่อยได้อย่างอิสระ พร้อมทั้งใช้เทคนิคซุปเปอร์ไปป์ไลน์ร่วมกับการจัดสรรงานที่ดีที่สุด และในหัวข้อที่ 4.1.3 เสนอการใช้ระบบได้เต็มประสิทธิภาพ 100% ของเครือข่ายแบบ PMINs$^+$ และจัดตารางงานได้อย่างมีประสิทธิภาพ

4.1.1 การปรับพอร์ตของเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด d x d

สำหรับเครือข่ายแบบ PMINs$^+$ มีองค์ประกอบหลักคือเครือข่าย MINs แบบสมบูรณ์ (MINs$^+$) ซึ่งประกอบด้วยเครือข่ายแบบโอเมก้า [4] เครือข่ายแบบฟลิบ [1] เครือข่ายแบบคิวบ์ [8] เครือข่ายแบบอาร์ [7] และเครือข่ายแบบ MINs$^+$ [9] (เช่น เครือข่ายบันยานพลัส เครือข่ายแบบบัตเตอร์ฟลายพลัส เครือข่ายแบบเบสไลน์พลัส และเครือข่ายแบบอินเวอร์สเบสไลน์พลัส) ในเครือข่าย MINs แบบเดิม [1, 4, 7, 8, 15] และเครือข่ายแบบ MINs$^+$ แบบใหม่ [9] จะใช้สวิทช์ขนาด 2x2 เป็นหลัก (ในรูปที่ 4.1) ในบทนี้จะกล่าวถึงส่วนเครือข่ายแบบ MINs$^+$ โดยทั่วไปที่ใช้สวิทช์ขนาด dxd (ในรูปที่ 4.2 สำหรับ d = 4, k = 2) และได้พิสูจน์ความถูกต้องของการจัดเส้นทางภายในด้วยต้นทางและปลายทาง ((S, D) Self-Routing) ดังต่อไปนี้

<u>เครือข่ายแบบโอเมก้า ที่ใช้สวิทช์ขนาด 4x4</u> (4-nary Omega Network) รูปแบบการเชื่อมต่อระหว่างสเตจ (Inter Stage Connection : ISC) ของจำนวนสเตจ $m=\log_4 N$ สเตจ คือการเลื่อนซ้าย 2 บิตแล้ววนกลับ ในการจัดเส้นทางของหน่วยประมวลผล (I, J) จากต้นทาง $I = (i_{n-1}\ i_{n-2}...i_2 i_1 i_0)$ ไปยังปลายทาง $J = (j_{n-1}j_{n-2}\ ...\ j_2 j_1 j_0)$ ส่งข้อมูลผ่าน m ISCs และ m สวิทช์ (ใช้พอร์ตเข้า I (in-port : $i_{t+1}i_t$) ไปยังพอร์ตออก J (out-port : $j_{t+1}j_t$)) ได้ดังนี้

ISC_0-sw: $\mathbf{i_{n-1}i_{n-2}}$ .. $i_3 i_2\ i_1 i_0$ ⟶ $i_{n-3}i_{n-4}$ .. $i_1 i_0\ \mathbf{j_{n-1}j_{n-2}}$

ISC_1-sw: $\mathbf{i_{n-3}i_{n-4}}..i_1 i_0\ j_{n-1}j_{n-2}$ ⟶ $i_{n-5}i_{n-6}$ .. $i_1 i_0\ j_{n-1}j_{n-2}\ \mathbf{j_{n-3}j_{n-4}}$

...

ISC_{m-1}-sw: $\mathbf{i_1 i_0}\ j_{n-1}j_{n-2}$ .. $j_3 j_2$ ⟶ $j_{n-1}j_{n-2}$ .. $j_3 j_2\ \mathbf{j_1 j_0}$

<u>เครือข่ายแบบฟลิบ ที่ใช้สวิทช์ขนาด 4x4</u> (4-nary Flip Network) จะใช้รูปแบบการเชื่อมต่อระหว่างสเตจ (ISC) แบบเลื่อนขวา 2 บิตแล้ววนกลับ สำหรับเชื่อมระหว่าง $m = \log_4 N$ สเตจ โดยที่สามารถจัดเส้นทางภายในได้ถูกต้องจากต้นทาง I ไปยังปลายทาง J พร้อมการปรับสวิทช์ในแต่ละสเตจได้ดังนี้

sw-ISC_1: $i_{n-1}i_{n-2}$ .. $i_3 i_2\ \mathbf{i_1 i_0}$ ⟶ $\mathbf{j_1 j_0}\ i_{n-1}i_{n-2}$ .. $i_5 i_4\ i_3 i_2$

sw-ISC_2: $j_1 j_0\ i_{n-1}i_{n-2}$ .. $\mathbf{i_3 i_2}$ ⟶ $j_3 \mathbf{j_2}\ j_1 j_0\ i_{n-1}i_{n-2}$ .. $i_5 i_4$

...

sw-ISC_m: $j_{n-3}j_{n-4}$ .. $j_1 j_0\ \mathbf{i_{n-1}i_{n-2}}$ ⟶ $j_{n-1}\mathbf{j_{n-2}}\ j_{n-3}j_{n-4}$ .. $j_3 j_2\ j_1 j_0$

<u>เครือข่ายแบบบันยานพลัส ที่ใช้สวิทช์ขนาด 4x4</u> (4-nary Banyan$^+$ Network) ใช้รูปแบบของ ISC แตกต่างกันในแต่ละคู่ของสเตจ (ISC_1, ISC_2, ... , ISC_s, ... , ISC_{m-1}) สำหรับเชื่อมต่อระหว่าง m สเตจ กำหนดให้ s = 1, 2, ... , m-1 สำหรับ ISC_s คือการอินเวอร์ระหว่างคู่ของตำแหน่ง $2(s+1)$ บิต โดยที่ ISC_m สุดท้าย (เลื่อนขวา 2 บิตแบบวนกลับ) เป็นการปรับพอร์ตทีหลัง (Post-Port Adjusting) จากสเตจที่ m-1 ไปยังส่วนนำออก สำหรับเครือข่ายแบบบันยานพลัสการจัดเส้นทางภายในได้ถูกต้องจากต้นทาง $I = (i_{n-1}i_{n-2}\ ...\ i_2 i_1 i_0)$ ไปยังปลายทาง $J = (j_{n-1}j_{n-2}...j_2 j_1 j_0)$ พร้อมการปรับสวิทช์ในแต่ละสเตจดังต่อไปนี้

sw-ISC_1: $i_{n-1}i_{n-2}$ .. $(i_3 i_2)\ (\mathbf{i_1 i_0})$ ⟶ $i_{n-1}i_{n-2}$ .. $(j_1 j_0)\ (i_3 i_2)$

sw-ISC_2: $i_{n-1}i_{n-2}..(i_5 i_4)\ j_1 j_0\ (\mathbf{i_3 i_2})$ ⟶ $i_{n-1}i_{n-2}\ ..(\mathbf{j_3 j_2})\ j_1 j_0\ (i_5 i_4)$

...

sw-ISC_{m-1}: $(i_{n-1}i_{n-2})..j_1 j_0\ (\mathbf{i_{n-3}i_{n-4}})$ ⟶ $(\mathbf{j_{n-3}j_{n-4}})..j_1 j_0\ (i_{n-1}i_{n-2})$

[sw-ISC_m: $j_{n-3}j_{n-4}$ .. $j_1 j_0\ \mathbf{i_{n-1}i_{n-2}}$ ⟶ $\mathbf{j_{n-1}j_{n-2}}\ j_{n-3}j_{n-4}$ .. $j_3 j_2\ j_1 j_0$]

<u>เครือข่ายแบบบัตเตอร์ฟลายพลัส ที่ใช้สวิทช์ขนาด 4x4</u> (4-nary Butterfly$^+$ Network) รูปแบบของ ISC_s ในเครือข่ายแบบบัตเตอร์ฟลายเดิม คือการอินเวอร์คู่ของตำแหน่งที่ n-$2(s$-1) บิต โดยที่ s = 1, 2, ... , m-1 และในเครือข่ายแบบบัตเตอร์ฟลายพลัสได้ปรับพอร์ตที่ ISC_0 (เลื่อนซ้าย 2 บิตแบบวนกลับ) เรียกว่าเป็นการปรับพอร์ตก่อน (Pre-Port Adjusting) ซึ่งจะสามารถจัดเส้นทางจากต้นทาง I ไปยังปลายทาง J ในแต่ละ ISC พร้อมการปรับสวิทช์ในแต่ละสเตจให้ถูกต้องได้ดังนี้

[ISC_0-sw $\mathbf{i_{n-1}i_{n-2}}i_{n-3}i_{n-4}$ .. $i_3 i_2 i_1 i_0$ ⟶ $i_{n-3}i_{n-4}$ .. $i_3 i_2 i_1 i_0 \mathbf{j_{n-1}j_{n-2}}$]

ISC_1-sw: $(i_{n-3}i_{n-4})..\ i_1i_0(j_{n-1}j_{n-2}) \longrightarrow (j_{n-1}j_{n-2})\ ..\ i_3i_2i_1i_0(j_{n-3}j_{n-4})$

ISC_2-sw: $j_{n-1}j_{n-2}(i_{n-5}i_{n-6})..i_1i_0(j_{n-3}j_{n-4}) \longrightarrow j_{n-1}j_{n-2}(j_{n-3}j_{n-4})..i_1i_0(j_{n-5}j_{n-6})$

...

ISC_{m-2}-sw: $j_{n-1}j_{n-2}\ ..\ (i_3i_2)i_1i_0(j_5j_4) \longrightarrow j_{n-1}j_{n-2}\ ..\ (j_5j_4)\ i_1i_0\ (j_3j_2)$

ISC_{m-1}-sw: $j_{n-1}j_{n-2}\ ..\ j_5j_4\ (i_1i_0)\ (j_3j_2) \longrightarrow j_{n-1}j_{n-2}\ ..\ j_5j_4\ (j_3j_2)\ (j_1j_0)$

<u>เครือข่ายแบบเบสไลน์พลัส ที่ใช้สวิทช์ขนาด 4x4</u> (4-nary Baseline$^+$ Network) รูปแบบของ ISC (ISC_1 - ISC_{m-1}) คือการออกแบบให้เป็นแบบรีเคอร์ซีฟ หรือเพื่อเชื่อมระหว่างสเตจที่ s-1 และ s (s = 1, 2, ... , m-1) โดยรูปแบบของ ISC_s คือการเลื่อนขวา 2 บิตแบบวนกลับ (ภายใน n-2(s-1) บิต) และเพิ่ม ISC_m พิเศษ (การกลับตำแหน่งบิตแบบเป็นคู่) เรียกว่าเป็นการปรับพอร์ตทีหลัง (Post - Port Adjusting) ดังนั้นการจัดเส้นทางภายในที่ถูกต้องของเครือข่ายแบบเบสไลน์พลัสจากต้นทาง I ไปยังปลายทาง J พร้อมการปรับสวิทช์ในแต่ละสเตจดังต่อไปนี้

sw-ISC_1: $(i_{n-1}i_{n-2}\ ..\ i_3i_2\ i_1i_0) \longrightarrow (j_1j_0\ i_{n-1}i_{n-2}\ ..\ i_5i_4\ i_3i_2)$

sw-ISC_2: $j_1j_0\ (i_{n-1}i_{n-2}\ ..\ i_5i_4\ i_3i_2) \longrightarrow j_1j_0\ (j_3j_2\ i_{n-1}i_{n-2}\ ..\ i_5i_4)$

...

sw-ISC_{m-1}: $j_1j_0\ j_3j_2..(i_{n-1}i_{n-2}i_{n-3}i_{n-4}) \longrightarrow j_1j_0\ j_3j_2..(j_{n-3}j_{n-4}\ i_{n-1}i_{n-2})$

[sw-ISC_m: $(j_1j_0)(j_3j_2)..(j_{n-3}j_{n-4})(i_{n-1}i_{n-2}) \longrightarrow (j_{n-1}j_{n-2})..(j_3j_2)(j_1j_0)$]

<u>เครือข่ายแบบอินเวอร์สเบสไลน์พลัส ที่ใช้สวิทช์ขนาด 4x4</u> (4-nary Inverse Baseline$^+$ Network)สร้างโดยใช้รูปแบบของ ISC_s (การเลื่อนซ้าย 2 บิตแบบวนกลับใน 2(s+1) บิตโดยที่ s = 1, 2, .. , m-1) และ ISC_0 ที่ถูกปรับพอร์ตก่อน (คือ การกลับตำแหน่งบิตทีละ 2 บิตเป็นคู่ๆ) เพื่อจัดเส้นทางที่ถูกต้องจากต้นทาง I ไปยังปลายทาง J พร้อมการปรับสวิทช์ในแต่ละสเตจดังนี้

[ISC_0-sw $(i_{n-1}i_{n-2})(i_{n-3}i_{n-4})..(i_3i_2)(i_1i_0) \longrightarrow (i_1i_0)(i_3i_2)..(i_{n-3}i_{n-4})(j_{n-1}j_{n-2})$]

ISC_1-sw: $i_1i_0\ i_3i_2\ ..\ (i_{n-3}i_{n-4}\ j_{n-1}j_{n-2}) \longrightarrow i_1i_0\ i_3i_2\ ..\ (j_{n-1}j_{n-2}\ j_{n-3}j_{n-4})$

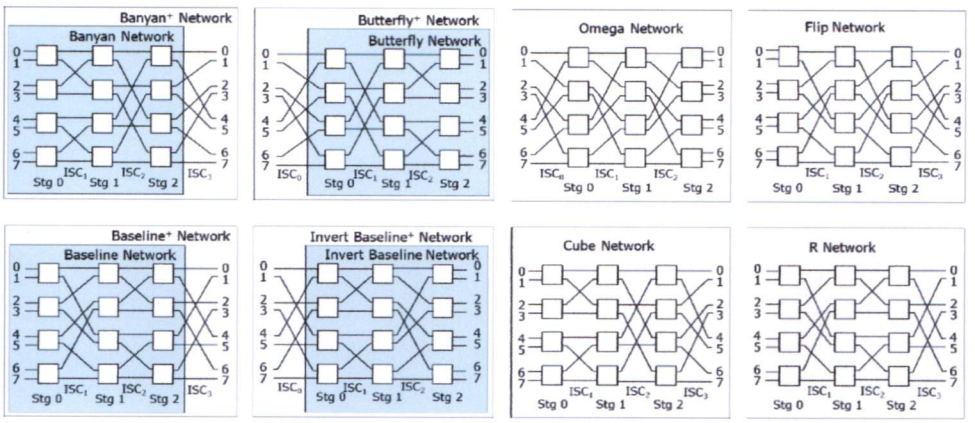
ISC_2-sw: $i_1i_0\ i_3i_2..(i_{n-5}i_{n-6}j_{n-1}j_{n-2}\ j_{n-3}j_{n-4}) \longrightarrow i_1i_0\ i_3i_2..(j_{n-1}j_{n-2}\ j_{n-3}j_{n-4}j_{n-5}j_{n-6})$

...

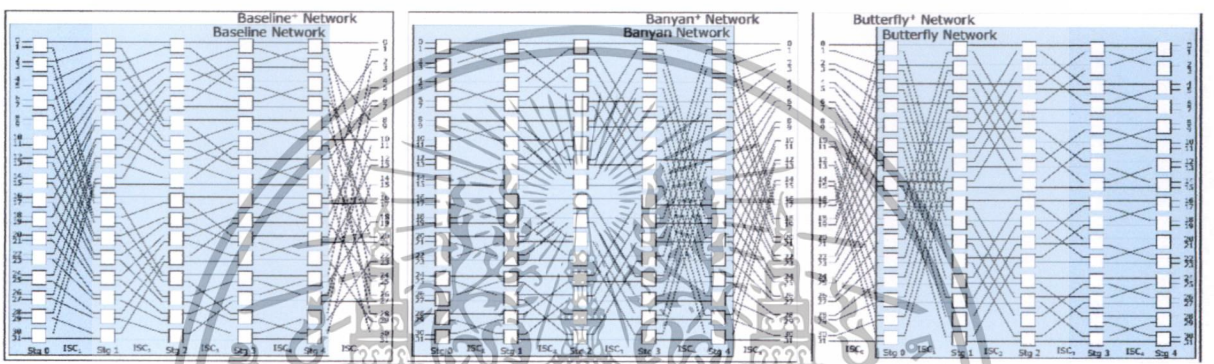
ISC_{m-2}-sw: $i_1i_0\ (i_3i_2\ j_{n-1}j_{n-2}\ ..\ j_5j_4) \longrightarrow i_1i_0\ (j_{n-1}j_{n-2}\ ..\ j_5j_4\ j_3j_2)$

ISC_{m-1}-sw: $(i_1i_0\ j_{n-1}j_{n-2}\ ..\ j_5j_4\ j_3j_2) \longrightarrow (j_{n-1}j_{n-2}\ ..\ j_5j_4\ j_3j_2\ j_1j_0)$

ตัวอย่างโครงสร้างพื้นฐานของเครือข่ายแบบ MINs$^+$ ที่ใช้สวิทช์ขนาด 2x2 ในการเชื่อมต่อระหว่างสเตจ ดังในรูปที่ 4.1(ก) กำหนดให้ N = 8 มีจำนวนสเตจ n = $\log_2 N$ = 3 สเตจ และมีจำนวนสวิทช์ต่อสเตจ $N/2$ = 4 สวิทช์ และรูปที่ 4.1(ข) สำหรับเครือข่ายแบบ MINs$^+$ ที่กำหนดให้ N = 32 (มีจำนวนสเตจ n = $\log_2 N$ = 5 สเตจ และมีจำนวนสวิทช์ต่อสเตจ $N/2$ = 16 สวิทช์) ในรูปที่ 4.2 แสดงโครงสร้างของเครือข่ายแบบ MINs$^+$ โดยใช้สวิทช์ขนาด 4x4 (N = 64 จำนวนสเตจ m = $\log_4 N$ = 4 สเตจและมีจำนวนสวิทช์ต่อสเตจ $N/4$ = 16 สวิทช์ต่อสเตจ)

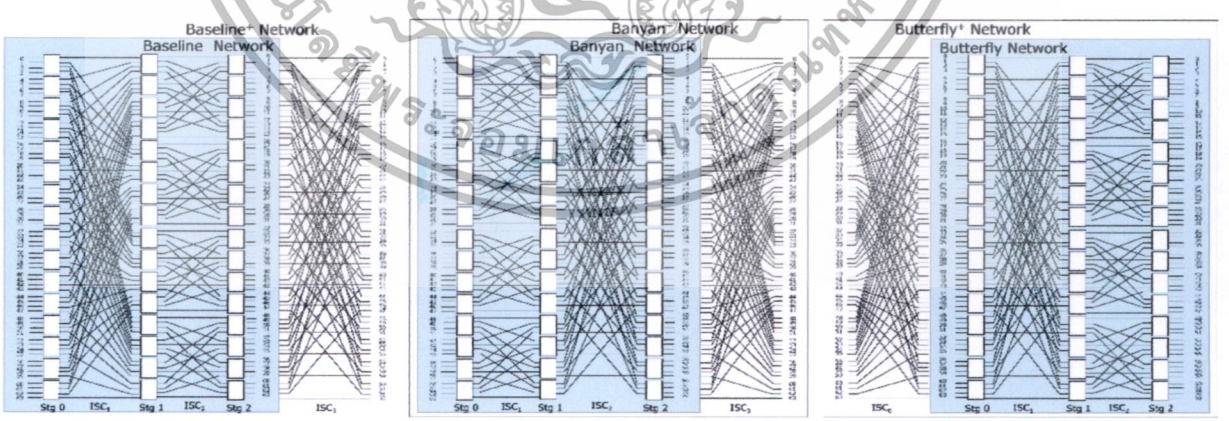


(ก)



(ข)

รูปที่ 4.1 ตัวอย่างการใช้สวิทช์ขนาด 2x2 ของ MINs แบบสมบูรณ์ประกอบด้วย (ก) เครือข่ายแบบโอเมก้า ฟลิบ คิวบ์ อาร์ บัตเตอร์ฟลายพลัส บันยานพลัส เบสไลน์พลัส และอินเวอร์สเบสไลน์พลัส โดยกำหนดให้ $N = 8$ และ $d = 2$ (ข) เครือข่ายแบบเบสไลน์พลัส บันยานพลัส และบัตเตอร์ฟลายพลัส กำหนดให้ $N = 32$ และ $d = 2$

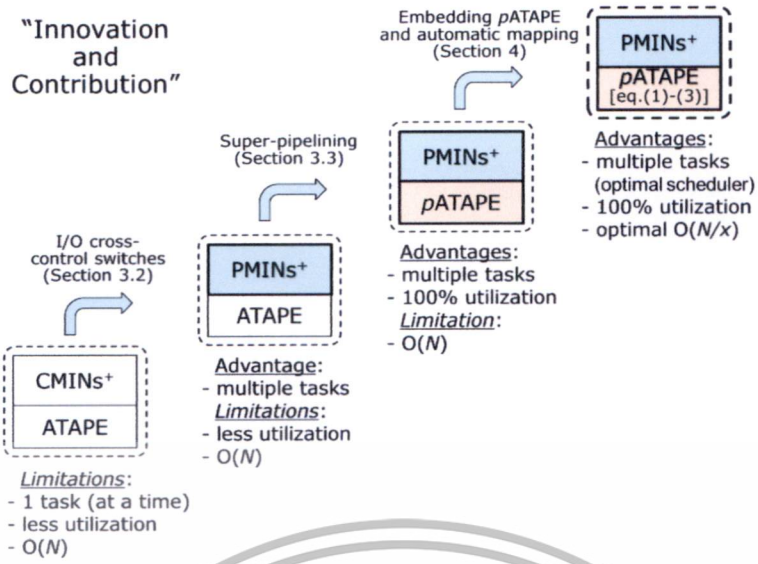


รูปที่ 4.2 ตัวอย่างของการใช้สวิทช์ขนาด 4x4 บนเครือข่ายแบบ MINs+ เช่นเครือข่ายแบบเบสไลน์พลัส บันยานพลัส และบัตเตอร์ฟลายพลัส โดยกำหนดให้ $N = 64$ และ $d = 4$

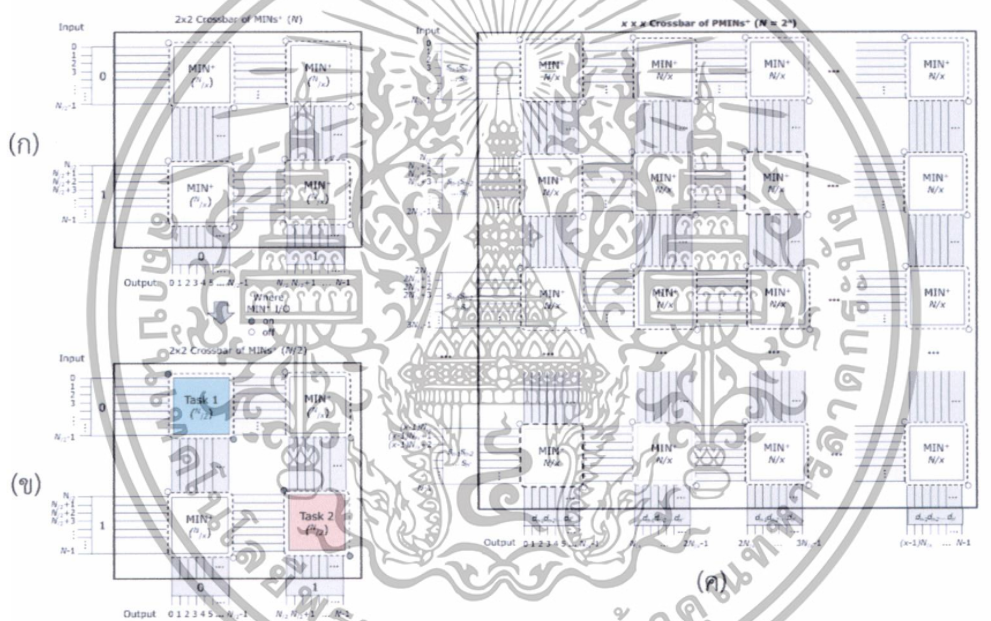
4.1.2 การใช้ I/O สวิทช์ควบคุมแบบครอสส์บนเครือข่ายแบบ PMINs$^+$

ในหัวข้อนี้นำเสนอ $x\ x\ x$ ครอสส์บาร์ของการแบ่ง PMINs$^+$ เป็นกลุ่มย่อย (Crossbar of Partitionable MINs$^+$: PMINs$^+$) ที่ได้ปรับปรุงเครือข่ายมาจากครอสส์บาร์ของ MINs$^+$ (Crossbar of MINs$^+$: CMINs$^+$) [9] เพื่อให้สามารถประมวลผลงานขนาดต่างๆ กันหลายๆ งานได้พร้อมกัน โดยที่งานมีขนาดต่างกันดังนี้ $N"= N/2^t$ (หรือ $N, N/2, ..., N/x$) ซึ่งได้ทำการเพิ่ม *I/O* สวิทช์ควบคุมแบบครอสส์ (I/O Cross-Control Switches) เพื่อกำหนดค่าระบบย่อยได้หลากหลายและเป็นอิสระ ให้สามารถใช้ระบบย่อยๆ ในเครือข่ายแบบ PMINs$^+$ ได้เต็มประสิทธิภาพใน 100% ในรูปที่ 4.3 จะแสดงให้เห็นถึงภาพรวมของนวัตกรรมและผลงานของวิทยานิพนธ์นี้ (เช่น เครือข่ายแบบ PMINs$^+$ และ ATAPE แบบใหม่ที่สามารถแบ่งย่อยได้แบบอัตโนมัติ) ซึ่งมีความซับซ้อนด้านเวลาในอุดมคติสำหรับแต่ละงานของ ATAPE (ของขนาด $\le N$) คือ $O(N/x + n')$ โดยใช้เทคนิคของซุปเปอร์ไปป์ไลน์ บนเครือข่ายแบบ PMINs$^+$ ซึ่งใช้เวลาน้อยกว่าบนเครือข่ายแบบ CMINs$^+$ ด้วยความซับซ้อนด้านเวลา $O(N + n')$ โดยที่กำหนดให้ระบบย่อย MIN$^+$ ประกอบด้วย $N' = N/x$ มีจำนวนสเตจ $n' = \log_2 N'$ สเตจ และมีจำนวนสวิทช์ต่อสเตจ $N'/2$ สวิทช์ เพื่อที่จะสามารถประมวลผลงานจำนวนงานหลายๆ งาน ขนาด $N" = N/2^t$ ในระบบเครือข่ายแบบ PMINs$^+$ สามารถแบ่งออกเป็น x กลุ่ม (Sections) โดยกำหนดให้กลุ่มย่อย $y = 0, 1, 2, ..., x\text{-}1$ เพื่อคำนวณ $N"/x$ รอบต่องาน (โดยที่ใช้ x ค่าควบคุมหลักบน x กลุ่มต่อรอบ) ด้วยความซับซ้อนด้านเวลา $O(N/x + n')$ สำหรับงานขนาด N และ $N"$ ดังแสดงการเปรียบเทียบในตารางที่ 4.2 ($x = 2, 4, 8, ...$)

ตัวอย่างของ I/O สวิทช์ควบคุมขนาด 2x2 บนเครือข่ายแบบ PMIN$^+$ (รูปที่ 4.4 (ก)) งาน ATAPE (N) สามารถประมวลผลได้ใน $N/2$ ขั้นตอน (2 รอบของค่าควบคุมหลัก C ต่อขั้นตอน) บนการแบ่งกลุ่มย่อย 2 กลุ่ม ของขนาด N กลุ่มย่อยแรกได้ทำการกำหนดค่า *I/O* ครอสส์สวิทช์ (I/O-Switch) 0-0 และ 1-1 และอีกกลุ่มย่อยหนึ่งก็ทำการกำหนด *I/O* ครอสส์สวิทช์ 0-1 และ 1-0 ตามฟังก์ชัน ATAPE ที่ใช้ตัวดำเนินการแบบ XOR คือฟังก์ชัน $D = S \oplus (C + order) \bmod N$ ในกรณีที่มีจำนวนรอบ N รอบของค่าควบคุมหลัก (C) ได้ทำการเรียงลำดับใหม่ (เช่น $C = 0, N/2, 1, N/2+1, 2, N/2+2, ..., N/2\text{-}1, N\text{-}1$) เพื่อที่จะสามารถใช้ระบบย่อยได้แตกต่างและพร้อมกันโดยการเริ่มในทุกๆ $1/x$ สัญญาณนาฬิกา (สำหรับการใช้งานระบบได้ 100%) โดยได้แสดงการพิสูจน์ในทฤษฎีบทที่ 4.1 – 4.2 ปกติการใช้ค่าควบคุมหลัก (ของตัวดำเนินแบบ XOR) $C = 0, 1, 2, 3, ..., N/2\text{-}1$ จะสามารถจัดสรรให้ระบบย่อย 0-0 และ 1-1 และค่าควบคุมหลัก C ที่เหลือ $C = N/2, N/2+1, N/2+2, ..., N\text{-}1$ จะจัดสรรให้ระบบย่อยที่เหลือ คือ 0-1 และ 1-0 ดังนั้นทั้ง 2 กลุ่มย่อยสามารถกำหนดให้ทำการประมวลผลงานแบบ ATAPE จำนวน 2 งานขนาด N แบบสลับกันไป โดยใช้ค่าควบคุมหลักที่จัดลำดับให้เหมาะสมกับงานนั้นๆ ในทำนองเดียวกันสำหรับงานขนาด $N/2$ (ดังรูปที่ 4.4 (ข)) ทั้ง 2 งานจะสามารถประมวลผลบน 2 ระบบย่อยของ PMINs$^+$ ของแต่ละกลุ่มย่อยได้ในเวลาเดียวกัน รูปที่ 4.4 (ค) แสดงโครงสร้างของ $x\ x\ x$ ครอสส์บาร์ของ PMINs$^+$ (ของขนาด N) ประกอบด้วย x^2 ของระบบย่อย MIN$^+$ (ของขนาด $N' = N/x$) สำหรับประมวลผลงานหลายๆ งาน (ของขนาด $N" = N, N/2, ... N/2^t, ...,$ หรือ N/x) โดยจัดสรรการใช้ระบบย่อยให้เต็มประสิทธิภาพ 100% ด้วยการวนใช้งานของแต่ละกลุ่มตามเข็มนาฬิกา



รูปที่ 4.3 ภาพรวมของนวัตกรรมและผลงานของวิทยานิพนธ์นี้

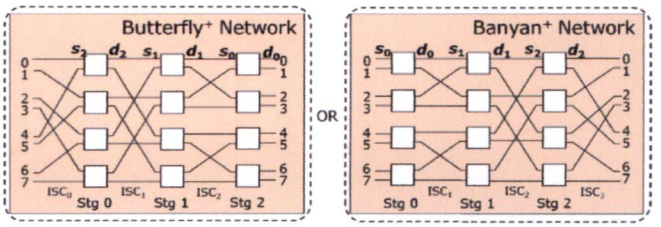


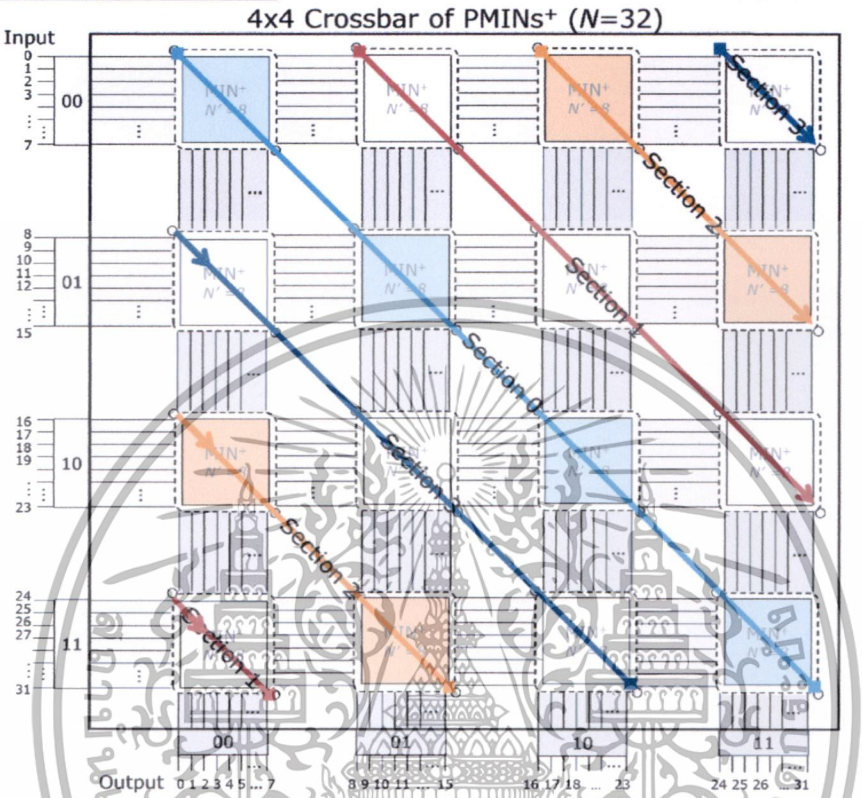
รูปที่ 4.4 (ก) 2x2 ครอสส์บาร์ของ PMINs$^+$ สำหรับการแบ่งกลุ่มย่อยร่วมกับ I/O สวิทช์ควบคุมแบบครอสส์และ (ข) การกำหนดระบบย่อย 2 ระบบสำหรับงาน ATAPE จำนวน 2 งาน (ของงานขนาด $N/2$) (ค) โครงสร้างของ x x x ครอสส์บาร์ของ PMINs$^+$ โดยที่ $N=2^n$ $x=2^b$ และระบบย่อย MIN$^+$ ($N' = N/x$, $n' = \log_2 N/x$)

ตารางที่ 4.2 แสดงความซับซ้อนด้านเวลาและจำนวนของงานบนเครือข่ายแบบ PMINs$^+$ (ขนาด $N=2^n$)

ขนาดงาน (Task(s)) ($N"=N/2^t$, $1 \le 2^t \le x$)	x x x ครอสส์ของ PMINs$^+$		x x x ครอสส์ของ MINs$^+$ [9]	
	ความซับซ้อนด้านเวลา	จำนวนงาน	ความซับซ้อนด้านเวลา	จำนวนงาน
one T (ขนาด N)	$O(N/x + n')$	1	$O(N+ n')$	1
p Ts (ขนาด $N/2$)	$O(N/x + n')$	$2^{2t} = 4$	$O(N/2 + n')$	1
pTs (ขนาด $N/4$)	$O(N/x + n')$	$2^{2t} = 16$	$O(N/4 + n')$	1
...	...	...	...	1
p Ts (ขนาด N/x)	$O(N/x + n')$	$2^{2t} = x^2$	$O(N/x + n')$	1

C^v_l	I/O Cross-Control Switches	Self-Routing Tag on MINs+ (N'=8)		
S^v_l	s_4s_3	s_2	s_1	s_0
D^v_l	d_4d_3	d_2	d_1	d_0
Forward routing i.e. Butterfly+ network		Stg 0	Stg 1	Stg 2
Backward routing i.e. Banyan+ network		Stg 2	Stg 1	Stg 0





รูปที่ 4.5 กลุ่มย่อย 4 กลุ่มบน 4x4 ครอสส์บาร์ของ PMINs+ โดยที่ N = 32

รูปที่ 4.5 แสดงตัวอย่างการกำหนดค่าของ 4 กลุ่มย่อย (แต่ละกลุ่มย่อยของขนาด N) บน 4x4 ครอสส์บาร์ของเครือข่ายแบบ PMINs+ เพื่อที่จะสามารถประมวลผลแต่ละงานได้อย่างอิสระในแต่ละกลุ่มย่อย ในการใช้เทคนิคซุปเปอร์ไปป์ไลน์ (ดีกรี = 4) ขั้นตอนแรกใช้ค่าของค่าควบคุมหลัก 4 ค่า (C=0, $N/4$, $2N/4$, $3N/4$) ของงานแบบ ATAPE (ขนาด N) สามารถทำการประมวลผลบน 4 กลุ่มย่อยที่แตกต่าง (Section 0, 1, 2, 3) พร้อมๆ กันโดยสามารถใช้ระบบได้ 100% ดังนั้นระบบย่อย MINs+ จำนวน x^2 ระบบจะใช้เพื่อเชื่อมต่อระหว่าง N หน่วยประมวลผลต่อสัญญาณนาฬิกา สำหรับกลุ่มย่อยแรก I/O ครอสส์สวิทช์ของกลุ่มย่อยที่ 0 (00-00, 01-01, 10-10, 11-11) ที่ C = 0 เพื่อจะเริ่มให้ทุกๆ หน่วยประมวลผล S และ D ทำการประมวลผลได้พร้อมกันหลังจากนั้น ¼ สัญญาณนาฬิกา จะเริ่มกลุ่มถัดไปที่ C = $N/4$ คือการประมวลผลในกลุ่มย่อยที่ 1 (สำหรับ I/O ครอสส์สวิทช์ที่ 00-01, 01-10, 10-11, 11-00) ในทำนองเดียวกันหลังจาก ¼ สัญญาณนาฬิกาถัดไปจะเริ่มการประมวลผลในกลุ่มย่อยที่ 2 คือ (00-10, 01-11, 10-00, 11-01) ที่ C = $2N/4$ และ ¼-สัญญาณนาฬิกาหลังจากนั้นคือการประมวลผลในกลุ่มย่อยที่ 3 (00-11, 01-00, 10-01, 11-10) โดยที่ C = $3N/4$ ซึ่งเป็นการกำหนดค่าวนตามเข็มนาฬิกาต่อจากนั้นจะทำการประมวลผลในทำนองเดียวกันโดยการทำซ้ำเหมือนเดิมในรอบถัดไป แต่ทำการเพิ่มค่า C ของแต่ละกลุ่มบวกเพิ่มหนึ่ง (C=1, $N/4$+1, $2N/4$+1, $3N/4$+1) จนกระทั่งถึงขั้นตอนที่ N/x (C=$N/4$-1, $2N/4$-1, $3N/4$-1, N-1) เป็นขั้นตอนสุดท้าย

<u>ทฤษฎีบทที่ 4.1</u> I/O สวิทช์ควบคุมแบบครอสส์ (I/O Cross-Control Switches) ของระบบย่อยร่วมกับเทคนิคซุปเปอร์ไปป์ไลน์สำหรับ CPI = 1/*x* สามารถแบ่งเครือข่ายแบบ PMINs^{+} เป็น *x* กลุ่มย่อย (Sections) สำหรับงานแบบ ATAPE ขนาด *N* ซึ่งแต่ละกลุ่มย่อย *y* (= 0, 1, 2, 3, ..., *x*-1) จะจัดสรรให้ใช้กลุ่มย่อยโดยวนตามเข็มนาฬิกาซึ่งใช้ค่าควบคุมหลักของการก้าวกระโดด (Jumping Control) C^y ของ ATAPE เพื่อกำหนดพอร์ตเข้า/ออกของ *S* และ *D* ภายใน *l* = *N*/*x* ขั้นตอน (*x* รอบต่อขั้นตอน) ดังนี้

$$C_l^y = \left(\frac{yN}{x} + l\right) mod\ N \qquad (1)$$

โดยที่ O-Switch = (I-Switch + *y*) mod *x*

(สำหรับการกำหนดค่าควบคุมหลักของสวิทช์ตามเข็มนาฬิกา)

โดยที่ I-Switch = 0, 1,..., *x*-1 และ $0 \le y < x$

และ *l* = 0, 1, 2,..., *N*/*x*-1 (สำหรับ 1 งาน), ..., *N*-1 (สำหรับ *x* งาน)

พิสูจน์ กำหนดให้ *x* x *x* ครอสส์บาร์ของ PMINs^{+} ($x = 2^b$ และ $x^2 < N$) ได้แบ่งออกเป็น *x* กลุ่มย่อย โดยให้วนกำหนดค่าเครือข่ายตามเข็มนาฬิกา เรียกว่ากลุ่มย่อย *y* โดยที่ *y* = 0, 1, ... , *x*-1 โดยที่แต่ละสวิทช์ของค่าควบคุมหลักแบบไขว้สำหรับสลับข้อมูลเข้า/ออกจะใช้ *b* บิตที่มีนัยสำคัญสูงของ *S* (i_{n-1} .. i_{n-b}) เพื่อตั้งค่าให้กับสวิทช์นำข้อมูลเข้า (I-Switch) และ *b* บิตที่มีนัยสำคัญสูงของ $D = S \oplus C_l^y$ สำหรับตั้งค่าให้กับสวิทช์นำข้อมูลออก (O-Switch) หรือ O = (I + *y*) mod *x* ดังนั้นในทุกๆ การเชื่อมต่อระหว่าง (*S*, *D*) ที่แตกต่างของฟังก์ชัน ATAPE สามารถคำนวณโดยใช้ตัวดำเนินการแบบ XOR ได้ตามสมการที่ (1) เพื่อที่จะทำการจัดสรรงานสำหรับ *x* กลุ่มย่อยได้อย่างอิสระ โดยใช้ค่าควบคุมหลักแบบก้าวกระโดด C_l^y ในการจัดสรรงานที่ดีที่สุด (ที่จัดสรรได้ครบทั้ง 3 ปัจจัย) คือสามารถจัดลำดับงานย่อยให้เหมาะกับกลุ่มย่อยในเวลาที่เหมาะสม ดังต่อไปนี้ อับดับแรก ความสามารถในการจัดลำดับงานย่อยให้เหมาะกับกลุ่มย่อยด้วยค่า $C^y = yN/x$ เพื่อจัดสรรในกลุ่มย่อย *y* = 0, 1, 2, ... , *x*-1 สำหรับค่าควบคุมหลัก *x* กลุ่มของงาน อันดับที่สอง การจัดลำดับของงานแบบ ATAPE ในเวลาที่เหมาะสม สำหรับชุดของค่าควบคุมหลักอื่นๆ คือ $C_l^y = yN/x + l$ ในการประมวลผลร่วมกับตัวนับค่าบนชิป *l* = 0, 1, 2, 3, ... , *N*/*x* ดังนั้น

กลุ่ม 0 $\longrightarrow C_l^0$: {0, 1, 2, 3, ..., *N*/*x*-1}

โดยการกำหนด I/O ครอสส์สวิทช์ที่พอร์ต 0-0, 1-1, 2-2, ..., (*x*-1)-(*x*-1)

กลุ่ม 1 $\longrightarrow C_l^1$: {*N*/*x*, *N*/*x*+1, *N*/*x*+2, ..., 2*N*/*x*-1}

โดยการกำหนด I/O ครอสส์สวิทช์ที่พอร์ต 0-1, 1-2, 2-3, ..., (*x*-1)-0

กลุ่ม 2 $\longrightarrow C_l^2$: {2*N*/*x*, 2*N*/*x*+1, 2*N*/*x*+2, ..., 3*N*/*x*-1}

โดยการกำหนด I/O ครอสส์สวิทช์ที่พอร์ต 0-2, 1-3, 2-4, ..., (*x*-1)-1

...

กลุ่ม *x*-1 $\longrightarrow C_l^{x-1}$: {(*x*-1)*N*/*x*, (*x*-1)*N*/*x*+1, (*x*-1)*N*/*x*+2, ..., *N*-1}

โดยการกำหนด I/O ครอสส์สวิทช์ที่พอร์ต 0-(*x*-1), 1-0, 2-1, ..., (*x*-1)-(*x*-2)

ในการที่จะใช้เทคนิคซุปเปอร์ไปป์ไลน์ได้อย่างมีประสิทธิภาพด้วยค่าควบคุมหลัก C_l^y ที่เหมาะสมเพื่อทำให้ *x* กลุ่มย่อยได้ทำงานต่อเนื่องกัน (*y* = 0, 1, ... , *x*-1) โดยเริ่มแต่ละค่า *y* ทุกๆ 1/*x* สัญญาณนาฬิกา (โดยที่ CPI = 1/*x*) และในกลุ่มย่อย *y* เดียวกันสำหรับการเชื่อมต่อ (ในรอบถัดไป *l*) ก็ยังคงทำงานในทุกๆ สัญญาณนาฬิกา ดังนั้น

รอบแรกประมวลผลใน x กลุ่มย่อย คือกลุ่มย่อยที่ 0 จนถึง x-1 ที่ l = 0 (โดยกำหนดให้ค่า C_l^y สำหรับแต่ละกลุ่มย่อย ดังนี้ $C_l^0 = 0$, $C_l^1 = N/x$, $C_l^2 = 2N/x$, ... , $C_l^{x-1} = (x\text{-}1)N/x$) ซึ่งจะสามารถทำงานพร้อมๆ กันได้ในทุกๆ กลุ่มย่อยในหนึ่งสัญญาณนาฬิกา (หรือ $1/x$ สัญญาณนาฬิกาต่อ C) เช่น

ในกลุ่มย่อยที่ 0 I/O ครอสส์สวิทช์คือ 0–0 (หรือ $0..00_2 - 0..00_2$), 1–1 (หรือ $0..01_2 - 0..01_2$), ... , (x-1) – (x-1) (หรือ $1..11_2 - 1..11_2$) ตาม b บิตที่มีนัยสัญญัคูสูงของ N ต้นทาง S (= 0, 1, 2, ... , N-1) และ N ปลายทาง $D = S \oplus C_l^y$

ในกลุ่มย่อยที่ 1 I/O ครอสส์สวิทช์คือ 0–1 (หรือ $0..00_2 - 0..01_2$), 1–2 (หรือ $0..01_2 - 0..10_2$), ... , (x-1) – 0 (หรือ $1..11_2 - 0..00_2$).

