การเพิ่มประสิทธิภาพ context modeling แบบ pass-pipelined ของ JPEG2000 โดยใช้เอฟพีจีเอ

Efficient Pass-Pipelined Architecture for Context Modeling of JPEG2000 using FPGA

คมกริช มาเที่ยง

KHOMKRIS MATHIANG

วิทยานิพนธ์นี้สำหรับการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2552

KMITL-2009-EN-M-070-162

การเพิ่มประสิทธิภาพ context modeling แบบ pass-pipelined ของ JPEG2000 โดยใช้เอฟพีจีเอ

Efficient Pass–Pipelined Architecture for Context Modeling of JPEG2000 using FPGA

คมกริช มาเที่ยง

KHOMKRIS MATHIANG

วิทยานิพนธ์นี้สำหรับการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2552

KMITL-2009-EN-M-070-162

Efficient Pass–Pipelined Architecture for Context Modeling of JPEG2000 using FPGA

KHOMKRIS MATHIANG

A THESIS SUBMITTED IN FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN COMPUTER ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2009

KMITL-2009-EN-M-070-162

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การเพิ่มประสิทธิภาพ context modeling แบบ pass-pipelined ของ JPEG2000โดยใช้เอฟพีจีเอ

Thesis Title Efficient Pass-Pipelined Architecture for Context Modeling of JPEG2000 using FPGA

นักศึกษา นายคมกริช มาเที่ยง

รหัสประจำตัว 49060651

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์ ผศ.ดร.อรฉัตร จิตต์โสภักตร์

หมายเลขวิทยานิพนธ์ KMITL-2009-EN-M-070-162

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ผศ.ดร.สุรินทร์	กิตติธรกุล	
ดร.ปกรณ์	วัฒนจตุรพร	
รศ.ดร.สมศักดิ์	มิตะถา	
ดร.ศุภกานต์	พิมลธเรศ	
ผศ.ดร.อรฉัตร	จิตต์โสภักตร์	

วัน / เดือน/ ปี ที่สอบ วันพฤหัสบดีที่ 8 ตุลาคม พ.ศ. 2552 เวลา 11.30-13.30 น.

สถานที่สอบ ณ อาคาร A ชั้น 3 ห้องประชุม 2

คณะวิศวกรรมศาสตร์ รับรองแล้ว

(รองศาสตราจารย์ ดร.กอบชัย เดชหาญ)

คณบดี คณะวิศวกรรมศาสตร์

วันที่ 8 ตุลาคม พ.ศ. 2552

หัวข้อวิทยานิพนธ์	การเพิ่มประสิทธิภาพ context modeling แบบ pass-pipelined ของ JPEG2000 โดยใช้เอฟพีจีเอ
นักศึกษา	นายคมกริช มาเที่ยง
รหัสประจำตัว	49060651
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2552
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ.ดร.อรฉัตร จิตต์โสภักตร์

บทคัดย่อ

ในปัจจุบันข้อมูลข่าวสารมีความสำคัญและมีส่วนเกี่ยวข้องกับการดำเนินชีวิตของมนุษย์เป็นอันมาก เพื่อให้การสื่อสารข้อมูลข่าวสารเป็นไปด้วยความรวดเร็วทำให้มาตรฐานการบีบอัดสัญญาณรูปภาพของงานวิจัยส่วนใหญ่มุ่งเน้นไปที่กระบวนการบีบอัดสัญญาณรูปภาพและการเพิ่มประสิทธิภาพของการบีบอัดสัญญาณรูปภาพ สำหรับงานวิจัยที่ได้นำเสนอนี้มุ่งเน้นไปที่การเพิ่มประสิทธิภาพของการบีบอัดสัญญาณรูปภาพในส่วนของการทำงาน Context Modeling ของการบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 ด้วยฮาร์ดแวร์โดยใช้เอฟพีจีเอ (FPGA) เป็นอุปกรณ์ต้นแบบในการออกแบบและทดสอบประสิทธิภาพของระบบ

ระบบโครงสร้างที่นำเสนอเป็นโครงสร้างการทำงานแบบ Pass-pipelined ที่แบ่งการทำงานออกเป็นส่วนย่อย ซึ่งช่วยลดเวลาในการประมวลผลและช่วยลดส่วนที่หน่วงประสิทธิภาพการทำงาน (Critical Path Delay) เนื่องจากการออกแบบระบบใช้หน่วยความจำแบบคู่ (Dual Memories) และการเลือกข้อมูลทำให้การเข้าถึงข้อมูลในหน่วยความจำเป็นไปอย่างต่อเนื่อง ระบบโครงสร้างที่นำเสนอสามารถประมวลผลด้วยความเร็วที่มากกว่า 100 เมกะเฮิรตซ์ และสามารถสร้างคู่อันดับสัญลักษณ์ (Context-decision Pairs) ได้ถึง 22 คู่อันดับสัญลักษณ์ใน 1 สัญญาณนาฬิกา

Thesis Title	Efficient Pass-Pipelined Architecture for Context Modeling of JPEG2000 using FPGA
Student	Mr.Khomkris Mathiang
Student ID	49060651
Degree	Master of Engineering
Program	Computer Engineering
Year	2009
Thesis Advisor	Asst. Prof. Dr.Orachat Chitsobhuk

ABSTRACT

Presently, information has become increasingly significant and turned to be a part of human's lift. In order to transmit information as fast as possible, it has to be compressed. Therefore, the compression standard is needed to help reduce amount of information to be transferred. To accelerate to compression process, several researchers have targeted to improve the performance of the compression process. JPEG2000 is one of the famous image compression standards, which is used to compress images for effective transmission and storage. This thesis introduces an alternative architecture of context modeling process, which is one of the most time consuming processes in JPEG2000. The proposed architecture is designed and implemented on Field Programmable Gate Array (FPGA) to increase the coding performance of the context modeling process.

In this thesis, a design of pass-pipelined architecture for JPEG2000 context modeling implemented on FPGA is proposed. The architecture is separated into pipelined stages. As a result, the processing time and critical path delay can be reduced while the multiple symbol context pairs are allowed to be generated simultaneously. Moreover, dual memories and data multiplexer are employed in order to accelerate the memory access. The proposed pass-pipelined architecture can process with the speed greater than 100 MHz and can generate up to 22 context-decision pairs in one clock cycle.

กิตติกรรมประกาศ

ในชีวิตของคนๆ หนึ่งที่ได้เกิดมาบนโลกใบนี้ ต่างก็มีความฝัน จุดมุ่งหมายปลายทางในชั่วขณะเวลาหนึ่งที่ต่างออกไป บางคนก็มีความอยากจะเป็นนักร้อง บางคนก็อยากเป็นบุคคลที่ได้รับการนับหน้าถือตา ไม่ต่างอะไรจากชีวิตของข้าพเจ้าที่มีสถานะภาพของความเป็นมนุษย์ทั่วไปที่มีความฝันและความสำเร็จในการดำเนินชีวิต ณ. ขณะเวลาก่อนหน้านี้ ซึ่งข้าพเจ้าได้กำลังศึกษาและทำการวิจัย ข้าพเจ้ามีความหวังว่า "เราจะต้องสำเร็จการศึกษาปริญญาโทให้ได้ภายในระยะเวลา 2 ปี" แต่ในขณะเวลานี้ ล่วงเลยมาจนจะครบเวลา 3 ปีแล้ว กลับพบว่าข้าพเจ้ายังคงใช้ชีวิตวนเวียนในรั้วของสถาบันไม่ต่างจากเมื่อตอนเริ่มการศึกษา คำถามมากมายจากในใจของข้าพเจ้า จากเพื่อน จากญาติมิตร และบุคคลรอบข้างที่ข้าพเจ้ารู้จัก ต่างถาโถมเข้ามา เมื่อไหร่จะสำเร็จการศึกษา เมื่อไหร่จะจบ ด้วยคำพูดเหล่านี้เหมือนจะกลายมาเป็นแรงผลักดันให้ข้าพเจ้ามีแรงสู้แต่กลับกลายเป็นว่าในบางเวลาแรงกดดันมหาศาลเหล่านั้นกลับเป็นสิ่งที่ทำให้ข้าพเจ้ารู้สึกเหนื่อย

แต่ด้วยความเมตตา ความอารีของ ผศ. ดร. อรฉัตร จิตต์โสภักตร์ ซึ่งเป็นอาจารย์ที่ปรึกษาของข้าพเจ้า ซึ่งคอยเคี่ยวเข็ญให้ข้าพเจ้ามีแรงในการศึกษาให้สำเร็จลุล่วงตลอดจนคอยชี้แนะแนวทางให้ข้าพเจ้าหลุดออกมาจากความมืดของความไม่รู้ต่างๆ ดังคำกล่าวของเดนเซล วอชิงตัน (Denzel Washington) ที่ว่า "I say luck is when an opportunity comes along, and you're prepared for it" สามารถแปลเป็นไทยได้ว่า "ผมว่าโชคดี คือ เมื่อโอกาสมาถึงและคุณพร้อมสำหรับมัน" จะเห็นได้ว่าในเวลานี้ข้าพเจ้าได้รับความโชคดีนั่นแล้ว และข้าพเจ้าพร้อมที่จะรับความโชคดีจากคำชี้แนะของอาจารย์ที่ปรึกษาของข้าพเจ้าแล้ว ข้าพเจ้าคงไม่อาจที่จะปฏิเสธได้เลยที่จะกล่าวคำว่า "ขอขอบพระคุณเป็นอย่างสูง" มอบแด่อาจารย์ที่ปรึกษาของข้าพเจ้า

อีกทั้งยังขอขอบพระคุณต่อความใจดี ความเสียสละ และสิ่งต่างๆ ที่ บิดามารดาของข้าพเจ้าที่ได้ให้ข้าพเจ้าได้เกิดขึ้นมาในโลกนี้ คอยเลี้ยงดู คอยอบรมสั่งสอนข้าพเจ้าให้เป็นคนดีในสังคม ไม่ว่าจะมีคุณความดีใดๆ ที่วิทยานิพนธ์เล่มนี้ได้รับ ไม่ว่าจะเล็กหรือใหญ่ มากน้อยสักเพียงใด ข้าพเจ้าขอมอบแด่บิดามารดาและอาจารย์ที่ปรึกษาของข้าพเจ้าทั้งหมด หากมีความผิดพลาดใดๆ ที่ได้เกิดขึ้นในวิทยานิพนธ์เล่มนี้ได้สร้างขึ้นมา ข้าพเจ้าขอน้อมรับความผิดพลาดนั้นๆ แต่เพียงผู้เดียว

"คนเราเกิดมาและมีชีวิตอยู่รอด เพื่อรอความตายในวินาทีถัดไป" สำหรับข้าพเจ้าคงไม่พ้นจากข้อความในข้างต้น แต่วิทยานิพนธ์เล่มนี้ที่มีองค์ความรู้ถึงแม้จะไม่มากก็ยังคงอยู่ไปยาวนานกว่าชีวิตของข้าพเจ้า ดังนั้นข้าพเจ้าขอขอบคุณต่อผู้ที่ได้ศึกษาหาความรู้จากวิทยานิพนธ์เล่มนี้และนำไปพัฒนา สร้างสรรค์สิ่งที่ดีงามต่อโลกที่เรายังคงใช้ชีวิตอยู่นี้ต่อไป

คมกริช มาเที่ยง (10 กันยายน 2552)

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการสื่อสารข้อมูลมีความสำคัญต่อการดำเนินชีวิตเป็นอย่างยิ่งและมีความเกี่ยวข้องกับมนุษย์เป็นอันมาก เพื่อให้การสื่อสารข้อมูลข่าวสารเป็นไปด้วยความรวดเร็วทำให้มีการพัฒนามาตรฐานต่างๆ ขึ้นมาเป็นจำนวนมาก โดยยังผลให้สามารถส่งถ่ายข้อมูลที่มีจำนวนมากๆ ได้ภายในระยะเวลาอันสั้น ทั้งข้อมูลในรูปแบบของ เสียง ข้อความอักขระ ยังรวมถึงข้อมูลรูปภาพ เป็นต้น

ด้วยเหตุผลดังที่ได้กล่าวข้างต้นทำให้มาตรฐานการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000 ได้ถือกำเนิดขึ้นมา ด้วยอัลกอริธึมที่มีความซับซ้อนจึงทำให้การบีบอัดสัญญาณข้อมูลที่เป็นรูปภาพสามารถบีบอัดข้อมูลให้มีขนาดเล็กลงได้เป็นอย่างมาก แต่ความเร็วในการประมวลผลข้อมูลก็จะสูงตามไปด้วย เพราะความสลับซับซ้อนของอัลกอริธึมนั่นเอง โดยเฉพาะอย่างยิ่งในส่วนของการเข้ารหัสระนาบบิต (Bit Plane Coding) ที่เป็นส่วนหนึ่งของอัลกอริธึมที่ใช้ในการบีบอัดสัญญาณข้อมูลที่เป็นรูปภาพตามมาตรฐาน JPEG2000 ดังนั้นในงานวิจัยนี้จึงนำเสนอการวิจัยและพัฒนาการประมวลผลของการเข้ารหัสระนาบบิตโดยการใช้ FPGA (Field Programmable Gate Array) เพื่อช่วยเพิ่มประสิทธิภาพของการบีบอัดข้อมูลรูปภาพให้มีความเร็วมากยิ่งขึ้น

สำหรับในงานวิจัยที่นำเสนอจะเลือกการทำงานแบบส่งต่อ (Pipeline) มาช่วยในการเพิ่มประสิทธิภาพ รวมทั้งการออกแบบระบบโครงสร้างในการเข้ารหัสระนาบบิตทางฮาร์ดแวร์เข้าร่วมด้วย สาเหตุที่เลือกการทำงานแบบส่งต่อมาใช้ในการออกแบบเนื่องจากสามารถทำให้ระบบมีความเร็วในการประมวลผลที่สูงและผลของจำนวนผลลัพธ์ที่ได้ต่อเวลามากยิ่งขึ้นด้วย

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

1.2.1 ศึกษากระบวนการในการเข้ารหัสระนาบบิตที่เป็นไปตามมาตรฐานการบีบอัดสัญญาณข้อมูลที่เป็นรูปภาพตามมาตรฐาน JPEG2000

1.2.2 ศึกษาการออกแบบวงจรรวมโดยใช้ FPGA

1.2.3 ศึกษาและปรับปรุงวิธีในการเพิ่มประสิทธิภาพการประมวลผลของการเข้ารหัสระนาบบิต โดยอาศัยสมมติฐาน ความเข้าใจ และทฤษฎีในการเพิ่มประสิทธิภาพ

1.2.4 สามารถประเมินประสิทธิภาพ รวมทั้งสรุปวิเคราะห์ผลลัพธ์ที่ได้จากงานวิจัยที่นำเสนอ

1.2.5 เป็นแหล่งอ้างอิงและองค์ความรู้สำหรับผู้ที่สนใจในงานวิจัยทางด้านนี้ต่อไป

1.3 สมมติฐานของการศึกษา

การประมวลผลการเข้ารหัสระนาบบิตที่เป็นไปตามมาตรฐานการบีบอัดสัญญาณที่เป็นรูปภาพตามมาตรฐาน JPEG2000 ที่ใช้ซอฟแวร์ประมวลผลจะมีความล่าช้า เพราะการทำงานของ CPU จะเป็นแบบลำดับขั้น (Sequential) หากนำกระบวนการในการเข้ารหัสระนาบบิตดังกล่าวมาออกแบบวงจรดิจิตอลโดยใช้ FPGA เพื่อเพิ่มประสิทธิภาพในการประมวลผลการเข้ารหัสระนาบบิต ซึ่งน่าจะใช้เวลาในการประมวลผลน้อยลง เนื่องจาก FPGA สามารถทำงานหลายอย่างพร้อมกันได้ (Parallel Processing) อีกทั้งยังออกแบบการทำงานของการเข้ารหัสระนาบบิตโดยใช้ความสามารถในการทำงานแบบส่งต่อ โดยแบ่งการทำงานของการเข้ารหัสระนาบบิตให้เป็นส่วนย่อยๆ คาดว่าจะสามารถเพิ่มความเร็วในการประมวลผล รวมถึงปริมาณของจำนวนผลลัพธ์ที่ได้ต่อหน่วยเวลามีค่าที่สูง ดังนั้นในงานวิจัยที่นำเสนอจึงทำการศึกษาและออกแบบการเข้ารหัสระนาบบิตโดยการแบ่งส่วนการทำงานออกเป็นส่วนย่อยๆ และให้ระบบทำงานแบบส่งต่อกันไปบน FPGA

