การออกแบบตัวควบคุมสำหรับแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบบางส่วนได้แบบอสมวาร

นายปกรณ์ ทู้ไพเราะ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2552

A DESIGN OF CONTROLLER FOR ASYNCHRONOUS PARTIAL RECONFIGURABLE COMPUTING ARRAY

Mr. Pakon Thuphairo

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic year 2009

หัวข้อวิทยานิพนธ์	การออกแบบตัวควบคุมสำหรับแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบบางส่วนได้แบบอสมวาร
โดย	นายปกรณ์ ทู้ไพเราะ
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร. อาทิตย์ ทองทักษ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.สาธิต วงศ์ประทีป)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

..... กรรมการ
(อาจารย์ ดร.เกริก ภิรมย์โสภา)

..... กรรมการภายนอกมหาวิทยาลัย
(อาจารย์ ดร.เนื่องวงศ์ ทวยเจริญ)

ปกรณ์ ทู้ไพเราะ : การออกแบบตัวควบคุมสำหรับแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบบางส่วนได้แบบอสมวาร. (A DESIGN OF CONTROLLER FOR ASYNCHRONOUS PARTIAL RECONFIGURABLE COMPUTING ARRAY) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร.อาทิตย์ ทองทักษ์ 106 หน้า.

งานวิจัยนี้เสนอการออกแบบตัวควบคุมสำหรับแถวลำดับคำนวณที่สามารถปรับเปลี่ยนโครงแบบบางส่วนได้แบบอสมวาร รวมถึงได้เสนอการออกแบบแถวลำดับคำนวณ 2 มิติแบบอสมวารที่สามารถประมวลผลข้อมูลขนาด 8 บิต และ 1 บิตได้ มีการเชื่อมต่อกัน 4 ทิศทาง ได้แก่ บน ล่าง ซ้าย และขวา เส้นทางข้อมูลขนาด 1 บิต นอกจากประมวลผลข้อมูลขนาด 1 บิต แล้ว ยังสามารถถูกใช้เพื่อสร้างสัญญาณเงื่อนไขได้ และมีสัญญาณตัวทดที่ถูกแยกออกมาเพื่อให้ปรับเปลี่ยนโครงสร้างเป็นวงจรบวกและลบขนาดใหญ่ได้ การออกแบบตัวควบคุม จะแบ่งส่วนประกอบภายในเป็น 3 ส่วน ได้แก่ ตัวกำหนดการ ตัววาง และตัวบรรจุ ขั้นตอนวิธีมาก่อนให้บริการก่อนและขั้นตอนวิธีพื้นที่น้อยที่สุดก่อนถูกใช้เพื่อสร้างตัวกำหนดการ และนำขั้นตอนวิธีจากงานวิจัยหนึ่งมาปรับปรุงเพื่อสร้างตัววางในการหาพื้นที่ที่เหมาะสมในขั้นตอนการวาง งานวิจัยนี้ได้เสนอกลไกในการตรวจสอบการมาของสัญญาณ ซึ่งสามารถตรวจสอบได้ในระดับการไหลของข้อมูล การทดสอบหน้าที่การทำงานของแถวลำดับคำนวณที่ได้ออกแบบไว้ กำหนดให้ทดสอบแถวลำดับโดยปรับเปลี่ยนให้เป็นสายโซ่ของวงจรบวก และยังได้ทดสอบเพิ่มเติมด้วยการนำวงจรอสมวาร ซึ่งออกแบบเป็นวงจรที่ป้อนสัญญาณควบคุมให้กับวงจรที่มีลักษณะการทำงานแบบวนซ้ำ เพื่อแสดงถึงความเป็นไปได้ในการแปลภาษาของงานประยุกต์จริงที่ถูกเขียนด้วยภาษาระดับสูง รวมถึงได้ทดสอบตัวควบคุมโดยการจำลองการทำงานของส่วนประกอบภายในแต่ละส่วน

จากการทดสอบทั้งหมด พบว่าตัวควบคุมสามารถทำงานได้อย่างถูกต้องในระดับการไหลของข้อมูล และแถวลำดับคำนวณสามารถทำงานได้อย่างถูกต้องในระดับพฤติกรรม โดยใช้โปรแกรมบาลซ่าเวอร์ชั่น 3.5 และโมเดลซิมไซลิงซ์เอ็ดดิชั่นเวอร์ชั่น 6.4b

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต ปกรณ์ ทู้ไพเราะ

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก อาทิตย์ ทองทักษ์

ปีการศึกษา 2552

4870679521 : MAJOR COMPUTER ENGINEERING

KEYWORDS : ASYNCHRONOUS CIRCUIT / CONTROLLER / DATAFLOW COMPUTING / RECONFIGURABLE ARCHITECTURE

PAKON THUPHAIRO : A DESIGN OF CONTROLLER FOR ASYNCHRONOUS PARTIAL RECONFIGURABLE COMPUTING ARRAY. THESIS ADVISOR : ASST. PROF. ARTHIT THONGTAK, Ph.D., 106 pp.

This research proposes a design of controller for asynchronous partial reconfigurable computing array and also provides a design of asynchronous 2D computing array, which is capable of performing both of 8 bit and 1 bit data. The interconnection consists of 4 directions which are up, down, left and right. The 1 bit data path does not only perform 1 bit data itself but also be used to produce conditional signals. Additionally, the carry signals are routed separately in order to form larger adders and subtractors. The controller design is divided into 3 parts, which are scheduler, placer and loader. The First-Come First-Serve and Minimal Area First algorithms are used to implement the scheduler. An existing algorithm is adapted to implement the placer to find a suitable area in placement. This research also provides a mechanism to check the signal arrival at data flow level. Functional verification of the computing array is performed by reconfiguring an array to operate as chains of adders. Moreover, an asynchronous circuit producing loop control signals is employed to investigate the feasibility of compilation of real world applications written in high level computer languages. The controller is verified by individually simulating each of underlying components.

All verifications show that the controller operates correctly at data flow level and the computing array operates correctly at behavioral level. The tools used for verification are Balsa 3.5 and ModelSim XE 6.4b.

Department : Computer Engineering Student's Signature Pakon Thuphairo

Field of Study : Computer Engineering Advisor's Signature Arthit Thongtak

Academic Year : 2009

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สามารถสำเร็จลุล่วงได้ด้วยดี อันเกิดจาก ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์ ผู้เป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ระดับปริญญามหาบัณฑิต ได้ให้คำปรึกษาและช่วยเหลือทั้งการทำวิทยานิพนธ์เล่มนี้และการทำวิจัยด้วยดีตลอดช่วงเวลาที่ผู้วิจัยได้เข้าศึกษาในจุฬาลงกรณ์มหาวิทยาลัยแห่งนี้

ขอขอบพระคุณ รองศาสตราจารย์ ดร.สาธิต วงศ์ประทีป อาจารย์ ดร.เนื่องวงศ์ ทวยเจริญ อาจารย์ ดร.เกริก ภิรมย์โสภา และคณาจารย์ผู้ที่ได้สละเวลาให้คำปรึกษาในการแก้ไขวิทยานิพนธ์เล่มนี้ให้สำเร็จลุล่วงไปได้ด้วยดี ทั้งขอขอบพระคุณคณาจารย์ในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ได้ประสิทธิประสาทวิชาความรู้ต่างๆ ให้กับผู้ทำวิจัย อันส่งผลต่อแนวคิดทั้งสิ้นในการดำเนินการวิจัยนี้

กราบขอบพระคุณ บิดาและมารดา ที่ได้ให้สิ่งต่างๆ ที่สนับสนุนต่อการทำวิจัยมาโดยตลอด ขอขอบคุณเพื่อนนักวิจัยและผู้ที่เกี่ยวข้องทั้งหลาย ที่ได้ให้ข้อเสนอแนะ ไม่ว่าโดยทางตรงหรือทางอ้อมก็ตาม ที่ส่งผลต่อการทำวิจัย

ท้ายที่สุดนี้ ผู้ทำวิจัยขออ้างบุญอันเกิดจากการทำวิทยานิพนธ์เล่มนี้และบุญที่ได้ทำไว้ในทุกกาล ขออุทิศให้กับผู้มีพระคุณทั้งหลายของผู้ทำวิจัยทั้งที่ได้กล่าวมาข้างต้นและผู้มีพระคุณท่านอื่นที่มิได้กล่าวไว้ ณ ที่นี้ รวมถึงผู้ที่ได้อ่านวิทยานิพนธ์เล่มนี้ด้วย

สารบัญ

หน้า

บทคัดย่อภาษาไทย..... ง

บทคัดย่อภาษาอังกฤษ..... จ

กิตติกรรมประกาศ..... ฉ

สารบัญ..... ช

สารบัญตาราง..... ฎ

สารบัญภาพ..... ฎ

บทที่

1 บทนำ..... 1

1.1 ความเป็นมาและความสำคัญของปัญหา..... 1

1.2 วัตถุประสงค์ของการวิจัย..... 3

1.3 ขอบเขตของการวิจัย..... 3

1.4 ข้อจำกัดของการวิจัย..... 5

1.5 คำจำกัดความที่ใช้ในการวิจัย..... 5

1.6 ประโยชน์ที่คาดว่าจะได้รับ..... 5

1.7 วิธีดำเนินการวิจัย..... 5

1.8 ลำดับขั้นตอนในการเสนอผลการวิจัย..... 6

1.9 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์..... 6

2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง..... 7

2.1 ทฤษฎีที่เกี่ยวข้อง..... 7

2.1.1 วงจรอสมวาร..... 7

2.1.1.1 การแกว่งของสัญญาณนาฬิกา..... 8

2.1.1.2 การบริโภคพลังงาน..... 8

2.1.1.3 ประสิทธิภาพของวงจร..... 9

2.1.2 การออกแบบวงจรอสมวารด้วยวิธีโครงสร้างการไหลของข้อมูลแบบสถิต. 9

2.1.3 สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้..... 12

2.1.3.1 สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างละเอียด..... 14

บทที่ หน้า

2.1.3.2 สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างหยาบ..... 14
2.2 งานวิจัยที่เกี่ยวข้อง..... 16
2.2.1 งานวิจัยที่เกี่ยวข้องทางด้านสถาปัตยกรรมแบบอสมวาร..... 16
2.2.2 งานวิจัยที่เสนอแบบจำลองเพื่อจัดการโครงแบบ..... 19

3 การออกแบบแถวลำดับคำนวณและตัวควบคุม..... 21
3.1 การออกแบบแถวลำดับนวณ..... 21
3.1.1 หน่วยประมวลผลย่อย..... 22
3.1.2 ส่วนประกอบภายในหน่วยประมวลผลย่อย..... 23
3.1.2.1 ตัวเลือกข้อมูลขาเข้าขนาด 8 บิต..... 24
3.1.2.2 ตัวเลือกข้อมูลขาเข้าขนาด 1 บิต..... 24
3.1.2.3 หน่วยดำเนินการขนาด 8 บิต..... 24
3.1.2.4 หน่วยดำเนินการขนาด 1 บิต..... 26
3.1.2.5 ตัวจัดเส้นทางข้อมูลขาออกขนาด 8 บิต..... 26
3.1.2.6 ตัวจัดเส้นทางข้อมูลขาออกขนาด 1 บิต..... 26
3.1.2.7 ตัวผสานสัญญาณตัวทดขาเข้า..... 27
3.1.2.8 ตัวจัดเส้นทางสัญญาณตัวทด..... 27
3.1.2.9 ตัวเลือกสัญญาณตัวทด..... 27
3.1.2.10 ตัวกระจายสัญญาณตัวทด..... 27
3.1.2.11 ตัวกระจายข้อมูลขาออกของแลตช์ 1 บิต..... 27
3.1.2.12 ตัวผสานสัญญาณตัวทดขาออก..... 28
3.1.2.13 ตัวจัดเส้นทางสัญญาณตัวทดขาออก..... 28
3.2 ตัวควบคุม..... 28
3.2.1 โครงสร้างโดยรวมของตัวควบคุม..... 29
3.2.2 การออกแบบโครงสร้างของโครงแบบ..... 30
3.2.3 ตัวกำหนดการ..... 31
3.2.3.1 การออกแบบด้วยขั้นตอนวิธีแบบมาก่อนให้บริการก่อน..... 31
3.2.3.2 การออกแบบด้วยขั้นตอนวิธีแบบพื้นที่น้อยที่สุดก่อน..... 32
3.2.3.3 โครงสร้างโดยรวมของตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน..... 34

บทที่ หน้า

3.2.3.4 ส่วนประกอบภายในตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน..... 34
3.2.4 ตัววาง..... 36
3.2.4.1 โครงสร้างโดยรวมของตัววาง..... 37
3.2.5 ตัวบรรจุ..... 38
3.2.5.1 โครงสร้างโดยรวมของตัวบรรจุ..... 38

4 การทดลอง ผลการทดลอง และสรุปผลการทดลอง..... 41
4.1 การทดสอบแถวลำดับคำนวณ..... 41
4.1.1 การทดสอบแถวลำดับคำนวณขนาด 8X8..... 41
4.1.2 การทดสอบการนำวงจรอสมวารมาจัดกลุ่มลงบนแถวลำดับคำนวณ..... 43
4.2 การทดสอบตัวควบคุม..... 46
4.2.1 การทดสอบตัวกำหนดการ..... 46
4.2.1.1 การทดสอบตัวกำหนดการแบบมาก่อนให้บริการก่อน..... 46
4.2.1.2 การทดสอบตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน..... 47
4.2.2 การทดสอบตัววาง..... 48
4.2.3 การทดสอบตัวบรรจุ..... 50
4.3 การเปรียบเทียบกำลังไฟฟ้า..... 52
4.4 การประมาณค่าขนาดของตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธีที่แตกต่างกัน..... 55
4.5 การเปรียบเทียบประสิทธิภาพในการวางเมื่อใช้ตัวกำหนดการที่แตกต่างกัน..... 58
4.6 ความเป็นไปได้ในการนำวงจรอสมวารมาประมวลผลบนแถวลำดับคำนวณ..... 64
4.6.1 วงจรบวกแบบส่งต่อสัญญาณตัวทด..... 64
4.6.2 วงจรสร้างสัญญาณควบคุมวนซ้ำที่มีขนาด 16 บิต..... 65
4.7 สรุปผลการทดลอง..... 66

5 สรุปผลการวิจัย ข้อจำกัดของงานวิจัย และข้อเสนอแนะ..... 69
5.1 สรุปผลการวิจัย..... 69
5.2 ข้อจำกัดของงานวิจัย..... 70

บทที่ หน้า

5.3 ข้อเสนอแนะ..... 70

รายการอ้างอิง..... 73

บรรณานุกรม..... 75

ภาคผนวก..... 77

รหัสต้นฉบับตัวควบคุม..... 78

ประวัติผู้เขียนวิทยานิพนธ์..... 106

สารบัญตาราง

ตารางที่		หน้า
3.1	ตัวดำเนินการขนาด 8 บิต ที่อยู่ภายในหน่วยประมวลผลย่อย.....	24
3.2	ตัวดำเนินการขนาด 1 บิต ที่อยู่ภายในหน่วยประมวลผลย่อย.....	26
3.3	สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้.....	39
4.1	งานที่ใช้ในการทดสอบตัวกำหนดการแบบมาก่อนให้บริการก่อน.....	46
4.2	ผลการเปรียบเทียบขนาดของตัวกำหนดการ 2 วิธี.....	56
4.3	ขนาดและมิติของงานที่ถูกสุ่มเพื่อใช้ในการทดสอบประสิทธิภาพในการวาง.....	59
4.4	สถานะรอที่เกิดขึ้นเมื่อใช้ขั้นตอนวิธีมาก่อนให้บริการก่อน.....	62
4.5	สถานะรอที่เกิดขึ้นเมื่อใช้ขั้นตอนวิธีพื้นที่น้อยที่สุดก่อน.....	62
4.6	ผลการเปรียบเทียบประสิทธิภาพในการวางเมื่อใช้ตัวกำหนดการที่แตกต่างกัน..	63

สารบัญภาพ

ภาพที่		หน้า
2.1	การทำงานของวงจรสมวาร.....	7
2.2	การทำงานของวงจรอสมวาร.....	8
2.3	แลตช์.....	9
2.4	การออกแบบด้วยวิธีดิมส์.....	10
2.5	ตัวควบคุมการไหลแบบไม่มีเงื่อนไข 3 ประเภท.....	10
2.6	ตัวควบคุมการไหลแบบมีเงื่อนไข.....	11
2.7	อาร์บิเตอร์.....	12
2.8	ประสิทธิภาพและความยืดหยุ่นของการประมวลผลที่ปรับเปลี่ยนโครงแบบได้...	13
2.9	โครงสร้างที่มีการเชื่อมต่อกันแบบตาข่าย.....	15
2.10	โครงสร้างที่มีการเชื่อมต่อกันแบบเชิงเส้น.....	15
2.11	โครงสร้างที่มีการเชื่อมต่อกันแบบครอสส์บาร์.....	16
2.12	โครงสร้างภายในสถาปัตยกรรมไปป์เรนซ์.....	17
2.13	โครงสร้างการเชื่อมต่อของหน่วยประมวลผลย่อยของสถาปัตยกรรมอาร์กา.....	18
2.14	โครงสร้างภายในหน่วยประมวลผลย่อยของสถาปัตยกรรมอาร์กา.....	18
2.15	แบบจำลองที่ใช้ในการจัดการโครงแบบ.....	19
2.16	เส้นขอบบนและขอบล่างที่ใช้ในขั้นตอนวิธีของเอสมาอิลดัสต์และคณะ.....	20
3.1	โครงสร้างของแถวลำดับคำนวณแบบ 2 มิติ ขนาด 8X8.....	21
3.2	โครงสร้างโดยรวมของหน่วยประมวลผลย่อย.....	22
3.3	ส่วนประกอบภายในหน่วยประมวลผลย่อย.....	23
3.4	โครงสร้างโดยรวมของตัวควบคุม.....	29
3.5	โครงแบบลักษณะสี่เหลี่ยมมุมฉาก.....	30
3.6	โครงสร้างของโครงแบบ.....	30
3.7	เรจิสเตอร์แบบสายท่อที่ใช้เก็บโครงแบบ.....	31
3.8	อุปกรณ์ที่นำมาใช้ในการตรวจสอบการร้องขอจากโครงแบบและจากตัววาง.....	32
3.9	วงจรที่ใช้ตรวจสอบการมาของสัญญาณ.....	33
3.10	กลไกการตรวจสอบการมาถึงของสัญญาณ.....	33
3.11	โครงสร้างโดยรวมของตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน.....	34

ภาพที่		หน้า
3.12	ส่วนประกอบภายในตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน……	35
3.13	หน่วยความจำที่ใช้ในตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน……..	35
3.14	พื้นที่ที่ถูกตรวจสอบเพื่อหาจำนวนหน่วยประมวลผลรอบข้างที่ถูกจอง……	36
3.15	โครงสร้างโดยรวมของตัววาง……	37
3.16	โครงสร้างข้อมูลของโครงแบบที่ถูกส่งไปยังตัวบรรจุ……..……..	38
3.17	โครงสร้างโดยรวมของตัวบรรจุ……..	38
4.1	แถวลำดับคำนวณขนาด 8X8 ที่ถูกปรับเปลี่ยนเพื่อทดสอบการทำงาน……...	42
4.2	ผลการทดสอบแถวลำดับคำนวณบนโปรแกรมโมเดลซิม……	42
4.3	การจัดกลุ่มวงจรลงบนแถวลำดับคำนวณ……	43
4.4	การส่งข้อมูลระหว่างหน่วยประมวลผลย่อยหลังจากจัดกลุ่มแล้ว……	44
4.5	ผลการทดสอบวงจรที่ถูกวางลงบนแถวลำดับคำนวณบนโปรแกรมโมเดลซิม.…	45
4.6	ผลการทดสอบตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธีมาก่อนบริการก่อน..	47
4.7	ผลการทดสอบตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธีพื้นที่น้อยที่สุดก่อน..	47
4.8	สถานะของแถวลำดับคำนวณที่ใช้ทดสอบการวาง……..……	48
4.9	การกำหนดค่าสถานะปัจจุบันของแถวลำดับในโปรแกรมบาลซ่า……	49
4.10	ผลการทดสอบตัววาง……	49
4.11	โครงแบบที่ถูกวางลงบนแถวลำดับคำนวณที่ใช้ในการทดสอบ……	50
4.12	การบรรจุโครงแบบลงบนแถวลำดับคำนวณ……	51
4.13	ผลการทดสอบตัวบรรจุ……..	51
4.14	ขั้นตอนการทดสอบกำลังไฟฟ้า……	52
4.15	รหัสต้นฉบับที่ใช้ในการเก็บการเปลี่ยนแปลงสัญญาณ……	53
4.16	ผลการทดสอบกำลังไฟฟ้าของหน่วยประมวลผลย่อยแบบอสมวาร……	54
4.17	ผลการทดสอบกำลังไฟฟ้าของหน่วยประมวลผลย่อยแบบสมวาร……...	54
4.18	ผลการประมาณค่าขนาดของตัวกำหนดการแบบมาก่อนให้บริการก่อนที่มี เรจิสเตอร์ 4, 8, 12 และ 16 ตัว……	55
4.19	ผลการประมาณค่าขนาดของตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อนที่มี เรจิสเตอร์ 4, 8, 12 และ 16 ตัว……	56
4.20	กราฟความสัมพันธ์ระหว่างจำนวนเรจิสเตอร์และขนาดของตัวกำหนดการ…….	57
4.21	ลำดับการวางเมื่อใช้ขั้นตอนวิธีมาก่อนให้บริการก่อน……..	60

ภาพที่		หน้า
4.22	ลำดับการวางเมื่อใช้ขั้นตอนวิธีพื้นที่น้อยที่สุดก่อน.....	61
4.23	วงจรบวกแบบส่งต่อสัญญาณตัวทด.....	65
4.24	วงจรสร้างสัญญาณควบคุมวนซ้ำที่มีขนาด 16 บิต.....	65
4.25	ตัวอย่างการนำตัวควบคุมและสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้มาทำงานร่วมกัน.....	67

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

สถาปัตยกรรมที่อยู่ในรูปแบบไมโครโพรเซสเซอร์ (Microprocessor) สามารถประมวลผลงาน (Task) ประเภทต่างๆ ได้ เพราะเป็นสถาปัตยกรรมที่มีความยืดหยุ่นสูง สามารถประมวลผลซอฟต์แวร์ที่ถูกเขียนขึ้นด้วยภาษาคอมพิวเตอร์ระดับสูง (High level computer languages) ได้ แต่ด้วยความยืดหยุ่นของสถาปัตยกรรมนี้ ทำให้ส่งผลต่อประสิทธิภาพทางด้านความเร็ว เพราะไม่ได้ถูกออกแบบโครงสร้างให้เข้ากับลักษณะเฉพาะของงานใดงานหนึ่ง ส่วนวงจรรวมเฉพาะกิจ (ASIC: Application Specific Integrated Circuit) ถูกออกแบบขึ้นเพื่อใช้งานแต่ละประเภทโดยเฉพาะ จึงให้ประสิทธิภาพด้านความเร็วสูง แต่ไม่สามารถนำไปใช้กับการประมวลผลในงานอื่นได้

แนวคิดของสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ (Reconfigurable architecture) จึงเข้ามามีบทบาทในข้อได้เปรียบและเสียเปรียบทางด้านความเร็วและความยืดหยุ่นนี้ เนื่องจากสามารถปรับเปลี่ยนโครงแบบ (Configuration) ของสถาปัตยกรรมให้เหมาะสมกับลักษณะงานที่ต้องใช้การคำนวณที่แตกต่างกันออกไปได้

สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้สามารถแบ่งออกเป็น 2 ประเภท โดยแบ่งตามขนาดข้อมูลที่นำมาประมวลผล ได้แก่ สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างละเอียด (Fine grain reconfigurable architecture) เช่น เอฟพีจีเอ (FPGA: Field-Programmable Gate Array) เป็นต้น ส่วนสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างหยาบ (Coarse grain reconfigurable architecture) ถูกออกแบบมาเพื่อประมวลผลข้อมูลที่มีขนาดใหญ่กว่า เพื่อให้เหมาะสมกับจุดประสงค์ของงานที่ต้องการนำมาประมวลผล

สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างละเอียดมีข้อดี คือ สามารถปรับเปลี่ยนโครงแบบได้ยืดหยุ่นกว่าแบบสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างหยาบ ซึ่งสามารถปรับเปลี่ยนโครงแบบให้เหมาะสมกับการประมวลผลของงานที่มีลักษณะเฉพาะสูง แต่มีข้อเสีย คือ ต้องสูญเสียพื้นที่เป็นจำนวนมากในการสร้างสายเชื่อมโยง (Interconnection) ที่เชื่อมหน่วยประมวลผลย่อย (Processing element) เข้าด้วยกัน และใช้ปริมาณข้อมูลสำหรับการ

ปรับเปลี่ยนสูง ส่วนสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างหยาบจะให้อัตราระหว่างพื้นที่ของหน่วยประมวลผลย่อยกับพื้นที่ของสายเชื่อมโยงที่ดีกว่า ซึ่งสถาปัตยกรรมนี้อาจถูกปรับเปลี่ยนเฉพาะบางส่วนได้ เพื่อลดปริมาณข้อมูลและเวลาที่ใช้ในการปรับเปลี่ยน รวมถึงอาจถูกปรับเปลี่ยนโครงแบบบางส่วนบนพื้นที่ว่างโดยไม่จำเป็นต้องรอให้โครงแบบที่กำลังถูกประมวลผลทำงานเสร็จสิ้นก่อน

เนื่องจากวิธีออกแบบวงจรดิจิทัลที่แพร่หลายในปัจจุบันเป็นการออกแบบวงจรสมวาร (Synchronous circuit) ซึ่งใช้สัญญาณนาฬิการ่วม (Global clock signal) ในการควบคุมจังหวะการทำงานของวงจรเชิงลำดับ (Sequential circuit) สัญญาณนาฬิการ่วมนี้จะถูกส่งไปยังเรจิสเตอร์ (Register) ซึ่งเป็นวงจรที่สามารถใช้เป็นตัวเก็บข้อมูลและตัวเก็บสถานะของเครื่องจักรสถานะจำกัด (FSM: Finite State Machine) ได้ การใช้สัญญาณนาฬิการ่วมนี้ อาจเกิดปัญหาในการออกแบบวงจร เช่น ปัญหาการเหลื่อมของสัญญาณนาฬิกา (Clock skew) ปัญหาการกำหนดจังหวะ (Timing) และปัญหาการแทรกแซงของคลื่นแม่เหล็กไฟฟ้า (EMI: Electro Magnetic Interference) เป็นต้น

นอกจากปัญหาที่มีในการออกแบบวงจรสมวารดังที่ได้กล่าวแล้ว ข้อเสียหนึ่งของการออกแบบวงจรสมวาร คือ การใช้ความถี่สัญญาณนาฬิกาจะมีความสัมพันธ์กับเส้นทางวิกฤต (Critical path) กล่าวคือ หากเส้นทางวิกฤตมีความยาวมากขึ้น ความถี่ของสัญญาณนาฬิกาที่ใช้จะน้อยลง เนื่องจากเรจิสเตอร์ที่ใช้เก็บข้อมูลจากวงจรเชิงผสม (Combinational circuit) จำเป็นต้องรอการทำงานของเส้นทางวิกฤตให้เสร็จสิ้นก่อน จึงสามารถเก็บข้อมูลครั้งต่อไปได้

ด้วยข้อจำกัดจากเส้นทางวิกฤตที่พบในการออกแบบวงจรสมวารนี้ ทำให้เป็นตัวกำหนดความเร็วของวงจรโดยรวม ซึ่งสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ เป็นวงจรดิจิทัลประเภทหนึ่งที่มีขนาดใหญ่ เนื่องจากประกอบด้วยหน่วยประมวลผลย่อยเป็นจำนวนมาก ดังนั้นการวางตำแหน่ง (Placement) และการจัดเส้นทาง (Routing) จึงมีความสำคัญต่อสัญญาณนาฬิการ่วม ซึ่งแตกต่างจากการออกแบบวงจรอสมวาร (Asynchronous circuit) ที่ใช้สัญญาณร้องขอ (Request signal) และสัญญาณตอบรับ (Acknowledgement signal) ในการสื่อสารกันระหว่างวงจร โดยวงจรอสมวารจะให้ประสิทธิภาพทางด้านความเร็วในระดับค่าเฉลี่ย (Average case) ซึ่งดีกว่าวงจรสมวารที่ให้ประสิทธิภาพในระดับต่ำสุด (Worst case) และมีการบริโภคพลังงานที่ต่ำกว่า [5] จึงมีงานวิจัยที่นำเสนอสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้แบบ

อสมวารที่มีโครงสร้างแตกต่างกันออกไป ผลการทดสอบจากงานวิจัยของจาง (Zhang) และคณะ [11] พบว่า วงจรที่พัฒนาขึ้นมีความเร็วสูงและใช้พลังงานต่ำกว่าเมื่อเทียบกับแบบสมวาร

สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ถูกพัฒนาบนโครงสร้างที่แตกต่างกัน เช่น โครงสร้างแบบแถวลำดับ 1 มิติ (1D array) [2] โครงสร้างแบบแถวลำดับ 2 มิติ (2D array) [11] โครงสร้างแบบตาข่าย (Mesh) [6] เป็นต้น ซึ่งสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อาจต้องได้รับการจัดการจากระบบภายนอก ในการรับโครงแบบที่ถูกนำเข้ามาประมวลผลและส่งผลลัพธ์กลับไป และมีงานวิจัยที่เสนอแนวคิดในการจัดการโครงแบบสำหรับสถาปัตยกรรมที่มีโครงสร้างแบบ 2 มิติ [3],[8],[9] แต่งานวิจัยเหล่านี้ได้เสนอเป็นเพียงแบบจำลองเท่านั้น จากสิ่งที่ได้กล่าวนี้รวมกับข้อดีของวงจรอสมวารที่ได้กล่าวไปข้างต้น งานวิจัยนี้จึงมีแนวคิดในการออกแบบตัวควบคุม (Controller) ที่เป็นฮาร์ดแวร์เพื่อจัดการโครงแบบสำหรับการประมวลผลบนสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้แบบอสมวาร โดยเลือกสถาปัตยกรรมที่มีโครงสร้างแบบแถวลำดับ 2 มิติ ซึ่งเป็นโครงสร้างที่ซับซ้อนน้อย เพื่อเป็นสถาปัตยกรรมต้นแบบสำหรับการออกแบบตัวควบคุมในงานวิจัยนี้

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อออกแบบตัวควบคุมสำหรับแถวลำดับคำนวณ (Computing array) ที่ปรับเปลี่ยนโครงแบบบางส่วน (Partial reconfiguration) ได้อย่างหยาบแบบอสมวาร

1.3 ขอบเขตของการวิจัย

1) ออกแบบแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบบางส่วนได้อย่างหยาบแบบอสมวารที่มีคุณสมบัติดังต่อไปนี้

1.1 หน่วยประมวลผลย่อยมีขนาดอย่างต่ำ 8 บิต

1.2 หน่วยประมวลผลย่อยสามารถรับข้อมูลขาเข้าได้อย่างน้อย 2 ข้อมูล

1.3 หน่วยประมวลผลย่อยมีฟังก์ชันการทำงานอย่างน้อย คือ แอนด์ ออร์ เอ็กซ์ครูซีฟออร์ ตัวตรวจสอบศูนย์ ตัวตรวจสอบหนึ่ง วงจรเลื่อน วงจรบวก วงจรลบ และตัวส่งต่อข้อมูล

1.4 สายเชื่อมโยงข้อมูลระหว่างหน่วยประมวลผลย่อยมีลักษณะแบบเนียร์เรสต์เนเบอร์ (NN: Nearest Neighbor) คือ สามารถเชื่อมต่อกับหน่วยประมวลผลย่อยที่อยู่ด้านบน ด้างล่าง ด้านซ้าย และด้านขวา ได้เป็นอย่างน้อย

1.5 มิติของแถวลำดับคำนวณมีขนาดอย่างต่ำ 8X8

1.6 ออกแบบให้เป็นวงจรอสมวารแบบรางคู่ (Dual rail)

2) โครงแบบที่นำมาประมวลผลมีคุณสมบัติดังต่อไปนี้

2.1 โครงแบบที่นำมาประมวลผลมีลักษณะเป็นสี่เหลี่ยมมุมฉาก (Rectangular task) เป็นอย่างน้อย

2.2 มีขนาดเท่ากับหรือน้อยกว่ามิติของแถวลำดับคำนวณที่ออกแบบไว้

2.3 จำนวนข้อมูลขาเข้าของแต่ละตัวดำเนินการในโครงแบบต้องสอดคล้องกับจำนวนข้อมูลขาเข้าที่ได้ออกแบบไว้ในข้อ 1.2 คือ จำนวนข้อมูลขาเข้าต้องไม่มากกว่าจำนวนที่หน่วยประมวลย่อยสามารถรับได้

2.4 ขนาดของข้อมูลที่นำมาประมวลผลมีขนาด 8 บิต เป็นอย่างน้อย

3) ออกแบบตัวกำหนดการ (Scheduler) ด้วยขั้นตอนวิธี (Algorithm) ที่แตกต่างกันอย่างน้อย 2 วิธี

4) ทดสอบประสิทธิภาพเปรียบเทียบกับการออกแบบวงจรแบบสมวารโดยการจำลองการทำงานของวงจรในระดับพฤติกรรมเป็นอย่างน้อย

1.4 ข้อจำกัดของการวิจัย

ในการทำวิจัยนี้มีข้อจำกัด คือ ซอฟต์แวร์ที่ใช้ในการทำวิจัยต้องการใช้พื้นที่หน่วยความจำของเครื่องคอมพิวเตอร์เป็นจำนวนมาก และเกิดความผิดพลาดของซอฟต์แวร์ระหว่างการทดสอบ จึงทำให้สามารถทดสอบด้วยข้อมูลในปริมาณที่จำกัด รวมถึงยังขาดเครื่องมือที่ช่วยออกแบบในลักษณะแผนภาพ (Schematic) ซึ่งจะช่วยให้การดำเนินการวิจัยเป็นไปอย่างมีประสิทธิภาพและรวดเร็วขึ้น

1.5 คำจำกัดความที่ใช้ในการวิจัย

คำจำกัดความที่ใช้สื่อความหมายในการนำเสนองานวิจัยมีดังต่อไปนี้

1) สัญญาณแทรกแซง (Interference signal) หมายถึง สัญญาณภายในที่ถูกส่งให้อาร์บิเตอร์ (Arbiter) ตลอดเวลา เพื่อใช้ในการตรวจสอบการมาของสัญญาณที่ต้องการ

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้แนวทางในการออกแบบตัวควบคุมและแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบบางส่วนได้อย่างหยาบแบบอสมวาร

2) ได้ประสบการณ์ในการออกแบบวงจรอสมวารที่มีความซับซ้อน

1.7 วิธีดำเนินการวิจัย

1) ศึกษาแบบจำลองที่ใช้ในการออกแบบตัวควบคุมและสถาปัตยกรรมที่ใช้ในการออกแบบแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบได้