ในกลุ่มย่อยที่ 2 I/O ครอสส์สวิทช์คือ 0–2 (หรือ $0..00_2 - 0..10_2$), 1–3 (หรือ $0..01_2 - 0..11_{12}$), ... , (x-1) – 1 (หรือ $1..11_2 - 0..01_2$), ... , และ

สุดท้ายในกลุ่มย่อยที่ x-1 I/O ครอสส์สวิทช์คือ 0–(x-1) (หรือ $0..00_2 - 1..11_2$), 1–0 (หรือ $0..01_2 - 0..00_2$), ... , (x-1) – (x-2) (หรือ $1..11_2 - 1..10_2$)

ประมวลผลใน x กลุ่มย่อยในรอบถัดไปที่ l = 1 ($C_l^0 = 1$, $C_l^1 = N/x+1$, $C_l^2 = 2N/x+1$, ... , $C_l^{x-1} = (x\text{-}1)N/x+1$) เพื่อทำการประมวลผลในหนึ่งสัญญาณนาฬิกาก็ทำได้ในทำนองเดียวกัน

และการประมวลผลดังกล่าวจะทำซ้ำใน x กลุ่มย่อยอีกจนกระทั่งครบจำนวนรอบ x รอบที่ $l = x$ - 1 ($C_l^0 = N/x\text{-}1$, $C_l^1 = 2N/x\text{-}1$, $C_l^2 = 3N/x\text{-}1$, ... , $C_l^{x-1} = N\text{-}1$) ดังนั้นข้อมูลแบบเพอร์ซันนัลไลซ์ N ข้อมูลในทุกๆ กลุ่มย่อยจะถูกส่งไปยังทุกๆ ปลายทางใน N/x รอบ (N รอบต่องานและ x รอบต่อขั้นตอน) โดยใช้ค่าควบคุมหลัก C_l^y โดยที่ $0 \le y < x$ และ l = 0, 1, 2,..., N/x-1 (สำหรับหนึ่งงาน) และ N-1 (สำหรับ x งาน) □

ตัวอย่างเช่น สมมุติว่า 4x4 ครอสส์บาร์ของ PMIN$^+$ (N = 32 x = 4) สำหรับระบบย่อย MIN$^+$ (N'= $N/4$ = 8 และ n' = $\log_2 N'$ = 3 สเตจ) สามารถแบ่งออกเป็น 4 กลุ่มย่อย ดังรูปที่ 4.5 โดยที่ค่าควบคุมหลัก $N/4$ ค่าที่เหมาะสม (C^y) ของแต่ละกลุ่มย่อยจะสามารถคำนวณงานแบบ ATAPE ขนาด N ได้ดังนี้

กลุ่ม 0 (y = 0 เส้นสีฟ้า) สำหรับ C_l^0 : {0, 1, 2, 3, 4, ..., 7}
กลุ่ม 1 (y = 1 เส้นสีแดง) สำหรับ C_l^1 : {8, 9, 10, 11, ..., 15}
กลุ่ม 2 (y = 2 เส้นสีส้ม) สำหรับ C_l^2 : {16, 17, 18, 19,..., 23}
กลุ่ม 3 (y = 3 เส้นสีน้ำเงิน) สำหรับ C_l^3 : {24, 25, 26, 27, ..., 31}

ดังนั้นในกลุ่มย่อยที่ 2 บิตที่มีนัยสำคัญสูง 2 บิตของ S และ D ($S^{y=2}$: s_4s_3 และ $D^{y=2}$: d_4d_3) จะใช้สำหรับกำหนดค่าให้กับ I/O ครอสส์สวิทช์ที่พอร์ต 00_2 - 10_2, 01_2 - 11_2, 10_2 - 00_2 และ 11_2 - 01_2 ในการจัดเส้นทาง (S, D) บนแต่ละระบบย่อยของ MIN$^+$ จะใช้ 3 บิตที่เหลือ ($s_2s_1s_0$ และ $d_2d_1d_0$) ในการกำหนดสวิทช์บนสเตจภายในที่ 0 ถึง 2 (บนเครือข่าย MIN$^+$ แบบไปข้างหน้า (Forward MIN$^+$ Network)) หรือการกำหนดสวิทช์บนสเตจภายในที่ 2 กลับไปยัง 0 (บนเครือข่าย MIN$^+$ แบบย้อนกลับ (Backward MIN$^+$ Network)) ในกรณีที่รับข้อมูลในทุกๆ ปลายทาง ($D^{y=2}$) จากทุกๆ ต้นทาง ($S^{y=2}$) โดยผ่านระบบย่อยใน N/x รอบ (l = 0, 1, ... , N/x-1)

หมายเหตุ การส่ง ATAPE (N ค่า) ได้สำเร็จบนเครือข่ายแบบ PMINs$^+$ นี้ กลุ่มย่อยที่ 0 ถึง 3 จะต้องดำเนินการพร้อมๆ กันแบบสลับโดยใช้ซุปเปอร์ไปป์ไลน์ (ดีกรี = x) สำหรับ CPI = $1/x$

ในทำนองเดียวกันสำหรับการประมวลผลงานแบบ ATAPE ขนาด N สามารถทำได้พร้อมๆ กันมากถึง x = 4 งานบนกลุ่มย่อยที่ 0 ถึง 3 โดยการระบุรอบของ C_i^y เป็นจำนวน N ค่า สำหรับ l = 0, 1, 2, ..., N-1 โดยที่ทุกงานสามารถประมวลผลได้พร้อมๆ กันใน x กลุ่มย่อยดังนี้

กลุ่ม 0 (เส้นสีฟ้า) สำหรับ C_i^0 (ของงาน T_0): {0, 1, 2, 3, 4, ..., 31}

กลุ่ม 1 (เส้นสีแดง) สำหรับ C_i^1 (ของงาน T_1): {8, 9, 10, .., 31, 0, .., 7}

กลุ่ม 2 (เส้นสีส้ม) สำหรับ C_i^2 (ของงาน T_2): {16, 17, .., 31, 0, ..,15}

กลุ่ม 3 (เส้นสีน้ำเงิน) สำหรับ C_i^3 (ของงาน T_3): {24, 25, .. , 31, 0, .. ,23}

ยกตัวอย่างการประมวลผลพร้อมๆ กันของ C_i^0 = 0 (ของ T_0) C_i^1 = 8 (ของ T_1) C_i^2 = 16 (ของ T_2) และ C_i^3 = 24 (ของ T_3) บนกลุ่มย่อย 0 ถึง 3

สำหรับ C_i^0=0 (T_0) I/O ครอสส์สวิทช์ของกลุ่มย่อยที่ 0 (00-00, 01-01, 10-10, 11-11) สามารถทำให้ทุกๆ หน่วยประมวลผลต้นทางและปลายทาง (S, D) ของ T_0 และ

C_i^1 = 8 (T_1) บนกลุ่มย่อยที่ 1 (00-01, 01-10, 10-11, 11-00)

C_i^2 = 16 (T_2) บนกลุ่มย่อยที่ 2 (00-10, 01-11, 10-00, 11-01) และ

C_i^3 = 24 (T_3) บนกลุ่มย่อยที่ 3 (00-11, 01-00, 10-01, 11-10) ในหนึ่งสัญญาณนาฬิกา และจะทำการประมวลผลไปเรื่อยๆ สำหรับ 4 ค่าควบคุมหลักต่อไป (C^y = 1 (T_0), N/4+1 (T_1), 2N/4+1 (T_2), 3N/4+1 (T_3)), ... , จนกระทั่ง 4 ค่าควบคุมหลักสุดท้าย (C^y = N-1 (T_0), N/4-1 (T_1), 2N/4-1 (T_2), 3N/4-1 (T_3)) จึงได้รับข้อมูลครบทุกๆ งาน

4.1.3 การจัดสรรงานที่ดีที่สุด (ที่จัดสรรได้ครบทั้ง 3 ปัจจัย) บนเครือข่ายแบบ PMINs$^+$

สำหรับบน x x x ครอสส์บาร์ของ PMINs$^+$ การประมวลผลงานแบบ ATAPE หลายๆ งาน (p) ของขนาด $N/2^t$ (โดยที่ t = 0, 1, 2, ... , $\log_2 x$) บนกลุ่มย่อยที่แตกต่างกันโดยใช้เทคนิคซุปเปอร์ไปป์ไลน์ (ดีกรี = x) เพื่อสามารถจัดลำดับงานย่อยให้เหมาะกับกลุ่มย่อยในเวลาที่เหมาะสม (Right task, Right section, Right time) ได้เต็มประสิทธิภาพ

กำหนดให้ $T_{i,C}$ แทนงานที่ i แบบ ATAPE (ในรอบที่ C) โดยที่แต่ละ T_i ประกอบด้วย N รอบของค่าควบคุมหลัก (C) โดยทั่วไปลำดับปกติของค่าควบคุมหลักคือ C = 0, 1, 2, 3, ... , N-1 ดังนั้น ลำดับของ N งานย่อยของงานแบบ ATAPE ที่ T_i คือ $T_{i,0}$, $T_{i,1}$, $T_{i,2}$, $T_{i,3}$, ..., $T_{i,N-1}$ โดยที่ $T_{i,j}$ จะเริ่มในทุกๆ สัญญาณนาฬิกาเมื่อ $T_{i,j-1}$ ทำงานเสร็จเรียบร้อย และ $T_{i,j}$ จะทำงานเสร็จก่อนที่ $T_{i,j+1}$ จะเริ่มทำงาน ดังเช่นในรูปที่ 4.6 (ก) และสำหรับการประมวลผลงาน (Task) T_i แบบ ATAPE บน 2x2 ครอสส์บาร์ของ PMINs$^+$ ได้อย่างมีประสิทธิภาพใน 2 กลุ่มย่อยดังเช่นในรูปที่ 4.6 (ข) ซึ่งสามารถใช้ได้พร้อมๆ กันใน N/2 ขั้นตอนด้วยซุปเปอร์ไปป์ไลน์ (ดีกรี x = 2) โดยมีลำดับของการประมวลผลงานย่อย (Sub-Tasks $T_{i,j}$ โดยที่ j = 0, 1, 2, 3, ..., N-1) จำนวน N งานย่อยบน 2 กลุ่มย่อยด้วยเทคนิคซุปเปอร์ไปป์ไลน์ ดังนี้ กลุ่มย่อยที่ 0 กำหนดพอร์ต I/O ครอสส์สวิทช์ที่ 0-0 และ 1-1 สำหรับงานย่อย $T_{i,0}$, $T_{i,1}$, $T_{i,2}$, $T_{i,3}$, ..., $T_{i,N/2-1}$ และกลุ่มย่อยที่ 1 กำหนดพอร์ต I/O ครอสส์สวิทช์ที่ 0-1 และ 1-0 สำหรับงานย่อย $T_{i,N/2}$, $T_{i,N/2+1}$, $T_{i,N/2+2}$, ..., $T_{i,N-1}$ ในทำนองเดียวกันสำหรับการประมวลผลของ 2 งานแบบ ATAPE (T_0, T_1) ขนาด N ดังเช่นในรูปที่ 4.6 (ค) ลำดับของการประมวลผล N งานย่อยของแต่ละ T_i บน 2 กลุ่มย่อยด้วยการจัดสรรดังต่อไปนี้ กลุ่มย่อยที่ 0 (0-0 และ 1-1) สำหรับงานย่อย $T_{0,0}$, $T_{0,1}$, $T_{0,2}$, ..., $T_{0,N-1}$ และกลุ่มย่อยที่ 1 (0-1 และ 1-0) สำหรับงานย่อย $T_{1,N/2}$, $T_{1,N/2+1}$,...,$T_{1,N-1}$, $T_{1,0}$, $T_{1,1}$,...,$T_{1,N/2-1}$ ส่วนรูปที่ 4.6 (ง) แสดงลำดับที่ถูกต้องเหมาะสมของงานย่อย (Sub-Task $T_{i,j}$) งานแต่ละงาน T_i (T_0, T_1, T_2, T_3) ขนาด N"= N/2 จำนวน 4 งาน (Tasks) บน 2 กลุ่มย่อย (2 งานต่อกลุ่มย่อย) นอกจากนั้นสำหรับ 4x4 ครอสส์บาร์ของ PMIN$^+$ รูปที่ 4.6 (จ) แสดงลำดับที่เหมาะสมของงาน 1 งาน (ขนาด N) บน 4 กลุ่มย่อยด้วยเทคนิคซุปเปอร์ไปป์ไลน์ (ดีกรี = 4)

รูปที่ 4.7 แสดงแผนภาพเวลาของการจัดสรรงานที่ดีที่สุด (ที่จัดสรรได้ครบทั้ง 3 ปัจจัย) คือ สามารถจัดลำดับงานย่อยให้เหมาะกับกลุ่มย่อยในเวลาที่เหมาะสม ในการจัดสรรการใช้งานกลุ่มย่อยบนกลุ่มย่อย 0 และกลุ่มย่อย 1 ของงาน 2 งาน (ขนาด N) ในรูปที่ 4.7 (ก) และแสดงประสิทธิภาพด้วยแผนภาพแสดงช่วงเวลา ในรูปที่ 4.6 (ค) และ 4.7 (ค) ในทุกๆ ½ สัญญาณนาฬิกาทั้ง 2 กลุ่มย่อยจะสลับกันทำงาน ในการจัดสรรงานด้วยซุปเปอร์ไปป์ไลน์ทั้ง 2 กลุ่มย่อยสามารถประมวลผลใน $N/2$ รอบผ่าน n' (= $\log_2 N/2$) สเตจขนานกันด้วยความซับซ้อนด้านเวลา $O(N/2 + n')$ สำหรับ 1 งาน (ขนาด N) เพราะในรอบแรก C=0 ต้องการ n' สัญญาณนาฬิกาและ N-1 รอบที่เหลือต้องการ (N-1)/2 สัญญาณนาฬิกา

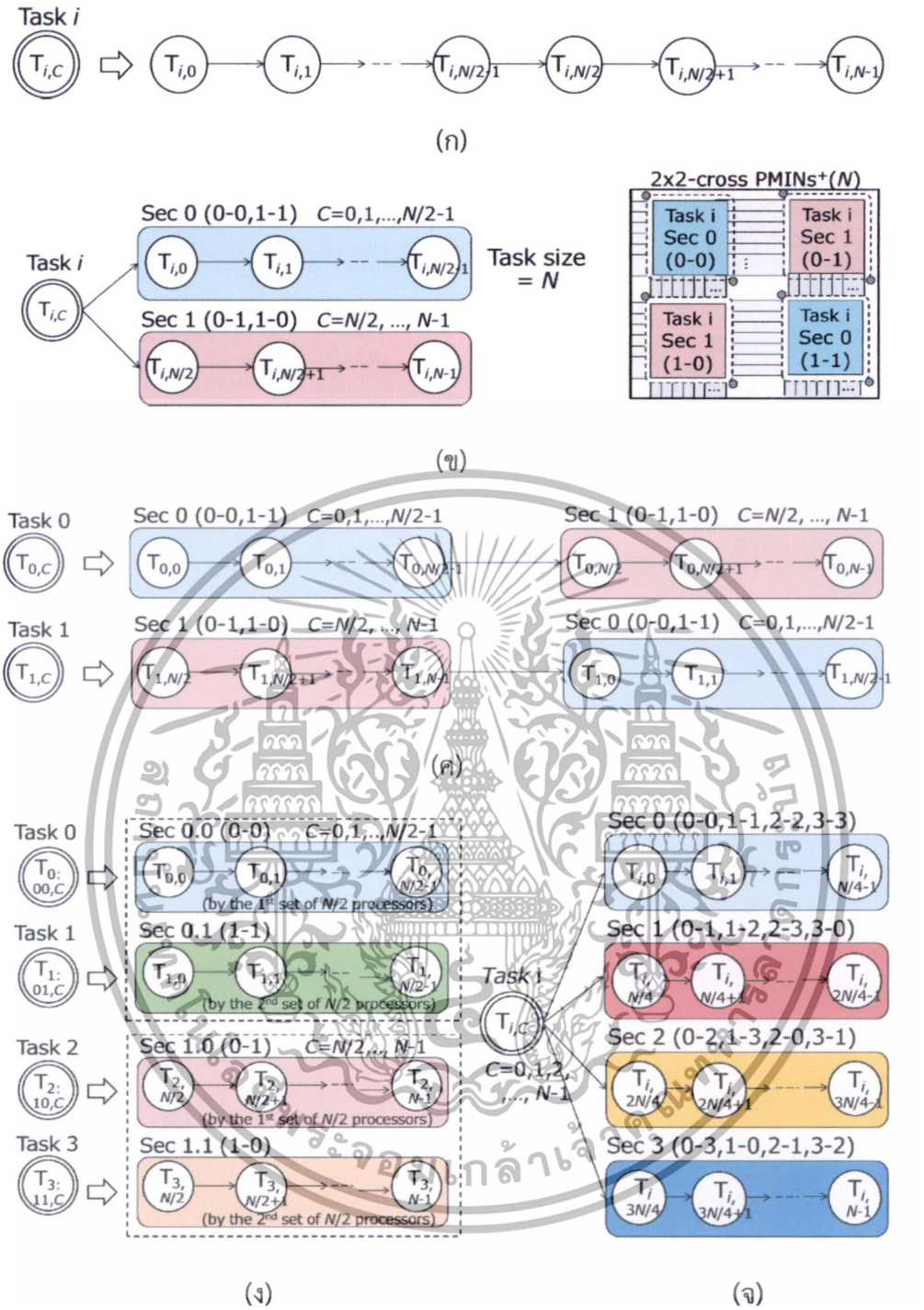
<u>ทฤษฎีบทที่ 4.2</u> บน $x \times x$ ครอสส์บาร์ของ PMINs$^+$ จำนวนงาน (Tasks) แบบ ATAPE (ขนาด $N" = N/2^t$) ที่สามารถจัดสรรให้ประมวลผลได้พร้อมๆ กันมากที่สุด คือ $p = 2^{2t}$ งาน ด้วยความซับซ้อนด้านเวลา $O(N/x+n')$ โดยที่ $1 \le 2^t \le x$

พิสูจน์ สำหรับการจัดสรรหลายๆ งาน (T_0, T_1, T_2, ..., T_{p-1}) บนเครือข่ายแบบ PMINs$^+$ I/O สวิทช์ควบคุมแบบครอสส์จะเปิดให้ใช้สำหรับกลุ่มย่อยที่ y (= 0, 1, 2, ..., x-1) แบบต่อเนื่องในทุกๆ $1/x$ สัญญาณนาฬิกา โดยมี $p = 2^{2t}$ งาน (ขนาด $N" = N, N/2, ..., N/2^t, ..., N/x$) บน x กลุ่มย่อย (2^t งานต่อกลุ่มย่อย) เพื่อสามารถใช้ระบบได้เต็มประสิทธิภาพ ภายใน $N/x+n'$ สัญญาณนาฬิกา (ในตารางที่ 4.2) เนื่องจากขนาดของตารางลาตินบนเครือข่ายแบบ PMINs$^+$ คือ $N \times N$ และงานขนาด $N"=N/2^t$ ต้องการตารางลาตินย่อยขนาด $N" \times N"$ ดังนั้นระบบนี้จะสามารถรองรับการจัดสรรงานได้ประมาณมากที่สุดจำนวนเท่ากับ $NxN / (N" \times N") = 2^t \times 2^t$ งาน เพื่อประมวลผลงานขนาด N (กรณีที่ 1) และงานขนาด $N"$ (กรณีที่ 2)

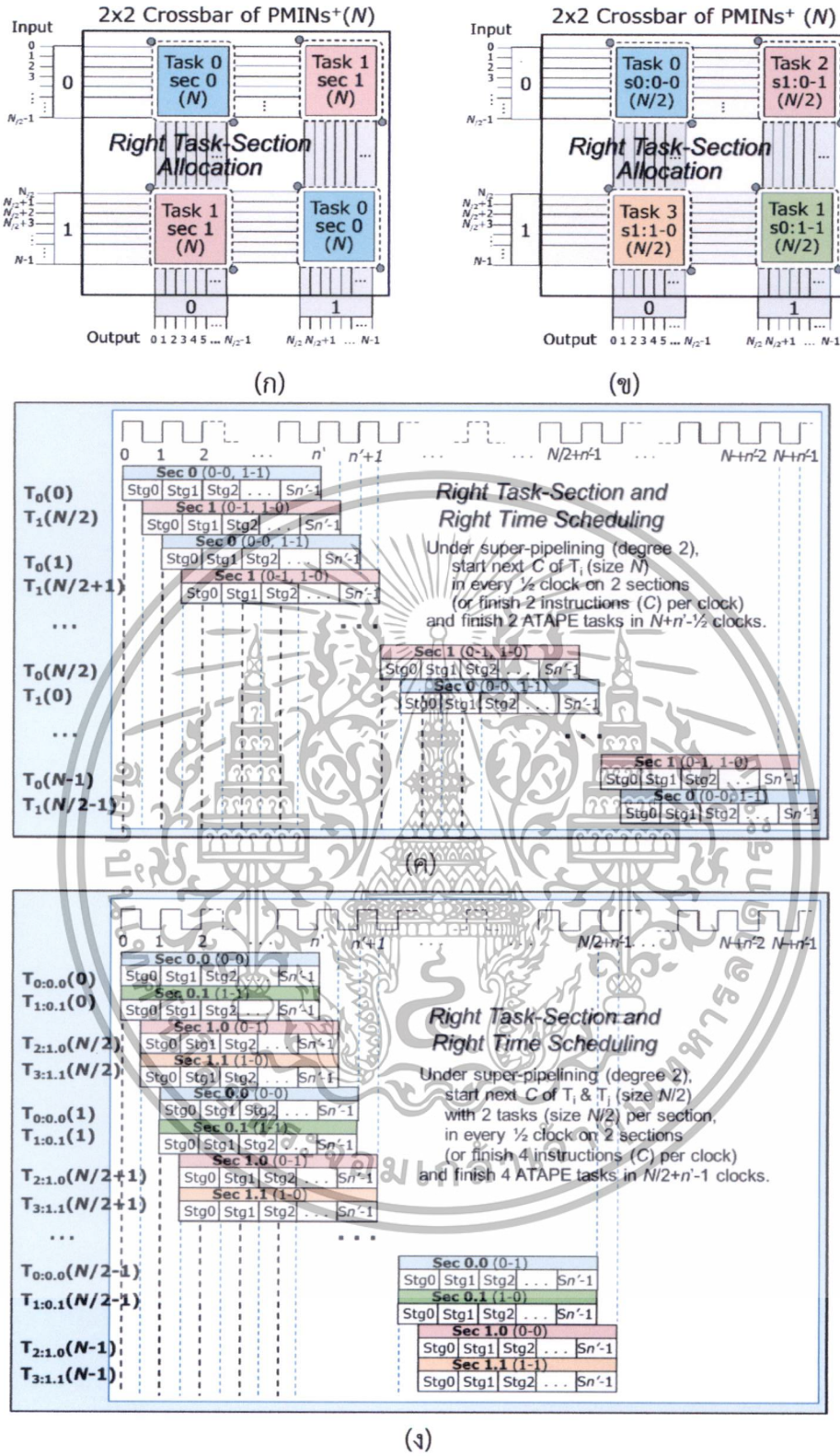
<u>กรณีที่ 1</u> สำหรับการประมวลผลงานแบบ ATAPE (ของขนาด N) ใน N/x รอบของงานที่ถูกกำหนดให้ทำการประมวลผลในกลุ่มย่อยที่ y (ของทุกๆ x กลุ่มย่อย) ด้วยซุปเปอร์ไปป์ไลน์ (ดีกรี = x) โดยจะสามารถเริ่มแต่ละกลุ่มย่อยในทุกๆ $1/x$ สัญญาณนาฬิกา ดังแสดงในตัวอย่างรูปที่ 4.6-4.7 (จ) เป็นการจัดสรรงานย่อย (Sub-Task T_{ij}) ของงาน T_i ที่ดีที่สุด (ที่จัดสรรได้ครบทั้ง 3 ปัจจัย) ของ N ค่าควบคุมหลัก (C) บน 4 กลุ่มย่อยต่อเนื่องกันบน 4x4 ครอสส์บาร์ของ PMINs$^+$

<u>กรณีที่ 2</u> สำหรับงานขนาด $N"=N/2^t$ โดยที่ $2 \le 2^t \le x$ จำนวนงานที่มากที่สุด $p = 2^{2t}$ งานโดยที่ 2^t งานสามารถจัดสรรให้อยู่ในแต่ละกลุ่มย่อย y และดังนั้นทุกๆ $p=2^{2t}$ งานในทุกๆ x กลุ่มย่อยจะถูกกำหนดให้ใช้งานได้พร้อมๆ กัน (รูปที่ 4.6-4.7 (ง) สำหรับ x=2 และ $N"=N/2$) □

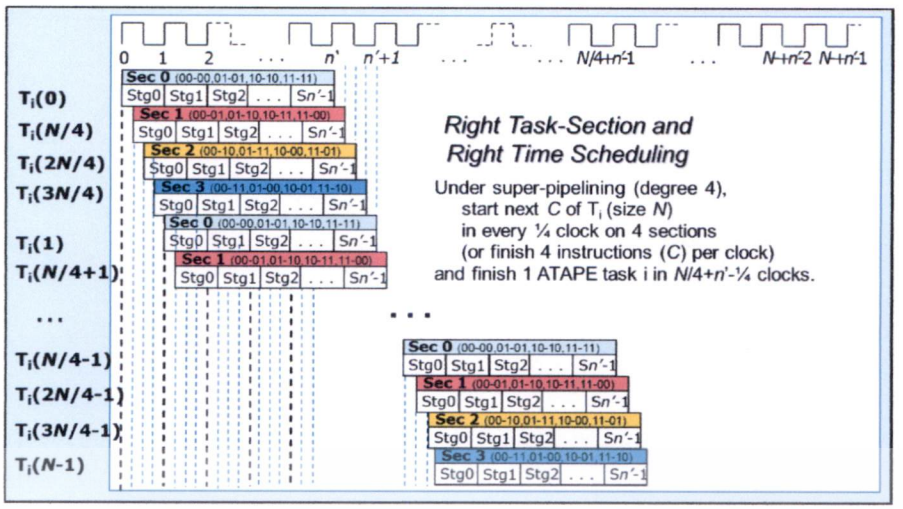
ในรูปที่ 4.6-4.7 (ง) งานแบบ ATAPE จำนวน 4 งาน (ของขนาด $N/2$) บน 2x2-PMIN$^+$ (รูปที่ 4.7 (ข)) งานที่ T_0 และ T_1 ถูกจัดสรรในกลุ่มย่อยที่ 0 (ภายในกลุ่มย่อยที่ 0-0 และ 1-1) งานที่ T_2 และ T_3 ถูกจัดสรรในกลุ่มย่อยที่ 1 (กลุ่มย่อยภายในที่ 0-1 และ 1-0) ด้วยการจัดสรรงานแบบซุปเปอร์ไปป์ไลน์ (เช่น ทุกๆ ½ สัญญาณนาฬิกาจะเริ่มกลุ่มย่อยถัดไป และในแต่ละกลุ่มย่อย ทุกๆ สัญญาณนาฬิกาสามารถประมวลผลในค่าควบคุมหลัก C รอบถัดไปผ่าน n' สเตจ) ส่วนในรูปที่ 4.6-4.7 (จ) แสดงตัวอย่างอื่นๆ ของการประมวลผลงานแบบ ATAPE (ของขนาด N) บน 4x4-PMIN$^+$ (ที่แสดงรายละเอียดของเครือข่ายในรูปที่ 4.5) ที่ใช้เวลาประมวลผลใน $N + n'$- ¼ สัญญาณนาฬิกาเนื่องจากในรอบแรกของ C ต้องการเวลา n' สัญญาณนาฬิกา และในรอบอื่นๆ ของ C (N-1) รอบประมวลผลด้วย ¼ สัญญาณนาฬิกาต่อ C) ต้องการเวลา (N-1)/4 สัญญาณนาฬิกา



รูปที่ 4.6 ตัวอย่างลำดับของงาน (ก) ลำดับของงานปกติ (ข) ลำดับของงานที่เหมาะสมบน 2x2 ครอสส์บาร์ของ PMINs$^+$ (ค) การจัดสรรงานของงานแบบ ATAPE จำนวน 2 งาน (ของขนาด N) (ง) งานแบบ ATAPE จำนวน 4 งาน (ของขนาด $N/2$) และ (จ) งานแบบ ATAPE จำนวน 1 งาน (ของขนาด N) บน 4x4-PMIN$^+$



รูปที่ 4.7 การจัดสรรงานแบบ ATAPE บน 2x2 ครอสส์บาร์ของ PMINs+ (ก) งานแบบ ATAPE จำนวน 2 งาน (ขนาด *N*) (ข) งานแบบ ATAPE จำนวน 4 งาน (ขนาด *N*/2) (ค) ไดอะแกรมเวลาของงานแบบ ATAPE จำนวน 2 งาน (ขนาด *N*) (ง) งานแบบ ATAPE จำนวน 4 งาน (ขนาด *N*/2) (จ) งานแบบ ATAPE จำนวน 1 งาน (ขนาด *N*) บน 4x4-PMIN+



(จ)

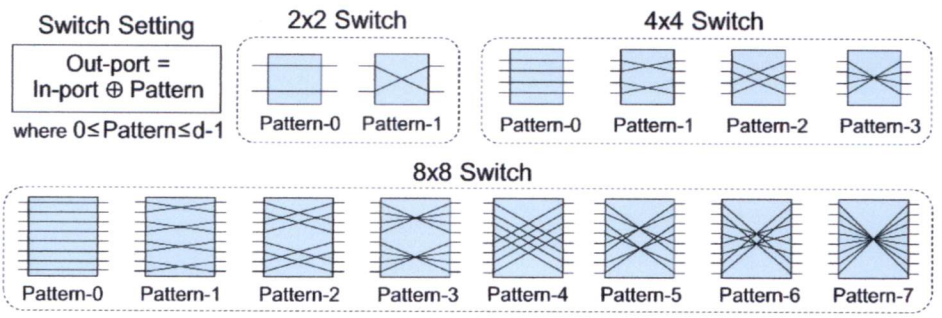
รูปที่ 4.7 (ต่อ)

4.2 ขั้นตอนวิธีแบบ "วดี" (VA-DE : Valuable ATAPE with Dynamic Embedding)

สำหรับการจัดสรรงานที่ดีที่สุด (ที่จัดสรรได้ครบทั้ง 3 ปัจจัย) สามารถใช้ตัวดำเนินการแบบ XOR เพื่อกำหนดแบบของสวิทช์จำนวน d แบบเสนอในหัวข้อที่ 4.2.1 ในหัวข้อที่ 4.2.2 เสนอตารางลาตินที่แบ่งได้ซึ่งมีความยืดหยุ่นและมีประสิทธิภาพ สำหรับการประมวลผลงานหลายๆ งานบน $x \times x$ ครอสส์บาร์ของ PMINs$^+$ และในหัวข้อที่ 4.2.3 เสนอขั้นตอนวิธีแบบ "วดี" (VA-DE : Valuable ATAPE with Dynamic Embedding) ที่เหมาะสมที่สุด

4.2.1 แบบของสวิทช์ของตัวดำเนินการแบบ XOR บนเครือข่ายแบบ PMINs$^+$

ประโยชน์ของตัวดำเนินการแบบ XOR บนเครือข่ายแบบ PMINs$^+$ คือความสามารถในการจัดเส้นทางด้วยแบบของสวิตช์บนสเตจ (Switch-Stage Pattern Control) ได้เหมือนกับการจัดเส้นทางจากต้นทางไปยังปลายทางโดยทั่วไป และการจัดสรรงานที่ดีที่สุดเพื่อที่จะได้จัดลำดับของงานได้เหมาะสม (ของค่าควบคุมหลัก C) บนทุกๆ กลุ่มเพื่อการใช้งานระบบได้ทั้งระบบ 100% รวมทั้งลักษณะพิเศษของตัวดำเนินการแบบ XOR คือแบบของสวิทช์ภายในที่สามารถกำหนดโดยการแบ่ง k บิตของบิตที่มีทั้งหมด n บิต $D = S \oplus C$ ($d_{n-1}=s_{n-1}\oplus c_{n-1}$, ..., $d_2=s_2\oplus c_2$, $d_1=s_1\oplus c_1$, $d_0=s_0\oplus c_0$) โดยที่สามารถดำเนินการได้ภายใน 1 สัญญาณนาฬิกาและใช้การแบ่ง k บิต สำหรับในแต่ละสเตจ เช่น การแบ่งของ $(d_{k-1} \ldots d_1 d_0) = (j_{k-1} \ldots j_1 j_0)$ โดยที่ไม่ต้องทำการ XOR ภายใน $J = I \oplus P$ ในแต่ละสวิทช์ รูปที่ 4.8 แสดงการกำหนดค่าของแบบสวิทช์ขนาด $d \times d$ จำนวน d แบบ



รูปที่ 4.8 แบบของสวิทช์โดยการใช้ตัวดำเนินการแบบ XOR (d = 2, 4 และ 8)

4.2.2 การแบ่งตารางลาตินเพื่อประมวลผลงานแบบ ATAPE หลายๆ งาน

การกำหนดตำแหน่งที่อยู่แบบเฉพาะกลุ่มย่อยและทั้งระบบ (Local and Global Addressing) สำหรับการติดต่อสื่อสารระหว่างหน่วยประมวลผล (PEs) บน $x \times x$ ครอสส์บาร์ของ PMINs$^+$ โดยใช้บิตทั้งหมดจำนวน n บิตเพื่อทำการเชื่อมต่อระหว่าง (S, D) ของงานขนาด $N" = N/2^t$ ในแต่ละระบบย่อย (MINs$^+$ โดยที่ $N' = N/x = 2^{n'}$) บิตที่มีนัยสำคัญต่ำจำนวน n' บิตจะใช้สำหรับจัดเส้นทางภายในแบบสมบูรณ์ผ่าน n' สเตจบนระบบย่อย MINs$^+$ ดังเช่นรูปที่ 4.9 ในการกำหนดตำแหน่งที่อยู่แบบเฉพาะกลุ่มย่อยและทั้งระบบ กำหนดให้ S^y และ D^y แทนการเชื่อมต่อระหว่าง (S, D) ในกลุ่มย่อยที่ y บน $x \times x$ ครอสส์บาร์ของ PMINs$^+$ (ดังเช่นทฤษฎีบทที่ 4.1) ในกรณีของการกำหนดตำแหน่งที่อยู่ของบิตทั้งหมดจำนวน n บิตของต้นทาง S^y: $s_{n-1}s_{n-2} \ldots s_2s_1s_0$ และปลายทาง D^y: $d_{n-1}d_{n-2} \ldots d_2d_1d_0$ จะถูกแบ่งตำแหน่งที่อยู่ออกเป็น 2 ส่วนสำหรับค่าควบคุมหลักของ I/O ครอสส์สวิทช์ และการจัดเส้นทางภายในของ MIN$^+$ ดังต่อไปนี้

1) ค่าควบคุมหลักของ I/O ครอสส์สวิทช์ (โดยที่ O-Switch = (I-Switch + y) mod x)) ใช้ b บิตที่มีนัยสำคัญสูงของ S^y และ D^y สำหรับการติดต่อสื่อสารในกลุ่มย่อย y โดยในการประมวลผลงานแบบ ATAPE (ขนาด N) แต่ละกลุ่มย่อย y (ประกอบด้วย ระบบย่อย MINs$^+$ จำนวน x ระบบ) สามารถประมวลผลแบบขนานด้วยการกำหนด I/O ครอสส์สวิทช์จากค่าของบิตที่มีนัยสำคัญสูง b บิต ของ S^y (สำหรับข้อมูลเข้าใช้แถว (I-Row Enable)) และ D^y (สำหรับข้อมูลออกใช้คอลัมน์ (O-Column Enable)) ยิ่งไปกว่านั้นสำหรับงานแบบ ATAPE หลายงานขนาด $N"=N/2^t$ จะใช้บิตที่มีนัยสำคัญสูง t บิต ($b_{n-1} .. b_{n-t}$) ของ S^y ในการระบุงานย่อยๆ (จำนวน 2^t งานย่อยต่อกลุ่มย่อย (Section)) ขนาด $N/2^t$ (ดังรูปที่ 4.8 (ค-ง))

2) การจัดเส้นทางภายในแบบสมบูรณ์บนแต่ละส่วนของระบบย่อย MIN$^+$ จะใช้บิตที่มีนัยสำคัญต่ำ n' บิตตั้งค่าสวิทช์โดยผ่าน n' สเตจของสวิทช์ขนาด $d \times d$ ($n' = n - b$, $d = 2^k$) โดยใช้ k บิตต่อสวิทช์สำหรับการจัดเส้นทางของ MINs$^+$ แบบไปข้างหน้าและย้อนกลับ

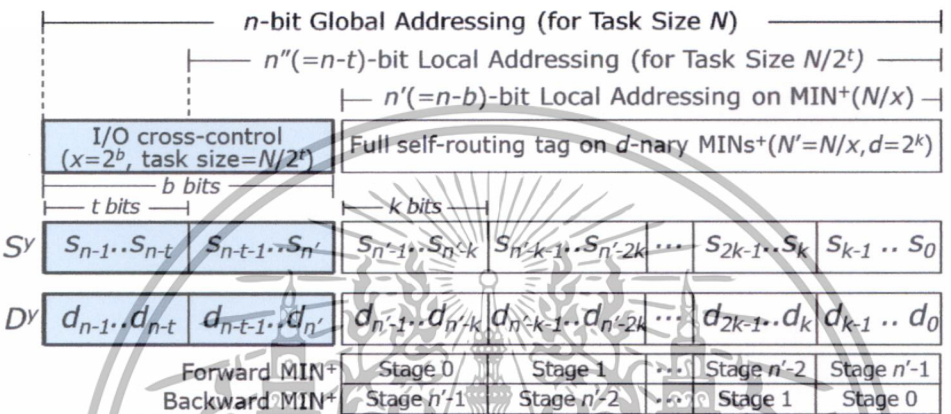
สำหรับการคำนวณตารางลาติน (Latin Square : LS) และการแบ่งตารางลาตินสำหรับประมวลผลในกลุ่มย่อย (ดังรูปที่ 4.10) จะดำเนินการตามสมการที่ (2) เมื่อกำหนดให้การเชื่อมต่อระหว่าง (S^y, D^y) ในกลุ่มย่อย y บนเครือข่ายแบบ PMINs$^+$ ใช้ลำดับของค่าควบคุมหลัก ($C_0^y, C_1^y, C_2^y, \ldots, C_l^y$) เพื่อกำหนดปลายทางที่สอดคล้องกัน (ในการส่งและรับข้อมูลแบบเพอร์ชันนัลไลซ์จำนวน N_{All}^y ข้อมูล) ในกลุ่มย่อย y ได้ดังนี้

$$D^y = S^y \oplus C_l^y \text{ และ } C_l^y = (yN/x + l) \bmod N \qquad (2)$$

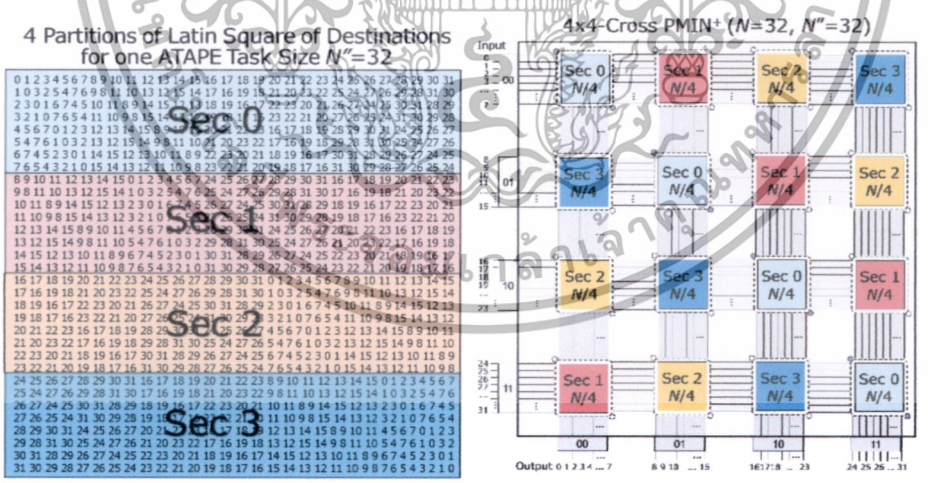
โดยที่ $l = 0, 1, 2, \ldots, < N$

ในแต่ละ x ระบบย่อยของกลุ่มย่อย y สามารถใช้ n' บิตที่มีนัยสำคัญต่ำของ S^y และ D^y ด้วยการคำนวณในสมการที่ 2 ทุกๆ ต้นทางจะมีเส้นทางที่แตกต่างกัน (ระหว่าง 0 ถึง N-1) โดยที่สามารถส่งข้อมูลแบบเพอร์ชันนัลไลซ์ N_{All}^y ข้อมูลไปยังทุกๆ ปลายทางใน N/x ขั้นตอน (ของ N รอบ) สำหรับตัวอย่างบน 4x4 ครอสส์บาร์ของ PMINs$^+$ (N=32) วิธีการแบ่งตารางลาตินแสดงในรูปที่ 4.10 สามารถแบ่งออกเป็น 4 กลุ่มย่อย (y = 0 (00_2), 1 (01_2), 2 (10_2), 3 (11_2)) สำหรับงานหลายๆ งานขนาด $N"= N/2^t$ โดยที่ค่าควบคุมหลัก C_l^y ของกลุ่มย่อย y (l = 0, 1, ..., N/4-1 เช่น y=0 จะใช้ C_l^y = 0-7 เป็นต้น)

ในรูปที่ 4.10 (ก) แสดงการแบ่งตารางลาติน N/x ส่วนสำหรับงานของขนาด N''= 32 (x = 4) โดยที่ทุกๆ n บิตของการกำหนดตำแหน่งที่อยู่แบบทั้งระบบ (Global Addressing) จะใช้ใน x กลุ่มย่อยที่แตกต่าง (y = 0, 1, 2, 3) พร้อมๆ กันสำหรับ l = 0, 1, 2, ... , $N/4$-1 สำหรับรูปที่ 4.10 (ข-ค) แสดงตัวอย่างงานขนาดอื่นๆ ($N''=N/2$ และ $N/4$) ของการกำหนดตำแหน่งที่อยู่ของบิตภายใน n'' บิต (n''-bit Local Addressing) และบิตทั้งหมด n บิต (n-bit Global Addressing) ของงานขนาด $N''= N/2^t$ ซึ่งบรรจุในกลุ่มย่อย y (= 0, 1, 2,... , x-1) ด้วยงานจำนวน 2^t งาน (ต่อกลุ่มย่อย) โดยที่แต่ละงานต้องการ $n'' = n - t$ บิตสำหรับกำหนดตำแหน่งที่อยู่ภายในกลุ่มย่อย $LS^y = S^y \bmod N''$ และ $LD^y = D^y \bmod N''$

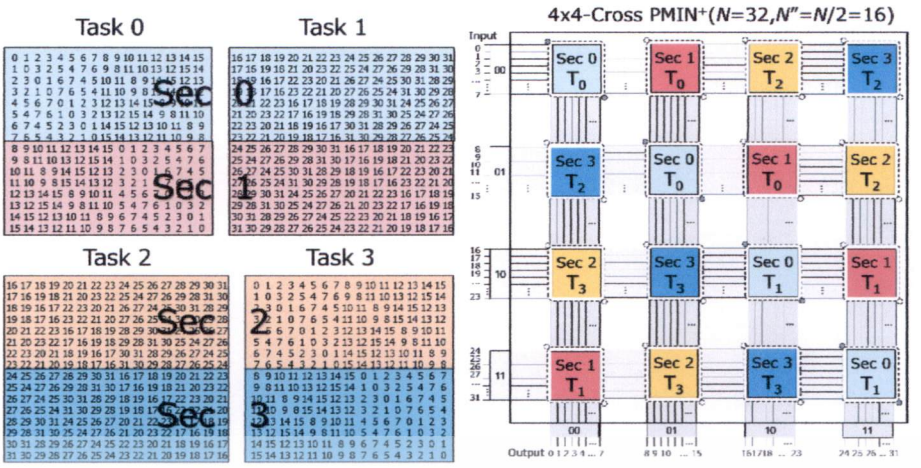


รูปที่ 4.9 การกำหนดตำแหน่งที่อยู่แบบเฉพาะกลุ่มย่อยและทั้งระบบของการติดต่อสื่อสารแบบ ATAPE บน x x x ครอสส์บาร์ของ PMINs$^+$ โดยที่เครือข่ายแบบ MINs$^+$ มีสวิทช์ขนาด dxd ในแต่ละกลุ่มย่อย y

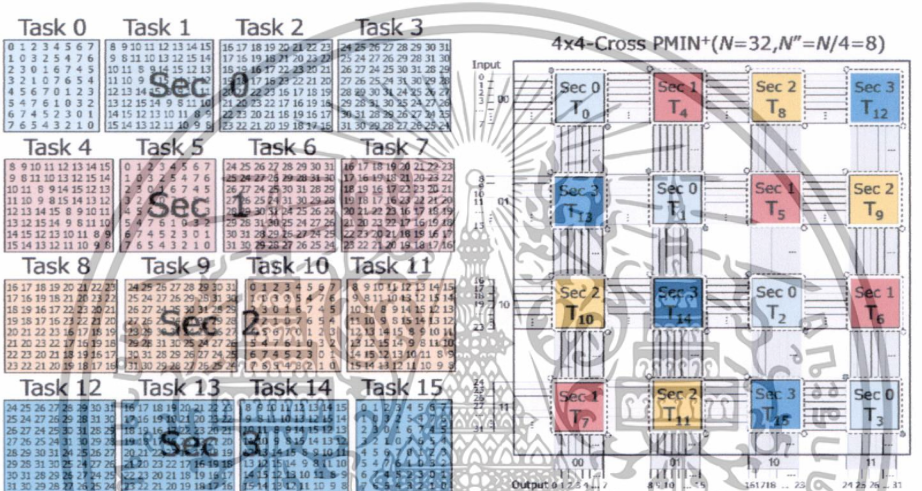


(ก)