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

เพื่อให้สามารถทำการออกแบบการประมวลผลการเข้ารหัสระนาบบิตที่เป็นไปตามมาตรฐานการบีบอัดสัญญาณที่เป็นรูปภาพตามมาตรฐาน JPEG2000 นั้นจะอาศัยหลักการและทฤษฎีต่างๆที่เกี่ยวข้องดังต่อไปนี้

- หลักการการทำงานแบบส่งต่อ
- หลักการการเข้ารหัสระนาบบิตที่เป็นไปตามมาตรฐานการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000
- ทฤษฎีการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000
- หลักการออกแบบระบบโครงสร้างทางฮาร์ดแวร์บน FPGA

1.5 ขอบเขตการวิจัย

ในงานวิจัยที่นำเสนอนี้จะทำการศึกษาและดำเนินการวิจัยเพื่อออกแบบระบบโครงสร้างที่เพิ่มประสิทธิภาพในการประมวลผลการเข้ารหัสระนาบบิตที่เป็นไปตามมาตรฐานการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000 โดยระบบโครงสร้างที่นำเสนอจะถูกออกแบบให้ทำงานบน FPGA ซึ่งระบบโครงสร้างที่ออกแบบจะแบ่งการทำงานของการเข้ารหัสระนาบบิตออกเป็นส่วนย่อยๆ ทำให้สามารถประมวลผลแบบส่งต่อกันไปได้ และทำการเปรียบเทียบผลการ

ทำงานกับงานวิจัยอื่น ระบบโครงสร้างที่ออกแบบจะใช้วิธีการประมวลผลของพาส (Pass) ที่อยู่ในส่วนของการเข้ารหัสระนาบบิตที่ทำงานแบบส่งต่อกันไป เพื่อให้การประมวลผลการเข้ารหัสระนาบบิตมีความต่อเนื่องของข้อมูล ในระบบโครงสร้างได้มีการนำการทำงานโดยใช้หน่วยความจำคู่มาเพิ่มประสิทธิภาพของการนำส่งข้อมูลให้ระบบการเข้ารหัสระนาบบิต ในการตรวจสอบประสิทธิภาพของการประมวลผลในแต่ละระบบโครงสร้างจะตรวจสอบความเร็วที่แต่ละระบบโครงสร้างสามารถทำงานได้ ทรัพยากรที่ใช้ในการออกแบบบน FPGA ผลตอบสนองต่อการสั่งการ และปริมาณผลลัพธ์ต่อเวลา

1.6 ขั้นตอนของการศึกษา

กำหนดวัตถุประสงค์และขอบเขตของงานวิจัย เพื่อนำเสนอและศึกษางานวิจัยในแต่ละหัวข้อมากน้อยเพียงใด

ศึกษาค้นคว้าทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการเข้ารหัสระนาบบิตที่ถูกออกแบบทางฮาร์ดแวร์ แล้วนำมาวิเคราะห์ ข้อดี ข้อเสีย และประเด็นที่น่าสนใจของงานวิจัยอื่นๆ ที่สามารถนำมาปรับปรุงและประยุกต์ใช้ เพื่อก่อให้เกิดประโยชน์ต่องานวิจัยที่นำเสนอ

ตั้งสมมติฐานของการศึกษาและกำหนดแนวความคิดของงานวิจัย โดยมีการอ้างอิงทฤษฎีหรือหลักการที่เกี่ยวข้องเพื่อที่จะบรรลุตามวัตถุประสงค์ที่ได้กำหนดไว้

เตรียมฐานข้อมูลรูปภาพเพื่อนำมาใช้ทดสอบงานวิจัย โดยฐานข้อมูลดังกล่าวประกอบไปด้วยข้อมูลรูปภาพที่มีความหลากหลาย เพื่อสามารถนำมาใช้ตรวจสอบประสิทธิภาพและความถูกต้องของระบบโครงสร้างที่นำเสนอได้อย่างเหมาะสม

ทำการออกแบบระบบโครงสร้างการเข้ารหัสระนาบบิตที่มาจากงานวิจัยอื่นและระบบโครงสร้างที่นำเสนอ โดยในการพัฒนาจะใช้โปรแกรมจำลองการทำงานทางฮาร์ดแวร์มาใช้ในการทดสอบและตรวจสอบประสิทธิภาพของระบบโครงสร้างทั้งสองระบบ เพื่อนำมาวิเคราะห์ปรับปรุงงานวิจัยต่อไป

นำผลลัพธ์ที่ได้จากการทดสอบมาวิเคราะห์และประเมินงานวิจัย ทั้งในแง่ของความถูกต้องและประสิทธิภาพ โดยเปรียบเทียบกับงานวิจัยอื่นๆ และสรุปผลเพื่อนำเสนอผลงานวิจัย

1.7 เครื่องมือและอุปกรณ์ที่ใช้ในงานวิจัย

เครื่องมือและอุปกรณ์ที่ใช้ในงานวิจัย

- เครื่องคอมพิวเตอร์ส่วนบุคคลใช้หน่วยประมวลผลกลาง AMD Athlon™ XP 2500+ หน่วยความจำ 1.25 GBytes จำนวน 1 เครื่อง
- ระบบปฏิบัติการวินโดวส์ XP

- Xilinx ISE Webpack software 8.2i พร้อมทั้งชุดปรับปรุง service pack 3
- โปรแกรมจำลองการทำงานทางฮาร์ดแวร์ ModelSim

1.8 โครงสร้างของวิทยานิพนธ์

วิทยานิพนธ์ฉบับนี้แบ่งออกเป็น 6 บทแต่ละบทมีเนื้อหาดังต่อไปนี้

บทที่ 1 กล่าวถึง ความเป็นมาและความสำคัญของปัญหา จุดมุ่งหมายและวัตถุประสงค์ของการศึกษา สมมติฐานของการศึกษา รวมทั้งทฤษฎีหรือแนวคิดที่ใช้ในการศึกษา ขอบเขตของการศึกษา และขั้นตอนของการศึกษา

บทที่ 2 กล่าวถึง งานวิจัยที่เกี่ยวข้องที่ออกแบบทางฮาร์ดแวร์ และงานวิจัยอื่นๆ ที่เกี่ยวข้อง

บทที่ 3 กล่าวถึง ความรู้พื้นฐานที่เกี่ยวข้องกับงานวิจันที่นำเสนอ ประกอบไปด้วย การบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000 การเข้ารหัสระนาบบิต และการทำงานแบบส่งต่อ

บทที่ 4 กล่าวถึง การออกแบบระบบโครงสร้างที่นำเสนอ ที่แยกอธิบายย่อยออกเป็นส่วนๆ

บทที่ 5 กล่าวถึง การทดลองและผลการทดลอง

บทที่ 6 กล่าวถึง การวิเคราะห์ ข้อเสนอแนะ และแนวทางในการพัฒนาต่อไปในอนาคต

บทที่ 2
งานวิจัยที่เกี่ยวข้อง

2.1 บทนำ

ปัจจุบันการสื่อสารมีความสำคัญต่อการดำเนินชีวิตมากยิ่งขึ้น โดยเฉพาะข้อมูลข่าวสารทั้งข้อมูลที่เป็นตัวอักขระ ข้อมูลที่เป็นเสียง และรูปภาพ ข้อมูลข่าวสารดังกล่าวนับวันยิ่งมีขนาดที่ใหญ่มากขึ้น ทำให้ระบบการสื่อสารข้อมูลต้องพัฒนาให้มีความเร็วมากขึ้นตามไปด้วย แต่ในอีกแนวทางหนึ่งที่ทำให้การสื่อสารข้อมูลเป็นไปด้วยความรวดเร็วมากยิ่งขึ้นนั่นคือการลดขนาดของข้อมูลที่ต้องการสื่อสารให้มีขนาดที่เล็กลง หนึ่งในอัลกอริธึมที่ใช้ในการลดขนาดหรือบีบอัดข้อมูลรูปภาพที่งานวิจัยนี้ได้ทำการศึกษา คือ การบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000 [1] โดยในงานวิจัยที่นำเสนอจะมุ่งเน้นไปที่การเพิ่มประสิทธิภาพในการทำงานของระบบการเข้ารหัสระนาบบิตอันเป็นส่วนหนึ่งของอัลกอริธึมที่ใช้ในการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000

การออกแบบระบบโครงสร้างที่งานวิจัยที่นำเสนอนี้จะถูกออกแบบให้สามารถทำงานอยู่บนชิพวงจรรวม FPGA เพื่อใช้ความสามารถของอุปกรณ์ที่เป็นฮาร์ดแวร์ในการเพิ่มประสิทธิภาพของการประมวลผลการเข้ารหัส เนื่องจากว่าการประมวลผลของอุปกรณ์ที่เป็น FPGA มีความสามารถในการประมวลผลแบบขนานทำให้สามารถประมวลผลงานที่ไม่ขึ้นตรงต่อกันไปพร้อมๆ กันได้

2.2 ปัญหาในการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000

การบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000 [1] เป็นอัลกอริธึมที่ได้ถูกนำเสนอขึ้นมาเป็นมาตรฐานที่ใช้ในการบีบอัดสัญญาณข้อมูลรูปภาพ เพื่อให้สามารถบีบอัดข้อมูลให้มีขนาดที่เล็กลงมากกว่า JPEG เดิม โดยในส่วนของอัลกอริธึมจะแบ่งการทำงานออกเป็นหลายๆ ส่วน ซึ่งในแต่ละส่วนจะมีความซับซ้อนในการทำงานที่แตกต่างกันไป

จากงานวิจัยของ C Lian, *et.al.* [2] ได้ทำการวิเคราะห์กระบวนการในการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000 ในส่วนของ EBCOT (Embedded Block Coding with Optimized Truncation) จะพบว่าเวลาที่ใช้ในการบีบอัดสัญญาณข้อมูลรูปภาพจะเป็นไปตามตารางที่ 2.1

ตารางที่ 2.1 แสดงเวลาในการประมวลผลในแต่ละส่วนของ JPEG2000 จากงานวิจัย [2]

ส่วนย่อยๆที่ประมวลผลในการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000	เปอร์เซ็นต์เวลาในการประมวลผลของแต่ละส่วน (%)			
	รูปภาพที่เป็นภาพขาวดำ		รูปภาพที่เป็นภาพสี	
	การเข้ารหัสที่ไม่ยอมให้มีการสูญเสียข้อมูล	การเข้ารหัสที่ยอมให้มีการสูญเสียข้อมูล	การเข้ารหัสที่ไม่ยอมให้มีการสูญเสียข้อมูล	การเข้ารหัสที่ยอมให้มีการสูญเสียข้อมูล
ส่วนในการแปลงสี (Color Transform)	-	-	0.91	14.12
การแปลงสัมประสิทธิ์ทางเวฟเลท (DWT)	10.81	26.38	11.90	23.97
การลดระดับความละเอียดของข้อมูล (Quantization)	-	6.42	-	5.04
EBCOT Tier 1	71.63	52.26	69.29	43.85
Pass 1	14.89	14.82	13.90	12.39
Pass 2	10.85	7.00	10.94	5.63
Pass 3	26.14	16.09	25.12	13.77
Arithmetic Encoding	19.75	14.35	19.33	12.06
EBCOT Tier 2	17.56	14.95	17.90	13.01

จากตารางที่ 2.1 แสดงให้เห็นได้ว่า เวลาที่ใช้ในการประมวลผลส่วนใหญ่กว่าครึ่งในการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000 จะอยู่ในส่วนของการทำงาน EBCOT Tier 1 โดยเฉพาะอย่างยิ่งในช่วงของการเข้ารหัสระนาบบิต (Pass 1: SPP, Pass 2: MRP และ Pass 3: CUP) เมื่อเปรียบเทียบกับส่วนการทำงานย่อยอื่นแล้วจะใช้เวลามากกว่าส่วนย่อยอื่นๆ ทุกส่วนด้วย และในงานวิจัย [2] นี้ยังได้นำเสนอระบบโครงสร้างที่ประมวลผลบนฮาร์ดแวร์ที่เป็น ASIC (Application-Specific Integrated Circuit) ซึ่งจะนำเสนอในหัวข้อต่อๆ ไป

ปัญหาหลักๆ ที่เกิดขึ้นในการเข้ารหัสระนาบบิตจะเป็นเรื่องของการประมวลผลข้อมูลที่เป็นไปตามลำดับขั้นโดยข้อมูลที่ถูกประมวลผลจะถูกประมวลผลไปทีละตำแหน่งข้อมูล แต่ละ

ระนาบบิต เริ่มต้นตั้งแต่ระนาบบิตที่มีนัยสำคัญมากที่สุดเรื่อยไปจนถึงระนาบบิตที่มีนัยสำคัญน้อยที่สุด

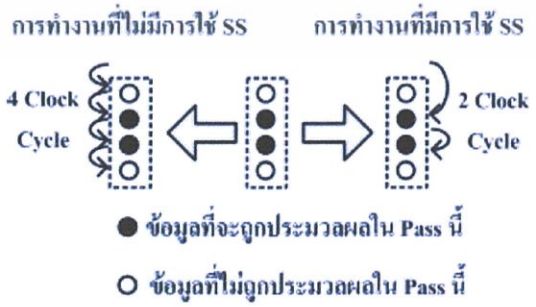
ด้วยสาเหตุของปัญหาดังที่ได้กล่าวข้างต้น หากสามารถลดเวลาในการประมวลผลการเข้ารหัสระนาบบิตลงได้ โดยการใช้การประมวลผลแบบขนานที่มีอยู่ในโครงสร้างทางฮาร์ดแวร์มาประยุกต์เพื่อเพิ่มประสิทธิภาพในการทำงานของการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000 ประสิทธิภาพที่ได้น่าจะเพิ่มสูงขึ้นเป็นอันมาก ดังนั้นในงานวิจัยต่างๆ ที่ได้ศึกษาส่วนใหญ่จะมุ่งไปที่การเพิ่มประสิทธิภาพที่ออกแบบระบบโครงสร้างบนฮาร์ดแวร์

2.3 แนวทางในการแก้ไขปัญหาของแต่ละงานวิจัย

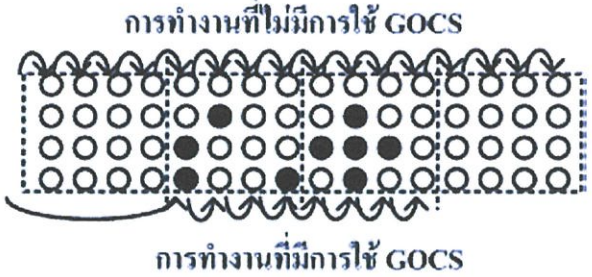
จากหัวข้อที่กล่าวมาจะกล่าวถึงปัญหาที่เกิดขึ้นในการบีบอัดสัญญาณข้อมูลรูปภาพตามมาตรฐาน JPEG2000 ในส่วนของหัวข้อนี้จะเป็นการนำเสนองานวิจัยที่เกี่ยวข้อง ซึ่งออกแบบการทำงานบนอุปกรณ์ฮาร์ดแวร์ โดยนำมาวิเคราะห์จุดเด่นของงานวิจัยต่างๆ เพื่อนำมาใช้ในการพิจารณาการออกแบบระบบโครงสร้างที่จะได้นำเสนอต่อไป