2) ออกแบบโครงสร้างของตัวควบคุมและแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบได้

3) ศึกษาวิธีการออกแบบวงจรอสมวาร

4) ออกแบบตัวควบคุมและแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบได้แบบอสมวาร

5) ทดสอบตัวควบคุมและแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบได้ด้วยการจำลองการทำงาน

6) สรุปผลและเรียบเรียงวิทยานิพนธ์

1.8 ลำดับขั้นตอนในการเสนอผลการวิจัย

1) กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้องกับสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ โดยจะกล่าวถึงข้อดี ข้อเสีย และการนำแนวคิดของงานวิจัยที่เกี่ยวข้องมาใช้ในงานวิจัยนี้

2) เสนอการออกแบบแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบบางส่วนได้แบบอสมวาร

3) เสนอการออกแบบตัวควบคุมที่ใช้จัดการการส่งโครงแบบไปยังแถวลำดับคำนวณที่ได้ออกแบบไว้ในงานวิจัยนี้

4) เสนอผลการทดสอบของตัวควบคุมและแถวลำดับคำนวณที่ได้ออกแบบไว้ในงานวิจัยนี้

5) กล่าวถึงการสรุปผลการวิจัย ข้อเสนอแนะ และการพัฒนาต่อยอดจากงานวิจัยนี้

1.9 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

1) “A design of Asynchronous Double Grain Reconfigurable Computing Array”, โดย ปกรณ์ ทู้ไพเราะ, อาทิตย์ ทองทักษ์, ในงานประชุมวิชาการ 13th international Annual Symposium on Computational Science and Engineering ณ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ กรุงเทพมหานคร ประเทศไทย ระหว่างวันที่ 25-27 มีนาคม พ.ศ. 2551

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

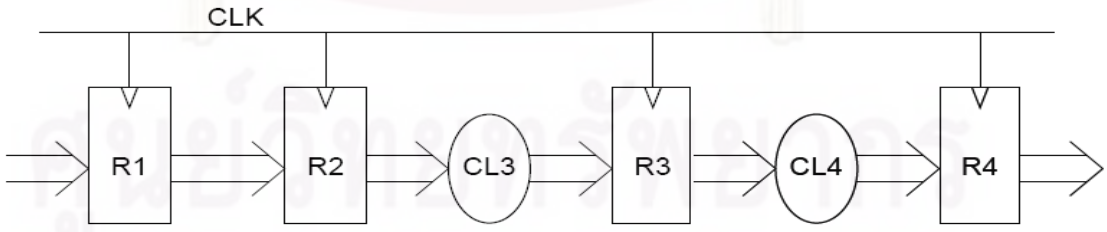
ในบทนี้จะกล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง ส่วนแรกจะกล่าวถึง วงจรอสมวาร การออกแบบวงจรอสมวารด้วยวิธีโครงสร้างการไหลของข้อมูลแบบสถิต (Static Data-flow Structures) [1] และสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ ส่วนที่สองจะกล่าวถึง งานวิจัยที่เกี่ยวข้องกับสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้แบบอสมวารและงานวิจัยที่เสนอแบบจำลองเพื่อจัดการโครงแบบ โดยมีรายละเอียดดังต่อไปนี้

2.1 ทฤษฎีที่เกี่ยวข้อง

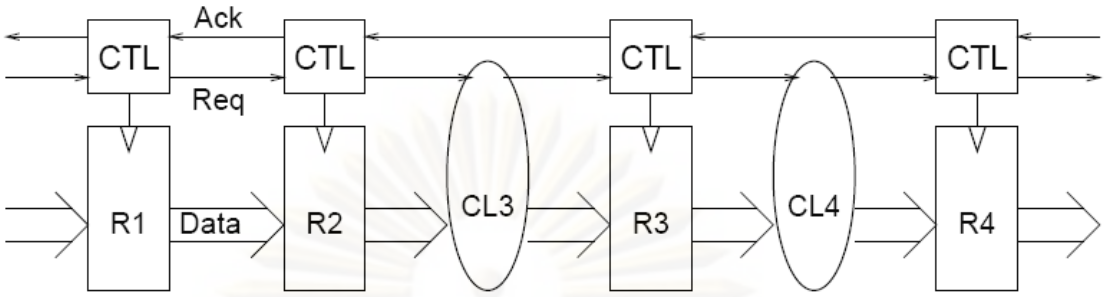
ในส่วนนี้จะอธิบายทฤษฎีที่เกี่ยวข้องกับงานวิจัย คือ วงจรอสมวาร การออกแบบวงจรอสมวารด้วยวิธีการไหลของข้อมูลแบบสถิต และสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ โดยมีรายละเอียดดังต่อไปนี้

2.1.1 วงจรอสมวาร (Asynchronous circuit)

วงจรอสมวารเป็นวงจรที่ไม่ใช้สัญญาณนาฬิการ่วมในการกำหนดจังหวะในการทำงานของวงจร แต่ใช้สัญญาณร้องขอและสัญญาณตอบรับในการส่งข้อมูลกันระหว่างวงจรแต่ละส่วน



รูปที่ 2.1 การทำงานของวงจรสมวาร [1]



รูปที่ 2.2 การทำงานของวงจรอสมวาร [1]

จากรูปที่ 2.1 แสดงการทำงานของวงจรสมวารโดยใช้สัญญาณนาฬิการ่วมในการกำหนดจังหวะการบันทึกผลลัพธ์ที่ได้จากวงจรเชิงผสม CL ให้กับเรจิสเตอร์ R ในจังหวะเดียวกัน โดยรอให้วงจรเชิงผสมที่มีเส้นทางวิกฤตทำงานเสร็จสิ้นก่อน แล้วจึงเก็บผลลัพธ์พร้อมกัน ส่วนการทำงานของวงจรอสมวาร แสดงได้ดังรูปที่ 2.2 จะทำงานโดยใช้สัญญาณร้องขอและตอบรับกำหนดเวลาที่เหมาะสมสำหรับวงจรเชิงผสมแต่ละตัวในการบันทึกข้อมูลให้กับเรจิสเตอร์โดยไม่ขึ้นอยู่กับเวลาที่ใช้ของวงจรเชิงผสมอื่น โดยมีตัวอย่างข้อดีของวงจรอสมวารมีดังนี้ [5]

2.1.1.1 **การแกว่งของสัญญาณนาฬิกา** (Clock skew)

การแกว่งของสัญญาณนาฬิกา คือ การที่สัญญาณนาฬิกามาถึงส่วนต่างๆ ของวงจรด้วยเวลาที่แตกต่างกันทำให้เกิดปัญหาในการทำงานในวงจรสมวาร แต่การออกแบบวงจรอสมวารไม่มีการใช้สัญญาณนาฬิการ่วมในการกำหนดจังหวะของวงจรโดยรวม จึงไม่ทำให้เกิดปัญหานี้กับวงจรอสมวาร

2.1.1.2 **การบริโภคพลังงาน** (Energy consumption)

เนื่องจากวงจรสมวารใช้สัญญาณนาฬิการ่วมในการกำหนดจังหวะการทำงานวงจรต่างๆ ภายในอาจมีโอกาสถูกกระตุ้นให้ทำงานได้ แม้ไม่อยู่ในช่วงเวลาที่ต้องการใช้งานก็ตาม จึงทำให้สิ้นเปลืองพลังงานไปกับวงจรเหล่านั้น แต่อย่างไรตามในการออกแบบวงจรสมวารก็มีกลไลในการแก้ปัญหานี้ เช่น การใช้สัญญาณเปิดทาง (Enable) เป็นต้น แต่การทำงานของวงจรอสมวารจะมีการเปลี่ยนแปลงสัญญาณภายในเฉพาะเมื่อวงจรส่วนนั้นถูกร้องขอเท่านั้น

2.1.1.3 **ประสิทธิภาพของวงจร** (Circuit performance)

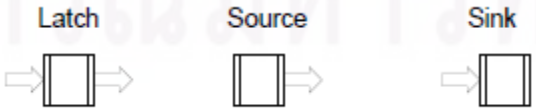
ในการออกแบบวงจรสมวารจะใช้เส้นทางวิกฤตในการกำหนดความถี่ของสัญญาณนาฬิกา ด้วยเหตุผลนี้จึงส่งผลให้วงจรเชิงผสมที่มีเส้นทางสั้นกว่าต้องรอการทำงานของวงจรที่มีเส้นทางวิกฤตให้ทำงานเสร็จสิ้นก่อน จึงสามารถเริ่มต้นทำงานต่อไปได้ ส่งผลให้ได้ความเร็วในระดับค่าต่ำสุด ส่วนวงจรอสมวารสามารถทำงานได้อย่างอิสระโดยไม่ขึ้นอยู่กับระยะทางของเส้นทางอื่น โดยใช้สัญญาณร้องขอและตอบรับสื่อสารกันระหว่างวงจรแต่ละส่วน จึงได้ค่าความเร็วในระดับค่าเฉลี่ย

2.1.2 การออกแบบวงจรอสมวารด้วยวิธีโครงสร้างการไหลของข้อมูลแบบสถิต (Asynchronous circuit design using Static Data-flow Structures)

การออกแบบวงจรอสมวารสามารถออกแบบได้หลายระดับ เช่น ระดับสูง โดยใช้เครื่องมือในการช่วยสังเคราะห์ (Synthesize) พฤติกรรมออกมาเป็นวงจรตามต้องการ การออกแบบระดับการถ่ายโอนเรจิสเตอร์ (RTL: Register Transfer Level) การออกแบบระดับประตูสัญญาณ (Gate level) และระดับทรานซิสเตอร์ (Transistor) เป็นต้น

ในส่วนนี้จะอธิบายถึงการออกแบบวงจรอสมวารด้วยวิธีโครงสร้างการไหลของข้อมูลแบบสถิต วิธีนี้สามารถออกแบบวงจรอสมวารได้ทั้งวงจรเชิงผสมและวงจรเชิงลำดับ การออกแบบวงจรอสมวารด้วยวิธีนี้สามารถเทียบเคียงได้กับการออกแบบวงจรสมวารในระดับการถ่ายโอนเรจิสเตอร์ กล่าวคือ การออกแบบระดับนี้จะไม่ระบุถึงรายละเอียดในการออกแบบด้วยเทคโนโลยีของฮาร์ดแวร์แต่อย่างใด แต่จะคำนึงถึงการไหลของข้อมูลระหว่างมอดูล (Module) โดยสามารถนำส่วนประกอบต่างๆ ดังต่อไปนี้มาสร้างเป็นวงจรอสมวารได้ตามต้องการ

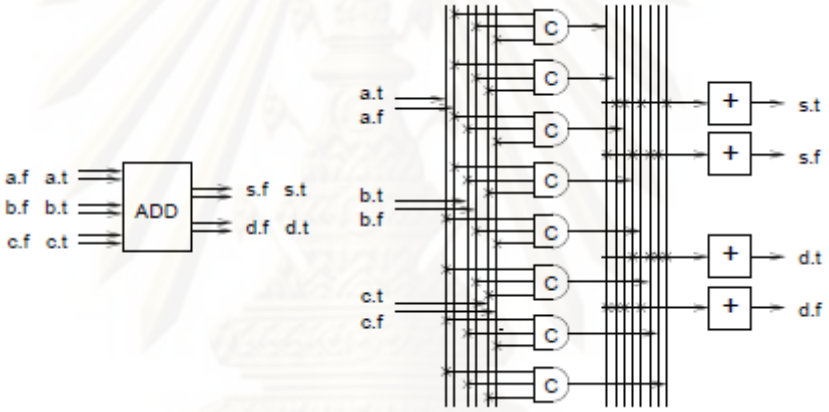
1) แลตช์ (Latch) ทำหน้าที่เก็บข้อมูลสำหรับตัวแปรต่างๆ และสร้างสัญญาณแฮนด์เชค (Handshake) ของโพรโทคอล (Protocol) ในการสื่อสาร เช่น โพรโทคอลส่งข้อมูลรวมชุด 4 เฟส (4-phase bundle data) เป็นต้น แสดงแลตช์ได้ดังรูปที่ 2.3



รูปที่ 2.3 แลตช์ [1]

จากรูปที่ 2.3 แสดงถึงแลตช์ 3 ประเภท คือ แลตช์ที่รับและส่งข้อมูล แลตช์ที่ส่งค่าคงที่ และแลตช์ที่ทำหน้ารับข้อมูลอย่างเดียว แลตช์ประเภทสุดท้ายนี้ จะถูกใช้เมื่อต้องการนำโทเค็น (Token) ออกจากระบบ กล่าวคือ แลตช์ประเภทนี้จะรับข้อมูลเข้ามาอย่างเดียวโดยข้อมูลไม่ได้ถูกส่งต่อไปวงจรส่วนอื่น

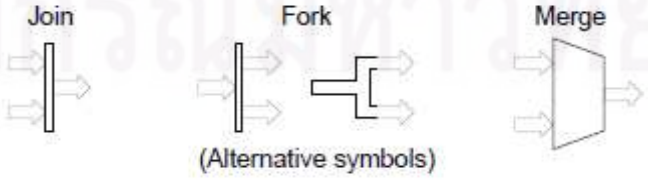
2) ฟังก์ชันบล็อก (Function block) ส่วนประกอบนี้สามารถเทียบเคียงได้กับวงจรเชิงผสมในวงจรสมวาร ฟังก์ชันบล็อกจะไม่สร้างโทเค็นให้กับระบบ แต่ทำหน้าที่ประมวลผลข้อมูลเช่นเดียวกับวงจรเชิงผสมแบบสมวาร แสดงได้ดังรูปที่ 2.4



รูปที่ 2.4 การออกแบบด้วยวิธีดิมส์ (DIMS: Delay Insensitive Minterm Synthesis) [1]

จากรูปที่ 2.4 แสดงการออกแบบวงจรบวกด้วยวิธีดิมส์ ซึ่งจะใช้ซีอีลีเม็นต์ (C-element) และประตูสัญญาณออร์นำมาเชื่อมต่อกันจนได้เป็นวงจรบวกดังรูป

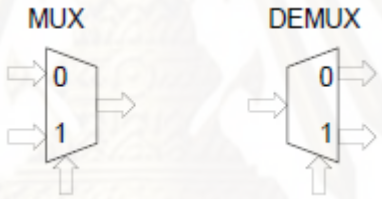
ตัวควบคุมการไหลแบบไม่มีเงื่อนไข (Unconditional flow control) ส่วนประกอบนี้จะทำหน้าที่ควบคุมการไหลของข้อมูลอย่างไม่มีเงื่อนไข แสดงได้ดังรูปที่ 2.5



รูปที่ 2.5 ตัวควบคุมการไหลแบบไม่มีเงื่อนไข 3 ประเภท [1]

จากรูปที่ 2.5 แสดงตัวควบคุมการไหลแบบไม่มีเงื่อนไข 3 ประเภท ได้แก่ ตัวเชื่อม (Join) ตัวกระจาย (Fork) และตัวผสาน (Merge) การทำงานของตัวเชื่อมจะนำข้อมูลอย่างน้อย 2 ข้อมูลเชื่อมต่อกันเป็นข้อมูลเดียว เช่นเดียวกับการนำข้อมูลหลายบิตมาเชื่อมต่อเป็นสัญญาณเดียวกันในวงจรสมวาร ตัวกระจายทำหน้าที่นำข้อมูลหนึ่งตัวมาสำเนาแล้วส่งไปยังส่วนประกอบปลายทางมากกว่า 1 ตัว ส่วนตัวผสานทำหน้าที่รับสัญญาณจาก 2 แหล่งขึ้นไป สามารถเทียบเคียงได้กับการใช้ประตูสัญญาณออร์ในวงจรสมวาร การใช้ตัวผสานในการออกแบบควรระมัดระวังเรื่องการชนกันของสัญญาณขาเข้า เนื่องจากตัวผสานจะยอมให้สัญญาณมากกว่า 1 สัญญาณไหลผ่านไปได้พร้อมกัน จึงทำให้เกิดความผิดพลาดในการส่งสัญญาณได้

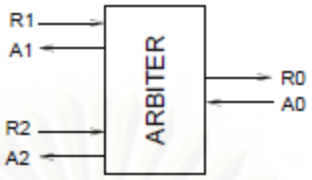
4) ตัวควบคุมการไหลแบบมีเงื่อนไข (Conditional flow control) ส่วนประกอบนี้จะทำหน้าที่ควบคุมการไหลของสัญญาณอย่างมีเงื่อนไข แสดงได้ดังรูปที่ 2.6



รูปที่ 2.6 ตัวควบคุมการไหลแบบมีเงื่อนไข [1]

จากรูปที่ 2.6 แสดงตัวควบคุม 2 ประเภท คือ อุปกรณ์รวมส่งสัญญาณ (MUX: Multiplexer) และอุปกรณ์แยกส่งสัญญาณ (DEMUX: Demultiplexer) อุปกรณ์ทั้ง 2 ตัวนี้จะทำหน้าที่เหมือนกับอุปกรณ์รวมส่งสัญญาณและอุปกรณ์แยกส่งสัญญาณในวงจรสมวาร

5) อาร์บิเตอร์ (Arbiter) ส่วนประกอบนี้จะทำหน้าที่พิเศษกว่าส่วนประกอบอื่นที่ได้กล่าวมาข้างต้น คือ มีลักษณะการทำงานคล้ายกับตัวผสาน คือ รับข้อมูลจาก 2 แหล่งแล้วส่งข้อมูลครั้งละตัวออกไปยังช่องสัญญาณขาออก โดยอาร์บิเตอร์สามารถแก้ปัญหาการชนกันของสัญญาณ 2 สัญญาณที่ต้องการใช้วงจรปลายทางเดียวกันได้ ซึ่งสัญญาณที่มาเป็นอันดับ 2 จะถูกอาร์บิเตอร์เลือกส่งไปยังช่องสัญญาณขาออกต่อจากสัญญาณที่มาเป็นอันดับแรก แสดงได้ดังรูปที่ 2.7



รูปที่ 2.7 อาร์บิเตอร์ [1]

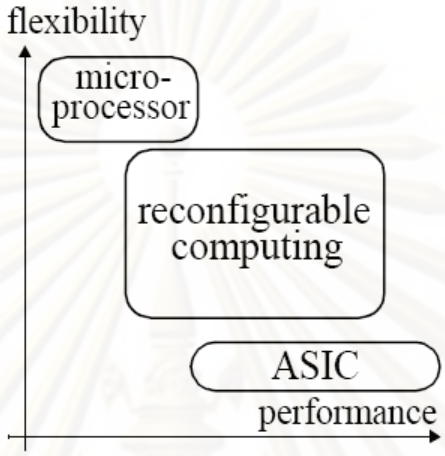
จากรูปที่ 2.7 แสดงถึงอาร์บิเตอร์ที่มีจำนวนสัญญาณข้อมูลขาเข้า 2 สัญญาณ ทั้งข้อมูลขาเข้าและขาออกจะประกอบด้วยสัญญาณร้องขอ R และสัญญาณตอบรับ A ในแต่ละครั้งที่สัญญาณขาเข้าถูกเลือกเข้ามา อาบิเตอร์จะส่งสัญญาณตอบรับกลับไปให้เฉพาะสัญญาณขาเข้าที่ถูกเลือกส่งออกไปยังสัญญาณขาออกเท่านั้น

ในงานวิจัยนี้ได้นำวิธีการออกแบบวงจรอสมวารด้วยวิธีการไหลของข้อมูลแบบสถิต มาเป็นวิธีการออกแบบหน่วยประมวลผลย่อย โดยโครงสร้างภายในหน่วยประมวลผลย่อยจะถูกสร้างขึ้นจากส่วนประกอบในวิธีดังกล่าวนี้ และใช้อาร์บิเตอร์เป็นอุปกรณ์ช่วยในการตรวจสอบการมาถึงของสัญญาณในตัวควบคุม

2.1.3 **สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้** (Reconfigurable architecture)

ในปัจจุบันมีงานประยุกต์ (Application) หลายประเภทที่จำเป็นต้องอาศัยหน่วยประมวลผลที่มีความเร็วสูงในการตอบสนองต่อการคำนวณ เช่น การเข้ารหัสลับ (Encryption) การประมวลผลภาพดิจิตอล (Digital image processing) และการประมวลผลสัญญาณเสียง (Audio signal processing) เป็นต้น สถาปัตยกรรมของไมโครโพรเซสเซอร์อยู่ในรูปแบบการทำงานตามลำดับของชุดคำสั่ง (Instruction set) ซึ่งถูกแปลมาจากตัวแปลโปรแกรม (Compiler) แม้ว่าความยืดหยุ่นในการคำนวณที่ได้จากระบบไมโครโพรเซสเซอร์จะอยู่ในระดับสูงด้วยความสามารถของการเขียนซอฟต์แวร์เพื่อตอบสนองต่องานด้านต่างๆ แต่อย่างไรก็ตามความเร็วของระบบไมโครโพรเซสเซอร์นั้นถูกจำกัดด้วยสถาปัตยกรรมที่ไม่ได้ออกแบบมาโดยเฉพาะเพื่อการคำนวณงานในแต่ละประเภท ดังนั้นการใช้ไมโครโพรเซสเซอร์จึงไม่สามารถตอบสนองงานบางอย่างได้ดีเท่าที่ควร ดังได้ยกตัวอย่างไว้ข้างต้น แนวคิดของสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้สามารถรักษาข้อดีของความยืดหยุ่นในการประมวลผลงานที่แตกต่างกันได้เมื่อเทียบกับวงจรเฉพาะกิจซึ่งถูกออกแบบมาเพื่อใช้กับงานใดงานหนึ่งโดยเฉพาะ รวมถึงการ

ปรับเปลี่ยนโครงแบบให้อยู่ในรูปแบบที่เหมาะสมกับแต่ละลักษณะงานส่งผลให้ได้ความเร็วในการประมวลผลที่สูงกว่าการใช้ไมโครโพรเซสเซอร์



รูปที่ 2.8 ประสิทธิภาพและความยืดหยุ่นของการประมวลผลที่ปรับเปลี่ยนโครงแบบได้ [6]

จากรูปที่ 2.8 แสดงถึงการเปรียบเทียบประสิทธิภาพทางด้านความเร็วและความยืดหยุ่นของไมโครโพรเซสเซอร์ การประมวลผลที่ปรับเปลี่ยนโครงแบบได้ และวงจรเฉพาะกิจ การประมวลผลที่ปรับเปลี่ยนโครงแบบได้จะให้ประสิทธิภาพและความยืดหยุ่นอยู่ระหว่างสถาปัตยกรรมทั้ง 2 แบบ เนื่องจากมีโครงสร้างของสายเชื่อมโยงข้อมูลระหว่างหน่วยประมวลผลย่อย ทำให้ประสิทธิภาพทางด้านความเร็วด้อยกว่าวงจรเฉพาะกิจซึ่งถูกออกแบบวงจรมาโดยเฉพาะสำหรับแต่ละงานประยุกต์ แต่ประสิทธิภาพจะสูงกว่าสถาปัตยกรรมแบบไมโครโพรเซสเซอร์เนื่องจากมีหน่วยประมวลผลย่อยภายในหลายตัวและสามารถประมวลผลงานประยุกต์พร้อมกันได้หลายงาน ในขณะที่จะให้ความยืดหยุ่นน้อยกว่า

สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้สามารถแบ่งตามความละเอียดของการปรับเปลี่ยนได้ 2 ประเภท คือ สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างละเอียด และสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างหยาบ โดยมีรายละเอียดดังนี้

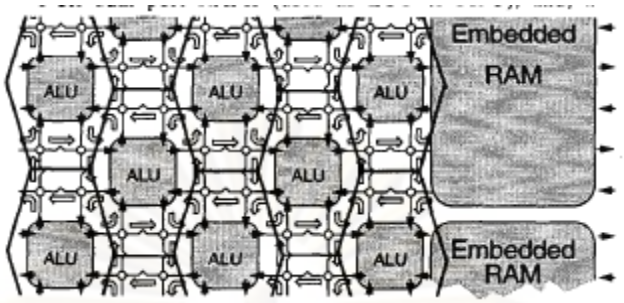
2.1.3.1 **สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างละเอียด** (Fine grain reconfigurable architecture)

สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างละเอียด สามารถปรับเปลี่ยนโครงแบบเพื่อรองรับการประมวลผลที่มีขนาดสัญญาณขาเข้าและขาออกขนาด 1 บิต หรือมากกว่าได้ สถาปัตยกรรมโดยส่วนใหญ่อยู่ในรูปแบบของตารางค้นหา (Look-up table) เช่น โครงสร้างในเอฟพีจีเอที่สามารถปรับเปลี่ยนตารางค้นหาเพื่อให้ทำหน้าที่ตามต้องการ โครงสร้างแบบตารางค้นหานี้สามารถใช้ในการออกแบบให้เป็นวงจรดิจิทัลทั่วไปได้ แต่มีข้อเสีย คือ โครงสร้างของการปรับเปลี่ยนแบบนี้ต้องใช้พื้นที่ พลังงาน และมีค่าหน่วงเวลามากกว่าสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างหยาบ

2.1.3.2 **สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างหยาบ** (Coarse grain reconfigurable architecture)

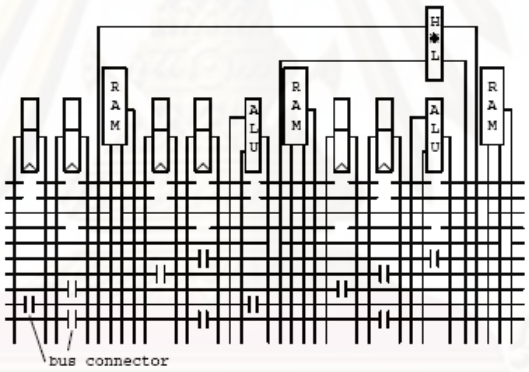
สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างหยาบถูกออกแบบมาเพื่อรองรับการประมวลผลข้อมูลที่มีขนาดใหญ่ ภายในโครงสร้างประกอบด้วยหน่วยประมวลผลย่อยและการเชื่อมต่อที่แตกต่างกันในแต่ละสถาปัตยกรรม สถาปัตยกรรมแบบนี้จะให้ความยืดหยุ่นในการปรับเปลี่ยนน้อยกว่า แต่มีประสิทธิภาพด้านพื้นที่มากกว่าเนื่องจากมีจำนวนสายเชื่อมโยงที่ใช้เชื่อมระหว่างหน่วยประมวลผลย่อยน้อยกว่า รวมถึงใช้หน่วยความจำในการเก็บโครงแบบและเวลาที่ใช้ในการปรับเปลี่ยนโครงแบบน้อยกว่าด้วย

ฮาร์เตนสไตน์ (Hartenstein) [6] ได้สำรวจสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้อย่างหยาบ โดยได้สรุปโครงสร้างไว้ 3 แบบ คือ แบบตาข่าย แบบเชิงเส้น (linear) และแบบครอสส์บาร์ (Crossbar) โครงสร้างโดยรวมแบบตาข่ายเป็นสถาปัตยกรรมในลักษณะแถวลำดับ 2 มิติ มีการเชื่อมต่อกันในลักษณะเนียร์เรสต์เนเบอร์ เช่น หน่วยประมวลผลย่อยจะถูกเชื่อมต่อกับหน่วยประมวลย่อยตัวอื่นที่อยู่ด้านบน ด้านล่าง ด้านซ้าย และด้านขวา เป็นต้น ส่วนโครงสร้างแบบเชิงเส้นมีลักษณะการเชื่อมต่อกันเป็นแถวลำดับ 1 มิติ คือ มีลักษณะการเชื่อมต่อกันของหน่วยประมวลผลย่อยเป็นเส้นตรง และแบบครอสส์บาร์ มีลักษณะของสายเชื่อมต่อที่คร่อมวางขวางกันแสดงตัวอย่างของสถาปัตยกรรมได้ดังรูปที่ 2.9 รูปที่ 2.10 และรูปที่ 2.11



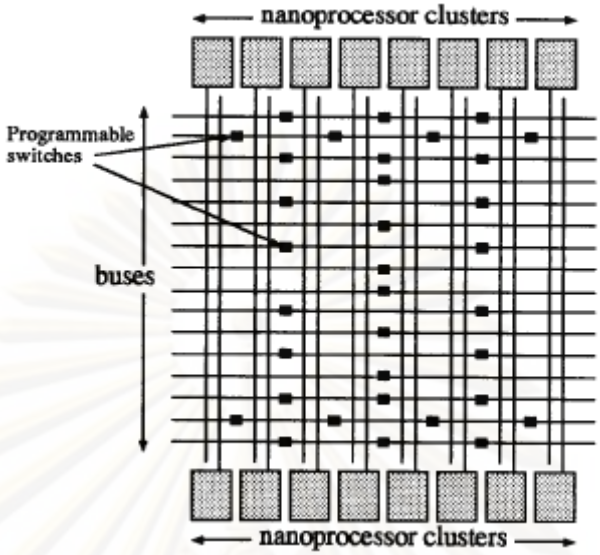
รูปที่ 2.9 โครงสร้างที่มีการเชื่อมต่อกันแบบตาข่าย [6]

ตัวอย่างโครงสร้างแบบตาข่าย เช่น เชสส์ (CHESS) [6] ประกอบด้วยหน่วยคำนวณและตรรกะ (ALU: Arithmetic-Logic Unit) และหน่วยความจำเชื่อมต่อกันในลักษณะตาข่าย แสดงได้ดังรูปที่ 2.9



รูปที่ 2.10 โครงสร้างที่มีการเชื่อมต่อกันแบบเชิงเส้น [2]

ตัวอย่างโครงสร้างแบบเชิงเส้น เช่น เรปิด (RaPID: Reconfigurable Pipelined Datapath) โครงสร้างภายในเมื่อถูกปรับเปลี่ยนโครงแบบจะทำให้ส่วนประกอบภายในถูกเชื่อมต่อกันในลักษณะเชิงเส้น แสดงได้ดังรูปที่ 2.10



รูปที่ 2.11 โครงสร้างที่มีการเชื่อมต่อกันแบบครอสส์บาร์ [10]

ตัวอย่างโครงสร้างแบบครอสส์บาร์ เช่น แพ็ดดี้ทู (PADDI2: Programmable Arithmetic Devices for high speed Digital signal processing 2) [10] โครงสร้างภายในเมื่อถูกปรับเปลี่ยนจะทำให้หน่วยประมวลผลย่อยภายในสามารถติดต่อสื่อสารกันได้ผ่านทางบัสที่ถูกวางขวางกันอยู่ แสดงได้ดังรูปที่ 2.11

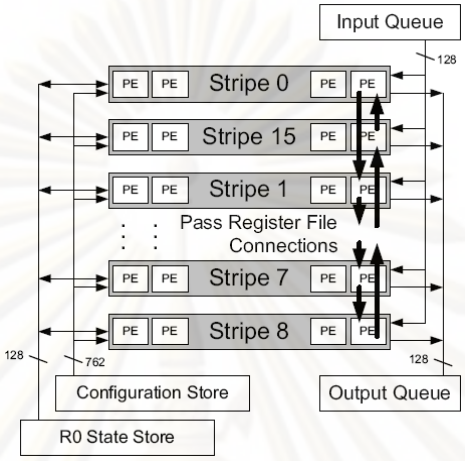
2.2 งานวิจัยที่เกี่ยวข้อง

ในส่วนนี้จะอธิบายถึงงานวิจัยที่เกี่ยวข้อง โดยแบ่งหัวข้อเป็น 2 ส่วน คือ งานวิจัยทางด้านสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้แบบอสมวารและงานวิจัยที่เสนอแบบจำลองเพื่อจัดการโครงแบบ โดยมีรายละเอียดดังต่อไปนี้

2.2.1 งานวิจัยที่เกี่ยวข้องทางด้านสถาปัตยกรรมแบบอสมวาร

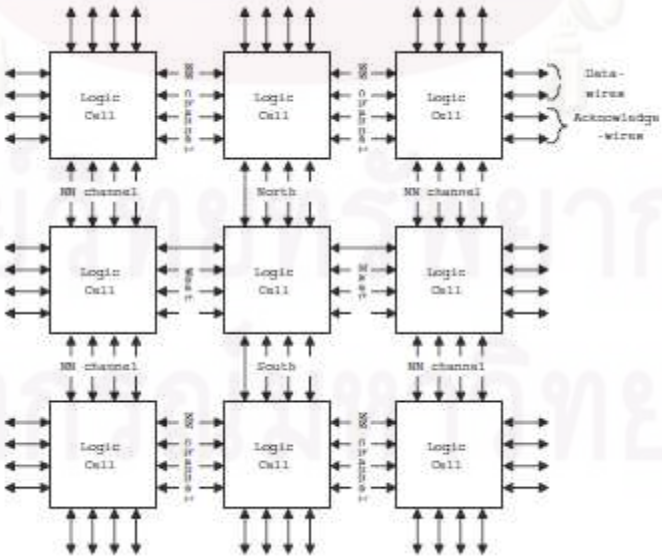
งานวิจัยของจางและคณะ [11] ได้นำเสนออาร์กา (ARCA: Asynchronous Reconfigurable Computing Array) สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้แบบอสมวาร โดยมีหน่วยประมวลผลย่อยภายในโครงสร้างเชื่อมต่อกันในลักษณะแถวลำดับ 2 มิติ ผลการทดสอบพบว่าวงจรที่พัฒนาขึ้นมีความเร็วสูงและใช้พลังงานต่ำกว่าเมื่อเทียบกับแบบสมวาร และงานวิจัยของคาโงะตานิ (Kagotani) และชมิต (Schmit) [7] ได้นำไปป์เรนช์ (PipeRench) [4] ซึ่งเดิมเป็นแบบสมวารมาออกแบบให้เป็นแบบอสมวาร ผลการทดสอบพบว่าสถาปัตยกรรมนี้ยังไม่เหมาะสมในการนำมาทำเป็นแบบอสมวารเพื่อนำไปใช้งานจริง เนื่องจากต้องจัดการกับ

และสัญญาณตอบรับซึ่งมีอยู่เป็นจำนวนมาก แสดงโครงสร้างของสถาปัตยกรรมไปป์เรนซ์ได้ดังรูปที่ 2.12



รูปที่ 2.12 โครงสร้างภายในสถาปัตยกรรมไปป์เรนซ์ [7]