รูปที่ 4.10 การแบ่งตารางลาตินสำหรับงานขนาด ($N/2^t$, $2^t \le 4$) บน 4x4 ครอสส์บาร์ของ PMINs$^+$ (ก) งานแบบ ATAPE จำนวน 1 งาน (N=32) (ข) งานแบบ ATAPE จำนวน 4 งาน (N''=16) (ค) งานแบบ ATAPE จำนวน 16 งาน (N''= 8)

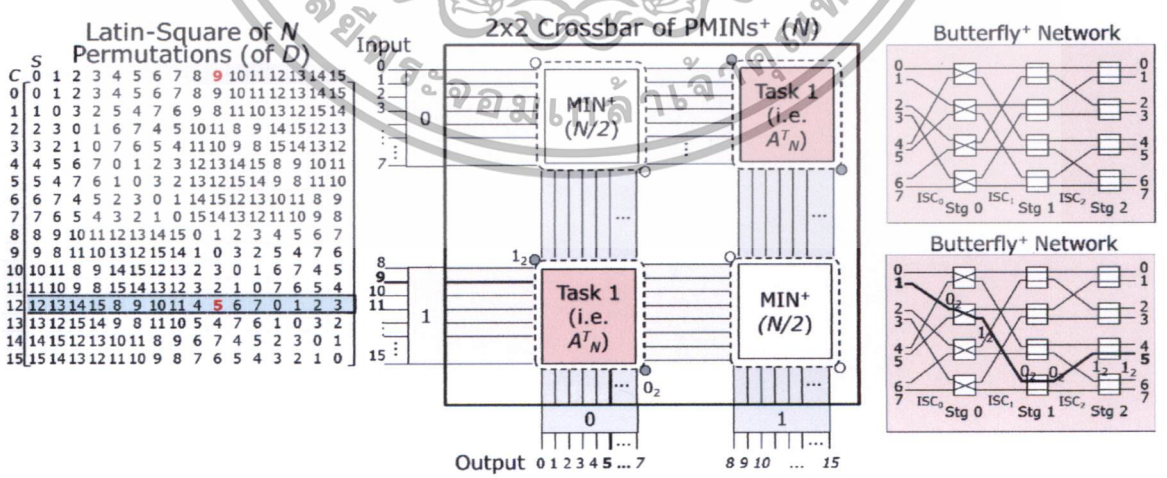


(ข)



(ค)

รูปที่ 4.10 (ต่อ)



รูปที่ 4.11 ตัวอย่างของการจัดเส้นทางภายในจากต้นทางไปยังปลายทาง (*S*, *D*) ของงานที่ 1 (*N"* = 16) บน 2x2 ครอสส์บาร์ของ PMINs+ (*N* = 16, *N'* = 8, *x* = 2)

รูปที่ 4.11 แสดงตัวอย่างอื่นๆ อีกตัวอย่างบน 2x2 ครอสส์บาร์ของ PMINs$^+$ (N=16, N'=8) สำหรับงานแรก (เช่น การทรานสโพสเมตริกซ์ (A^T) ที่ N = 16) จะทำการกำหนด I/O ครอสส์สวิทช์พอร์ต 0-1 และ 1-0 ของกลุ่มย่อยที่ 1 เช่นในกรณีของตำแหน่งที่อยู่ของทั้งระบบของทุกๆ ต้นทางและปลายทาง (S^1, D^1) การเชื่อมต่อในตารางลาติน (N = 16) เช่นกำหนดให้ l = 4 และ C_4^1 = 12 การเชื่อมต่อทั้งระบบจาก N ต้นทางคือ [12, 13, 14, 15, 8, 9, 10, 11, 4, 5, 6, 7, 0, 1, 2, 3] บน 2 ระบบย่อย MINs$^+$ ดังนี้

1-0 (I/O ครอสส์สวิทช์) สำหรับ MIN$^+$ {0⟶12, 1⟶13, 2⟶14, ..., 7⟶11}

0-1 (I/O ครอสส์สวิทช์) สำหรับ MIN$^+$ {8⟶4, 9⟶5, 10⟶6, ..., 15⟶3}

โดยเฉพาะการกำหนดตำแหน่งที่อยู่ภายในกลุ่มย่อยและทั้งระบบในการจัดเส้นทางระหว่าง (S, D) บนแต่ละระบบย่อยจากต้นทาง S^1 = 9 (1001_2) ด้วยค่าควบคุมหลัก C_4^1 = 12 (1100_2) ในกลุ่มย่อยที่ 1 (y = 1) สำหรับเชื่อมต่อไปยังปลายทาง $D^1 = S^1 \oplus C^1$ = 9 (1001_2) $\oplus$ 12 (1100_2) = 5 (0101_2) ดังนั้นการเชื่อมต่อระหว่างพอร์ตเข้า/พอร์ตออกของครอสส์บาร์ คือการกำหนดโดยใช้บิตที่มีนัยสำคัญสูง 1 บิต (s_3 = 1, d_3 = 0) ของต้นทาง S และปลายทาง D

4.2.3 ขั้นตอนวิธีแบบ "วดี" ที่เหมาะสมที่สุดบนเครือข่ายแบบ PMINs$^+$

ขั้นตอนวิธีแบบ ATAPE ที่สามารถแบ่งได้ ในที่นี้เรียกว่าขั้นตอนวิธีแบบ "วดี" (VA-DE : Valuable ATAPE with Dynamic Embedding) ที่เหมาะสมที่สุด ด้วยการจัดสรรงานที่ดีที่สุด (ที่จัดสรรได้ครบทั้ง 3 ปัจจัย) โดยที่งานจำนวน nT งาน ($T_0, T_1, \ldots, T_{nT-1}$) ของขนาดที่แตกต่าง $N'' = N/2^t$ บน $x \times x$ ครอสส์บาร์ของ PMINs$^+$ ในขั้นตอนวิธีที่ 4.1 งานขนาด N สามารถประมวลผลโดยใช้เทคนิคซุปเปอร์ไปป์ไลน์ (ดีกรี x) บนทุกๆ กลุ่มย่อยด้วยเวลาที่เหมาะสม $O(N/x + n')$ พิสูจน์ในทฤษฎีบทที่ 4.1 ซึ่งดีกว่าบนเครือข่ายแบบ CMINs$^+$ [9] ที่ได้ผลลัพธ์ด้วยความซับซ้อนด้านเวลา $O(N + n')$ ในการจัดสรรงานที่เหมาะสม ในหัวข้อนี้ได้เสนอการแปลงตำแหน่งที่อยู่ภายในกลุ่มย่อยและทั้งระบบของต้นทางและปลายทาง (S, D) แบบใหม่ในสมการที่ (3) โดยการจัดสรรตำแหน่งที่อยู่จริงของงาน 2^t งาน ขนาด $N''=N/2^t$ ได้อย่างมีประสิทธิภาพยิ่งขึ้นในทุกๆ กลุ่มย่อยด้วยเวลาที่เหมาะสมยิ่งขึ้น $O(N''/x + n')$

$$lS^y = S^y \bmod N'' \text{ และ } lD^y = (D^y \bmod N'') \oplus c^z \qquad (3)$$

โดยที่ $c^z = zN''/x$, $N'' = N/2^t$ และ z = 0, 1, 2, ..., 2^t-1

<u>ทฤษฎีบทที่ 4.3</u> ความซับซ้อนด้านเวลาของการประมวลผลงานแบบ ATAPE จำนวน 2^t งาน ขนาด $N''= N/2^t$ โดยที่ $t \le \log_2 x$ (2^t งานต่อกลุ่มย่อย) บน $x \times x$ ครอสส์บาร์ของ PMINs$^+$ (N และ $N'=N/x$) คือ $O(N''/x+n')$ ซึ่งเป็นเวลาที่ดีที่สุดด้วยการจัดสรรงานแบบซุปเปอร์ไปป์ไลน์ (ดีกรี x) บน x กลุ่มย่อยได้พร้อมๆ กัน

พิสูจน์ สำหรับการติดต่อสื่อสารแบบ ATAPE ของงาน (ขนาด N) บน PMIN$^+$ แต่ละกลุ่มย่อย y ของทุกๆ x กลุ่มย่อยจะเริ่มใช้ x ค่าควบคุมหลักได้ (ในแต่ละ N/x ขั้นตอน) ในหนึ่งสัญญาณนาฬิกา

- ในขั้นตอนแรก (l = 0) แต่ละค่าของค่าควบคุมหลัก x ค่า C_0^0 = 0, $C_0^1 = N/x$, ..., $C_0^{x-1} = (x-1)N/x$ สามารถเริ่มได้ในทุกๆ $1/x$ สัญญาณนาฬิกา (บนกลุ่มย่อย y = 0 จนถึง x-1) เพื่อส่ง x ชุดของข้อมูลจากทุกๆ N ต้นทาง S^y ผ่านสวิทช์ภายใน n' สเตจของแต่ละ MINs$^+$ ไปยังทุกๆ N ปลายทาง $D^y = S^y \oplus C_0^y$

- ในขั้นตอนที่สอง (l = 1) ค่าควบคุมหลักอีก x ค่าถัดไป C_1^0 = 1, C_1^1 = N/x+1, ..., C_1^{x-1} = (x-1)N/x+1 จะเริ่มในทุกๆ 1/x สัญญาณนาฬิกาเพื่อส่งข้อมูล x ชุดถัดไป
- สามารถทำการประมวลผลด้วยกระบวนการเดิมซ้ำๆ สำหรับ l = 2, 3, ..., N/x-1 ดังนั้นในขั้นตอนสุดท้าย l = N/x-1 ค่าควบคุมหลัก x ค่าสุดท้าย $C_{N/x}^0$= N/x-1, $C_{N/x}^1$= 2N/x-1,..., $C_{N/x}^{x-1}$= N-1 จะเริ่มในทุกๆ 1/x สัญญาณนาฬิกาเพื่อส่งข้อมูล x ชุดสุดท้าย

ดังนั้นความซับซ้อนด้านเวลาของการประมวลผลงานของขนาด N คือ $O(N/x+n')$ เพราะว่าค่าควบคุมหลักแรก C = 0 ใช้เวลา n' สัญญาณนาฬิกาและค่าควบคุมหลัก (N-1) ที่เหลือใช้เวลา (N-1)/x สัญญาณนาฬิกา ในทำนองเดียวกันสำหรับงานจำนวน 2^t งาน (ขนาด N'' = $N/2^t$) โดย x ค่าควบคุมหลักทั้งหมดจะถูกประมวลผลพร้อมๆ กัน (สำหรับ l = 0, 1, ... , N''/x) ในเวลา N''/x + n'-1 สัญญาณนาฬิกา ดังนั้น $O(N''/x + n')$ คือเวลาที่เหมาะสมที่สุดดังในรูปที่ 4.12 (ก-ข) สำหรับ N''=$N/2$ ในกรณี (N'' < N) จะมีกระบวนการแปลงตำแหน่งที่อยู่ภายในกลุ่มย่อยและทั้งระบบได้อย่างมีประสิทธิภาพในสมการที่ (3) ที่ถูกนำมาประยุกต์ใช้ (ดังในรูปที่ 4.13 (ก)) สำหรับการจัดสรรงานที่เหมาะสม □

สำหรับตัวอย่างในรูปที่ 4.12 (ก-ข) แสดงลำดับและการจัดสรรงานที่เหมาะสมบนกลุ่มย่อยที่ถูกต้องของงานจำนวน 2 งาน (ขนาด N'' = $N/2$) บน x x x ครอสส์บาร์ของ PMINs$^+$ (x = 4) ที่ l = 0, 1, 2,..., $N''/4$ ในรูปที่ 4.13 (ก) แสดงการแบ่งตารางลาตินทั้งหมดของค่าควบคุมหลัก C_l^y สำหรับ 2 รอบภายใน (ต่อกลุ่มย่อย) ประมวลผลใน 4 กลุ่มย่อย y (= 0, 1, 2, 3) สำหรับ N'' = $N/2$ การแปลงตำแหน่งที่อยู่ภายในกลุ่มย่อยคือ IS^y = S^y mod N'' และ ID^y = (D^y mod N'') $\oplus$ c^z โดยที่ c^z = zN''/x ในกรณีนี้จะมี N''/x = 4 รอบของทั้งระบบ (l = 0, 1, 2, 3) และ N/N'' = 2 รอบตำแหน่งที่อยู่ภายในกลุ่มย่อย (z = 0, 1) เช่นสมมุติว่าวิธีเรียงสับเปลี่ยนของปลายทางทั้งระบบคือ [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 | 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31] จะแบ่งออกเป็นวิธีเรียงสับเปลี่ยนของภายในกลุ่มย่อยได้ 2 ชุดคือ [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15] และ [4 5 6 7 0 1 2 3 12 13 14 15 8 9 10 11] ในทำนองเดียวกัน สำหรับกรณี N'' = $N/4$ (ในรูปที่ 4.13 (ข)) จะมีจำนวนรอบของทั้งหมด N''/x = 2 รอบ (l = 0, 1) และจำนวนรอบภายในกลุ่มย่อยจำนวน N/N'' = 4 รอบ (z = 0, 1, 2, 3) ต่อหนึ่งกลุ่มย่อย ซึ่งผลลัพธ์ของ pATAPE ที่สมบูรณ์ได้แสดงดังรูปที่ 4.17 (N = 32 N''= 8 และ x = 4)

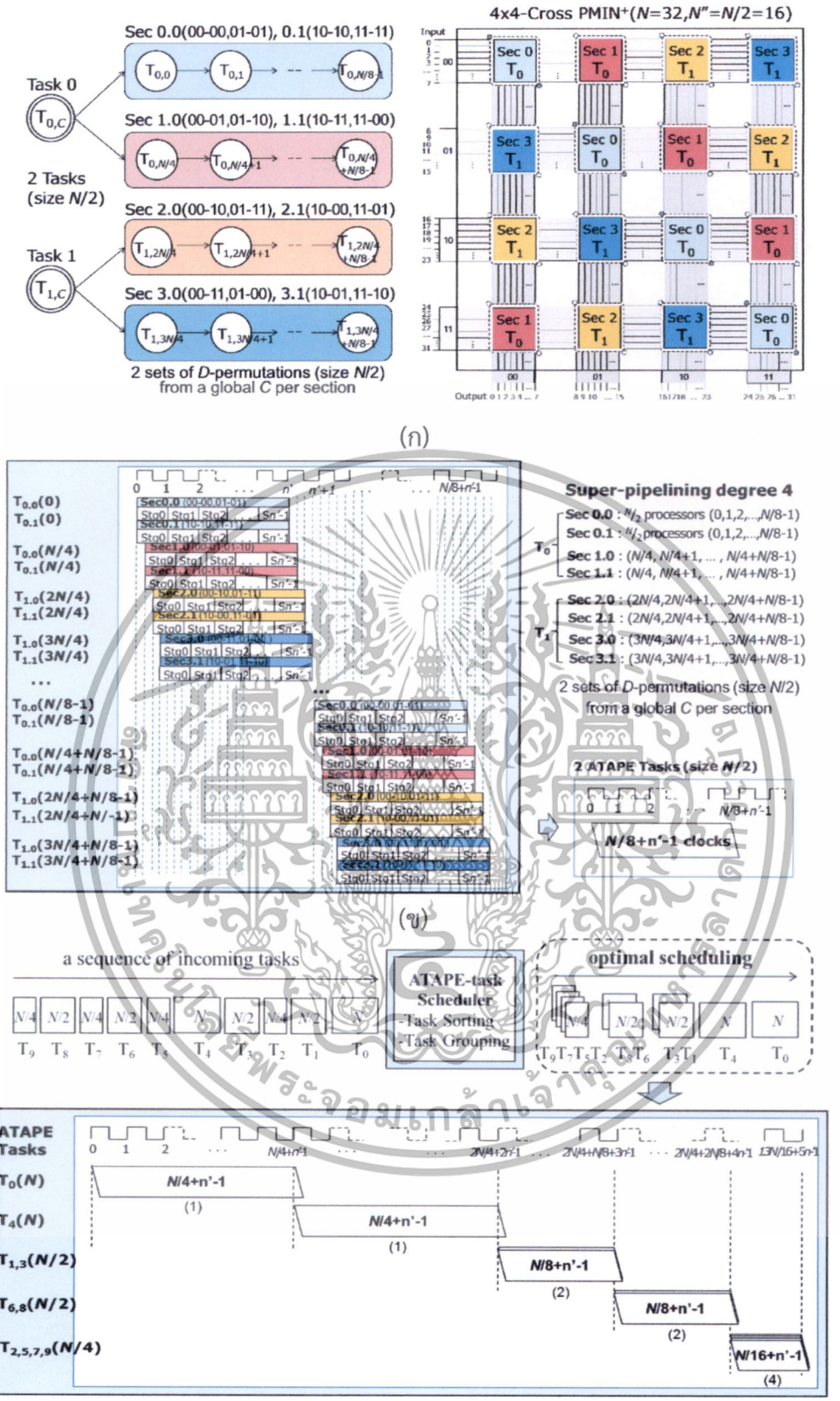
หลังจากนั้นในการประมวลผลงานหลายๆ งานซึ่งมีขนาดที่แตกต่างบนเครือข่ายแบบ PMINs$^+$ จะกำหนดให้ T_0, T_1, T_2, T_3, ..., T_{nT-1} เป็นลำดับของงานแบบ ATAPE จำนวน nT งาน (ที่มีขนาดใดๆ ก็ได้ระหว่าง N/x และ N) ในการจัดสรรงานที่ดีที่สุดจะประมวลผลงานหลังจากการเรียงลำดับงานจากมากไปน้อยและจัดสรรให้งานที่มีขนาดใหญ่ถูกประมวลผลก่อน โดยที่งาน (ขนาด N) ใช้เวลาประมวลผล $N/x+n'-1/x$ สัญญาณนาฬิกา และงาน (ขนาด $N/2$) จำนวน 2 งาน สามารถประมวลผลได้อย่างมีประสิทธิภาพ (ในเวลาเดียวกัน) ต่อหนึ่งกลุ่มย่อย ในเวลา $N/(2x)$ +n'-1 สัญญาณนาฬิกา ในทำนองเดียวกัน งานจำนวน 4 งาน (ขนาด $N/4$) ใน $N/(4x)$ +n'-1 สัญญาณนาฬิกา งาน (ขนาด $N/2^t$) จำนวน 2^t งาน ใน $N/(2^tx)$ +n'-1 สัญญาณนาฬิกา และงาน (ขนาด N/x) จำนวน x งาน ใน $N/(x^2)$ +n'-1 สัญญาณนาฬิกา เช่นในรูปที่ 4.12 (ค) แสดงการจัดสรรงานที่เหมาะสมของการประมวลผลงานจำนวน 10 งาน ($T_0(N)$, $T_1(N/2)$, $T_2(N/4)$, $T_3(N/2)$, $T_4(N)$, $T_5(N/4)$, $T_6(N/2)$,) $T_7(N/4)$, $T_8(N/2)$, $T_9(N/4)$ บน 4x4 ครอสส์บาร์ของ PMINs$^+$ ภายหลังจากการเรียงลำดับงานจำนวน 10 งานสามารถจัดสรรเพื่อประมวลผลตามลำดับดังนี้ $T_0(N)$, $T_4(N)$, [$T_1(N/2)$, $T_3(N/2)$], [$T_6(N/2)$, $T_8(N/2)$], [$T_2(N/4)$, $T_5(N/4)$, $T_7(N/4)$, $T_9(N/4)$] โดยใช้เวลาเท่ากับ 2($N/4$+n'-1)+2($N/8$+n'-1)+($N/16$+n'-1) =13$N/16$+5(n'-1) สัญญาณนาฬิกา

ขั้นตอนวิธีที่ 4.1 ขั้นตอนวิธีแบบ "วดี" (VA-DE : Valuable ATAPE with Dynamic Embedding) ที่เหมาะสมที่สุดด้วยการจัดสรรงานที่ดีที่สุด (ที่จัดสรรได้ครบทั้ง 3 ปัจจัย) บน $x \times x$ ครอสส์บาร์ของ PMINs$^+$

Network System: an $x \times x$ crossbar of PMINs$^+$ ($N = 2^n$, $x = 2^b$) and
MIN$^+$ subsystem ($N' = 2^{n'}$; $n' = \log_2 N'$ stages).
ATAPE Task: task size $N'' = N/2^t$ (i.e., N, $N/2$, ..., N/x); $n'' = n-t$.
Task Scheduler : a sequence of many tasks ($T_0, T_1, T_2, ..., T_{nT}$).

For a set of 2^t tasks (size $N''=N/2^t$) with 2^t local controls per section)
  For all sections y = 0 to x-1
    (using I/O-switches: I = 0,1,...,x-1; O = (I+y) mod x)
  Call pATAPE(y) with super-pipelining degree x
    (every 1/x clock can start the next successive section y,
    and hence the same section can start in every clock.)
  end For all y
end For a set of 2^t tasks.

Function pATAPE(y) begin
  For l = 0 to $N''/x - 1$
    (with multistage super-pipelining (degree x))
    For all processors S ($0 \leq S < N$) do in parallel
      $c_l^y = yN/x + l$
      $D^y = s^y \oplus c_l^y$
    (all local source processors $ls^y = s^y$ mod N'' send their personalized massages to local destination processors $ld^y = (D^y \bmod N'') \oplus c^z$; where $c^z = zN''/x$, z = 0,1,..,2^t-1)
  end For all processors
  end For l
end Function.



รูปที่ 4.12 ตัวอย่างของการจัดสรรงานที่ดีที่สุด บน $x \times x \times x$ ครอสส์บาร์ของ PMINs$^+$ ($N = 2^n$, x=4) (ก) ขั้นตอนการทำงานของค่าควบคุมหลักสำหรับ 4 กลุ่มย่อยด้วยเทคนิคซุปเปอร์ไปป์ไลน์ สำหรับงานของขนาด $N'' = N/2$ (ข) แผนภาพเวลาของงาน (ขนาด $N/2$) และ (ค) ลำดับของงานขนาดระหว่าง $N/4$ และ N จำนวน 10 งาน (T_0 - T_9)

4.3 การจัดสรรงานงานแบบ ATAPE หลายๆ งานบน $x \times x \times x$ ครอสส์บาร์ของ PMINs$^+$

การวัดประสิทธิภาพของการจัดสรรงานแบบ ATAPE บน $x \times x \times x$ ครอสส์บาร์ของ PMINs$^+$ อันดับแรกหัวข้อที่ 4.3.1 แสดงผลลัพธ์ของการประมวลผลทีละขั้นตอน เพื่อให้เห็นภาพรวมทั้งหมดชัดเจนจำนวน 2 ผลลัพธ์ของแอพพลิเคชั่นแบบ ATAPE ที่สามารถแบ่งกลุ่มย่อยได้ โดยที่ผลลัพธ์ของงาน (ขนาด $N" = N$) บนเครือข่าย $N = 16$, $N' = 8$, $x = 2$ (ในรูปที่ 4.16) และอีกผลลัพธ์ของหลายๆ งาน (ขนาด $N" = N/2$) บนเครือข่าย $N = 32$, $N' = 8$, $x = 4$ แสดง (ในรูปที่ 4.17) ดังนั้นประสิทธิภาพของอัตราเร็วที่เพิ่มขึ้น (Speedup) และปริมาณงานที่เสร็จภายในช่วงเวลาหนึ่งๆ เช่น (1 ชั่วโมงต่อ 1 วัน เป็นต้น) (Throughput) ของการจัดสรรงานที่เหมาะสมที่สุดบน $x \times x \times x$ ครอสส์บาร์ของ PMINs$^+$ ในหัวข้อที่ 4.3.2 (ในตารางที่ 4.3 - 4.4) ซึ่งเป็นผลลัพธ์จากการทดลอง (แสดงในรูปที่ 4.14-4.15) โดยการจำลองให้มีหน่วยประมวลผล N หน่วยได้ถึง 32,768 หน่วยประมวลผล

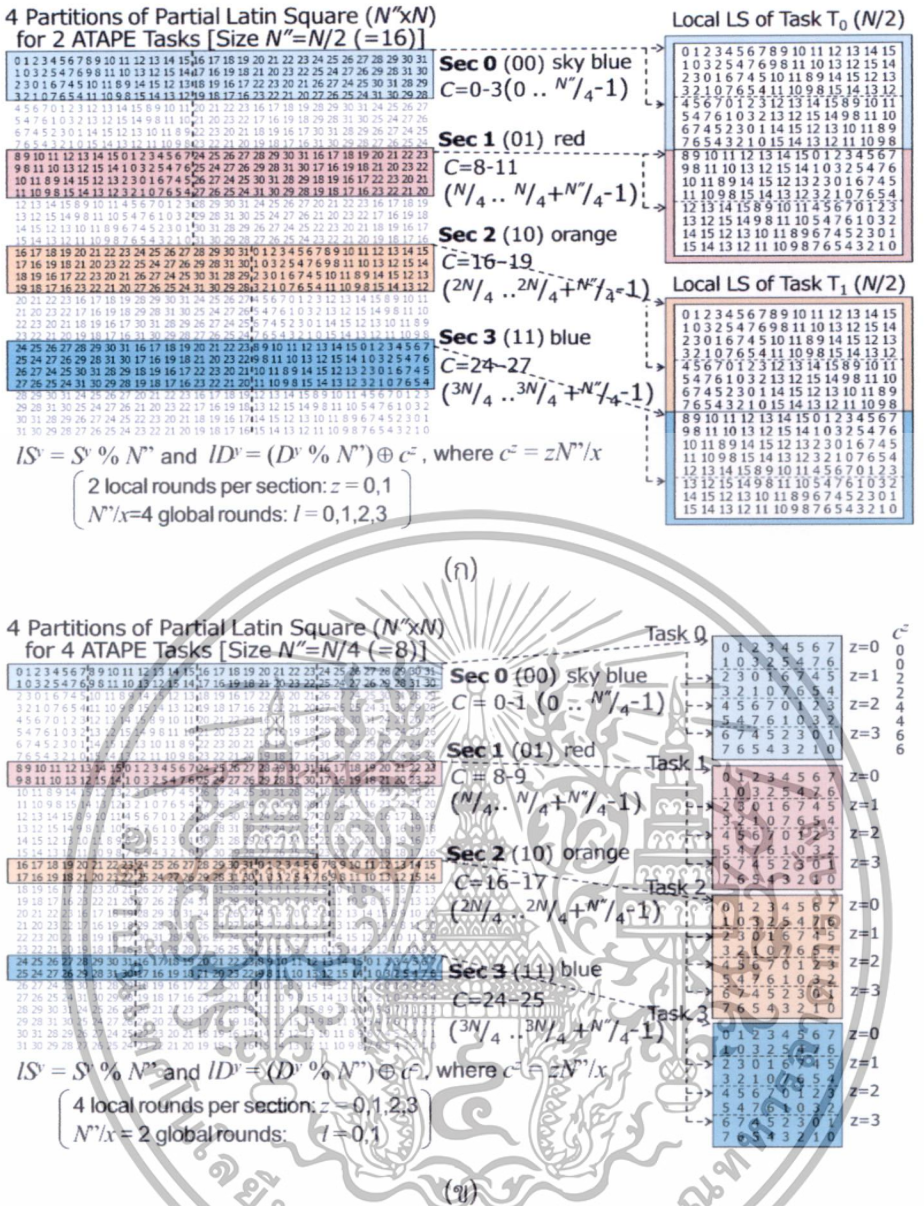
4.3.1 งานแบบ ATAPE ที่สามารถแบ่งกลุ่มย่อยได้บนเครือข่ายแบบ PMINs$^+$

ในรูปที่ 4.16 แสดงการนำผลลัพธ์ของขั้นตอนวิธี *p*ATAPE ไปใช้ได้อย่างมีประสิทธิภาพสำหรับการจัดสรรงานของงานแบบ ATAPE ขนาด $N" = N$ บน 2x2 ครอสส์บาร์ของ PMINs$^+$ ($N = 16$, $x = 2$, $b = 1$) ด้วยระบบย่อยของ MINs$^+$ ($N' = 8$, $n' = 3$) โดยเทคนิคซุปเปอร์ไปป์ไลน์ (ดีกรี $x = 2$) มีกลุ่มย่อย 2 กลุ่ม (กลุ่ม $y = 0, 1$) ที่สลับกันใช้ระบบ (ในทุกๆ ½ สัญญาณนาฬิกา) สำหรับค่าควบคุมหลัก N รอบภายใน $N/2$ ขั้นตอน ($l = 0, 1, 2, ..., 7$) ดังนี้

กลุ่ม 0 ($y = 0$) $\longrightarrow C_l^0 : \{0, 1, 2, 3, 4, ..., 7\}$ และ $D^0 = S^0 \oplus C_l^0$
โดย I/O ครอสส์สวิทช์ที่ 0-0, 1-1

กลุ่ม 1 ($y = 1$) $\longrightarrow C_l^1 : \{8, 9, 10, 11, ..., 15\}$ และ $D^1 = S^1 \oplus C_l^1$
โดย I/O ครอสส์สวิทช์ที่ 0-1, 1-0

ในแต่ละรอบ (ในกลุ่มย่อย y) การจัดเส้นทางภายในของ (S^y, D^y) คำนวณด้วยฟังก์ชัน *p*ATAPE คือ $D^y = S^y \oplus C_l^y$ แสดงในแต่ละแถวของตารางตินเพื่อติดต่อสื่อสารภายในกลุ่มย่อย 0 และ 1 พร้อมๆ กันโดยเริ่มห่างกันครึ่งสัญญาณนาฬิกา ในรูปที่ 4.16 การกำหนดค่าเครือข่าย N ค่า ($N = 16$) ของการนำฟังก์ชัน *p*ATAPE ไปใช้บนครอสส์บาร์ของ MINs$^+$ แบบไปข้างหน้า ($N' = 8$) สำหรับทุกๆ การเชื่อมต่อระหว่างต้นทางและปลายทางภายในแต่ละกลุ่มย่อย y (S^y, D^y) ในรอบแรกทุกๆ ข้อมูลแบบเพอร์ซันนัลไลซ์ของกลุ่มย่อยที่ 0 ($N_{All}^{y=0}$) ด้วยการใช้ค่าควบคุมหลัก ($C_0^0 = 0$) เพื่อส่งข้อมูลผ่าน I/O ครอสส์สวิทช์เพื่อทำการกำหนดระบบย่อยที่ (0, 0) และ (1, 1) ให้สามารถใช้งานได้ในอีกครึ่งสัญญาณนาฬิกาถัดไปทุกๆ ข้อมูลแบบเพอร์ซันนัลไลซ์ในกลุ่มย่อยที่ 1 ($N_{All}^{y=1}$) ด้วยการใช้ค่าควบคุมหลัก ($C_0^0 = 1$) เพื่อส่งข้อมูลผ่านค่าควบคุมหลักของ I/O ครอสส์สวิทช์เพื่อทำการกำหนดระบบย่อยที่ (0, 1) และ (1, 0) ให้สามารถใช้งานได้ โดยเฉพาะอย่างยิ่งในกรณี $l = 0$ (หรือ 000_2) จะใช้ 2 ค่าควบคุมหลัก (C_0^0 และ C_0^1) ในการกำหนดแบบของสวิทช์ในทุกๆ สวิทช์ผ่าน n'(= 3) สเตจ ในทำนองเดียวกัน $l = 1$ (หรือ 001_2) ใช้ 2 ค่าควบคุมหลัก ($C_1^0 = 1$ และ $C_1^1 = 9$) ในการกำหนดแบบของสวิทช์แบบ 001_2 (คือกำหนดแบบตรง (-) ในสเตจที่ 0-1 และกำหนดแบบไขว้ (x) ในสเตจที่ 2) และทำในทำนองเดียวกันสำหรับ $l = 2, ..., 7$ โดยรอบสุดท้ายของขั้นตอนสุดท้าย $l = 7$ (= 111_2) จะใช้ 2 ค่าควบคุมหลัก $C_7^0 = 7$ และ $C_7^1 = 15$ เพื่อทำการส่งทุกๆ ข้อมูลแบบเพอร์ซันนัลไลซ์ ($N_{All}^{y=0}$, $N_{All}^{y=1}$) ได้ครบทุกๆ ข้อมูล



รูปที่ 4.13 การแปลงตำแหน่งที่อยู่ภายในกลุ่มย่อยและทั้งระบบสำหรับงานของขนาด $N" = N/2^t$ จำนวน 2^t งาน บน $x \times x$ ครอสส์บาร์ของ PMINs$^+$ ($N = 32$, $x = 4$) (ก) งานขนาด $N" = N/2$ และ (ข) งานขนาด $N" = N/4$

สำหรับในรูปที่ 4.17 แสดงอีกตัวอย่างของการใช้ฟังก์ชัน *p*ATAPE เพื่อจัดสรรงานขนาด $N" = N/4$ จำนวน 4 งาน บน 4x4 ครอสส์บาร์ของ PMINs$^+$ ($N = 32$, $N' = 8$, $x = 4$) โดยเน้นที่การแปลงตารางลาตินของกลุ่มย่อยเทียบกับตารางลาตินทั้งหมด ในกรณีที่เครือข่ายถูกแบ่งออกเป็น 4 กลุ่มย่อย ($y = 0, 1, 2, 3$) โดยที่ค่าควบคุมหลัก C_l^y ของแต่ละกลุ่มย่อย y จะทำการประมวลผลใน $N"/x$ ขั้นตอน ($l = 0, 1, 2, \ldots, N"/x\text{-}1$)

กลุ่ม 00 ($y = 0$) $\longrightarrow$ $C_l^0 : \{0, 1\}$ และ $D^0 = S^0 \oplus C_l^0$

ใช้ I/O ครอสส์สวิทช์ที่ 00-00, 01-01, 10-10, 11-11

กลุ่ม 11 ($y = 1$) $\longrightarrow$ $C_l^1 : \{8, 9\}$ และ $D^1 = S^1 \oplus C_l^1$

ใช้ I/O ครอสส์สวิทช์ที่ 00-01, 01-10, 10-11, 11-00

กลุ่ม 10 (y = 2) ⟶ C_l^2 : {16, 17} และ $D^2 = S^2 \oplus C_l^2$

ใช้ I/O ครอสส์สวิทช์ที่ 00-10, 01-11, 00-00, 11-01

กลุ่ม 11 (y = 3) ⟶ C_l^3 : {24, 25} และ $D^3 = S^3 \oplus C_l^3$

ใช้ I/O ครอสส์สวิทช์ที่ 00-11, 01-00, 10-01, 11-10

ในแต่ละรอบ (ของกลุ่มย่อย y = 0, 1, 2, 3) การจัดเส้นทางภายในระหว่างต้นทางและปลายทาง (S^y, D^y) จะถูกคำนวณด้วยฟังก์ชัน pATAPE ของตำแหน่งที่อยู่ทั้งระบบด้วย $D^y=S^y \oplus C_l^y$ ในรูปที่ 4.13 (ข) แสดงตารางลาตินของการติดต่อสื่อสารระหว่างต้นทางและปลายทางในกลุ่มย่อย y (S^y, D^y) ของตำแหน่งที่อยู่ทั้งระบบ ซึ่งหมายถึงการติดต่อสื่อสารระหว่าง (lS^y, lD^y) ในกลุ่มย่อยสำหรับแต่ละงานในการประมวลผลแบบซุปเปอร์ไปป์ไลน์ (ดีกรี x = 4) โดยกลุ่มย่อยที่ 0 ถึง 3 จะถูกจัดสรรพร้อมๆ กันโดยเริ่มในทุกๆ ¼ สัญญาณนาฬิกา ดังนั้นการกำหนดค่าเครือข่ายขนาด $N'' = N/4$ ต่องาน (ดังแสดงผลลัพธ์ในรูปที่ 4.17) การนำ pATAPE ไปใช้สำหรับงาน (ของขนาด N'') จำนวน 4 งาน บนครอสส์บาร์ของ MINs$^+$ แบบย้อนกลับ (N' = 8, n' = 3) ผ่าน n' สเตจ ในกรณีที่ l = 0, 1

กลุ่มย่อยที่ 0 จะถูกแบ่งสำหรับ T_0 โดยใช้ค่าควบคุมหลัก C_l^0

กลุ่มย่อยที่ 1 จะถูกแบ่งสำหรับ T_1 โดยใช้ค่าควบคุมหลัก C_l^1

กลุ่มย่อยที่ 2 จะถูกแบ่งสำหรับ T_2 โดยใช้ค่าควบคุมหลัก C_l^2 และ

กลุ่มย่อยที่ 3 จะถูกแบ่งสำหรับ T_3 โดยใช้ค่าควบคุมหลัก C_l^3

ดังนั้นตำแหน่งที่อยู่ทั้งระบบของผลลัพธ์ในแต่ละรอบจะถูกแบ่งและแปลงเป็นผลลัพธ์ของกลุ่มย่อยในตารางลาติน 4 ผลลัพธ์ เช่นวิธีเรียงสับเปลี่ยนของตำแหน่งที่อยู่ทั้งระบบ ในรอบที่ 0 ประกอบด้วย [0 1 2 3 4 5 6 7| 8 9 10 11 12 13 14 15 | 16 17 18 19 20 21 22 23 | 24 25 26 27 28 29 30 31] หมายถึงการติดต่อสื่อสารของ 4 วิธีเรียงสับเปลี่ยนในกลุ่มย่อย y=0 ดังนี้ [0 1 2 3 4 5 6 7], [2 3 0 1 6 7 4 5], [4 5 6 7 0 1 2 3] และ [6 7 4 5 2 3 0 1] ส่วนกลุ่มย่อยอื่นๆ (y = 1, 2, 3) ก็จะดำเนินการในทำนองเดียวกัน

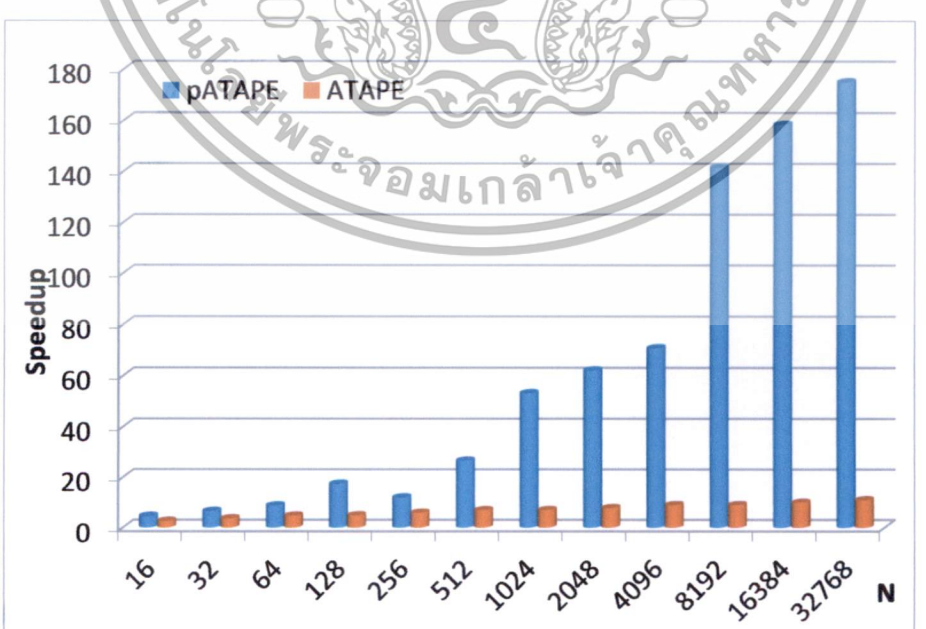
4.3.2 การจัดสรรงานของงานแบบ ATAPE หลายๆ งานบนเครือข่ายแบบ PMINs$^+$

การวัดประสิทธิภาพ (โดยการศึกษาแบบจำลองระบบให้หน่วยประมวลผลมีขนาด $N \leq$ 32768 และดีกรีของครอสส์บาร์ขนาด $x \leq 16$) ซึ่งได้แสดงผลการเปรียบเทียบค่าอัตราเร็วที่เพิ่มขึ้น (Speedup) และปริมาณงานที่ประมวลผลเสร็จในช่วงเวลาที่ทำการสังเกตและวัดผล (Throughput) ของ ATAPE แบบเดิม [9] กับผลลัพธ์ของ pATAPE แบบใหม่ในตารางที่ 4.3 (อัตราเร็วที่เพิ่มขึ้น) และตารางที่ 4.4 (ปริมาณงาน) สำหรับ N ที่มีขนาดเล็กจะใช้บน 2x2 ครอสส์บาร์ สำหรับ N ที่มีขนาดใหญ่จะใช้บน 16x16 ครอสส์บาร์ (เช่น x = 16 สำหรับ N = 8192 จนถึง 32768) ในการทดลองค่าอัตราเร็วที่เพิ่มขึ้นของ pATAPE จะทำได้ดีกว่าของ ATAPE แบบเดิม [9] มากถึง x เท่า เนื่องจากมีประสิทธิภาพของซุปเปอร์ไปป์ไลน์เข้ามาช่วยด้วยค่า CPI = 1/x บนเครือข่ายแบบ PMINs$^+$ ดังแสดงผลจากการทดลองทั้งหมดในตารางที่ 4.4 และแสดงการเปรียบเทียบในรูปที่ 4.14

ส่วนปริมาณงาน (Throughput) ของผลลัพธ์ (รูปที่ 4.15) ได้มีการวัดในช่วงเวลาที่สังเกต (เช่น ช่วงเวลา 1M สัญญาณนาฬิกา) โดยที่จำนวนงานที่เสร็จสมบูรณ์ต่อเวลาในการสังเกตเพิ่มขึ้นอย่างมากเมื่อมีการประมวลผลงานขนาดเล็กๆ และปริมาณลดลงเมื่อทำการประมวลผลงานขนาดใหญ่ๆ เช่นในการเปรียบเทียบค่าดังกล่าว pATAPE สามารถทำงานขนาด N ด้วย PMINs$^+$ ขนาด N = 8,192 เสร็จสมบูรณ์จำนวน 32,264 งาน เทียบกับ cATAPE แบบเดิม [9] สามารถประมวลผลได้เพียง 128 งานเท่านั้นในช่วงเวลาเดียวกัน (ตารางที่ 4.4)

ตารางที่ 4.3 อัตราเร็วที่เพิ่มขึ้น (Speedup = Tw/o pipelining / Twith pipelining) ของ ATAPE เดิมบนเครือข่ายแบบ CMINs$^+$ (CPI = 1) [9] และ *p*ATAPE แบบใหม่บนเครือข่ายแบบ PMINs$^+$ (CPI = 1/*x*)

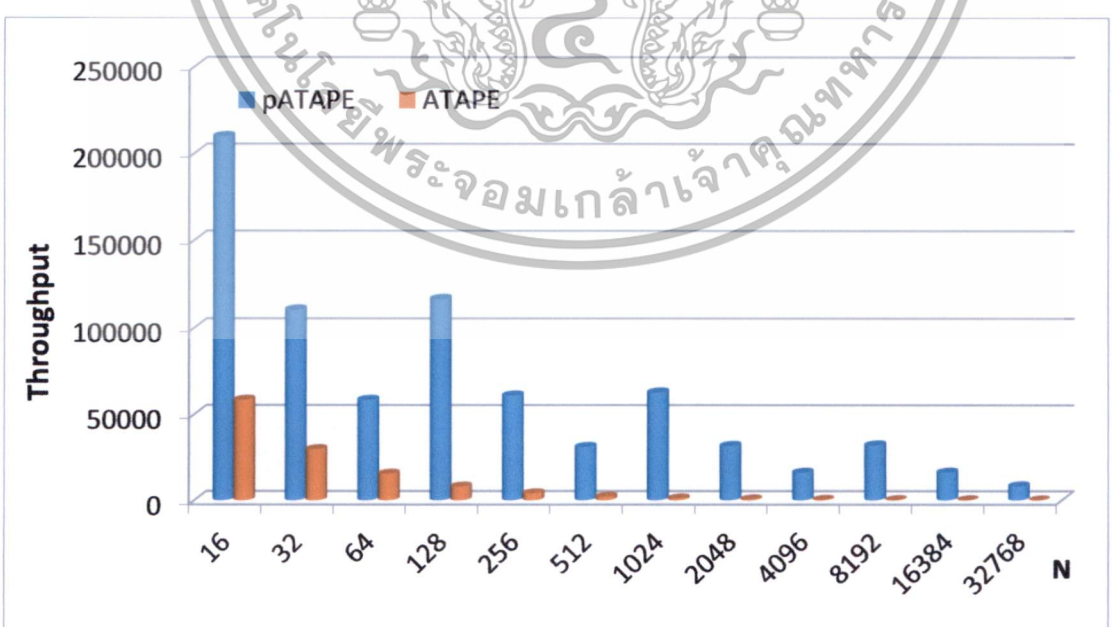
x x x crossbar of MINs$^+$ / PMINs$^+$	เวลาในการประมวลผล (Computing Time) (จำนวนสัญญาณนาฬิกา)		ค่าอัตราเร็วที่เพิ่มขึ้น (Speedup)	
	ATAPE เดิม [9] T= $N + n'-1$	*p*ATAPE ใหม่ T=$N/x+n'-1/x$	ATAPE เดิม [9]	*p*ATAPE ใหม่
2x2 crossbar				
N =16, N' = 8	18.0	10.5	2.67	4.57
N =32, N' = 16	35.0	19.5	3.66	6.56
N =64, N' = 32	68.0	36.5	4.71	8.77
4x4 crossbar				
N =128, N' = 32	132.0	36.8	4.85	17.41
N =256, N' = 64	261.0	69.8	5.89	12.02
N =512, N' = 128	518.0	134.8	6.92	26.60
8x8 crossbar				
N =1024, N' = 128	1,030.0	134.9	6.96	53.15
N =2048, N' = 256	2,055.0	263.9	7.97	62.09
N =4096, N' = 512	4,104.0	520.9	8.98	70.77
16x16 crossbar				
N = 8192, N' = 512	8,200.0	520.9	8.99	141.53
N =16384, N' = 1024	16,393.0	1,033.9	9.99	158.46
N =32768, N' = 2048	32,778.0	2,058.9	11.0	175.07