2.3.1 งานวิจัยของ C. Lian *et. al.* [2]

ในส่วนของงานวิจัยนี้จะออกแบบการประมวลผลของการเข้ารหัสระนาบบิตในแบบข้ามข้อมูลในตำแหน่งที่ไม่ต้องประมวลผลและกลุ่มของข้อมูลที่ไม่ต้องประมวลผล (SS & GOCS: Sample Skipping & Group-Of-Column Skipping) ซึ่งการทำงานจะประมวลผลข้อมูลในแต่ละคอลัมน์แบบขนาน โดยมีส่วนที่ใช้ในการควบคุมการทำงานสำหรับการตรวจสอบข้อมูลในแต่ละตำแหน่งว่าข้อมูลในตำแหน่งใดบ้างจะถูกประมวลผล หากมีข้อมูลในตำแหน่งแถวก่อนหน้าที่ไม่ถูกประมวลผล ข้อมูลในตำแหน่งดังกล่าวจะถูกเพิกเฉยและข้ามการประมวลผลในข้อมูลตำแหน่งดังกล่าวไปประมวลผลในแถวถัดไปที่ต้องประมวลผลทันที โดยรูปที่ใช้แสดงการทำงานของ SS และ GOCS สามารถแสดงได้ดังรูปที่ 2.1 และรูปที่ 2.2



รูปที่ 2.1 แสดงวิธีการในการข้ามข้อมูลที่ไม่ต้องถูกประมวลผล (SS: Sample Skipping)



รูปที่ 2.2 แสดงวิธีการข้ามกลุ่มของข้อมูลที่ไม่ต้องถูกประมวลผล

จากรูปที่ 2.1 แสดงตัวอย่างของวิธีการในการข้ามข้อมูลในตำแหน่งที่ไม่ต้องถูกประมวลผล (SS: Sample Skipping) ซึ่งจากในรูปจะเห็นว่าเมื่อมีข้อมูลที่ต้องถูกประมวลผลอยู่เพียง 2 ตัว คือ ข้อมูลในตำแหน่งแถวที่ 1 และ 2 หากระบบที่ออกแบบไม่มีการใช้วิธีการแบบ SS จำเป็นต้องใช้จำนวนลูกสัญญาณนาฬิกาเป็นจำนวนทั้งสิ้น 4 ลูกสัญญาณนาฬิกา แต่เมื่อมีการนำวิธีการแบบ SS มาช่วยในการทำงาน ปรากฏว่าจากจำนวนลูกสัญญาณนาฬิกาที่ต้องใช้ 4 ลูกสัญญาณนาฬิกากลับเหลือที่ต้องใช้เพียง 2 ลูกสัญญาณนาฬิกาเท่านั้น ทำให้สามารถลดเวลาที่ไม่จำเป็นต้องประมวลผลลงได้

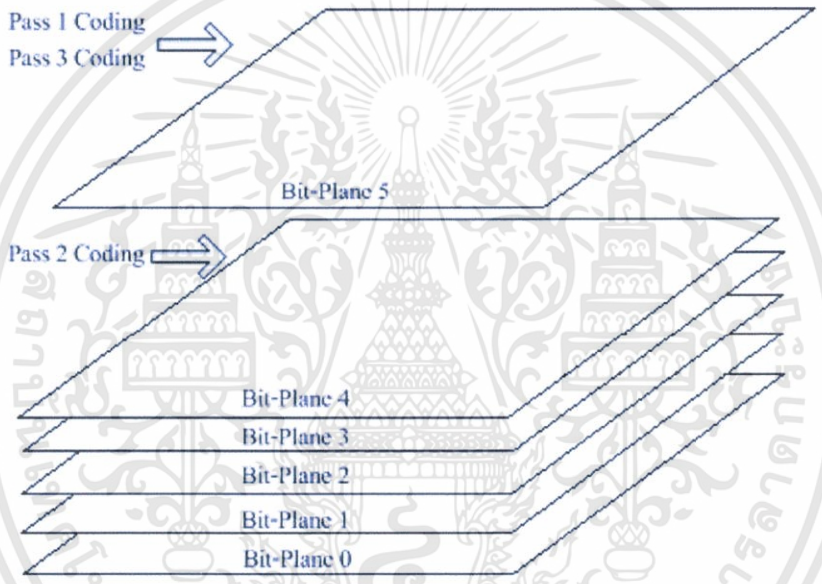
หากข้อมูลที่กำลังจะถูกประมวลผลมีจำนวนของคอลัมน์ที่ไม่ต้องถูกประมวลผลเป็นจำนวนมาก แสดงได้ดังรูปที่ 2.2 หากเป็นระบบที่มีการทำงานแบบธรรมดา คือ ทำงานเป็นลำดับขั้นไป ข้อมูลในตำแหน่ง 4 คอลัมน์แรกและ 4 คอลัมน์สุดท้าย จะถูกประมวลผลเพื่อตรวจสอบการเข้ารหัสข้อมูล แต่ในงานวิจัย [2] จะทำการข้ามกลุ่มของข้อมูลที่ไม่ต้องถูกประมวลผล โดยการข้ามจะข้ามเป็นจำนวนกลุ่มๆ ในแต่ละกลุ่มของข้อมูลที่ข้ามมีขนาด 4 คอลัมน์ ทำให้การทำงานในส่วนนี้ยิ่งสามารถลดเวลาการประมวลผลการเข้ารหัสได้มากยิ่งขึ้น การข้ามข้อมูลเป็นกลุ่มจะกำหนดการข้ามข้อมูลกลุ่มละ 4 คอลัมน์ หากข้อมูลที่สามารถข้ามได้มี 2 คอลัมน์ การข้ามข้อมูลที่เป็นกลุ่มๆ นี้จะไม่ข้ามข้อมูลเหล่านั้น แต่วิธีการข้ามข้อมูลที่เป็น SS จะเข้ามาทำงานทดแทน

แต่การออกแบบการทำงานของการเข้ารหัสข้อมูล จะต้องมีความซับซ้อนเพื่อให้สามารถบ่งบอกได้ว่าข้อมูลในตำแหน่งใดบ้างที่ต้องถูกข้ามไป หรือข้อมูลในตำแหน่งใดต้องถูกประมวลผล ทำให้ขนาดทรัพยากรที่ต้องใช้มีขนาดที่ใหญ่ เพื่อแลกกับความเร็วที่เพิ่มมากขึ้นนั่นเอง โดยความเร็วที่ระบบโครงสร้างจากงานวิจัย [2] จะอยู่ที่ 50 เมกะเฮิรตซ์ บนเทคโนโลยี TSMC 0.35 um

จำนวนโลจิกเกตที่ใช้เป็นจำนวน 19,000 เกต กับหน่วยความจำ 13,000 บิต สามารถประมวลผลได้ 4.6 ล้านจุดภาพ สำหรับความคุ้มค่าของระบบโครงสร้างนี้จะเป็น 4.6 ล้านจุดภาพ / 19000 เกต จะเท่ากับ 242.105 จุดภาพต่อเกต

2.3.2 งานวิจัยของ J. Chiang *et. al.* [3 - 6]

สำหรับงานวิจัยของ J. Chiang *et. al.* จะใช้วิธีในการออกแบบการประมวลผลข้อมูลแบบพาสขนาน (PPCM: Pass Parallel Context Modeling) และการประมวลผลแบบขนานระนาบบิต (Concurrent Bit-Plane Coding)

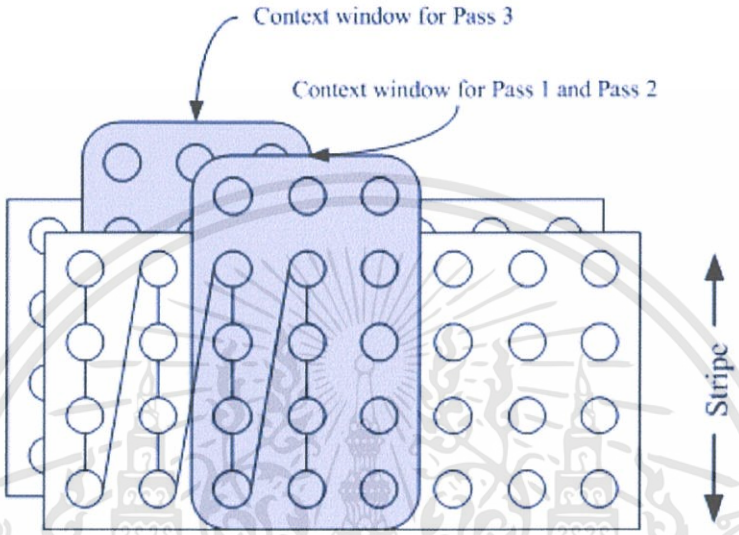


รูปที่ 2.3 แสดงการเข้ารหัสระนาบบิตแบบขนานระนาบบิต (Concurrent Bit-plane Coding) และแบบขนานพาส (Pass Parallel Context Modeling)

จากรูปที่ 2.3 แสดงตัวอย่างของการเข้ารหัสระนาบบิตแบบขนานระนาบบิตและแบบพาสขนาน วิธีของการเข้ารหัสระนาบบิตแบบขนานระนาบบิตจะใช้การประมวลผลข้อมูล 2 ระนาบบิตไปพร้อมๆ กัน จากในตัวอย่าง ข้อมูลในระนาบบิตที่ 5 จะถูกประมวลผลในพาสที่ 1 และพาสที่ 3 ส่วนข้อมูลในระนาบบิตที่ 4 จะประมวลผลในพาสที่ 2 เมื่อข้อมูลในตำแหน่งใดก็ตามที่อยู่ในระนาบบิตที่ 5 ถูกประมวลผลและมีการปรับค่าสถานะนัยสำคัญที่ตำแหน่งใด ข้อมูลที่ตรงกันในตำแหน่งนั้นของระนาบบิตที่ 4 จะถูกประมวลผลด้วยพาสที่ 2 เมื่อข้อมูลถูกประมวลผลไปจนหมดสิ้นทั้งระนาบบิตแล้ว การประมวลผลของพาสที่ 1 และ 3 จะเริ่มประมวลผลใน ระนาบบิตที่ 4 ส่วนการประมวลผลของพาสที่ 2 จะย้ายมาประมวลผลในระนาบบิตที่ 3 แทน การประมวลผลดังที่กล่าว

จะทำงานไปจนกว่าการประมวลผลการเข้ารหัสในพาสที่ 1 และพาสที่ 3 จะประมวลผลข้อมูลในระนาบบิตที่ 0 จนเสร็จ จะเห็นว่าข้อมูลจะถูกประมวลผลไปทีละ 2 ระนาบบิต

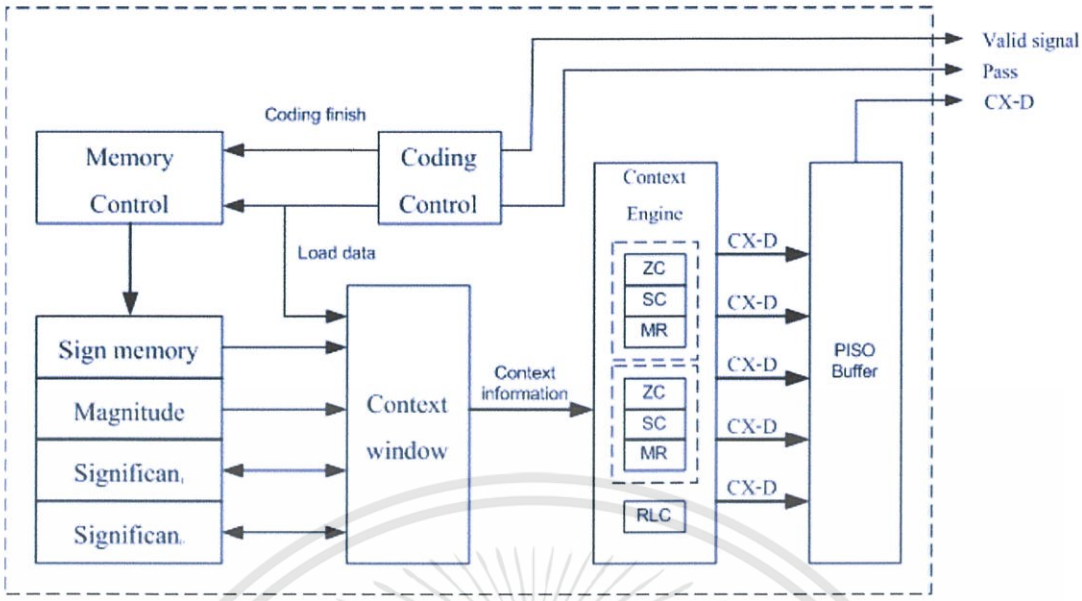
สำหรับการประมวลผลการเข้ารหัสแบบพาสขนานจะเป็นการเข้ารหัสระนาบบิตของพาสที่ 1 และพาสที่ 3 พร้อมกัน โดยข้อมูลที่จะถูกเข้ารหัสจะเหลื่อมกันคนละคอลัมน์



รูปที่ 2.4 แสดงวิธีการเข้ารหัสระนาบบิตแบบพาสขนาน

ในรูปที่ 2.4 แสดงการเข้ารหัสระนาบบิตที่ประมวลผลแบบพาสขนาน จากรูปจะเห็นว่าข้อมูลที่ถูกประมวลผลในพาสที่ 1 กับพาสที่ 3 จะมีการเหลื่อมล้ำข้อมูลกันเป็นระยะห่าง 1 คอลัมน์ ส่วนการประมวลผลของพาส 1 และพาส 2 สามารถประมวลผลข้อมูลในตำแหน่งคอลัมน์เดียวกันได้ เนื่องจากค่าสถานะนัยสำคัญที่ถูกปรับจะมีเพียงช่วงพาส 1 กับ 3 เท่านั้น (ในพาสที่ 2 ไม่มีการปรับค่าสถานะนัยสำคัญ)

จากรูปที่ 2.5 แสดงโครงสร้างของงานวิจัย [3 – 6] โดยงานวิจัยของทั้ง [3 – 6] จะมีระบบโครงสร้างที่เหมือนกัน โดยในระบบโครงสร้างจะใช้หน่วยความจำสำหรับเก็บค่าสถานะนัยสำคัญทั้งหมด 2 ตัว โดยตัวแรกสำหรับช่วงการประมวลผลพาส 1 (SPP) ส่วนอีกตัวหนึ่งสำหรับช่วงการประมวลผลพาส 3 (CUP) ในส่วนของการควบคุมจะส่งสัญญาณไปควบคุมการทำงานของส่วนควบคุมหน่วยความจำและส่วนหน้าต่าง Context สัญญาณ Context ที่ได้จะส่งไปยังตัวดำเนินการพื้นฐานการเข้ารหัส ผลลัพธ์คู่อันดับสัญลักษณ์ที่ได้จะถูกส่งไปยัง Parallel Input Serial Output (PISO) โดยในส่วนของ PISO บัฟเฟอร์จะเป็นส่วนที่ใช้จัดเก็บผลลัพธ์คู่อันดับสัญลักษณ์ที่ได้เพื่อเรียงผลลัพธ์และส่งออกให้ถูกต้องตามลำดับพาส



รูปที่ 2.5 แสดงโครงสร้างของงานวิจัย [3 – 6]