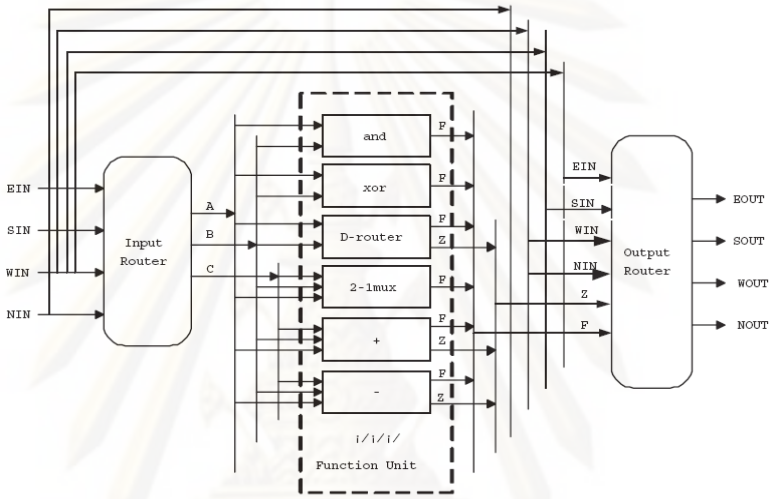
จากรูปที่ 2.12 จะเห็นได้ว่า โครงสร้างภายในสถาปัตยกรรมไปป์เรนซ์จะประกอบด้วยสเตจ (Stage) ของสายท่อ (Pipeline) อยู่ในแนวตั้ง ในแต่ละสเตจจะประกอบด้วยหน่วยประมวลผลย่อยหลายตัว โดยหน่วยประมวลผลย่อยในแต่ละสเตจสามารถติดต่อสื่อสารกันได้ทุกตัว รวมถึงสามารถติดต่อไปยังหน่วยประมวลผลย่อยในสเตจถัดไปได้ทุกตัวเช่นเดียวกัน ทำให้ต้องใช้สัญญาณร้องขอและตอบรับเป็นจำนวนมากเมื่อนำมาทำเป็นวงจรอสมวาร



รูปที่ 2.13 โครงสร้างการเชื่อมต่อของหน่วยประมวลผลย่อยของสถาปัตยกรรมอาร์กา [11]

จากรูปที่ 2.13 แสดงถึงการเชื่อมต่อของหน่วยประมวลผลย่อยภายในสถาปัตยกรรมอาร์กา โดยมีกลุ่มสัญญาณข้อมูลและกลุ่มสัญญาณตอบรับซึ่งจะติดต่อสื่อสารกันได้ใน 4 ทิศทาง คือ เหนือ ใต้ ตะวันออก และตะวันตก

ภายในหน่วยประมวลผลย่อยของสถาปัตยกรรมอาร์กาประกอบด้วย 3 ส่วน คือ ตัวจัดเส้นทางข้อมูลขาเข้า (Input router) หน่วยฟังก์ชัน (Functional unit) และตัวจัดเส้นทางข้อมูลขาออก (Output router) แสดงส่วนประกอบภายใน ได้ดังรูปที่ 2.14



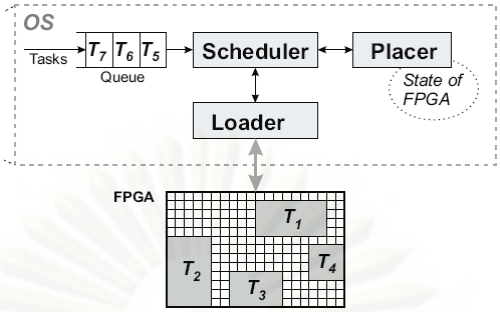
รูปที่ 2.14 โครงสร้างภายในหน่วยประมวลผลย่อยของสถาปัตยกรรมอาร์กา [11]

จากรูปที่ 2.14 ข้อมูลที่รับเข้ามาจากทั้ง 4 ทิศ จะถูกส่งให้กับตัวจัดเส้นทางข้อมูลขาเข้าและจะถูกส่งไปยังตัวจัดเส้นทางข้อมูลขาออกโดยตรงเพื่อส่งต่อข้อมูลไปยังหน่วยประมวลผลย่อยตัวอื่น ตัวจัดเส้นทางข้อมูลขาเข้าจะเลือกข้อมูล 3 ตัว เพื่อส่งให้กับหน่วยฟังก์ชัน ซึ่งทำหน้าที่ประมวลผลข้อมูลขนาด 1 บิต และส่งผลลัพธ์ไปยังตัวจัดเส้นทางข้อมูลขาออกต่อไป

งานวิจัยนี้ได้นำสถาปัตยกรรมอาร์กา มาเป็นต้นแบบในการออกแบบแถวลำดับคำนวณที่ใช้ร่วมกับการออกแบบตัวควบคุมเพื่อใช้ในการทดสอบ เนื่องจากเป็นโครงสร้างที่มีความซับซ้อนน้อยทั้งหน่วยประมวลผลย่อยและการเชื่อมต่อระหว่างกัน

2.2.2 งานวิจัยที่เสนอแบบจำลองเพื่อจัดการโครงแบบ

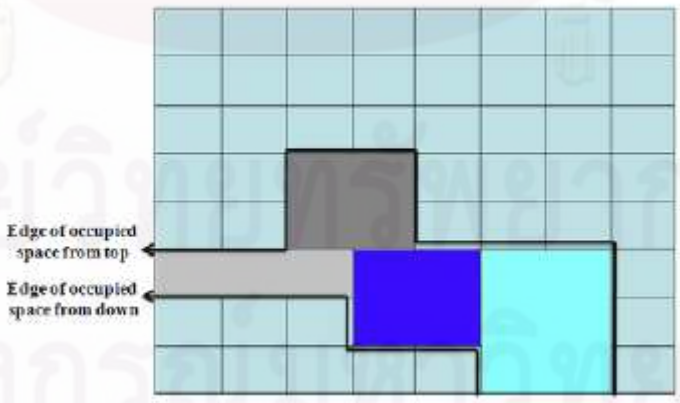
งานวิจัยของแพลตซ์เนอร์ (Platzner) และคณะ [8] ได้นำเสนอวิธีการจัดการพื้นที่ว่างในสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้โดยการทำฟังก์ชันแฮชชิง (Hashing function) เพื่อบันทึกพื้นที่ว่างที่อยู่ในลักษณะสี่เหลี่ยมลงในตารางโดยมีค่าความซับซ้อนทางเวลา (Time complexity) เป็นค่าคงที่ และได้นำเสนอแบบจำลองของระบบที่ใช้จัดการโครงแบบ โดยแบ่งเป็น 3 ส่วน คือ ตัวกำหนดการ (Scheduler) ตัววาง (Placer) และ ตัวบรรจุ (Loader) แสดงได้ดังรูปที่ 2.15



รูปที่ 2.15 แบบจำลองที่ใช้ในการจัดการโครงแบบ [8]

ในส่วนของวิธีการที่ใช้ในตัวกำหนดการแพลตซ์เนอร์และแวลเดอร์ (Walder) [9] ได้ใช้ขั้นตอนวิธี 2 วิธีที่ไม่มีการหยุดชะงัก (Non-preemptive) ของงานที่กำลังประมวลผลอยู่ คือ เอฟซีเอฟเอส (FCFS: First Come First Serve) โดยจัดลำดับของงานตามลำดับของงานที่มาถึง ส่วนอีกวิธีหนึ่ง คือ เอสเจเอฟ (SJF: Shortest Job First) จะจัดลำดับงานตามเวลาที่ใช้ในการประมวลผล โดยจะเลือกงานที่ใช้เวลาประมวลผลน้อยที่สุดก่อน และแบบที่มีการหยุดชะงัก (Preemptive) 2 วิธี คือ เอสอาร์พีที (SRPT: Shortest Remaining Processing Time) โดยเรียงลำดับของงานตามเวลาประมวลผลที่เหลืออยู่และแบบอีดีเอฟ (EDF: Earliest Deadline First) จะเรียงลำดับตามเส้นตายของงานที่ใกล้ที่สุดก่อน

งานวิจัยของเอสมาอิลดัสต์ (Esmaeildoust) และคณะ [3] ได้เสนอขั้นตอนวิธีสำหรับการวางแบบไม่ทราบงานล่วงหน้า โดยวางโครงแบบจากการพิจารณาพื้นที่ที่สามารถวางได้ที่มีจำนวนหน่วยประมวลผลรอบข้างมากที่สุด และเก็บเส้นขอบ (Edge) ที่เกิดจากการลากบนกลุ่มของโครงแบบที่ถูกวางอยู่เพื่อช่วยค้นหาตำแหน่งที่ใช้ในการวางต่อไป แสดงได้ดังรูปที่ 2.16



รูปที่ 2.16 เส้นขอบบนและขอบล่างที่ใช้ในขั้นตอนวิธีของเอสมาอิลดัสต์และคณะ [3]

จากรูปที่ 2.16 แสดงถึงเส้นขอบบน (Top-edge) และขอบล่าง (Bottom-edge) ที่ช่วยในการค้นหาพื้นที่ที่ใช้ในการวาง หากไม่สามารถวางโครงแบบบนเส้นดังกล่าวได้ จะค้นหาพื้นที่ว่างจากมุมล่างซ้ายจนถึงเส้นขอบล่าง ถ้าสามารถวางได้จะเก็บเส้นขอบบนเป็นเส้นที่ 2 (Second top-edge) เพื่อช่วยในการวางต่อไป

งานวิจัยนี้ใช้แบบจำลองของแพลตซ์เนอร์และคณะ มาเป็นแนวทางในการออกแบบโครงสร้างภายในตัวควบคุม และได้นำบางส่วนของขั้นตอนวิธีของเอสมาอิลดัสต์และคณะ คือการตรวจสอบหาหน่วยประมวลผลรอบข้างที่ถูกจอง (Reserve) ไว้ของพื้นที่สามารถวางได้มาเป็นวิธีการในการวาง

ในบทนี้ได้กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง ซึ่งจะกล่าวถึงการออกแบบแถวลำดับคำนวณและตัวควบคุมเป็นลำดับต่อไปในบทที่ 3

บทที่ 3

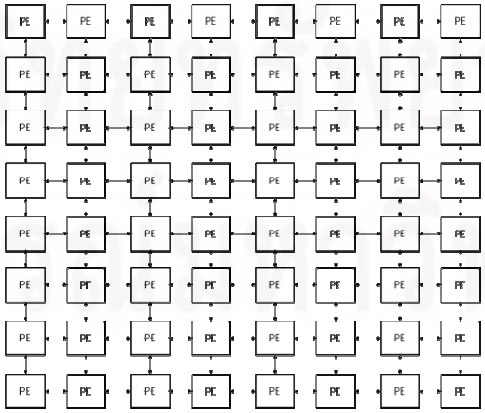
การออกแบบแถวลำดับคำนวณและตัวควบคุม

สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ที่มีโครงสร้างแบบ 2 มิติประกอบด้วยหน่วยประมวลผลย่อยที่ถูกเชื่อมต่อกันเพื่อถูกปรับเปลี่ยนเป็นวงจรที่รองรับโครงแบบที่เข้ามา ส่วนการจัดการโครงแบบเป็นหน้าที่ของตัวควบคุมซึ่งจะจัดการสิ่งต่อไปนี้ คือ การจัดลำดับของโครงแบบ การหาพื้นที่ที่เหมาะสมให้กับโครงแบบ และการบรรจุโครงแบบลงบนแถวลำดับคำนวณ

ในงานวิจัยนี้ได้ออกแบบตัวควบคุมและแถวลำดับคำนวณแบบอสมวาร โดยมีรายละเอียดดังนี้

3.1 การออกแบบแถวลำดับคำนวณ

แถวลำดับคำนวณ คือ หน่วยประมวลผลย่อยที่เชื่อมต่อกันในลักษณะแถวลำดับ (Array) โดยแถวลำดับคำนวณนี้จะถูกปรับเปลี่ยนการเชื่อมต่อเพื่อให้ประมวลผลงานประยุกต์ที่อยู่ในรูปของโครงแบบ งานวิจัยนี้ได้นำสถาปัตยกรรมอาร์กา [11] มาเป็นต้นแบบในการออกแบบให้เป็นแถวลำดับ 2 มิติ ขนาด 8X8 ที่มีลักษณะการต่อช่องสัญญาณข้อมูลแบบเนียร์เรสต์เนเบอร์ ข้อดีของการออกแบบการเชื่อมต่อแบบนี้ คือ ไม่ต้องคำนึงถึงการจัดเส้นทางโดยมีเส้นทางที่ตายตัวอยู่ 4 ทิศทาง คือ บน ล่าง ซ้าย และขวา รวมถึงสามารถลดปริมาณหน่วยความจำที่ใช้เก็บโครงแบบด้วย เมื่อเทียบกับโครงสร้างที่มีการจัดการเส้นทาง แสดงโครงสร้างของแถวลำดับคำนวณที่ออกแบบไว้ได้ดังรูปที่ 3.1

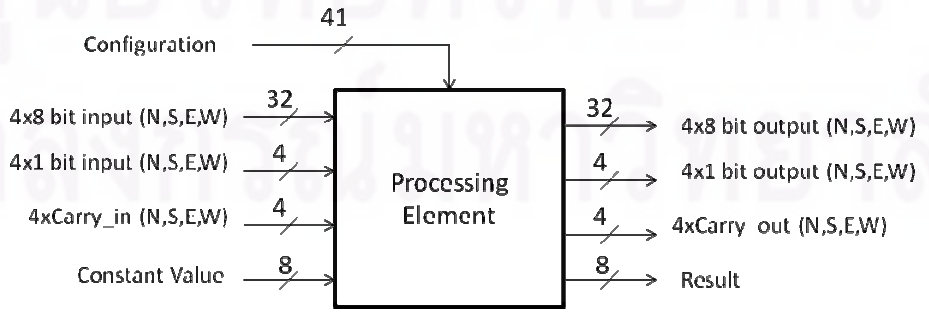


รูปที่ 3.1 โครงสร้างของแถวลำดับคำนวณแบบ 2 มิติ ขนาด 8X8

จากรูปที่ 3.1 แสดงให้เห็นถึงโครงสร้างโดยรวมของแถวลำดับคำนวณ 2 มิติที่ออกแบบในงานวิจัยนี้ ในรูปจะประกอบด้วยมอดูล PE ซึ่งแสดงถึงหน่วยประมวลผลย่อยที่ภายในมีตัวดำเนินการ (Operator) ทำหน้าที่ประมวลผลข้อมูลที่ได้รับเข้ามา หน่วยประมวลผลย่อยทุกตัวสามารถรับข้อมูลได้ทั้งจากภายนอกแถวลำดับคำนวณและจากหน่วยประมวลผลย่อยตัวอื่นที่อยู่ในแถวลำดับคำนวณ ด้วยโครงสร้างการเชื่อมต่อที่มีความซับซ้อนน้อย การเพิ่มขนาดของแถวลำดับจึงสามารถทำได้ง่าย โดยการนำหน่วยประมวลผลย่อยมาเชื่อมต่อเพิ่มในแนวนอนและแนวตั้งจนครบขนาดที่ต้องการ

3.1.1 หน่วยประมวลผลย่อย

หน่วยประมวลผลย่อยในสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ทำหน้าที่ประมวลผลสัญญาณข้อมูลที่ได้รับเข้ามา อาจถูกส่งมาจากตัวควบคุมที่จัดการโครงแบบหรือถูกส่งมาจากระบบภายนอกโดยตรง โดยหน่วยประมวลผลย่อยที่ออกแบบในงานวิจัยนี้ มีเส้นทางข้อมูล (Data path) 3 ส่วน คือ เส้นทางข้อมูลหลายบิต (Multiple bit) เส้นทางข้อมูลบิตเดี่ยว (Single bit) และเส้นทางสัญญาณตัวทด (Carry signal) ที่ใช้สำหรับการบวกและลบ การออกแบบขนาดของเส้นทางข้อมูลจะขึ้นอยู่กับความเหมาะสมกับชนิดของงานประยุกต์ที่ต้องการนำมาประมวลผล ถ้าขนาดต่ำสุดของข้อมูลมีค่ามากจะช่วยลดปริมาณสายสัญญาณในแถวลำดับคำนวณลงได้ เนื่องจากสามารถกำหนดให้หน่วยประมวลผลแต่ละตัวรองรับขนาดของงานที่มีขนาดใหญ่ได้ ในงานวิจัยนี้ได้กำหนดให้เส้นทางข้อมูลหลายบิตมีขนาด 8 บิต และเส้นทางข้อมูลบิตเดี่ยวมีขนาด 1 บิต เพื่อให้รองรับได้ทั้งข้อมูลทั่วไปและข้อมูลที่มีขนาดบิตเดียว รวมถึงเส้นทางข้อมูลบิตเดี่ยวนี้ยังสามารถถูกปรับเปลี่ยนให้เป็นสัญญาณเงื่อนไขของงานประยุกต์ขนาดหลายบิตได้อีกด้วย แสดงโครงสร้างโดยรวมของหน่วยประมวลผลย่อยได้ดังรูปที่ 3.2



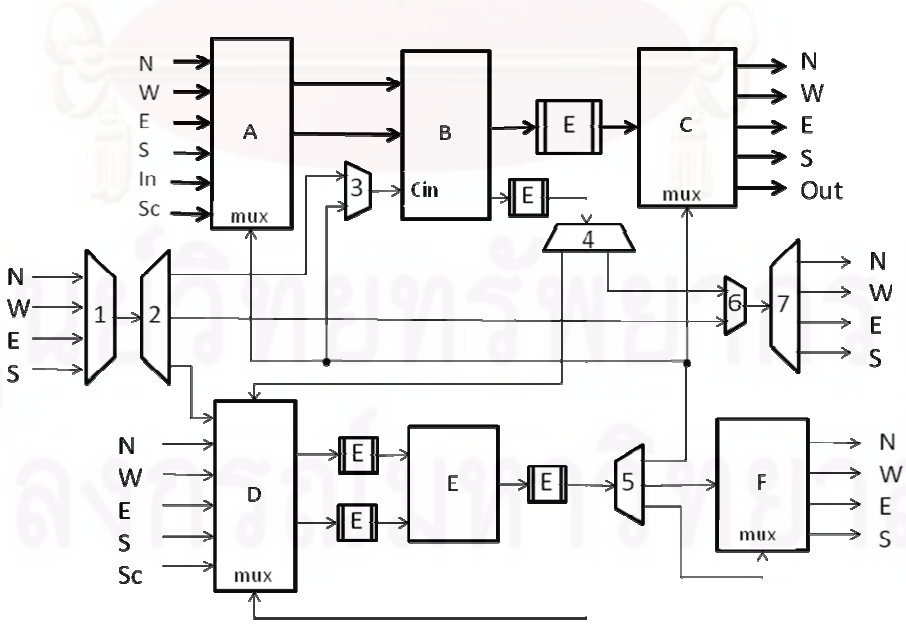
รูปที่ 3.2 โครงสร้างโดยรวมของหน่วยประมวลผลย่อย

หน่วยประมวลผลย่อยจะมีข้อมูลขาเข้าประกอบด้วย สัญญาณควบคุมโครงแบบข้อมูลที่ต้องการประมวลผลขนาด 8 บิต ขนาด 1 บิต สัญญาณตัวทด และค่าคงที่ขนาด 8 บิต สัญญาณข้อมูลขนาด 1 บิต สามารถถูกใช้เพื่อรองรับงานที่มีขนาด 1 บิต รวมถึงใช้เป็นสัญญาณเงื่อนไข (Conditional signal) ที่มีอยู่ในงานที่มีขนาด 8 บิต ส่วนสัญญาณตัวทดสามารถถูกใช้เพื่อสร้างโครงแบบให้เป็นวงจรบวกหรือลบที่มีขนาดมากกว่า 8 บิต ได้ และอาจนำไปเป็นสัญญาณขาเข้าของตัวดำเนินการที่อยู่ภายในหน่วยประมวลผลย่อยเอง ส่วนข้อมูลขาออกจากหน่วยประมวลผลย่อยประกอบด้วย สัญญาณผลลัพธ์จากการประมวลผลขนาด 8 บิต ขนาด 1 บิต สัญญาณตัวทด ซึ่งอาจเป็นสัญญาณตัวทดที่เกิดจากภายในหน่วยประมวลผลหรือเกิดจากการส่งต่อจากหน่วยประมวลผลตัวอื่น และผลลัพธ์ที่ต้องการส่งไปยังระบบที่อยู่นอกแถวลำดับคำนวณ

ต่อไปจะกล่าวถึงส่วนประกอบภายในหน่วยประมวลผลย่อยที่ได้ออกแบบไว้ โดยมีรายละเอียดดังนี้

3.1.2 **ส่วนประกอบภายในหน่วยประมวลผลย่อย**

ภายในหน่วยประมวลผลย่อยประกอบด้วย 3 เส้นทางหลัก คือ เส้นทางข้อมูล 8 บิต เส้นทางข้อมูล 1 บิต และเส้นทางสัญญาณตัวทด โดยแต่ละเส้นทางจะมีการส่งสัญญาณเข้าหากันเป็นบางส่วน แสดงได้ดังรูปที่ 3.3



รูปที่ 3.3 ส่วนประกอบภายในหน่วยประมวลผลย่อย

จากรูปที่ 3.3 สายสัญญาณข้อมูลที่มีขนาดหนากว่าจะแสดงถึงเส้นทางข้อมูลที่มีขนาด 8 บิต ส่วนสายสัญญาณข้อมูลที่มีขนาดบางจะมีขนาด 1 บิต คือ เส้นทางข้อมูลของสัญญาณตัวทดและสัญญาณข้อมูลขนาด 1 บิต มีรายละเอียดในส่วนประกอบต่างๆ ดังนี้

3.1.2.1 **ตัวเลือกข้อมูลขาเข้าขนาด 8 บิต** (8 bit input selector)

ส่วนประกอบนี้ทำหน้าที่เลือกข้อมูลขาเข้ามายังหน่วยประมวลผลจาก 4 ทิศทาง คือ บน ล่าง ซ้าย และขวา รวมถึงมีการเลือกข้อมูลจากภายนอกแถวลำดับคำนวณและจากค่าคงที่ด้วย แสดงได้ดังส่วนประกอบ A ในรูปที่ 3.3

3.1.2.2 **ตัวเลือกข้อมูลขาเข้าขนาด 1 บิต** (1 bit input selector)

ส่วนประกอบนี้จะทำหน้าที่เช่นเดียวกับตัวเลือกข้อมูลขาเข้าขนาด 8 บิต แต่ไม่มีการรับข้อมูลจากภายนอกขนาด 1 บิต แสดงได้ดังส่วนประกอบ D ในรูปที่ 3.3

3.1.2.3 **หน่วยดำเนินการขนาด 8 บิต** (8 bit operational unit)

ภายในหน่วยประมวลผลย่อยจะมีวงจรที่ทำหน้านี้ประมวลผลสัญญาณขนาด 8 บิต ซึ่งประกอบด้วยตัวดำเนินการจำนวน 14 ตัว แสดงได้ดังตารางที่ 3.1

ตารางที่ 3.1 ตัวดำเนินการขนาด 8 บิต ที่อยู่ภายในหน่วยประมวลผลย่อย

ตัวดำเนินการ	**หน้าที่**
NOP	ไม่มีการดำเนินการใดกับข้อมูล
NEG	สลับค่าบิต
ROL	หมุน (Rotate) ค่าข้อมูลไปทางซ้าย
ROR	หมุนค่าข้อมูลไปทางขวา
ROLWC	หมุนค่าข้อมูลไปทางซ้ายแบบมีสัญญาณตัวทด
RORWC	หมุนค่าข้อมูลไปทางขวาแบบมีสัญญาณตัวทด

AND	ฟังก์ชันแอนด์
OR	ฟังก์ชันออร์
XOR	ฟังก์ชันเอ็กครูซีฟออร์
CMP8	เปรียบเทียบค่าข้อมูล
ADD	ฟังก์ชันบวก
SUB	ฟังก์ชันลบ
CHK0	ตรวจสอบค่าศูนย์
CHK1	ตรวจสอบค่าหนึ่ง

จากตารางที่ 3.1 สามารถอธิบายได้ดังนี้ ตัวดำเนินการ NOP จะไม่มีกระบวนการใดกับข้อมูลที่เข้ามา สามารถใช้เป็นตัวส่งต่อข้อมูลได้ ตัวดำเนินการ NEG จะสลับบิตทั้ง 8 บิต ให้มีค่าตรงข้าม ตัวดำเนินการ ROL และ ROR จะหมุนข้อมูลโดยไม่มีสัญญาณตัวทด ส่วนตัวดำเนินการ ROLWC และ RORWC จะนำสัญญาณตัวทดมาหมุนด้วย นอกจากใช้หมุนค่าข้อมูลได้แล้ว ยังสามารถใช้เลื่อนข้อมูลได้โดยกำหนดค่าสัญญาณตัวทดให้เป็นศูนย์ ตัวดำเนินการ AND OR และ XOR จะทำฟังก์ชันดังที่ได้กล่าวไปในตารางที่ 3.1 โดยจะกระทำกับค่าที่ได้รับมาแบบบิตไวซ์ (Bitwise) คือ กระทำกับค่าข้อมูลแต่ละบิต ตัวดำเนินการ CMP8 จะมีผลลัพธ์ขนาด 1 บิต และสามารถส่งไปประมวลผลต่อยังเส้นทางข้อมูลขนาด 1 บิต ในหน่วยประมวลผลตัวเดียวกันได้ เพื่อนำไปสร้างสัญญาณเงื่อนไขต่อไป ฟังก์ชันบวกและลบจะมีสายสัญญาณตัวทด ซึ่งสามารถถูกส่งไปยังเส้นทางข้อมูลขนาด 1 บิต ได้เช่นเดียวกัน และสามารถถูกส่งออกไปยังหน่วยประมวลผลตัวอื่นเพื่อสร้างเป็นวงจรบวกหรือลบที่มีขนาดมากกว่า 8 บิต ได้ ตัวดำเนินการ CHK0 และ CHK1 จะตรวจสอบค่าที่ได้รับเข้ามาว่าเป็นค่า 0 หรือ ค่า 1 หรือไม่ สามารถถูกใช้เป็นส่วนหนึ่งของวงจรเงื่อนไขได้เช่นเดียวกัน แสดงได้ดังส่วนประกอบ B ในรูปที่ 3.3

3.1.2.4 **หน่วยดำเนินการขนาด 1 บิต** (1 bit functional unit)

ตัวดำเนินการสำหรับข้อมูลขนาด 1 บิต ในหน่วยประมวลผลมีจำนวน 5 ตัว แสดงได้ดังตารางที่ 3.2

ตารางที่ 3.2 ตัวดำเนินการขนาด 1 บิต ที่อยู่ภายในหน่วยประมวลผลย่อย

ตัวดำเนินการ	**หน้าที่**
NOP	ไม่มีการดำเนินการใดกับข้อมูล
NEG	สลับค่าบิต
AND	ฟังก์ชันแอนด์
OR	ฟังก์ชันออร์
XOR	ฟังก์ชันเอ็กครูซีฟออร์

ตัวดำเนินการในส่วนประกอบนี้จะประมวลผลสัญญาณขนาด 1 บิต ซึ่งสามารถถูกปรับเปลี่ยนให้ใช้เป็นเส้นทางข้อมูลของสัญญาณเงื่อนไขได้ แสดงได้ดังส่วนประกอบ E ในรูปที่ 3.3

3.1.2.5 **ตัวจัดเส้นทางข้อมูลขาออกขนาด 8 บิต** (8 bit output router)

ส่วนประกอบนี้ถูกออกแบบให้สามารถเลือกกระจายข้อมูลได้ครั้งละ 1 หรือ 2 ทิศทาง โดยมีช่องผลลัพธ์ที่ถูกส่งออกไปนอกแถวลำดับเป็นทิศทางหนึ่งด้วย หากต้องการส่งมากกว่า 2 ทิศทางหรือหน่วยประมวลผลปลายทางอยู่ในตำแหน่งที่ไม่สามารถส่งไปถึงได้ จะต้องใช้หน่วยประมวลผลย่อยที่อยู่ติดกันเป็นตัวส่งต่อข้อมูลแล้วกระจายข้อมูลนั้นต่อไป แสดงได้ดังส่วนประกอบ C ในรูปที่ 3.3

3.1.2.6 **ตัวจัดเส้นทางข้อมูลขาออกขนาด 1 บิต** (1 bit output router)

ส่วนประกอบนี้ทำหน้าที่เช่นเดียวกับตัวจัดเส้นทางข้อมูลขาออกขนาด 8 บิต แต่ไม่มีการส่งข้อมูลไปยังช่องผลลัพธ์ขนาด 1 บิต เนื่องจากไม่ได้ออกแบบให้มีช่องสัญญาณนี้ แสดงได้ดังส่วนประกอบ F ในรูปที่ 3.3

3.1.2.7 **ตัวผสานสัญญาณตัวทดขาเข้า** (Input carry merger)

ส่วนประกอบนี้จะใช้ตัวผสานข้อมูลรับสัญญาณตัวทดจาก 4 ทิศทางเข้ามายังหน่วยประมวลผลย่อย แล้วจะส่งต่อไปยังตัวจัดเส้นทางสัญญาณตัวทดต่อไป แสดงได้ดังส่วนประกอบหมายเลข 1 ในรูปที่ 3.3

3.1.2.8 **ตัวจัดเส้นทางสัญญาณตัวทด** (Carry signal router)

ส่วนประกอบนี้จะทำหน้าที่จัดเส้นทางให้กับสัญญาณตัวทดที่ถูกรับเข้ามาภายในหน่วยประมวลผล โดยออกแบบให้ส่งไปยังหน่วยดำเนินการขนาด 8 บิต ตัวเลือกข้อมูลขาเข้าขนาด 1 บิต และตัวจัดเส้นทางสัญญาณตัวทดขาออก แสดงได้ดังส่วนประกอบหมายเลข 2 ในรูปที่ 3.3

3.1.2.9 **ตัวเลือกสัญญาณตัวทด** (Carry signal selector)

ส่วนประกอบนี้จะใช้ตัวผสานข้อมูลรับสัญญาณตัวทดจาก 2 แหล่ง คือ ตัวจัดเส้นทางสัญญาณตัวทดและจากผลลัพธ์ที่ได้จากหน่วยดำเนินการข้อมูลขนาด 1 บิต แสดงได้ดังส่วนประกอบหมายเลข 3 ในรูปที่ 3.3

3.1.2.10 **ตัวกระจายสัญญาณตัวทด** (Carry fork unit)

ส่วนประกอบนี้จะทำหน้าที่กระจายสัญญาณตัวทดไปยัง 2 ส่วน คือ ตัวเลือกข้อมูลขาเข้าขนาด 1 บิต โดยถูกส่งไปเป็นข้อมูลขาเข้าหรืออาจถูกส่งเป็นสัญญาณควบคุมสำหรับการเลือกข้อมูลขาเข้าอย่างมีเงื่อนไขก็ได้ และถูกส่งไปยังตัวผสานสัญญาณตัวทดเพื่อจัดการส่งต่อไปยังหน่วยประมวลผลอื่นต่อไป แสดงได้ดังส่วนประกอบหมายเลข 4 ในรูปที่ 3.3

3.1.2.11 **ตัวกระจายข้อมูลขาออกของแลตช์ 1 บิต** (1 bit latch output router)

ส่วนประกอบนี้สามารถกระจายข้อมูลจากแลตช์ที่รับข้อมูลขาออกจากหน่วยดำเนินการขนาด 1 บิต ไปยังส่วนประกอบอื่นภายในหน่วยประมวลผล คือ สามารถส่งไปยังตัวเลือกข้อมูลขาเข้าขนาด 8 บิต และตัวจัดเส้นทางข้อมูลขาออกขนาด 8 บิต เพื่อเป็นสัญญาณควบคุมภายในอีกหนึ่งสัญญาณในกรณีที่โครงแบบส่วนที่เป็นวงจรเลือกหรือส่งข้อมูลอย่างมีเงื่อนไข สามารถส่งไปยังหน่วยดำเนินการขนาด 8 บิต เพื่อนำสัญญาณไปเป็นสัญญาณตัวทดให้กับตัวดำเนินการบวก ตัวดำเนินการลบและกระบวนการหมุนได้ และส่งไปยังตัวจัดเส้นทาง

ข้อมูลขาออกขนาด 1 บิต เพื่อใช้เป็นผลลัพธ์ส่งไปยังหน่วยประมวลผลตัวอื่นต่อไปและเป็นสัญญาณควบคุมได้ด้วยเช่นกัน แสดงได้ดังส่วนประกอบหมายเลข 5 ในรูปที่ 3.3

3.1.2.12 **ตัวผสานสัญญาณตัวทดขาออก** (Output carry merger)

ส่วนประกอบนี้จะใช้ตัวผสานเพื่อรับสัญญาณตัวทดที่ส่งมาจากภายนอกโดยตรงผ่านทางตัวจัดเส้นทางสัญญาณตัวทดขาเข้าหรือรับจากสัญญาณตัวทดขาออกจากหน่วยดำเนินการก็ได้ แสดงได้ดังส่วนประกอบหมายเลข 6 ในรูปที่ 3.3

3.1.2.13 **ตัวจัดเส้นทางสัญญาณตัวทดขาออก** (Output carry router)

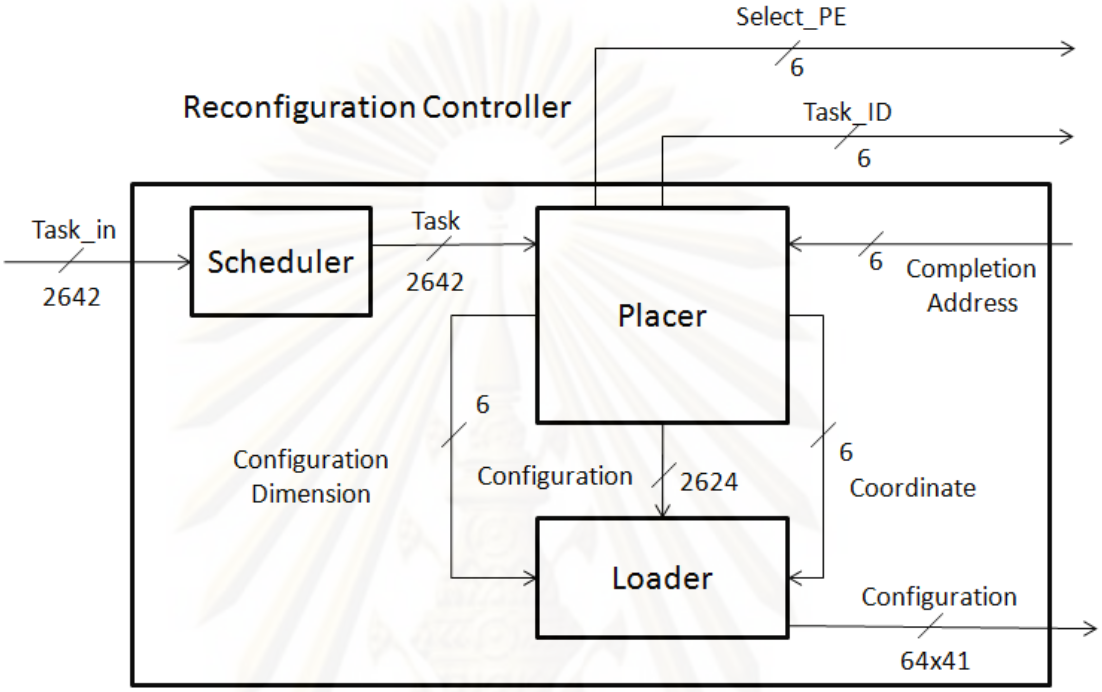
ส่วนประกอบนี้ทำหน้าที่เช่นเดียวกับตัวจัดเส้นทางข้อมูลขาออกขนาด 8 บิต และ 1 บิต โดยส่งสัญญาณตัวทดไปยังปลายทางที่ถูกเลือก แสดงได้ดังส่วนประกอบหมายเลข 7 ในรูปที่ 3.3