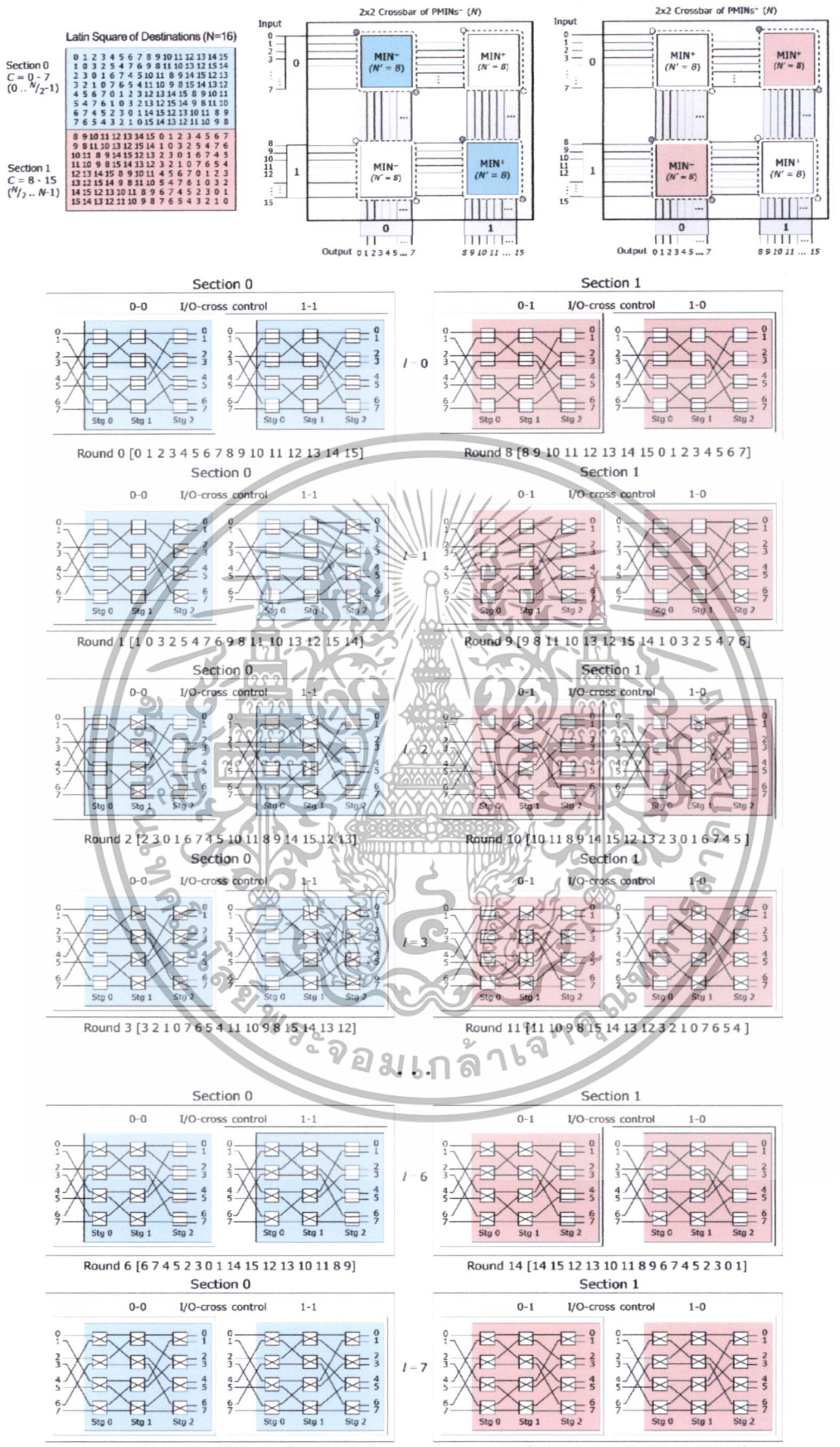
รูปที่ 4.14 อัตราเร็วที่เพิ่มขึ้นของงานแบบ *p*ATAPE ด้วยซุปเปอร์ไปป์ไลน์ (ดีกรี *x*) เปรียบเทียบกับ ATAPE แบบเดิมบนเครือข่ายแบบ CMINs$^+$ [9] ด้วยเทคนิค *n'* สเตจไปป์ไลน์

ตารางที่ 4.4 ปริมาณงาน (งานที่ทำเสร็จต่อช่วงเวลาที่สังเกต) ของ ATAPE แบบเดิมบนเครือข่ายแบบ CMINs$^+$ [9] และ *p*ATAPE แบบใหม่บนเครือข่ายแบบ PMINs$^+$

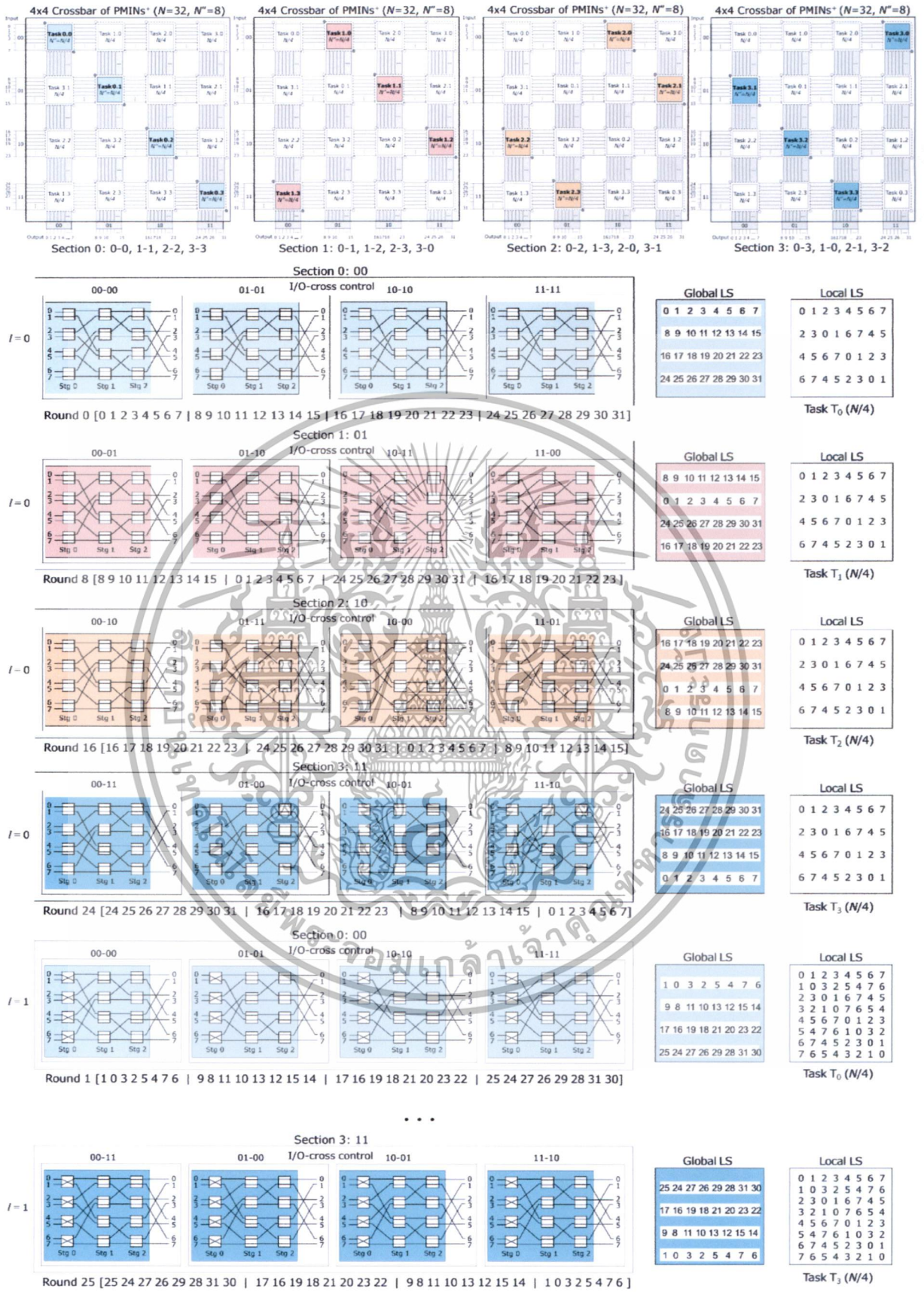
x x x crossbar of MINs$^+$ / PMINs$^+$	ขนาดของงาน = *N*		ขนาดของงานที่แตกต่างกัน (*N*, *N*/2, *N*/4, ..., *N*)	
	ATAPE เดิม [9]	*p*ATAPE ใหม่	ATAPE เดิม [9]	*p*ATAPE ใหม่
2x2 crossbar				
N =16, *N'* = 8	58254	209715	74898	262144
N =32, *N'* = 16	29959	110375	38836	139810
N =64, *N'* = 32	15420	58254	20165	74898
4x4 crossbar				
N =128, *N'* = 32	7944	116508	13329	185043
N =256, *N'* = 64	4018	60787	6794	99078
N =512, *N'* = 128	2024	31301	3442	51996
8x8 crossbar				
N =1024, *N'* = 128	1018	62602	2158	127100
N =2048, *N'* = 256	510	31896	1084	66052
N =4096, *N'* = 512	256	16132	544	33825
16x16 crossbar				
N = 8192, *N'* = 512	128	32264	329	81285
N =16384, *N'* = 1024	64	16241	165	41344
N =32768, *N'* = 2048	32	8158	83	20875



รูปที่ 4.15 ปริมาณงานของ *p*ATAPE ด้วยเทคนิคซุปเปอร์ไปป์ไลน์ (ดีกรี *x*) เปรียบเทียบกับ ATAPE แบบเดิมบนเครือข่ายแบบ CMINs$^+$ [9] ด้วยเทคนิค *n'* สเตจไปป์ไลน์



รูปที่ 4.16 ผลลัพธ์ของการจัดสรรงานแบบ *p*ATAPE (N"=16) จำนวน 2 งาน บน 2x2 ครอสส์บาร์ ของ PMINs^{+} (N=16) และระบบย่อย (โบนาร์สวิทช์ของ MINs^{+} ที่ $N' = 8, n' = 3$)



รูปที่ 4.17 ผลลัพธ์ของการจัดสรรงานแบบ *p*ATAPE ($N"$= 8) จำนวน 4 งาน บน 4x4 ครอสส์บาร์ ของ PMINs$^+$ (N=32) และระบบย่อย (ไบนารี่สวิทช์ของ MINs$^+$ ที่ N' = 8, n' = 3')

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

ในวิทยานิพนธ์นี้การศึกษาวิจัยในบทที่ 3 ได้เสนอฟังก์ชัน ATAPE (All-to-All Personalized Exchange) แบบฝังลงบนชิปสองชนิด คือ ฟังก์ชันแบบคงที่ $D = S \oplus (C + order) \bmod N$ และฟังก์ชันแบบปรับเปลี่ยนได้ (ฟังก์ชันทางเลือก) $D = \rho\,[\,(S + C + order) \bmod N]$ บนเครือข่ายแบบ MINs$^+$ ที่ได้ทำการปรับพอร์ตให้เป็นเครือข่าย MINs ที่สมบูรณ์ประกอบด้วย เครือข่ายแบบบันยานพลัส เครือข่ายแบบบัตเตอร์ฟลายพลัส เครือข่ายแบบเบสไลน์พลัส และเครือข่ายแบบอินเวอร์สเบสไลน์พลัส ทั้งสองฟังก์ชันสามารถจัดเส้นทางภายในแบบสมบูรณ์ได้ตลอดเส้นทาง (Full Self-Routing) ทั้งการจัดเส้นทางด้วยค่าควบคุมแบบสเตจ ค่าควบคุมแบบสวิทช์ และจัดเส้นทางภายในระหว่างต้นทางและปลายทาง (S, D) ได้ โดยที่ฟังก์ชัน ATAPE นี้จะใช้เวลาในอุดมคติตามค่าตัวนับบนชิปที่เพิ่มขึ้น (Incrementing On-Chip Counter) ร่วมกับมัลติสเตจไปป์ไลน์ (Multistage Pipelining) สำหรับถ่ายโอนข้อมูลระหว่างทุกๆ คู่ของหน่วยประมวลผลจำนวน N หน่วยประมวลผล โดยใช้ N รอบต่อเนื่อง (C = 0 ถึง N-1) ซึ่งจะอนุญาตให้ผู้ใช้โปแกรมสามารถระบุลำดับ (*order*) ของข้อความเริ่มต้นได้โดยที่ไม่ต้องทำการประมวลผลวิธีเรียงสับเปลี่ยนในตารางลาตินไว้ก่อน ด้วยเวลาที่เหมาะสมคือ $O(N + \log_2 N)$ โดยที่ $0 \le order < N$ ซึ่งทั้งสองฟังก์ชันสามารถฝังได้ดีที่สุดทั้งบนเครือข่ายแบบ MINs$^+$ และบนเครือข่ายแบบ CMINs$^+$ ที่เร็วขึ้น แต่เครือข่ายแบบ MINs$^+$ และ CMINs$^+$ ไม่สามารถแบ่งเป็นระบบย่อยได้ทำให้สามารถประมวลผลงานแบบ ATAPE ได้เพียงครั้งละงานเดียวเท่านั้น

ดังนั้นการศึกษาวิจัยในบทที่ 4 จึงได้เสนอ $x \times x$ ครอสส์บาร์ของ MINs$^+$ ที่สามารถแบ่งกลุ่มย่อยได้ (PMINs$^+$) เพื่อช่วยให้สามารถประมวลผลงานแบบ ATAPE ได้หลายๆ งานในเวลาเดียวกัน และเสนอขั้นตอนวิธีแบบวาดี (VA-DE : Valuable ATAPE with Dynamic Embedding) ที่เหมาะสมที่สุด ซึ่งก็คือขั้นตอนวิธีแบบ pATAPE แบบใหม่หรือ ATAPE ที่สามารถแบ่งกลุ่มย่อยได้เพื่อประมวลผลบนเครือข่ายแบบ PMINs$^+$ ด้วยมีความซับซ้อนด้านเวลาที่เหมาะสมที่สุดที่ดีกว่าที่มีอยู่เดิม และใช้ทรัพยากรได้เต็มประสิทธิภาพที่สุด สอดคล้องกับการจัดสรรงานที่ดีที่สุดที่จัดสรรได้ครบทั้ง 3 ปัจจัยสำคัญ (Triple-Right Scheduling) คือสามารถจัดลำดับงานย่อยให้เหมาะกับกลุ่มย่อยในเวลาที่เหมาะสม (Right task, Right section, Right time) โดยที่เครือข่ายแบบ PMINs$^+$ จะใช้ I/O สวิทช์ควบคุมแบบครอสส์ (I/O Cross-Control Switches) ร่วมกับเทคนิคซุปเปอร์ไปป์ไลน์ ในทุกๆ กลุ่มย่อย x กลุ่ม (y = 0, 1, 2, ... , x-1) สามารถใช้ระบบได้พร้อมๆ กัน เมื่อสามารถรองรับขนาดของงานที่หลากหลาย (N/x จนกระทั่ง N) และสามารถใช้งานระบบได้เต็มประสิทธิภาพ 100% ด้วยค่าควบคุมในแต่ละกลุ่ม y คือ $C_l^y = yN/x + l$ โดยที่ $0 \le y < x$ และ l = 0, 1, 2, ..., $< N$ ดังนั้นงานแบบ pATAPE (ของขนาด N) บนเครือข่ายแบบ PMINs$^+$ ทุกๆ ข้อมูลแบบเพอร์ซัลนัลไลซ์ N_{All}^y ในแต่ละกลุ่มย่อย y จากทุกๆ ต้นทาง (S^y) จะถูกส่งจากทุกๆ ปลายทาง ($D^y = S^y \oplus C_l^y$) ด้วยความซับซ้อนด้านเวลาเท่ากับ $O(N/x + n')$ ซึ่งดีกว่า $O(N + n')$ ของ ATAPE เดิมบนเครือข่ายแบบ CMINs$^+$ [9]

ในการทดลองเพื่อวัดประสิทธิภาพซึ่งวัดค่าอัตราเร็วที่เพิ่มขึ้นของการจัดสรรงานแบบ *p*ATAPE บนเครือข่ายแบบ PMINs$^+$ ให้ผลลัพธ์ที่ประสิทธิภาพสูงกว่าค่าดังกล่าวบนเครือข่ายแบบ CMINs$^+$ แบบเดิม ถึง x เท่า (โดยได้ทำการทดลองแบบการศึกษาจำลองระบบขนาด $N \leq 32{,}768$ หน่วยประมวลผลและกำหนดให้ $x \leq 16$)

5.2 ข้อเสนอแนะ

- ออกแบบขั้นตอนวิธี ATAPE เพื่อรองรับเครือข่ายที่มีขนาด $N \neq 2^n$ และเครือข่ายการเชื่อมต่อแบบมัลติสเตจพลัสที่ใช้สวิทช์ขนาดใดๆ ที่พอร์ตของสวิทช์นำข้อมูลเข้า a ไม่เท่ากับพอร์ตของสวิทช์นำข้อมูลออก b (in-port $a \neq$ out-port b)
- นำเสนอขั้นตอนวิธีใหม่สำหรับการติดต่อสื่อสารแบบ ATAPE ที่ง่ายและให้ผลดีกว่าแบบที่ได้มีการนำเสนอไว้แล้ว เช่น วิธีที่มีความซับซ้อนด้านเวลาน้อยกว่า $O(N/x)$
- ประยุกต์ใช้ขั้นตอนวิธีแบบขนานชนิดอื่นๆ และแอพพลิเคชั่นอื่นๆ พร้อมประสิทธิภาพที่เพิ่มขึ้น บนเครือข่ายการเชื่อมต่อแบบ PMINs$^+$

เอกสารอ้างอิง

[1] K. E. Batcher, The Flip Network in STARAN, in Processdings of International Conference on Parallel Processing, 1976, 65-71.

[2] Z. Chen, Z.J. Liu, Z.L. Qiu, Bidirectional shuffle–exchange network and tagbased routing algorithm, IEEE Commun. Lett. 7 (3) (2003) 121–123.

[3] W.Y. Chou, C. Chen, All-to-all personalized exchange in generalized shuffle-exchange networks, Theoret. Comput. Sci. 411 (3) (2010) 1669–1884.

[4] D.H. Lawrie, Access and alignment of data in an array processor, IEEE Trans.Comput. C-24 (12) (1975) 1145–1155.

[5] V.W. Liu, C. Chen, R.B. Chen, Optimal all-to-all personalized exchange in d-nary banyan multistage interconnection networks, J. Comb. Optim. 14 (10) (2007) 131–142.

[6] A. Massini, All-to-all personalized communication on multistage interconnection networks, Discrete Appl. Math. 128 (2) (2003) 435–446.

[7] D.S. Parker, Notes on shuffle/exchange-type switching networks, IEEE Trans. Comput. C-29 (3) (1980) 213–222.

[8] M.C. Pease, The indirect binary n-cube microprocessor array, IEEE Trans. Comput. C-26 (5) (1977) 458–473.

[9] R. Petagon and J. Werapun, Embedding the optimal all-to-all personalized exchange on multistage interconnection networks^{+}, J. Parallel Distrib. Comput. 88 (2016), 15-30.

[10] R. Petagon and J. Werapun, VA-DE: Valuable ATAPE with Dynamic Embedding and Super-pipeline Scheduling on Partitionable Multistage Interconnection Networks^{+}, J. Parallel Distrib. Comput. 102 (2017), 1-15.

[11] B. Prisacari, G. Rodriguez, C. Minkenberg, Generalized hierarchical all-to-all exchange patterns, in: 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, May 2013, pp. 537–547.

[12] D.S. Scott, Efficient all-to-all communication patterns in hypercube and mesh topologies, in: Proc. Sixth Distributed Memory Computing Conference, 1991, pp. 398–403.

[13] Y.J. Suh, K.G. Shin, Efficient all-to-all personalized exchange in multidi-mensional torus networks, in: Proc. Int. Conf. Parallel Processing, 1998, pp. 468–475.

[14] Y.J. Suh, K.G. Shin, All-to-all personalized communication in multidimensional torus and mesh networks, in: Proc. Int. Conf. Parallel Processing, Vol. 12, No. 1, 2001, pp. 38–59.

[15] C.-L. Wu, T.-Y. Feng, On a class of multistage interconnection networks, IEEE Trans. Comput. 29 (8) (1980) 694–702.

[16] Y. Yang, J. Wang, Optimal all-to-all personalized exchange in self-routable multistage networks, IEEE Trans. Parallel Distrib. Syst. 11 (3) (2000) 261–274.

[17] Y. Yang, J. Wang, Optimal all-to-all personalized exchange in a class of optical multistage networks, IEEE Trans. Parallel Distrib. Syst. 12 (9) (2001) 567–582.

[18] K. Vipin, G. Ananth, G. Anshul, and K. George, Introduction to Parallel computing, The Benjamin/Cummings Publishing Company, Inc., 1994.

ภาคผนวก

ภาคผนวก ก

ผลงานวิจัยตีพิมพ์ที่ Journal of Parallel and Distributed Computing

Journal homepage : www.elsevier.com/locate/jpdc

R. Petagon and J. Werapun, Embedding the optimal all-to-all personalized exchange on multistage interconnection networks+, J. Parallel Distrib. Comput. 88 (2016), 15-30.

J. Parallel Distrib. Comput. 88 (2016) 16–30

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Embedding the optimal all-to-all personalized exchange on multistage interconnection networks+

Roselin Petagon*, Jeeraporn Werapun

Department of Computer Science, Faculty of Science, King Mongkut's Institute of Technology, Ladkrabang, Bangkok 10520, Thailand

HIGHLIGHTS

- Embedding the optimal all-to-all personalized exchange (ATAPE) on MINs+ is proposed for the fast on-chip process.
- Static ATAPE ($D = S$ XOR $(C + order)$ mod N) and f-in-1 dynamic ATAPE ($D = \rho[(S + C + order)$ mod $N]$) are ultimately embedded for full (S, D)-routing on MINs+.
- Existing ATAPE approaches rely on switch-/stage-control routing on MINs by either S- or D-routing.
- Correctness of the proposed ATAPE functions on d-nary-switch MINs+ and a crossbar of MIN+ are proven.
- Experimental results are confirmed fruitfully for $N = 64$ to 16384 processors with the significant speedup.

ARTICLE INFO

Article history:
Received 6 May 2015
Received in revised form
18 August 2015
Accepted 14 October 2015
Available online 23 October 2015

Keywords:
All-to-all personalized exchange (ATAPE)
Embedding f-in-1 dynamic ATAPE function incorporate with multistage pipelining
Fully self-routable multistage interconnection networks (MINs+)
A crossbar of MINs+ and the ultimate ATAPE embedding

ABSTRACT

All-to-all personalized exchange (ATAPE) is an inspired process to speedup the parallel and distributed computing. Recently, ATAPE algorithms were successfully applied on multistage interconnection networks (MINs), including baseline and butterfly networks. However, routing of those algorithms on MINs relies on switch-patterns for stage-control from sources (S), which is a half-routing solution since they cannot perform a full self-routing with the (S, D) protocol for all MINs. In this paper, first we propose a full-routing solution of the realizing ATAPE on a class of d-nary-switch MINs+ (i.e., baseline+, butterfly+, etc.). Our ATAPE can be embedded on-chip effectively for not only (S, D) self-routing but also stage-/switch-control routing. Two embedded ATAPE functions incorporate with multi-stage pipelining are proposed in optimal $O(N + \log_2 N)$: 1. a (default) static function $D = S$ XOR $(C + order)$ mod N and 2. an (optional) f-in-1 dynamic function $D = \rho[(S + C + order)$ mod $N]$ with the incrementing counter $C = 0$ to $N - 1$. Second, we introduce a crossbar of MINs+ with fewer delay-stages to achieve the ultimate ATAPE embedding. Finally, experimental results of applying ATAPE on such MINs+ are confirmed fruitfully, including the ATAPE-based NxN-matrix transposition in $O(N + \log_2 N)$, which yields the significant speedup.

1. Introduction

All-to-all personalized exchange (ATAPE) is one of the esteemed functions, which is extensively utilized on parallel and distributed systems [2,4,5,8–11,13,14]. Applications using the ATAPE efficiently are parallel fast Fourier transformation (FFT), parallel matrix transposition, etc. On parallel and distributed machines, processors usually communicate with each other when executing parallel applications and parallel communications can be one-to-one, one-to-many, one-to-all, and all-to-all connections [2]. One of the typical communications, required in most of the parallel applications, is all-to-all send/receive messages, which every processor sends a message to all other processors. According to processes of the message being sent, all-to-all communications can be classified as all-to-all broadcast (ATAB) and all-to-all personalized exchange (ATAPE) [13]. In the all-to-all broadcasting, all N processors send the same message to all other processors. In the ATAPE, all processors send their specific messages to other corresponding processors. In practice, many parallel applications require an efficient ATAPE communication to optimize their functions such as the matrix transposition, etc.

Fig. 1 shows an obvious example of the ATAPE communication to perform the parallel matrix transposition using N rounds of the parallel point-to-point communications from all N sources (S) to

* Corresponding author.
E-mail addresses: s4650501@kmitl.ac.th, memorose@hotmail.com (R. Petagon), ksjeerap@kmitl.ac.th (J. Werapun).

http://dx.doi.org/10.1016/j.jpdc.2015.10.005
0743-7315/

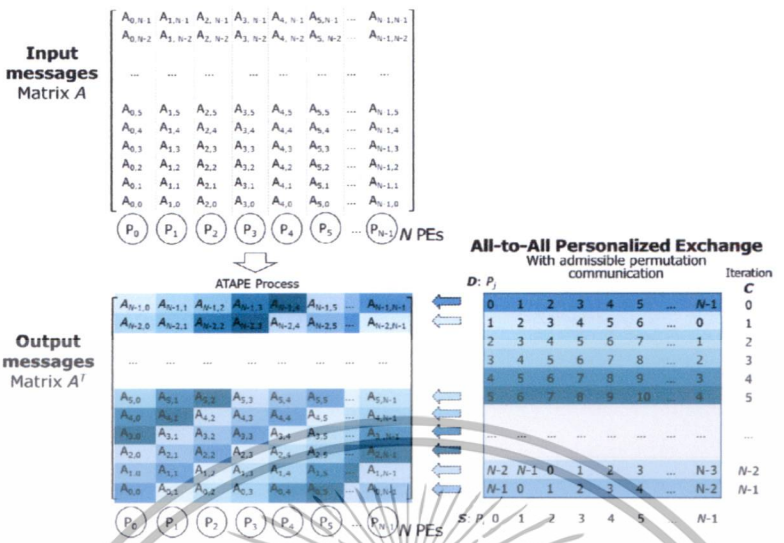


Fig. 1. All-to-all personalized exchange communication ($D = (S + C) \bmod N$) in $N \times N$ matrix transposition using N processors.

all N destinations (D), where $D = (S + C) \bmod N$. In the diagram, the output messages of the ATAPE processing represent the $N \times N$ matrix transposition or the Latin-square results of destinations. In particular, the parallel transposition of an $N \times N$ matrix by N processors is computed in N rounds ($C = 0$ to $N - 1$). Initially, N sections of the input matrix are partitioned and assigned one section per processor P_i ($i = 0, 1, 2, 3, \ldots, N - 1$). For instance, processor P_0 contains the elements of the matrix (in section 0) with indices $[0, 0], [0, 1], [0, 2], \ldots, [0, N - 1]$ and each of processors $P_1 - P_{N-1}$ contains another array of sections 1 to $(N-1)$, respectively. An output of the $N \times N$ matrix A^T is the transposition of an $N \times N$ matrix A, such that $A^T[i, j] = A[j, i]$ for $0 \le i, j < N$.

With that ATAPE communication (in Fig. 1), processors $P_0 - P_{N-1}$ can send/receive all elements in the matrix A (with N elements per round), based on the communication of $N - 1$ admissible permutations in row 1 to row $N - 1$ (since row 0 in the first round is the diagonal elements of A and A^T, which are already in $P_0 - P_{N-1}$). In the second round, the processors P_0 to P_{N-1} send/receive the corresponding N elements of the matrix A with indices $[S, D] = [0, 1], [1, 2], \ldots, [j, j + 1], \ldots, [N - 1, 0]$ from all processors P_S to P_D. After the final round, all elements of the matrix A^T are completely transferred. Clearly in this ATAPE application, every processor is able to send a distinct element of the matrix to every other processor in $O(N + \log_2 N)$ over $O(N \log_2 N)$ by other methods.

The ATAPE problem has been extensively studied in several network topologies [10,11,13]. Recently, the ATAPE applications have been focused on MINs (multistage interconnection networks) since they gave better scalability than static networks with $\log_2 N$ stages of dynamic communication. A class of MINs [1,3,6,7,12] includes omega, flip, cube, banyan, butterfly, and baseline networks.

In 2000, Yang and Wang [13] developed a Latin square method for decomposing ATAPE patterns into specific permutations, being realizable on MINs. That study used stage-control routing, a technique used to reduce the cost of network setting for ATAPE communication. Such an optimal ATAPE algorithm ($O(N + \log_2 N)$) provided shorter communication delay and better scalability than those ($O(N \log_2 N)$) of static networks (i.e., hypercubes, meshes).

In 2003, Massini [5] said that Yang's algorithm was dependent on the inter-stage connections of each network topology. Thus, Massini presented a new optimal algorithm (in $O(N + \log_2 N)$) using 2-way partition of each admissible permutation for switch-control routing that was independent on the network topology.

Later in 2007, Liu et al. [4] presented a new ATAPE for d-nary-switch MINs (N), where each MIN consisted of $\log_d N$ stages. That algorithm used $d \times d$ switch-boxes and a d-XOR operation (in each switch) with d-nary representation to generate d-nary switch-setting patterns. For 2×2-switch MINs (i.e., omega networks), the permutations of that ATAPE method by stage-control are similar to those of Yang and Wang [13].

Recently (in 2010), the algorithm of ATAPE for generalized shuffle exchange networks (GSENs) [2] was introduced to fulfill the networks with flexible processors N, where N ($\neq 2^n$) is an arbitrary even number. The shuffle exchange (or omega) network in that study was composed of $\lceil \log_2 N \rceil$ stages such that each stage contained $N/2$ switches, and the inter-stage connection (ISC) between stages was called the "perfect shuffle". In that stage & switch control-based algorithm, a single control bit (0 for straight (–) and 1 for cross (×)) could simply be used to control all switches in a corresponding stage.

However, those ATAPE methods use a crosstalk-free switch-pattern routing on MINs from sources (S) and move forward to destinations (D) by either stage- or switch-control routing. That technique is a half-routing solution, especially on the recursive MINs, since those ATAPE methods cannot be set by the (S, D) self-routing.

This paper proposes the full self-routing for the on-chip ATAPE embedding on completely self-routable MINs, called MINs$^+$. First, a class of MINs$^+$ are introduced, including omega (or shuffle exchange), flip, banyan$^+$ (or cube [7]), butterfly$^+$, baseline$^+$ (or R-network [6]), and inverse baseline$^+$ networks. Second, the function $D = S$ XOR $(C + order) \bmod N$ is proposed with three crosstalk-free routing methods: (1) stage-control routing, (2) switch-control routing, and (3) (S, D) self-routing, where the N successive controls are incremented by the on-chip counter $C (=0, 1, 2, \ldots, N - 1)$ in optimal time $O(N + \log_2 N)$. Specially, the (optional) f-in-1 dynamic function $D = \rho[(S + C + order) \bmod N]$ is also proposed for flexible setting one of fN ATAPE functions at run time, based on the requirement of user programs. Third, our ATAPE functions can be ultimately embedded on a variety of MINs$^+$ such as the d-nary-switch MINs$^+$ and a crossbar of MINs$^+$. In performance evaluation, experimental results of applying our ATAPE on MINs$^+$ have been recognized and the efficient ATAPE application is presented for

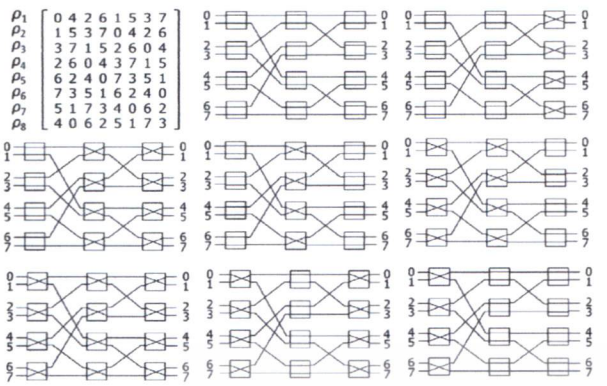


Fig. 2. Setting switches at each stage of the ATAPE [13] on baseline network with $N = 8$.

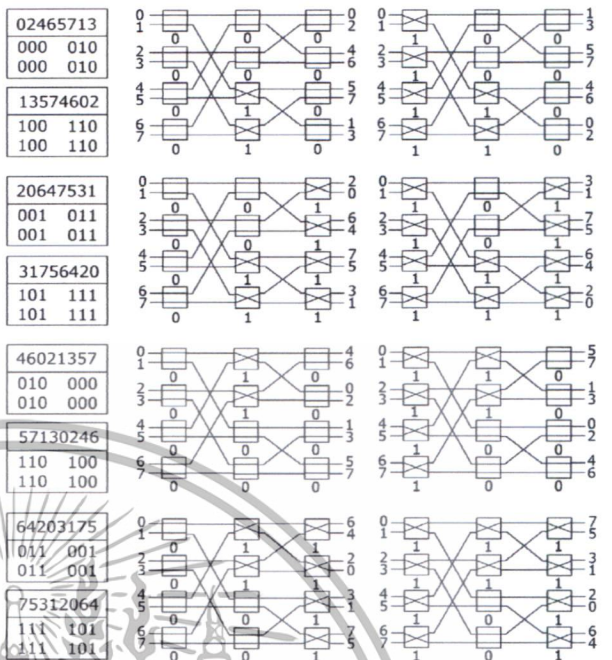


Fig. 3. 2-way partition of admissible permutations of the ATAPE using P^{18} and S^{18} [5] on butterfly network ($N = 8$, $l = 18$).

computing the parallel $N \times N$-matrix transposition on the MINs$^+$ in $O(N + \log_2 N)$, which yields the significant speedup.

The remaining part of this paper is organized as follows: Section 2 reviews some related works about existing ATAPE algorithms on MINs. Section 3 presents an on-chip ATAPE multistage-pipelining along with the optimal complexity to fulfill the process of all-to-all personalized exchange on the d-nary MINs$^+$ and a crossbar of MINs$^+$. Section 4 shows the applications and experimental results of our on-chip ATAPE embedding on the MINs$^+$, a crossbar of MINs$^+$, and the ATAPE communication of $N \times N$-matrix transposition. Finally, the last section contains the conclusion of our study.

2. Related work

In this section, existing ATAPE algorithms (see Table 1) for a class of MINs [2,4,5,13] are illustrated, including their advantages and limitations. In 2000, Yang and Wang [13] proposed an optimal ATAPE algorithm on scalable MINs, focusing on shuffle exchange (or omega), banyan, and baseline networks. That ATAPE algorithm constructed the special Latin Square for all possible permutations of the control-stage switch-setting for such MINs, especially the recursive MINs (i.e., the baseline, the banyan, etc.). With that ATAPE, each of those MINs needed a different Latin-square of N permutations, where each row (or permutation ρ) in the Latin Square is used to set switches at each stage to the destinations (D) by the D-control routing. Fig. 2 shows all N permutations ($\rho_1 - \rho_8$) for setting ATAPE communication for a baseline network ($N = 8$), where $\rho_1 = (0\ 4\ 2\ 6\ 1\ 5\ 3\ 7)$ representing connections $P_0 \rightarrow P_0, P_1 \rightarrow P_4, P_2 \rightarrow P_2, P_3 \rightarrow P_6, P_4 \rightarrow P_1, P_5 \rightarrow P_5, P_6 \rightarrow P_3, P_7 \rightarrow P_7$, but for a banyan network ($N = 8$) $\rho_1 = (0\ 2\ 4\ 6\ 1\ 3\ 5\ 7)$ or $P_0 \rightarrow P_0, P_1 \rightarrow P_2, P_2 \rightarrow P_4, P_3 \rightarrow P_6, P_4 \rightarrow P_1, P_5 \rightarrow P_3, P_6 \rightarrow P_5, P_7 \rightarrow P_7$, which can connect successfully by the D-routing. However on those recursive MINs, the (S, D) self-routing from each permutation cannot be set in each switch (across $n = \log_2 N$ stages) for certain sources (S) and destinations (D).

Later in 2003, Massini [5] introduced another ATAPE algorithm to support all-to-all personalized exchange. That algorithm can construct a Latin square by 2-way partitioning of admissible permutations (P) for $\log_2 N$ stages. The partitioning provides the $2^{(N/2)\log N}/N = N^{(N/2)-1}$ sets of P^l, where $l = 0, 1, 2, 3, \ldots, N^{(N/2)-1} - 1$, and the set of all possible binary values (S) of each permutation to represent the network configuration with switch setting status (i.e., 0: straight (-), 1: cross ($\times$) for 2×2 switch-boxes) for the switch-control routing. For example, Fig. 3 shows the 2-way partitioning of each admissible permutation on the butterfly network with $N = 8$, using $l = 18$ for P^l and S^l. Like the previous work [13], some (S, D) self-routing connections cannot be set on such MINs.

In 2007, Liu et al. [4] proposed an ATAPE algorithm on d-nary-switch MINs (N) which consisted of $\log_d N$ stages for switch-boxes of size $d \times d$. Let x, y be input-ports and output-ports of each d-nary switch, where $x, y = (0, 1, 2, 3, \ldots, d-1)$. In the d-nary switch setting, the set of utilized d patterns (P) was $P = \{0\text{-shift}, 1\text{-shift}, 2\text{-shift}, 3\text{-shift}, 4\text{-shift}, \ldots, (d-1)\text{-shift}\}$. Those switch patterns are generated by the d-XOR operation in the local switches, where $y = x \oplus_d p = (x + p) \bmod d$ is computed for each $(N/d) \times \log_d N$ switches (see Fig. 4(a) for $d = 4$, $p = 0, 1, 2, 3$). Fig. 4(b) shows 16 permutations (for $N = 16$) with ($\log_4 16$)-digit 4-nary representation. For example, there is a unique path from the source processor S to the destination processor D such as in round 3 ($=03_4$), the permutation [3 0 1 2 7 4 5 6 11 8 9 10 15 12 13 14] of the 16×16 Latin square is set as follows: $P_0 \rightarrow P_3, P_1 \rightarrow P_0, P_2 \rightarrow P_1, P_3 \rightarrow P_2, P_4 \rightarrow P_7, P_5 \rightarrow P_4, \ldots$, and $P_{15} \rightarrow P_{14}$.

Recently in 2010, Chou and Chen [2] proposed an ATAPE for GSENs (the generalized shuffle exchange networks) of size $N \neq 2^n$. That approach used an alternating stage control technique ($A = a_{n-1}2^{n-1} + \cdots + a_2 2^2 + a_1 2^1 + a_0 2^0$) with $0 \leq A < 2^n$. For switch setting, the alternating between straight (-) or 0 and cross ($\times$) or 1 was assigned (by a control bit a_ℓ), as follows: For $N \equiv 2 \pmod 4$, let a_ℓ denote the control to set switches at stage $n - 1$-ℓ such that

$a_\ell = 0$ mean the setting 0, 1, 0, 1, 0, 1, and so on.
$a_\ell = 1$ mean the setting 1, 0, 1, 0, 1, 0, and so on.

Thus N alternate configurations ($A_0, A_1, A_2, \ldots, A_{N-1}$), computed by $A_k = k \oplus \lfloor k/2 \rfloor$, $k = 0, 1, 2, 3, \ldots, N - 1$. For example, in the GSENs with $N = 10$, there are 10 alternate configurations ($A_0 = 0, A_1 = 1, A_2 = 3, A_3 = 2, A_4 = 6, A_5 = 7, A_6 = 9, A_7 = 10, A_8 = 12, A_9 = 13$), see Fig. 5 with 10 network configurations.

Table 1 presents the comparison between the key features of the ATAPE communication of existing methods (i.e., flexible self-routing, Latin-Square (LS) results, etc.). We summarize each of those ATAPE solutions [2,4,5,13], as follows: For 2×2 switches, Yang and Wang [13] presented the original ATAPE on MINs but

Table 1
The comparison between routing features of the existing ATAPE methods and our proposed ATAPE embedding function.

Comparison of ATAPE routing features	Static networks	Dynamic networks					
	Hypercube dragonfly	MINs			GSENs	$MINs^+$	Crossbar of $MINs^+$
		$O(N + \log N)$					
	1991 [9]	2000 [13]	2003 [5]	2007 [4]	2010 [2]	Our ATAPE	
	2013 [8]					$D = S \oplus (C + order)_{\text{mod } N}$	
						$D = \rho[(S + C + order)_{\text{mod } N}]$	
Static self-routing solution by functions	✓						
Dynamic self-routing solution		Half	Half	Half	Half	Full	Full
o Switch-control routing			✓		✓	✓	✓
o Stage-control routing		✓		✓	✓	✓	✓
o D-control routing		✓	✓	✓		✓	✓
o (S, D) self-routing						✓	✓
Latin-square space for destinations							
o Need a specific LS for each MIN		✓	✓	✓	✓		
o Can use a standard LS for all $MINs^+$						✓	✓
2×2 switches		✓	✓	✓	✓	✓	✓
d-nary switches				✓		✓	✓
Network sizes $N \neq 2^n$					✓		

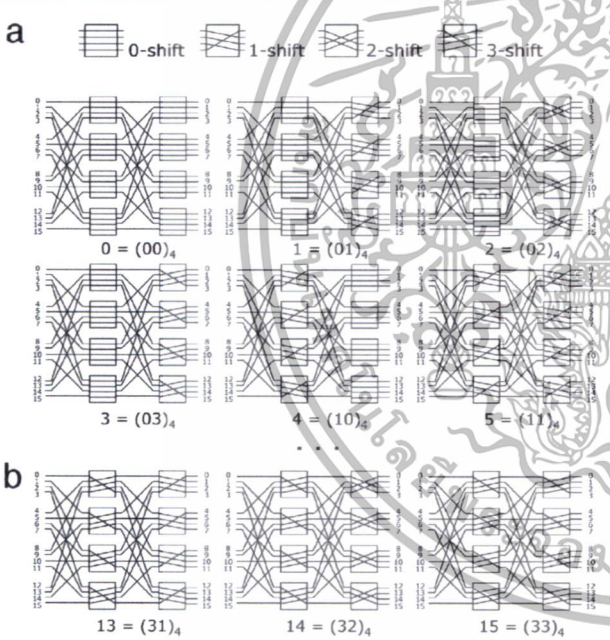


Fig. 4. (a) Four patterns of 4×4 switch setting for stage control and (b) 16 permutations of ATAPE [4] on a self-routable MIN ($N = 16$).

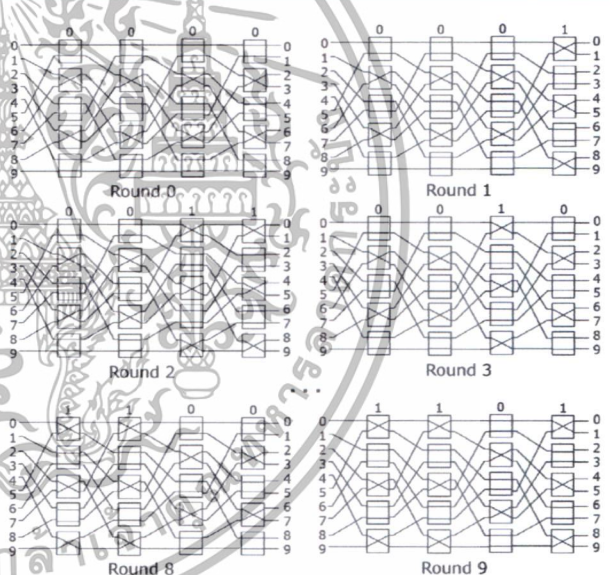


Fig. 5. An example of ATAPE on GSENs [2] with $N = 10$.

their pre-processing LSs rely on the certain MINs. Massini [5] introduced the independent LSs over MINs using 2×2 switches but the process required the switch-patterns for the ATAPE communication. For $d \times d$ switches, Liu et al. [4] proposed the switch-based ATAPE for the self-routable MINs, but that method needed a special function, included in each switch (for all $N/d \times \log_d N$ switches). Thus, for the scalable networks, the implementation of the MIN may be too expensive. Lastly, for $N \neq 2^n$ those existing methods could not be applied directly. Thus, Chou and Chen [2] introduced specific functions for shuffle–exchange networks (GSENs) but $N \neq 2^n$ is not scalable for large N.