จุดเด่นของงานวิจัย [3 – 6] จะเป็นเรื่องของการเข้ารหัสแบบพาสขนานทำให้สามารถประมวลผลข้อมูลหลายๆ พาสพร้อมๆ กันใน 1 ลูกสัญญาณนาฬิกา และอีกจุดหนึ่ง คือ ระบบโครงสร้างนี้สามารถประมวลผลที่ความเร็ว 185 เมกะเฮิรตซ์ ในส่วนของจุดด้อย คือ ระบบโครงสร้างมีส่วนที่ใช้ในการทำงานแบบขนานหลายจุดทำให้การใช้ทรัพยากรน่าจะมีขนาดที่สูง

จากตารางที่ 2.2 แสดงผลการทำงานของงานวิจัย [3 -6] เปรียบเทียบกับระบบโครงสร้างแบบขนาน จะเห็นได้ว่าค่าเวลาเฉลี่ยที่ระบบโครงสร้างของงานวิจัย [3 – 6] มีประสิทธิภาพที่ดีกว่าระบบโครงสร้างแบบขนานถึง 25.01%

ตารางที่ 2.2 แสดงผลการทำงานของงานวิจัย [3 – 6] เทียบกับระบบโครงสร้างแบบขนาน

Test Image	Execution Time (Clock Cycle)		Decreased Percentage (%)
	Parallel	[3 – 6]	
Lena	1431739	10803918	24.29
Jet	1748425	1383706	25.22
Baboon	1309989	979650	20.86
Boat	1359648	1017169	25.19
Pepper	1277950	945675	26.00
Zelda	1142081	816326	28.52
Average	1378305	1037740	25.01

2.3.3 งานวิจัยของ Y. Li *et. al.* [7]

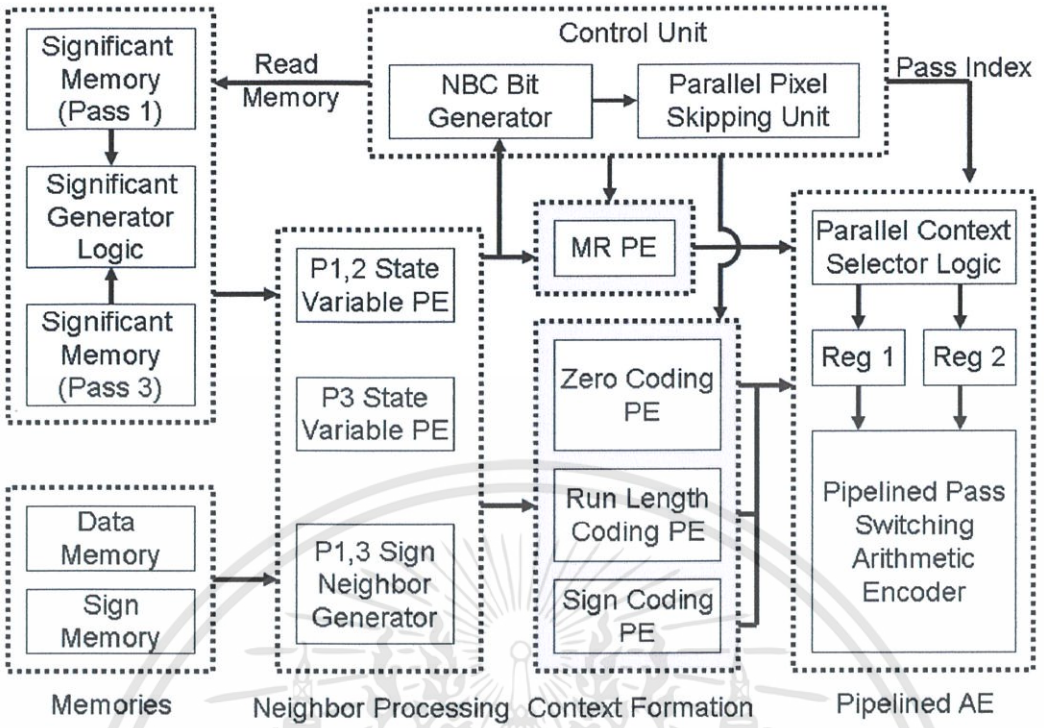
ในงานวิจัย [7] นี้ ระบบโครงสร้างที่ได้ถูกนำเสนอจะใช้หลักการออกแบบที่เป็นแบบพาสขนาน โดยการประมวลผลของพาส 1 และพาส 2 จะประมวลผลข้อมูลพร้อมกัน ส่วนพาส 3 จะประมวลผลในคอลัมน์ที่เหลื่อมถัดมา โดยระบบโครงสร้างของงานวิจัย [7] จะใช้วิธีการคล้ายกับงานวิจัย [3 – 6] อีกทั้งระบบโครงสร้างที่ได้ถูกนำเสนอนี้ยังมีส่วนที่ใช้ควบคุมการข้ามการประมวลผลข้อมูลที่ไม่จำเป็นของการประมวลผลในแต่ละพาส สำหรับระบบโครงสร้างของงานวิจัย [7] สามารถแสดงได้ดังรูปที่ 2.6

จากรูปที่ 2.6 ระบบโครงสร้างของงานวิจัย [7] ในส่วนของโครงสร้างประกอบไปด้วยส่วนของหน่วยความจำใช้เก็บสัญญาณข้อมูลขนาดและสัญญาณข้อมูลเครื่องหมาย ส่วนที่ใช้เก็บค่าสถานะของการเข้ารหัส โดยในโครงสร้างนี้จะใช้หน่วยความจำในการเก็บค่าสถานะนัยสำคัญ 2 ชุด พร้อมทั้งส่วนที่ใช้สร้างสัญญาณสถานะให้กับพาสการทำงานในแต่ละช่วง

ในส่วนของการควบคุมการทำงานของระบบจะมีส่วนที่ใช้ในการหาตำแหน่งข้อมูลที่จะต้องถูกประมวลผล (NBC: Need-Be-Code) และส่งค่าตำแหน่งให้กับส่วนที่ใช้ข้ามการประมวลผลข้อมูลในบางตำแหน่ง (Parallel Pixel Skipping Unit) เพื่อควบคุมตำแหน่งการเข้ารหัสข้อมูลที่ไม่ต้องประมวลผลการเข้ารหัส

ส่วนของการประมวลผลข้อมูลข้างเคียง (Neighbor Processing) เป็นส่วนที่ใช้ประมวลผลข้อมูลข้างเคียงที่เกี่ยวข้องกับตำแหน่งที่กำลังถูกประมวลผล แล้วนำข้อมูลที่ได้ส่งไปให้กับส่วนของตัวดำเนินการพื้นฐานการเข้ารหัสประมวลผลและได้สัญญาณคู่อันดับสัญลักษณ์ ในระบบโครงสร้างของงานวิจัย [7] ยังมีส่วนที่ใช้ในการเข้ารหัสเลขคณิต (Arithmetic Coding) อีกด้วย ในงานวิจัย [7] นี้อัลกอริธึมที่ใช้จะคล้ายกับงานวิจัย [3 – 6] ตรงที่จะใช้กระบวนการในการข้ามข้อมูลที่ไม่ถูกประมวลผลในช่วงของพาสการทำงานต่างๆ

ในส่วนของจุดเด่นในงานวิจัย [7] นี้จะเป็นส่วนของการประมวลผลข้อมูลที่เป็นแบบพาสขนานในช่วงที่การประมวลผลพาสที่ 1 และพาสที่ 2 พร้อมกับมีส่วนที่ใช้ตำแหน่งของข้อมูลที่ไม่ต้องประมวลผลการเข้ารหัสในแต่ละช่วงพาสการทำงาน อีกทั้งการที่มีส่วนของการเข้ารหัสเลขคณิต ในส่วนของจุดด้อยของระบบโครงสร้างจากงานวิจัย [7] คือ เรื่องของการใช้ทรัพยากรมากเนื่องจากมีส่วนที่ใช้ในการควบคุมการข้ามตำแหน่งของข้อมูล



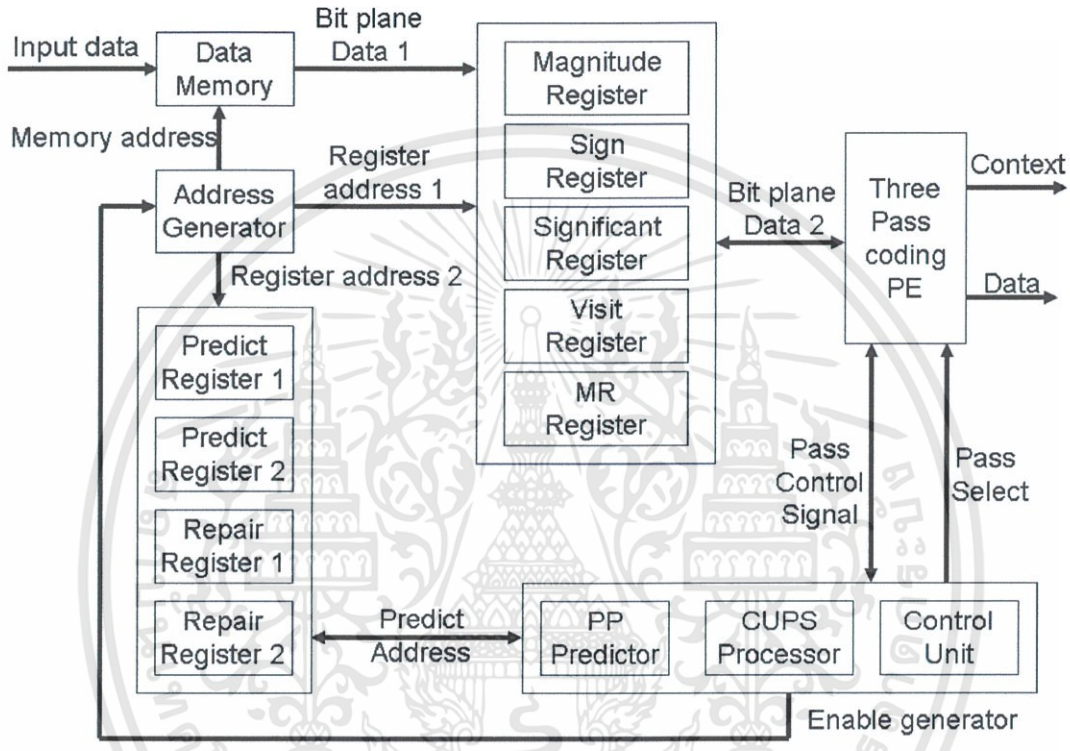
รูปที่ 2.6 แสดงโครงสร้างของงานวิจัย [7]

โดยในงานวิจัยนี้ได้ทำการเปรียบเทียบผลที่ได้กับระบบโครงสร้างแบบพาสขนาน ด้วยวิธีการจำลองประสิทธิภาพของเวลาที่ใช้ไปของระบบโครงสร้างแบบพาสขนาน ซึ่งการจำลองการทำงานของระบบโครงสร้างแบบพาสขนานที่งานวิจัย [7] ได้จำลองออกมาให้เวลาในการประมวลผลเท่ากับ 3.43 ลูกสัญญาณนาฬิกาต่อการเข้ารหัสข้อมูล 1 คอลัมน์ แต่ในงานวิจัย [7] นี้ได้กล่าวไว้ว่าผลการทำงานของงานวิจัย [7] มีประสิทธิภาพที่ดีกว่า อยู่ถึง 16.6 % ของระบบโครงสร้างแบบพาสขนาน และมีประสิทธิภาพที่ดีกว่าระบบโครงสร้างแบบลำดับถึง 38 % เมื่อได้ทำการคำนวณผลของประสิทธิภาพที่ได้จากงานวิจัย [7] แล้ว ทำให้ได้ค่าของปริมาณผลลัพธ์ต่อเวลาประมาณ 6.348 ล้านจุดภาพต่อวินาที แต่เนื่องจากว่าระบบโครงสร้างของงานวิจัย [7] มีการรวมนำเอาส่วนการทำงาน MQ-coder เข้ามาด้วย อีกทั้งการทำงานยังมีส่วนควบคุมการข้ามตำแหน่งข้อมูลที่ไม่ต้องประมวลผล ทำให้ทรัพยากรที่ใช้ไปของระบบโครงสร้างในงานวิจัย [7] มีขนาดที่ใหญ่

2.3.4 งานวิจัยของ T.Tsai *et. al.* [8]

ในงานวิจัย [8] ได้เสนอระบบโครงสร้างการทำงานแบบการพยากรณ์การทำงานพาสและการข้ามส่วนการทำงานของ Cleanup Pass (PP & CUPS: Pass Predicting Method & Cleanup

Pass Skipping Method) ในส่วนของการพยากรณ์การทำงานพาส จะพยากรณ์จุดภาพในตำแหน่งปัจจุบันว่าจะถูกประมวลผลในพาสที่ 1 หรือพาสที่ 2 เพื่อให้ข้อมูลในตำแหน่งปัจจุบันไม่ต้องถูกประมวลผลซ้ำซ้อนขึ้น สำหรับส่วนของการข้ามการทำงานของ Cleanup Pass จะสั่งให้ระบบข้ามการทำงานของ Cleanup Pass หากข้อมูลไม่ต้องถูกประมวลผลใน Cleanup Pass ทั้งระนาบบิต สำหรับรูปที่แสดงโครงสร้างของงานวิจัย [8] สามารถแสดงได้ดังรูปที่ 2.7



รูปที่ 2.7 แสดงโครงสร้างของงานวิจัย [8]

ในระบบโครงสร้างของงานวิจัย [8] ประกอบด้วย ส่วนที่ใช้เก็บข้อมูลที่นำเข้า ส่วนที่ใช้สร้างตำแหน่งของข้อมูลเพื่อคอยบอกตำแหน่งของข้อมูลถัดไปที่ต้องถูกประมวลผล โดยจะมีส่วนสร้างสัญญาณควบคุมตำแหน่งการเข้าถึงข้อมูลในหน่วยความจำมาจากส่วนควบคุมหลักอีกที ส่วนต่อมาจะเป็นส่วนของการประมวลผลการเข้ารหัสข้อมูลที่เป็นส่วนสร้างผลลัพธ์คู่อันดับสัญลักษณ์ได้ผลลัพธ์ออกมาโดยข้อมูลของส่วนการเข้ารหัสจะรับข้อมูลสถานะของการเข้ารหัสมาจากส่วนที่ใช้เก็บค่าสถานะของการเข้ารหัส

โดยจุดเด่นของงานวิจัย [8] จะเป็นส่วนของการพยากรณ์ข้อมูลที่ถูกข้ามได้ ทั้งการข้ามในแต่ละตำแหน่งจุดภาพและการข้ามการประมวลผลของ CUP แต่เมื่อข้อมูลสามารถที่จะถูกข้าม

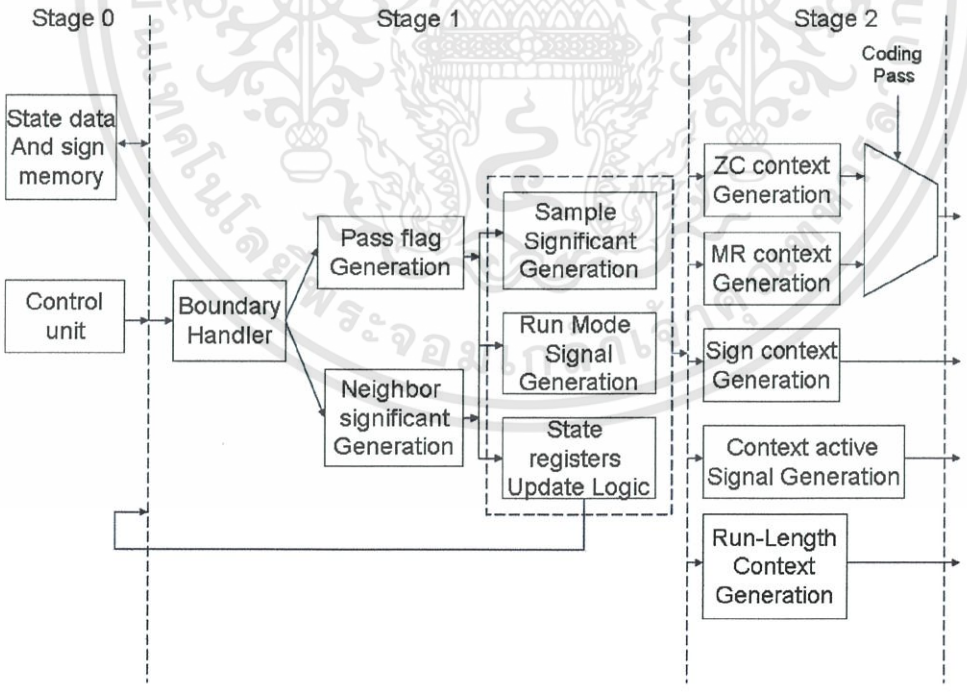
ตำแหน่งได้ทำให้ทรัพยากรที่ใช้ในการเก็บค่าสถานะของตำแหน่งข้อมูลเพิ่มขึ้นมาเพื่อใช้ในการพยากรณ์ตำแหน่งที่ถูกข้าม