จากส่วนประกอบภายในจะเห็นได้ว่า มีแลตช์ขนาด 1 บิต จำนวน 4 ตัว เนื่องจากมีเส้นทางวนเกิดขึ้นภายในหน่วยประมวลผล จะต้องใช้แลตช์อย่างน้อย 3 ตัว วางอย่างอนุกรม [1] ในเส้นทางวนนั้น เพื่อให้วงจรในส่วนนี้ทำงานได้ถูกต้อง รวมถึงการใส่แลตช์เพิ่มและเปลี่ยนลำดับก่อนหลังในวงจรอสมวารจะไม่ส่งผลต่อการทำงานของวงจร รวมถึงยังทำให้เกิดเส้นทางที่เป็นแบบสายท่อ ทำให้เพิ่มประสิทธิภาพทางด้านความเร็วของวงจรได้ งานวิจัยนี้จึงได้ออกแบบให้มีแลตช์ถูกวางอยู่หลังหน่วยดำเนินการขนาด 8 บิต และ 1 บิต อย่างละ 1 ตัว เพื่อลดปริมาณการใช้หน่วยประมวลผลในขั้นตอนการแปลงวงจรอสมวารที่ถูกออกแบบโดยวิธีโครงสร้างการไหลของข้อมูลแบบสถิตมาใช้บนแถวลำดับคำนวณ

3.2 ตัวควบคุม

ในส่วนนี้จะอธิบายถึงการออกแบบตัวควบคุม ซึ่งประกอบด้วย โครงสร้างโดยรวมของตัวควบคุม การออกแบบโครงสร้างของโครงแบบ การออกแบบตัวกำหนดการ การออกแบบตัววาง และการออกแบบตัวบรรจุ โดยมีรายละเอียดดังนี้

3.2.1 โครงสร้างโดยรวมของตัวควบคุม



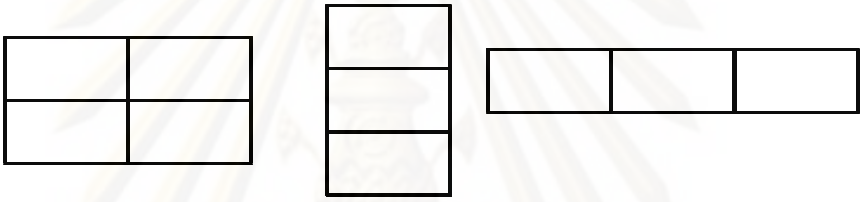
รูปที่ 3.4 โครงสร้างโดยรวมของตัวควบคุม

จากรูปที่ 3.4 แสดงโครงสร้างโดยรวมของตัวควบคุม ซึ่งประกอบด้วยส่วนประกอบภายใน 3 ส่วน คือ ตัวกำหนดการ ตัววางและตัวบรรจุ ตัวกำหนดการจะทำหน้าที่จัดลำดับโครงแบบที่เข้ามาเพื่อส่งต่อให้กับตัววาง ในงานวิจัยนี้ได้ออกแบบให้ตัวกำหนดการส่งโครงแบบไปยังตัววางได้ครั้งละ 1 โครงแบบเท่านั้น เมื่อเริ่มต้นการวางแล้วจะต้องวางทุกโครงแบบที่อยู่ในหน่วยความจำสำหรับเก็บโครงแบบที่รับมาจากภายนอกให้เสร็จสิ้นทั้งหมดก่อนการรับโครงแบบครั้งต่อไปจากภายนอกเพื่อป้องกันการตกค้างของโครงแบบที่มีขนาดใหญ่กว่าที่อาจถูกข้ามลำดับจากโครงแบบที่มีขนาดเล็กอยู่เสมอ เมื่อได้พื้นที่ที่เหมาะสมแล้วจะนำโครงสร้างของโครงแบบมาแบ่งหมายเลขประจำงานออกแล้วส่งเฉพาะส่วนที่เป็นมิติของโครงแบบ Configuration Dimension ข้อมูลสัญญาณควบคุม Configuration ที่จะถูกกระจายไปยังหน่วยประมวลผลย่อยแต่ละตัว และพิกัด Coordinate สำหรับการบรรจุโครงแบบลงบนแถวลำดับคำนวณให้กับตัวบรรจุ ส่วนหมายเลขประจำงาน Task_ID นั้นจะถูกเก็บไว้เพื่อส่งให้วงจรภายนอกสามารถรู้ได้ว่าผลลัพธ์ที่เสร็จสิ้นล่าสุดนั้นเป็นของงานใดเมื่อได้รับสัญญาณ Completion

Address คือ สัญญาณตำแหน่งหน่วยประมวลผลย่อยที่ให้ผลลัพธ์ส่งไปยังภายนอก ส่วนสัญญาณ Select_PE จะถูกส่งให้วงจรภายนอกเพื่อเลือกผลลัพธ์จากหน่วยประมวลผลย่อยตัวใดตัวหนึ่ง ซึ่งมีค่าเดียวกับสัญญาณ Completion Address ที่ได้รับเข้ามาล่าสุด

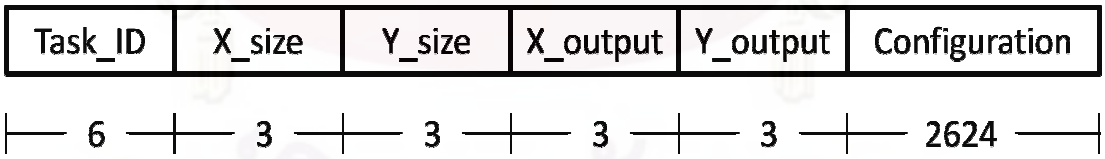
3.2.2 การออกแบบโครงสร้างของโครงแบบ

โครงแบบที่ถูกนำมาประมวลผลในงานวิจัยนี้จะต้องมีลักษณะเป็นสี่เหลี่ยมมุมฉาก หากโครงแบบต้นฉบับมีลักษณะอื่นจะต้องถูกปรับให้เป็นสี่เหลี่ยมมุมฉากก่อน แสดงลักษณะของโครงแบบสี่เหลี่ยมมุมฉากได้ดังรูปที่ 3.5



รูปที่ 3.5 โครงแบบลักษณะสี่เหลี่ยมมุมฉาก

ในงานวิจัยนี้ออกแบบให้โครงสร้างของโครงแบบมีขนาดไม่เกิน 8X8 ใช้เนื้อที่ข้อมูลในการเก็บโครงแบบเท่ากับ 2642 บิต ซึ่งประกอบด้วย 4 ส่วน แสดงได้ดังรูปที่ 3.6



รูปที่ 3.6 โครงสร้างของโครงแบบ

ในแต่ละโครงแบบจะถูกแบ่งออกเป็น 4 ส่วน โดยแต่ละส่วนมีรายละเอียดดังนี้

1) Task_ID คือ หมายเลขประจำงาน ในงานวิจัยนี้กำหนดให้มีขนาด 6 บิต ซึ่งจะรอบรับโครงแบบที่มีหมายเลขแตกต่างกันได้ 64 โครงแบบ โดยจะถูกใช้ภายหลังที่โครงแบบถูกประมวลผลเสร็จแล้ว เพื่อแนบกับผลลัพธ์ของโครงแบบและส่งต่อไปยังวงจรภายนอกต่อไป

2) X_size และ Y_size คือ ขนาดของโครงแบบในแนวนอนและแนวตั้ง

3) X_output และ Y_outout คือ พิกัดแนวนอนและแนวตั้งของโครงแบบที่จะส่งผลลัพธ์ออกมา ค่าพิกัดนี้จะถูกใช้ในตัววางเพื่อตรวจสอบว่าสัญญาณสิ้นสุด (Completion signal) ของหน่วยประมวลผลที่ได้รับเป็นของโครงแบบใด

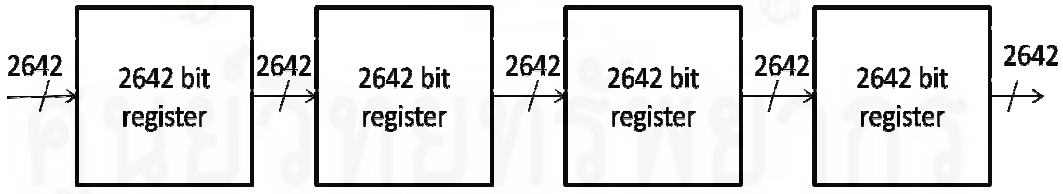
4) Configuration คือ สัญญาณควบคุมที่ถูกส่งไปยังหน่วยประมวลผลย่อยแต่ละตัว มีขนาดเท่ากับขนาดสัญญาณควบคุมของแต่ละหน่วยประมวลผลย่อยคูณด้วยจำนวนหน่วยประมวลผลทั้งหมดที่มีอยู่ในแถวลำดับคำนวณ ในงานวิจัยนี้ออกแบบให้มีจำนวนหน่วยประมวลผลย่อย 64 ตัว ซึ่งต้องใช้ Configuration ขนาด 2624 บิต

3.2.3 ตัวกำหนดการ

ในงานวิจัยนี้ได้ใช้ขั้นตอนวิธี 2 วิธี ในการออกแบบตัวกำหนดการที่แตกต่างกัน โดยมีรายละเอียดดังนี้

3.2.3.1 การออกแบบด้วยขั้นตอนวิธีแบบมาก่อนให้บริการก่อน (First Come First Serve)

การออกแบบตัวกำหนดการให้เป็นไปตามขั้นตอนวิธีนี้ จะใช้เรจิสเตอร์สายท่อ (Pipeline register) เพื่อเก็บโครงแบบที่เข้ามาสู่ตัวควบคุมแล้วส่งไปยังตัววางตามลำดับการมาถึงของโครงแบบ โดยกำหนดให้มีเรจิสเตอร์ 4 ตัว แสดงโครงสร้างของเรจิสเตอร์แบบสายท่อได้ดังรูปที่ 3.7



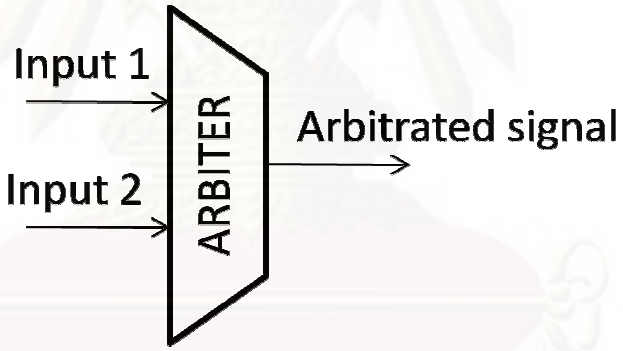
รูปที่ 3.7 เรจิสเตอร์แบบสายท่อที่ใช้เก็บโครงแบบ

จากรูปที่ 3.7 จะเห็นได้ว่า ไม่จำเป็นต้องใช้สัญญาณพร้อม (Ready) และสัญญาณร้องขอเหมือนในการออกแบบวงจรสมวารเพื่อกำหนดจังหวะในการส่งข้อมูลแต่ละครั้ง

เนื่องจากในวงจรสมวารจะมีสัญญาณร้องขอและตอบรับที่ใช้ในการสื่อสารกับวงจรที่ต้องการส่งข้อมูลอยู่แล้ว

3.2.3.2 การออกแบบด้วยขั้นตอนวิธีแบบพื้นที่น้อยที่สุดก่อน (MIAF: Minimal Area First)

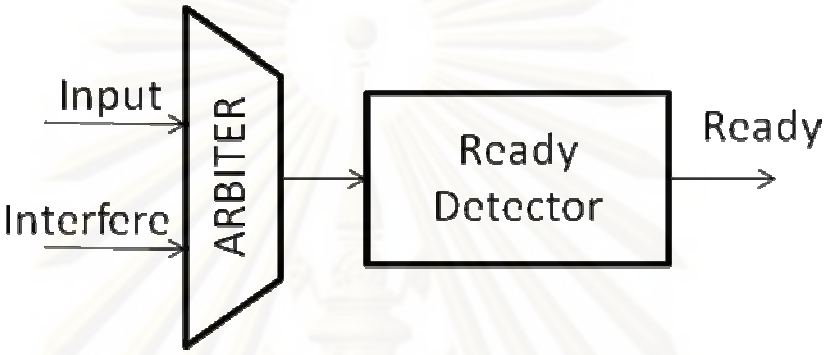
ขั้นตอนวิธีนี้จะคำนวณหาพื้นที่ของโครงแบบที่มีขนาดน้อยที่สุดแล้วส่งโครงแบบไปยังตัววาง ในการออกแบบตัวกำหนดการด้วยวิธีนี้ จะยินยอมให้ตัววางร้องขอโครงแบบ ณ เวลาใดเวลาหนึ่งได้โดยไม่ต้องรอให้หน่วยความจำเต็ม โดยมีเรจิสเตอร์ 4 ตัว เช่นเดียวกับขั้นตอนแบบมาก่อนให้บริการก่อน ตัวกำหนดการจะตรวจสอบการมาของโครงแบบและการร้องขอจากตัววาง ซึ่งในวงจรอสมวารสามารถตรวจสอบความพร้อมของสัญญาณโดยใช้สัญญาณตอบรับ แต่ในงานวิจัยนี้ได้ออกแบบวงจรในระดับการไหลของข้อมูล (Data flow) ทำให้ไม่สามารถนำสัญญาณนี้มาตรวจสอบได้ จึงต้องมีกลไกในการตรวจสอบความพร้อมของสัญญาณที่ต้องการด้วยวิธีอื่น ซึ่งอธิบายได้ดังต่อไปนี้



รูปที่ 3.8 อุปกรณ์ที่นำมาใช้ในการตรวจสอบการร้องขอจากโครงแบบและจากตัววาง

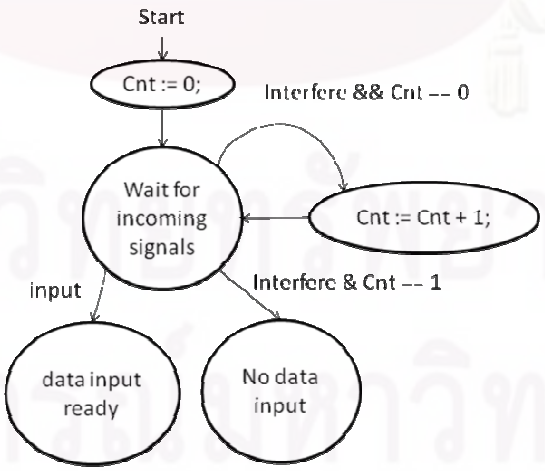
จากรูปที่ 3.8 แสดงถึงอุปกรณ์ที่ใช้ในการตรวจสอบการมาของโครงแบบและการร้องขอจากตัววาง โดยจะใช้อุปกรณ์พิเศษ คือ อาร์บิเตอร์ ซึ่งทำหน้าที่เลือกสัญญาณที่มาถึงก่อนส่งไปยังช่องสัญญาณขาออก ในเวลาที่วงจรปลายทางยังไม่ได้รับสัญญาณที่ถูกเลือกจากอาร์บิเตอร์ หากมีสัญญาณในอีกช่องสัญญาณหนึ่งที่ต้องการตรวจสอบมาถึงอาร์บิเตอร์ สัญญาณนี้จะถูกส่งออกไปหลังจากที่วงจรภายนอกได้รับสัญญาณที่ถูกเลือกก่อนหน้าเรียบร้อยแล้ว

ในการออกแบบกลไกนี้จะใช้สัญญาณอีกหนึ่งสัญญาณ ซึ่งจะถูกป้อนเข้ามาที่อาร์บิเตอร์ตลอดเวลา เพื่อใช้ตรวจสอบการมาของโครงแบบและการร้องขอจากตัววาง ในงานวิจัยนี้กำหนดให้เรียกว่า สัญญาณแทรกแซง อธิบายการทำงานได้ดังรูปที่ 3.9



รูปที่ 3.9 วงจรที่ใช้ตรวจสอบการมาของสัญญาณ

จากรูปที่ 3.9 แสดงวงจรที่ใช้ตรวจสอบสัญญาณที่ต้องการ โดยวงจร Ready Detector จะรับสัญญาณที่ถูกเลือกจากอาร์บิเตอร์แล้วนำมาตรวจสอบถึงการมาถึงของสัญญาณที่ต้องการ โดยมีขั้นตอนการตรวจสอบแสดงได้ดังรูปที่ 3.10

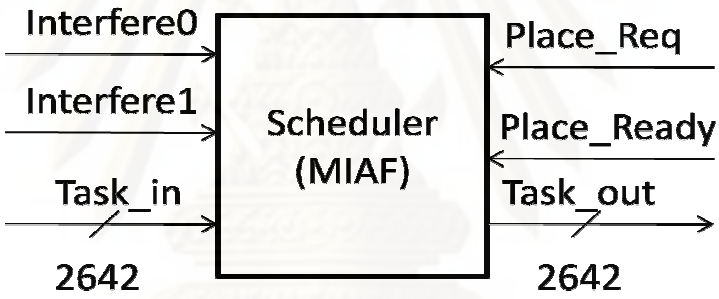


รูปที่ 3.10 กลไกการตรวจสอบการมาถึงของสัญญาณ

อาร์บิเตอร์จะเลือกสัญญาณแทรกแซงในครั้งแรกเสมอ เนื่องจากสัญญาณแทรกแซงเป็นสัญญาณที่อยู่ภายในวงจร จึงสามารถถูกส่งมายังอาร์บิเตอร์ได้เร็วกว่าสัญญาณที่ได้รับจากภายนอก หากอาร์บิเตอร์เลือกค่าสัญญาณแทรกแซง 2 ครั้งติดต่อกัน วงจรที่ใช้ตรวจสอบจะส่งผลลัพธ์ออกมาว่าไม่มีการมาถึงของสัญญาณที่ต้องการ

3.2.3.3 โครงสร้างโดยรวมของตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน

ภายในตัวกำหนดการที่ออกแบบด้วยขั้นตอนวิธีพื้นที่น้อยที่สุดก่อนจะนำกลไกดังกล่าวมาตรวจสอบการมาของโครงแบบและการร้องขอการวางจากตัววาง โดยใช้สัญญาณแทรกแซงจำนวน 2 สัญญาณเพื่อตรวจสอบการมาถึงของโครงแบบและสัญญาณ Place_Req ซึ่งเป็นสัญญาณร้องขอจากตัววาง โครงสร้างโดยรวมสามารถแสดงได้ดังรูปที่ 3.11

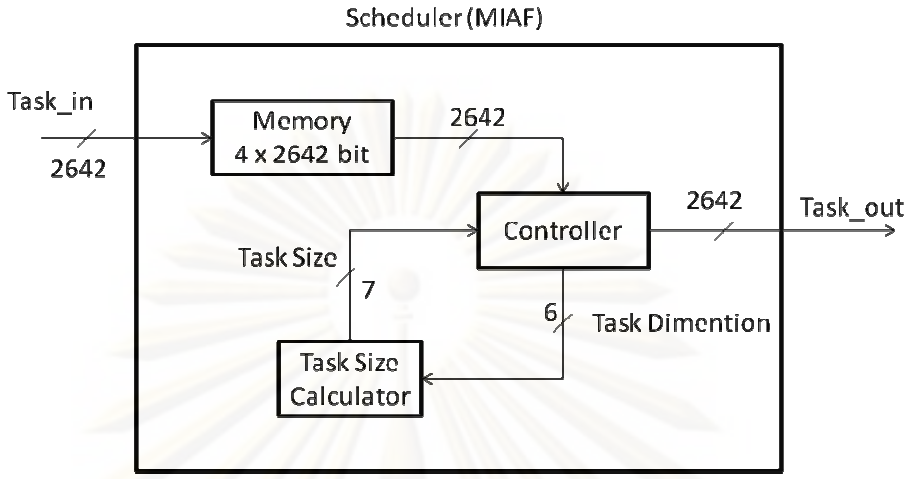


รูปที่ 3.11 โครงสร้างโดยรวมของตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน

จากรูปที่ 3.11 สัญญาณ Place_Ready จะถูกส่งมาจากตัววางเพื่อบอกกับตัวกำหนดการว่าการหาพื้นที่ให้กับโครงแบบนั้นสิ้นสุดแล้ว และจะส่งสัญญาณ Place_Req เมื่อตัววางมีความพร้อมในการรับโครงแบบครั้งต่อไป

3.2.3.4 ส่วนประกอบภายในตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน

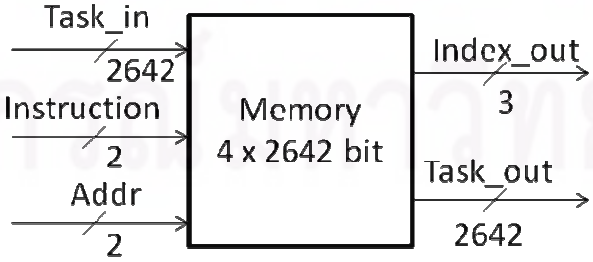
ภายในตัวกำหนดการจะประกอบด้วย 3 ส่วน คือ หน่วยความจำที่ใช้เก็บโครงแบบ วงจรคำนวณขนาดโครงแบบ และวงจรควบคุมที่ใช้จัดการการทำงานของตัวกำหนดการแสดงได้ดังรูปที่ 3.12



รูปที่ 3.12 ส่วนประกอบภายในตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน

รายละเอียดภายในส่วนประกอบแต่ละตัวสามารถอธิบายได้ดังนี้

1) หน่วยความจำที่ใช้ในตัวกำหนดการมีขนาด 4 ตำแหน่ง ตำแหน่งละ 2642 บิต ซึ่งถูกใช้สำหรับเก็บโครงแบบที่รับเข้ามา โดยมีคำสั่ง 4 คำสั่ง คือ อ่าน เขียน ลบ และอ่านค่าดัชนี โดยค่าดัชนีจะเป็นตัวบอกตำแหน่งถัดไปจากตำแหน่งสุดท้ายที่ถูกใช้ คำสั่งเขียนจะบันทึกโครงแบบที่เข้ามาไว้ตำแหน่งซ้ายสุด โดยจะเลื่อนโครงแบบที่เข้ามาก่อนไปด้านขวาโครงแบบละ 1 ตำแหน่ง เพื่อรักษาลำดับของโครงแบบที่เข้ามาในการตัดสินใจการส่งโครงแบบให้กับตัววางในกรณีที่มีโครงแบบที่มีขนาดเท่ากันมากกว่า 1 โครงแบบ ส่วนคำสั่งลบจะลบโครงแบบในตำแหน่งที่ระบุและเลื่อนโครงแบบด้านขวาของตำแหน่งที่ลบมาทางด้านซ้ายเพื่อไม่ให้เกิดตำแหน่งว่าง คำสั่งอ่านค่าดัชนีจะส่งค่าดัชนีไปยังวงจรควบคุมเพื่อตรวจสอบว่าหน่วยความจำเต็มหรือไม่ แสดงโครงสร้างโดยรวมของหน่วยความจำได้ ดังรูปที่ 3.13



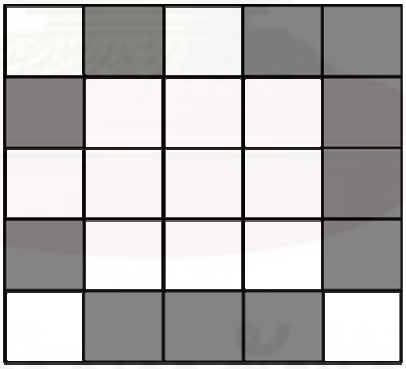
รูปที่ 3.13 หน่วยความจำที่ใช้ในตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน

2) วงจรคำนวณขนาดโครงแบบมีหน้าที่คำนวณขนาดโครงแบบ โดยขนาดโครงแบบจะเท่ากับค่าแนวนอนคูณด้วยค่าแนวตั้ง และส่งผลลัพธ์ไปยังวงจรควบคุมเพื่อหาโครงแบบที่มีขนาดน้อยที่สุดที่มีอยู่ในหน่วยความจำ

3) วงจรควบคุมจะมีอาบิเตอร์ 2 ตัว ที่ถูกใช้เพื่อตรวจสอบการมาของโครงแบบและสัญญาณร้องขอจากตัววาง รวมถึงทำหน้าที่ส่งมิติของโครงแบบให้กับวงจรคำนวณขนาดโครงแบบและส่งโครงแบบที่มีขนาดน้อยที่สุดให้กับตัววาง

3.2.4 **ตัววาง**

ตัววางในสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้จะทำหน้าที่จัดการการหาพื้นที่ที่เหมาะสมสำหรับแต่ละโครงแบบ งานวิจัยนี้ได้นำส่วนหนึ่งของขั้นตอนวิธีของเอสมาอิลดัสต์ [3] มาใช้เพื่อออกแบบตัววาง คือ การหาจำนวนหน่วยประมวลผลรอบข้างที่ถูกจองไว้มากที่สุด เป็นพื้นที่สำหรับการวาง โดยจะกำหนดให้เรียกว่าเอ็มเอโอเอ็นซี (MAONC: Maximal Occupied Neighbor Cells) ตัววางจะไม่รับโครงแบบที่เข้ามาใหม่จนกว่าจะสามารถวางโครงแบบที่ติดค้างอยู่ได้



รูปที่ 3.14 พื้นที่ที่ถูกตรวจสอบเพื่อหาจำนวนหน่วยประมวลผลรอบข้างที่ถูกจอง

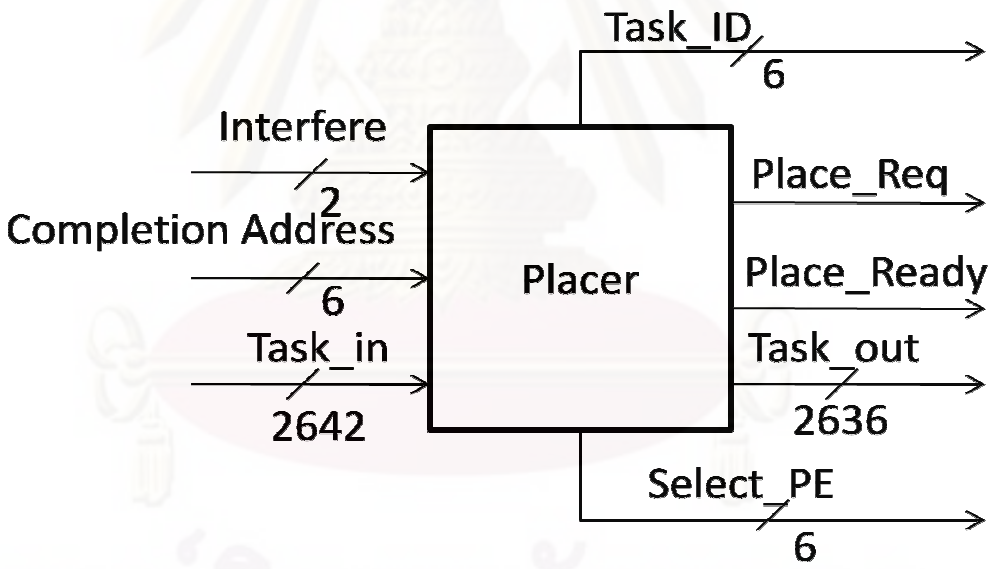
จากรูปที่ 3.14 สี่เหลี่ยมที่ถูกระบายสีหมายถึงหน่วยประมวลผลย่อยที่ถูกจอง และสี่เหลี่ยมที่เป็นสีขาวแสดงถึงหน่วยประมวลผลย่อยที่ว่างอยู่ ในตัวอย่างนี้แสดงการหาจำนวนหน่วยประมวลผลย่อยรอบข้างที่ถูกจองของพื้นที่ขนาด 3X3 ซึ่งสามารถหาได้จากสมการดังต่อไปนี้

จำนวนหน่วยประมวลผลย่อยรอบข้างที่ถูกจอง = จำนวนที่ถูกจองด้านบน + จำนวนที่ถูกจองด้านล่าง + จำนวนที่ถูกจองด้านซ้าย + จำนวนที่ถูกจองด้านขวา (3.1)

จากการคำนวณหาค่าจำนวนที่ถูกจองจากรูปที่ 3.14 ได้ค่าเท่ากับ 10 ซึ่งหากพบพื้นที่ที่มีค่าเท่ากันจะเลือกพื้นที่ที่พบก่อนไล่จากซ้ายไปขวาและบนลงล่างก่อน

3.2.4.1 โครงสร้างโดยรวมของตัววาง

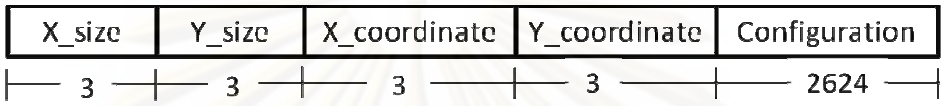
ในการออกแบบตัววางจะใช้อาร์บิเตอร์และสัญญาณแทรกแซงเช่นเดียวกับการออกแบบตัวกำหนดการ ตัววางจะตรวจสอบการสิ้นสุดของการประมวลผลของหน่วยประมวลผลย่อยในแถวลำดับคำนวณเพื่อนำพิกัดของพื้นที่ที่ถูกจองไว้จากโครงแบบที่ถูกประมวลผลเสร็จสิ้นล่าสุดมาปรับค่าให้เป็นค่าว่างในหน่วยความจำ แสดงโครงสร้างโดยรวมของตัววางได้ดังรูปที่ 3.15



รูปที่ 3.15 โครงสร้างโดยรวมของตัววาง

ความแตกต่างระหว่างตัววางที่ใช้สำหรับตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธีมาก่อนให้บริการก่อนและพื้นที่น้อยที่สุด คือ ตัววางแบบหลังจะมีสัญญาณควบคุมเพิ่มเติมอีก 3 สัญญาณ คือ สัญญาณ Place_Req สัญญาณ Place_Ready และสัญญาณ Interfere ดังที่ได้กล่าวไปแล้วข้างต้น สัญญาณ Completion Address คือ สัญญาณตำแหน่งที่ได้รับจากภายนอก

คือ สัญญาณสิ้นสุดของหน่วยประมวลผลย่อยแต่ละตัว ซึ่งจะถูกส่งซ้ำกลับไปยังวงจรภายนอกเป็นสัญญาณ Select_PE ส่วนสัญญาณ Task_out คือ โครงแบบที่ถูกส่งไปยังตัวบรรจุ โดยจะถูกถอดหมายเลขประจำงาน Task_ID และพิกัดของหน่วยประมวลผลที่มีสัญญาณผลลัพธ์ของงานออกแล้วเพิ่มพิกัดที่โครงแบบจะถูกวางลงบนแถวลำดับคำนวณเข้าไป แสดงโครงสร้างข้อมูลของโครงแบบได้ดังรูปที่ 3.16



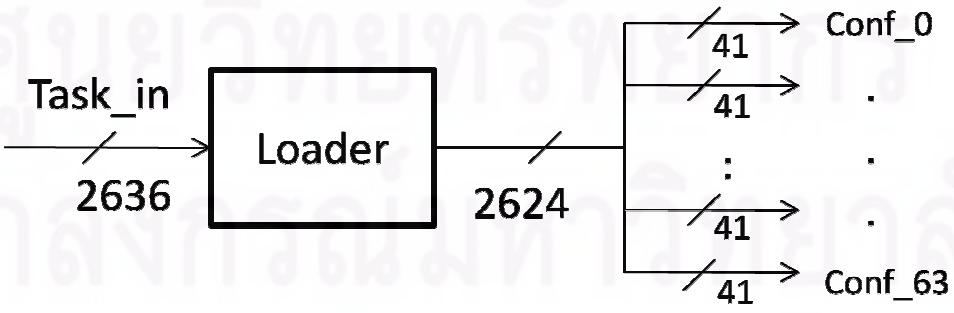
รูปที่ 3.16 โครงสร้างข้อมูลของโครงแบบที่ถูกส่งไปยังตัวบรรจุ

3.2.5 ตัวบรรจุ

ตัวบรรจุในสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้จะทำหน้าที่ส่งข้อมูลการปรับเปลี่ยนโครงแบบไปยังหน่วยประมวลผลย่อยแต่ละตัว ซึ่งคือสัญญาณควบคุมที่ควบคุมการไหลของข้อมูลภายในหน่วยประมวลผลย่อยและการส่งต่อผลลัพธ์ไปยังหน่วยประมวลย่อยตัวอื่น

3.2.5.1 โครงสร้างโดยรวมของตัวบรรจุ

โครงสร้างโดยรวมของตัวบรรจุจะมีสัญญาณขาเข้าเป็นโครงแบบที่ได้รับมาจากตัววางซึ่งถูกแทรกพิกัดที่ต้องการวาง โดยตัวบรรจุจะนำค่าพิกัดนี้ไปคำนวณหาค่าพิกัดสัมบูรณ์ (Absolute coordinate) สำหรับสัญญาณควบคุมแต่ละตัว แสดงได้ดังรูปที่ 3.17



รูปที่ 3.17 โครงสร้างโดยรวมของตัวบรรจุ

จากรูปที่ 3.17 สามารถอธิบายได้ว่า ตัวบรรจุจะรับโครงแบบที่มีขนาด 2636 บิต เข้ามา โดยมีตำแหน่งที่ต้องการวางโครงแบบ ขนาดของโครงแบบ และข้อมูลสัญญาณควบคุม ตัวบรรจุจะส่งสัญญาณควบคุมไปยังหน่วยประมวลผลย่อยแต่ละตัวที่ตรงกับค่าพิกัดสัมบูรณ์ที่คำนวณไว้ โดยจะส่งสัญญาณควบคุมเริ่มจากมุมบนซ้ายของพื้นที่ที่ถูกวาง และจะส่งสัญญาณควบคุมต่อไปทางด้านขวาจนถึงหน่วยประมวลผลย่อยด้านขวาสุดแล้วจึงเริ่มส่งสัญญาณควบคุมจากด้านซ้ายอีกครั้งหนึ่งในแถวต่อไปจนครบหน่วยประมวลผลทุกตัวในพื้นที่ที่ถูกจองไว้

สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ที่ถูกพัฒนาขึ้น มีความหลากหลายทั้งโครงสร้างการเชื่อม การปรับเปลี่ยนโครงแบบ ขนาดของหน่วยประมวลผลย่อยและรายละเอียดอื่นๆ สามารถแสดงการเปรียบเทียบโครงสร้างของสถาปัตยกรรมที่ถูกพัฒนาขึ้นในงานวิจัยกับงานวิจัยอื่นได้ดังตารางที่ 3.3