Next in Section 3, we propose the new ATAPE embedding with the fast on-chip process, where the network solution and the software solution are resolved with the XOR-based ATAPE for all self-routable $MINs^+$. Note: The XOR-function $D = S \oplus C$ for ATAPE on static networks (i.e., hypercube, mesh, etc.) was introduced in 1991 by Scott [9]. However this function cannot be applied on dynamic MINs directly, especially on recursive MINs (i.e., baseline, butterfly, and banyan networks). Recently, hierarchical ATAPE (with the function $h^D = (h^S + h^P) \bmod N^l$ in level l) was introduced on the dragonfly static networks by Prisacari et al. [8] but that function could not be operated on the existing MINs. Obviously, the ATAPE algorithms on the MINs [2,4,5,13] focused on routing by either the S-control with switch patterns (the stage-/switch-control routing) or by the D-control with the preprocessing LS of admissible permutations for each MIN. Those ATAPE methods require the patterns of N admissible permutations on certain MINs. For the existing MINs, such control techniques are half-routing solutions since those ATAPE approaches cannot set fully by the (S, D) routing through switches and stages on the recursive MINs.

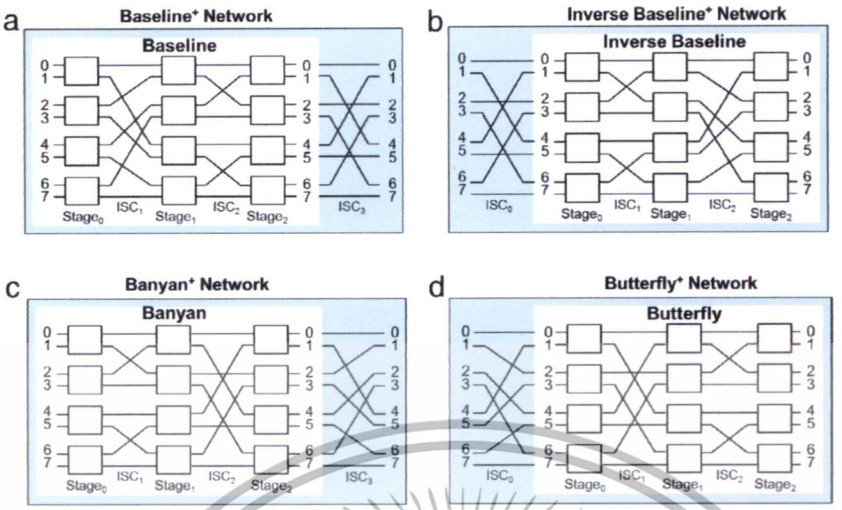


Fig. 6. MINs$^+$ ($N = 8$): (a) baseline$^+$ network, (b) inverse baseline$^+$ network, (c) banyan$^+$ network, and (d) butterfly$^+$ network.

3. On-chip ATAPE embedding on d-nary MINs$^+$

We propose a full-routing solution for the on-chip ATAPE embedding on d-nary-switch MINs$^+$, including omega, flip, banyan$^+$, butterfly$^+$, baseline$^+$, and inverse baseline$^+$ networks. In our innovation and contribution, we introduce the fully self-routable networks, called MINs$^+$ first (see Section 3.1). Then, two efficient ATAPE functions are proposed for on-chip embedding on such MINs$^+$ and allow user to adjust at runtime: 1. the (default) static XOR-function $D = S \oplus (C + order) \bmod N$ (for N Latin squares of ATAPE) and 2. the (optional) f-in-1 dynamic function $D = \rho[(S + C + order) \bmod N]$ for fN Latin squares of flexible ATAPE destinations, while the existing functions [8,9] provide only one Latin square per function. For the f-in-1 dynamic function, we embed only one abstract function (the array of pointers to the destinations) for optional setting one of fN Latin squares by the user program (see Section 3.2). Both of our ATAPE functions can be embedded on-chip efficiently (i.e., fast processing, easy implementation, and inexpensive cost). Using our ATAPE approach, not only the correct (S, D) routing but also the stage-/switch-control routing are available for all MINs$^+$ with d-nary switches. Moreover, the fast crossbar of MINs$^+$ and the powerful ATAPE embedding are presented in Section 3.3.

3.1. The fully self-routable MINs$^+$

In the past, each recursive MIN (i.e., baseline, banyan, butterfly, etc.) composes of the complex ISCs (inter-stage connections), and hence it is difficult to embed the bitwise function $D = S \oplus C$ since the results of this function are not admissible permutations on the recursive MINs. Therefore, to make this be possible, we add a special ISC for port-adjusting on each of the recursive MIN types, called MINs$^+$. For example (see Fig. 6), we include ISC_n (the reverse n-bit address) on the baseline$^+$ network, ISC_0 (the reverse n-bit address) on the inverse baseline$^+$ network, ISC_n (the circular right-shift) on the banyan$^+$ network, and ISC_0 (the circular left-shift) on the butterfly$^+$ network in the construction of the fully routable MINs$^+$.

Theoretically, the correct self-routing of the baseline$^+$ network (from S or input $I = (i_{n-1}i_{n-2} \ldots i_2i_1i_0)$ to output $J = (j_{n-1}j_{n-2} \cdots j_2j_1j_0)$ or D) is derived as follows:

$\text{sw-ISC}_1 : (i_{n-1}i_{n-2} .. i_4i_3i_2i_1\mathbf{i_0}) \rightarrow (\mathbf{j_0}i_{n-1}i_{n-2} .. i_3i_2i_1)$

$\text{sw-ISC}_2 : j_0(i_{n-1}i_{n-2} .. i_4i_3i_2\mathbf{i_1}) \rightarrow j_0(\mathbf{j_1}i_{n-1}i_{n-2} .. i_3i_2)$

. . .

$\text{sw-ISC}_{n-1} : j_0j_1j_2j_3 .. j_{n-3}(i_{n-1}\mathbf{i_{n-2}}) \rightarrow j_0j_1j_2j_3 .. j_{n-3}(\mathbf{j_{n-2}}i_{n-1})$

$[\text{sw-ISC}_n : (j_0j_1j_2j_3 .. j_{n-3}i_{n-2}\mathbf{i_{n-1}}) \rightarrow (\mathbf{j_{n-1}}j_{n-2} .. j_3j_2j_1j_0)]$.

For the banyan$^+$ network, the successful self-routing is

$\text{sw-ISC}_1 : i_{n-1}i_{n-2} .. i_4i_3i_2(i_1\mathbf{i_0}) \rightarrow i_{n-1}i_{n-2} .. i_3i_2(\mathbf{j_0}i_1)$

$\text{sw-ISC}_2 : i_{n-1}i_{n-2} .. i_4i_3(i_2\mathbf{j_0}i_1) \rightarrow i_{n-1}i_{n-2} .. i_3(\mathbf{j_1}j_0i_2)$

. . .

$\text{sw-ISC}_{n-1} : (i_{n-1}\mathbf{j_{n-3}} .. j_2j_1j_0\mathbf{i_{n-2}}) \rightarrow (\mathbf{j_{n-2}}j_{n-3} .. j_2j_1j_0i_{n-1})$

$[\text{sw-ISC}_n : j_{n-2}\mathbf{j_{n-3}} .. j_3j_2j_1j_0\mathbf{i_{n-1}} \rightarrow \mathbf{j_{n-1}}j_{n-2}j_{n-3} .. j_2j_1j_0]$.

For the butterfly$^+$ network, the complete self-routing is

$[\text{ISC}_0\text{-sw}\,\mathbf{i_{n-1}}i_{n-2}i_{n-3} .. i_3i_2i_1\mathbf{i_0} \rightarrow i_{n-2}i_{n-3}i_{n-4} .. i_2i_1i_0\mathbf{j_{n-1}}]$

$\text{ISC}_1\text{-sw} : (\mathbf{i_{n-2}}i_{n-3} .. i_2i_1i_0\mathbf{j_{n-1}}) \rightarrow (\mathbf{j_{n-1}}i_{n-3}i_{n-4} .. i_2i_1i_0\mathbf{j_{n-2}})$

$\text{ISC}_2\text{-sw} : j_{n-1}(\mathbf{i_{n-3}}i_{n-4} .. i_1i_0j_{n-2}) \rightarrow j_{n-1}(j_{n-2}i_{n-4} .. i_2i_1i_0\mathbf{j_{n-3}})$

. . .

$\text{ISC}_{n-2}\text{-sw} : j_{n-1}j_{n-2} .. j_4\mathbf{j_3}(\mathbf{i_1}i_0j_2) \rightarrow j_{n-1}j_{n-2} .. j_4j_3(j_2i_0\mathbf{j_1})$

$\text{ISC}_{n-1}\text{-sw} : j_{n-1}j_{n-2} .. j_4j_3j_2(\mathbf{i_0}j_1) \rightarrow j_{n-1}j_{n-2} .. j_4j_3j_2(j_1\mathbf{j_0})$.

In our programming test, the correct routing of these MINs$^+$ are also verified by varying N up to 2^{20} processors.

Note: For d-nary switches ($d = 2^k$), the k-bit ISC patterns are utilized on our fully self-routable MINs$^+$, see Figs. 10–11.

3.2. The on-chip ATAPE embedding on MINs$^+$

This paper proposes an on-chip ATAPE embedding algorithm on the d-nary-switch MINs$^+$ for the fast clock-speed processing. For both ATAPE functions (Algorithm 3.1), different destinations (D) can receive personalized messages from the particular processors (S) in round r ($= 0$ to $N - 1$) by using the on-chip counter for bitwise control $C = r$. The incrementing counter can be operated in the register, which is much faster than fetching instruction from the memory. In applying ATAPE on the MINs$^+$, the (S, D) connections by incrementing the counter C can be utilized efficiently for N successive rounds. In addition, our ATAPE embedding can be used to fulfill ATAPE communication on MINs$^+$ efficiently with multistage pipelining in ideal optimal $O(N + \log_2 N)$, proven in Theorem 3.1.

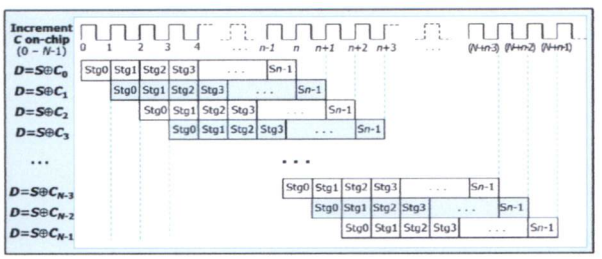


Fig. 7. Timing diagram of ATAPE multistage pipelining by incrementing the on-chip counter control ($C_0, C_1, \ldots, C_{N-1}$).

Algorithm 3.1 On-chip All-to-All Personalized Exchange (ATAPE) embedding on d-nary-switch MINs$^+$.

1. **Incrementing the on-chip counter control** C (=0 to N-1) (incorporate with multistage pipelining, see Fig.7.)
2. **In parallel,** ***all*** **processors** S ($0 \le S < N$) ***do***
3. $D = S \oplus (C+order) \bmod N$ (default)
   or $D = \rho[(S+C+order) \bmod N]$ (option)
4. all processors S prepare and send their personalized massages to processors D (k-bit control tag per switch per stage.)
5. ***end*** **parallel processors**
6. ***end*** **the on-chip counter** C

Note: For routing the message, the control tag F (or B) $= S \oplus (C + order) \bmod N$ may be used as a header with the k-bit control, where $F = (f_{n-1}f_{n-2} \ldots f_1f_0)_2$ or $B = (b_{n-1}\, b_{n-2} \ldots b_1b_0)_2$ represents the destination processor D. Then, for the next stage, k bits of the remaining bits (in F or B) is discarded by each successive switch to use the first k-bit element in the control tag across $m = \log_k N$ stages, such as forward MS k-bit for omega and butterfly$^+$ networks and backward LS k-bit for other MINs$^+$ (i.e., flip, cube, banyan$^+$, and baseline$^+$ networks).

Theorem 3.1. *Time complexity of our on-chip ATAPE embedding algorithm on MINs$_+$ is* $O(N + \log_2 N)$, *which is optimal.*

Proof. Each of the N processors can receive the first personalized message in the first round, specified by the counter control $C = 0$, passing through $n = \log_2 N$ stages in n time-units (assume one clock per time-unit for on-chip processing) and receive the remaining $N-1$ personalized messages in $N-1$ clocks (one clock per round), according to successive multistage pipelining, as demonstrated in Fig. 7. The idea of executing successive N rounds $(C_0, C_1, C_2, \ldots, C_{N-1})$ on the multistage pipelining is that the next round (C_{i+1}) can start after the previous round (C_i) finished stage 0 and so on for other successive stages. This is the fully multistage pipelining because it is a register-based process (by incrementing the counter C), which is effectively faster than the fetching N instructions (or rounds) from the memory of the existing ATAPE methods, where one memory-access time (c) is usually greater than one clock. Thus, time complexity of our ATAPE is ideal optimal $O(N + \log_2 N)$ since it can finish in $N + n - 1$ clocks (i.e., n clocks for the first round through n stages plus $N-1$ clocks for the remaining $N-1$ rounds) whereas the existing ATAPE methods require $cN + n - 1$ clocks. □

3.2.1. Latin square of admissible permutations

Usually one ATAPE function can generate only one Latin-square (LS) of admissible permutations, which is not flexible. In order to provide many ATAPE Latin-squares for flexible (S, D) routing paths w/o LS preprocessing, we propose two ATAPE embedding functions: 1. the static $D = S \oplus (C + order) \bmod N$ that can start with an arbitrary order and 2. the dynamic $D = \rho[(S + C + order) \bmod N]$ for flexible changing by the user programs during runtime. In our ATAPE embedding, all (S, D) connections can select an arbitrary order before sending the messages automatically. Fig. 8(a) shows the destination results ($D = S \oplus (C + order) \bmod N$) for $N = 8$ with 8 shuffle orders for N Latin squares. Each of Latin squares can be applied for all types of MINs$^+$ (i.e., the omega, flip, cube or banyan$^+$, butterfly$^+$, baseline$^+$, inverse baseline$^+$ networks, etc.), see examples in Fig. 14.

For flexible ATAPE communication, the dynamic ATAPE function $D = \rho[(S + C + order) \bmod N]$ is introduced for allowing the user programs be able to set the distinct initial permutation for sending personalized messages. This f-in-1 abstract function provides f initial permutations ($\rho[i]$, $i = 0, 1, \ldots, N-1$) of fN Latin squares. Fig. 8(b) displays the Latin-square results of setting two initial permutations from two static functions through the dynamic $D = \rho[(S + C + order) \bmod N]$: 1. $\rho[0..N-1] = (0, 1, 2, \ldots, N-1)$ of $D = (S + C + order) \bmod N$ and 2. $\rho[0..N-1] = (N-1, N-2, \ldots, 2, 1, 0)$ of $D = (2N - 1 - (S + C) + order) \bmod N$. This flexible ATAPE function can utilize all $d!$ switch configurations, while the XOR-based ATAPE uses only d patterns.

3.2.2. Admissible permutations of our ATAPE for N network configurations on d-nary MINs$^+$

By utilizing the default function $D = S \oplus (C + order) \bmod N$, the variation of self-routing of our ATAPE permits (1) the stage-control routing, (2) the switch-control routing, and (3) the full (S, D) self-routing. Like the existing ATAPE [13], for the stage control of each switch on MINs$^+$, the process is defined as the network configuration setting for single-pass permutations (with no switch conflict). The (n-bit) stage control will be operated in binary $(c_{n-1}\, c_{n-2}\, c_{n-3} \ldots c_1c_0)_2$, which can be simply referred by a decimal number C, where $0 \le C < 2^n$. For the single-pass permutations, there is one-to-one permutation mapping $\rho = (a_0\, a_1 \ldots a_i \ldots a_{N-1})$ between sources (S) and destinations $\rho(S)$ of MIN$^+$, where $\rho(S) = D \in \{0, 1, \ldots, N-1\}$ for $S = 0, 1, \ldots, N-1$.

For the (S, D) self-routing, let a connection (S, D) be a fixed path for sending a message from processor S to processor D. On the MINs$^+$, such a connected path across n stages can be set by a sequence of n control bits (corresponding to S and D) for successive switches along that path. Let $\oplus$ denote the bitwise-XOR operation ($0 \oplus 0 = 0, 0 \oplus 1 = 1, 1 \oplus 0 = 1, 1 \oplus 1 = 0$). For $S = (s_{n-1}s_{n-2} \ldots s_2s_1s_0)_2$ and $C = (c_{n-1}c_{n-2} \ldots c_2c_1c_0)_2$, then $D = (d_{n-1}d_{n-2} \ldots d_2d_1d_0)_2 = S \oplus C = (s_{n-1} \oplus c_{n-1}\, s_{n-2} \oplus c_{n-2} \ldots s_1 \oplus c_1\, s_0 \oplus c_0)_2$. In our ATAPE approach, both of the stage-control routing and the (S, D) switch-control routing can be processed by N configuration networks. Thus, the configurations of 2×2 switches, set by N stage controls (C), are reconfigured within two of $2! = 2$ patterns (cross (×) 0-1 or 1-0 and straight (-) 0-0 or 1-1) without conflict. On the other hand, our ATAPE routing can use one bit (partitioned from n-bit S and $D = S \oplus (C + order) \bmod N$) for switch setting at each stage to connect from N sources to N destinations. Note that we use the bitwise-XOR operation of the control C for setting two different patterns on each switch with a crosstalk-free (since $d_i = s_i \oplus 0 = s_i$ (for straight (-)) and $d_i = s_i \oplus 1 = s_i'$ for cross (×), see Fig. 9(a)) and use the counter control C by incrementing from 0 to $N-1$ for the N distinct configurations on MINs$^+$.

When the stage control technique is employed, let $R(N)$ denote the number of network configurations required to recognize all-to-all communications in the MINs$^+$, as proven in Theorems 3.2–3.3. A round r ($0 \le r < N$) denotes the round of transmitting all N personalized messages from N sources to N destinations by using a bitwise-control $C = (c_{n-1}c_{n-2} \ldots c_1c_0)_2$ for switch setting at each stage, where all N different binary-control tags are $000\ldots00_2, 000\ldots01_2, 000\ldots10_2, \ldots, 111\ldots10_2, 111\ldots11_2$ that can be incremented by the on-chip counter.

Let the standard Latin-square permutations be defined by the most effective function $D = S \oplus C$, where N controls for getting N permutations are incremented by the counter $C = 0\,(000_2), 1\,(001_2), 2\,(010_2), \ldots, N-1\,(111_2)$ for $N = 8$, as shown in Fig. 8(a) with $order = 0$. Usually the embedded function is not

order = 0

0	1	2	3	4	5	6	7
1	0	3	2	5	4	7	6
2	3	0	1	6	7	4	5
3	2	1	0	7	6	5	4
4	5	6	7	0	1	2	3
5	4	7	6	1	0	3	2
6	7	4	5	2	3	0	1
7	6	5	4	3	2	1	0

order = 1

1	0	3	2	5	4	7	6
2	3	0	1	6	7	4	5
3	2	1	0	7	6	5	4
4	5	6	7	0	1	2	3
5	4	7	6	1	0	3	2
6	7	4	5	2	3	0	1
7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7

order = 2

2	3	0	1	6	7	4	5
3	2	1	0	7	6	5	4
4	5	6	7	0	1	2	3
5	4	7	6	1	0	3	2
6	7	4	5	2	3	0	1
7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7
1	0	3	2	5	4	7	6

order = 3

3	2	1	0	7	6	5	4
4	5	6	7	0	1	2	3
5	4	7	6	1	0	3	2
6	7	4	5	2	3	0	1
7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7
1	0	3	2	5	4	7	6
2	3	0	1	6	7	4	5

order = 4

4	5	6	7	0	1	2	3
5	4	7	6	1	0	3	2
6	7	4	5	2	3	0	1
7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7
1	0	3	2	5	4	7	6
2	3	0	1	6	7	4	5
3	2	1	0	7	6	5	4

order = 5

5	4	7	6	1	0	3	2
6	7	4	5	2	3	0	1
7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7
1	0	3	2	5	4	7	6
2	3	0	1	6	7	4	5
3	2	1	0	7	6	5	4
4	5	6	7	0	1	2	3

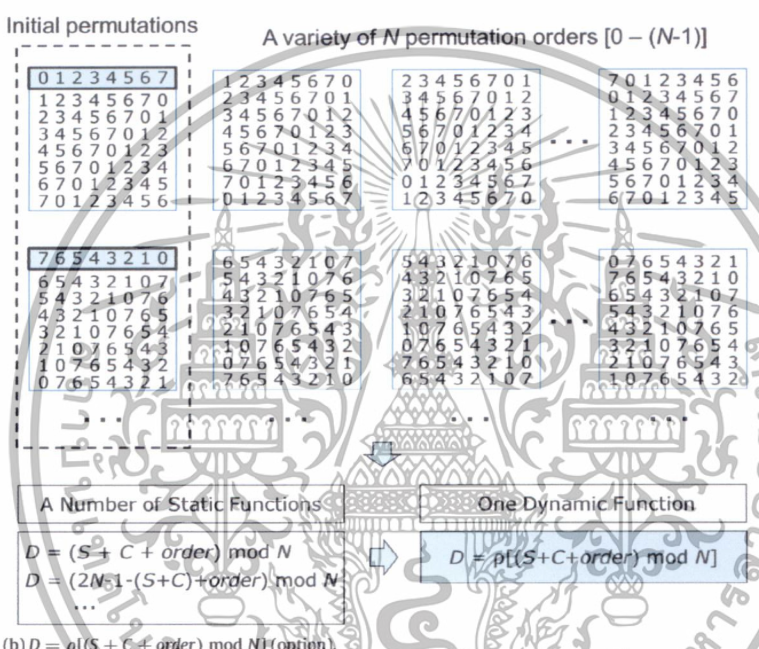
order = 6

6	7	4	5	2	3	0	1
7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7
1	0	3	2	5	4	7	6
2	3	0	1	6	7	4	5
3	2	1	0	7	6	5	4
4	5	6	7	0	1	2	3
5	4	7	6	1	0	3	2

order = 7

7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7
1	0	3	2	5	4	7	6
2	3	0	1	6	7	4	5
3	2	1	0	7	6	5	4
4	5	6	7	0	1	2	3
5	4	7	6	1	0	3	2
6	7	4	5	2	3	0	1

(a) $D = S \oplus (C + order) \bmod N$ (default).



(b) $D = \rho[(S + C + order) \bmod N]$ (option).

Fig. 8. Latin-square results (permutations) of applying our ATAPE embedding for the (S, D) self-routing ($N = 8$) with orders 0 to $N - 1$.

flexible and could not be modified. Thus, in this paper the function $D = S \oplus (C + order) \bmod N$ is introduced to provide flexible orders of permutations in the Latin square (Fig. 8(a) *order* 0–7) that may require by the parallel program during runtime. The particular order is varying between 0 and $N-1$ so that the order 0 means $D = S \oplus C$ (for constructing the standard Latin-square of permutations); the order 1 means to shuffle the order such that the first row of a new Latin-square is the second row of the standard Latin-square and so on. For example, the Latin square with order = 2 starts with the first (S, D) connections by the permutation [2 3 0 1 6 7 4 5] are $(0, 2), (1, 3), (2, 0), (3, 1), (4, 6), (5, 7), (6, 4)$ and $(7, 5)$ automatically. Although we prepare N Latin squares with flexible N orders (each uses the same N successive controls $C = 0, 1, 2, \ldots, N - 1$), one Latin square (LS) is used at a time for N network configurations without LS-preprocessing.

Theorem 3.2. *In each d-nary MIN$^+$, the number of network configurations with the variation of controls is $R(N) = N = 2^n$.*

Proof. Let the network configuration be set by a control $C = (c_{n-1}c_{n-2} \ldots c_\ell c_{\ell+1} \ldots c_1 c_0)_2$. The N unique routing paths start from all source processors ($S = 0, 1, 2, 3, \ldots, N - 1$) to destinations ($D$) specified by a permutation $\rho = (a_0\ a_1\ \ldots\ a_l\ \ldots\ a_{N-1})$. As a result of the bitwise XOR-function $D = S \oplus C$, each network configuration (for all connections (S, D) according to a fixed control C ($0 \le C < 2^n$)) can send only N personalized messages to N corresponding processors. For generating a standard Latin-square of destinations (D) with N single-pass permutations for self-routing, we use n-bit counter $C = (c_{n-1}c_{n-2} \ldots c_1c_0)_2$ for N different permutations of controls (i.e., 0000…00, 0000 …01, 0000 … 10, 0000 … 11, …, 1111 … 10, and 1111 … 11). For example, see Fig. 9(a)–(b) for N network configurations, when $N = 16$, $d = 2$ (applying a 1-bit partition per switch (of size 2×2) per stage) and $N = 64$, $d = 4$ (using a 2-bit partition per switch (of size 4×4) per stage). For each network configuration, one control (or round) $r = C$ for N source processors (S) is applied at a time to send N personalized messages (in parallel) to each of particular processors (D) with all-to-all communications. For all sources ($S = 0, 1, 2, 3, \ldots, N - 1$) and a control C, the first permutation of the Lain-square is $D = S \oplus$ 0000…00; the next permutation is $D = S \oplus$ 0000 … 01; and so on for other different permutations ($D = S \oplus C$). Clearly, the results of a Latin-square (in Fig. 8) represent N network configurations of all (S, D) connections for the ATAPE process. Finally for each network configuration, N different admissible permutations can be applied successfully on all of the MINs$^+$ (i.e., omega, flip, banyan$^+$, butterfly$^+$, baseline$^+$, etc.). For other orders (1 to $N - 1$),

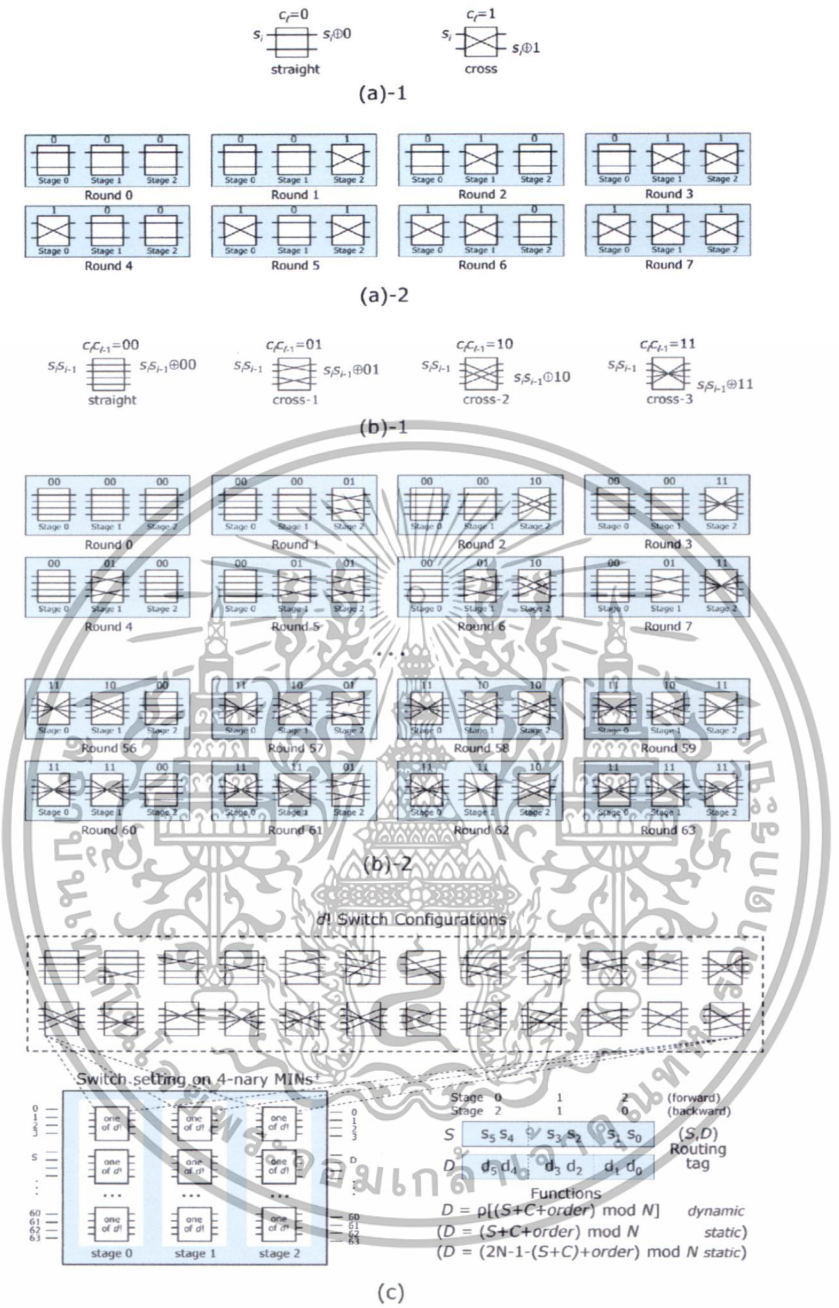


Fig. 9. N network configurations: (a) with $c_\ell = 0$ or 1 (2 patterns of 2×2 switches) for $N = 8$, (b) with $c_\ell c_{\ell-1} = 00$ to 11 (4 patterns of 4×4 switches) for $N = 64$, and (c) all possible $d!$ switch patterns ($d = 4$).

see Fig. 8(a), the function $D = S \oplus (C + order) \bmod N$ will generate other Latin-squares with a specific order. In addition, applying the f-in-1 dynamic function $D = \rho[(S + C + order) \bmod N]$ provides other flexible fN Latin squares (Fig. 8(b)) to allow use programs set the initial permutation of their ATAPE functions. Fig. 9(c) illustrates flexible network configurations, set by the dynamic ATAPE for the (S, D) routing with $d!$ switch patterns. A permutation in each round of this ATAPE is one of $(d!)^{N/d \times \log N}$ admissible permutations on each MIN$^+$ that supports parallel point-to-point communication without switch conflict. Obviously, the k-bit S–D set in each switch is a 1-1 mapping since all distinct Ds are computed with the same C and $order$ for all $0 \le Ss < N$. □

Theorem 3.3. *For d-nary MINs$^+$ ($N = 2^n$, $d = 2^k$, and $m = \log_d N$), the network configuration is set with incrementing the on-chip counter $C = (c_{n-1}c_{n-2} \ldots c_1c_0)_2$ and sources S (=0, 1, 2, …, $N-1$) by the k-bit control setting per switch at each stage to connect to corresponding destinations D by functions (3.1) or (3.2).*

$$F \text{ (or } B) = S \oplus (C + order) \bmod N \text{ (default)} \tag{3.1}$$

$$\text{or } F \text{ (or } B) = \rho[(S + C + order) \bmod N] \text{ (option)} \tag{3.2}$$

where F (forward control tag), B (backward control tag),

C (on-chip control) $= 0, 1, 2, \ldots, 2^n - 1$ and $0 \le order < 2^n$.

Proof. The variation of self-routing MINs$^+$ are based on two different rules: (1) the most significant (MS) bit first applies the forward control tag on the forward routing networks (i.e., the omega, the butterfly$^+$, etc.); and (2) the least significant (LS) bit first uses the backward control tag on the backward routing networks (i.e., the flip, the banyan$^+$, the baseline$^+$, etc.). Hence, for the forward routing networks (see Fig. 10(a) for 4-nary switches), stage 0 uses two MS bits of $C(c_{n-1}c_{n-2})$ and stage $m-1$ uses two LS bits of C (c_1c_0) from the most significant to the least significant positions, where $m = \log_d N$. On the other hand, for the backward routing networks (see Fig. 10(b) for 4×4 switches), stage 0 uses two LS bits of C (c_1c_0) and stage $m-1$ uses two MS bits of C $(c_{n-1}c_{n-2})$ from the least significant to the most significant positions. In a particular order $= 0$ for $D = S \oplus C$, the forward control tag $F = S \oplus C$ or the backward control tag $B = S \oplus C$ is utilized with the k-bit partition (of S and D) per switch to set the corresponding switches to connect from all sources S to all destinations D.

According to the Latin-square permutations of destinations $(D = S \oplus (C + order) \bmod N$ or $D = \rho[(S + C + order) \bmod N])$ for the ATAPE on MINs$^+$ $(N = 2^n, d = 2^k)$, the local k-bit partitions of S and D are used to control switches in each stage for setting the connected paths through a sequence of switches across $\log_d N$ stages. □

For example $(N = 16, d = 4, k = 2)$, Fig. 10(a)-2 shows the forward routing of a connection from $S = 1\ (0001_2)$ to $D = S \oplus C = 2\ (0010_2)$ by the control $C = 3\ (0011_2)$. For the backward routing, Fig. 10(b)-2 displays the connection of $S = 1(0001_2)$ with the control $C = 6\ (0110_2)$ to $D = 7\ (0111_2)$.

3.2.3. *k*-bit routing by *stage-control on d-nary MINs*$^+$

For a network configuration using $d \times d$ switches, let C denotes the control for stage-control represented by $(c_{n-1}c_{n-2} \ldots c_\ell c_{\ell-1} \ldots c_3c_2c_1c_0)_2$, where $0 \le C < 2^n$. Thus, setting N network configurations by the default ATAPE function can be set with d patterns (per switch and stage).

Let $c_\ell c_{\ell-1}$ denote the 2-bit control from $C = (c_{n-1}c_{n-2}\ c_{n-3}\ c_{n-4} \ldots c_\ell c_{\ell-1} \ldots c_1c_0)_2$ at stage 0 to stage $m-1$ of 4×4 switches $(d = 4, k = 2)$ and the stage-control operations (see Fig. 9(b)) are

- $c_\ell c_{\ell-1} = 00$ mean that the switch setting is straight for all switches per stage.
- $c_\ell c_{\ell-1} = 01$ mean that the switch setting is cross-1 for all switches per stage.
- $c_\ell c_{\ell-1} = 10$ mean that the switch setting is cross-2 for all switches per stage.
- $c_\ell c_{\ell-1} = 11$ mean that the switch setting is cross-3 for all switches per stage.

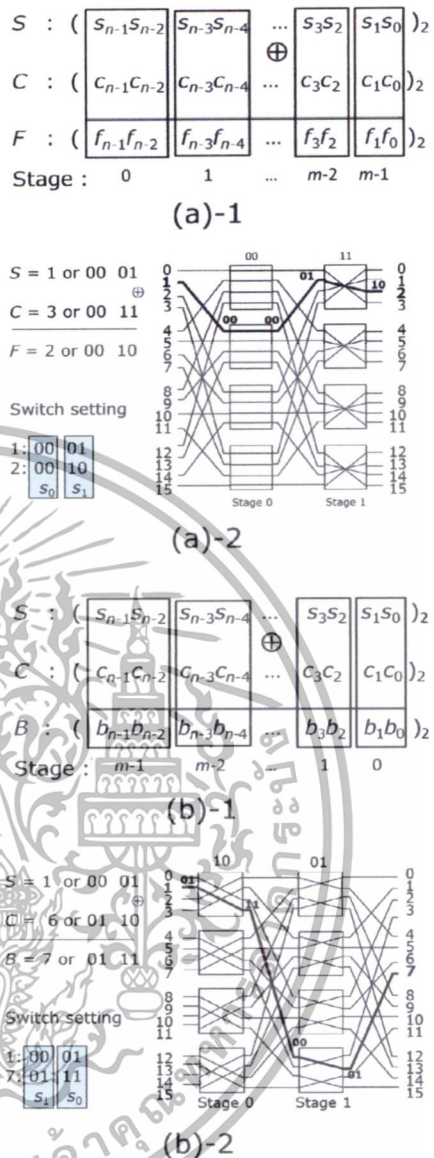


Fig. 10. $C = (c_{n-1}c_{n-2}c_{n-3} \ldots c_\ell \ldots c_1c_0)_2$ control bits for m stages (k bits per switch per stage) on MINs$^+$ $(N = 16, d = 4, k = 2)$: (a) the forward routing network and (b) the backward routing network.

The k-bit stage-control routing can be applied on both forward and backward networks across $\log_4 N$ stages. See examples in Fig. 9(b) the N switch configurations $(N = 64, n = 6, m = 3, d = 4,$ and $k = 2)$ and Fig. 11 (the ATAPE on MINs$^+$).

3.3. *The ultimate ATAPE embedding on a crossbar of MINs*$^+$

For the faster ATAPE communication, we also introduce a crossbar of MINs$^+$ that combines the fast communication of crossbar networks and the inexpensive cost of MINs$^+$. The crossbar network can be visualized as a single-stage network with the highest bandwidth (to support 1-1 connection for communication among processors directly) but it is the most expensive dynamic network (requiring $N \times N$ cross-points of switching to set N pairs of (S, D) connections at a single-stage delay). In contrary, each type of inexpensive MINs$^+$ with 2×2 switch boxes $(N = 2^n)$ uses only $nN/2$ switches but it requires $\log_2 N$ delay-stages for longer communication between processors since each MIN$^+$ topology must have $n = \log_2 N$ stages ($N/2$ switches per stage) and n ISCs.

In this study, the crossbar of MINs$^+$ is constructed with $x \times x$ cross-point switches $(x = 2^b < N)$ to select the certain MIN$^+$ subsystems for connections between sources and destinations (such as processors-to-processors, processor-to-memory modules, etc.). However, the construction cost and communication time of the crossbar of MINs$^+$ depend on the degree of the crossbar being utilized.

Table 2 illustrates different degrees $(x \times x)$ on the crossbar of MINs$^+$ $(x = 2, 4, 8,$ and 16) in order to compare cost and delay time (i.e., in terms of a number of switches and delay-stages), where the higher degree, the faster communication but the more expensive cost. For $N = 64$, the existing MINs$^+$ require $\log_2 N = 6$ delay-stages and 192 switch boxes. Alternatively, the 2×2 crossbar of MINs$^+$ is a faster type of MINs$^+$ for communication among N processors. In this network, each cross-point represents

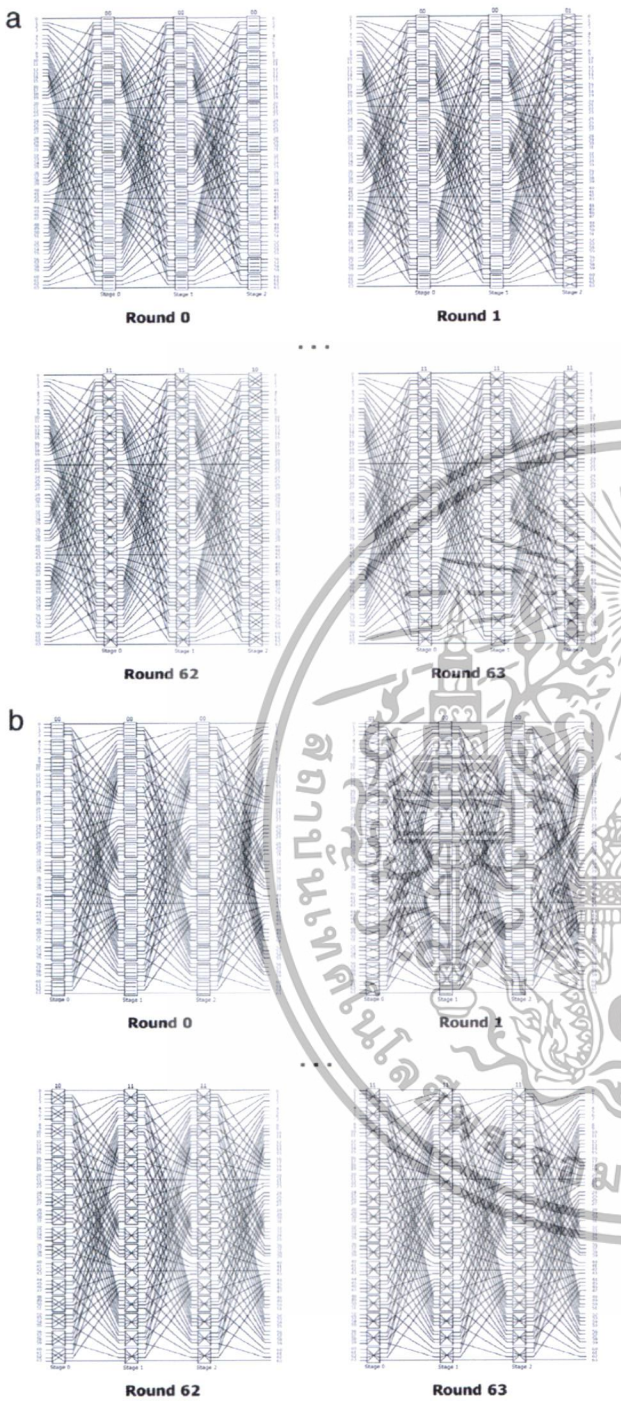


Fig. 11. ATAPE results on 4-nary-switch MINs$^+$ ($N = 64, d = 4$): (a) the forward omega network and (b) the backward flip network.

a subsystem MIN$^+$ with $N' = N/x = 32$ and $\log_2 N' = 5$ stages so that there exists one-stage decreased for the (S, D) connections and the entire system uses $N'/2 \times \log_2 N' = 320$ switch boxes. Thus, on this network our ATAPE communication is able to exchange messages with N pairs of processors through one-stage reduction. Similarly, the 4×4 crossbar of MINs$^+$, each cross-point is a subsystem MIN$^+$ with $N' = 16$ and $\log_2 N' = 4$ stages and the (S, D) connections can be processed faster with 2-stage reduction. In general, the $x \times x$ crossbar of MINs$^+$ ($x = 2^b, N = 2^n, N' = 2^{n-b}$) can be built in $n - b$ stages (or b stages decreased) with $x^2 (n - b) (N/x)/2$ switches.

Table 2
Cost and delay time of crossbar of MINs$^+$ ($N = 64$).

Crossbar degree	Size of MIN$^+$ subsystem	Number of stages decreased	Cost of a crossbar of MINs$^+$
16×16	4	4 stages	1024 switches
8×8	8	3 stages	768 switches
4×4	16	2 stages	512 switches
2×2	32	1 stage	320 switches

As a family of the fully self-routable MINs$^+$, the specific types of subsystems include omega, flip, cube or banyan$^+$, butterfly$^+$, baseline$^+$ and inverse baseline$^+$ networks. Fig. 12 depicts an example ($N = 64$), a crossbar of the butterfly$^+$ networks ($N' = N/x = 32$) with the crossbar degree $= 2 \times 2$ to allow the connections among 64 processors to 64 processors across the 5-stage MIN$^+$ subsystems.

On this special type of MINs$^+$, our ATAPE is certainly embedded as follows: Let us consider the function $D = S \oplus (C + order) \bmod N$, where $S = (s_{n-1}s_{n-2} \ldots s_1 s_0)$ and $D = (d_{n-1}d_{n-2} \ldots d_1 d_0)$. In this case, the n bits of S and D are partitioned to operate along the $x \times x$ crossbar first and then move forward to the MIN$^+$ subsystems across $n - b = \log_2 N/x$ stages. Let $x \times x = 2^b \times 2^b$ denote rows and columns of the crossbar of MINs$^+$. The b high-order bits of $S = (s_{n-1} .. s_{n-b})$ is used to select row (of input-ports) and the b high-order bits of $D = (d_{n-1} .. d_{n-b})$ is used to select column (of output-ports). The remaining $n - b$ bits are used with the variation of self-routing on each MIN$^+$ (in Section 3.1).

Fig. 12 illustrates an example of self-routing on a 2×2 crossbar of MINs$^+$ ($N = 64, x = 2$, and $N' = 32$). For single-pass permutations, there is 1-to-1 permutation mapping $\rho = (a_0 a_1 \ldots a_S \ldots a_{N-1})$ among sources (S) and destinations $\rho(S)$ of the network, where $\rho(S) = D \in \{0, 1, 2, 3, \ldots, N - 1\}$ for $S = 0, 1, 2, 3, \ldots, N - 1$. For example, the network configuration is set by the permutation $\rho = (7\,6\,5\,4\,3\,2\,1\,0\,15\,14\,13\,12\,11\,10 \ldots 56)$ in order to map 64 inputs to 64 outputs by a (6-bit) stage-control 000111_2 with the most significant one-bit of S for row-selection and that of D for column-selection. The remaining five bits for one-bit partition (per switch) per stage can set each of all switches independently by using the same C (i.e., input 0 to output 7, input 1 to output 6, input 2 to output 5, ..., and input 63 to output 56). Those N processors are set for (S, D) connections in parallel. For instance, the connection $(2, 5)$ with the control $C = 7$ (or $000111)_2$ and order $= 0$, the destination $D = S \oplus (C + 0)_{\bmod 16} = 5$ (or $000101)_2$ can be mapped to the corresponding cross-point to subsystems by using one bit s_5 (=0) to select row 0 (in-port of crossbar) and one bit d_5 (=0) to select column 0 (out-port of crossbar) and the remaining 5 bits are assigned for the fully self-routable MIN$^+$ subsystems. Thus, for the (2, 5) connection on the butterfly$^+$ network with $N' = 32$, five switches on five stages can be set via in-port 0 of S to out-port 0 of D at stage 0 and stage 1, in-port 0 of S to out-port 1 of D at stage 2, in-port 1 of S to out-port 0 of D at stage 3, and in-port 0 of S to out-port 1 of D at stage 4. Similarly, the dynamic f-in-1 ATAPE function $D = \rho[(S + C + order) \bmod N]$ can also be operated on the crossbar of MINs$^+$ by the (S, D) self-routing efficiently.