ระบบโครงสร้างของงานวิจัย [8] ได้ถูกออกแบบบนชิป ASIC เทคโนโลยี TSMC 0.25um โดยมีขนาด 2.7 x 2.7 mm2 ซึ่งใช้ทรัพยากรเป็นโลจิกเกตถึง 105.9 x 103 เกตและหน่วยความจำขนาด 256 x 22 บิต กับ 64 x 10 บิต อีกอย่างละตัว ความเร็วที่ระบบโครงสร้างสามารถทำได้ที่ 40 MHz ซึ่งใช้พลังงานไฟฟ้าในการทำงานที่ 232.5 mW

ในด้านของประสิทธิภาพทางด้านการประมวลผลข้อมูลจะอยู่ที่ 4.2 ล้านจุดภาพต่อวินาที นั่นคือสามารถประมวลผลรูปภาพขนาด 640 x 480 จุดภาพ จำนวน 13.67 รูปเสร็จในเวลา 1 วินาที

2.3.5 งานวิจัยของ A.K. Gupta *et. al.* [9]

สำหรับงานวิจัย [9] ได้นำเสนอระบบโครงสร้างการทำงานบนฮาร์ดแวร์ โดยการประมวลผลจะเป็นการประมวลผลข้อมูลไปทีละคอลัมน์ ซึ่งจะมีส่วนที่ใช้ในการทดสอบความเป็นสมาชิกของแต่ละพาส ทำให้สามารถประมวลผลข้อมูลทั้งคอลัมน์ในแต่ละตำแหน่งแบบขนานกันไปได้ จึงทำให้มีผลการทำงานในด้านปริมาณงานต่อเวลาที่สูง (Throughput) ระบบโครงสร้างของงานวิจัย [9] สามารถแสดงได้ดังรูปที่ 2.8



รูปที่ 2.8 แสดงโครงสร้างของงานวิจัย [9]

จากรูปที่ 2.8 แสดงโครงสร้างงานวิจัย [9] ซึ่งการทำงานจะแบ่งการทำงานออกเป็น 3 ส่วนการทำงาน (Stage) โดยส่วนการทำงานที่ 1 (Stage 0) เป็นส่วนของการควบคุมและการเข้าถึงข้อมูลในหน่วยความจำ โดยในส่วนการควบคุมทำหน้าที่ควบคุมการทำงานในส่วนต่างๆ ของทั้งระบบให้ทำงานมีความสอดคล้องกัน และในส่วนของการเข้าถึงและจัดเก็บข้อมูลจะทำหน้าที่เก็บข้อมูลของค่าสถานะในการเข้ารหัสและค่าข้อมูลเครื่องหมาย ในส่วนลำดับที่ 2 (Stage 1) เป็นส่วนที่ทำการเตรียมข้อมูลเพื่อให้การเข้ารหัสสามารถประมวลผลได้อย่างรวดเร็ว และทำการปรับค่าสถานะที่มีการเปลี่ยนแปลงไปยังหน่วยความจำ โดยในส่วนของการตรวจสอบข้อมูลบริเวณขอบ (Boundary Handler) จะทำหน้าที่คอยตรวจสอบข้อมูลในบริเวณขอบของรูปภาพ หากข้อมูลอ้างอิงถึงอยู่ในบริเวณนอกขอบเขตของรูปภาพ ข้อมูลในตำแหน่งเหล่านั้นจะถูกแทนค่าด้วยศูนย์ ส่วนย่อยถัดมาคือส่วนของ Pass Flag Generator ทำหน้าที่ทดสอบข้อมูลในแต่ละคอลัมน์ที่กำลังถูกประมวลผลด้วยวิธีการของการทดสอบความเป็นสมาชิกพาส อีกทั้งในส่วนย่อยของเส้นปะใช้สำหรับเตรียมค่าข้อมูลของสัญญาณสถานะที่จำเป็นต่อการเข้ารหัสเตรียมไว้ก่อน เพื่อลดความซับซ้อนในส่วนของการทำงานในส่วนสุดท้าย สำหรับส่วนสุดท้าย (Stage 2) จะเป็นส่วนของการเข้ารหัสข้อมูลที่ได้มาจากการเตรียมข้อมูลในส่วนที่ 2 ออกมาเป็นผลลัพธ์คู่อันดับสัญลักษณ์ ด้วยตัวดำเนินการพื้นฐานในการเข้ารหัส โดยในส่วนนี้มีการใช้งานตัวดำเนินการพื้นฐานแบบแบ่งปันทรัพยากร กล่าวคือ การประมวลผลของตัวดำเนินการพื้นฐานในช่วงเวลาการทำงาน SPP และ CUP ตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์ (ZC) และตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมาย (SC) สามารถใช้งานร่วมกันได้ เนื่องจากการทำงานในแต่ละช่วงเวลาการประมวลผลในแต่ละพาส จะทำงานเพียงพาสการเข้ารหัสเดียว

สำหรับระบบโครงสร้างของงานวิจัย [9] ได้ถูกออกแบบบน FPGA ด้วยเทคโนโลยี APEX20KE ซึ่งได้ทดการทดสอบผลทางด้านจำนวนสัญญาณนาฬิกา และปริมาณผลลัพธ์ทางด้านเอาท์พุต แสดงได้ดังตารางที่ 2.3

ตารางที่ 2.3 แสดงผลลัพธ์ของงานวิจัย [9] เทียบกับระบบโครงสร้างอื่น

Architecture	Clock (MHz)	Cycles/ Code Block	Area (Gates)	Throughput (10^6 Samples/sec)
Simple Timing	51.7	156590	631	1.17
Sample Skipping	38.6	89170	710	1.77
[9]	51.7	46080	956	4.59

จะเห็นได้ว่าจุดเด่นของงานวิจัย [9] จะอยู่ที่สามารถบ่งบอกได้ว่าข้อมูลในแต่ละตำแหน่งของทั้งคอลัมน์ที่กำลังถูกประมวลผลสามารถรู้ได้ว่าจะต้องถูกประมวลผลในพาสการทำงานใด แต่

ว่าการทำงานในส่วนของงานวิจัย [9] ยังคงต้องประมวลผลไปทีละพาส ทำให้ข้อมูลในระนาบบิตเดียวต้องถูกประมวลผลถึง 3 ครั้ง ซึ่งเป็นการสิ้นเปลืองเวลาในการประมวลผลการเข้ารหัส

บทที่ 3
ทฤษฎีพื้นฐานที่เกี่ยวข้อง

3.1 บทนำ

ในบทนี้จะกล่าวถึงทฤษฎีพื้นฐานที่ใช้ในงานวิจัยที่นำเสนอ เพื่อให้ผู้ที่จะศึกษาในงานวิจัยที่นำเสนอนี้ มีความรู้ความเข้าใจพื้นฐาน ทั้งในส่วนของกระบวนการในการบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 รายละเอียดของกระบวนการในการเข้ารหัสระนาบบิต (Context Modeling) และกระบวนการในการทำงานแบบส่งต่อ (Pipeline)

3.2 การบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000

การบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 เป็นมาตรฐานใหม่สำหรับการบีบอัดสัญญาณรูปภาพที่ถูกพัฒนาโดย International Organization for Standardization (ISO) ร่วมกับ International Electrotechnical Commission (IEC) และถูกนำเสนอโดย International Telecommunication Union (ITU)

3.2.1 คุณสมบัติของ JPEG2000

สำหรับคุณสมบัติสำคัญบางส่วนของการบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 เปรียบเทียบกับมาตรฐาน JPEG เดิม สามารถแสดงได้ดังต่อไปนี้

ประสิทธิภาพที่เหนือกว่าด้วยการบีบอัดที่บิตเรตต่ำ: การบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 สามารถบีบอัดข้อมูลได้ในขนาดเล็กและมีค่าความทนทานต่อสัญญาณรบกวนที่สูง หากบีบอัดสัญญาณรูปภาพที่บิตเรต 0.25 บิตต่อจุดภาพทั้งมาตรฐาน JPEG และ JPEG2000 คุณภาพของการบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 สามารถให้คุณภาพที่ดีกว่ามาตรฐาน JPEG

ความต่อเนื่องของการไล่ระดับของทั้งภาพสีและภาพขาวดำ: การบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 สามารถให้คุณภาพของรูปภาพที่มีการไล่ระดับของสี หรือการไล่ระดับของระดับความขาวดำของทั้งรูปภาพสีและรูปภาพขาวดำที่มีความต่อเนื่อง ไม่เกิดการกระโดดของระดับของสีในรูปภาพที่มีการไล่ระดับของสี ซึ่งต่างจากมาตรฐาน JPEG เดิมที่จะเกิดการกระโดดของระดับของสีหากบีบอัดสัญญาณที่บิตเรตต่ำๆ

การบีบอัดรูปภาพที่มีช่วงของข้อมูลในแต่ละจุดภาพที่มีค่ากว้าง: การบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 สามารถเข้ารหัสข้อมูลและถอดรหัสข้อมูลสัญญาณรูปภาพของแต่ละ

สีในช่วงที่กว้างได้ โดยสามารถทำความละเอียดได้ถึง 38 บิตต่อจุดภาพ ซึ่งในส่วนคุณสมบัตินี้ขึ้นอยู่กับการทำงานของทั้งซอฟท์แวร์และฮาร์ดแวร์ที่ใช้

รูปภาพที่มีขนาดใหญ่: การบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 สามารถรองรับรูปภาพที่มีขนาดใหญ่ได้ถึง $(2^{32}-1)\times(2^{32}-1)$ ซึ่งสามารถนำไปใช้กับงานทางด้านภาพถ่ายดาวเทียมที่ต้องบีบอัดข้อมูลรูปภาพที่มีขนาดใหญ่

การบีบอัดข้อมูลที่มีการยอมให้มีการสูญเสียและไม่สูญเสียข้อมูล: โครงสร้างการบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 สามารถรองรับการบีบอัดสัญญาณรูปภาพที่มีการสูญเสียข้อมูลและการบีบอัดที่ไม่สูญเสียข้อมูล ซึ่งรูปภาพทางการแพทย์การเก็บข้อมูลรูปภาพจะไม่ยอมให้มีการสูญเสียข้อมูลมิเช่นนั้นจะทำให้เกิดการวิเคราะห์ผลการรักษาที่ผิดพลาดได้ ซึ่งการบีบอัดสัญญาณรูปภาพมาตรฐานทั้ง JPEG เดิม และ JPEG2000 สามารถรองรับการบีบอัดแบบนี้ได้

การเข้ารหัสพื้นที่ที่สนใจ: ผู้ใช้งานสามารถกำหนดส่วนของรูปภาพที่มีความสำคัญให้เข้ารหัสข้อมูลที่ระดับสูงกว่าข้อมูลในส่วนอื่นของรูปภาพได้ โดยการกำหนดให้จุดที่สนใจมีค่าของข้อมูลที่มีนัยสำคัญกว่าส่วนอื่น

ความทนทานต่อความผิดพลาดของบิตข้อมูล: ความทนทานต่อความผิดพลาดเป็นผลดีต่อระบบการส่งข้อมูลของรูปภาพบนช่องการสื่อสารที่มีการรบกวน ด้วยการเข้ารหัสข้อมูลขนาดบล๊อกเล็กๆ ทำให้สามารถกำหนดจุดของการเข้าจังหวะของการส่งข้อมูลใหม่ในแต่ละส่วนของรูปภาพที่เข้ารหัสได้ ซึ่งเหมาะกับการรับส่งข้อมูลของระบบการสื่อสารไร้สายในยุคที่ 3 (3G)

3.2.2 โครงสร้างส่วนประกอบใน JPEG2000

ในขณะที่วิทยานิพนธ์นี้ถูกเขียนขึ้น การบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 ประกอบไปด้วย 11 ส่วน ซึ่งเป็นไปตามด้านล่างนี้

ส่วนที่ 1 : ระบบการเข้ารหัสแกนกลางถูกนำเสนอโดย International Standard ISO/IEC 15444-1:2000 โดยในส่วนนี้กล่าวถึงกลุ่มของคุณสมบัติพื้นฐานและรูปแบบการเข้ารหัส

ส่วนที่ 2 : ส่วนขยายของส่วนที่ 1 ในส่วนนี้ได้ถูกเพิ่มคุณสมบัติของการเข้ารหัสมากขึ้น

ส่วนที่ 3 : ภาพเคลื่อนไหว JPEG2000 ในส่วนนี้จะกล่าวถึงอัลกอริธึมในการเข้ารหัสข้อมูลที่เป็นลำดับเรียงต่อกัน

ส่วนที่ 4 : โครงร่างการทดสอบเป็นการกำหนดรูปแบบของความเข้ากันได้สำหรับการเข้าและถอดรหัสที่ใช้ส่วนที่ 1 ของมาตรฐาน

ส่วนที่ 5 : ซอฟต์แวร์อ้างอิงจะประกอบด้วยชุดของต้นฉบับซอฟต์แวร์ 2 แบบ ซึ่งถูกนำเสนอใช้เป็นตัวทดสอบและระบุความถูกต้องของระบบที่ถูกพัฒนาขึ้นสำหรับ JPEG2000

ส่วนที่ 6 : รูปแบบของไฟล์รูปภาพหลากองค์ประกอบ (Compound Image File Format) เป็นไฟล์รูปภาพที่ประกอบไปด้วยหลากหลายชั้น (Layer) โดยแต่ละชั้นอาจเป็นข้อมูลที่แตกต่างกันได้ เช่น ข้อมูลภาพที่ได้มาจากหลายต้นฉบับ ได้แก่ จากการถ่ายภาพ การสแกน หรือ ภาพสังเคราะห์ (Synthetic Image) และ ข้อมูลตัวอักษร เป็นต้น ในส่วนนี้เป็นการนำเสนอรูปแบบของไฟล์รูปภาพหลากองค์ประกอบ (.jpm)

ส่วนที่ 7 : ในส่วนนี้ถูกปล่อยว่างไว้

ส่วนที่ 8 : ความปลอดภัยของข้อมูลใน JPEG2000 ในส่วนนี้กล่าวถึงระบบความปลอดภัยของการประยุกต์ใช้ JPEG2000 เช่นการเข้ารหัสลับ การใส่ลายน้ำ และอื่นๆ

ส่วนที่ 9 : ความสัมพันธ์ระหว่างเครื่องมือ ส่วนติดต่อโปรแกรมประยุกต์และโปรโตคอล ในส่วนนี้เป็นส่วนแสดงโปรโตคอลการสื่อสารที่เกี่ยวข้องกับการแลกเปลี่ยนรูปภาพมาตรฐาน JPEG2000 กับข้อมูลที่เกี่ยวข้อง