ตารางที่ 3.3 สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้

สถาปัตยกรรม	**โครงสร้าง**	**ขนาด**	**ชนิดของวงจร**
Spartan [15]	แถวลำดับ 2 มิติ	1 บิต	สมวาร
Virtex [16]	แถวลำดับ 2 มิติ	1 บิต	สมวาร
CHESS [6]	ตาข่าย	4 บิต	สมวาร
RaPID [2]	แถวลำดับ 1 มิติ	16 บิต	สมวาร
PADDI-2 [10]	ครอสส์บาร์	16 บิต	สมวาร
PipeRench [4]	แถวลำดับ 1 มิติ	128 บิต	สมวาร
ARCA [11]	แถวลำดับ 2 มิติ	1 บิต	อสมวาร
Asynchronous PipeRench [7]	แถวลำดับ 1 มิติ	128 บิต	อสมวาร
งานวิจัยนี้	แถวลำดับ 2 มิติ	8 บิต/1 บิต	อสมวาร

จากตารางที่ 3.3 เป็นการเปรียบเทียบโครงสร้างของสถาปัตยกรรม ขนาดของหน่วยประมวลผลย่อย และชนิดของวงจร สถาปัตยกรรมส่วนใหญ่ที่ถูกพัฒนาขึ้นจะเป็นวงจรแบบสมวาร มีสถาปัตยกรรมที่ใช้กันอย่างแพร่หลายในปัจจุบัน ได้แก่ สถาปัตยกรรมสปาร์ตัน (Spartan) และเวอร์เท็กซ์ (Virtex) ซึ่งเป็นโครงสร้างของเอฟพีจีเอ สถาปัตยกรรมเซสส์จะมีโครงสร้างการเชื่อมต่อแบบตาข่าย ซึ่งสามารถติดต่อกับหน่วยประมวลผลย่อยอื่นได้ในแนวทแยงมุม รวมทั้งหมด 8 ทิศ คือ บน บนซ้าย ซ้าย ล่างซ้าย ล่าง ล่างขวา ขวา และบนขวา ทำให้มีความยืดหยุ่นในการเชื่อมต่อมากกว่า 4 ทิศทาง ในโครงสร้างของเซสส์จะมีหน่วยความจำถูกฝังอยู่ด้วย เพื่อใช้สำหรับงานประยุกต์ที่ต้องการใช้ข้อมูลเป็นจำนวนมาก สถาปัตยกรรมเรปิดจะมีโครงสร้างเป็นแบบแถวลำดับ 1 มิติ ซึ่งภายในจะมีมอดูลสำหรับประมวลผลข้อมูลต่างๆ เช่น หน่วยคำนวณและตรรกะ เรจิสเตอร์ และหน่วยความจำ เป็นต้น เป็นโครงสร้างที่เหมาะสำหรับงานประยุกต์ที่มีลักษณะสายท่อ สถาปัตยกรรมแพ็ดดี้ทูมีโครงสร้างแบบครอสส์บาร์ มีข้อดี คือ สามารถปรับเปลี่ยนให้เชื่อมต่อหน่วยประมวลผลภายในได้ง่าย โดยเลือกเส้นทางการไหลของข้อมูลผ่านบัสที่วางขวางกัน สถาปัตยกรรมไปป์เรนช์จะมีโครงสร้างแบบแถวลำดับ 1 มิติ เช่นเดียวกับ สถาปัตยกรรมเรปิด แต่สเตจที่อยู่ในสายท่อ จะประกอบด้วยหน่วยประมวลผลย่อยหลายตัว เหมาะสำหรับงานประยุกต์ที่มีลักษณะสายท่อเช่นเดียวกัน ส่วนสถาปัตยกรรมที่ได้รับการพัฒนาให้เป็นวงจรอสมวารมีด้วยกัน 3 สถาปัตยกรรมได้แก่ อาร์กา อะซิงโครนัสไปป์เรนช์ (Asynchronous PipeRench) และสถาปัตยกรรมที่ถูกออกแบบในงานวิจัยนี้ งานวิจัยนี้ได้นำอาร์กามาเป็นต้นแบบในการออกแบบโครงสร้างของแถวลำดับคำนวณ โดยมีข้อแตกต่าง คือ โครงสร้างของอาร์กาจะรองรับการประมวลผลข้อมูลเพียง 1 บิต เท่านั้น ซึ่งมีข้อเสีย คือ จะต้องใช้สายเชื่อมโยงข้อมูลระหว่างหน่วยประมวลผลย่อยเป็นจำนวนมาก ใช้เวลาในการวางและหน่วยความจำเพื่อเก็บโครงแบบมากกว่าเมื่อเทียบสถาปัตยกรรมที่ถูกออกแบบขึ้นในงานวิจัยนี้ ส่วนอะซิงโครนัสไปป์เรนช์มีข้อเสียดังที่ได้กล่าวไปแล้วในบทที่ 2 คือ มีการเชื่อมต่อของหน่วยประมวลผลย่อยภายในสเตจเดียวกันและสเตจถัดไปอยู่เป็นจำนวนมาก จึงทำให้ไม่เหมาะสมในการนำมาทำเป็นวงจรอสมวาร

ในบทนี้ได้กล่าวถึงการออกแบบแถวลำดับคำนวณ 2 มิติ และการออกแบบตัวควบคุม จะกล่าวถึงการทดสอบแถวลำดับคำนวณและตัวควบคุมที่ได้ออกแบบไว้ในบทที่ 4

บทที่ 4

การทดลอง ผลการทดลอง และสรุปผลการทดลอง

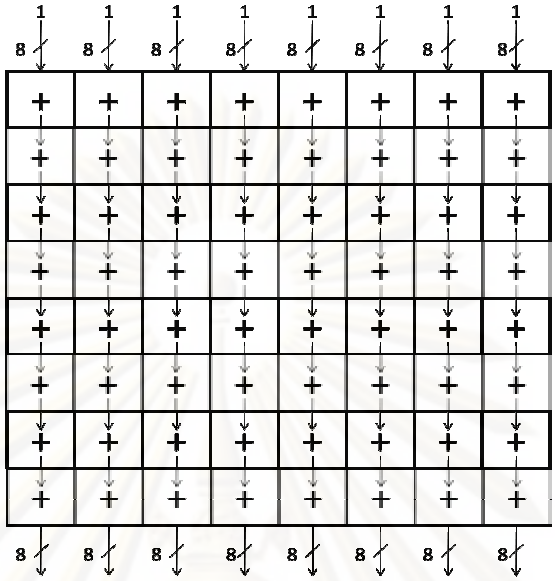
การทดสอบแถวลำดับคำนวณและตัวควบคุมในงานวิจัยนี้จะมีการทดสอบดังต่อไปนี้ คือ การทดสอบแถวลำดับคำนวณ การทดสอบแต่ละส่วนประกอบในตัวควบคุม การเปรียบเทียบกำลังไฟฟ้า การประมาณค่าขนาดของตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธีที่แตกต่างกัน และการเปรียบเทียบประสิทธิภาพในการวางเมื่อใช้ตัวกำหนดการที่แตกต่างกัน ซึ่งมีการทดสอบทั้งในระดับการไหลของข้อมูลและระดับพฤติกรรม รวมถึงมีการอธิบายถึงความเป็นไปได้ในการนำวงจรอสมวารมาประมวลผลกับแถวลำดับคำนวณที่ออกแบบไว้ และสรุปผลการทดลองเป็นลำดับสุดท้าย โดยมีรายละเอียดดังนี้

4.1 การทดสอบแถวลำดับคำนวณ

การทดสอบนี้จะแบ่งออกเป็น 2 ส่วน คือ การทดสอบการทำงานโดยรวมของแถวลำดับขนาด 8X8 และการทดสอบการนำวงจรอสมวารมาจัดกลุ่มลงบนแถวลำดับคำนวณ โดยใช้โปรแกรมไอเอสอี เว็บแพ็ค (ISE Webpack) [13] ร่วมกับโปรแกรมโมเดลซิมไซลิงซ์เอ็ดดิชั่น (ModelSim Xilinx Edition) [14] เพื่อทดสอบการทำงาน

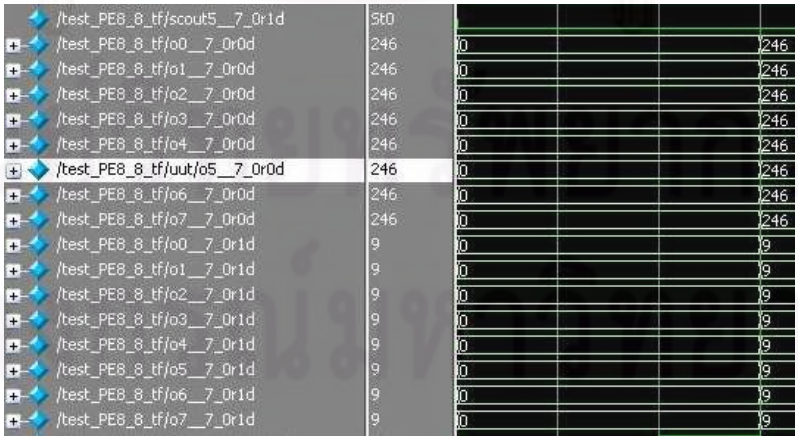
4.1.1 การทดสอบแถวลำดับคำนวณขนาด 8X8

ในการทดสอบการทำงานของแถวลำดับคำนวณบนโปรแกรมบาลซ่า (Balsa) [12] มีข้อจำกัดในการจำลองการทำงานของวงจร คือ ไม่สามารถทดสอบแถวลำดับคำนวณที่มีขนาดใหญ่กว่า 4X4 ได้ ดังนั้นงานวิจัยนี้จึงทดสอบแถวลำดับคำนวณขนาด 8X8 โดยการนำแฟ้มข้อมูลภาษาเวริล็อก (Verilog) ที่โปรแกรมบาลซ่าสร้างขึ้น มาทดสอบในระดับพฤติกรรมกับโปรแกรมโมเดลซิม โดยทดสอบการทำงานของแถวลำดับคำนวณด้วยการปรับเปลี่ยนโครงสร้างให้คำนวณฟังก์ชันบวกโดยรับสัญญาณขาเข้าจากแถวบนสุดของหน่วยประมวลผลย่อย และส่งผลลัพธ์ออกมาทางช่องสัญญาณขาออกของแถวล่างสุด แสดงได้ดังรูปที่ 4.1



รูปที่ 4.1 แถวลำดับคำนวณขนาด 8X8 ที่ถูกปรับเปลี่ยนเพื่อทดสอบการทำงาน

จากรูปที่ 4.1 กำหนดให้ข้อมูลขาเข้าขนาด 8 บิต เข้าสู่ฟังก์ชันการบวก โดยกำหนดให้มีค่าเท่ากับ 1 ในแถวบนสุดของแถวลำดับคำนวณ และมีการใส่ค่าคงที่ซึ่งมีค่าเท่ากับ 1 ลงในหน่วยประมวลผลย่อยแต่ละตัว หน่วยประมวลผลย่อยแต่ละตัวจะส่งผลลัพธ์ไปยังหน่วยประมวลผลย่อยด้านล่างที่อยู่ถัดลงมาเป็นลำดับจนถึงตัวสุดท้ายด้านล่างจะส่งผลลัพธ์ออกมานอกแถวลำดับคำนวณ

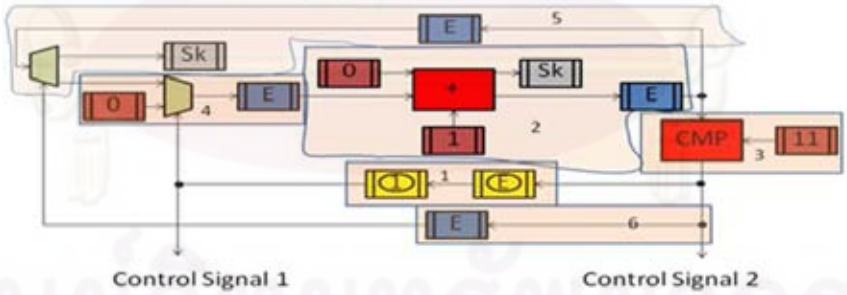


รูปที่ 4.2 ผลการทดสอบแถวลำดับคำนวณบนโปรแกรมโมเดลซิม

จากรูปที่ 4.2 แสดงการจำลองการทำงานของวงจรในระดับพฤติกรรมบนโปรแกรมโมเดลซิม พบว่าผลลัพธ์ถูกสะสมมาในแต่ละหน่วยประมวลผลย่อยมีค่าถูกต้อง โดยจะมีสายคู่สัญญาณค่า 1 และ 0 เนื่องจากวงจรที่ออกแบบเป็นวงจรอสมวารที่ใช้การเข้ารหัสแบบรางคู่ กล่าวคือ กลุ่มสัญญาณค่า 9 เกิดจากสายคู่สัญญาณค่า 1 เป็นค่าผลลัพธ์ของการบวกในการทดสอบของหน่วยประมวลผลในแต่ละหลักในแนวตั้ง และกลุ่มสัญญาณค่า 246 เป็นสายคู่สัญญาณค่า 0 ซึ่งเป็นค่าที่ได้จากการสลับบิตค่า 9 ที่มีขนาด 8 บิต

4.1.2 การทดสอบการนำวงจรอสมวารมาจัดกลุ่มลงบนแถวลำดับคำนวณ

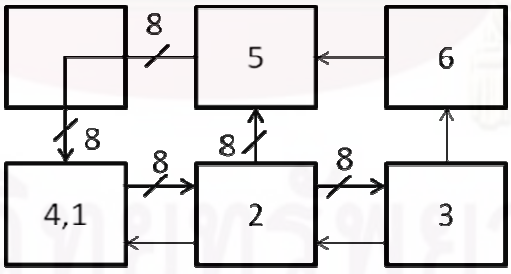
เนื่องจากการแปลงงานประยุกต์เพื่อให้ใช้ได้กับแถวลำดับคำนวณ เป็นขั้นตอนที่ซับซ้อน ดังนั้นงานวิจัยนี้จึงเลือกงานประยุกต์พื้นฐานจำนวน 1 งาน เพื่อทดสอบกับแถวลำดับคำนวณที่มีขนาด 3X2 โดยออกแบบวงจรสร้างสัญญาณควบคุมสำหรับวงจรที่มีลักษณะการทำงานวนซ้ำ (Loop) ซึ่งออกแบบโดยใช้วิธีการไหลของข้อมูลแบบสถิต ภายในวงจรประกอบด้วยตัวดำเนินการต่างๆ ที่อยู่ภายในหน่วยประมวลผลย่อยที่ได้ออกแบบไว้ แล้วนำวงจรที่ได้ออกแบบมาจัดกลุ่มเพื่อปรับเปลี่ยนโครงแบบให้หน่วยประมวลผลย่อยทำงานได้ตามหน้าที่ของแต่ละกลุ่ม แสดงได้ดังรูปที่ 4.3



รูปที่ 4.3 การจัดกลุ่มวงจรลงบนแถวลำดับคำนวณ

จากรูปที่ 4.3 โครงสร้างของวงจรสร้างสัญญาณควบคุมจะมีทั้งเส้นทางข้อมูลที่มีขนาด 8 บิต และ 1 บิต แลตช์ที่มีเลขหนึ่งและตัวอักษร E อยู่ในเครื่องหมายวงกลมจะถูกตั้งค่าเริ่มต้นเพื่อให้เกิดการไหลของข้อมูลได้ภายในวงจร ซึ่งแสดงถึงโทเค็น 2 ชนิด คือ โทเค็นข้อมูล และโทเค็นว่างที่ปรากฏอยู่ในการทำงานของวงจรอสมวารแบบมีการกลับสู่ศูนย์ (Return to zero)

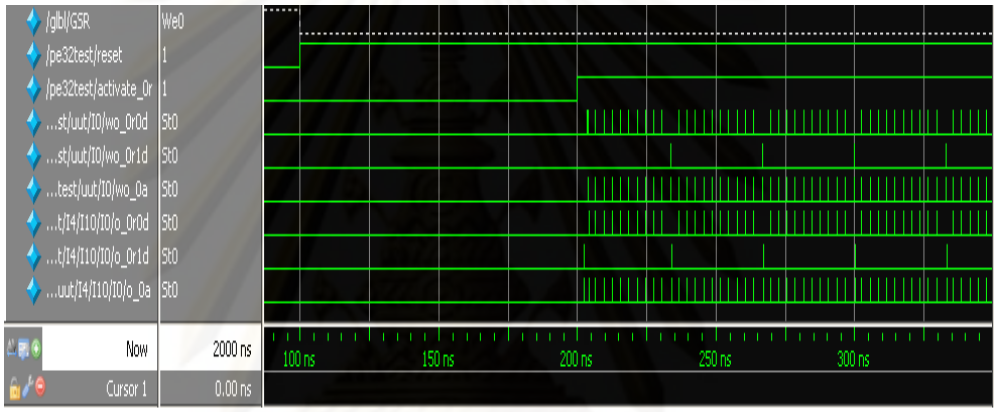
ภายในวงจรกลุ่มที่ 2 จะมีวงจรบวก ซึ่งมีสัญญาณตัวทดที่มีค่าเท่ากับ 0 และข้อมูลขาเข้าตัวที่หนึ่งเป็นค่าคงที่เพื่อเพิ่มค่าผลลัพธ์ครั้งละหนึ่ง ส่วนข้อมูลขาเข้าตัวที่สองจะเป็นผลลัพธ์ของวงจรบวกที่วนซ้ำอยู่ภายในวงจร ในวงจรอสมวารเชิงผสมแบบรางคู่ สัญญาณข้อมูลขาเข้าและขาออกทุกตัวจะมีการเปลี่ยนแปลงไม่ว่าสัญญาณนั้นจะถูกนำไปใช้หรือไม่ จึงต้องใช้แลตช์ Sk ซึ่งจะรับค่าตัวทดที่เป็น 0 เสมอ เพื่อให้เกิดการแฮนด์เชคที่สมบูรณ์ในการทำงานของวงจรอสมวาร ในวงจรกลุ่มที่ 4 จะมีแลตช์ที่ป้อนค่าคงที่เท่ากับ 0 ในการเริ่มต้นการทำงานของวงจรเพื่อเป็นข้อมูลขาเข้าตัวที่สองของวงจรบวก ในวงจรกลุ่มที่ 5 จะมีอุปกรณ์แยกส่งสัญญาณซึ่งจะนำผลลัพธ์ที่ครบตามจำนวนวนซ้ำที่ต้องการแล้วส่งไปยังแลตช์ Sk เพื่อเป็นการนำโทเค็นข้อมูลที่ไม่ต้องการใช้แล้วออกจากวงจรนี้ แลตช์ที่ไม่ได้ถูกตั้งค่าเริ่มต้นทุกตัวในวงจรนี้จะถูกใส่เพิ่มเข้ามาเพื่อทำให้การไหลของข้อมูลในเส้นทางวนทำงานได้อย่างถูกต้อง จากรูปจะเห็นได้ว่า เส้นทางวนที่เกิดขึ้นทุกเส้นทางจะประกอบด้วยแลตช์อย่างน้อย 3 ตัว สัญญาณ Control Signal1 และ Control Signal2 จะถูกส่งไปยังวงจรที่เกิดการวนซ้ำเพื่อไปกระตุ้นการทำงานของตัวอุปกรณ์รวมส่งสัญญาณและอุปกรณ์แยกส่งสัญญาณตามลำดับ เมื่อวงจรที่เกิดการวนซ้ำเริ่มทำงานอุปกรณ์รวมส่งสัญญาณจะรับข้อมูลจากภายนอกเข้ามา ขณะที่กำลังเกิดการวนซ้ำ อุปกรณ์รวมส่งสัญญาณและอุปกรณ์แยกส่งสัญญาณนี้จะควบคุมให้ข้อมูลไหลอยู่ภายในวงจรเท่านั้น หลังจากครบการวนซ้ำตามที่กำหนดแล้ว อุปกรณ์แยกส่งสัญญาณจะส่งผลลัพธ์ออกไปยังวงจรอื่นต่อไป แสดงการวางโครงแบบที่ได้จากการจัดกลุ่มของวงจรสัญญาณสร้างควบคุมนี้ลงบนหน่วยประมวลผลย่อย ได้ดังรูปที่ 4.4



รูปที่ 4.4 การส่งข้อมูลระหว่างหน่วยประมวลผลย่อยหลังจากจัดกลุ่มแล้ว

จากรูปที่ 4.4 แสดงให้เห็นถึงการส่งข้อมูลกันระหว่างหน่วยประมวลผลย่อย จะเห็นได้ว่ากลุ่มของวงจรหมายเลข 4 และหมายเลข 1 สามารถถูกวางลงบนหน่วยประมวลย่อยตัวเดียวกันได้ เนื่องจากเป็นชนิดของเส้นทางข้อมูลละชนิดกันและจะส่งข้อมูลเข้าหากันเองภายใน

หน่วยประมวลผลย่อย วงจรกลุ่มที่ 3 จะส่งค่าผลลัพธ์จากตัวเปรียบเทียบผ่านกลุ่มที่ 2 ไปยังกลุ่มที่ 1 ได้ เพื่อลดปริมาณการใช้หน่วยประมวลผลย่อยในแถวลำดับคำนวณ เนื่องจากเส้นทางข้อมูลขนาด 1 บิต ของหน่วยประมวลผลย่อยที่ทำงานให้กับกลุ่ม 2 ยังว่างอยู่ ส่วนหน่วยประมวลผลย่อยมุมบนซ้ายจะถูกใช้เป็นตัวส่งต่อข้อมูล เนื่องจากการส่งข้อมูลกันระหว่างหน่วยประมวลผลย่อยไม่สามารถทำในลักษณะทแยงมุมได้ เส้นทางข้อมูลขนาด 8 บิต ที่แสดงในรูปจะแสดงถึงเส้นทางที่ผลลัพธ์ที่ถูกสะสมมาจากการบวก ส่วนเส้นทางข้อมูลขนาด 1 บิต จะเป็นเส้นทางข้อมูลของสัญญาณเงื่อนไขที่เกิดขึ้นภายในวงจร



รูปที่ 4.5 ผลการทดสอบวงจรที่ถูกวางลงบนแถวลำดับคำนวณบนโปรแกรมโมเดลซิม

จากการจำลองการทำงานในระดับพฤติกรรมบนโปรแกรมโมเดลซิมพบว่าการทำงานของวงจรสร้างสัญญาณควบคุมทำงานได้อย่างถูกต้อง โดยอธิบายจากรูปที่ 4.5 ได้ว่าทุกครั้งที่วงจรสะสมค่าที่เกิดจากวงจรบวกครบตามจำนวนที่กำหนดในการวนซ้ำแล้ว ตัวเปรียบเทียบจะส่งค่า 1 ออกมาสู่วงจรภายนอก ในการทดสอบวงจรนี้จะส่งสัญญาณที่ปรับเปลี่ยนโครงแบบเข้าสู่หน่วยประมวลผลย่อยตลอดเวลา ทำให้เกิดผลลัพธ์วนซ้ำดังรูป

จากการทดสอบในหัวข้อนี้ได้แสดงถึงความเป็นไปได้ในการนำงานประยุกต์จริงที่ถูกเขียนขึ้นด้วยภาษาระดับสูงมาจัดกลุ่มแบ่งลงบนหน่วยประมวลผลย่อยได้

4.2 การทดสอบตัวควบคุม

การทดสอบตัวควบคุมจะแบ่งการทดสอบออกเป็น 3 ส่วน คือ การทดสอบตัวกำหนดการ การทดสอบตัววาง และการทดสอบตัวบรรจุ ซึ่งจะกำหนดให้ขนาดของสัญญาณควบคุมโครงแบบมีขนาดเท่ากับ 1 เพื่อลดปริมาณขนาดข้อมูลที่ใช้ทำการทดสอบ โดยมีรายละเอียดดังนี้

4.2.1 การทดสอบตัวกำหนดการ

การทดสอบตัวกำหนดการจะแบ่งออกเป็น 2 ส่วน โดยจะทดสอบตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธีมาก่อนให้บริการก่อนและขั้นตอนวิธีพื้นที่น้อยที่สุดก่อน มีรายละเอียดดังนี้

4.2.1.1 การทดสอบตัวกำหนดการแบบมาก่อนให้บริการก่อน

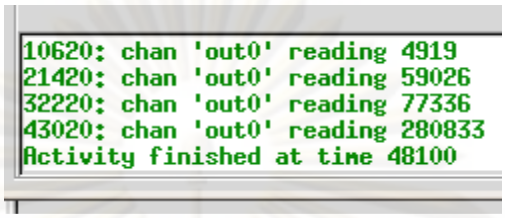
เนื่องจากการทดสอบได้ถูกกำหนดให้ขนาดของสัญญาณควบคุมมีค่าเท่ากับ 1 ดังนั้นขนาดของโครงสร้างข้อมูลที่ใช้เก็บโครงแบบจะมีขนาดเท่ากับ 19 บิต โดยสุ่มงาน 4 งานที่มีค่าแสดงได้ดังตารางที่ 4.1

ตารางที่ 4.1 งานที่ใช้ในการทดสอบตัวกำหนดการแบบมาก่อนให้บริการก่อน

หมายเลขประจำงาน	ขนาดของงานในแนวนอน	ขนาดของงานในแนวตั้ง	พิกัดผลลัพธ์แนวนอน	พิกัดผลลัพธ์แนวตั้ง	รหัสสัญญาณควบคุม
0	5	7	4	4	1
7	2	6	2	2	0
9	4	5	2	5	0
34	3	3	1	1	1

จากงานที่แสดงได้ในตารางที่ 4.1 สามารถแปลงออกมาเป็นค่าที่ส่งให้กับตัวกำหนดการตามลำดับ ได้แก่ 4919_{10} ($000000\_100\_110\_011\_011\_1_2$), 59026_{10}

(000111_001_101_001_001_0_2), 77336_{10} (001001_011_100_001_100_0), 280833_{10} (100010_010_010_000_000_1_2)

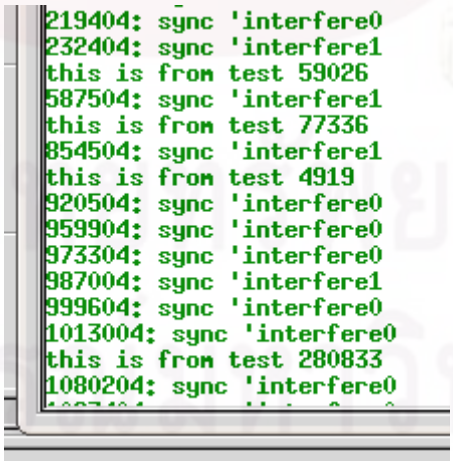


รูปที่ 4.6 ผลการทดสอบตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธีมาก่อนให้บริการก่อน

จากผลการทดสอบในรูปที่ 4.6 แสดงให้เห็นว่าตัวกำหนดการทำงานได้อย่างถูกต้อง โดยจะส่งงานออกมาตามลำดับที่ได้รับจากวงจรภายนอก ซึ่งจะส่งค่าที่แปลงเป็นเลขฐานสิบออกมาที่ช่องสัญญาณขาออก out0

4.2.1.2 **การทดสอบตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน**

การทดสอบนี้จะใช้งานเดียวกันกับการทดสอบในหัวข้อ 4.2.1.1 โดยจะจำลองให้ตัวกำหนดการถูกร้องขอจากตัววางเมื่อมีงานอยู่ในหน่วยความจำจำนวน 3 งาน งานทั้ง 3 นี้จะถูกส่งไปยังตัววางทั้งหมดและหลังจากนั้นจะรับงานที่ 4 เข้ามาในหน่วยความจำเป็นลำดับต่อไป

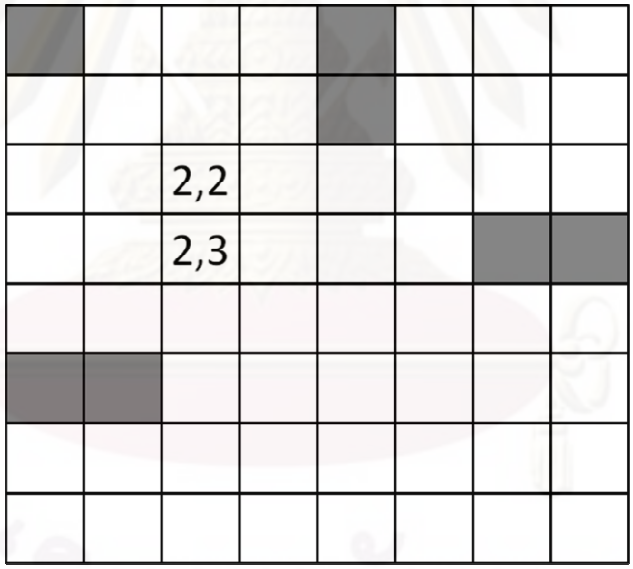


รูปที่ 4.7 ผลการทดสอบตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธีพื้นที่น้อยที่สุดก่อน

จากรูปที่ 4.7 แสดงให้เห็นได้ว่า ลำดับของงานที่ถูกส่งไปยังตัววาง คือ งานหมายเลข 7 (59026_{10}) ซึ่งมีขนาด 12 งานหมายเลข 9 (77336_{10}) ซึ่งมีขนาด 20 และงานหมายเลข 0 (4919_{10}) ซึ่งมีขนาด 35 เรียงจากน้อยไปมาก ระหว่างนี้ตัวกำหนดการจะไม่รับงานจากภายนอกจนกว่าจะส่งให้ตัววางได้ครบทั้งหมด หลังจากนั้นจึงสามารถรับงานหมายเลข 34 และถูกร้องขอจากตัววาง ตัวกำหนดการจึงส่งงานหมายเลข 34 (280833_{10}) เป็นลำดับสุดท้าย

4.2.2 การทดสอบตัววาง

การทดสอบตัววางจะกำหนดให้หน่วยความจำที่เก็บสถานะของแถวลำดับคำนวณมีสถานะดังรูปที่ 4.8 และทดสอบการวางโครงแบบที่มีขนาด 4x5 ลงบนแถวลำดับคำนวณ จากรูปจะเห็นได้ว่า มีพื้นที่สี่เหลี่ยมที่มีมุมบนซ้ายที่ตำแหน่ง 2, 2 และ 2, 3 เท่านั้นที่สามารถวางโครงแบบได้



รูปที่ 4.8 สถานะของแถวลำดับคำนวณที่ใช้ทดสอบการวาง

ในการทดสอบจะกำหนดค่าให้กับสถานะของแถวลำดับคำนวณให้เป็นดังรูปที่ 4.8 โดยกำหนดค่าในหน่วยความจำที่ใช้เก็บสถานะปัจจุบันในโปรแกรมบาลซ่า แสดงได้ดังรูปที่ 4.9

```
end --shared

begin

row0 := {1,0,0,0,1,0,0,0}||
row1 := {0,0,0,0,1,0,0,0}||
row2 := {0,0,0,0,0,0,0,0}||
row3 := {0,0,0,0,0,0,1,1}||
row4 := {0,0,0,0,0,0,0,0}||
row5 := {1,1,0,0,0,0,0,0}||
row6 := {0,0,0,0,0,0,0,0}||
row7 := {0,0,0,0,0,0,0,0}||
array_empty := 0||
state := 0||
unfinished_place := 0||
is_no_arb := 0;

loop
case state of
 0 then arbitrate interfere2
```

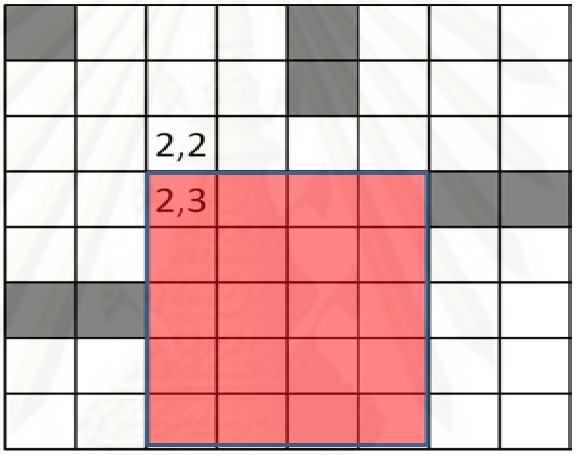
รูปที่ 4.9 การกำหนดค่าสถานะปัจจุบันของแถวลำดับในโปรแกรมบาลซ่า

จากรูปที่ 4.9 เป็นรหัสต้นฉบับภาษาบาลซ่า ซึ่งตัวแปร row0 ถึง row7 คือ ตัวแปรที่เก็บค่าสถานะการจองของแถวลำดับคำนวณ ในการทดสอบนี้จะกำหนดค่าเริ่มต้นให้กับตัวแปรเหล่านี้ โดยกำหนดค่า 1 สำหรับหน่วยประมวลผลที่ถูกจองแล้ว และกำหนดค่า 0 สำหรับหน่วยประมวลผลที่ยังไม่ได้ถูกจอง กำหนดตัวแปร array_empty ให้มีค่าเท่ากับ 0 เพื่อบอกให้ตัววางทราบว่าแถวลำดับคำนวณไม่ได้อยู่ในสถาะว่างทั้งหมด

```
occ_nb_cell is 6
can_x is 2can_y is 3
before read x,y is 3 3
before read x,y is 4 3
before read x,y is 5 3
before read x,y is 6 3
not_enough when x,y =6 3
before read x,y is 4 3
before read x,y is 5 3
before read x,y is 6 3
not_enough when x,y =6 3
can_x and can_y is 2 3
```

รูปที่ 4.10 ผลการทดสอบตัววาง

จากรูปที่ 4.10 จะเห็นได้ว่า ผลการทดสอบให้ค่าตำแหน่งสำหรับการวางเท่ากับ 2,3 จากสัญญาณ can_x และ can_y เป็นค่าที่ถูกต้องจากขั้นตอนวิธีตรวจสอบหน่วยประมวลผลรอบข้างที่ถูกจองมากที่สุด ได้ค่าเท่ากับ 6 ซึ่งมากกว่าในตำแหน่ง 2,2 ซึ่งมีค่าเท่ากับ 3 จากรูปจะเห็นได้ว่าแถวลำดับคำนวณจะถูกตรวจสอบไล่จากซ้ายไปขวาและบนลงล่างครั้งละตำแหน่ง เมื่อพบการจองของหน่วยประมวลผลในพื้นที่ที่ถูกตรวจสอบเพียงตำแหน่งเดียว พื้นที่นั้นก็จะถูกยกเลิกการตรวจสอบทันทีเนื่องจากไม่สามารถใช้วางได้ และเลื่อนพื้นที่การตรวจสอบไปทางด้านขวาจนสุดและเริ่มใหม่จากทางด้านซ้ายในแนวตั้งต่อไป



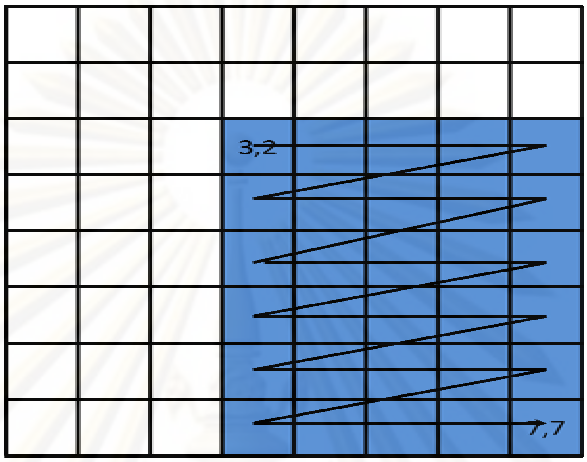
รูปที่ 4.11 โครงแบบที่ถูกวางลงบนแถวลำดับคำนวณที่ใช้ในการทดสอบ