4. Applications of our ATAPE embedding

This section presents the empirical results of applying our proposed ATAPE algorithm with the variation of self-routing for any network belonging to a class of d-nary MINs$^+$. For example ($N = 8, d = 2$), the ATAPE algorithm generates 8 configuration patterns for $n = \log_2 N = 3$ stages, according to round 0 ($C = 0\,0\,0_2$) to round 7 ($C = 1\,1\,1_2$), by using the same 1-bit control per

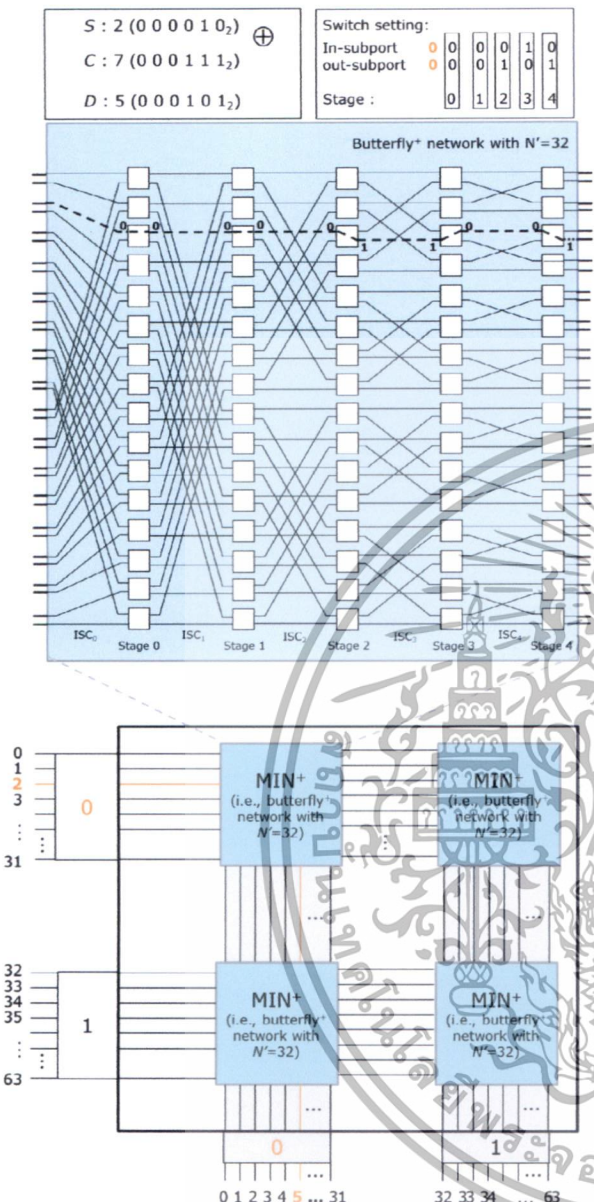


Fig. 12. An example of a crossbar of MINs$^+$ ($N = 64$, Degree $= 2 \times 2$) with each cross-point representing the butterfly$^+$ network ($N' = 32$) and (S, D) routing by using the control $C = 7_{10}$ (or 000111_2).

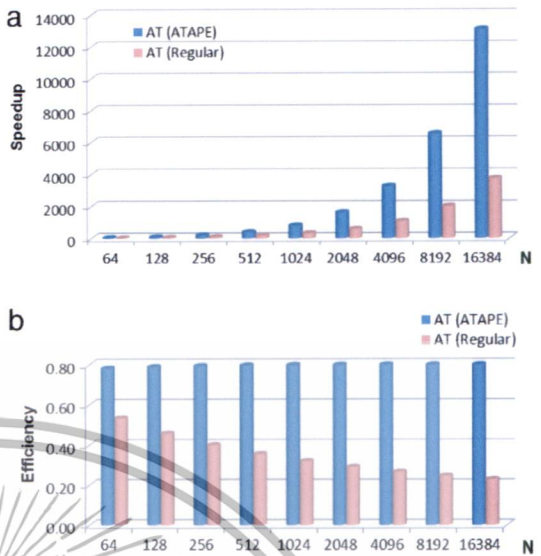


Fig. 13. Performance of the A^T_{ATAPE} compared to the regular A^T.

switch per stage (of $N/2$ switches) on such MINs$^+$, as presented in Section 4.1 (Fig. 14) and on a 2×2 crossbar of MINs$^+$ ($N = 16$) in Section 4.2 (Fig. 15). Lastly, an efficient application of ATAPE in a parallel matrix transposition (Fig. 16) and experimental results are presented in Section 4.3.

4.1. Our ATAPE on the d-nary MINs$^+$

Our ATAPE algorithm is applicable on all types of the d-nary MINs$^+$, which includes forward routing networks (i.e., omega, butterfly$^+$, and ibaseline$^+$ networks) and backward routing networks (i.e., flip, cube or banyan$^+$, and baseline$^+$ networks). The results of applying our ATAPE on the forward routing networks with $d \times d$ switches ($d = 2$) for N network configurations are shown in Fig. 14(a) by using the counter control $C = (c_{n-1}c_{n-2}c_{n-3} \ldots c_1c_0)_2$ for self-routing with the one-bit partitions (on stages $0, 1, 2, \ldots, n-1$) from the most significant to the least significant bits. For example, in round 2, N processors are set for (S, D) connections in parallel, which are $P_0 \to P_2$, $P_1 \to P_3$, $P_2 \to P_0$, $P_3 \to P_1$, $P_4 \to P_6$, $P_5 \to P_7$, $P_6 \to P_4$, $P_7 \to P_5$. In particular, the case of $S = 1$ (or $001)_2$ and $C = 2$ (or $010)_2$ provides a switch setting for the destination processor $D = S \oplus (C + order) \bmod N = 3$ (or $011)_2$ with order $= 0$ (i.e., stage 0 is connecting with local ports 0 to 0, stage 1 with local ports 0 to 1, and stage 2 with local ports 1 to 1).

The backward routing networks ($d = 2$) are set for (S, D) connections in parallel the same as the forward routing networks, except the (n-bits) counter control C is used from the least significant to the most significant bits, as shown in Fig. 14(b) (i.e., the connection $(1, 3)$ of round $r = 2$ can also set all switches via local ports 1 to 1 at stage 0, with local ports 0 to 1 at stage 1, and with local ports 0 to 0 at stage 2). Lastly, Fig. 14(c) displays the backward routing of a 4-nary switch MIN$^+$ with $N = 16$ and $d = 4$. Note that for other types of the d-nay MINs$^+$, both ATAPE functions have been verified successfully by the programming test for increasing MIN$^+$ sizes (N) up to 2^{20}.

4.2. The powerful ATAPE on a crossbar of MINs$^+$

For the crossbar of MINs$^+$, first we apply the ATAPE on a 2×2 crossbar of MINs$^+$ ($N = 16 = 2^4, x = 2, d = 2$) with MIN$^+$ subsystems ($N' = N/x = 8$) and then other crossbar degrees ($\times$) and N have been evaluated correctly by programming. Thus, all (S, D) connections use one high-order bit of sources $S = (s_{n-1}s_{n-2} \ldots s_1s_0)$ anddestination $D = (d_{n-1}d_{n-2} \ldots d_1d_0)$ for selecting row and column of the crossbar. Then, the remaining $n-1$ bits are used to set self-routing of MIN$^+$ across $n-1$ stages. The results of applying our ATAPE on the crossbar of forward/backward routing networks for N network configurations are shown in Fig. 15 ($N = 16$), the crossbar of the omega and cube networks. Routing on the crossbar of self-routable MINs$^+$ contains two parts: (1) the most significant 1-bit (or a high-order bit) for the crossbar selection (i.e., the crossbar is selected with row 0 ($s_3 = 0$) and column 0 ($d_3 = 0$)); and (2) the remaining 3-bit for switch setting along the routing paths of MIN$^+$ subsystems. Fig. 15(a) presents the results of our ATAPE on the 2×2 crossbar of omega networks ($N = 16, x = 2, N' = N/x = 8$), where the first half ($N/2$) permutations of

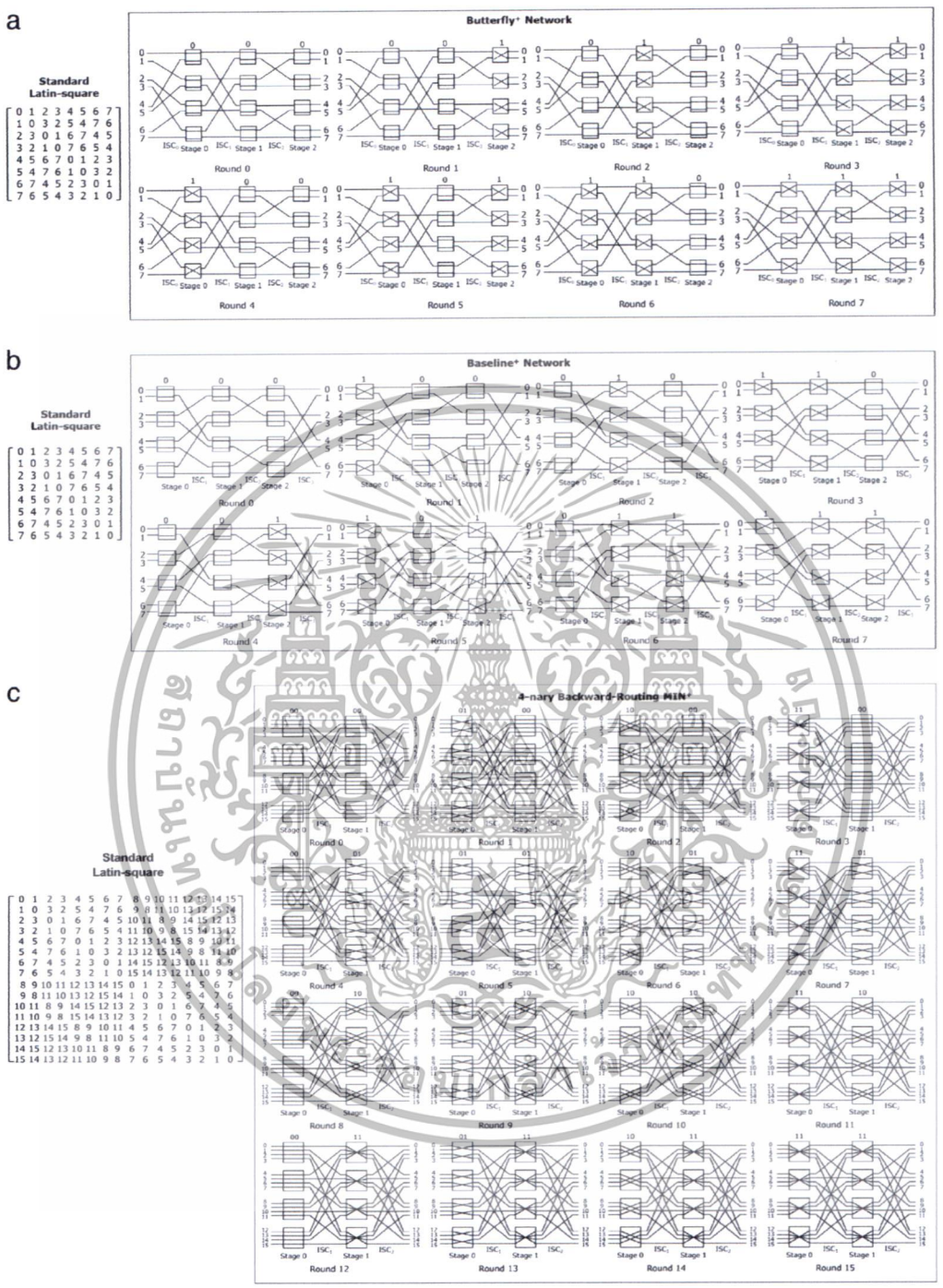


Fig. 14. ATAPE empirical results on MINs$^+$ including (a) butterfly$^+$ network ($N = 8, d = 2$), (b) baseline$^+$ network ($N = 8, d = 2$), and d-nary backward routing MIN$^+$ ($N = 16, d = 4$).

the Latin-square are processed on the cross-subsystems (0, 0) and (1, 1) and the second half ($N/2$) permutations are processed on the cross-subsystems (0, 1) and (1, 0), according to the results of $D = S \oplus C$. For instance, with round $r = 1$ (or $001)_2$, all connections ($P_0 \to P_1, P_1 \to P_0, P_2 \to P_3, P_3 \to P_2, P_4 \to P_5, \ldots, P_{15} \to P_{14}$) are set via local ports 0-0 at stage 0, local ports 1-1 at stage 1, and local ports 0-1 at stage 2. Similarly, Fig. 15(b) demonstrates the backward routing network, the 2×2 crossbar of banyan$^+$ networks ($N = 16, x = 2, N' = N/x = 8$).

4.3. Fast ATAPE process for matrix transposition

This section presents an efficient application of our ATAPE communication in the parallel $N \times N$-matrix transposition on

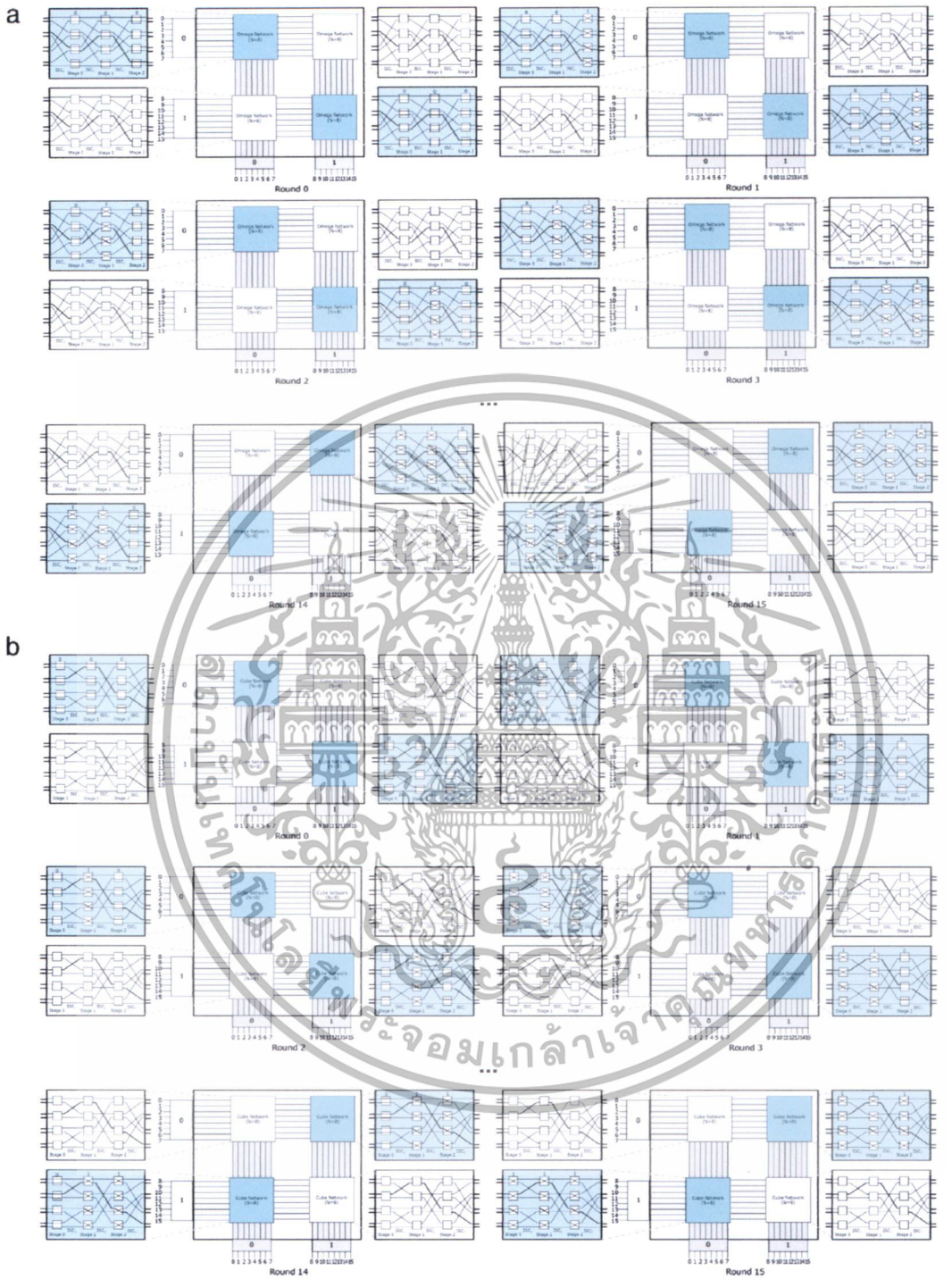


Fig. 15. ATAPE empirical results on crossbar of MINs$^+$ including (a) crossbar of omega networks and (b) crossbar of banyan$^+$ networks ($N = 16, x = 2, d = 2$) with Degree $= 2 \times 2$.

N-processor MINs$^+$ in optimal $O(N + \log_2 N)$, while that of the sequential $A^T = O(N^2)$ and that of the parallel A^T on the $N \times N$ torus networks $= O(N)$ and $O(N \log_2 N)$ on the hypercube networks of size $N = 2^n$.

In addition, in the performance evaluation we perform the experiments (by simulation study) to compare the speedup ($S_P = T_1/T_P$) and efficiency ($E_P = S_P/P$) of our ATAPE-based matrix transposition (A^T_{ATAPE}) and the regular A^T on the MINs$^+$, where T_1 refers to the response time of A^T by one PE (processing element) and T_P refers to the response time of parallel A^T processing using P PEs. For the simulated network systems, let us define the following parameters: a clock period of communication time (p_{comm}) is a 1.25-fold of a clock period of computing time (p_{comp}) and p_{comp} is a 4-fold of p_{delay} (a clock period of delay time through

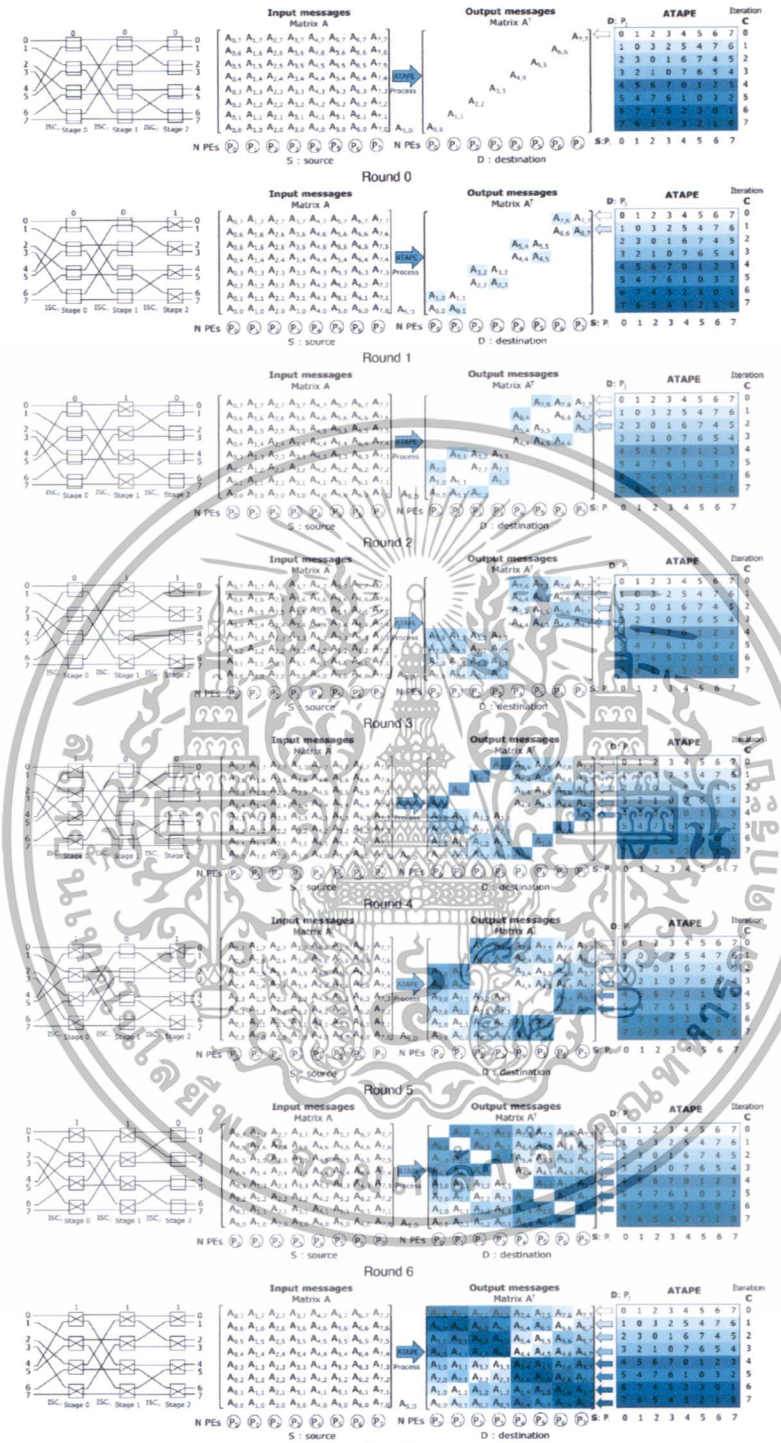


Fig. 16. ATAPE communication results of parallel matrix transposition with $N = 8$.

a stage). Fig. 13 shows the performance of the ATAPE-based A^T under the fine-grain computing ($P = N$), which yielded speedup close to P and efficiency close to 1 for all network sizes N (= 64–16,384 PEs), according to the optimal ATAPE communication across the multistage pipelining using the on-chip counter, while the efficiency of the regular A^T decreased when N increased.

Next, in order to confirm "How our ATAPE communication works efficiently and correctly", we demonstrate the empirical results of the ATAPE-based A^T ($N = 8$) step-by-step on the butterfly$^+$ network (see Fig. 16), where the A^T is computed in N rounds (except round 0) with the function $D = S \oplus C$ and the control $C = 1, 2, 3, \ldots, N-1$. Firstly, the matrix A is divided into N sections and assigned one section of N elements $(A_{i,0}, A_{i,1}, A_{i,2}, \ldots, A_{i,7})$ to each processor P_i, where $i = 0, 1, 2, 3, \ldots,$ and $N-1$. Finally, after send/receive $A_{S,D}$ (the personalized message of the matrix A) from all sources (S) to all

destinations (D) in $N-1$ rounds, the results of the A^T are in N processors.

For instance, in round 1, the $A_{S,D}$ results of the permutation [1 0 3 2 5 4 7 6] are $A_{0,1}$ (from P_0 to P_1), $A_{1,0}$ (from P_1 to P_0), $A_{2,3}$ (from P_2 to P_3), $A_{3,2}$ (from P_3 to P_2), $A_{4,5}$ (from P_4 to P_5), $A_{5,4}$ (from P_5 to P_4), $A_{6,7}$ (from P_6 to P_7), $A_{7,6}$ (from P_7 to P_6) and so on in round 2 to round 7. After the final round, the $A_{S,D}$ results of the permutation [7 6 5 4 3 2 1 0] transfer the last N elements ($A_{0,7}, A_{1,6}, A_{2,5}, A_{3,4}, A_{4,3}, A_{5,2}, A_{6,1}, A_{7,0}$) from all processors P_S to P_D.

5. Conclusion

In this paper, we present the on-chip ATAPE (all-to-all personalized exchange) embedding on the fully self-routable MINs$^+$ with $d \times d$ switches, including omega, flip, banyan$^+$, butterfly$^+$, baseline$^+$ networks, etc. Our ATAPE communication can be embedded on chip directly and can be operated by the (S, D) self-routing and the stage-/switch-control routing. Our ATAPE on such MINs$^+$ works ideal time by the on-chip counter incorporate with the multistage pipelining to transfer specific messages by every pair of N processors, using N successive rounds ($C = 0$ to $N-1$). Our method allows user programs to be able to specify a particular order of messages with either the (default) static function $D = S \oplus (C + order) \mod N$ or the (optional) f-in-1 dynamic function $D = \rho[(S + C + order) \mod N]$ in optimal $O(N + \log_2 N)$, where $0 \leq order < N$. In addition, our ATAPE functions can be ultimately embedded on the d-nary-switch MINs$^+$ and the crossbar of MINs$^+$. By simulation study, the experimental results of the ATAPE-based matrix transposition A^T on MINs$^+$ were presented with the significant speedup being close to P for all network sizes $N = 64$–16,384 PEs.

Acknowledgments

We would like to gratefully thank the Office of the Higher Education Commission (Bangkok, Thailand) for the research grant OHEC 04-01-04-001 (2011–2014) to Miss Roselin Petagon to fulfill the requirement of doctoral degree (Ph.D.) in Computer Science at KMITL and also gratefully thank King Mongkut's Institute of Technology, Ladkrabang (KMITL), Bangkok, Thailand, and Chiang Mai Rajabhat University (Chiang Mai, Thailand) for their great support and corporation.

References

[1] Z. Chen, Z.J. Liu, Z.L. Qiu, Bidirectional shuffle–exchange network and tag-based routing algorithm, IEEE Commun. Lett. 7 (3) (2003) 121–123.
[2] W.Y. Chou, C. Chen, All-to-all personalized exchange in generalized shuffle–exchange networks, Theoret. Comput. Sci. 411 (3) (2010) 1669–1884.
[3] D.H. Lawrie, Access and alignment of data in an array processor, IEEE Trans. Comput. C-24 (12) (1975) 1145–1155.
[4] V.W. Liu, C. Chen, R.B. Chen, Optimal all-to-all personalized exchange in d-nary banyan multistage interconnection networks, J. Comb. Optim. 14 (10) (2007) 131–142.
[5] A. Massini, All-to-all personalized communication on multistage interconnection networks, Discrete Appl. Math. 128 (2) (2003) 435–446.
[6] D.S. Parker, Notes on shuffle/exchange-type switching networks, IEEE Trans. Comput. C-29 (3) (1980) 213–222.
[7] M.C. Pease, The indirect binary n-cube microprocessor array, IEEE Trans. Comput. C-26 (5) (1977) 458–473.
[8] B. Prisacari, G. Rodriguez, C. Minkenberg, Generalized hierarchical all-to-all exchange patterns, in: 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, May 2013, pp. 537–547.
[9] D.S. Scott, Efficient all-to-all communication patterns in hypercube and mesh topologies, in: Proc. Sixth Distributed Memory Computing Conference, 1991, pp. 398–403.
[10] Y.J. Suh, K.G. Shin, Efficient all-to-all personalized exchange in multidimensional torus networks, in: Proc. Int. Conf. Parallel Processing, 1998, pp. 468–475.
[11] Y.J. Suh, K.G. Shin, All-to-all personalized communication in multidimensional torus and mesh networks, in: Proc. Int. Conf. Parallel Processing, Vol. 12, No. 1, 2001, pp. 38–59.
[12] C.-L. Wu, T.-Y. Feng, On a class of multistage interconnection networks, IEEE Trans. Comput. 29 (8) (1980) 694–702.
[13] Y. Yang, J. Wang, Optimal all-to-all personalized exchange in self-routable multistage networks, IEEE Trans. Parallel Distrib. Syst. 11 (3) (2000) 261–274.
[14] Y. Yang, J. Wang, Optimal all-to-all personalized exchange in a class of optical multistage networks, IEEE Trans. Parallel Distrib. Syst. 12 (9) (2001) 567–582.

Roselin Petagon received the B.S. degree in 2005 from ChiangMai Rajabhat University, ChiangMai, Thailand and the M.S. degree in Computer Science in 2008 from King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand. Since 2008, she has been a lecturer of Computer Science at ChiangMai Rajabhat University, ChiangMai, Thailand. Miss Petagon is a Ph.D. candidate (in Computer Science), at KMITL, Bangkok, Thailand. Her research interests include parallel algorithms and architectures, interconnection networks, and distributed computing.

Jeeraporn Werapun received the B.S. degree from Kasetsart University, Bangkok, Thailand, the M.S. degree from Chulalongkorn University, Bangkok, Thailand, the M.S. degree in Computer Science in 1993, and the D.Sc. degree in Computer Engineering (Parallel and Distributed Systems) in 1999 from the George Washington University, Washington, DC, USA. Dr. Werapun is an Associate Professor of Computer Science at King Mongkut's Institute of Technology, Ladkrabang (KMITL), Bangkok, Thailand. Her research interests include parallel and distributed systems, multistage interconnection networks, and parallel algorithms.

ภาคผนวก ข

ผลงานวิจัยตีพิมพ์ที่ Journal of Parallel and Distributed Computing

Journal homepage : www.elsevier.com/locate/jpdc

R. Petagon and J. Werapun, VA-DE: Valuable ATAPE with Dynamic Embedding and Super-pipeline Scheduling on Partitionable Multistage Interconnection Networks+, J. Parallel Distrib. Comput. 102 (2017), 1-15.

J. Parallel Distrib. Comput. 102 (2017) 1–15

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

VA-DE: Valuable ATAPE with dynamic embedding and super-pipeline scheduling on partitionable multistage interconnection networks+

Roselin Petagon *, Jeeraporn Werapun

Department of Computer Science, Faculty of Science, King Mongkut's Institute of Technology, Ladkrabang, Bangkok 10520, Thailand

HIGHLIGHTS

- Propose prototype of partitionable MINs+ using I/O cross-control switches.
- Optimal VA-DE (Valuable ATAPE with Dynamic Embedding) supports multiple tasks.
- Embedding new ATAPE with super-pipelining and triple-right scheduling.
- Prove correctness of d-nary MINs+ and improve complexity of new ATAPE = $O(N/x)$.
- Remarkable speedup and throughput were strongly confirmed on PMINs+.

ARTICLE INFO

Article history:
Received 5 December 2015
Received in revised form
29 July 2016
Accepted 23 November 2016
Available online 30 November 2016

Keywords:
Partitionable MINs+ (multistage interconnection networks+) using I/O cross-control switches
Embedding partitionable ATAPE (all-to-all personalized exchange) and automatic mapping on PMINs+
Optimal VA-DE (valuable ATAPE with dynamic embedding)
Double jumping of controls and super-pipelining for the triple-right scheduling (right task, right section, right time)

ABSTRACT

The optimal ATAPE (all-to-all personalized exchange) is a key solution to achieve the optimal communication in parallel matrix-transposition, parallel fast-Fourier-transformation, etc. The ATAPE has been extensively studied on static networks and dynamic binary-switch MINs (multistage interconnection networks). Recently, the ultimate ATAPE was successfully embedded on MINs+ and the fast crossbar of MINs+. However, the MINs+ cannot be partitioned for multiple tasks and the crossbar of MINs+ is more expensive and less subsystem-utilization. Therefore, this paper proposes the prototype of the $x \times x$ crossbar of partitionable MINs+ (PMINs+) using I/O cross-control switches and the generalized d-nary-switch MINs+. In addition, the optimal VA-DE (valuable ATAPE with dynamic embedding) with triple-right scheduling (right-task, right-section, right-time) is presented for multiple tasks of flexible sizes $N'' = N/2^t$ (where $t \leq \log_2 x$) and the efficient subsystem-utilization on the PMINs+. New partitionable ATAPE embedding and automatic mapping for the partitionable section y (=0 to $x-1$) are $D^y = S^y \oplus C_I^y$ (global) and $ID^y = (D^y \% N'') \oplus c^z$ (local) using double-jumping controls $C_I^y = (yN/x + I) \mod N$ (to right section) and $c^z = zN''/x$ (to right subtask), incorporated with super-pipelining for outperforming optimal time $O(N/x)$ over $O(N)$, the best of existing ATAPEs. In experiments, remarkable speedup and throughput were strongly confirmed on the simulated PMIN+ systems.

1. Introduction

Many parallel and distributed computing program-tasks in engineering and applied science (such as matrix transposition, fast Fourier transformation (FFT), all-to-all broadcast, etc.) involve a large amount of data sending/receiving overhead. Thus, all-to-all personalized exchange (ATAPE) is the most effective communication that can reduce such an overhead of the parallel and distributed systems. The ATAPE function relies on the efficient parallel point-to-point communication of sending and receiving messages, which each pair of processors can exchange the specific message to other corresponding processors in optimal time ($O(N + \log_2 N)$). In the past, a number of ATAPE approaches were proposed on both static networks [9–12] and dynamic MINs (multistage interconnection networks) [2,4,5,8,14,15]. On the static networks (i.e., hypercube, mesh, torus, etc.), the *XOR*-based ATAPE function $D = S \oplus C$ [10] could be utilized efficiently but the communication latency between processors on those networks was not consistency. To solve that problem, in 2013 the hierarchical ATAPE $h^D = (h^S + h^P)\% N^I$ [9] for unique addresses of tasks was introduced on static dragonfly networks through N^I switching layers.

On the other hand, a variety of the optimal ATAPE algorithms were implemented on a class of MINs [1,3,6–8,13] including

* Corresponding author.
E-mail addresses: 54650501@kmitl.ac.th (R. Petagon), jeeraporn.we@kmitl.ac.th (J. Werapun).

http://dx.doi.org/10.1016/j.jpdc.2016.11.010
0743-7315/

omega, flip, cube, banyan, butterfly, and baseline networks since the dynamic configurations of MINs have more consistency in communication-latency and better scalability for large networks than those of the static networks. In each type of the MINs (of size N) using 2×2 switch boxes, the network consists of $n = \log_2 N$ delay-stages and $N/2$ switches per stage. Most of the existing ATAPE methods [2,4,5,14,15] relied on the specific switch-patterns for certain switch-/stage-control routing on each MIN (i.e., baseline, banyan, butterfly, etc.). Those ATAPE contributions focused on the preprocessing of the specific Latin square of admissible permutations for unique-path connections of sources (S) and destinations (D). However, a limitation of their self-routing methods using such permutations of switches was (after starting from S) only D driven for routing across n switch-stages.

Recently, in early 2016 the ATAPE on-chip embedding [8] was proposed to solve that problem on the complete MINs$^+$ with special port-adjusting, including the baseline$^+$, the butterfly$^+$, etc. The fast ATAPE embedding can operate by the (S, D) self-routing and the switch-/stage-control routing on the fully self-routable MINs$^+$. For all MINs$^+$, two ATAPE functions were embedded on-chip in $O(N + \log_2 N)$ time for flexible Latin squares of admissible permutations with N options according to any permutation order between 0 and N. In that study, the (default) ATAPE was the static *XOR*-function $D = S \oplus (C + \text{order}) \bmod N$, where C refers to incrementing on-chip counter control (0 to $N - 1$). The second ATAPE was the f-in-1 dynamic function $D = \rho\,[(S + C + \text{order}) \bmod N]$ for fN Latin squares, introduced for flexible mapping other ATAPE functions during runtime. Lastly, those ultimate ATAPE functions could be adjusted and applied successfully on a new fast MIN$^+$, called the $x \times x$ crossbar of MINs$^+$ (CMINs$^+$), for the faster ATAPE communication throughout the fewer delay-stages ($n' = \log_2 N/x$) and the improved optimality in $O(N + \log_2 N/x)$ time.

The advantage of the crossbar of MINs$^+$ is its fast communication but its limitations are more expensive cost and less subsystem utilization, while the MINs$^+$ cannot be partitioned effortlessly for multiple tasks. In practice, the scalable MIN-based systems must be able to process many tasks at a time. However, the existing ATAPE methods on MINs [2,4,5,14,15] and on MINs$^+$ [8] can execute only one ATAPE-task at a time since it is too complicated to partition the existing MINs for executing multiple tasks.

In this paper, our innovation and contribution is that we propose the prototype of partitionable MINs$^+$ (PMINs$^+$) and introduce the optimal VA-DE (valuable ATAPE with dynamic embedding). The optimal VA-DE is a flexible ATAPE embedding and automatic mapping on the PMINs$^+$ with triple-right scheduling (right task, right section, right time) for executing multiple tasks with flexible sizes N'' (=$N, N/2, \ldots, N/2^t, \ldots, N/x$). In this study, we focus on the efficient utilization of the $x \times x$ crossbar of PMINs$^+$ ($N = 2^n$, $x = 2^b$) by partitioning the PMINs$^+$ into subsystems (of sizes $\leq N$) to be able to process multiple tasks at a time with 100% utilization. On the feasible PMIN$^+$, we add I/O cross-control switches for setting section y ($0 \leq y < x$) independently, O-switch = (I-switch + y) mod x, where I-switch = $0, 1, 2, \ldots, x - 1$. For the triple-right scheduling of multiple tasks on the PMINs$^+$, we employ the super-pipelining technique to be able to start the next instruction on the different section y in every $1/x$ clock. On the PMINs$^+$ with super-pipeline scheduling (degree x), the system is partitioned into x sections ($y = 0, 1, 2, \ldots, x - 1$) and utilized automatically by double jumping of controls $C_l^y = (yN/x + l) \bmod N$ to next section and $c^z = zN''/x$ to next subtask (where $0 \leq l < N/x$ and $0 \leq z < 2^t$) for all (S^y, D^y) communications, according to the global $D^y = S^y \oplus C_l^y$ mapping to the local $lD^y = (D^y \bmod N'') \oplus c^z$ of task size N'', where $lS^y = S^y \bmod N''$ and $S^y = 0, 1, 2, \ldots, N - 1$. In experiments, the remarkable speedup and plentiful throughput are achieved on the simulated PMINs$^+$ (using $N \leq 32\,768$ processors and crossbar degrees $x \leq 16$).

The remaining part of this paper is organized as follows: Section 2 reviews related work of existing ATAPE methods on MINs and MINs$^+$. Section 3 proposes the prototype of partitionable MINs$^+$ using I/O cross-control switches and super-pipelining. Section 4 presents the optimal VA-DE algorithm, the triple-right scheduling of ATAPE for the optimal embedding and mapping on the PMINs$^+$. Section 5 presents the experimental results of our partitionable ATAPE and super-pipeline scheduling on the PMINs$^+$. Section 6 encloses the conclusion of our study.

2. Related works

For dynamic MIN networks, the existing *XOR*-based ATAPE function $D = S \oplus C$ [10] on static networks cannot be directly applied on the recursive MINs (i.e., baseline, butterfly, etc.) because of their unique self-routing. To overcome the routing limitation of such MINs, the other ATAPE methods (see Table 1) were presented by using admissible permutations of non-conflict switch patterns for each MIN using 2×2 switches [2,5,14,15] and d-nary switches [4]. With the preprocessing Latin square of admissible permutations on each of the MINs, the specific switch-/stage-/D-control routing was applicable but that restrict method is too complicated for embedding on-chip.

2.1. ATAPE on-chip embedding on MINs$^+$

In early 2016, the efficient ATAPE function [8] was proposed for embedding on-chip on a class of fully self-routable MINs$^+$. In that full-routing solution, each of recursive MINs$^+$ (i.e., the baseline$^+$, the butterfly$^+$, the banyan$^+$, etc.) was fulfilled with a special ISC (inter-stage connection) for port-adjusting to solve the (S, D) self-routing (i.e., ISC$_r$ (circular right-shift) on the banyan$^+$, ISC$_0$ (circular left-shift) on the butterfly$^+$, etc.). The complete self-routing on those MINs$^+$ supports switch-/stage-/D-control routing and (S, D) self-routing. In that study, two efficient ATAPE functions were embedded on-chip in optimal $O(N + \log_2 N)$ by incrementing the counter control $C = 0$ to $N - 1$. First, the (default) static function is $D = S \oplus (C + \text{order}) \bmod N$, which provided N sets of Latin squares, specified by the arbitrary order between 0 and N (see an example in Fig. 1). Second, the f-in-1 dynamic function is $D = \rho[(S + C + \text{order}) \bmod N]$ for flexible f initial permutations specified by users ($\rho[i]$, $i = 0, 1, 2, \ldots, N - 1$) for other fN Latin squares. In that function, the pointer to the destinations allowed the user programs to be able to specify their own functions by setting the initial permutation only.

2.2. Ultimate ATAPE on a crossbar of MINs$^+$

On a crossbar of MINs$^+$ (CMINs$^+$) [8], two ATAPE functions in Section 2.1 could be embedded with faster communication. In that study, the CMIN$^+$ contribution preserved the fast communication of crossbar networks and the low cost of MINs$^+$, according to the crossbar degree (x). In the CMINs$^+$ ($N = 2^n$), each cross-point switch represented MIN$^+$ subsystem with $N' = N/x = 2^{n'}$ and $n' = \log_2 N'$ stages. Thus, self-routing on the CMINs$^+$ contained two addressable parts: (1) the b high-order bits for the crossbar selection, and (2) the remaining n' bits for switch setting on the routing paths of MIN$^+$ subsystems. For example, Fig. 2 displays N network configurations of applying the ATAPE function $D = S \oplus (C + \text{order}) \bmod N$ (using order = 0) on the 2×2 crossbar of MINs$^+$ ($N = 16$, $x = 2$). On such a CMIN$^+$, the 4-bit addresses of sources (S) and destinations (D) are partitioned into 2 parts for the (S, D) self-routing: (1) one high-order bit for the crossbar selection and (2) the low-order 3 bits for switch setting across 3 stages of the MIN$^+$ subsystems. Applying the function $D = S \oplus C$, the first half ($N/2$) permutations of the Latin-square (rounds 0–7) are

Table 1
The comparison between the existing ATAPE methods on MINs.

ATAPE routing features	MINs			GSENs	MINs$^+$	CMINs$^+$
	2000 [14,15]	2003 [5]	2007 [4]	2010 [2]	On-chip ATAPE embedding [8]	
Dynamic self-routing	Half	Half	Half	Half	Full	Full
○ Switch-control routing		✓		✓	✓	✓
○ Stage-control routing	✓		✓	✓	✓	✓
○ D-control routing	✓	✓	✓	✓	✓	✓
○ (S, D) Self-routing					✓	✓
Latin-square (LS) of destinations (D) for self-routing						
○ Need a specific LS for each MIN	✓	✓	✓	✓		
○ Can use a standard LS for all MINs$^+$					✓	✓

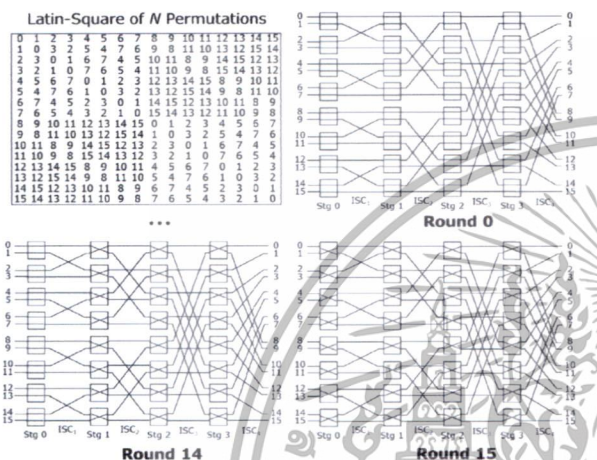


Fig. 1. ATAPE on MINs$^+$: $D = S \oplus (C + \text{order}) \bmod N$ (using order $= 0$) on the banyan$^+$ network ($N = 16$).

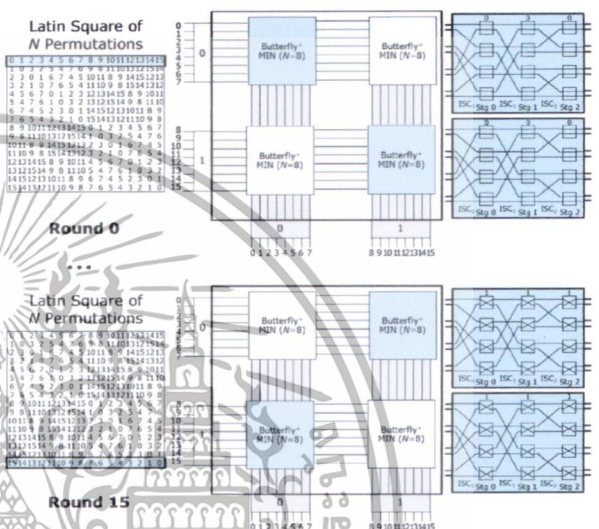


Fig. 2. ATAPE on a 2×2 crossbar of MINs$^+$ ($N = 16, x = 2$): $D = S \oplus (C + \text{order}) \bmod N$ (using order $= 0$) on the butterfly$^+$ network ($N' = 8$).

processed on the subsystems (0, 0) and (1, 1) and the second half permutations (rounds 8–15) are processed on the subsystems (0, 1) and (1, 0).

Although communication of the CMINs$^+$ is faster but the construction cost is expensive and the subsystem utilization is $(1/x)$ 100% only, while the MINs$^+$ accomplish the 100% utilization but the MINs$^+$ cannot be simply partitioned for executing multiple tasks. However, the existing ATAPE methods on the MINs [2,4,5,14,15], the MINs$^+$ and the crossbar of MINs$^+$ [8] could manage many ATAPE tasks but only one-by-one at a time whereas the scalable parallel and distributed systems must be able to execute multiple tasks at the same time.

Table 2 displays advantages and limitations of the existing ATAPE approaches on the MINs. In order to overcome the limitations (i.e., only one task at a time, less utilization on CMINs$^+$, non-partitionable MINs$^+$, etc.), in this study we introduce a new embedded partitionable ATAPE function $D^y = S^y \oplus C_l^y$ (where $C_l^y = (yN/x + l) \bmod N$; $0 \le y < x$; $0 \le l < N/x$) on the partitionable MINs$^+$ (PMINs$^+$) with 100% utilization (in Section 3). Embedding pATAPE with super-pipelining (in Section 4) supports multiple tasks of flexible size N'' $(=N, N/2, N/4, \ldots, N/2^t, \ldots, N/x)$. By the triple-right scheduling, the experimental results of pATAPE on PMINs$^+$ (in Section 5) outperform those of ATAPE on CMINs$^+$ [8], according to the super-pipelining with CPI (clock per instruction) $= 1/x$.