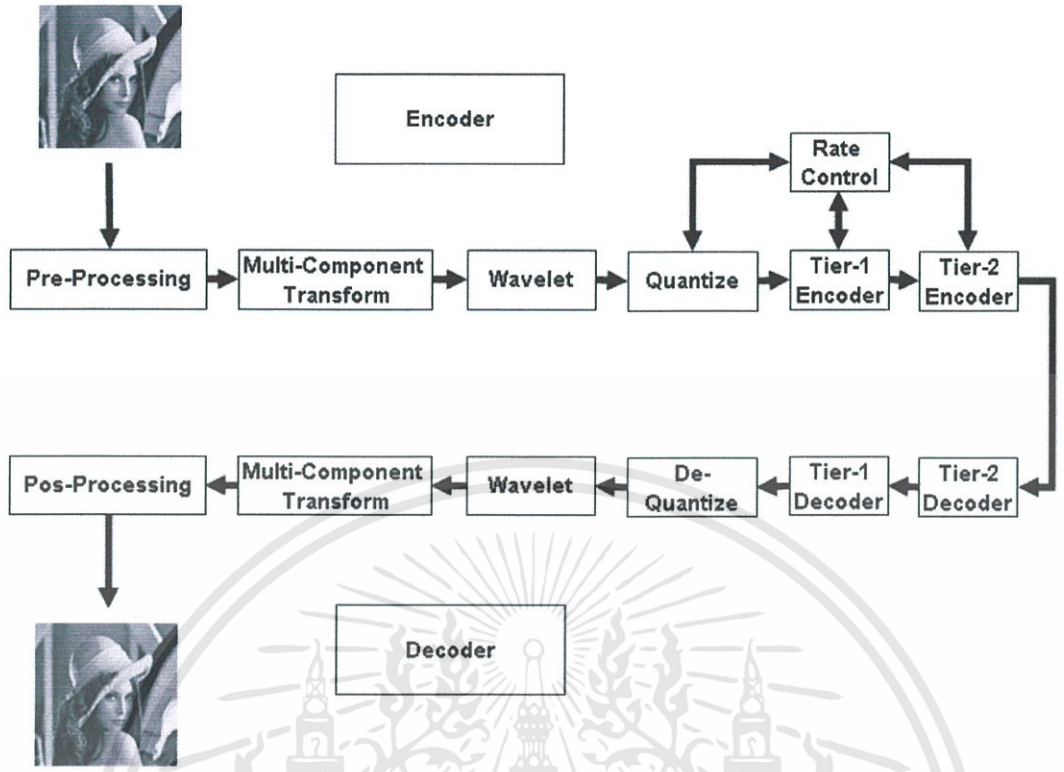
ส่วนที่ 10 : ข้อมูล 3 มิติ และข้อมูลจุดทศนิยม (JP3D) ในส่วนนี้เป็นการพัฒนารูปภาพที่เป็น 3 มิติ เช่น รูปภาพทางการแพทย์แบบ 3 มิติ

ส่วนที่ 11 : การสื่อสารไร้สาย (JPWL) ในส่วนนี้เป็นการพัฒนาสำหรับการประยุกต์สื่อสารไร้สาย โดยมีการป้องกันการเกิดความผิดพลาด การตรวจพบความผิดพลาด และการแก้ไขความผิดพลาด

ส่วนที่ 12 : รูปแบบพื้นฐานของมัลติมีเดียไฟล์มาตรฐาน ISO/IEC 14496-12 สำหรับ MPEG-4

3.2.3 โครงสร้างส่วนประมวลผลโดยรวมของ JPEG2000

กระบวนการเข้ารหัสตามมาตรฐานการบีบอัดสัญญาณรูปภาพ JPEG2000 สามารถแสดงแผนภาพการทำงานได้ดังรูปที่ 3.1 การบีบอัดสัญญาณรูปภาพเริ่มจาก การนำรูปภาพต้นแบบส่งเข้าบล็อก Pre-processing เพื่อแบ่งสัญญาณรูปภาพออกเป็นส่วนย่อยๆ (Tiling) หลังจากนั้นสัญญาณรูปภาพที่ถูกแบ่งจะถูกแปลงสีด้วยบล็อก Multi-component Transform แล้วผ่านการแปลงสัญญาณรูปภาพด้วย เทคนิคการแปลงเวฟเลท (Wavelet Transform) เพื่อให้ได้ค่าสัมประสิทธิ์ของเวฟเลทออกมา ค่าสัมประสิทธิ์ของเวฟเลทที่ได้จะผ่านการลดระดับค่าของข้อมูลด้วยบล็อก Quantize ให้มีระดับของข้อมูลที่น้อยลง ค่าที่ได้จากการลดระดับถูกส่งเข้าสู่บล็อกการเข้ารหัส Tier-1 (Tier-1 encoder) เพื่อเข้ารหัสสัญญาณข้อมูลและได้ผลลัพธ์เป็นสายรหัสของข้อมูล (Sequence of Compressed Data) และกระบวนการสุดท้าย สายรหัสของข้อมูลที่ผ่านการเข้ารหัส Tier-1 เพื่อบีบอัดข้อมูลจะถูกจัดแพคเก็จด้วยการเข้ารหัส Tier-2 (Tier-2 encoder) สำหรับรายละเอียดการทำงานของแต่ละบล็อกจะถูกกล่าวในหัวข้อต่อๆ ไป

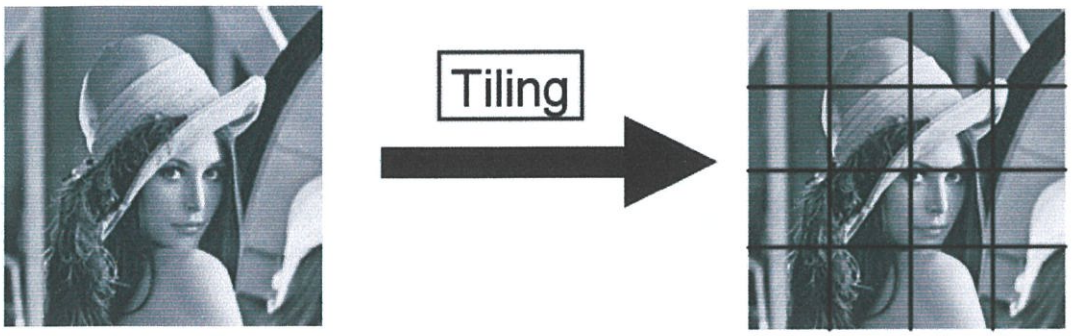


รูปที่ 3.1 แสดงแผนภาพการทำงานตามมาตรฐานการบีบอัดสัญญาณรูปภาพ JPEG2000

3.2.3.1 การเตรียมความพร้อมก่อนการบีบอัดภาพ (Pre-processing)

การประมวลผลแรกสุดในการบีบอัดสัญญาณรูปภาพคือการแบ่งข้อมูลสัญญาณรูปภาพออกเป็นบล๊อก โดยส่วนที่ทำการแบ่งข้อมูลสัญญาณรูปภาพนี้เรียกว่า "Tiling" โดยกระบวนการนี้เริ่มแรกสัญญาณรูปภาพต้นฉบับที่เข้ามาจะถูกแบ่งออกเป็นบล๊อก แต่ละบล๊อกเป็นพื้นที่สี่เหลี่ยมที่ไม่ทับซ้อนกัน โดยแต่ละบล็อกของข้อมูลที่ถูกแบ่งออกมาเรียกว่า "Tile" โดยขนาดของแต่ละ Tile มีขนาดที่เท่ากัน ยกเว้นในส่วนพื้นที่ขอบของรูปภาพ สำหรับรูปภาพที่ประกอบไปด้วยหลายสี สัญญาณข้อมูลในแต่ละ Tile จะประกอบไปด้วยสัญญาณข้อมูลหลายสีเช่นเดียวกัน หากสัญญาณข้อมูลของรูปภาพประกอบไปด้วยสีระดับเทา สัญญาณข้อมูลในแต่ละ Tile ที่แบ่งออกมาจะเป็นสัญญาณข้อมูลระดับสีเทาเช่นเดียวกับรูปต้นฉบับ

อย่างไรก็ดีเมื่อทำการออกแบบฮาร์ดแวร์ที่ใช้ในการประมวลผลสำหรับมาตรฐาน JPEG2000 ในส่วนการแบ่งขนาดของข้อมูลนี้ ถ้าขนาดข้อมูลที่ต้องการแบ่งในแต่ละ Tile มีขนาดใหญ่มากทำให้จำเป็นที่ระบบต้องจองหน่วยความจำขนาดใหญ่มากตามไปด้วย ซึ่งจะทำให้ขนาดของฮาร์ดแวร์ใหญ่เกินความจำเป็น ดังนั้นขนาดของ Tile ที่เหมาะสมในการประยุกต์ใช้งานการบีบอัดภาพบนฮาร์ดแวร์นั้น อยู่ที่ขนาดประมาณ 256 x 256 หรือ 512 x 512 จุดภาพ



รูปที่ 3.2 การแบ่งพื้นที่ของข้อมูลสัญญาณรูปภาพ (Tiling)

3.2.3.2 ส่วนการแปลงสัญญาณสีของรูปภาพ (Multi-component Transformation)

การแปลงสัญญาณสีของรูปภาพเป็นการช่วยลดความสัมพันธ์ระหว่างองค์ประกอบสีที่พิจารณา ผลที่ได้จะช่วยเพิ่มประสิทธิภาพในการบีบอัด ซึ่งมาตรฐาน JPEG2000 รองรับการแปลงสีของรูปภาพ 2 แบบคือ (1) Reversible Color Transformation (RCT) และ (2) Irreversible Color Transformation (ICT) โดยระบบ RCT จะรองรับการบีบอัดทั้งแบบที่มีการสูญเสียข้อมูลและไม่สูญเสียข้อมูล ส่วน ICT จะรองรับการบีบอัดสัญญาณรูปภาพที่ยอมสูญเสียข้อมูลเพียงอย่างเดียว

สมการที่ใช้ในการแปลงสัญญาณสีของรูปภาพสำหรับระบบ RCT สามารถแสดงได้ดังสมการที่ (3.1) และ (3.2)

Forward RCT:

$$\begin{aligned} Y_r &= \left\lfloor \frac{R+2G+B}{4} \right\rfloor \\ U_r &= B-G \\ V_r &= R-G \end{aligned} \tag{3.1}$$

Invert RCT:

$$\begin{aligned} G &= Y_r - \left\lfloor \frac{U_r+V_r}{4} \right\rfloor \\ R &= V_r+G \\ B &= U_r+G \end{aligned} \tag{3.2}$$

ในส่วนของสมการที่ใช้ในการแปลงสัญญาณสีของรูปภาพสำหรับระบบ ICT สามารถแสดงได้ดังสมการที่ (3.3) และ (3.4)

Forward ICT:

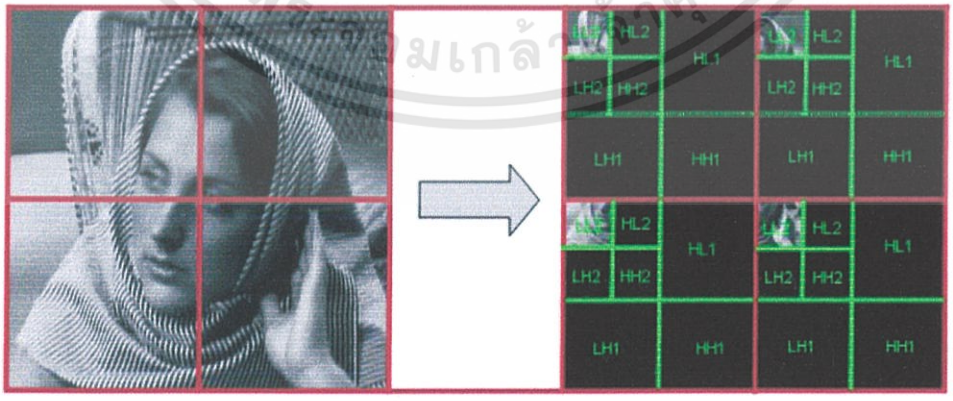
$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299000 & 0.587000 & 0.114000 \\ -0.168736 & -0.331264 & 0.500000 \\ 0.500000 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (3.3)$$

Invert ICT:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0 & 1.402000 \\ 1.0 & -0.344136 & -0.714136 \\ 1.0 & 1.772000 & 0.0 \end{bmatrix} \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} \quad (3.4)$$

3.2.3.3 ส่วนการแปลงสัญญาณรูปภาพด้วยกระบวนการแปลงเวฟเลท (Wavelet Transform)

ความแตกต่างหลักระหว่าง JPEG และ JPEG2000 คือ การเพิ่มทางเลือกในการเลือกเทคนิคในการแปลงสัญญาณ โดยใน JPEG2000 สามารถเลือกใช้เทคนิคการแปลงสัญญาณได้ทั้งเทคนิคการแปลงเวฟเลท (DWT: Discrete Wavelet Transform) และ การแปลงสัมประสิทธิ์โคไซน์ (DWT: Discrete Cosine Transform) ในขณะที่ JPEG เดิมใช้เทคนิคการแปลงสัมประสิทธิ์โคไซน์เท่านั้น โดยเทคนิคการแปลงเวฟเลทเป็นที่นิยมมากกว่า เนื่องจากเป็นการแปลงที่ทำให้บีบอัดได้ด้วยคุณภาพที่สูงกว่าในอัตราการบีบอัดเท่ากัน



รูปที่ 3.3 แสดงตัวอย่างภาพภายหลังการแปลงเวฟเลท

รูปที่ 3.3 แสดงลักษณะของรูปภาพที่ผ่านการแปลงเวฟเลท ซึ่งรูปภาพที่ยกมาแสดงนี้แบ่งพื้นที่ของรูปออกเป็น 4 Tile โดยในแต่ละ Tile ผ่านการแปลงเวฟเลทออกเป็น 2 ระดับ สำหรับระดับที่ 1 คือ LH1, HL1 และ HH1 ส่วนระดับที่ 2 คือ LL2, LH2, HL2 และ HH2

สำหรับการบีบอัดสัญญาณรูปภาพแบบที่ยอมให้มีการสูญเสีย ในส่วนการแปลงสัญญาณข้อมูลเวฟเลทนี้จะใช้ตัวกรองเวฟเลทแบบ Daubechies (9, 7) ซึ่งตัวกรองเวฟเลท (9, 7) ในช่วงการวิเคราะห์จะใช้ตัวกรองความถี่ต่ำ 9-tap และตัวกรองความถี่สูง 7-tap สำหรับตัวกรองในการวิเคราะห์ข้อมูลสามารถแสดงได้ดังข้างล่าง

ตัวกรองความถี่ต่ำ 9-tap: $[h_{-4}, h_{-3}, h_{-2}, h_{-1}, h_0, h_1, h_2, h_3, h_4]$

$$h_4 = h_{-4} = \quad +0.026748757410810$$
$$h_3 = h_{-3} = \quad -0.016864118442875$$
$$h_2 = h_{-2} = \quad -0.078223266528988$$
$$h_1 = h_{-1} = \quad +0.266864118442872$$
$$h_0 = \quad +0.602949018236358$$

ตัวกรองความถี่สูง 7-tap: $[g_{-3}, g_{-2}, g_{-1}, g_0, g_1, g_2, g_3]$

$$g_3 = g_{-3} = \quad +0.0912717631142495$$
$$g_2 = g_{-2} = \quad -0.057543526228500$$
$$g_1 = g_{-1} = \quad -0.591271763114247$$
$$g_0 = \quad +1.115087052456994$$

สำหรับตัวกรองที่ใช้สังเคราะห์ในการแปลงกลับเวฟเลทในส่วนของตัวกรองความถี่ต่ำจะใช้ตัวกรอง 7-tap และตัวกรองความถี่สูงจะใช้ตัวกรอง 9-tap ค่าของตัวกรองความถี่สามารถแสดงได้ดังด้านล่าง