จากรูปที่ 4.11 แสดงถึงโครงแบบที่ถูกวางลงบนแถวลำดับคำนวณตามผลการทดสอบข้างต้น ซึ่งเป็นตำแหน่งที่ให้ค่าจำนวนหน่วยประมวลผลรอบข้างที่ถูกจองมากกว่า

4.2.3 การทดสอบตัวบรรจุ

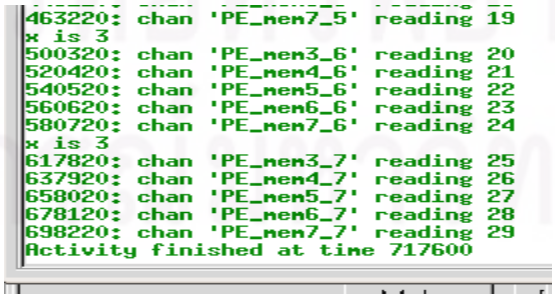
การทดสอบตัวบรรจุจะกำหนดให้ทดสอบกับงานที่มีขนาด 5X6 ซึ่งทำกับหน่วยประมวลผลย่อย 30 ตัว โดยกำหนดค่าสัญญาณควบคุมให้มีค่าจาก 0 ถึง 29 เพื่อทดสอบความถูกต้องของการส่งสัญญาณควบคุมออกไปทางสัญญาณขาออกจนถึงเส้นสุดท้ายที่ต้องส่งสัญญาณ กำหนดให้ตำแหน่งที่วาง คือ ตำแหน่ง 3, 2 กำหนดให้ค่า 0 ถึง 7 เก็บค่าความยาว 1 ถึง 8 ตามลำดับ ดังนั้นตำแหน่งสุดท้ายที่ตัวบรรจุจะต้องวางคือ (3+4), (2+5) = 7, 7 คือ

ตำแหน่งล่างขวาสุดของแถวลำดับคำนวณ โดยจะตรวจสอบลำดับการส่งและความถูกต้องของเส้นสัญญาณขาออก แสดงได้ดังรูปที่ 4.12



รูปที่ 4.12 การบรรจุโครงแบบลงบนแถวลำดับคำนวณ

จากรูปที่ 4.12 สามารถอธิบายการบรรจุโครงแบบลงบนแถวลำดับคำนวณได้ว่า ตัวบรรจุจะส่งสัญญาณควบคุมไปปรับเปลี่ยนการทำงานของหน่วยประมวลผลย่อยครั้งละ 1 ตัว กล่าวถือ ปรับเปลี่ยนเส้นทางการรับส่งข้อมูล เลือกตัวดำเนินการและการส่งข้อมูลภายในหน่วยประมวลผลย่อย ขณะที่ตัวบรรจุส่งสัญญาณควบคุมไปยังหน่วยประมวลผลย่อยตัวใดตัวหนึ่ง หน่วยประมวลผลย่อยตัวนั้นสามารถประมวลผลได้ทันทีโดยไม่ต้องรอให้ตัวบรรจุส่งสัญญาณควบคุมให้ครบทั้งโครงแบบ โดยตัวบรรจุจะส่งสัญญาณควบคุมไล่จากซ้ายไปขวา เมื่อถึงด้านขวาสุดของโครงแบบจะเริ่มส่งสัญญาณควบคุมต่อไปจากทางด้านซ้ายในแถวถัดไปจนครบทั้งโครงแบบ

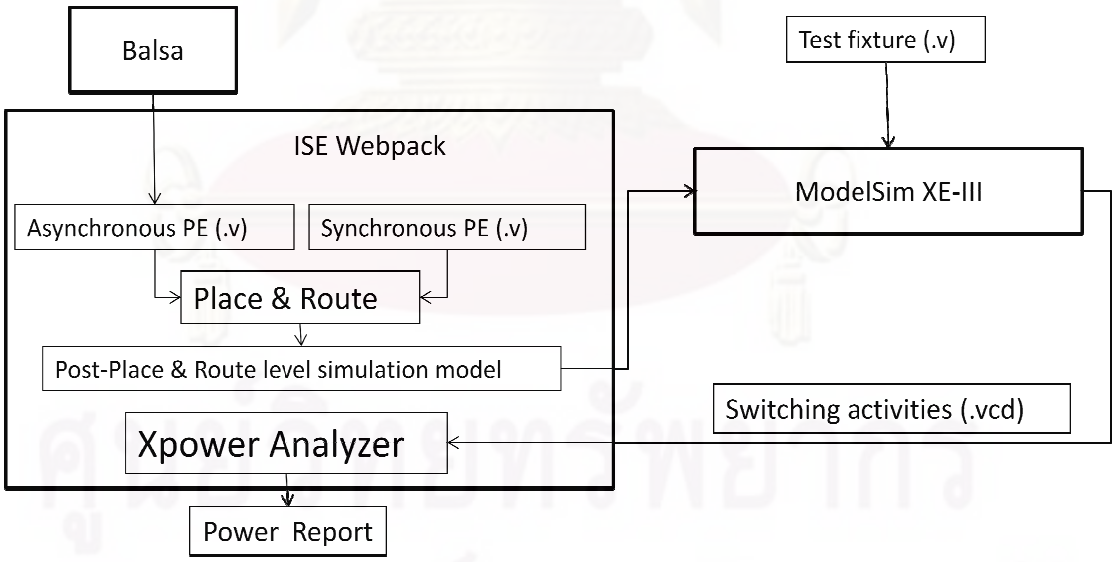


รูปที่ 4.13 ผลการทดสอบตัวบรรจุ

จากรูปที่ 4.13 แสดงผลการทดสอบตัวบรรจุ ซึ่งสามารถเลือกค่าสัญญาณควบคุมของหน่วยประมวลผลย่อยที่ได้รับจากตัววางส่งออกไปยังเส้นสัญญาณขาออกได้อย่างถูกต้อง โดยจะเริ่มส่งสัญญาณในแนวนอนจนครบ หลังจากนั้นจะเพิ่มค่าแกนตั้งแล้วส่งสัญญาณจนครบพื้นที่ทั้งหมด

4.3 การเปรียบเทียบกำลังไฟฟ้า

ในส่วนนี้จะนำหน่วยประมวลผลย่อยทั้งแบบสมวารและอสมวารมาทดสอบกำลังไฟฟ้าที่ถูกใช้ไป เนื่องจากหน่วยประมวลผลย่อยแบบอสมวารถูกสังเคราะห์ออกมาเป็นวงจรรหัสรางคู่แบบกลับสู่ศูนย์ ทำให้เกิดการเปลี่ยนแปลงของสัญญาณข้อมูลไม่ว่าจะมีค่าเป็น 0 หรือ 1 ก็ตาม รวมถึงสัญญาณแต่ละเส้นจะต้องกลับสู่ศูนย์หลังจากที่การสื่อสารเสร็จสิ้นแล้ว จึงกำหนดให้มีข้อมูลขาเข้าและขาออกมีค่า 1 เป็นส่วนใหญ่เพื่อความเท่าเทียมกันในการทดสอบ โดยจะทดสอบการบวกเลข 127 และ 128 ซึ่งจะได้ผลลัพธ์เท่ากับ 255 ซึ่งได้สัญญาณผลลัพธ์เป็น 1 ทุกตัว



รูปที่ 4.14 ขั้นตอนการทดสอบกำลังไฟฟ้า

จากรูปที่ 4.14 แสดงถึงขั้นตอนการทดสอบกำลังไฟฟ้า โดยจะเริ่มขั้นตอนจากโปรแกรมบาลซ่า เป็นขั้นตอนที่หน่วยประมวลผลย่อยแบบอสมวารถูกสร้างเป็นแฟ้มข้อมูลภาษาเวริล็อกเพื่อนำมาเปรียบเทียบกับหน่วยประมวลผลแบบสมวารที่ถูกเขียนด้วยภาษาเวริล็อกเช่นเดียวกัน หลังจากนั้นแฟ้มข้อมูลนี้จะถูกส่งมายังกระบวนการวางและจัดเส้นทางเพื่อสร้างเป็นแบบจำลองที่ใช้ในการทดสอบหลังจากผ่านกระบวนการวางและจัดเส้นทางบนเอฟพีจีเอแล้ว แบบจำลองนี้จะถูกส่งไปยังโปรแกรมโมเดลซิมเพื่อใช้ทดสอบร่วมกับแฟ้มข้อมูลที่ถูกเขียนขึ้นเพื่อใส่ค่าทดสอบให้กับแบบจำลอง โปรแกรมโมเดลซิมจะสร้างแฟ้มข้อมูลซึ่งเก็บการเปลี่ยนแปลงของสัญญาณทั้งหมด โดยการเปลี่ยนแปลงสัญญาณในวงจรดิจิทัลจะบอกถึงกำลังไฟฟ้าแบบพลวัต (Dynamic power) จึงต้องมีการเก็บการเปลี่ยนแปลงของสัญญาณที่ได้จากการจำลองการทำงานของวงจร แล้วนำการเปลี่ยนแปลงนั้นส่งให้กับโปรแกรมเอ็กซ์พาวเวอร์ อนาไลเซอร์ (XPower Analyzer) เพื่อวิเคราะห์การใช้กำลังไฟฟ้าต่อไป แสดงรหัสต้นฉบับ (Source code) ที่ต้องกำหนดค่าเริ่มต้นให้เก็บการเปลี่ยนแปลงสัญญาณได้ดังรูปที่ 4.15

```
initial begin
   // Initialize Inputs
   $dumpfile("test_PE_sync.vcd");
   $dumpvars(0,uut);
   clk = 0;
   MISconf = 0;
   FUconf = 0;
   CarryForkconf = 0;
   LatchForkconf = 0;
   MOSconf = 0;
   FUSingleoConf = 0;
   PEcoutConf = 0;
   SISconf = 0;
```

รูปที่ 4.15 รหัสต้นฉบับที่ใช้ในการเก็บการเปลี่ยนแปลงสัญญาณ

จากรูปที่ 4.15 เป็นการแทรกรหัสที่ทำให้โปรแกรมจำลองการทำงานของวงจรเก็บการเปลี่ยนแปลงของสัญญาณโดยบรรทัดแรกใช้คำสั่ง $dumpfiles เพื่อกำหนดชื่อแฟ้มข้อมูลที่ใช้เก็บการเปลี่ยนแปลงของสัญญาณ และคำสั่ง $dumpvars เพื่อกำหนดมอดูลและระดับของมอดูลที่ต้องการเก็บการเปลี่ยนแปลงของสัญญาณ โดยกำหนดให้มีค่าเท่ากับ 0 เพื่อเก็บทุกการเปลี่ยนแปลงของสัญญาณ

Name	Value	Used	Total Available	Utilization (%)
Clocks	0.00002 (W)	12	---	---
Logic	0.00040 (W)	14533	46560	31.2
Signals	0.00178 (W)	14744	---	---
IOs	0.00581 (W)	333	360	92.5
Total Quiescent Power	**1.50774 (W)**			
Total Dynamic Power	**0.00801 (W)**			
Total Power	**1.51575 (W)**			
Junction Temp	**50.0 (degrees C)**			

รูปที่ 4.16 ผลการทดสอบกำลังไฟฟ้าของหน่วยประมวลผลย่อยแบบอสมวาร

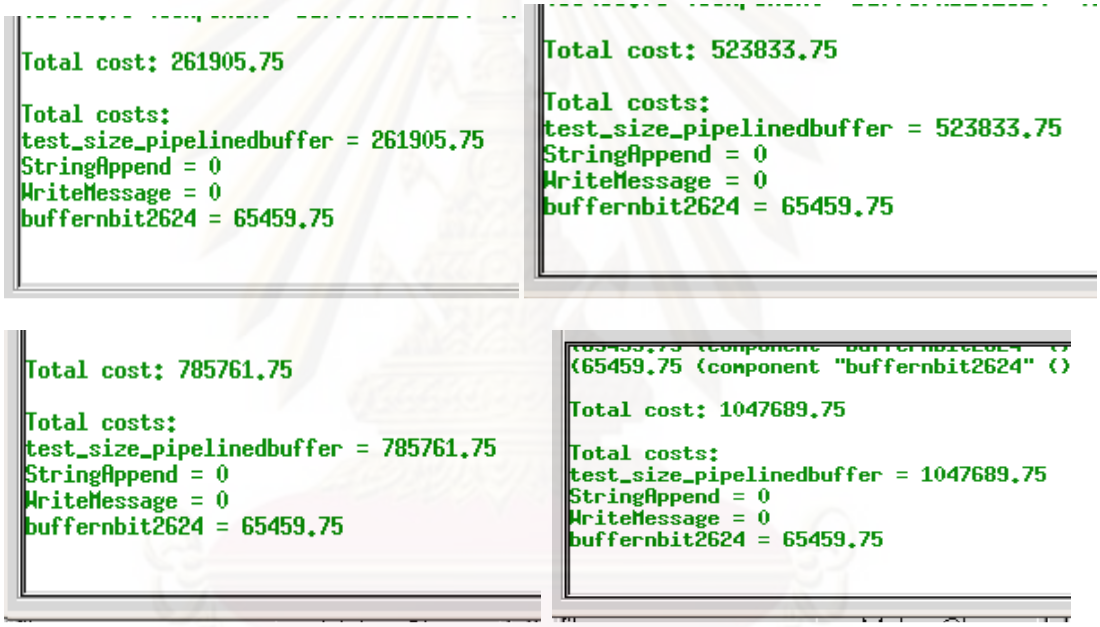
Name	Value	Used	Total Available	Utilization (%)
Clocks	0.00003 (W)	1	---	---
Logic	0.00002 (W)	330	46560	0.7
Signals	0.00013 (W)	425	---	---
IOs	0.00168 (W)	147	360	40.8
Total Quiescent Power	**1.50774 (W)**			
Total Dynamic Power	**0.00186 (W)**			
Total Power	**1.50959 (W)**			
Junction Temp	**50.0 (degrees C)**			

รูปที่ 4.17 ผลการทดสอบกำลังไฟฟ้าของหน่วยประมวลผลย่อยแบบสมวาร

จากผลการทดสอบ ดังรูปที่ 4.16 และรูปที่ 4.17 พบว่าหน่วยประมวลผลย่อยแบบอสมวารใช้เนื้อที่บนเอฟพีจีเอมากกว่าเนื่องจากเป็นการสร้างวงจรที่ใช้การเข้ารหัสรางคู่ และบริโภคกำลังไฟฟ้ามากกว่าแบบสมวาร ทั้งนี้ผลการทดสอบไม่สามารถสรุปได้อย่างชัดเจน เนื่องจากเอฟพีจีเอที่ใช้ในปัจจุบันถูกออกแบบมาเพื่อการสังเคราะห์วงจรแบบสมวาร การทดสอบกำลังไฟฟ้าที่มีความน่าเชื่อถือสูง ควรทดสอบการใช้กำลังไฟฟ้าด้วยแบบจำลองเมื่อแถวลำดับคำนวณถูกนำไปสร้างจริงเป็นวงจรเฉพาะกิจแล้ว

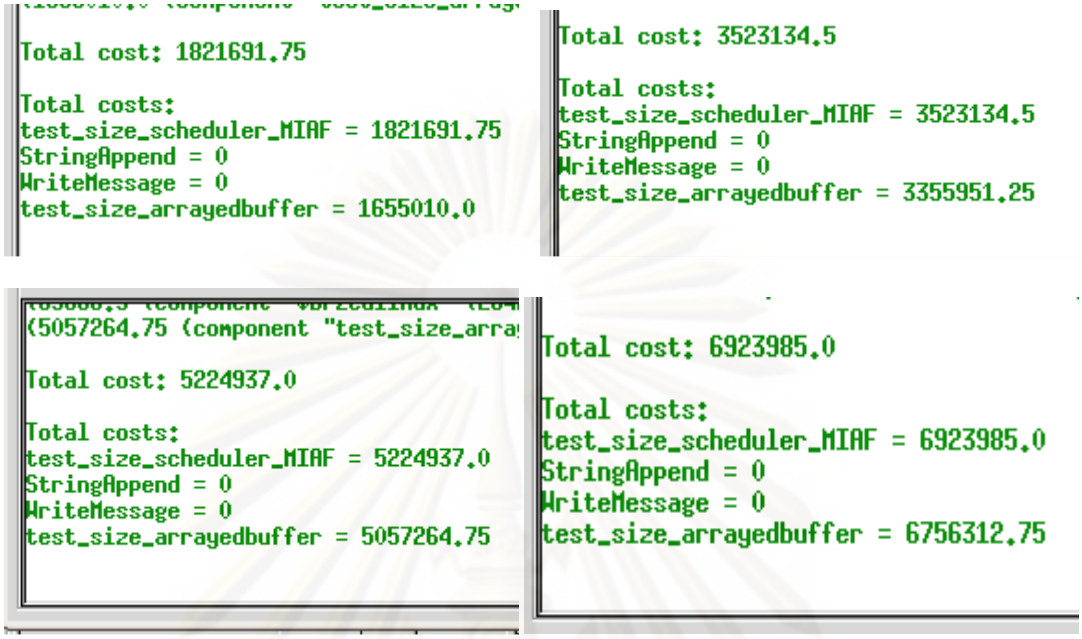
4.4 การประมาณค่าขนาดของตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธีที่แตกต่างกัน

ในส่วนนี้จะทดสอบเพื่อหาค่าขนาดของตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธี 2 วิธี คือ ขั้นตอนวิธีมาก่อนให้บริการก่อนและแบบพื้นที่น้อยที่สุดก่อน การทดสอบจะใช้เครื่องมือย่อยที่มาพร้อมกับโปรแกรมบาสซ่า คือ ตัวประมาณค่าใช้จ่ายพื้นที่ (Area cost estimator) ซึ่งจะบอกค่าขนาดของพื้นที่ที่ใช้ในวงจร โดยจะทดสอบกับตัวกำหนดการแต่ละแบบที่มีจำนวนเรจิสเตอร์ภายในที่แตกต่างกัน คือ มีจำนวน 4, 8, 12 และ 16 ตัว ผลการทดสอบเพื่อหาค่าขนาดของวงจร แสดงได้ดังรูปที่ 4.18 และ 4.19



รูปที่ 4.18 ผลการประมาณค่าขนาดของตัวกำหนดการแบบมาก่อนให้บริการก่อน

ที่มีเรจิสเตอร์ 4, 8, 12 และ 16 ตัว



รูปที่ 4.19 ผลการประมาณค่าขนาดของตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน

ที่มีเรจิสเตอร์ 4, 8, 12 และ 16 ตัว

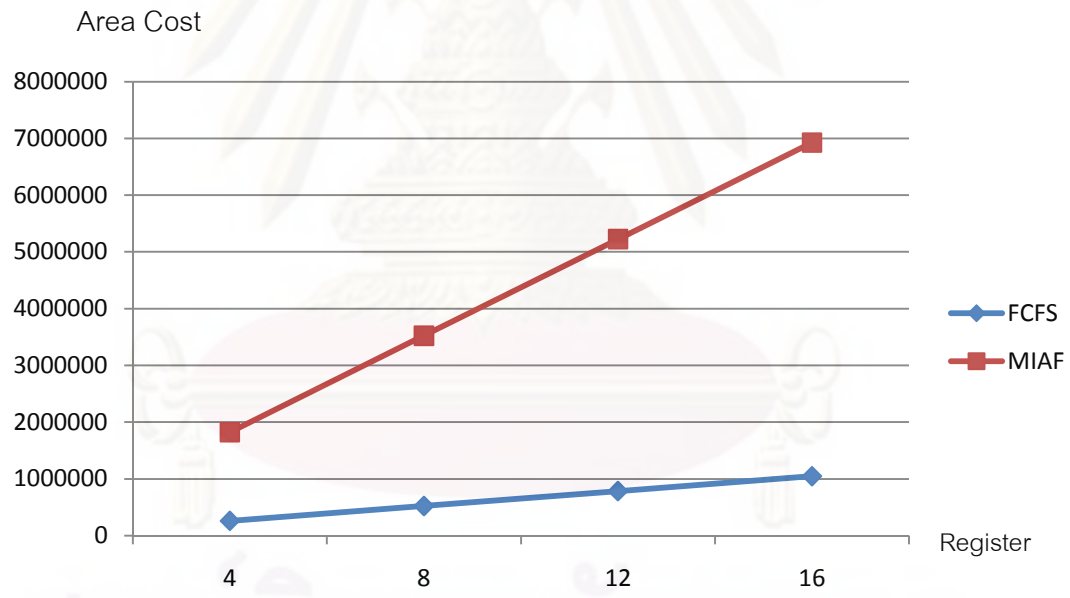
จากรูปที่ 4.18 และ 4.19 แสดงให้เห็นได้ว่า ขนาดของตัวกำหนดการทั้ง 2 แบบ จะเพิ่มขึ้นตามจำนวนเรจิสเตอร์ที่เพิ่มขึ้น โดยจะแสดงผลการทดสอบในรูปจากซ้ายไปขวาและบนลงล่าง ในรูปผลการทดสอบจะแสดงทั้งขนาดของหน่วยความจำที่ใช้ในตัวกำหนดการและผลรวมของขนาดของตัวกำหนดการเอง สามารถนำค่าจากผลการทดสอบมาแสดงได้ดังตารางที่ 4.2

ตารางที่ 4.2 ผลการเปรียบเทียบขนาดของตัวกำหนดการ 2 วิธี

จำนวนเรจิสเตอร์	**ขนาดของตัวกำหนดการ**	
	มาก่อนให้บริการก่อน	**พื้นที่น้อยที่สุดก่อน**
4	261,905.75	1,821,691.75
8	523,833.75	3,523,134.5

12	785,761.75	5,224,937
16	1,047,689.75	6,923,985

จากตารางที่ 4.2 จะเห็นได้ว่า ตัวกำหนดการที่ถูกออกแบบด้วยขั้นตอนวิธีมาก่อนให้บริการก่อนจะมีขนาดน้อยกว่าแบบพื้นที่น้อยที่สุดก่อนประมาณ 85% เนื่องจากตัวกำหนดการแบบมาก่อนให้บริการก่อนมีส่วนประกอบเพียงเรจิสเตอร์สายท่อเท่านั้น ส่วนตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อนใช้พื้นที่มากกว่า เนื่องจากมีส่วนประกอบย่อย 3 ส่วน ที่ใช้ในการหาค่าขนาดที่น้อยที่สุดของโครงแบบ ดังที่ได้กล่าวไปในหัวข้อที่ 3.2.3.4 สามารถนำค่าในตารางนี้ไปแสดงเป็นกราฟความสัมพันธ์ระหว่างจำนวนเรจิสเตอร์และขนาดของตัวกำหนดการได้ดังรูปที่ 4.20



รูปที่ 4.20 กราฟความสัมพันธ์ระหว่างจำนวนเรจิสเตอร์และขนาดของตัวกำหนดการ

จากรูปที่ 4.20 แสดงความสัมพันธ์ระหว่างจำนวนเรจิสเตอร์และขนาดของตัวกำหนดการทั้ง 2 วิธี จะเห็นได้ว่าตัวกำหนดการทั้ง 2 วิธี จะมีอัตราการเพิ่มของขนาดเป็นแบบเชิงเส้น โดยอัตราของตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อนจะมีค่ามากกว่าแบบมาก่อนให้บริการก่อนประมาณ 6.5 เท่า เนื่องจากหน่วยความจำที่ใช้เก็บโครงแบบมีความซับซ้อนกว่า รวมถึงมี

อาร์บิเตอร์เพื่อใช้ในการตรวจสอบการมาของสัญญาณ วงจรคำนวณค่าขนาดของโครงแบบ และวงจรควบคุมเพิ่มเข้ามาด้วย

4.5 การเปรียบเทียบประสิทธิภาพในการวางเมื่อใช้ตัวกำหนดการที่แตกต่างกัน

ในหัวข้อนี้จะกล่าวถึงการทดสอบประสิทธิภาพในการวางเมื่อทำงานร่วมกับตัวกำหนดการที่ออกแบบด้วยขั้นตอนวิธีมาก่อนให้บริการก่อนและแบบพื้นที่น้อยที่สุดก่อน ซึ่งอยู่บนสมมติฐานที่กำหนดให้แต่ละงานใช้เวลาเท่ากัน โดยสุ่มโครงแบบจำนวน 8 โครงแบบเพื่อส่งไปยังตัววางเพื่อทดสอบประสิทธิภาพ โดยมีตัวชี้วัดประสิทธิภาพดังนี้

1) จำนวนเซลล์ที่ถูกครอบครองต่ำสุด (MIOC : Minimal Occupied Cells) ตัวชี้วัดนี้จะแสดงถึงจำนวนต่ำสุดของหน่วยประมวลผลที่ถูกจองก่อนเกิดการรอเนื่องจากไม่สามารถวางโครงแบบต่อไปได้ ซึ่งจะบอกประสิทธิภาพที่ดีหากตัวชี้วัดนี้มีค่าสูง

2) จำนวนเซลล์ที่ถูกครอบครองสูงสุด (MAOC: Maximal Occupied Cells) ตัวชี้วัดนี้จะแสดงถึงจำนวนสูงสุดของหน่วยประมวลผลที่ถูกจองก่อนเกิดการรอเนื่องจากไม่สามารถวางโครงแบบต่อไปได้ ซึ่งจะบอกประสิทธิภาพที่ดีหากตัวชี้วัดนี้มีค่าสูง

3) จำนวนสถานะรอสูงสุด (MAWS: Maximal Wait States) ตัวชี้วัดนี้จะแสดงถึงจำนวนสถานะรอสูงสุดที่เกิดขึ้นกับโครงแบบ โดยจะนับจำนวนสถานะรอที่เกิดขึ้นก่อนโครงแบบถูกวางลงบนแถวลำดับคำนวณ ซึ่งจะบอกประสิทธิภาพที่ดีหากตัวชี้วัดนี้มีค่าต่ำ

4) จำนวนโครงแบบที่ถูกบรรจุสูงสุด (MALC: Maximal Loaded Configurations) ตัวชี้วัดนี้จะแสดงถึงจำนวนโครงแบบที่ถูกบรรจุสูงสุด โดยจะนับจำนวนโครงแบบที่ถูกวางลงบนแถวลำดับคำนวณก่อนเกิดสถานะรอในแต่ละครั้ง ซึ่งจะบอกประสิทธิภาพที่ดีหากตัวชี้วัดนี้มีค่าสูง กล่าวคือ มีจำนวนโครงแบบที่ทำงานในลักษณะขนาน (Parallel) อยู่ในแถวลำดับคำนวณมากกว่า

5) ค่าเฉลี่ยของสถานะรอ (AWS: Average of Wait States) ตัวชี้วัดนี้จะแสดงถึงค่าเฉลี่ยของสถานะรอ โดยแบ่งเป็นค่าเฉลี่ย 2 แบบ คือ ค่าเฉลี่ยเลขคณิต (Arithmetic Average) และค่ารากกำลังสองเฉลี่ย (RMS: Root Mean Square) ซึ่งจะบอกประสิทธิภาพที่ดีหากตัวชี้วัดนี้มีค่าต่ำ โดยสามารถคำนวณจากสมการดังต่อไปนี้

$$AWS_{\text{AVG}} = \frac{\sum_{i=1}^{n} WS_i}{n} \quad (4.1)$$

$$AWS_{RMS} = \sqrt{\frac{1}{\text{n}}\sum_{i=1}^{n} WS_i^2} \quad (4.2)$$

เมื่อ

WS_i คือ จำนวนสถานะรอของโครงแบบตัวที่ i

n คือ จำนวนโครงแบบที่ถูกส่งให้ตัววาง

6) ค่ามัธยฐานของสถานะรอ (MEWS: Median of Wait States) ตัวชี้วัดนี้จะแสดงถึงค่ามัธยฐานของสถานะรอ คือ ค่ากลางที่แบ่งครึ่งกลุ่มค่ามากของสถานะรอและกลุ่มค่าน้อยของสถานะรอที่เกิดขึ้นทั้งหมดกับงานที่ถูกส่งให้กับตัววาง ซึ่งจะบอกประสิทธิภาพที่ดีหากตัวชี้วัดนี้มีค่าต่ำ

7) ค่าฐานนิยมของสถานะรอ (MOWS: Mode of Wait States) ตัวชี้วัดนี้จะแสดงถึงค่าฐานนิยมของสถานะรอ คือ จำนวนความถี่สูงสุดของสถานะรอที่เกิดขึ้นทั้งหมดกับงานที่ถูกส่งให้กับตัววาง ซึ่งจะบอกประสิทธิภาพที่ดีหากตัวชี้วัดนี้มีค่าต่ำ

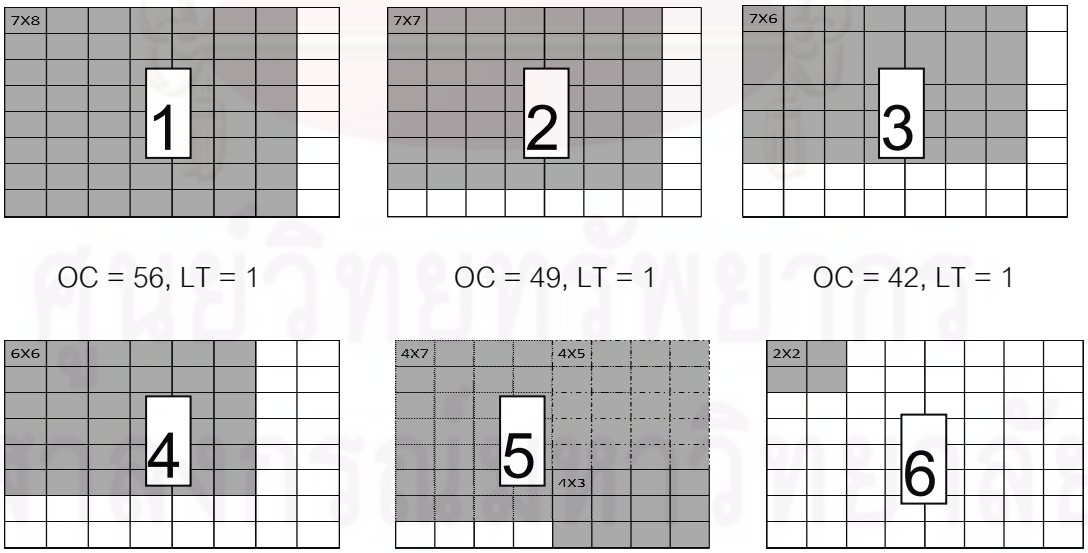
การเปรียบเทียบประสิทธิภาพจะสุ่มโครงแบบจำนวน 8 โครงแบบ ซึ่งมีขนาดแตกต่างกัน ผลต่างของขนาดโครงแบบในตัวถัดไปจะมีค่าใกล้เคียงหรือเท่ากับ 8 แสดงได้ดังตารางที่ 4.3

ตารางที่ 4.3 ขนาดและมิติของงานที่ถูกสุ่มเพื่อใช้ในการทดสอบประสิทธิภาพในการวาง

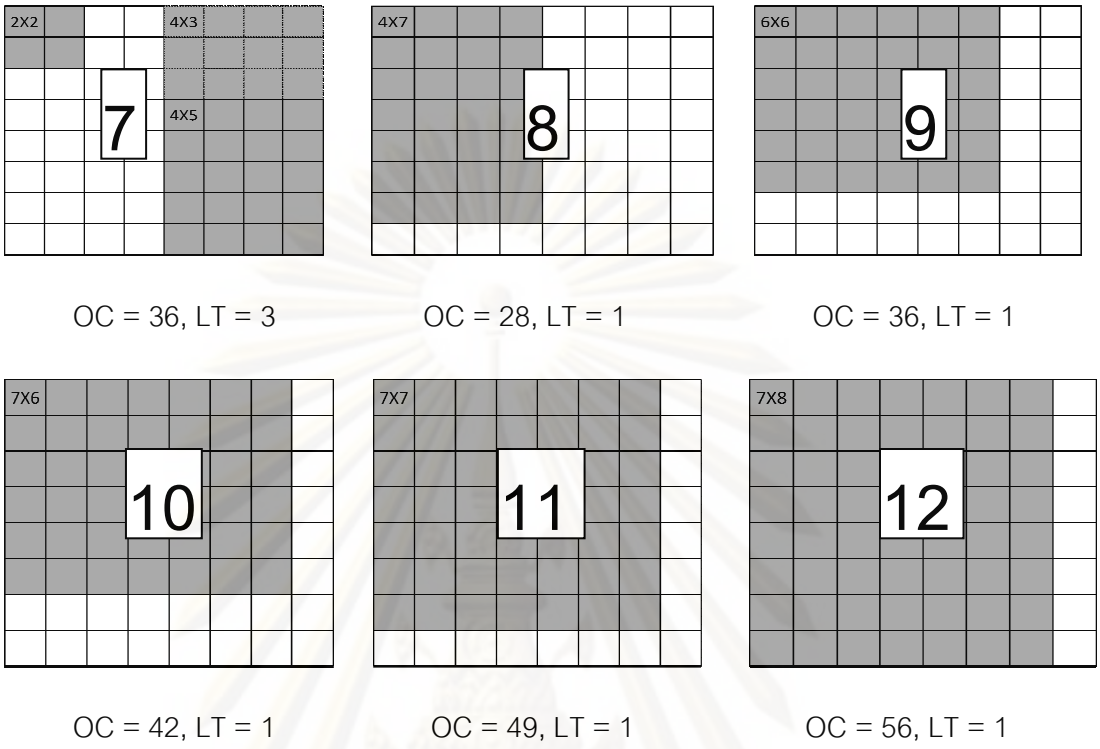
หมายเลขประจำงาน	ขนาด	มิติ
0	4	2X2
1	12	4X3
2	20	4X5

3	28	4X7
4	36	6X6
5	42	7X6
6	49	7X7
7	56	7X8

จากตารางที่ 4.3 แสดงขนาดและมิติของงานที่ถูกสุ่มเพื่อใช้ในการทดสอบการวาง โดยกำหนดให้ค่าขนาดแต่ละตัวต่างจากตัวต่อไปเท่ากับ 8 หรือใกล้เคียงที่สุดเท่าที่จะเป็นไปได้ เพื่อทำให้ขนาดของงานที่นำมาทดสอบมีการกระจายอย่างสม่ำเสมอ โดยงานหมายเลข 5, 6 และ 7 จะกำหนดให้มีขนาด 42, 49 และ 56 เนื่องจากมิติในแถวลำดับคำนวณไม่สามารถให้ค่า 44, 52 และ 60 ได้ จึงได้ใช้ค่าดังกล่าวซึ่งเป็นค่าที่ใกล้เคียงที่สุดในการทดสอบ แล้วนำไปวางลงบนแถวลำดับคำนวณขนาด 8X8 โดยกำหนดลำดับของขนาดโครงแบบที่วางให้มีค่าจากมากไปหาน้อย แล้วทดสอบด้วยขั้นตอนวิธีมาก่อนให้บริการก่อนและแบบพื้นที่น้อยที่สุดก่อน เพื่อนำไปหาค่าชี้วัด โดยจะแสดงลำดับการวางของทั้ง 2 วิธี ได้ดังรูปต่อไปนี้