3. The proposed partitionable MIN$^+$ prototype

This paper proposes the prototype of the partitionable MINs$^+$ (PMINs$^+$) by renovating the crossbar of MINs$^+$ [8] with I/O cross-control switches. First, Section 3.1 presents the generalized d-nary MINs$^+$ using special port-adjusting. Then, Section 3.2 proposes the $x \times x$ crossbar of PMINs$^+$ ($N = 2^n$, $x = 2^b$), by adding the I/O cross-control switches for external signal-controlling of independent subsystems. Finally, Section 3.3 demonstrates the effective 100% subsystem utilization of the PMINs$^+$ by applying a super-pipelining technique incorporated with the triple-right scheduling (right task, right section, and right time).

3.1. The generalized d-nary MINs$^+$ with port adjusting

In the feasible PMINs$^+$, the primitive subsystems must be the complete MINs, including the omega [3], the R-net [6], the cube [7], and the MINs$^+$ [8] (i.e., the banyan$^+$, the butterfly$^+$, the baseline$^+$, and the invert baseline$^+$ networks), etc. In our previous work, new recursive MINs$^+$ [8] were introduced with port-adjusting for the full (S, D) routing. However, the complete MINs [3,6,7] and MINs$^+$ [8] were presented with correctness proven on 2×2 switches, see examples in Fig. 3. In general, this section presents the generalized d-nary MINs$^+$ with the general port-adjusting, derived for each of the recursive MINs$^+$ (see Fig. 4 for $d = 4$, $k = 2$) and the correctness of (S, D) self-routing, as follows: In the 4-nary omega network, the uniform ISC (inter stage connection) for $m = \log_4 N$ stages, is the 2-bit circular left-shift operation. The correct self-routing of processors (I, J) is set from input $I = (i_{n-1}i_{n-2}\ldots i_2 i_1 i_0)$ to output $J = (j_{n-1}j_{n-2}\ldots j_2 j_1 j_0)$ through m ISCs and m switches (using in-port $(i_{\ell+1}i_\ell)$ of I to out-port $(j_{\ell+1}j_\ell)$ of J), derived as follows:

0. ISC$_0$-sw: $\mathbf{i_{n-1}i_{n-2}}..i_3i_2i_1i_0 \rightarrow i_{n-3}i_{n-4}..i_1i_0\mathbf{j_{n-1}j_{n-2}}$
1. ISC$_1$-sw: $\mathbf{i_{n-3}i_{n-4}}..i_1i_0j_{n-1}j_{n-2} \rightarrow i_{n-5}i_{n-6}..i_1i_0j_{n-1}j_{n-2}\mathbf{j_{n-3}j_{n-4}}$

...

$m-1$. ISC$_{m-1}$-sw: $\mathbf{i_1i_0}j_{n-1}j_{n-2}..j_3j_2 \rightarrow j_{n-1}j_{n-2}..j_3j_2\mathbf{j_1j_0}$.

Table 2
The advantages and limitations of ATAPE methods on MINs.

ATAPE partitioning and scheduling	MINs [5,14]	MINs$^+$ [8]	An $x \times x$ crossbar of MINs$^+$ [8]	An $x \times x$ crossbar of partitionable MINs$^+$
	$O(N + \log_2 N)$		$O(N + \log_2 N/x)$	$O(N/x + \log_2 N/x)$
	Switch-pattern ATAPE	ATAPE [8] $D = S \oplus (C + \text{order})\%N$ $D = \rho[(S + C + \text{order})\%N]$		pATAPE $D^y = S^y \oplus C^y_l$ $C^y_l = (yN/x + l)\ \%N$
Communication latency or stage-delay	n-stage	n-stage	n'-stage$< n$	n'-stage $< n$
Focus on MIN-switch	2×2	2×2	2×2	$d \times d$
Partitionable network	–	–	–	✓
Partitionable ATAPE	–	–	–	✓
Scheduling tasks	One	One	One	Many
Subsystem utilization	100%	100%	(1/x)100%	100%
CPI (clock per instr.)	1	1	1	$1/x$

The 4-nary flip network is constructed with the 2-bit circular right-shift ISC to connect between $m = \log_4 N$ stages, where the correct self-routing is defined as follows:

1. sw-ISC$_1$: $i_{n-1}i_{n-2}..i_3i_2\mathbf{i_1i_0} \rightarrow \mathbf{j_1j_0}i_{n-1}i_{n-2}..i_5i_4i_3i_2$
2. sw-ISC$_2$: $j_1j_0\mathbf{i_{n-1}i_{n-2}}..\mathbf{i_3i_2} \rightarrow \mathbf{j_3j_2}j_1j_0i_{n-1}i_{n-2}..i_5i_4$

...

m. sw-ISC$_m$: $j_{n-3}j_{n-4}..j_1j_0\mathbf{i_{n-1}i_{n-2}} \rightarrow \mathbf{j_{n-1}j_{n-2}}j_{n-3}j_{n-4}..j_3j_2j_1j_0$.

The 4-nary banyan$^+$ network uses the complex ISC patterns (ISC$_1$, ISC$_2$, ..., ISC$_s$, ..., ISC$_{m-1}$) to connect between m stages. For $s = 1, 2, \ldots, m-1$, the ISC$_s$ is the inverse pair-address of $2(s+1)$ bits. In our study, the last ISC$_m$ (the 2-bit circular right-shift operation) is included for post-port adjusting from stage $m-1$ to the outputs. For the banyan$^+$, the success self-routing from input $I = (i_{n-1}i_{n-2}\ldots i_2i_1i_0)$ to output $J = (j_{n-1}j_{n-2}\ldots j_2j_1j_0)$ is presented as follows:

1. sw-ISC$_1$: $i_{n-1}i_{n-2}..(i_3i_2)(\mathbf{i_1i_0}) \rightarrow i_{n-1}i_{n-2}..(\mathbf{j_1j_0})(i_3i_2)$
2. sw-ISC$_2$: $i_{n-1}i_{n-2}..(i_5i_4)j_1j_0(\mathbf{i_3i_2}) \rightarrow i_{n-1}i_{n-2}..(\mathbf{j_3j_2})j_1j_0(i_5i_4)$

...

$m-1$. sw-ISC$_{m-1}$: $(i_{n-1}i_{n-2})..j_1j_0(\mathbf{i_{n-3}i_{n-4}}) \rightarrow (\mathbf{j_{n-3}j_{n-4}})..j_1i_0(i_{n-1}i_{n-2})$

m. sw-ISC$_m$: $j_{n-3}j_{n-4}..j_1j_0\mathbf{i_{n-1}i_{n-2}} \rightarrow \mathbf{j_{n-1}j_{n-2}}j_{n-3}j_{n-4}..j_3j_2j_1j_0$.

In the 4-nary butterfly$^+$ network, the ISC$_s$ pattern is an inverse pair-address of $n - 2(s-1)$ bits, where $s = 1, 2, \ldots, m-1$. The butterfly$^+$ requires the pre-port adjusting ISC$_0$ (the 2-bit circular left-shift) for the following correct-routing.

0. ISC$_0$-sw $\mathbf{i_{n-1}i_{n-2}}i_{n-3}i_{n-4}..i_3i_2i_1i_0 \rightarrow i_{n-3}i_{n-4}..i_3i_2i_1i_0\mathbf{j_{n-1}j_{n-2}}$
1. ISC$_1$-sw: $(\mathbf{i_{n-3}i_{n-4}})..i_1i_0(j_{n-1}j_{n-2}) \rightarrow (j_{n-1}i_{n-2})..i_3i_2i_1i_0(\mathbf{j_{n-3}j_{n-4}})$
2. ISC$_2$-sw: $j_{n-1}j_{n-2}(\mathbf{i_{n-5}i_{n-6}})..i_1i_0(j_{n-3}j_{n-4}) \rightarrow j_{n-1}j_{n-2}(j_{n-3}j_{n-4})..i_1i_0(\mathbf{j_{n-5}j_{n-6}})$

...

$m-2$. ISC$_{m-2}$-sw: $j_{n-1}j_{n-2}..(\mathbf{i_3i_2})i_1i_0(j_5j_4) \rightarrow j_{n-1}j_{n-2}..(j_5j_4)i_1i_0(\mathbf{j_3j_2})$

$m-1$. ISC$_{m-1}$-sw: $j_{n-1}j_{n-2}..j_5j_4(\mathbf{i_1i_0})(j_3j_2) \rightarrow j_{n-1}j_{n-2}..j_5j_4(j_3j_2)(\mathbf{j_1j_0})$.

For the 4-nary baseline$^+$ network, the ISC patterns (ISC$_1$ – ISC$_{m-1}$) are designed recursively to connect between stages s-1 and s ($s = 1, 2, \ldots, m-1$). The ISC$_s$ pattern is the 2-bit circular right-shift (in $n - 2(s-1)$ bits) and the special ISC$_m$ (the reverse pair-addresses) is introduced for post-port adjusting. Then, the correct self-routing of the baseline$^+$ network is demonstrated as follows:

1. sw-ISC$_1$: $(i_{n-1}i_{n-2}..i_3i_2\mathbf{i_1i_0}) \rightarrow (\mathbf{j_1j_0}i_{n-1}i_{n-2}..i_5i_4i_3i_2)$
2. sw-ISC$_2$: $j_1j_0(i_{n-1}i_{n-2}..i_5i_4\mathbf{i_3i_2}) \rightarrow j_1j_0(\mathbf{j_3j_2}i_{n-1}i_{n-2}..i_5i_4)$

...

$m-1$. sw-ISC$_{m-1}$: $j_1j_0j_3j_2..(i_{n-1}i_{n-2}\mathbf{i_{n-3}i_{n-4}}) \rightarrow j_1j_0j_3j_2..(\mathbf{j_{n-3}j_{n-4}}i_{n-1}i_{n-2})$

m. sw-ISC$_m$: $(j_1j_0)(j_3j_2)..(j_{n-3}j_{n-4})(\mathbf{i_{n-1}i_{n-2}}) \rightarrow (\mathbf{j_{n-1}j_{n-2}})..(j_3j_2)(j_1j_0)$.

The 4-nary invert baseline$^+$ network is constructed with the ISC$_s$ pattern (the 2-bit circular left-shift in $2(s+1)$ bits, $s = 1, 2, \ldots, m-1$) and the special ISC$_0$ (the reverse pair-addresses) for pre-port adjusting. The correct routing is

0. ISC$_0$-sw$(\mathbf{i_{n-1}i_{n-2}})(i_{n-3}i_{n-4})..(i_3i_2)(i_1i_0) \rightarrow (i_1i_0)(i_3i_2)..(i_{n-3}i_{n-4})(\mathbf{j_{n-1}j_{n-2}})$
1. ISC$_1$-sw: $i_1i_0i_3i_2..(\mathbf{i_{n-3}i_{n-4}}j_{n-1}j_{n-2}) \rightarrow i_1i_0i_3i_2..(j_{n-1}j_{n-2}\mathbf{j_{n-3}j_{n-4}})$
2. ISC$_2$-sw: $i_1i_0i_3i_2..(\mathbf{i_{n-5}i_{n-6}}j_{n-1}j_{n-2}j_{n-3}j_{n-4}) \rightarrow i_1i_0i_3i_2..(j_{n-1}j_{n-2}j_{n-3}j_{n-4}\mathbf{j_{n-5}j_{n-6}})$

...

$m-2$. ISC$_{m-2}$-sw: $i_1i_0(\mathbf{i_3i_2})j_{n-1}j_{n-2}..j_5j_4) \rightarrow i_1i_0(j_{n-1}j_{n-2}..j_5j_4\mathbf{j_3j_2})$

$m-1$. ISC$_{m-1}$-sw: $(\mathbf{i_1i_0})j_{n-1}j_{n-2}..j_5j_4j_3j_2) \rightarrow (j_{n-1}j_{n-2}..j_5j_4j_3j_2\mathbf{j_1j_0})$.

For instance, Fig. 4 displays the topologies of MINs$^+$ with the derived pre-/post-port adjusting using 4×4 switches ($N = 64$, $m = \log_4 N = 3$ stages, $N/4 = 16$ switches per stage). In general, the port-adjusting and correct routing of each d-nary MIN$^+$ ($d = 2^k$) can be derived through each ISC and switch by the k-bit shift $(i_{k-1}..i_0)$ of each (n-bit) ISC and the k-bit switch $= j_{k-1}..j_0$, similar to the simplified case $d = 4$.

3.2. The PMINs$^+$ and I/O cross-control switches

The superb PMINs$^+$ (the $x \times x$ crossbar of partitionable MINs$^+$) are the renovation of the CMINs$^+$ with the external I/O cross-control switches to support multiple tasks of sizes $N'' = N/2^t$ ($\leq N$) on independent subsystems, which guarantees the fully 100% utilization. Fig. 5 presents the innovation and contribution of our study (i.e., the PMINs$^+$ and new partitionable ATAPE). For ATAPE applications on PMINs$^+$ with super-pipelining, the time complexity (for task size N) $O(N/x + n')$ outperforms $O(N + n')$ on CMINs$^+$ with pipelining, where a subsystem MIN$^+$ contains $N' = N/x$, $n' = \log_2 N'$ stages, and $N'/2$ switches per stage. On the new PMINs$^+$, we can achieve the processing of multiple tasks (of size $N'' = N/2^t$) by partitioning the network system into x sections ($y = 0, 1, 2, \ldots, x-1$) to operate simultaneously in only N''/x main steps (with x controls on x sections per step) and in $O(N/x + n')$ for task size N. For example, on a 2×2 crossbar of PMINs$^+$ (Fig. 6(a)), the ATAPE-task (size N) can be processed in $N/2$ main steps (2 rounds of controls (C) per step) on 2 partitions (or sections) of size N. Under super-pipelining, the first section ($y = 0$) is set with I/O switches 0-0 and 1-1 and another section ($y = 1$) with I/O switches 0-1 and 1-0, corresponding to the right-jumping control (C^y), derived in Eq. (1), for fruitfully embedding new partitionable ATAPE $D^y = S^y \oplus C^y$ in Eq. (2), where $y = 0, 1, \ldots, x-1$.

In this study, N rounds of the controls (C) are reordered with the right sequencing (i.e., $C = 0, N/2, 1, N/2+1, 2, N/2+$

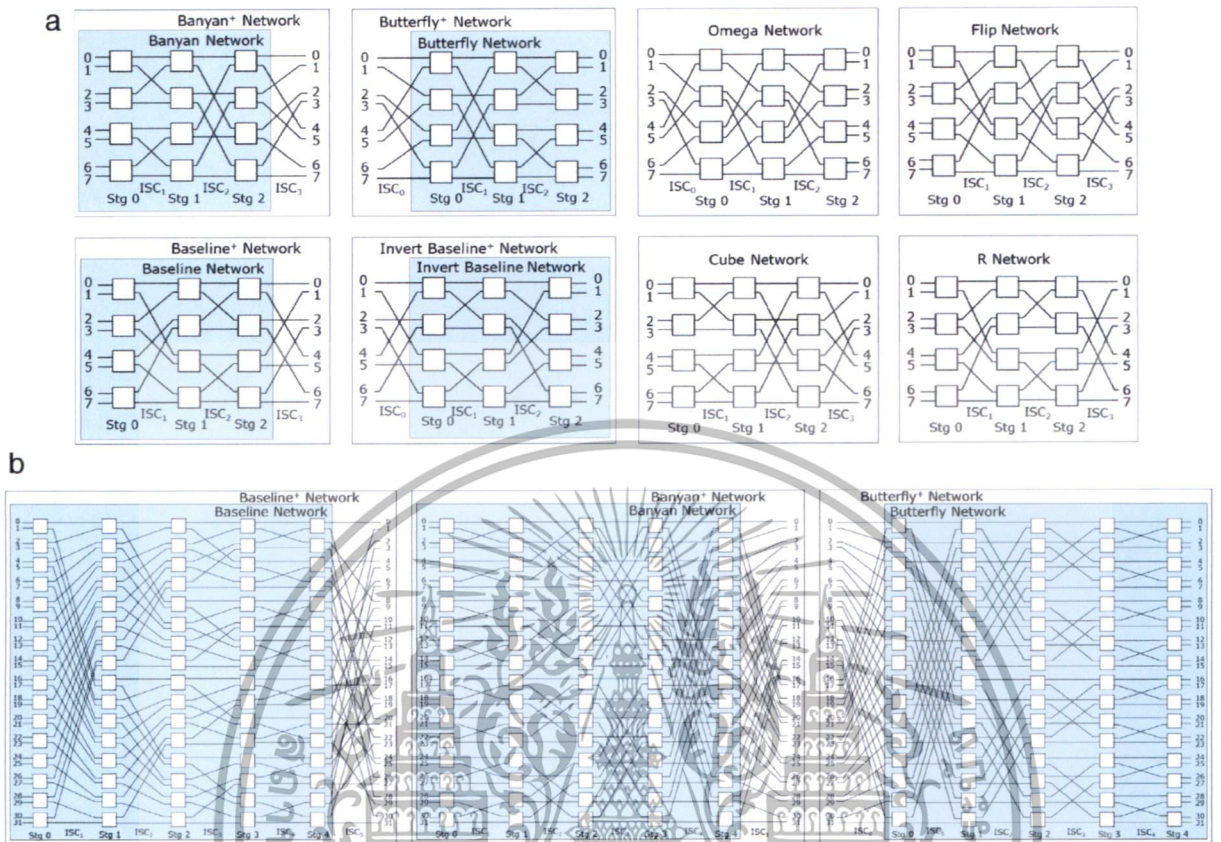


Fig. 3. Examples of the complete 2×2-switch MINs: (a) omega, flip, cube, R-network, butterfly$^+$, banyan$^+$, baseline$^+$, invert baseline$^+$ networks with $N = 8, d = 2$; and (b) baseline$^+$, banyan$^+$ and butterfly$^+$ networks with $N = 32, d = 2$.

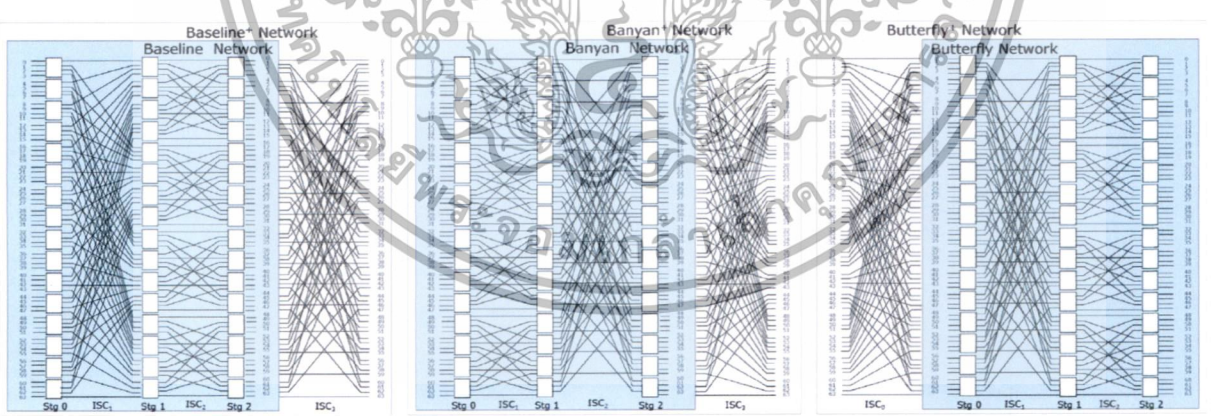


Fig. 4. Examples of the complete 4-nary MINs$^+$ such as baseline$^+$, banyan$^+$, and butterfly$^+$ networks with $N = 64, d = 4$.

$2, \ldots, N/2 - 1, N - 1$) on PMINs$^+$ in order to use different sections simultaneously by starting in every of $1/x$-clock (for 100% utilization), proven in Theorems 3.1–3.2. Unlike processing on CMINs$^+$, incrementing controls $C = 0, 1, 2, 3, \ldots, N/2 - 1$ will be allocated on section 0 (0-0 and 1-1) only and the remaining $C = N/2, N/2+1, N/2+2, \ldots, N-1$ will be assigned to section 1 (0-1 and 1-0) repeatedly while section 0 is idle. On the new PMINs$^+$, those two sections can be utilized alternately without the idle section for executing the ATAPE task. Moreover, for task sizes $N/2$ (Fig. 6(b)), 2 tasks can be started at the same time and executed on 2 distinct sub-MINs$^+$ in the same section.

Fig. 6(c) depicts the general structure of the $x \times x$ crossbar of PMINs$^+$ (of size N), which composes of x^2 MIN$^+$ subsystems (of sizes $N' = N/x$) for executing several tasks (of sizes $N'' = N, N/2, \ldots, N/2^t, \ldots,$ or N/x) using sub-system allocation in a clockwise direction. Fig. 7 shows an example of setting 4 sections (each of size N) on the PMIN$^+$ ($x = 4$) to operate multiple instructions of controls C independently. With super-pipelining (degree 4), in step 1 the controls ($C = 0, N/4, 2N/4, 3N/4$) of the ATAPE task (N) can be executed on 4 different sections simultaneously with 100% utilization since all x^2 sub-MINs$^+$ are used to set different connections among N processors per clock.

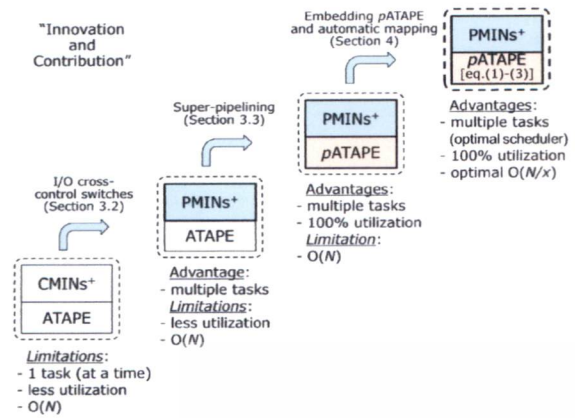


Fig. 5. A big picture of the innovation and contribution of our study.

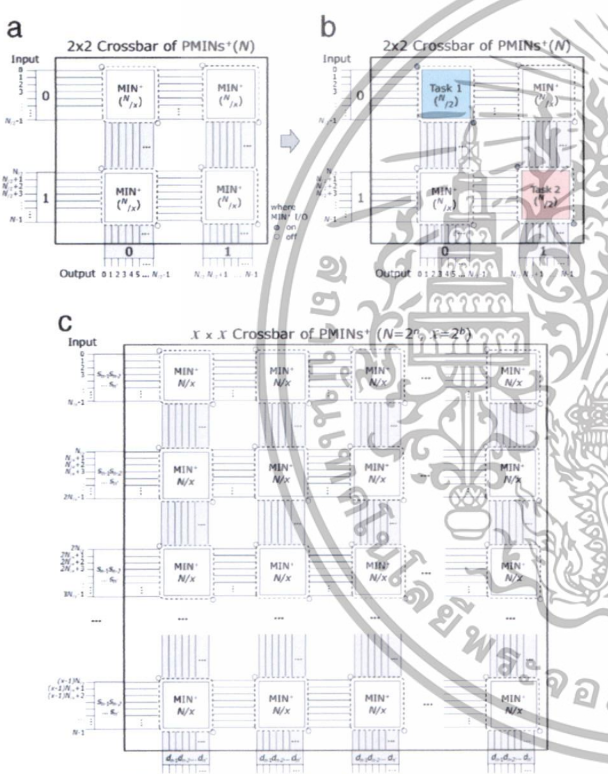


Fig. 6. (a) A 2×2 crossbar of PMINs$^+$ (N); (b) I/O switch setting for 2 tasks (of $N/2$); and (c) the general structure of the $x \times x$ crossbar of PMINs$^+$ with $N = 2^n, x = 2^b$ and sub-MINs$^+$ ($N' = N/x, n' = \log_2 N/x$).

Next, the same process is repeated for step 2 ($C = 1, N/4 + 1, 2N/4 + 1, 3N/4 + 1$) up to step N/x($C = N/4 - 1, 2N/4 - 1, 3N/4 - 1, N - 1$).

Theorem 3.1. *Automatic setting the I/O cross-control switches of subsystems incorporated with the super-pipelining for* CPI $= 1/x$ *can partition the* PMINs$^+$ *into x sections for ATAPE tasks of size N. Each section y is allocated in a clockwise direction by the right-jumping control C^y (of ATAPE) to set in/out ports of S and D within $l = N/x$ steps (x rounds per step), as follows:*

$$C_l^y = \left(\frac{yN}{x} + l\right) \bmod N \qquad (1)$$

with the (external) O-switch $=$ (I-switch $+ y$) mod x, where the (external) I-switch $= 0, 1, \ldots, x-1$; $0 \le y < x$ and $l = 0, 1, 2, \ldots, N/x - 1$ (for one task), …, $N - 1$ (for x tasks).

Proof. Let the $x \times x$ crossbar of PMINs$^+$ ($x = 2^b, x^2 < N$) be partitioned into x sections with the clockwise direction, called section y (=0, 1, 2, …, $x - 1$). Each of the I/O cross-control switches is set automatically by the b high-order bits of $S(i_{n-1}..i_{n-b})$ for setting the I-switch and the b high-order bits of $D = S \oplus C_l^y$ for setting the O-switch (or $O = (I + y) \bmod x$). Then, all unique (S, D) connections of this *XOR* operation can be computed by Eq. (1) for scheduling x sections independently. The right-jumping control C_l^y is derived to accomplish the triple-right scheduling (right task, right section, right time), as follows: First, we assign the right section to the right task with $C^y = yN/x$ on section $y = 0, 1, 2, \ldots, x - 1$ for x controls of tasks. Second, the right sequencing of the ATAPE task for other sets of x controls is $C_l^y = yN/x + l$ with the on-chip incrementing counter $l = 0, 1, 2, 3, \ldots, N/x - 1$. Thus,

Sec 0 → C_l^0: {0, 1, 2, 3, …, $N/x - 1$} by I/O switches 0-0, 1-1, 2-2, …, $(x-1)-(x-1)$

Sec 1 → C_l^1: {N/x, $N/x + 1$, $N/x + 2$, …, $2N/x - 1$} by I/O switches 0-1, 1-2, 2-3, …, $(x-1)-0$

Sec 2 → C_l^2: {$2N/x$, $2N/x + 1$, $2N/x + 2$, …, $3N/x - 1$} by I/O switches 0-2, 1-3, 2-4, …, $(x-1)-1$

…

Sec $x-1$ → C_l^{x-1}: {$(x-1)N/x$, $(x-1)N/x + 1$, $(x-1)N/x + 2$, …, $N-1$} by I/O switches $0-(x-1)$, 1-0, 2-1, …, $(x-1)-(x-2)$.

Next, in order to apply the super-pipelining efficiently, we assign the proper control C_l^y to activate each of x successive sections ($y = 0, 1, \ldots, x-1$) in every $1/x$ clock (to achieve CPI $= 1/x$), while the same section y is used for the connection (for the next round l) in every clock. Thus, the first x rounds ($C_l^0 = 0, C_l^1 = N/x, C_l^2 = 2N/x, \ldots, C_l^{x-1} = (x-1)N/x$) can be operated simultaneously in all sections (by starting $1/x$ clock per C). In section 0, the I/O-switch setting is 0-0 (or $0..00_2$–$0..00_2$), 1-1 (or $0..01_2$–$0..01_2$), …, $(x-1)-(x-1)$ (or $1..11_2$–$1..11_2$), according to the b high-order bits of N sources S (=0, 1, 2, …, $N-1$) and N destinations $D = S \oplus C_l^y$. In section 1, the I/O-switch setting is 0-1 (or $0..00_2 - 0..01_2$), 1-2 (or $0..01_2$–$0..10_2$), …, $(x-1)-0$ (or $1..11_2$–$0..00_2$). In section 2, the I/O-switch setting is 0-2 (or $0..00_2 - 0..10_2$), 1-3 (or $0..01_2 - 0..11_2$), …, $(x-1)-1$ (or $1..11_2 - 0..01_2$), …, and lastly in section $x-1$, the I/O-switch setting is $0-(x-1)$ (or $0..00_2$–$1..11_2$), 1-0 (or $0..01_2$–$0..00_2$), …, $(x-1)-(x-2)$ (or $1..11_2$–$1..10_2$). Similarly, the next x rounds ($C_l^0 = 1, C_l^1 = N/x + 1, C_l^2 = 2N/x + 1, \ldots, C_l^{x-1} = (x-1)N/x + 1$) are processed on section 0 to $x - 1$. The same process is repeated until the last x rounds ($C_l^0 = N/x-1, C_l^1 = 2N/x-1, C_l^2 = 3N/x-1, \ldots, C_l^{x-1} = N-1$) are operated on sections 0 to $x - 1$.

Thus, N personalized messages in all sections are sent to all destinations in N/x steps (N rounds per task and x rounds per step) by using the control C_l^y where $0 \le y < x$ and $l = 0, 1, 2, \ldots, N/x-1$ (for a task) and $N - 1$ (for x tasks). □

For example, suppose a 4×4 crossbar of PMINs$^+$ ($N = 32, x = 4$) with MIN$^+$ subsystems ($N' = N/4 = 8, n' = \log_2 N' = 3$ stages) can be partitioned into 4 sections (see Fig. 7), where the sequencing of $N/4$ controls (C^y) of each section can be incremented for the ATAPE task of size N, as follows:

Sec 0 ($y = 0$, sky blue line) with C_l^0: {0, 1, 2, 3, 4, …, 7}
Sec 1 ($y = 1$, red line) with C_l^1: {8, 9, 10, 11, …, 15}
Sec 2 ($y = 2$, orange line) with C_l^2: {16, 17, 18, 19, …, 23}
Sec 3 ($y = 3$, blue line) with C_l^3: {24, 25, 26, 27, …, 31}.

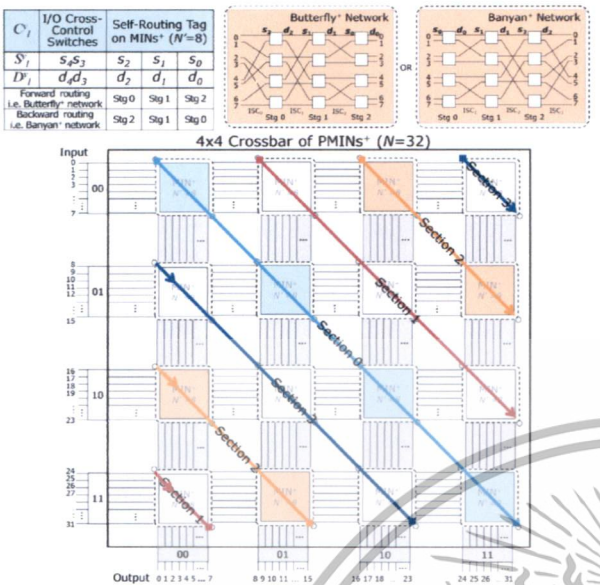


Fig. 7. Four sections on a 4×4 crossbar of PMINs$^+$ with $N = 32$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

For instance, in section 2 the 2-high order bits of S and D ($S^{y=2}$: s_4s_3, $D^{y=2}$: d_4d_3) are used to set the I/O switches $00_2 - 10_2$, $01_2 - 11_2$, $10_2 - 00_2$ and $11_2 - 01_2$. For the (S, D) routing on each MIN$^+$ subsystem, the remaining 3-bits ($s_2s_1s_0$ and $d_2d_1d_0$) are used for setting switches along the local stages 0 to 2 (on the forward MIN$^+$) or the stages 2 back to 0 (on the backward MIN$^+$). In all sections, the messages of all destinations ($D^{y=2}$) can be received from all sources ($S^{y=2}$) by passing throughout subsystems in N/x rounds ($l = 0, 1, \ldots, N/x - 1$). To complete the ATAPE in N rounds, sections 0–3 must be operated simultaneously using the super-pipelining (degree x) for CPI $= 1/x$.

Similarly, for executing the maximum ($x = 4$) ATAPE tasks of size $N = 32$ on sections 0–3, the N rounds of C_l^y for $l = 0, 1, 2, \ldots, N - 1$ are required, where each of x tasks can be processed simultaneously in x sections, as follows:

Sec 0 (sky blue line) with C_l^0(of T_0): {0, 1, 2, 3, 4, . . ., 31}
Sec 1 (red line) with C_l^1(of T_1): {8, 9, 10, . . ., 31, 0, . . ., 7}
Sec 2 (orange line) with C_l^2(of T_2): {16, 17, . . ., 31, 0, . . ., 15}
Sec 3 (blue line) with C_l^3(of T_3): {24, 25, .., 31, 0, .., 23}.

For example, $C_l^0 = 0$(of T_0), $C_l^1 = 8$ (of T_1), $C_l^2 = 16$ (of T_2), and $C_l^3 = 24$ (of T_3) are executed simultaneously on sections 0–3. The same process is repeated for the next 4 controls ($C^y = 1(T_0), N/4 + 1 = 9(T_1), 2N/4 + 1 = 17(T_2), 3N/4 + 1 = 25(T_3)$), . . . , until the last 4 controls ($C^y = N-1 = 31(T_0), N/4-1 = 7(T_1), 2N/4-1 = 15(T_2), 3N/4-1 = 23(T_3)$) to complete those 4 tasks.

3.3. The PMINs$^+$ and triple-right scheduling

On the $x \times x$ crossbar of PMINs$^+$, multiple ATAPE tasks (p) of size $N/2^t$ (where $t = 0, 1, 2, \ldots, \log_2 x$) can be operated on different sections by using the super-pipelining (degree x) in order to allocate the right task sequencing on the right section in efficient (right) time.

Let $T_{i,C}$ denote the ATAPE task i (in round C), where each T_i is processed by N rounds of the controls (C) and the regular order of $C = 0, 1, 2, 3, \ldots, N - 1$. Hence, a sequence of N subtasks (T_i, C) of the ATAPE task T_i is $T_{i,0}, T_{i,1}, T_{i,2}, T_{i,3}, \ldots, T_{i,N-1}$, where $T_{i,j}$ is started in every clock when $T_{i,j-1}$ is completed and $T_{i,j}$ is completed before $T_{i,j+1}$ is starting, as shown in Fig. 8(a). For executing the ATAPE task T_i on a 2×2 crossbar of PMINs$^+$ efficiently, two sections (Fig. 8(b)) can be used simultaneously in $N/2$ steps with super-pipelining (degree x). The right sequencing of N controls on 2 successive sections with super-pipelining (degree 2) are processed as follows: section 0 is set by I/O switches 0-0 and 1-1 for subtasks $T_{i,0}, T_{i,1}, T_{i,2}, T_{i,3}, \ldots, T_{i,N/2-1}$ and section 1 is set by I/O switches 0-1 and 1-0 for subtasks $T_{i,N/2}, T_{i,N/2+1}, T_{i,N/2+2}, \ldots, T_{i,N-1}$. Similarly, for operating two ATAPE tasks (T_0, T_1) of size N (Fig. 8(c)), the sequencing of N controls of each T_i on two sections is allocated, as follows: section 0 (0-0 and 1-1) for subtasks $T_{0,0}, T_{0,1}, \ldots, T_{0,N-1}$ and section 1 (0-1 and 1-0) for subtasks $T_{1,N/2}, T_{1,N/2+1}, \ldots, T_{1,N-1}, T_{1,0}, T_{1,1}, \ldots, T_{1,N/2-1}$. Fig. 8(d) shows the right sequencing of N'' subtasks of each T_i(T_0, T_1, T_2, T_3) of size $N'' = N/2$ on two sections (2 tasks per section). Lastly, for the 4×4 crossbar of PMINs$^+$ Fig. 8(e) displays the right sequencing of one task (size N) on 4 sections of super-pipelining (degree 4).

Under the triple-right scheduling (right task, right section, right time), Fig. 9(a) displays subsystem allocation on section 0 and section 1 of two tasks (size N) in Fig. 8(c) and its efficient timing diagram in Fig. 9(c), where for every 1/2 clock each of two sections is enabled alternately. With super-pipelining (degree 2), each of two sections is able to execute each of $N/2$ rounds through n' ($= \log_2 N/x$) stages of sub-MIN$^+$ in parallel in $O(N/2 + n')$ for one task (size N) since round 0 ($C = 0$) needs n' clocks and others ($N - 1$ rounds of controls $C = 1, 2, \ldots, N - 1$) need $(N - 1)/2$ clocks.

Theorem 3.2. *On the $x \times x$ crossbar of* PMINs$^+$ *with super-pipelining, the maximum number of allocated ATAPE tasks (size $N'' = N/2^t$) are $p = 2^{2t}$ tasks in $O(N/x + n')$, where $1 \le 2^t \le x$.*

Proof. For scheduling multiple tasks ($T_0, T_1, T_2, \ldots, T_{p-1}$) on PMINs$^+$, the I/O cross-control switches are enabled for using successive sections y (=0, 1, 2, . . ., $x - 1$) in every $1/x$ clock. There are $p = 2^{2t}$ tasks (size $N'' = N, N/2 \ldots, N/2^t, \ldots$, or N/x) on x sections (2^t tasks per section) for utilizing the system efficiently within $N/x + n'$ clocks (see Table 3) since a Latin-square (LS) size on PMINs$^+$ is $N \times N$ and a task size $N'' = N/2^t$ requires a local LS size $N'' \times N''$. Thus, the system can be allocated $N \times N/(N'' \times N'') = 2^t \times 2^t$ tasks.

Case 1: For executing an ATAPE task (of size N), the N/x rounds per task are assigned to each section y (of all x sections) with super-pipelining (degree x) by starting each section in every $1/x$ clock. For instance, Figs. 8–9(e) show the triple-right scheduling of N controls (C) of a task on 4 successive sections of the 4×4-cross PMIN$^+$.

Case 2: For task size $N'' = N/2^t$; $2 \le 2^t \le x$, the maximum $p = 2^{2t}$ tasks, where 2^t tasks can be allocated in each section y, and hence all $p = 2^{2t}$ tasks in all x sections are enabled simultaneously (see Figs. 8–9(d) for $x = 2$, $N'' = N/2$). □

In Figs. 8–9(d), 4 tasks (of size $N/2$) are allocated on the 2×2-cross PMIN$^+$ (Fig. 9(b)) or tasks T_0, T_1 in section 0 (0-0, 1-1) and T_2, T_3 in section 1 (0-1, 1-0) with super-pipeline scheduling (i.e., every 1/2 clock will start next section and in each section every clock can process next control C through n' stages). Fig. 9(e) shows another example of executing an ATAPE task (size N) on the 4×4-cross PMIN$^+$ (Fig. 7) in $N/4 + n' - 1/4$ clocks since the 1st round of C needs n' clocks and others ($N - 1$ rounds with 1/4 clock per C) require $(N - 1)/4$ clocks.

Table 3
Time complexity and #tasks (size $N'' = N/2^t$) on PMINs$^+$ (size N).

Task(s) ($N'' = N/2^t$, $1 \le 2^t \le x$)	The $x \times x$-cross PMINs$^+$		The $x \times x$-cross MINs$^+$ [8]	
	Time complexity	#tasks	Time complexity	#task
One T (size N)	$O(N/x + n')$	1	$O(N + n')$	1
pTs (size $N/2$)	$O(N/x + n')$	$2^{2t} = 4$	$O(N/2 + n')$	1
pTs (size $N/4$)	$O(N/x + n')$	$2^{2t} = 16$	$O(N/4 + n')$	1
...	...	...	...	1
pTs (size N/x)	$O(N/x + n')$	$2^{2t} = x^2$	$O(N/x + n')$	1

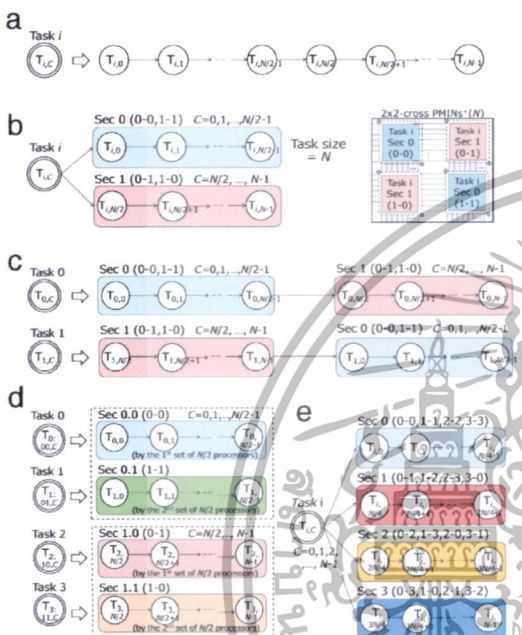


Fig. 8. Examples of task sequencing: (a) a regular task sequencing; (b) the right task sequencing on a 2×2-cross PMIN$^+$; (c) 2 tasks (size N); (d) 4 tasks (size $N/2$); and (e) one task (size N) on a 4×4-cross PMIN$^+$.

4. The optimal VA-DE (valuable ATAPE with dynamic embedding) algorithm

For general embedding the partitionable ATAPE on the PMINs$^+$, Section 4.1 presents the generalized d-switch patterns. Section 4.2 introduces the flexible and efficient Latin square partitioning for multiple tasks on PMINs$^+$. Finally, the optimal VA-DE (valuable ATAPE with dynamic embedding) algorithm is proposed in Section 4.3.

4.1. The generalized d-switch patterns

The advantages of the *XOR*-operation on the PMINs$^+$ are: (1) The switch-/stage-control routing of the partitionable ATAPE is the same as the (S, D) routing; (2) The triple-right scheduling is achieved by the right task sequencing (of controls C) on all sections for 100% utilization. Moreover, using the *XOR*-operation, the local switch patterns can be set by using a k-bit partition of the global n-bit $D = S \oplus C(d_{n-1} = s_{n-1} \oplus c_{n-1}, \ldots, d_2 = s_2 \oplus c_2, d_1 = s_1 \oplus c_1, d_0 = s_0 \oplus c_0)$ that can be operated only once and any partition of $(d_{k-1} \ldots d_1 d_0) = (j_{k-1} \ldots j_1 j_0)$ can be used without the local *XOR*-operation $J = I \oplus P$ in each switch. Fig. 10 displays the configurations of the d switch-patterns for the switch-/stage-control routing.

4.2. Latin-square partitioning for multiple ATAPE-tasks

On the PMINs$^+$, the local and global addressing of (S, D) communication for task-sizes $N'' = N/2^t$ are defined by using n global bits of S (sources) and D (destinations) for (S, D) connections, as follows: In each sub-MIN$^+$ (with $N' = N/x = 2^{n'}$), the n' low-order bits are used for self-routing across n' local stages, as demonstrated in Fig. 11. Let S^y and D^y denote (S, D) connections in section y (see Theorem 3.1) on the $x \times x$ crossbar of PMINs$^+$. The n-bits global addressing of sources S^y: $s_{n-1}s_{n-2} \ldots s_2 s_1 s_0$ and destinations D^y: $d_{n-1}d_{n-2} \ldots d_2 d_1 d_0$ are divided into two parts for the I/O cross-control setting (b bits) and the local MIN$^+$ routing ($n - b$ bits).

(1) External I/O cross-control switches (where O-switch = (I-switch + y) mod x) use b high-order bits of S^y and D^y for communication in section y. For an ATAPE task (size N), each section y (containing x sub-MINs$^+$) can execute in parallel with b-bit I/O switch setting of S^y (for I-row enabled) and D^y (for O-column enabled). In addition, for multiple tasks of size $N'' = N/2^t$ (2^t tasks per section), t high-order bits $(s_{n-1} \ldots, s_{n-t})$ of S^y are used to identify those certain tasks (size N'') in each section.

(2) Routing on sub-MIN$^+$ will be processed by n' low-order bits across n' stages of d-nary switches ($n' = n - b$, $d = 2^k$), k bits per switch for the forward/backward routing.

For Latin-square partitioning of the partitionable ATAPE (see Fig. 12), let (S^y, D^y) connections in section y on PMINs$^+$ be defined and operated automatically by a sequence of controls $(C_0^y, C_1^y, C_2^y, \ldots, C_l^y)$ for destinations (to send/receive N_{All}^y messages) in section y.

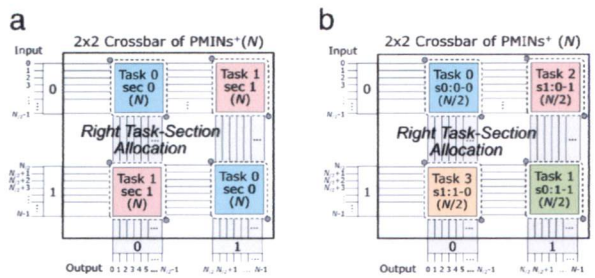
$$D^y = S^y \oplus C_l^y \quad \text{and} \quad C_l^y = (yN/x + l) \bmod N \quad \text{where } l = 0, 1, 2, \ldots, <N. \tag{2}$$

Each sub-system in section y can use n' low-order bits of S^y and D^y in Eq. (2). All sources have a unique path (between 0 to $N - 1$), through which can be sent N_{All}^y personalized messages to all destinations in N/x steps (for N rounds).

For example, on a 4×4-cross PMIN$^+$($N = 32$) the Latin-square partitioning is illustrated in Fig. 12 to be able to operate on four sections ($y = 0(00_2), 1(01_2), 2(10_2), 3(11_2)$) for many tasks of sizes $N'' = N/2^t$, where the controls C_l^y of each section y ($l = 0, 1, \ldots, N/4 - 1$ such as $y = 0$, $C_l^y = 0 - 7$, etc.).