ตัวกรองความถี่ต่ำ 7-tap: $[h'_{-3}, h'_{-2}, h'_{-1}, h'_0, h'_1, h'_2, h'_3]$

$$h'_3 = h'_{-3} = \quad -0.0912717631142495$$
$$h'_2 = h'_{-2} = \quad -0.057543526228500$$
$$h'_1 = h'_{-1} = \quad +0.591271763114247$$
$$h'_0 = \quad +1.115087052456994$$

ตัวกรองความถี่สูง 9-tap: $[g'_{-4}, g'_{-3}, g'_{-2}, g'_{-1}, g'_0, g'_1, g'_2, g'_3, g'_4]$

$$g'_4 = g'_{-4} = \quad +0.026748757410810$$
$$g'_3 = g'_{-3} = \quad +0.016864118442875$$
$$g'_2 = g'_{-2} = \quad -0.078223266528988$$
$$g'_1 = g'_{-1} = \quad -0.266864118442872$$
$$g'_0 = \quad +0.602949018236358$$

ส่วนการบีบอัดสัญญาณรูปภาพที่ไม่ยอมให้มีการสูญเสียข้อมูล ในการแปลงเวฟเลทจะใช้ตัวกรองแบบ Le Gall (5, 3) ซึ่งตัวกรองที่กล่าวนี้สามารถประยุกต์ไปใช้ในการแปลงเวฟเลทที่ยอมให้มีการสูญเสียข้อมูลทดแทนตัวกรอง Daubechies (9, 7) ได้ อย่างไรก็ตามตัวกรอง Daubechie (9, 7) มีประสิทธิภาพในการบีบอัดและคุณภาพในการบีบอัดสัญญาณรูปภาพที่ยอมให้มีการสูญเสียข้อมูลที่ดีกว่าตัวกรอง Le Gall (5, 3) ในการวิเคราะห์สัญญาณรูปภาพของการแปลงเวฟเลทสามารถแสดงได้ดังด้านล่าง

ตัวกรองความถี่ต่ำ 5-tap: $[h_{-2}, h_{-1}, h_0, h_1, h_2]$

$$h_2 = h_{-2} = \quad -1/8$$
$$h_1 = h_{-1} = \quad ¼$$
$$h_0 = \quad ¾$$

ตัวกรองความถี่สูง 3-tap: $[g_{-1}, g_0, g_1]$

$$g_1 = g_{-1} = \quad -1/2$$
$$g_0 = \quad 1$$

ส่วนการสังเคราะห์ข้อมูลของการแปลงกลับเวฟเลทสามารถแสดงได้ดังด้านล่าง

ตัวกรองความถี่ต่ำ 3-tap: $[h'_{-1}, h'_0, h'_1]$

$$h'_1 = h'_{-1} = \quad ½$$
$$h'_0 = \quad 1$$

ตัวกรองความถี่สูง 5-tap: $[g'_{-2}, g'_{-1}, g'_0, g'_1, g'_2]$

$$g'_2 = g'_{-2} = \text{-1/8}$$

$$g'_1 = g'_{-1} = \text{-1/4}$$

$$g'_0 = \text{¾}$$

3.2.3.4 ส่วนการลดระดับสัญญาณข้อมูล (Quantization)

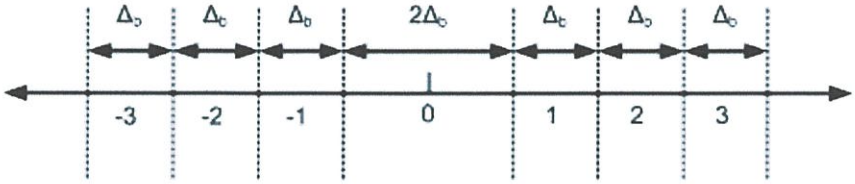
ภายหลังจากการแปลงเวฟเลท ข้อมูลทุกๆ Code-block จะถูกลดระดับสัญญาณข้อมูลเมื่อเป็นการบีบอัดสัญญาณรูปภาพที่ยอมให้มีการสูญเสียข้อมูล (Lossy) การลดระดับสัญญาณข้อมูลเป็นสาเหตุหลักหนึ่งที่ทำให้เกิดการสูญเสียข้อมูล ในส่วนที่ 1 ของการบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 ใช้การลดระดับสัญญาณข้อมูลแบบ Uniform โดยค่าในแต่ละระดับมีระยะความกว้าง Δ_b

สมการที่ใช้ในการหาค่าสัญญาณข้อมูลที่ผ่านการลดระดับสัญญาณข้อมูลสามารถแสดงดังสมการที่ (3.5)

$$q_b(i,j) = sign(y_b(i,j)) \left\lfloor \frac{|y_b(i,j)|}{\Delta_b} \right\rfloor \tag{3.5}$$

ค่า $y_b(i,j)$ คือ ค่าของข้อมูลที่คิดเครื่องหมายในตำแหน่ง (i, j) ส่วนค่า $q_b(i,j)$ คือค่าของข้อมูลที่ผ่านการลดระดับสัญญาณข้อมูล ณ. ตำแหน่งเดียวกัน

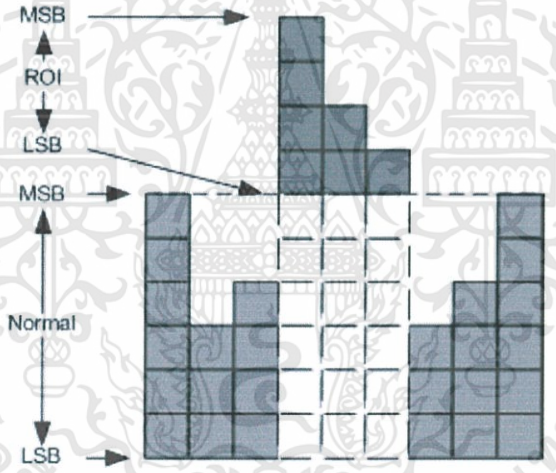
โดยความสัมพันธ์ระหว่างค่า Δ_b กับค่า q_b นั่นคือ เมื่อค่า Δ_b (Quantization Step) มีค่ากว้างมากขึ้น จะทำให้ค่าของระดับสัญญาณที่ลดระดับ q_b มีช่วงค่าที่กว้างน้อยลง ส่งผลให้จำนวนบิตของการจัดระดับข้อมูลมีจำนวนลดลงด้วย หากค่า Δ_b มีความกว้างน้อย จะทำให้ค่าของระดับสัญญาณที่ลดระดับ q_b ก็จะมีช่วงค่าที่กว้างมากขึ้น เช่น ข้อมูลก่อนผ่านการลดระดับข้อมูลมีจำนวนบิตเท่ากับ 16 ระนาบบิต เมื่อค่า Δ_b มีค่าเท่ากับ 2 จะทำให้จำนวนระนาบบิตของข้อมูลที่ผ่านการลดระดับข้อมูลมีจำนวนบิต เท่ากับ 15 บิต หากค่า Δ_b มีค่าเท่ากับ 32 จะทำให้การลดระดับข้อมูลสามารถลดจำนวนบิตของข้อมูลลงได้ถึง 5 บิต หรือข้อมูลที่ผ่านการลดระดับข้อมูลมีจำนวนบิตเท่ากับ 11 บิต จะเห็นได้ว่ายิ่งค่าของ Δ_b มีความกว้างมากๆ จะทำให้ช่วงค่าของการลดระดับข้อมูลมีค่ากว้างน้อยลง ส่งผลให้จำนวนบิตของการลดระดับข้อมูลยิ่งน้อยตามลงไปด้วย และทำให้ข้อมูลในส่วนรายละเอียดของรูปภาพที่จะถูกเข้ารหัสบีบอัดข้อมูลยิ่งลดน้อยตามลงไปด้วย



รูปที่ 3.4 การลดระดับสัญญาณข้อมูลแบบ Uniform

3.2.3.5 ส่วนการเข้ารหัสของพื้นที่ที่สนใจ (Region of Interest Coding)

การเข้ารหัสพื้นที่ที่สนใจ (ROI: Region of Interest) เป็นหนึ่งในคุณสมบัติของการบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 โดยยอมให้การเข้ารหัสสัญญาณรูปภาพต่างกันในแต่ละพื้นที่ ในส่วนที่ 1 ของการบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 กรรมวิธีในการเข้ารหัสของพื้นที่ที่สนใจนี้ถูกเรียกว่า "MAXSHIFT"



รูปที่ 3.5 แสดงการยกระดับพื้นที่ที่สนใจด้วยวิธี MAXSHIFT

จากรูปที่ 3.5 แสดงการยกระดับระนาบบิตที่มีนัยสำคัญต่ำที่สุดของ ROI ให้ระนาบบิตเท่ากับระนาบบิตที่มีนัยสำคัญสูงที่สุดของพื้นที่ที่ไม่ได้สนใจ ในการนำเสนอค่าของข้อมูลสัมประสิทธิ์เวฟเลทที่ผ่านการลดระดับระนาบบิตลงแล้วนั้น ปัญหาที่เกิดการสเกลค่าในส่วน ROI อาจทำให้เกิดการล้นของข้อมูลเนื่องจากระดับความละเอียดของข้อมูลกว้างมากขึ้น เพื่อป้องกันการเกิดปัญหา จึงลดสเกลของพื้นที่ที่ไม่สนใจให้ต่ำกว่าแทน สำหรับข้อมูลในการทำ ROI จะถูกรวมเข้าไปในส่วนหัวของแต่ละบล็อก เพื่อให้การถอดรหัสข้อมูลกลับสามารถปรับระดับของระนาบบิตให้ถูกต้องดังเดิมได้

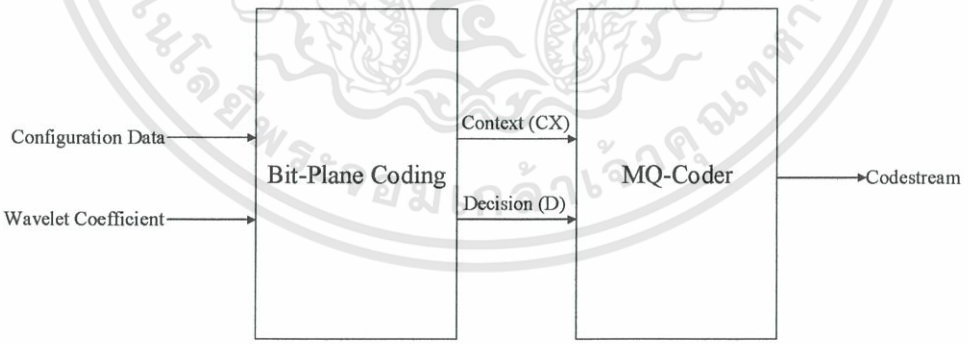
3.2.3.6 ส่วนควบคุมอัตราบิตเรต (Rate Control)

ส่วนควบคุมอัตราบิตเรตเป็นประเด็นหนึ่งที่ถกเถียงในการบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 ซึ่งส่วนควบคุมอัตราบิตเรตเป็นส่วนการประมวลผลที่ทำให้อัตราบิตเรตของแต่ละรูปภาพ มีอัตราบิตเรตที่เหมาะสมสำหรับช่องทางที่เลือกใช้ในการสื่อสาร อัตราบิตเรทที่เลือกจะมีผลกระทบโดยตรงต่อค่าความผิดพลาดของการถอดการบีบอัด ดังนั้นอัตราบิตเรตที่เหมาะสมที่สุด คือ จุดที่ให้ค่า MSE (Mean Square Energy) มีค่ามากที่สุด

สำหรับวิธีการที่ง่ายที่สุดในการควบคุมอัตราบิตเรต โดยอาศัยการเลือกความกว้างของการลดระดับสัญญาณข้อมูล (Δ_b) ขนาดความกว้างของการลดระดับสัญญาณข้อมูลที่มีค่ากว้างมากขึ้นทำให้อัตราบิตเรตยิ่งลดต่ำลง อย่างไรก็ตามการลดระดับสัญญาณข้อมูลสามารถประยุกต์ได้กับการบีบอัดสัญญาณรูปภาพแบบที่ยอมให้สูญเสียข้อมูล ทุกครั้งที่ความกว้างของการลดระดับสัญญาณเปลี่ยนแปลง การเข้ารหัส Tier-1 จะถูกประมวลผลใหม่ ฮาร์ดแวร์ที่ใช้ในส่วนการควบคุมอัตราบิตเรต ส่วนใหญ่จะใช้ไมโครคอลโทรลเลอร์ในการประมวลผล

3.2.3.7 ส่วนการเข้ารหัส Tier-1 (Tier-1 Coding)

ในส่วนการเข้ารหัส Tier-1 เป็นขั้นตอนการเข้ารหัสที่ดำเนินการหลังจากการแปลงสัญญาณด้วยเทคนิคการแปลงเวฟเลท โดยข้อมูลในแต่ละ Code-block จะถูกเข้ารหัสโดยไม่ขึ้นต่อกัน การเข้ารหัสในส่วนของ Tier-1 นี้ สามารถแบ่งได้เป็น 2 ส่วน คือ ส่วนการเข้ารหัสระนาบบิตและการเข้ารหัส MQ-coder



รูปที่ 3.6 แสดงโครงสร้างส่วน Tier-1 Encoder ตามมาตรฐาน JPEG2000

เนื่องจากข้อมูลของรูปภาพที่จะถูกส่งหรือเก็บลงสื่อบันทึกข้อมูล หากเก็บข้อมูลรูปภาพเป็นแบบข้อมูลดิบ พื้นที่ที่ใช้เก็บข้อมูลจะมีขนาดใหญ่มาก เมื่อข้อมูลรูปภาพผ่านการบีบอัดข้อมูลจะมีขนาดที่เล็กลง แต่ว่าข้อมูลที่จะถูกบีบอัดนั้นในแต่ละตำแหน่งของข้อมูลมีความหลากหลาย ทำ

ให้กระบวนการในการบีบอัดข้อมูลมีประสิทธิภาพน้อย หากข้อมูลในแต่ละตำแหน่งของรูปภาพมีลักษณะของข้อมูลที่มีความเหมือนกันมาก การบีบอัดข้อมูลรูปภาพจะยิ่งมีประสิทธิภาพมากขึ้น เพราะว่าข้อมูลที่ซ้ำซ้อนกันเหล่านั้นสามารถแทนได้ด้วยสัญลักษณ์ที่ต้องการพื้นที่ในการจัดเก็บที่น้อยๆ ทำให้สามารถลดขนาดพื้นที่ของรูปภาพได้ ดังนั้นในส่วนของการเข้ารหัสระนาบบิตจึงพยายามสร้างรูปแบบของข้อมูลที่มีความหลากหลายของข้อมูลลดลง ตัวอย่างเช่น ถ้าค่าของข้อมูลแทนด้วยจำนวนบิต 16 บิต นั่นคือสามารถแสดงข้อมูลรูปภาพที่มีค่า 2^{16} สัญลักษณ์ การเข้ารหัสระนาบบิตจะทำการลดความหลากหลายของข้อมูล ให้อยู่ในรูปแบบคู่อันดับของ (Context, Decision) จำนวน Context ทั้งสิ้น 19 สัญลักษณ์และ Decision 2 สัญลักษณ์ รวมแล้วจะมีความหลากหลายของข้อมูลทั้งสิ้น 38 ลักษณะ เมื่อทำการลดความหลากหลายของข้อมูลลง จะส่งผลให้การบีบอัดข้อมูลในส่วนของการเข้ารหัสเลขคณิต มีประสิทธิภาพมากขึ้นจึงส่งผลให้ได้อัตราการบีบอัดที่สูงขึ้น