รูปที่ 4.21 ลำดับการวางเมื่อใช้ขั้นตอนวิธีมาก่อนให้บริการก่อน



รูปที่ 4.22 ลำดับการวางเมื่อใช้ขั้นตอนวิธีพื้นที่น้อยที่สุดก่อน

จากรูปที่ 4.21 และ 4.22 ค่า OC (Occupied Cells) และค่า LT (Loaded Tasks) แสดงถึงจำนวนเซลล์ที่ถูกจองและจำนวนงานที่ถูกวางลงบนแถวลำดับคำนวณตามลำดับภาพแถวลำดับคำนวณแต่ละภาพจะแสดงถึงลำดับการวางโครงแบบบนแถวลำดับคำนวณ โดยไล่จากซ้ายไปขวาและบนลงล่าง จะเห็นได้ว่าการวางที่ได้จากขั้นตอนวิธีมาก่อนให้บริการจะเริ่มวางโครงแบบจากขนาด 7X8 ซึ่งมีขนาดมากที่สุด หลังจากนั้นจะเกิดสถานะรอจนกว่าโครงแบบจะประมวลผลเสร็จสิ้นแล้วจึงวางโครงแบบต่อไป คือ ขนาด 7X7 จนครบ 8 โครงแบบ โดยวางโครงแบบขนาด 2X2 เป็นลำดับสุดท้าย ส่วนการวางที่ได้จากขั้นตอนวิธีพื้นที่น้อยที่สุดก่อนจะเริ่มวางจากโครงแบบขนาด 2X2 และมีอีก 2 โครงแบบที่สามารถถูกวางได้พร้อมกันโดยไม่เกิดสถานะรอ หลังจากนั้นโครงแบบขนาด 4X7 จะต้องรอให้โครงแบบทั้ง 3 ทำงานเสร็จสิ้นก่อน จึงถูกวางเป็นลำดับต่อไปบนแถวลำดับคำนวณ และจะวางโครงแบบที่เหลือต่อไปแสดงได้ดังรูปจนครบถึงงานที่มีขนาด 7X8 เป็นลำดับสุดท้าย จากการทดลองนี้สามารถแสดงสถานะรอของแต่ละงานได้ดังตารางที่ 4.4 และตารางที่ 4.5

ตารางที่ 4.4 สถานะรอที่เกิดขึ้นเมื่อใช้ขั้นตอนวิธีมาก่อนให้บริการก่อน

หมายเลขประจำงาน	ขนาด	มิติ	จำนวนสถานะรอที่เกิดขึ้นก่อนถูกวาง
0	4	2X2	5
1	12	4X3	4
2	20	4X5	4
3	28	4X7	4
4	36	6X6	3
5	42	7X6	2
6	49	7X7	1
7	56	7X8	0

จากตารางที่ 4.4 จะเห็นได้ว่างานหมายเลข 7 ไม่มีสถานะรอเนื่องจากถูกวางเป็นลำดับแรก หลังจากนั้นงานต่อไปจะถูกวางได้ครั้งละหนึ่งงาน เนื่องจากมีขนาดใหญ่จนไม่สามารถทำให้งานต่อไปถูกวางได้ เมื่อถึงสถานะรอครั้งที่ 4 งานที่ 1, 2 และ 3 จึงสามารถถูกวางได้พร้อมกัน และเกิดสถานะรออีกหนึ่งครั้ง หลังจากนั้น งานหมายเลข 0 จึงถูกวางลงบนแถวลำดับคำนวณเป็นลำดับสุดท้าย

ตารางที่ 4.5 สถานะรอที่เกิดขึ้นเมื่อใช้ขั้นตอนวิธีพื้นที่น้อยที่สุดก่อน

หมายเลขประจำงาน	ขนาด	มิติ	จำนวนสถานะรอที่เกิดขึ้นก่อนถูกวาง
0	4	2X2	0
1	12	4X3	0
2	20	4X5	0

3	28	4X7	1
4	36	6X6	2
5	42	7X6	3
6	49	7X7	4
7	56	7X8	5

จากตารางที่ 4.5 จะเห็นได้ว่า งานหมายเลข 0, 1 และ 2 ไม่มีสถานะรอก่อนถูกวาง เนื่องจากมีพื้นที่เพียงพอสำหรับ 3 งานนี้ หลังจากนั้นงานแต่ละงานจะมีขนาดใหญ่จนไม่สามารถทำให้งานลำดับถัดไปถูกวางลงได้พร้อมกัน จึงต้องรองานที่ถูกวางให้ทำงานเสร็จสิ้นไปเป็นลำดับจนถึงการวางงานหมายเลข 7 เป็นลำดับสุดท้าย

จากค่าสถานะรอที่เกิดขึ้นกับโครงแบบดังได้แสดงในตารางข้างต้น สามารถนำมาคำนวณหาค่าชี้วัดประสิทธิภาพ แสดงได้ดังตารางที่ 4.6

ตารางที่ 4.6 ผลการเปรียบเทียบประสิทธิภาพในการวางเมื่อใช้ตัวกำหนดการที่แตกต่างกัน

ค่าชี้วัด		FCFS	MIAF
MIOC		4	28
MAOC		60	56
MALC		3	3
MAWS		5	5
AWS	AWS_{AVG}	2.875	1.875
	AWS_{RMS}	3.3	2.62

MEWS	3.5	1.5
MOWS	4	0

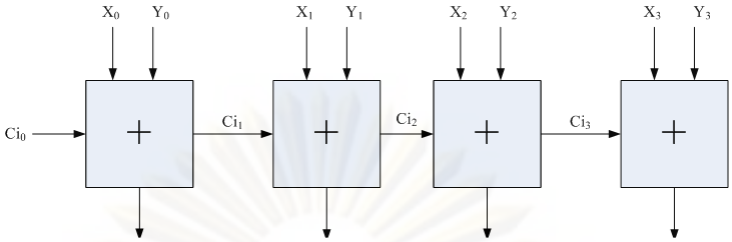
จากตารางที่ 4.6 แสดงการเปรียบเทียบค่าชี้วัดของตัวกำหนดการทั้ง 2 วิธี โดยค่าชี้วัดที่ถูกขีดเส้นใต้ในตารางจะแสดงถึงค่าที่ดีกว่า จะเห็นได้ว่า ค่าชี้วัดของขั้นตอนวิธีพื้นที่น้อยที่สุดก่อนจะให้ค่าเท่ากับหรือดีกว่าแบบมาก่อนให้บริการก่อนเป็นส่วนใหญ่ จากตารางจะเห็นได้ว่าค่า MIOC คือ ค่าจำนวนเซลล์ที่ถูกครอบครองต่ำสุดของตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อนให้ค่าที่สูงกว่า ซึ่งมีค่าเท่ากับ 28 แสดงได้ดังภาพแถวลำดับคำนวณที่ 8 และค่า MIOC ของวิธีมาก่อนให้บริการก่อนเท่ากับ 4 แสดงได้ดังภาพแถวลำดับคำนวณที่ 6 ซึ่งจะให้ค่าหน่วยประมวลผลที่ถูกครอบครองดีกว่าถึง 24 ตัว ในขณะที่ค่า MAOC ของวิธีมาก่อนให้บริการก่อนจะให้ค่าครอบครองที่ดีกว่าเพียง 4 ตัวเท่านั้น ค่า MALC และ MAWS ของทั้งสองวิธีจะมีค่าเท่ากัน คือ มีจำนวนโครงแบบสูงสุดที่ทำงานพร้อมกันในแถวลำดับคำนวณ 3 โครงแบบ และมีค่าสถานะรอสูงสุดเท่ากับ 5 สถานะ ค่าชี้วัดที่เหลือที่เป็นค่าทางสถิติ คือ ค่าเฉลี่ยเลขคณิต AWS_{AVG} และค่ารากกำลังสองเฉลี่ย AWS_{RMS} ของวิธีพื้นที่น้อยที่สุดก่อนจะให้ค่าที่ดีกว่าวิธีมาก่อนให้บริการก่อน ส่วนค่ามัธยฐานและค่าฐานนิยมของวิธีพื้นที่น้อยที่สุดให้ค่าที่ดีกว่าเช่นเดียวกัน

4.6 **ความเป็นไปได้ในการนำวงจรอสมวารมาประมวลผลบนแถวลำดับคำนวณ**

ในหัวข้อนี้จะวิเคราะห์เพิ่มเติมจากหัวข้อ 4.1.2 ซึ่งเป็นการทดสอบวงจรอสมวารที่สร้างสัญญาณไปควบคุมการไหลของข้อมูลในวงจรที่มีลักษณะวนซ้ำ หัวข้อนี้จะยกตัวอย่างวงจรอีก 2 วงจร เพื่ออธิบายถึงความเป็นไปได้ในการนำวงจรอสมวารที่ออกแบบโดยวิธีโครงสร้างการไหลของข้อมูลแบบสถิตมาประมวลผลบนแถวลำดับคำนวณ โดยมีตัวอย่างดังนี้

4.6.1 **วงจรบวกแบบส่งต่อสัญญาณตัวทด** (RCA: Ripple Carry Adder)

ในหัวข้อนี้จะกล่าวถึงตัวอย่างของวงจรเชิงผสม ซึ่งแตกต่างจากในหัวข้อ 4.1.2 ที่เป็นวงจรสร้างสัญญาณควบคุม ตัวอย่างที่จะกล่าวถึงในหัวข้อนี้ คือ วงจรบวกแบบส่งต่อสัญญาณตัวทด วงจรบวกประเภทนี้จะประกอบขึ้นจากวงจรบวกหลายตัวต่อกันในลักษณะลูกโซ่ที่ส่งต่อสัญญาณตัวทดไปยังวงจรบวกตัวถัดไปเพื่อบวกเลขหลายหลัก แสดงได้ดังรูปที่ 4.23

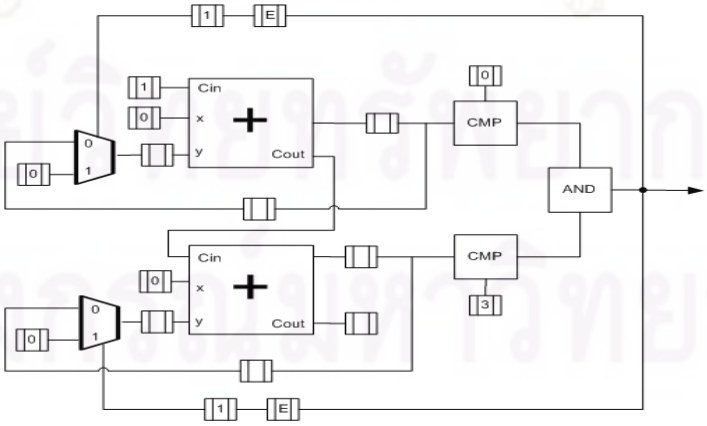


รูปที่ 4.23 วงจรบวกแบบส่งต่อสัญญาณตัวทด

จากรูปที่ 4.23 แสดงถึงความเป็นไปได้ในการนำวงจรบวกนี้มาประมวลผลบนแถวลำดับคำนวณ คือ สามารถปรับเปลี่ยนโครงแบบหน่วยประมวลผลย่อยให้เป็นวงจรบวกและกำหนดให้สัญญาณตัวทดถูกส่งต่อไปยังหน่วยประมวลผลย่อยตัวถัดไป หากมีหน่วยประมวลผลย่อยรอบข้างที่ถูกจองแล้วและใช้เฉพาะส่วนเส้นทางข้อมูล 8 บิต ก็ยังคงสามารถส่งต่อสัญญาณตัวทดผ่านไปได้เนื่องจากในหน่วยประมวลผลย่อยมีเส้นทางส่งต่อสัญญาณตัวทดที่แยกออกจากกันไว้

4.6.2 วงจรสร้างสัญญาณควบคุมวนซ้ำที่มีขนาด 16 บิต

วงจรนี้จะแตกต่างจากวงจรสร้างสัญญาณควบคุมที่อธิบายในหัวข้อ 4.1.2 โดยจะใช้วงจรบวก 2 ตัว เพื่อสร้างสัญญาณควบคุมที่มีการวนซ้ำมากกว่า 255 ครั้ง แสดงได้ดังรูปที่ 4.16



รูปที่ 4.24 วงจรสร้างสัญญาณควบคุมวนซ้ำที่มีขนาด 16 บิต

จากรูปที่ 4.24 เป็นวงจรที่ถูกสร้างขึ้นโดยวิธีโครงสร้างการไหลของข้อมูลแบบสถิต โดยมีการกำหนดค่าเริ่มต้นให้กับแลตช์ที่ส่งค่าให้กับตัวเลือกส่งสัญญาณเป็น 1 เพื่อเลือกรับสัญญาณจากแลตช์ที่ป้อนค่าคงที่ 0 เป็นการเริ่มต้นการทำงานของวงจร ในตัวอย่างนี้มีค่าการวนซ้ำเท่ากับการนำค่าในแลตช์ตัวล่างและแลตช์ตัวบนที่ต่อเข้ากับตัวเปรียบเทียบมาประชิดกัน ซึ่งได้ค่า 00000011_00000000 ในเลขฐาน 2 เท่ากับค่า 768 ในเลขฐาน 10 เป็นจำนวนครั้งของการวนซ้ำ

4.7 **สรุปผลการทดลอง**

ในหัวข้อนี้จะสรุปผลการทดลองทั้งหมดโดยแบ่งการสรุปตามหัวข้อที่ได้กล่าวมาข้างต้นได้ดังนี้

1) การทดสอบการทำงานของแถวลำดับคำนวณขนาด 8X8 ที่ได้ออกแบบไว้ พบว่าวงจรสามารถทำงานได้อย่างถูกต้องในระดับการไหลของข้อมูล และการทดสอบกับแถวลำดับคำนวณขนาด 3X2 กับโครงแบบที่มีการใช้เส้นทางข้อมูลทั้งที่มีขนาด 8 บิต และ 1 บิต ได้ทดสอบการทำงานในระดับพฤติกรรม โดยนำงานประยุกต์มาจัดกลุ่มและวางลงบนแถวลำดับคำนวณ พบว่าวงจรสามารถทำงานได้อย่างถูกต้องเช่นเดียวกัน และยังแสดงถึงความเป็นไปได้ในการนำงานประยุกต์ที่เขียนจากรหัสต้นฉบับในภาษาระดับสูงมาประมวลผลบนแถวลำดับคำนวณ

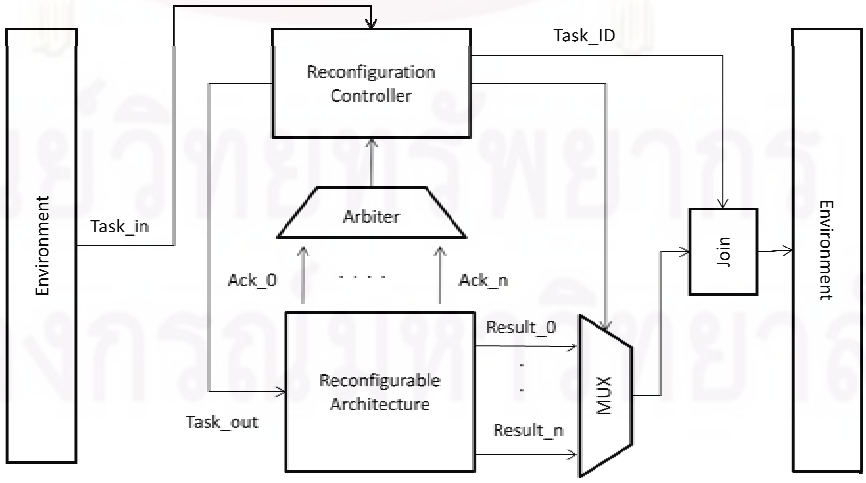
2) การทดสอบการทำงานของตัวควบคุมได้ทดสอบการทำงานแต่ละส่วนประกอบภายใน คือ ตัวกำหนดการ ตัววาง และตัวบรรจุ การทดสอบการทำงานทั้ง 3 ส่วนนี้ ได้ทดสอบในระดับการไหลของข้อมูลเช่นเดียวกับการทดสอบแถวลำดับคำนวณ พบว่าวงจรสามารถทำงานได้อย่างถูกต้อง

3) การทดสอบกำลังไฟฟ้าเปรียบเทียบระหว่างหน่วยประมวลผลแบบสมวารและแบบอสมวาร ได้ทดสอบในระดับเบื้องต้นโดยการปรับเปลี่ยนให้หน่วยประมวลผลย่อยเป็นวงจรบวก และได้ทดสอบการใช้กำลังไฟฟ้าบนเทคโนโลยีเอฟพีจีเอในระดับวางและจัดเส้นทาง พบว่าหน่วยประมวลผลแบบอสมวารใช้พื้นที่มากกว่าและสิ้นเปลืองพลังงานมากกว่า การทดสอบที่มีความน่าเชื่อถือสูง ควรทดสอบเมื่อนำหน่วยประมวลผลย่อยมาเชื่อมต่อกันเป็นแถวลำดับคำนวณแล้วทดสอบในแบบจำลองที่เป็นวงจรเฉพาะกิจ

4) จากการทดสอบการหาค่าพื้นที่ที่ใช้ของตัวกำหนดการทั้ง 2 แบบ สามารถสรุปได้ว่า ตัวกำหนดการแบบมาก่อนให้บริการก่อนใช้พื้นที่น้อยกว่าถึง 85% เมื่อมีจำนวนเรจิสเตอร์ที่อยู่ภายในเท่ากัน ซึ่งเหมาะสมกับการนำตัวควบคุมนี้ไปใช้จริงในกรณีที่ให้ความสำคัญเรื่องขนาดของวงจร และถ้าระบบภายนอกเรียงลำดับงานให้เหมาะสมอยู่แล้ว จะส่งผลให้ไม่จำเป็นต้องใช้ตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน ซึ่งจะสามารถลดพื้นที่และขั้นตอนการเรียงลำดับของโครงแบบลงได้

5) จากการทดสอบวัดค่าประสิทธิภาพในการวางเมื่อใช้ตัวกำหนดการที่แตกต่างกันทั้ง 2 วิธี สามารถสรุปได้ว่า ตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อนสามารถช่วยให้การวางของตัววางมีประสิทธิภาพมากขึ้น กล่าวคือ ให้ค่าจำนวนหน่วยประมวลผลย่อยที่ถูกครอบครองน้อยที่สุดก่อนเกิดสถานะรอมากกว่าและให้ค่าสถานะรอของโครงแบบที่น้อยกว่า ทั้งนี้จะต้องเสียพื้นที่ในวงจรมากขึ้นดังที่ได้กล่าวไปแล้วข้างต้น หากให้ความสำคัญต่อเวลารอคอยของงานที่ถูกนำเข้ามาประมวลผล

งานวิจัยนี้ได้ออกแบบและทดสอบตัวควบคุมและแถวลำดับคำนวณที่ปรับเปลี่ยนโครงแบบบางส่วนได้แบบอสมวาร วงจรทั้งสองส่วนนี้จะต้องทำงานประสานกัน การนำตัวควบคุมและแถวลำดับคำนวณมาเชื่อมต่อเพื่อทำงานร่วมกันนั้น อาจมีรายละเอียดปลีกย่อยที่ต้องเพิ่มเติมหรือดัดแปลง เมื่อสองส่วนประกอบนี้ถูกออกแบบด้วยขั้นตอนวิธีที่ใช้ในตัวกำหนดการและตัววางที่แตกต่างออกไป รวมถึงวิธีการออกแบบแถวลำดับคำนวณด้วยเช่นกัน ในบทนี้จึงได้เสนอแนวทางในการเชื่อมต่อของตัวควบคุมและแถวลำดับคำนวณไว้ดังรูปที่ 4.25



รูปที่ 4.25 ตัวอย่างการนำตัวควบคุมและสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้มาทำงานร่วมกัน

จากรูปที่ 4.25 แสดงถึงการนำตัวควบคุมมาใช้ในการปรับเปลี่ยนโครงแบบให้กับสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ โดยตัวควบคุมจะรับงานเข้ามาจากวงจรภายนอกแล้วเลือกงานที่เหมาะสมให้กับสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ เมื่องานถูกประมวลผลเสร็จจะส่งสัญญาณตอบกลับ Ack ไปยังอาร์บิเตอร์เพื่อเรียงลำดับของผลลัพธ์ที่จะถูกส่งออกไปภายนอก เมื่อตัวควบคุมได้รับสัญญาณจากอาร์บิเตอร์ซึ่งเป็นตำแหน่งของหน่วยประมวลผลที่มีสัญญาณผลลัพธ์แล้ว จะนำค่านี้ไปหาพื้นที่ที่จะถูกคืนให้กับระบบ และส่งสัญญาณเลือกข้อมูลให้กับอุปกรณ์รวมส่งสัญญาณเพื่อเลือกผลลัพธ์นั้นส่งออกไปยังวงจรภายนอกต่อไป โดยผลลัพธ์นี้จะถูกแนบหมายเลขประจำงานเพิ่มเข้าไปเพื่อให้วงจรภายนอกทราบได้ว่าเป็นผลลัพธ์จากงานใด

ในบทนี้ได้กล่าวถึงการทดสอบแถวลำดับคำนวณและตัวควบคุม การเปรียบเทียบกำลังไฟฟ้า การประมาณค่าขนาดของตัวกำหนดการ การเปรียบเทียบประสิทธิภาพในการวางความเป็นไปได้ในการนำวงจรอสมวารมาประมวลผลบนแถวลำดับคำนวณที่ออกแบบไว้ และสรุปผลการทดลอง การทดสอบยังคงเป็นไปได้อย่างจำกัด เนื่องจากเครื่องมือที่ใช้ออกแบบและทดสอบวงจรอสมวารมีความแพร่หลายน้อยมากเมื่อเทียบกับเครื่องมือที่ใช้พัฒนาวงจรสมวาร ต่อไปจะกล่าวถึงการสรุปผลการวิจัย ข้อจำกัดของงานวิจัย และข้อเสนอแนะ ในบทที่ 5

บทที่ 5

สรุปผลการวิจัย ข้อจำกัดของงานวิจัย และข้อเสนอแนะ

ในบทนี้จะกล่าวถึงการสรุปผลการวิจัยถึงสิ่งที่ได้ออกแบบไว้และข้อดีต่างๆ กล่าวถึงข้อจำกัดของงานวิจัย และข้อเสนอแนะของงานที่สามารถนำไปพัฒนาต่อยอดได้ โดยมีรายละเอียดดังนี้

5.1 สรุปผลการวิจัย

สถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้มีอยู่หลายโครงสร้างและจุดประสงค์ของการใช้งานที่แตกต่างกันออกไป ซึ่งโดยส่วนใหญ่จะเป็นโครงสร้างแบบแถวลำดับ 2 มิติ รวมถึงมีงานวิจัยเป็นจำนวนมากที่ทำการวิจัยเกี่ยวกับการจัดการโครงแบบสำหรับโครงสร้างประเภทนี้ งานวิจัยนี้จึงได้เลือกโครงสร้างแบบแถวลำดับ 2 มิติ มาเป็นแนวทางในการออกแบบตัวควบคุมและแถวลำดับคำนวณ โดยได้ออกแบบตัวควบคุมที่ใช้ตัวกำหนดการไว้ 2 วิธี คือ วิธีมาก่อนให้บริการก่อนและวิธีพื้นที่น้อยที่สุดก่อน ได้ทดสอบเพื่อเปรียบเทียบข้อดีของทั้ง 2 วิธีนี้ วิธีแรกมีข้อดีเมื่อเทียบกับวิธีที่สอง คือ ใช้พื้นที่น้อยกว่าและสามารถออกแบบได้ง่ายโดยใช้เรจิสเตอร์สายท่อเท่านั้น ส่วนวิธีที่สองมีความซับซ้อนและใช้พื้นที่มากกว่า แต่ช่วยให้ตัววางสามารถวางโครงแบบได้อย่างมีประสิทธิภาพมากขึ้น และงานวิจัยนี้ยังได้ออกแบบแถวลำดับคำนวณ 2 มิติ ที่สามารถปรับเปลี่ยนโครงแบบบางส่วนอย่างหยาบได้แบบบอสมวาร คือ สามารถถูกปรับเปลี่ยนพื้นที่บางส่วนได้ในขณะที่พื้นที่ส่วนอื่นกำลังทำงานอยู่ โครงสร้างของแถวลำดับคำนวณและหน่วยประมวลผลย่อยในงานวิจัยนี้ ถูกออกแบบโดยมีจุดประสงค์เพื่อให้รองรับการประมวลผลงานประยุกต์ทั่วไปได้ จึงได้ออกแบบให้มีเส้นทางข้อมูลขนาด 1 บิต เพื่อรองรับงานที่ต้องการความยืดหยุ่นสูง แต่อย่างไรก็ตามการเพิ่มเส้นทางข้อมูลขนาด 1 บิต นี้ไม่ได้ส่งผลกระทบต่อการใช้พื้นที่ในการออกแบบโดยตรง เนื่องจากโดยปกติแล้วงานประเภทต่างๆ มักจะมีส่วนของสัญญาณเงื่อนไขรวมอยู่ด้วย เส้นทางข้อมูลขนาด 1 บิต นี้ สามารถถูกปรับเปลี่ยนเพื่อรองรับการทำงานในส่วนนี้ได้เช่นกัน

5.2 ข้อจำกัดของงานวิจัย

ในหัวข้อนี้จะกล่าวถึงข้อจำกัดของงานวิจัย ซึ่งข้อจำกัดเหล่านี้สามารถเป็นประเด็นวิจัยเพื่อพัฒนาต่อยอดจากงานวิจัยนี้ได้ โดยมีรายละเอียดดังนี้

1) โครงแบบที่ถูกส่งให้ตัวควบคุมเพื่อนำไปประมวลผลจะต้องมีลักษณะเป็นสี่เหลี่ยมมุมฉาก ซึ่งหากโครงแบบที่ต้องการนำมาประมวลผลมีลักษณะอื่น จะต้องถูกทำให้มีลักษณะเป็นสี่เหลี่ยมมุมฉาก จึงทำให้เกิดการจองพื้นที่ในแถวลำดับคำนวณโดยเปล่าประโยชน์ และยังต้องใช้หน่วยความจำเพื่อเก็บโครงแบบในส่วนที่ถูกเติมเต็มก่อนส่งเข้าตัวควบคุมด้วยเช่นกัน

2) โครงแบบที่ถูกนำมาประมวลผลในแถวลำดับคำนวณไม่สามารถถูกแทรกแซงได้ จะต้องทำงานจนกว่าจะได้ผลลัพธ์ออกมา

3) โครงแบบที่ถูกนำมาประมวลผลสามารถประมวลผลจนสิ้นสุดได้ 1 ครั้งเท่านั้น หลังจากนั้นพื้นที่ที่ถูกจองไว้จะถูกคืนให้กับระบบเพื่อรองรับโครงแบบอื่นต่อไป

4) การกำหนดการที่ใช้ในงานวิจัยนี้ไม่ได้คำนึงเวลาที่โครงแบบใช้ประมวลผลในแถวลำดับคำนวณ ซึ่งเป็นตัวแปรหนึ่งที่สามารถทำให้การจัดการโครงแบบมีประสิทธิภาพมากขึ้นได้

5) โครงแบบที่ถูกนำมาประมวลผลจะต้องมีสัญญาณขาออกเพียงหนึ่งสัญญาณเท่านั้น

5.3 ข้อเสนอแนะ

งานวิจัยนี้ยังสามารถถูกนำไปพัฒนาต่อยอดได้อีกในหลายประเด็น ดังต่อไปนี้

1) พัฒนาตัวกำหนดการให้ทำงานร่วมกับตัววางในการหาพื้นที่ที่เหมาะสมที่สุดในสภาวะต่างๆ

2) พัฒนาให้สามารถรองรับโครงแบบที่มีลักษณะอื่นที่ไม่ใช่สี่เหลี่ยมมุมฉากเพื่อการใช้แถวลำดับคำนวณได้อย่างมีประสิทธิภาพมากขึ้น

3) พัฒนาให้สามารถใช้เส้นทางข้อมูลขนาด 1 บิต พร้อมกับเส้นทางที่มีขนาด 8 บิต ได้ โดย 2 เส้นทางนั้นมาจากโครงแบบคนละตัวกัน

4) พัฒนาซอฟต์แวร์เพื่อช่วยในการออกแบบและทดสอบในการพัฒนางานวิจัยนี้ต่อไป เนื่องจากปัจจุบันซอฟต์แวร์ที่ถูกใช้เป็นเครื่องมือในการออกแบบวงจรอสมวารยังไม่มีความแพร่หลายเมื่อเทียบกับการออกแบบวงจรสมวาร

5) โครงสร้างของแถวลำดับคำนวณที่ได้เสนอไว้ในงานวิจัยนี้ ยังคงต้องได้รับการวิจัยต่อไปถึงความสามารถในการรองรับงานประยุกต์ในด้านต่างๆ

6) พัฒนาให้มีการบรรจุโครงแบบซ้ำได้หลังจากการประมวลผลสิ้นสุดลง เพื่อรองรับการประมวลผลครั้งใหม่ที่ใช้โครงแบบเดิมแต่มีข้อมูลขาเข้าเปลี่ยนไป

7) พัฒนาให้สามารถหมุนโครงแบบ (Task rotation) เพื่อให้สามารถวางบนพื้นที่ที่มีความกว้างและความยาวที่เพียงพอหลังจากการหมุนแล้ว

8) พัฒนาตัวแปลโปรแกรมเพื่อให้สามารถนำงานประยุกต์ที่ถูกเขียนขึ้นด้วยภาษาคอมพิวเตอร์ระดับสูง มาใช้กับสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ ที่พัฒนาขึ้นในงานวิจัยนี้ได้โดยอย่างอัติโนมัติ

9) พัฒนาให้สามารถเก็บโครงแบบด้วยโครงสร้างข้อมูลที่ยืดหยุ่นกับขนาดของโครงแบบ

10) มีการกำหนดลำดับความสำคัญ (Priority) หรือการกำหนดเส้นตาย (Deadline) ให้กับงานที่เข้ามา ซึ่งเหมาะสมกับงานประยุกต์แบบเวลาจริง (Real time) หรืองานที่มีเวลาจำกัดในการรอการประมวลผล และสามารถเพิ่มประสิทธิภาพให้กับตัวกำหนดการในการยินยอมให้งานที่เข้ามาใหม่สามารถถูกวางได้ก่อนงานที่ติดค้างอยู่

11) พิจารณาถึงการออกแบบตัวเลือกข้อมูลขาเข้าขนาด 8 บิต และ 1 บิต ภายในหน่วยประมวลผลย่อย เนื่องจากการส่งข้อมูลระหว่างหน่วยประมวลผลอาจมีเส้นทางตายตัวที่ถูกกำหนดจากตัวจัดเส้นทางข้อมูลขาออกไปยังปลายทางอยู่แล้ว จึงอาจไม่จำเป็นต้องมีการเลือกข้อมูลขาเข้า หากสามารถปรับปรุงส่วนประกอบ 2 ส่วนนี้ให้เป็นตัวผสานข้อมูลได้ จะทำให้สามารถลดปริมาณสัญญาณควบคุมที่ใช้สำหรับปรับเปลี่ยนโครงแบบลงได้

จากการทำวิจัยนี้ ผู้วิจัยพบว่า ในปัจจุบันเครื่องมือที่ช่วยออกแบบวงจรอสมวารมีความแพร่หลายน้อยมากเมื่อเทียบกับวงจรสมวาร ทำให้เกิดอุปสรรคในการทำวิจัยทั้งในเรื่องของการออกแบบ การทวนสอบ (Verification) และการแก้จุดบกพร่อง (Debug) รวมถึงการทำวิจัยเรื่องสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ มีรายละเอียดปลีกย่อยค่อนข้างมากที่สามารถนำไปพัฒนาต่อยอดได้ งานวิจัยนี้ยังคงเป็นเพียงการออกแบบตัวควบคุมและสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ที่มีความซับซ้อนน้อย การออกแบบทั้ง 2 ส่วนนี้ จะส่งผลถึงงานวิจัยในส่วนอื่นที่เกี่ยวข้อง เช่น ความสามารถในการรองรับงานประยุกต์ การออกแบบตัวแปลภาษา เป็นต้น ผู้วิจัยมีข้อเสนอแนะในการนำงานวิจัยนี้ไปพัฒนาต่อยอดว่า ควรพิจารณาถึงความเหมาะสมของเครื่องมือและอุปกรณ์ที่ใช้ในการออกแบบและการทดลอง ให้มีความเพียงพอต่องานที่ออกแบบไว้ ผู้วิจัยที่จะนำงานวิจัยนี้ไปพัฒนาต่อ อาจต้องสร้างเครื่องมือที่ช่วยทดลองขึ้นเอง เช่น โปรแกรมที่ช่วยทดสอบการทำงานของตัวควบคุมและสถาปัตยกรรม เป็นต้น เพื่อลดเวลาในการดำเนินงานวิจัย การวัดประสิทธิภาพทางด้านความเร็ว การใช้พื้นที่ และกำลังไฟฟ้า ควรถูกทดสอบกับเทคโนโลยีที่ใช้ในการสร้างวงจรเฉพาะกิจ เพื่อให้ได้ผลการทดสอบที่มีความน่าเชื่อถือสูง

ตัวควบคุมและสถาปัตยกรรมที่ปรับเปลี่ยนโครงแบบได้ในงานวิจัยนี้ ถูกออกแบบขึ้นในระดับการไหลของข้อมูล เพื่อเสนอเป็นแนวทางในการออกแบบในระดับสูง จึงทำให้มีปัจจัยอีกหลายอย่างที่สามารถเพิ่มประสิทธิภาพให้กับวงจรได้ เช่น การออกแบบวงจรบางส่วนในระดับเกตหรือระดับทรานซิสเตอร์ และการออกแบบให้เหมาะสมกับข้อจำกัดของเทคโนโลยีที่คาดว่าจะนำไปใช้จริง เป็นต้น

รายการอ้างอิง

[1] Sparso, J. principles of asynchronous circuit design - A systems Perspective". Kluwer Academic Publishers : Kluwer Academic Publishers, 2001.

[2] Cronquist, D., Ebeling, C. Rapid – reconfigurable pipelined datapath. Proc. of International Workshop on Field Programmable Logic and Applications 1996 : 126 – 135.

[3] Esmaeildoust, M., Fazlali, M., Zakerolhosseini, A. An Efficient Algorithm for Online Placement in a Reconfigurable System. 11th International Conference on Optimization of Electrical and Electronic (OPTIM 2008) 2008 : 69-74.

[4] Goldstein, S. C., Schmit, H., Budiu, M., Cadambi, S. Moe, M., and Taylor, R. PipeRench: A Reconfigurable Architecture and Compiler. *IEEE Computer* 2000 : 70 – 77.

[5] Hauck, S. Asynchronous Design Methodologies: An Overview. Proceedings of the IEEE 1995 : 69-93.