Fig. 12(a) demonstrates N/x Latin-square partitioning for a task of size $N'' = 32(x = 4)$, where all n bits of the global addressing are used in x different sections ($y = 0, 1, 2, 3$) simultaneously for $l = 0, 1, 2, \ldots, N/4 - 1$. Fig. 12(b)–(c) display examples ($N'' = N/2$ and $N/4$) of the n''-bit local- and n-bit global-addressing for task sizes $N'' = N/2^t$ allocated on each section y (=0, 1, 2, ..., $x - 1$) with 2^t tasks per section, where each task needs $n'' = n - t$ bits for defining the local addressing $lS^y = S^y \bmod N''$ and $lD^y = D^y \bmod N''$.

Fig. 13 displays another example of a 2×2 crossbar of PMINs$^+$ ($N = 16, N' = 8, x = 2$) for assigning a certain task 1 of size $N'' = N$ (such as matrix transpositions (A^T) with $N = 16$) set by I/O-switches 0-1 and 1-0 of section $y = 1$. In



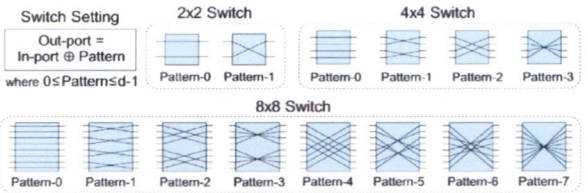
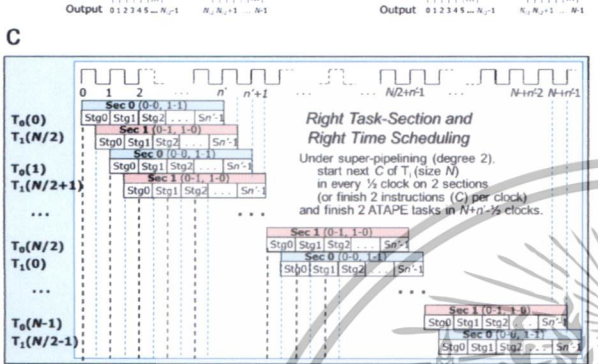
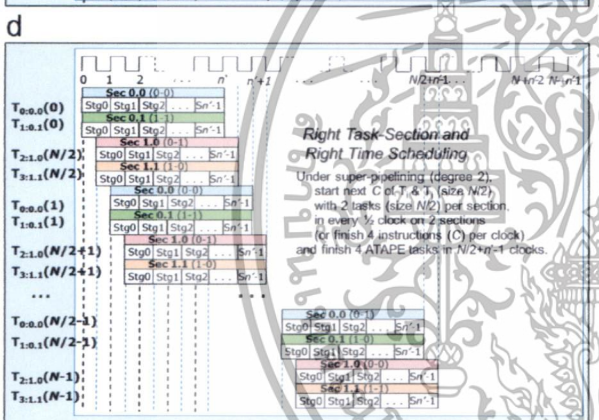


Fig. 10. Switch patterns of the *XOR*-operation ($d = 2$, 4, and 8).





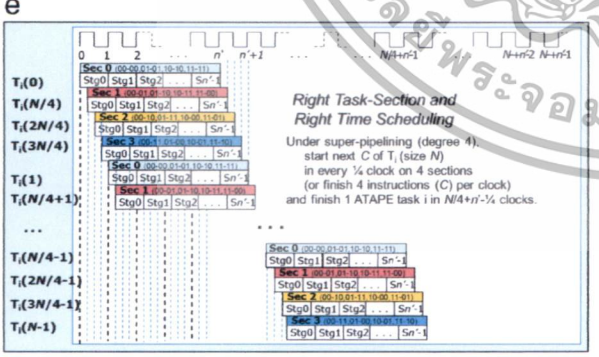


Fig. 9. A 2×2-cross PMIN$^+$ with ATAPE allocation: (a) 2 tasks (size N); (b) 4 tasks (size $N/2$); (c) timing diagram of 2 tasks (sizes N); (d) 4 tasks (sizes $N/2$); and (e) one ATAPE task (sizes N) on a 4×4-cross PMIN$^+$.

this special case, the global addresses of all (S^1, D^1) connections are presented in the Latin square ($N = 16$). For instance, with $l = 4$ and $C_4^1 = 12$, the global connections from all N sources are [12, 13, 14, 15, 8, 9, 10, 11, 4, 5, 6, 7, 0, 1, 2, 3] on two subsystems, as follows:

0-1 (I/O-switch) for MIN$^+${$0 \rightarrow 12, 1 \rightarrow 13, 2 \rightarrow 14, \ldots, 7 \rightarrow 11$}

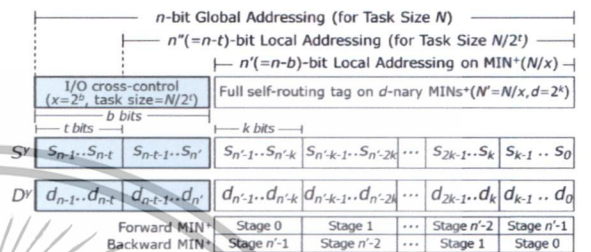


Fig. 11. Global and local addressing of the ATAPE communication on the $x \times x$ crossbar of PMINs$^+$ with d-nary MIN$^+$ subsystems.

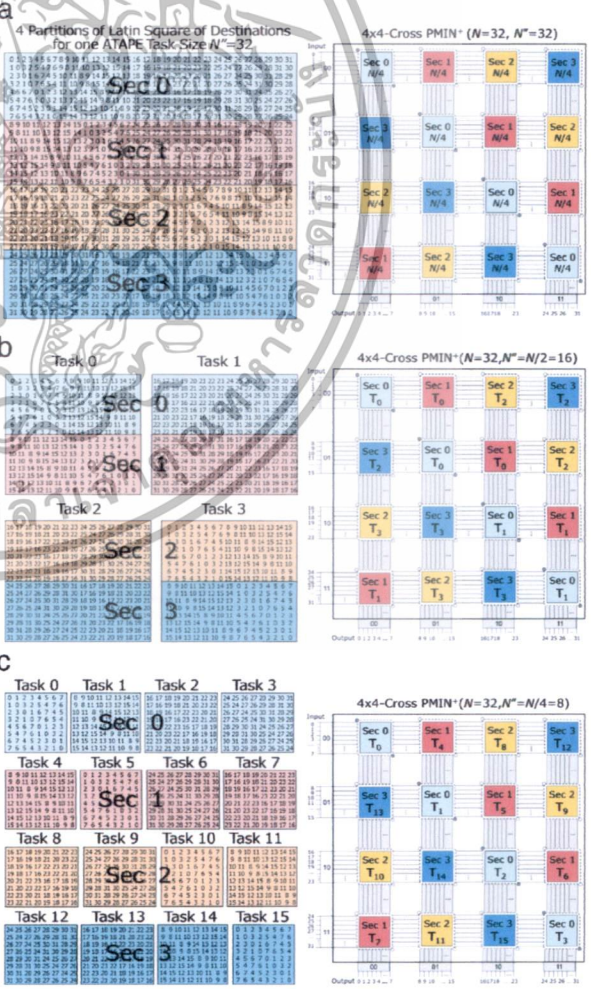


Fig. 12. Latin-square partitioning for task sizes $N/2^t$ on a 4×4-cross PMIN$^+$: (a) one task ($N'' = 32$); (b) 4 tasks ($N'' = 16$); and (c) 16 tasks ($N'' = 8$).

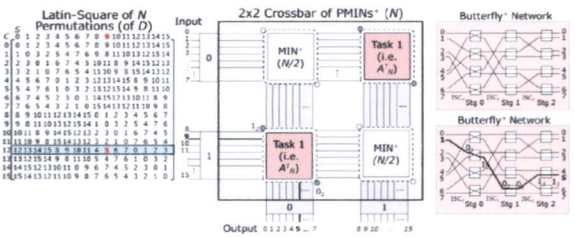


Fig. 13. An example of (S, D) self-routing of a task ($N'' = 16$) on a 2×2-cross PMIN$^+$ ($N = 16, N' = 8, x = 2$).

1-0 (I/O-switch) for MIN$^+$ $\{8 \to 4, 9 \to 5, 10 \to 6, \ldots, 15 \to 3\}$.

In particular, the local vs. global addressing of the (S, D) routing on each subsystem is presented from the source $S^1 = 9(1001_2)$ with the control $C_4^1 = 12(1100_2)$ in section 1 ($y = 1$) to connect to the destination $D^1 = S^1 \oplus C^1 = 9(1001_2) \oplus 12(1100_2) = 5(0101_2)$. Thus this I/O cross-connection is set by using 1-high order bit ($s_3 = 1, d_3 = 0$) of those S and D.

4.3. The optimal VA-DE Algorithm on the PMINs$^+$

This section presents the partitionable ATAPE (pATAPE), called the optimal VA-DE (valuable ATAPE with dynamic embedding) by the triple-right scheduling of a number of tasks $(T_0, T_1, \ldots, T_{nT-1})$ of various sizes $N'' = N/2^t$ on the $x \times x$ crossbar of PMINs$^+$. In Algorithm 4.1, a task of size N can be executed by super-pipelining (degree x) on all sections in optimal $O(N/x + n')$ or $O(N/x)$, proven in Theorem 4.1, that outperforms $O(N + n')$ or $O(N)$ on CMINs$^+$ [8]. For the optimal scheduling, we represent the new local vs. global (S, D) mapping in Eq. (3) to allocate 2^t tasks of size $N'' = N/2^t$ efficiently on all sections in optimal $O(N''/x + n')$.

$$lS^y = S^y \bmod N'' \quad \text{and} \quad lD^y = (D^y \bmod N'') \oplus c^z$$
$$\text{where } c^z = zN''/x,\ N'' = N/2^t,\ \text{and } z = 0, 1, 2, \ldots, 2^t - 1. \quad (3)$$

Theorem 4.1. *The time complexity of executing 2^t ATAPE tasks of size $N'' = N/2^t$; $t \le \log_2 x$ (2^t tasks per section) on the $x \times x$ crossbar of PMIN$^+(N, N' = N/x)$ is optimal $O(N''/x + n')$ with super pipelining (degree x) on x successive sections.*

Proof. For ATAPE communication of a task (size N) on the PMIN$^+$, each section y of all x sections is enabled to start x controls (in each of N/x steps) in one clock.

- In the first step ($l = 0$), each of x controls $C_0^0 = 0, C_0^1 = N/x, \ldots, C_0^{x-1} = (x-1)N/x$ can start in every of $1/x$ clock (on sections $y = 0$ to $x-1$) to send x sets of messages (N_{All}^y) from all N sources S^y through n' stages of subMINs$^+$ to all N destinations $D^y = S^y \oplus C_0^y$.
- In the second step ($l = 1$), the next x controls $C_1^0 = 1, C_1^1 = N/x + 1, \ldots, C_1^{x-1} = (x-1)N/x + 1$ will be started in every of $1/x$ clock to send the next x sets of messages.
- The same processes are repeated for others $l = 2, 3, \ldots, N/x - 1$. Therefore, in the last step $l = N/x - 1$, the last x controls $C_{N/x}^0 = N/x - 1, C_{N/x}^1 = 2N/x - 1, \ldots, C_{N/x}^{x-1} = N - 1$ will be started in $1/x$ clock to send the last x sets of messages.

Clearly, time complexity of executing a task of size N is $O(N/x + n')$ since the first control $C = 0$ requires n' clocks and the remaining $(N-1)$ controls need $(N-1)/x$ clocks. Similarly, for 2^t tasks (of size $N'' = N/2^t$)x global controls ($l = 0, 1, \ldots, N''/x - 1$ steps) are processed simultaneously in $N''/x + n' - 1$ clocks (2^t local controls per section), and hence $O(N''/x + n')$, which is optimal, see Fig. 14(a)–(b) for $N'' = N/2$. In this case ($N'' < N$), the efficient local vs. global mapping in Eq. (3) is applied (Fig. 15(a)) for the optimal scheduling. □

Algorithm 4.1. Optimal VA-DE (valuable ATAPE with dynamic embedding) by the triple-right scheduling on the PMINs$^+$.

```
Network System: an x x x crossbar of PMINs+ (N = 2^n, x = 2^k) and
                MIN+ subsystem (N' = 2^n' ; n' = log2N' stages).
ATAPE Task: task size N'' = N/2^t (i.e., N, N/2, ..., N/x); n'' = n-t.
Task Scheduler: a sequence of many tasks (T0,T1,T2,..., TnT).
-------------------------------------------------------------
For 2^t tasks (size N''=N/2^t) with 2^t local controls per section
  For all sections y = 0 to x-1
        (using I/O-switches: I = 0,1,...,x-1; O = (I+y) mod x)
    Call pATAPE(y) with super-pipelining degree x
        (every 1/x clock can start the next successive section y,
         and hence the same section can start in every clock.)
  end For all y
end For 2^t tasks.

Function pATAPE(y) begin
  For l = 0 to N''/ x - 1
        (with multistage super-pipelining (degree x))
    For all processors S (0 ≤ S < N) do in parallel
        C_l^y = yN/x+ l
        D^y = S^y ⊕ C_l^y
        (all local source processors lS^y = S^y mod N'' send their
         personalized massages to local destination processors
         lD^y = (D^y mod N'') ⊕ c^z; where c^z = zN''/x, z = 0,1,..,2^t-1)
    end For all processors
  end For l
end Function.
```

For example, Fig. 14(a)–(b) show the right task-sequencing and scheduling on the right section for 2 tasks (size $N'' = N/2$) on the PMIN$^+$ ($x = 4$) with $l = 0, 1, \ldots, N''/4 - 1$. Fig. 15(a) displays the global Latin-square partitioning of any control C_l^y for executing two local rounds (per section) in four sections y (=0, 1, 2, 3) by applying Eq. (3). For $N'' = N/2$, the local mapping is $lS^y = S^y \bmod N''$ and $lD^y = (D^y \bmod N'') \oplus c^z$, where $c^z = zN''/x$. In this case, there are $N''/x = 4$ global rounds ($l = 0, 1, 2, 3$) and $N/N'' = 2$ local rounds ($z = 0, 1$) per section. For instance, one global permutation [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 | 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31] represents two local permutations [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15] and [4 5 6 7 0 1 2 3 12 13 14 15 8 9 10 11]. Similarly, for $N'' = N/4$ (Fig. 15(b)), there are $N''/x = 2$ global rounds ($l = 0, 1$) and $N/N'' = 4$ local rounds ($z = 0, 1, 2, 3$) per section, see also the complete pATAPE results in Fig. 19 ($N = 32, N'' = 8, x = 4$).

Lastly, for scheduling a number of tasks on PMINs$^+$, let $T_0, T_1, T_2, T_3, \ldots, T_{nT-1}$ be a sequence of nT ATAPE tasks (sizes N/x to N). After sorting, the ATAPE-task scheduler will allocate the largest task first, where a task (size N) is processed in $N/x + n' - 1/x$ clocks and two tasks (sizes $N/2$) can be assigned per section at a time in $N/(2x) + n' - 1$ clocks. Similarly, following tasks can be allocated efficiently: four of tasks (sizes $N/4$) in $N/(4x) + n' - 1$ clocks, 2^t of tasks (sizes $N/2^t$) in $N/(2^t x) + n' - 1$ clocks, and x tasks (sizes N/x) in $N/(x^2) + n' - 1$ clocks. For example, Fig. 14(c) shows two main functions of the task-scheduler and the optimal scheduling of ten tasks $T_0(N), T_1(N/2), T_2(N/4), T_3(N/2), T_4(N), T_5(N/4), T_6(N/2), T_7(N/4), T_8(N/2), T_9(N/4)$ on the 4×4 crossbar of PMINs$^+$. After sorting, those ten tasks are allocated in the following sequence: $T_0(N), T_4(N), [T_1(N/2), T_3(N/2)], [T_6(N/2), T_8(N/2)], [T_2(N/4), T_5(N/4), T_7(N/4), T_9(N/4)]$ in $2(N/4 + n' - 1) + 2(N/8 + n' - 1) + (N/16 + n' - 1) = 13N/16 + 5(n' - 1)$ clocks.

5. Performance of multiple ATAPE tasks on the $x \times x$ crossbar of partitionable MINs$^+$

This section provides the performance evaluation of the ATAPE-task scheduling on the $x \times x$ crossbar of PMINs$^+$. First, two obvious results of the partitionable ATAPE (pATAPE) applications are demonstrated in Section 5.1 on two network systems ($N = 16, N' = 8, x = 2$ and $N = 32, N' = 8, x = 4$). Finally, speedup and throughput performance of the optimal scheduling on

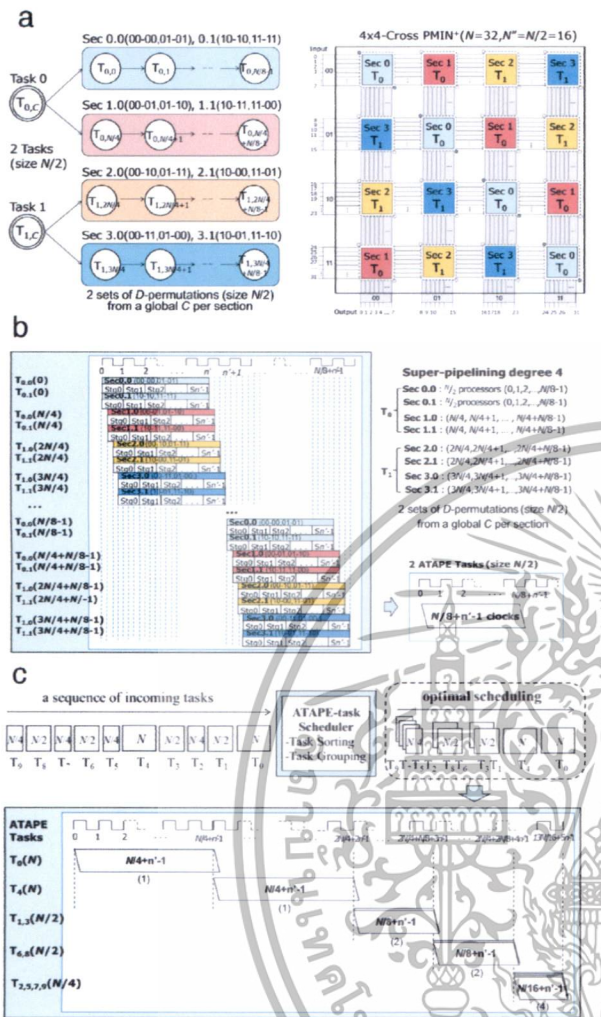


Fig. 14. An example of the triple-right scheduling on an $x \times x$ crossbar of PMIN$^+$($N = 2^n, x = 4$): (a) task flow of controls for 4-section super-pipelining for 2 tasks (size $N'' = N/2$); (b) timing diagram of 2 tasks (size $N/2$); and (c) the optimal scheduling of 10 distinct tasks ($T_0 - T_9$) of sizes $N/2^t$ (between $N/4$ and N).

Fig. 15. Efficient local vs. global mapping for 2^t tasks of size $N'' = N/2^t$ on the $x \times x$ crossbar of PMINs$^+$ ($N = 32, x = 4$): (a) $N'' = N/2$ and (b) $N'' = N/4$.

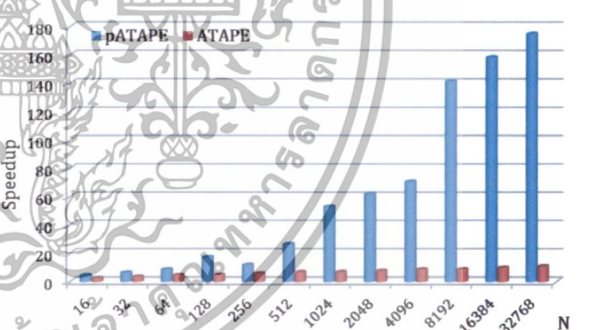


Fig. 16. Speedup of new *p*ATAPE with super-pipelining compared to that of the ATAPE with pipelining [8].

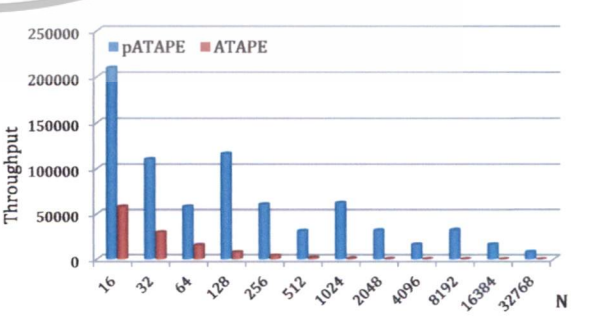


Fig. 17. Throughput of new *p*ATAPE with super-pipelining compared to that of the ATAPE with pipelining [8].

the PMINs$^+$ are presented in Section 5.2 (Tables 4–5, Figs. 16–17), experimented by the simulation study on N up to 32768 processors. Moreover, in our future study, Eqs. (1)–(3) can be easily applied with super-pipelining fruitfully for other applications.

5.1. *Partitionable ATAPE tasks on the* PMINs$^+$

Fig. 18 demonstrates the simplified results of applying our efficient *p*ATAPE for scheduling an ATAPE task of size $N'' = N$ on the 2×2 crossbar of PMINs$^+$ ($N = 16, x = 2, b = 1$) with MIN$^+$ subsystems ($N' = 8, n' = 3$). By super-pipelining (degree 2), two sections ($y = 0, 1$) are utilized successively (in every 1/2 clock) for controlling N rounds within $N/2$ steps ($l = 0, 1, 2, \ldots, 7$), as follows:

Sec 0 ($y = 0$) $\rightarrow$ C_l^0: $\{0, 1, 2, 3, 4, \ldots, 7\}$ and $D^0 = S^0 \oplus C_l^0$ by I/O cross controls 0-0, 1-1.

Sec 1 ($y = 1$) $\rightarrow$ C_l^1: $\{8, 9, 10, 11, \ldots, 15\}$ and $D^1 = S^1 \oplus C_l^1$ by I/O cross controls 0-1, 1-0.

For each round (in section y), the (S^y, D^y) self-routing is computed by the *p*ATAPE function $D^y = S^y \oplus C_l^y$, as shown in each row of the Latin square for communication in sections 0 and 1 simultaneously by starting in a half clock. Clearly, Fig. 18 shows N network configurations ($N = 16$) of applying our *p*ATAPE on the forward

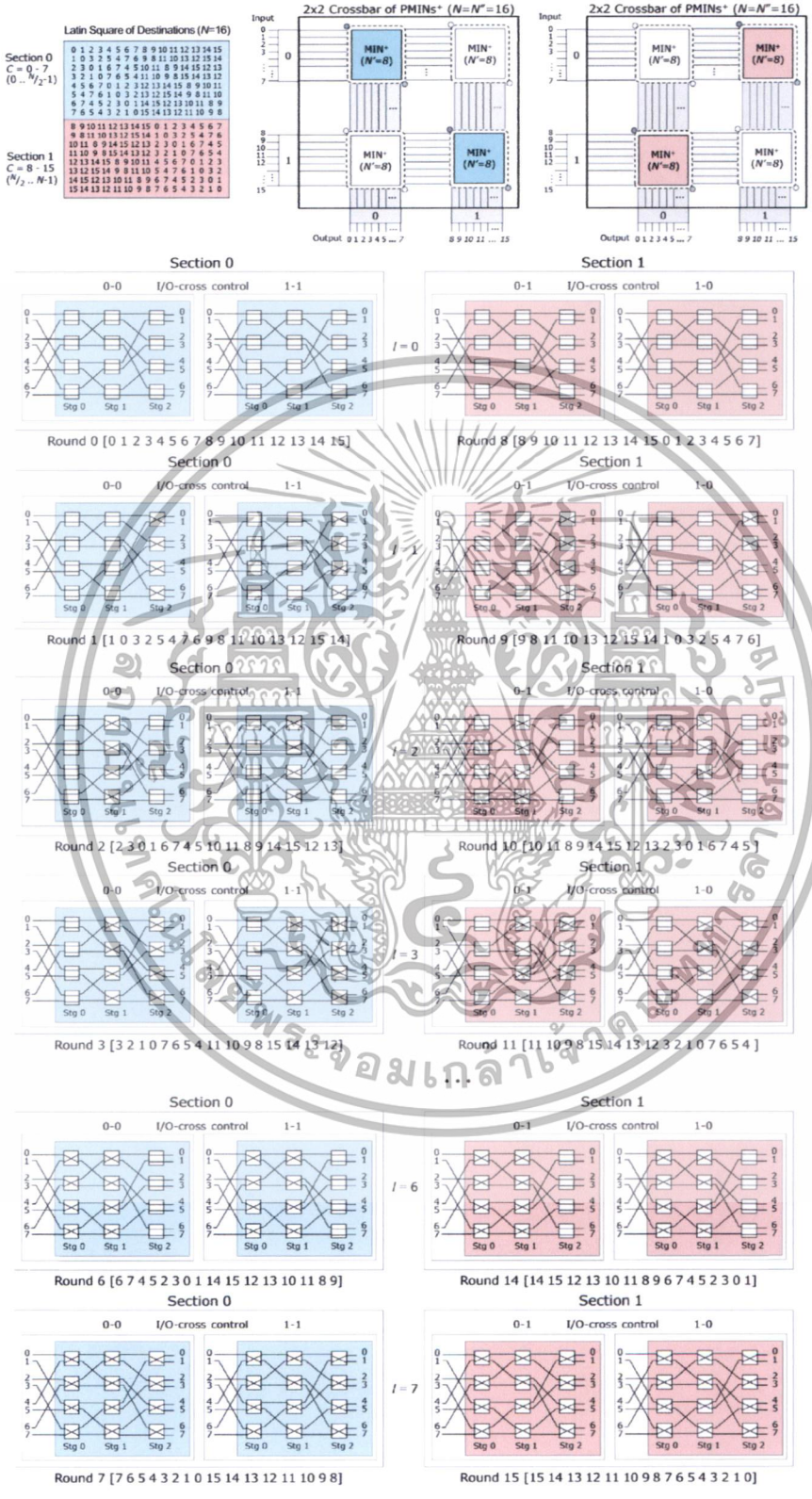


Fig. 18. An example of *p*ATAPE scheduling results ($N'' = 16$) on the 2×2 crossbar of PMINs$^+$ ($N = 16$) with MIN$^+$ subsystems ($N' = 8, n' = 3, d = 2$).

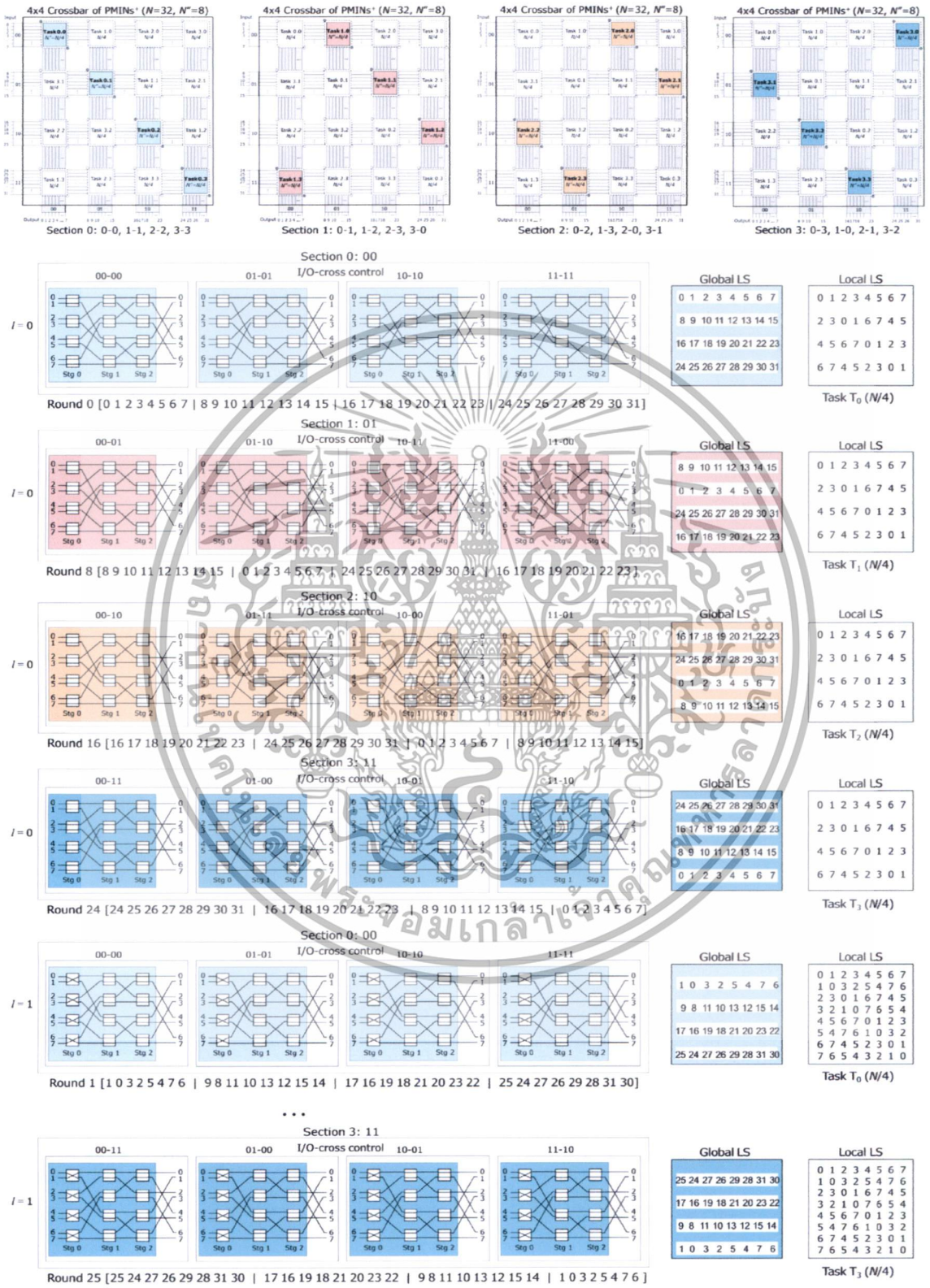


Fig. 19. An example of *p*ATAPE scheduling results ($N'' = 8$) on the 4×4 crossbar of PMINs$^+$ ($N = 32$) with MIN$^+$ subsystems ($N' = 8, n' = 3, d = 2$).

Table 4
Speedup ($T_{w/o\ pipelining}/T_{with\ pipelining}$) for task size ($N$) of the existing ATAPE on CMINs$^+$(CPI $=$ 1) [8] and *p*ATAPE on PMINs$^+$(CPI $= 1/x$).

$x \times x$ crossbar of MINs$^+$/PMINs$^+$	Computing time (in clocks)		Speedup	
	ATAPE [8] $T = N + n' - 1$	New *p*ATAPE $T = N/x + n' - 1/x$	ATAPE [8]	New *p*ATAPE
2×2 crossbar				
$N = 16,\ N' = 8$	18.0	10.5	2.67	4.57
$N = 32,\ N' = 16$	35.0	19.5	3.66	6.56
$N = 64,\ N' = 32$	68.0	36.5	4.71	8.77
4×4 crossbar				
$N = 128,\ N' = 32$	132.0	36.8	4.85	17.41
$N = 256,\ N' = 64$	261.0	69.8	5.89	12.02
$N = 512,\ N' = 128$	518.0	134.8	6.92	26.60
8×8 crossbar				
$N = 1024,\ N' = 128$	1 030.0	134.9	6.96	53.15
$N = 2048,\ N' = 256$	2 055.0	263.9	7.97	62.09
$N = 4096,\ N' = 512$	4 104.0	520.9	8.98	70.77
16×16 crossbar				
$N = 8192,\ N' = 512$	8 200.0	520.9	8.99	141.53
$N = 16\,384,\ N' = 1024$	16 393.0	1033.9	9.99	158.46
$N = 32\,768,\ N' = 2048$	32 778.0	2058.9	11.0	175.07

PMINs$^+$ ($N' = 8$) for all (S^y, D^y) connections in each section y. In the first round, all personalized messages of section 0 ($N_{All}^{y=0}$) with the control ($C_0^0 = 0$) for the I/O cross-control switches for setting subsystems (0, 0) and (1, 1) are transferred. Later, in the next half clock, all personalized messages in section 1 ($N_{All}^{y=1}$) with the control ($C_0^1 = 8$) are sent using the I/O cross-control switches by other subsystems (0, 1) and (1, 0). In particular, for $l = 0$ (or 000_2) both controls ($C_0^0 = 0$ and $C_0^1 = 8$) will be set by the straight-pattern in all switches across $n' = 3$ stages. Similarly, $l = 1$ (or 001_2), both controls ($C_1^0 = 1$ and $C_1^1 = 9$) will be set by the 001-pattern (straight (−) in stages 0 & 1 and cross (x) in stage 2) and so on for $l = 2, 3, \ldots, 7$. Finally after the last step $l = 7$ (=111_2) with $C_7^0 = 7$ and $C_7^1 = 15$, all personalized messages ($N_{All}^{y=0}$,$N_{All}^{y=1}$) are transferred completely.

Fig. 19 illustrates another example of applying the dynamic *p*ATAPE function for scheduling 4 task sizes $N'' = N/4$ on the 4×4 crossbar of PMINs$^+$ ($N = 32, N' = 8, x = 4$) focusing on the efficient local vs. global Latin-square mapping. In this case, the network is partitioned into four sections ($y = 0, 1, 2, 3$) using the control C_l^y of each section y to process in N''/x steps ($l = 0, 1, 2, \ldots, N''/x - 1$).

Sec 00 ($y = 0$) → C_l^0: {0, 1} and $D^0 = S^0 \oplus C_l^0$ by I/O cross controls 00-00, 01-01, 10-10, 11-11.
Sec 11 ($y = 1$) → C_l^1: {8, 9} and $D^1 = S^1 \oplus C_l^1$ by I/O cross controls 00-01, 01-10, 10-11, 11-00.
Sec 10 ($y = 2$) → C_l^2: {16, 17} and $D^2 = S^2 \oplus C_l^2$ by I/O cross controls 00-10, 01-11, 00-00, 11-01.
Sec 11 ($y = 3$) → C_l^3: {24, 25} and $D^3 = S^3 \oplus C_l^3$ by I/O cross controls 00-11, 01-00, 10-01, 11-10.

For each round (of section $y = 0, 1, 2, 3$), the (S^y, D^y) self-routing is computed by the global *p*ATAPE $D^y = S^y \oplus C_l^y$. Fig. 15(b) shows the simplified Latin square partitioning with the global (S^y, D^y) connection for the local (lS^y, lD^y) mapping for each task. With super-pipelining (degree 4), sections 0–3 are allocated simultaneously by starting in every 1/4 clock (for $N'' = N/4$ network configurations per task of *p*ATAPE for four tasks (of sizes N'') on the backward PMINs$^+$ ($N' = 8, n' = 3$)), see Fig. 19. In this case, for $l = 0, 1$, section 0 is partitioned for T_0 with the control C_l^0, section 1 for T_1 with the control C_l^1, section 2 for T_2 with the control C_l^2, and section 3 for T_3 with the control C_l^3. Clearly, each permutation of a global round is partitioned and mapped to 4 local results of the local Latin-square of destinations. For instance, a global permutation [0 1 2 3 4 5 6 7 | 8 9 10 11 12 13 14 15 | 16 17 18 19 20 21 22 23 | 24 25 26 27 28 29 30 31] in round 0 represents the communications of four local permutations [0 1 2 3 4 5 6 7], [2 3 0 1 6 7 4 5], [4 5 6 7 0 1 2 3], and [6 7 4 5 2 3 0 1] in the section $y = 0$ and so on.

5.2. Scheduling results of multiple ATAPE-tasks

In performance evaluation (by simulation study on $N \leq 32\,768$ processors and $x \leq 16$), we compare the speedup and throughput of the new *p*ATAPE and the existing ATAPE [8] in Tables 4 and 5. For small N, the 2×2 crossbar is used and for larger N the 16×16 crossbar is applied (i.e., $x = 16$ for $N = 8192$ to $32\,768$). Note that for the throughput of various task sizes, a number of incoming tasks are sent to the task scheduler (see Fig. 14(c)) before scheduling. In our experiments, the speedup of our *p*ATAPE (of size N) outperformed that of the recent ATAPE [8] by a factor of x because of the effective super-pipelining of $CPI = 1/x$ and $O(N/x)$ on the PMINs$^+$, see Table 4 and Fig. 16.

The throughput results (Fig. 17) have been evaluated in any observation period (such as a period of 1M clocks), where a number of finished tasks (in the observation period) usually increase when executing the smaller task sizes and decrease when running the larger task sizes. For instance (on $N = 8192$ and $x = 16$), new *p*ATAPE can finish 32 264 tasks of sizes ($N'' = 8192$) according to super-pipelining for multiple tasks (see Table 5), whereas the ATAPE [8] can process only 128 tasks during the same observation period.

6. Conclusion

This paper proposes (1) the prototype of the partitionable MINs$^+$ (PMINs$^+$) to support multiple tasks and (2) the optimal VA-DE (valuable ATAPE with dynamic embedding), which is the partitionable ATAPE (*p*ATAPE) embedding and automatic mapping on the $x \times x$ crossbar of PMINs$^+$ in optimal time complexity and space utilization, according to the triple-right scheduling (right task, right section, right time). On the superb PMINs$^+$ using the I/O cross-control switches incorporated with the super-pipelining, all x sections ($y = 0$ to $x - 1$) can be utilized simultaneously for

Table 5
Throughput (finished tasks per observation-period) of the existing ATAPE on CMINs$^+$ [8] and *p*ATAPE on PMINs$^+$.

$x \times x$ crossbar of MINs$^+$ /PMINs$^+$	Task size $= N$		Various sizes (N, $N/2$, $N/4$, ..., N)	
	ATAPE [8]	New *p*ATAPE	ATAPE [8]	New *p*ATAPE
2×2 crossbar				
$N = 16$, $N' = 8$	58 254	209 715	74 898	262 144
$N = 32$, $N' = 16$	29 959	110 375	38 836	139 810
$N = 64$, $N' = 32$	15 420	58 254	20 165	74 898
4×4 crossbar				
$N = 128$, $N' = 32$	7 944	116 508	13 329	185 043
$N = 256$, $N' = 64$	4 018	60 787	6 794	99 078
$N = 512$, $N' = 128$	2 024	31 301	3 442	51 996
8×8 crossbar				
$N = 1024$, $N' = 128$	1 018	62 602	2 158	127 100
$N = 2048$, $N' = 256$	510	31 896	1 084	66 052
$N = 4096$, $N' = 512$	256	16 132	544	33 825
16×16 crossbar				
$N = 8192$, $N' = 512$	128	32 264	329	81 285
$N = 16\,384$, $N' = 1024$	64	16 241	165	41 344
$N = 32\,768$, $N' = 2048$	32	8 158	83	20 875

various task sizes (N/x to N). For 100% utilization, the N network configurations are allocated with the right sequencing of the right-jumping control $C_l^y = yN/x + l$, where $0 \le y < x$ and $l = 0, 1, 2, \ldots, N/x-1$. Thus for any task (size N), all specific messages N_{All}^y in alternate section y from all sources (S^y) are transferred to all destinations ($D^y = S^y \oplus C_l^y$) in new optimal time $O(N/x)$ over $O(N)$ of the ATAPE on the CMINs$^+$ [8]. In the performance evaluation, the speedup of our *p*ATAPE scheduling on the PMINs$^+$ outperformed that of the existing ATAPE on the CMINs$^+$ by a factor of x, experimented on $N \le 32\,768$ and $x \le 16$.

Acknowledgments

We would like to gratefully thank the Office of the Higher Education Commission (Bangkok, Thailand) for the research grant OHEC 04-01-04-001 (2011–2014) to Miss Roselin Petagon to fulfill the requirement of doctoral degree (Ph.D.) in Computer Science at KMITL and also gratefully thank King Mongkut's Institute of Technology, Ladkrabang (KMITL), Bangkok, Thailand, and Chiang Mai Rajabhat University (Chiang Mai, Thailand) for their great support and corporation.

References

[1] Z. Chen, Z.J. Liu, Z.L. Qiu, Bidirectional shuffle–exchange network and tagbased routing algorithm, IEEE Commun. Lett. 7 (3) (2003) 121–123.
[2] W.Y. Chou, C. Chen, All-to-all personalized exchange in generalized shuffle–exchange networks, Theoret. Comput. Sci. 411 (3) (2010) 1669–1884.
[3] D.H. Lawrie, Access and alignment of data in an array processor, IEEE Trans. Comput. C-24 (12) (1975) 1145–1155.
[4] V.W. Liu, C. Chen, R.B. Chen, Optimal all-to-all personalized exchange in d-nary banyan multistage interconnection networks, J. Comb. Optim. 14 (10) (2007) 131–142.
[5] A. Massini, All-to-all personalized communication on multistage interconnection networks, Discrete Appl. Math. 128 (2) (2003) 435–446.
[6] D.S. Parker, Notes on shuffle/exchange-type switching networks, IEEE Trans. Comput. C-29 (3) (1980) 213–222.
[7] M.C. Pease, The indirect binary n-cube microprocessor array, IEEE Trans. Comput. C-26 (5) (1977) 458–473.
[8] R. Petagon, J. Werapun, Embedding the optimal all-to-all personalized exchange on multistage interconnection networks+, J. Parallel Distrib. Comput. 88 (2016) 15–30.
[9] B. Prisacari, G. Rodriguez, C. Minkenberg, Generalized hierarchical all-to-all exchange patterns, in: 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, May, 2013, pp. 537–547.
[10] D.S. Scott, Efficient all-to-all communication patterns in hypercube and mesh topologies, in: Proc. Sixth Distributed Memory Computing Conference, 1991, pp. 398–403.
[11] Y.J. Suh, K.G. Shin, Efficient all-to-all personalized exchange in multidimensional torus networks, in: Proc. Int. Conf. Parallel Processing, 1998, pp. 468–475.
[12] Y.J. Suh, K.G. Shin, All-to-all personalized communication in multidimensional torus and mesh networks, in: Proc. Int. Conf. Parallel Processing, vol. 12, No. 1, 2001, pp. 38–59.
[13] C.-L. Wu, T.-Y. Feng, On a class of multistage interconnection networks, IEEE Trans. Comput. 29 (8) (1980) 694–702.
[14] Y. Yang, J. Wang, Optimal all-to-all personalized exchange in self-routable multistage networks, IEEE Trans. Parallel Distrib. Syst. 11 (3) (2000) 261–274.
[15] Y. Yang, J. Wang, Optimal all-to-all personalized exchange in a class of optical multistage networks, IEEE Trans. Parallel Distrib. Syst. 12 (9) (2001) 567–582.

Roselin Petagon received the B.S. degree in 2005 from Chiang Mai Rajabhat University, Chiang Mai, Thailand and the M.S. degree in Computer Science in 2008 from King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand. Since 2008, she has been a lecturer of Computer Science at Chiang Mai Rajabhat University, Chiang Mai, Thailand. Miss Petagon is a Ph.D. candidate (in Computer Science), at KMITL, Bangkok, Thailand. Her research interests include parallel algorithms and architectures, interconnection networks, and distributed computing.

Jeeraporn Werapun received the B.S. degree from Kasetsart University, Bangkok, Thailand, the M.S. degree from Chulalongkorn University, Bangkok, Thailand, the M.S. degree in Computer Science in 1993, and the D.Sc. degree in Computer Engineering (Parallel and Distributed Systems) in 1999 from the George Washington University, Washington, D.C., USA. Dr. Werapun is an Associate Professor of Computer Science at King Mongkut's Institute of Technology, Ladkrabang (KMITL), Bangkok, Thailand. Her research interests include parallel and distributed systems, multistage interconnection networks, and parallel algorithms.

ประวัติผู้เขียน

ชื่อ	นางสาวรสลิน เพตะกร
วัน เดือน ปีเกิด	28 พฤศจิกายน 2525
ที่อยู่ปัจจุบัน	334/32 บ้านป่าไผ่ (พิมุกต์ 4) หมู่ที่ 3 ตำบลสันพระเนตร อำเภอสันทราย จังหวัดเชียงใหม่
ประวัติการศึกษา	(2548) ครุศาสตรบัณฑิต สาขาคอมพิวเตอร์ศึกษา เกรดเฉลี่ย 3.33 มหาวิทยาลัยราชภัฏเชียงใหม่ (2551) วิทยาศาสตรมหาบัณฑิต สาขาวิทยาการคอมพิวเตอร์ เกรดเฉลี่ย 3.47 สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ทุนการศึกษาที่ได้รับ	ทุนโครงการส่งเสริมงานวิจัยในอุดมศึกษา สำนักงานคณะกรรมการการอุดมศึกษา และทุนพัฒนาบุคลากรมหาวิทยาลัยราชภัฏเชียงใหม่
ผลงานทางวิชาการ	1. R. Petagon and J. Werapun, "Parallel Construction of Independent Spanning Trees on Dynamic Interconnection Networks." International Joint conference on computer science and Software Engineering (JCSSE 2008). Vol. 1: International Sessions, 2008, pp. 187-192. 2. R. Petagon and J. Werapun, Embedding the optimal all-to-all personalized exchange on multistage interconnection networks^{+}, J. Parallel Distrib. Comput. 88 (2016), 15-30. 3. R. Petagon and J. Werapun, VA-DE: Valuable ATAPE with Dynamic Embedding and Super-pipeline Scheduling on Partitionable Multistage Interconnection Networks^{+}, J. Parallel Distrib. Comput. 102 (2017), 1-15.