[6] Hartenstein, R. A decade of reconfigurable computing: a visionary retrospective. Proceedings of the conference on Design, automation and test in Europe 2001 : 642-649.

[7] Kagotani, H., Schmit, H. Asynchronous PipeRench: Architecture and Performance Estimations. 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines 2003 : 121.

[8] Platzner, M., Steiger, C., Walder, H. and. Fast Online Task Placement on FPGAs: Free Space Partitioning and 2D-Hashing. Proc. of International Parallel and Distributed Processing Symposium (IPDPS'03) 2003 : 178.2

[9] Platzner, M., Walder, H. Online Scheduling for Block-partitioned Reconfigurable Devices. Proceedings of Design, Automation and Test in Europe (DATE03) 2003 : 290–295.

[10] Rabaey, J. M., Yeung, A. K. W. A reconfigurable data-driven multiprocessor architecture for rapid prototyping of high throughput DSP algorithms. Proceedings HICCS Conference 1993 : 169-178.

[11] Zhang, J., Pan, X., Shen, H. Asynchronous Reconfigurable Computing Array Design. Second International Conference on Embedded Software and Systems (ICESS'05) 2005 : 284 – 291.

[12] Advanced Processor Technologies Group. Balsa. The University of Manchester : The University of Manchester, 2006.

[13] Xilinx Inc. ISE Webpack 11.2. Xilinx Inc. : Xilinx Inc, 2009.

[14] Xilinx Inc. ModelSim XE-III 6.4b. Xilinx Inc. : Xilinx Inc., 2009.

[15] Xilinx Inc. ug380.pdf. Available from: http://www.xilinx.com/support/documentation/user_guides/ug380.pdf [2009,September 30]

[16] Xilinx Inc. Virtex6_Product_Brief.pdf. Available from: http://www.xilinx.com/publications/prod_mktg/Virtex6_Product_Brief.pdf [2009,September 30]

บรรณานุกรม

[1] Bazargan, K., Kastner, R., Sarrafzadeh, M. Fast template placement for reconfigurable computing systems. Design & Test of Computers 2000 : 68-83.

[2] Bondalapati, K., Prasanna, V. K. Reconfigurable computing systems. Proceedings of the IEEE, VOL. 90, NO.7 2002 : 1201-1217.

[3] Cadambi, S., Goldstein, S. C. Efficient place and route for pipeline reconfigurable architectures. Proceedings of the 2000 IEEE International Conference on Computer Design: VLSI in Computers & Processors 2000 : 423-429.

[4] Cheung, P.Y.K., and others. Reconfigurable computing: architectures and design methods. IEE Proceedings - Computers and Digital Techniques 2005 : 193 - 207

[5] Compton, K., Hauck, S. Totem: Custom reconfigurable array generation. Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines 2001 : 111 - 119.

[6] Compton, K., Hauck, S., Li, Z. Fast template placement for reconfigurable computing systems. Proceedings of the 2000 IEEE Symposium on Field-Programmable Custom Computing Machines 2000 : 22-36.

[7] Cronquist, D. C., and others. Architecture design of reconfigurable pipelined datapaths. Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI 1999 : 23-40.

[8] Handa, M., Vemuri, R. A fast algorithm for finding maximal empty rectangles for dynamic FPGA placement. Proceedings of the conference on Design, automation and test in Europe - Volume 1 2004 : 10744.

[9] Handa M., Vemuri R. An efficient algorithm for finding empty space for online FPGA placement. Proceedings of the 41st annual Design Automation Conference 2004 : 960-965.

[10] Hauck, S. Configuration prefetch for single context reconfigurable coprocessors. Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays 1998 : 65-74.

[11] Hauck, S., Li, Z. Configuration prefetching techniques for partial reconfigurable coprocessor with relocation and defragmentation. Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays 2002 : 187 - 195.

[12] Irwin, M. J. Reconfigurable pipeline systems. Proceedings of the 1978 annual conference 1978 : 86-92.

[13] Lai, H., Lai, Y., Yeh, C. Placement for the reconfigurable datapath architecture. Proceedings of the 2005 IEEE International Symposium on Circuits and Systems 2005 : 1875-1878.

.

ภาคผนวก

รหัสต้นฉบับตัวควบคุม

ในส่วนนี้จะเป็นรหัสต้นฉบับของตัวควบคุม ซึ่งได้จากการออกแบบที่แสดงไว้ในบทที่ 3 โดยมีรหัสต้นฉบับของส่วนประกอบภายในทั้ง 3 ส่วน ดังนี้

1) ตัวกำหนดการแบบมาก่อนให้บริการก่อน

```
-- two dashes are used for single-line comment in Balsa.
import [balsa.types.basic]
procedure buffernbit(
input i : 2642 bits;
output o : 2642 bits ) is
variable x : 2642 bits
begin
loop
i -> x;
o <- x
end
end

--main
procedure pipelinedbuffer(
parameter 2642 : cardinal;
input in0 : 2642 bits;
output out0 : 2642 bits ) is
channel a,b,c : 2642 bits
begin
buffernbit(2642,in0,a)||
buffernbit(2642,a,b)||
buffernbit(2642,b,c)||
buffernbit(2642,c,out0)
```

```
end
```

2) หน่วยความจำที่ใช้ในตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน

```
import [balsa.types.basic]
procedure arrayedbuffer(
input in0 : 2642 bits;
input rwdi : 2 bits;
input addr : 2 bits;
output out0 : 2642 bits;
output index_out : 3 bits ) is
variable index : 3 bits
variable i : 2 bits
variable buffer : array 4 of 2642 bits
begin
index := 0;
loop –loop0
  rwdi -> then
  case rwdi of
    0 then –read
      addr -> then
        out0 <- buffer[addr]
      end
   | 1 then –write
      I := (index as 2 bits);[
      loop while I > 0 then
        buffer[i] := buffer[(I – 1 as 2 bits)];
        I := (I – 1 as 2 bits)
      end;
      in0 -> buffer[0]]||
      index := (index + 1 as 3 bits)
```

```
    | 2 then –delete
        addr -> I;
        loop while I < (index – 1 as 2 bits) then
         buffer[i] := buffer[(I + 1 as 2 bits)];
         I := (I + 1 as 2 bits)
        end;
        index := (index – 1 as 3 bits)
    | 3 then –read index
        index_out <- index
  end –rwd
 end –loop0
end –begin
end –procedure
```

3) ตัวกำหนดการแบบพื้นที่น้อยที่สุดก่อน

```
import [balsa.types.basic]
import [arrayedbuffer]
procedure MIAF_scheduler(
sync interfere0,interfere1;
input taskin : 2642 bits;
output taskout_to_placer : 2642 bits;
sync place_ready;
sync place_req ) is
variable i : 4 bits
variable task_temp : 2642 bits
variable task_size : 7 bits
variable task_size_old : 7 bits
variable addr_temp : 2 bits
variable x_size, y_size : 3 bits
variable index_out_temp : 3 bits
```

```
variable quit0 : bit
variable state : 2 bits
variable tbuffer_full : bit
variable tbuffer_empty : bit
variable is_placed : bit
variable is_no_task : bit
variable is_no_place_req : bit
variable unfinished_place : bit
variable index_temp : 2 bits
channel buffer_in0 :2642 bits
channel buffer_rwdi : 2 bits
channel buffer_addr : 2 bits
channel buffer_out0 : 2642 bits
channel buffer_index_out : 3 bits
variable buffer_index_out_temp : 3 bits
shared calculate_tasksize is
begin
 i := 0||
 task_size := 0;
 loop while i <= (y_size as 4 bits) then
   task_size := (task_size + x_size + 1 as 7 bits)||
   i := (i + 1 as 4 bits)
 end
end
begin
begin
tbuffer_full := 0||
unfinished_place := 0||
tbuffer_empty := 1||
```

```
is_no_place_req := 0||
is_no_task := 0||
state := 0;
loop
  case state of
    0 then arbitrate taskin then
          tbuffer_empty := 0||
          buffer_in0 <- taskin||
          buffer_rwdi <- 1;
          buffer_rwdi <- 3||
          buffer_index_out -> buffer_index_out_temp;
          if buffer_index_out_temp = 4 then
            --print "now buffer is full";
           tbuffer_full := 1||
            state := 1
          end --if
          |interfere0 then
            -- if empty then do nothing
            if tbuffer_empty = 0 then
              if is_no_task = 1 then
                 is_no_task := 0||
                state := 1
              else
                is_no_task := 1
              end
            end
          end --arbitrate
    |1   then arbitrate place_req  then
            state := 3
```

```
      |  interfere1 then
           if (unfinished_place = 0) and (tbuffer_full = 0) then
              if is_no_place_req = 1 then
               is_no_place_req := 0||
                state := 0
              else
              is_no_place_req := 1
               end --if
            end --if
          end --arbitrate
 |2 then  sync place_ready;
          buffer_rwdi <- 3||
         buffer_index_out -> buffer_index_out_temp;
if buffer_index_out_temp /= 0 then
           state := 1||
          tbuffer_empty := 0||
          unfinished_place := 1
         else --if
          unfinished_place := 0||
          tbuffer_empty := 1||
          state := 0
        end --if
 |3 then  if buffer_index_out_temp = 1 then
buffer_rwdi <- 0||
buffer_addr <- 0||
buffer_out0 -> taskout_to_placer;
buffer_rwdi <- 2||
buffer_addr <- 0
else
```

```
 task_size_old := 65|| --force the most-right task to be collected
 quit0 := 0||
 index_temp  := (buffer_index_out_temp - 1 as 2 bits) ; --index_out - 1 = right-most task
 addr_temp := index_temp ;
 loop while quit0 /= 1 then
   buffer_addr <- index_temp ||
   buffer_rwdi <- 0||
   buffer_out0 ->task_temp;
   x_size := (#task_temp[2635..2633] as 3 bits)||
   y_size := (#task_temp[2632..2630] as 3 bits);
  -- print "x and y = ",x_size, " ", y_size;
   calculate_tasksize();
   if task_size < task_size_old then
     addr_temp := index_temp ||
     task_size_old := task_size
   end;
   -- print "index_temp is", index_temp;
   if index_temp  /= 0 then
 index_temp  := (index_temp  - 1 as 2 bits) else quit0 := 1 end
 end; --loop
  buffer_rwdi <- 0||
 buffer_addr <- addr_temp||
 buffer_out0 -> taskout_to_placer;
 buffer_rwdi <- 2||
 buffer_addr <- addr_temp||
 tbuffer_full := 0
end||state:= 2

 end --case
```

```
end --loop
end
 ||  arrayedbuffer(buffer_in0,buffer_rwdi,buffer_addr,buffer_out0,buffer_index_out)
end
```

4) ตัววาง

```
import [balsa.types.basic]
procedure placer_for_FCFS_scheduler(
sync interfere2;
input ARB64 : 6 bits;
input  task_in : 2642 bits;
output task_to_loader : 2636 bits;
output task_id : 6 bits;
output select_output_addr : 6 bits
-- task_to_loader = x_size + y_size + x_coor + y_coor + conf
 ) is
variable row0 : array 0..7 of bit
variable row1 : array 0..7 of bit
variable row2 : array 0..7 of bit
variable row3 : array 0..7 of bit
variable row4 : array 0..7 of bit
variable row5 : array 0..7 of bit
variable row6 : array 0..7 of bit
variable row7 : array 0..7 of bit
variable tbuff : 2642 bits
variable x : 3 bits
variable y : 3 bits -- for read,release,reserve, etc
variable xtemp : 3 bits
variable ytemp : 3 bits -- start x,y (x0,y0)
variable tempbit : bit -- for read,release,reserve, etc
```

```
variable xloop : 3 bits
variable yloop : 3 bits --absolute coordinate of x1,y1
variable x_size : 3 bits
variable y_size : 3 bits -- task size
variable x_output : 3 bits
variable y_output : 3 bits -- output coordinate
variable can_x : 3 bits
variable can_y : 3 bits -- selected_area
variable enough : bit -- check if an area is enough
variable occ_nb_cell : 5 bits --occupied neighbour around
variable occ_nb_cell_old : 5 bits
variable quit3,quit2,quit0,quit1 : bit
variable is_selected_area : bit
variable max_x : 3 bits
variable max_y : 3 bits -- maximal x,y for checking areas
variable buffertemp : 18 bits --used to extract taskid, size and place
variable tid_size_and_place_addr : 6 bits
variable unfinished_place : bit
variable state : 2 bits --for the state machine
variable tid_size_and_place : array 64 of 18 bits
variable is_no_arb : bit
variable array_empty : bit
variable task_id_temp : 6 bits
shared update is
begin
 case y of
  0 then row0[x] := tempbit
 |1 then row1[x] := tempbit
 |2 then row2[x] := tempbit
```

```
 |3 then row3[x] := tempbit
 |4 then row4[x] := tempbit
 |5 then row5[x] := tempbit
 |6 then row6[x] := tempbit
 |7 then row7[x] := tempbit
 end
end
shared read is
begin
 case y of
  0 then tempbit := row0[x]
 |1 then tempbit := row1[x]
 |2 then tempbit := row2[x]
 |3 then tempbit := row3[x]
 |4 then tempbit := row4[x]
 |5 then tempbit := row5[x]
 |6 then tempbit := row6[x]
 |7 then tempbit := row7[x]
 end
end
shared inc_x is
begin
 x := (x + 1 as 3 bits)
end
shared inc_y is
begin
 y := (y + 1 as 3 bits)
end
shared update_rectangle is
```

```
begin
 quit0 := 0||
  xloop := (xtemp + x_size as 3 bits)||
  yloop := (ytemp + y_size as 3 bits)||
 y := ytemp;
 loop while quit0 /= 1 then
    quit1 := 0||
    x := xtemp;
    loop while quit1 /= 1 then
    --print "before update x,y is ",x, " ", y;
      update();
      if x = xloop then
       quit1 := 1
      else
       inc_x()
      end
    end; --loop
    if y = yloop then
     quit0 := 1
    else
     inc_y()
    end
 end --loop
end --end shared
shared reserve is
begin
 tempbit := 1
end
shared release is
```

```
begin
 tempbit := 0
end
shared find_neighbour is
begin
occ_nb_cell := 0;
--xloop := (xtemp + x_size as 3 bits)||
--   yloop := (ytemp + y_size as 3 bits) ];
  --input = x_size,y_size,xtemp,ytemp, memory_array
  --check up
  if ytemp = 0 then
    occ_nb_cell := (x_size + 1 as 5 bits)
  else
    y := (ytemp - 1 as 3 bits)||
    x := xtemp||
    quit0 := 0;
    loop while quit0 /= 1 then
      read();
      if tempbit = 1 then
        occ_nb_cell := (occ_nb_cell + 1 as 5 bits)
      end|| --if
      if x = xloop then
        quit0 := 1
      else
        inc_x()
      end --if
    end --loop
  end; --if
--check left
```

```
if xtemp = 0 then
    occ_nb_cell := (occ_nb_cell + y_size + 1 as 5 bits)
  else
    x := (xtemp - 1 as 3 bits)||
    y := ytemp||
    quit0 := 0;
    loop while quit0 /= 1 then
      read();
      if tempbit = 1 then
        occ_nb_cell := (occ_nb_cell + 1 as 5 bits)
      end|| --if
      if y = yloop then
        quit0 := 1
      else
        inc_y()
      end --if
    end
end; --if
--check right
if xloop = 7 then
    occ_nb_cell := (occ_nb_cell + y_size + 1 as 5 bits)
  else
    x := (xloop + 1 as 3 bits)||
    y := ytemp||
    quit0 := 0;
    loop while quit0 /= 1 then
      read();
      if tempbit = 1 then
        occ_nb_cell := (occ_nb_cell + 1 as 5 bits)
```

```
      end|| --if      --
      if y = yloop then
        quit0 := 1
      else
        inc_y()
      end --if
    end
end ;--if
--check down
if yloop = 7 then
    occ_nb_cell := (occ_nb_cell + x_size + 1 as 5 bits)
  else
    x := xtemp||
    y := (yloop + 1 as 3 bits)||
    quit0 := 0;
    loop while quit0 /= 1 then
      read();
      if tempbit = 1 then
        occ_nb_cell := (occ_nb_cell + 1 as 5 bits)
      end|| --if
      if x = xloop then
        quit0 := 1
      else
        inc_x()
      end --if
    end --loop
end --if
end --find neighbour
shared find_candidate is
```

```
begin
  is_selected_area := 0||
occ_nb_cell_old := 0||
 max_x := (7 - x_size as 3 bits)||
  max_y := (7 - y_size as 3 bits)||
  ytemp := 0||
quit3 := 0;
-- print "max_x is ",max_x,"max_y is ", max_y;
--if debug then maxy_t <- max_y end;
loop while quit3 /= 1 then --check if ytemp reaches max_y
-- if debug then  x_size_t <- x_size||  y_size_t <- y_size end;
  xtemp := 0||
  yloop := (ytemp + y_size as 3 bits)
||
--;print "yloop=",yloop;
  quit2 := 0;
--if debug then yloop_t <- yloop end;
  loop while quit2 /= 1 then   --check if xtemp reaches max_x
    xloop := (xtemp + x_size as 3 bits)||
    enough := 1||
    quit1 := 0||
    y := ytemp;
    loop while (quit1 /= 1) and (enough = 1) then
      quit0 := 0||
      x := xtemp;
      loop while (quit0 /= 1) and (enough = 1) then
     --   print "before read x,y is ",x," ",y;
        read();
--if debug then print "x is", x end;
```

```
        if tempbit = 1 then
      --   print "not_enough when x,y =",x," ",y;
         --print "x is ",x, "when tempbit = 1";
--if debug then enough_t <- enough end;
          enough := 0
        else
         --if debug then print "x is ",x end;
          if x = xloop then
            quit0 := 1
          else
--if debug then  x_t <- x end;
            inc_x()
          end --end if
        end --end if
      end; -- loop quit0
      if (y = yloop) or (enough = 0) then
        quit1 := 1
      else
--if debug then print "y is", y end;
--if debug then  y_t <- y||  enough_t <- enough end;
        inc_y()
      end
    end; --loop quit1
    if enough = 1 then
      find_neighbour();
    --  print "occ_nb_cell is ", occ_nb_cell;
      --if debug then print "nb_cell is ",occ_nb_cell end;
      if (xtemp = 0 and ytemp = 0) or
          (occ_nb_cell > occ_nb_cell_old) then
```

```
        occ_nb_cell_old := occ_nb_cell||
        can_x := xtemp||can_y := ytemp
 --;print "can_x is ", can_x ,"can_y is ", can_y
      end||
      is_selected_area := 1
    end;
    if xtemp = max_x then
      quit2 := 1 else xtemp := (xtemp + 1 as 3 bits)
   --;print "xtemp is ",xtemp
    end
  end; --loop max_x
  if ytemp = max_y then
    quit3 := 1 else ytemp := (ytemp + 1 as 3 bits)
   --;print "ytemp is ",ytemp
  end
end; --loop max_y
  if is_selected_area = 1 then
     xtemp := can_x||ytemp := can_y||
     reserve();
     update_rectangle()
  end
end --shared
begin
row0 := {0,0,0,0,0,0,0,0}||
row1 := {0,0,0,0,0,0,0,0}||
row2 := {0,0,0,0,0,0,0,0}||
row3 := {0,0,0,0,0,0,0,0}||
row4 := {0,0,0,0,0,0,0,0}||
row5 := {0,0,0,0,0,0,0,0}||
```

```
row6 := {0,0,0,0,0,0,0,0}||
row7 := {0,0,0,0,0,0,0,0}||
array_empty := 1||
state := 0||
unfinished_place := 0||
is_no_arb := 0;
loop
case state of
 0 then arbitrate interfere2 then
        if unfinished_place = 0 then
         if is_no_arb = 1 then
            is_no_arb := 0||
            state := 1||
            unfinished_place := 1||
           task_in -> tbuff
           ;print "tbuff is ",tbuff
         else
            is_no_arb := 1
         end
        end
        | ARB64 then
             buffertemp := tid_size_and_place[ARB64];
         task_id_temp := (#buffertemp[17..12] as 6 bits)||
            x_size := (#buffertemp[11..9] as 3 bits)||
            y_size := (#buffertemp[8..6] as 3 bits)||
            xtemp := (#buffertemp[5..3] as 3 bits)||
            ytemp := (#buffertemp[2..0] as 3 bits);         task_id <- task_id_temp||
         select_output_addr <- ARB64;
            release();
```

```
            update_rectangle()||
          if unfinished_place = 1 then
            state := 1
          else
            state := 0
          end
        end --arbitrate
|1 then task_id_temp := (#tbuff[2641..2636] as 6 bits)
||
--print "tid is",task_id_temp;
x_output := (#tbuff[2629..2627] as 3 bits)
||
--print "xo is",x_output;
    y_output := (#tbuff[2626..y_2624] as 3 bits)
||
--print "yo is",y_output;
    x_size := (#tbuff[2635..2633] as 3 bits)
||
--print "x_size is", x_size;
    y_size := (#tbuff[2632..2630] as 3 bits);
--print "yrec is ",y_size;
   -- print "x_size and y_size is ",x_size," ",y_size;
    if array_empty = 1 then
   --  print "array_empty";
      can_x := 0||can_y := 0;
      xtemp := can_x||ytemp := can_y;
      reserve();
      update_rectangle()||
is_selected_area := 1 else
```

```
  -- print "find a candidate";
   find_candidate()
    end;
 if is_selected_area = 1 then
    --print "x_size is ",x_size
--     ;print "y_size is ",y_size
--     ;print "can_x is ",can_x
--     ;print "can_y is ",can_y
--     ;print "conf is ",#tbuff[2623..0];
task_to_loader <-
(#tbuff[0..0]@#can_y[0..0]@#can_y[1..1]@#can_y[2..2]@#can_x[0..0]@#can_x[1..1]@#ca
n_x[2..2]@#y_size[0..0]@#y_size[1..1]@#y_size[2..2]@#x_size[0..0]@#x_size[1..1]@#x_si
ze[2..2] as 2636 bits)||
  -- print "is_selected_area = 1 ";
   array_empty := 0||
   case can_y+y_output of
    0 then tid_size_and_place_addr := (can_x+x_output as 6 bits)
   |1 then tid_size_and_place_addr := (can_x+x_output + 8 as 6 bits)
   |2 then tid_size_and_place_addr := (can_x+x_output + 16 as 6 bits)
   |3 then tid_size_and_place_addr := (can_x+x_output + 24 as 6 bits)
   |4 then tid_size_and_place_addr := (can_x+x_output + 32 as 6 bits)
   |5 then tid_size_and_place_addr := (can_x+x_output + 40 as 6 bits)
   |6 then tid_size_and_place_addr := (can_x+x_output + 48 as 6 bits)
   |7 then tid_size_and_place_addr := (can_x+x_output + 56 as 6 bits)
  else
   continue
   end;
   --print "size_and_place_addr is ", size_and_place_addr;
  -- print "tid_size_and_place_addr is ",tid_size_and_place_addr;
```

```
  -- print "going to save into tid_size_and_place";
  -- print "task_id_temp is ",task_id_temp;
   tid_size_and_place[tid_size_and_place_addr] :=
(#task_id_temp@#x_size@#y_size@#can_x@#can_y as 18 bits)||
    unfinished_place := 0
 end||
  state := 0
 else continue
end --case
--task_size -> z;
 -- x_size := (#z[0..2] as 3 bits);
 -- y_size := (#z[3..5] as 3 bits);
end --loop
end
```

5) ตัวบรรจุ

```
import [balsa.types.basic]
procedure loader_padded(
input x_size,x_coor : 3 bits;
input y_size,y_coor : 3 bits;
--array array_task_buffer_size of input task_conf : 41 bits;
input task_conf : array 64 of 41 bits;
--array 64 of output PE_mem : 41 bits
output PE_mem0_0 : 41 bits;
output PE_mem0_1 : 41 bits;
output PE_mem0_2 : 41 bits;
output PE_mem0_3 : 41 bits;
output PE_mem0_4 : 41 bits;
output PE_mem0_5 : 41 bits;
output PE_mem0_6 : 41 bits;
```

output PE_mem0_7 : 41 bits;

output PE_mem1_0 : 41 bits;

output PE_mem1_1 : 41 bits;

output PE_mem1_2 : 41 bits;

output PE_mem1_3 : 41 bits;

output PE_mem1_4 : 41 bits;

output PE_mem1_5 : 41 bits;

output PE_mem1_6 : 41 bits;

output PE_mem1_7 : 41 bits;

output PE_mem2_0 : 41 bits;

output PE_mem2_1 : 41 bits;

output PE_mem2_2 : 41 bits;

output PE_mem2_3 : 41 bits;

output PE_mem2_4 : 41 bits;

output PE_mem2_5 : 41 bits;

output PE_mem2_6 : 41 bits;

output PE_mem2_7 : 41 bits;

output PE_mem3_0 : 41 bits;

output PE_mem3_1 : 41 bits;

output PE_mem3_2 : 41 bits;

output PE_mem3_3 : 41 bits;

output PE_mem3_4 : 41 bits;

output PE_mem3_5 : 41 bits;

output PE_mem3_6 : 41 bits;

output PE_mem3_7 : 41 bits;

output PE_mem4_0 : 41 bits;

output PE_mem4_1 : 41 bits;

output PE_mem4_2 : 41 bits;

output PE_mem4_3 : 41 bits;

```
output PE_mem4_4 : 41 bits;
output PE_mem4_5 : 41 bits;
output PE_mem4_6 : 41 bits;
output PE_mem4_7 : 41 bits;
output PE_mem5_0 : 41 bits;
output PE_mem5_1 : 41 bits;
output PE_mem5_2 : 41 bits;
output PE_mem5_3 : 41 bits;
output PE_mem5_4 : 41 bits;
output PE_mem5_5 : 41 bits;
output PE_mem5_6 : 41 bits;
output PE_mem5_7 : 41 bits;
output PE_mem6_0 : 41 bits;
output PE_mem6_1 : 41 bits;
output PE_mem6_2 : 41 bits;
output PE_mem6_3 : 41 bits;
output PE_mem6_4 : 41 bits;
output PE_mem6_5 : 41 bits;
output PE_mem6_6 : 41 bits;
output PE_mem6_7 : 41 bits;
output PE_mem7_0 : 41 bits;
output PE_mem7_1 : 41 bits;
output PE_mem7_2 : 41 bits;
output PE_mem7_3 : 41 bits;
output PE_mem7_4 : 41 bits;
output PE_mem7_5 : 41 bits;
output PE_mem7_6 : 41 bits;
output PE_mem7_7 : 41 bits
) is
```

```
variable task_buffer : array 64 of 41 bits
variable x_rec,xtemp : 3 bits
variable y_rec,ytemp : 3 bits
variable addr : 6 bits
variable xloop: 3 bits
variable yloop: 3 bits
variable x : 3 bits
variable y : 3 bits
variable temp : 41 bits
variable quit0,quit1 : bit
shared load_conf_to_PE_mem is
begin
case x of
    0 then case y of
          0 then  PE_mem0_0 <-  temp
          |1 then PE_mem0_1  <- temp
          |2 then PE_mem0_2  <- temp
          |3 then PE_mem0_3  <- temp
          |4 then PE_mem0_4  <- temp
          |5 then PE_mem0_5  <- temp
          |6 then PE_mem0_6  <- temp
          |7 then PE_mem0_7  <- temp
          end
   |1  then case y of
          0 then  PE_mem1_0 <- temp
          |1 then PE_mem1_1 <- temp
          |2 then PE_mem1_2  <- temp
          |3 then PE_mem1_3  <- temp
          |4 then PE_mem1_4  <- temp
```

```
        |5 then PE_mem1_5  <- temp
        |6 then PE_mem1_6  <- temp
        |7 then PE_mem1_7  <- temp
        end
|2 then  case y of
        0 then  PE_mem2_0  <- temp
        |1 then PE_mem2_1  <- temp
        |2 then PE_mem2_2  <- temp
        |3 then PE_mem2_3  <- temp
        |4 then PE_mem2_4  <- temp
        |5 then PE_mem2_5  <- temp
        |6 then PE_mem2_6  <- temp
        |7 then PE_mem2_7  <- temp
        end
|3 then case y of
        0 then  PE_mem3_0  <- temp
        |1 then PE_mem3_1  <- temp
        |2 then PE_mem3_2  <- temp
        |3 then PE_mem3_3  <- temp
        |4 then PE_mem3_4  <- temp
        |5 then PE_mem3_5  <- temp
        |6 then PE_mem3_6  <- temp
        |7 then PE_mem3_7  <- temp
        end
|4 then  case y of
        0 then  PE_mem4_0  <- temp
        |1 then PE_mem4_1  <- temp
        |2 then PE_mem4_2  <- temp
        |3 then PE_mem4_3  <- temp
```

```
        |4 then PE_mem4_4  <- temp
        |5 then PE_mem4_5  <- temp
        |6 then PE_mem4_6  <- temp
        |7 then PE_mem4_7  <- temp
        end
|5 then case y of
        0 then  PE_mem5_0  <- temp
        |1 then PE_mem5_1  <- temp
        |2 then PE_mem5_2  <- temp
        |3 then PE_mem5_3  <- temp
        |4 then PE_mem5_4  <- temp
        |5 then PE_mem5_5  <- temp
        |6 then PE_mem5_6  <- temp
        |7 then PE_mem5_7  <- temp
        end
|6 then case y of
        0 then  PE_mem6_0  <- temp
        |1 then PE_mem6_1  <- temp
        |2 then PE_mem6_2  <- temp
        |3 then PE_mem6_3  <- temp
        |4 then PE_mem6_4  <- temp
        |5 then PE_mem6_5  <- temp
        |6 then PE_mem6_6  <- temp
        |7 then PE_mem6_7  <- temp
        end
|7 then case y of
        0 then  PE_mem7_0  <- temp
        |1 then PE_mem7_1  <- temp
        |2 then PE_mem7_2  <- temp
```

```
            |3 then PE_mem7_3  <- temp
            |4 then PE_mem7_4  <- temp
            |5 then PE_mem7_5  <- temp
            |6 then PE_mem7_6  <- temp
            |7 then PE_mem7_7  <- temp
            end
      end
end –shared
begin
addr := 0||
task_conf -> task_buffer||
x_size -> x_rec||
y_size -> y_rec||
x_coor -> xtemp||
y_coor -> ytemp;
xloop := (xtemp + x_rec as 3 bits)||
yloop := (ytemp + y_rec as 3 bits);
print xloop, " ", yloop;
y := (ytemp as 3 bits)||
quit1 := 0;
print "y is ", y;
loop while (quit1 /= 1) then  --y
x := (xtemp as 3 bits);
print "x is ", x;
quit0 := 0;
loop while (quit0 /= 1) then  --x
    temp := task_buffer[addr];
    load_conf_to_PE_mem();
addr := (addr + 1 as 6 bits)||
```

```
    if x /= xloop then
      x := (x + 1 as 3 bits)
     else quit0 := 1 end
end; --loopx
  if y /= yloop then
      y := (y + 1 as 3 bits)
  else  quit1 := 1 end
end --loopy
end –loader
```

ประวัติผู้เขียนวิทยานิพนธ์

นายปกรณ์ ทู้ไพเราะ เกิดเมื่อวันที่ 4 กรกฎาคม พ.ศ. 2526 ที่โรงพยาบาลศิริราช จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาระดับประถมศึกษาที่ โรงเรียนวัดอมรินทราราม จังหวัดกรุงเทพมหานคร และได้เข้าศึกษาต่อระดับมัธยมศึกษาตอนต้นที่ โรงเรียนนวมินทราชินูทิศ สตรีวิทยา พุทธมณฑล หลังจากนั้นได้ศึกษาต่อในระดับมัธยมศึกษาตอนปลายที่ โรงเรียนมหิดลวิทยานุสรณ์ ตำบลศาลายา อำเภอพุทธมณฑล และได้เข้าศึกษาต่อในระดับปริญญาตรี สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ ที่มหาวิทยาลัยสยาม จังหวัดกรุงเทพมหานคร หลังจากนั้นปี พ.ศ. 2548 ได้เข้าศึกษาในระดับปริญญาโท สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย มีผลงานทางวิชาการที่เผยแพร่ คือ “A design of Asynchronous Double Grain Reconfigurable Computing Array”

ผู้ทำวิจัยมีความสนใจทางด้านคอมพิวเตอร์ตั้งแต่ได้ศึกษาประมาณชั้นมัธยมศึกษาปีที่ 1 โดยได้เริ่มสนใจจากเล่นเกมส์คอมพิวเตอร์ตั้งแต่ยุคที่ระบบปฏิบัติการดอส (DOS: Disk Operating System) ได้รับความนิยมอย่างแพร่หลาย เริ่มศึกษาการใช้คำสั่งพื้นฐานและโปรแกรมอรรถประโยชน์ (Utility) ต่างๆ ภายในระบบปฏิบัติการดอสและศึกษาการเขียนโปรแกรมคอมพิวเตอร์ด้วยตนเอง โดยมีภาษาปาสคาล (Pascal) เป็นภาษาแรก เมื่อผู้ทำวิจัยศึกษาในระดับมัธยมศึกษาตอนต้น ได้มีความสนใจทางด้านคอมพิวเตอร์มาเป็นลำดับ ได้ทำโครงงานวิทยาศาสตร์ โดยนำวงจรอิเล็กทรอนิกส์มาเชื่อมต่อกับคอมพิวเตอร์เพื่อเปิดปิดอุปกรณ์ไฟฟ้าในระดับปริญญาตรีได้ทำโครงงานวิศวกรรมพิวเตอร์ที่เกี่ยวข้องกับการเข้ารหัสข้อมูล โดยได้ออกแบบวงจรเข้ารหัสเออีเอส (AES: Advanced Encryption Standard) ด้วยภาษาเวริล็อก

จากประสบการณ์ที่ผู้ทำวิจัยได้ใช้คอมพิวเตอร์ส่วนบุคคล (PC: Personal Computer) มาเป็นเวลานานหลายปี ผู้วิจัยมีความคิดเห็นว่า ระบบคอมพิวเตอร์มีการพัฒนาที่เร็วมาก ทั้งทางด้านความเร็วและความหลากหลาย สามารถนำมาใช้ประโยชน์ต่อการดำรงชีวิตได้หลายทิศทาง เช่น การพยากรณ์อากาศ การจำลองทางด้านวิทยาศาสตร์ และความเป็นจริงเสมือน (VR: Virtual Reality) เป็นต้น ผลกระทบอย่างหนึ่งที่ไม่สามารถหลีกเลี่ยงได้ คือ ขยะอิเล็กทรอนิกส์ ซึ่งมีผลเสียต่อสิ่งแวดล้อมและการใช้พลังงาน ผู้ทำวิจัยจึงมีความประสงค์ในการแนะนำให้ผู้ที่เกี่ยวข้องกับการใช้คอมพิวเตอร์และอุปกรณ์อิเล็กทรอนิกส์ต่างๆ มีความตระหนักถึงการอนุรักษ์พลังงานและทรัพยากรธรรมชาติเพื่อให้ชนรุ่นหลังได้ใช้สืบไป