ในส่วนการเข้ารหัสระนาบบิต ข้อมูลในแต่ละ Code-block จะถูกเข้ารหัสตั้งแต่ระนาบบิตที่มีนัยสำคัญมากที่สุดไปจนถึงระนาบบิตที่มีนัยสำคัญน้อยที่สุด ผลลัพธ์ที่ได้จากการเข้ารหัสระนาบบิต จะอยู่ในรูปของคู่อันดับของสัญลักษณ์และค่าการตัดสินใจ (Context, Decision) การเข้ารหัสระนาบบิตตามมาตรฐาน JPEG2000 ทำโดยอาศัยอัลกอริธึม EBCOT ซึ่งเป็นอัลกอริธึมการเข้ารหัสแต่ละระนาบบิตใน 3 พาส การเข้ารหัสของข้อมูลในแต่ละตำแหน่งของข้อมูลจะถูกเข้ารหัสพาสใดหนึ่งใน 3 พาส โดยผลของการเข้ารหัสในแต่ละพาสจะขึ้นอยู่กับข้อมูลแต่ละตำแหน่งในแต่ละระนาบบิตและค่าสถานะของการเข้ารหัส

ในส่วนการเข้ารหัส MQ-coder หรือการเข้ารหัสเลขคณิต ในส่วนนี้เป็นส่วนที่เข้ารหัสถัดจากการเข้ารหัสระนาบบิต โดยการนำคู่สัญลักษณ์จากส่วนของการเข้ารหัสระนาบบิตไปบีบอัดอีกต่อหนึ่ง โดยอาศัยการเข้ารหัสบนพื้นฐานของค่า Entropy ซึ่งคำนวณจากค่าโอกาสความน่าจะเป็นในการพบแต่ละสัญลักษณ์ที่ต้องการเข้ารหัส เช่นเดียวกับ Huffman และการเข้ารหัสเลขคณิตทำให้ข้อมูลที่ได้มีขนาดที่เล็กลงเป็นอันมาก

3.2.3.8 ส่วนการเข้ารหัส Tier-2 (Tier-2 Coding)

ภายหลังการเข้ารหัสใน Tier-1 จะได้ผลลัพธ์การบีบอัดเป็นสายบิตข้อมูล (Bit Stream) ซึ่งจะถูกส่งไปเข้ารหัสต่อในส่วนการเข้ารหัส Tier-2 โดยในส่วนการประมวลผลของส่วนนี้เป็นส่วนที่ทำหน้าที่ในการจัดรูปแบบของสายข้อมูลที่ผ่านการบีบอัดให้เป็นไปตามรูปแบบการเข้ารหัสตามมาตรฐานการบีบอัดสัญญาณข้อมูลที่เป็นรูปภาพ JPEG2000

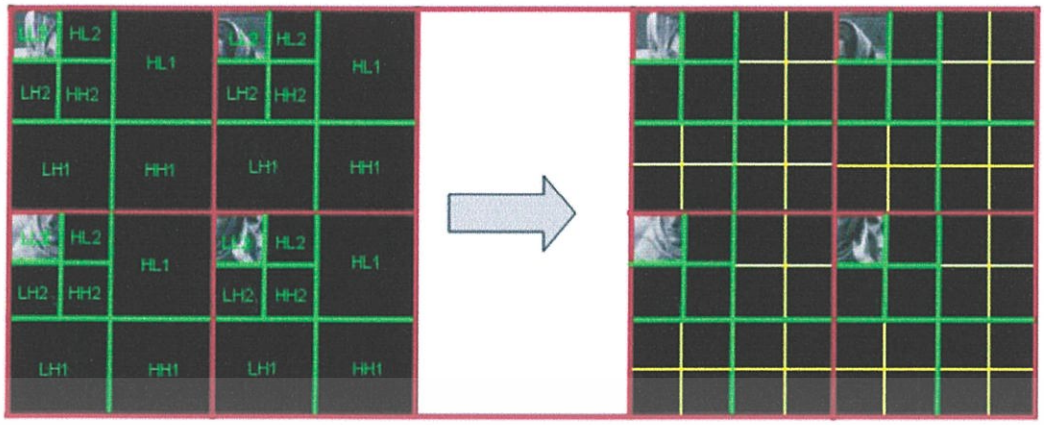
3.3 กระบวนการในการเข้ารหัสระนาบบิต (Context Modeling)

จากทฤษฎีข้อมูล (Information Theory) กล่าวว่า "ในข้อมูลประกอบด้วยเนื้อข้อมูล และข้อมูลซ้ำซ้อน (Redundancy)" โดยปริมาณเนื้อของข้อมูล สามารถหาได้จากค่าเฉลี่ยของความน่าจะเป็นของการพบสัญลักษณ์ต่างๆ ในข้อมูล โดยค่าเฉลี่ยนี้เป็นตัวแทนของปริมาณของเนื้อข้อมูลที่สัญลักษณ์หนึ่งได้ ซึ่งเป็นที่รู้จักในเรื่องของ Entropy นั่นคือ ขอบเขตของปริมาณข้อมูลที่มีความเป็นไปได้ที่จะบีบอัดข้อมูล ซึ่งเป็นที่นิยมในการวัดประสิทธิภาพในการบีบอัดข้อมูล

ตัวอย่างเช่น ข้อมูลชุด A_1 ประกอบด้วยสัญลักษณ์ {0, 1, 2, 3} โดยมีข้อมูล D = 0, 1, 1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 3, 3, 3, 3 ดังนั้นเมื่อความน่าจะเป็นของแต่ละสัญลักษณ์ในชุดข้อมูล A_1 จากลำดับข้อมูลจริง D ได้ดังนี้ p(0) = 0.05, p(1) = 1.10, p(2) = 0.20 และ p(3) = 0.65 เมื่อนำไปหาค่าเฉลี่ยปริมาณเนื้อข้อมูลต่อสัญลักษณ์ด้วยวิธีการคำนวณค่า Entropy จะได้ค่า Entropy = 1.42 บิตต่อสัญลักษณ์ ถ้านำเสนอข้อมูลของชุด A_1 ใหม่ด้วยวิธีการพิจารณาผลต่างของค่าปัจจุบันและค่าที่อยู่ตำแหน่งก่อนหน้า จะได้รูปแบบของข้อมูล D = 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 1, 0, 0, 0 ดังนั้น สัญลักษณ์ที่ปรากฏในกลุ่มข้อมูล A_2 = {-1, 0, 1} และค่าความน่าจะเป็นของการเกิดสัญลักษณ์ในชุด A_2 มีค่าดังนี้ p(-1) = 0.05, p(1) = 0.2 และ p(0) = 0.75 เมื่อนำไปวัดประสิทธิภาพด้วยวิธีการ Entropy จะได้ค่า Entropy = 0.992 บิตต่อสัญลักษณ์ จะเห็นได้ว่าความหลากหลายของรูปแบบสัญลักษณ์ในการแทนข้อมูลมีผลต่อขอบเขตของข้อมูลที่สามารถบีบอัด เมื่อความหลากหลายของข้อมูลลดลง ทำให้ขอบเขตการบีบอัดข้อมูลลดลง ส่งผลให้สามารถบีบอัดข้อมูลได้ปริมาณข้อมูลเฉลี่ยลดลงได้

จากตัวอย่างที่ได้ยกมาจะเห็นได้ว่าหากข้อมูลรูปภาพใน 1 จุดมีขนาดทั้งสิ้น 8 บิต รูปแบบที่เป็นไปได้ในการนำเสนอสามารถมีได้ถึง 2^8 หรือ 256 สัญลักษณ์ ดังนั้นในการประมวลผลของส่วนการเข้ารหัสระนาบบิตนี้เป็นการลดรูปแบบของข้อมูลรูปภาพ จาก 256 สัญลักษณ์ ลงเหลือเพียง 38 รูปแบบสัญลักษณ์ ส่งผลให้สามารถบีบอัดข้อมูลได้มากขึ้น

กระบวนการในการเข้ารหัสระนาบบิตเป็นขั้นตอนการเข้ารหัสที่สำคัญในส่วนของการเข้ารหัส Tier-1 ซึ่งดำเนินการหลังจากการแปลงสัญญาณด้วยเทคนิคการแปลงเวฟเลท จุดประสงค์หลักในการเข้ารหัสระนาบบิตนี้ เพื่อลดระดับของสัมประสิทธิ์เวฟเลท หรือ เป็นการลดจำนวนสัญลักษณ์ในการบีบอัด เพื่อส่งต่อไปยังส่วนการเข้ารหัส MQ-coder เพื่อทำการบีบอัดเพิ่มเติม การลดจำนวนของสัญญาณที่จะทำการบีบอัดนี้ ส่งผลให้ค่า Entropy ของข้อมูลลดลง ทำให้สามารถบีบอัดข้อมูลได้ด้วยอัตราการบีบอัดที่สูงขึ้น ในขั้นตอนนี้สัมประสิทธิ์เวฟเลทจากแต่ละ Subband ของแต่ละ Tile จะถูกแบ่งส่วนให้อยู่ในรูปของ Code-block ดังแสดงในรูปที่ 3.7 โดยข้อมูลแต่ละ Code-block นี้จะถูกส่งต่อไปเพื่อประมวลผลการเข้ารหัสระนาบบิตต่อไป



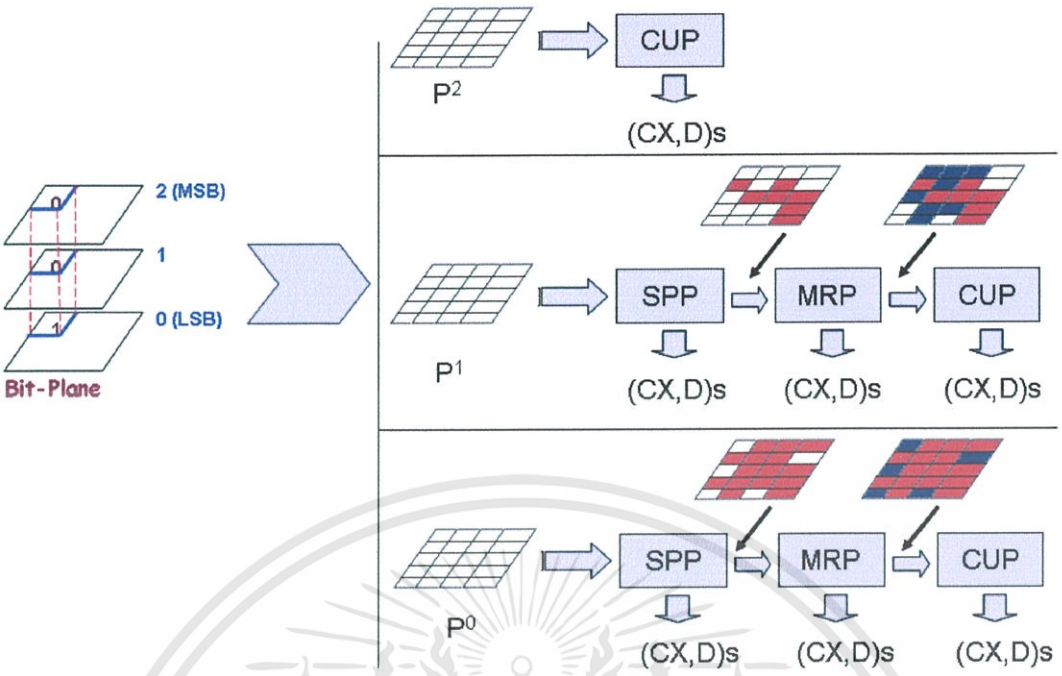
รูปที่ 3.7 แสดงตัวอย่างภาพภายหลังการแบ่งภาพออกเป็น Code-block ย่อยๆ

3.3.1 การแบ่งข้อมูลสำหรับการเข้ารหัสระนาบบิต

ตลอดช่วงเวลาในการเข้ารหัสระนาบบิต ข้อมูลสัมประสิทธิ์เวฟเลทในแต่ละ Subband จะถูกแบ่งเป็น Code-block โดยขนาดของแต่ละ Code-block จะมีขนาดความกว้างและความสูงที่เป็นกำลังของสอง (2^n) ขนาดของ Code-block ที่เล็กที่สุดสามารถมีขนาด 4x4 จุดภาพ และขนาดที่ใหญ่ที่สุดสามารถมีขนาด 1024x1024 จุดภาพ ถ้าให้ 2^x แทนขนาดของความกว้างและ 2^y แทนขนาดของความสูง ในแต่ละ Code-block ค่าของ $x + y$ ถูกจำกัดไว้ให้ไม่เกิน 12 จากข้อกำหนดนี้ทำให้โดยทั่วไปขนาดของแต่ละ Code-block จะอยู่ที่ขนาด 64 x 64 จุดภาพหรือ 32 x 32 จุดภาพ

3.3.2 การเข้ารหัสระนาบบิตใน JPEG2000

การเข้ารหัสระนาบบิตใน JPEG2000 ถูกนำเสนอโดย David S. Taubman [1] ซึ่งใช้อัลกอริธึม EBCOT ซึ่งอัลกอริธึมนี้ใช้ความสามารถของความสมดุลและความซ้ำซ้อนที่เกี่ยวข้องระหว่างระนาบบิตให้มีค่าที่น้อยที่สุด เพื่อจัดการและลดขนาดของสายข้อมูลที่จะถูกสร้างในส่วนของ BAC (Binary Arithmetic Coding) อัลกอริธึม EBCOT จะเข้ารหัสข้อมูลแต่ละระนาบบิตโดยผ่านข้อมูลไปประมวลผลใน 3 เส้นทาง (3 Pass) ซึ่งข้อมูลในตำแหน่งใดในระนาบบิตใดถูกเข้ารหัสในเส้นทางใดเส้นทางหนึ่งจาก 3 เส้นทางที่กำหนดแล้ว ข้อมูลในตำแหน่งนั้นจะไม่ถูกนำมาเข้ารหัสอีก ด้วยวิธีการดังที่ได้กล่าวจึงเป็นเหตุให้วิธีการนี้ถูกเรียกว่า “การเข้ารหัสระนาบบิตแบบปลีกย่อย (Fractional Bit-plane Coding)” เช่นเดียวกัน สัญลักษณ์ที่ได้จากการประมวลผลผ่านเส้นทางทั้ง 3 เส้นทางแล้ว จะถูกส่งไปทำการเข้ารหัสในส่วนของ MQ-coder ต่อไป เพื่อทำการบีบอัดข้อมูลเพิ่มเติมดังแสดงในรูปที่ 3.8



รูปที่ 3.8 แสดงลำดับการเข้ารหัสระนาบบิต

โดยการเข้ารหัสทั้ง 3 เส้นทาง ในการเข้ารหัสระนาบบิต เป็นดังต่อไปนี้

เส้นทางที่ 1 - SPP (Significant Propagation Pass): เป็นการตรวจสอบและเข้ารหัสบิตข้อมูลตำแหน่งที่มีความสำคัญ ซึ่งสังเกตได้จากค่าสถานะนัยสำคัญในตำแหน่งที่สนใจเปรียบเทียบกับค่าสถานะนัยสำคัญรอบข้าง หากค่าสถานะนัยสำคัญในตำแหน่งที่สนใจมีความสำคัญน้อยกว่ารอบข้างให้ถือว่าเป็นตำแหน่งจะถูกเข้ารหัสในพาสนี้

เส้นทางที่ 2 - MRP (Magnitude Refinement Pass): เป็นการตรวจสอบและเข้ารหัสบิตข้อมูลตำแหน่งที่สำคัญ โดยดูจากค่าสถานะนัยสำคัญในตำแหน่งที่สนใจซึ่งเป็นตำแหน่งที่เคยถูกเข้ารหัสของระนาบบิตก่อนหน้าในเส้นทาง SPP มาก่อน แต่ในระนาบบิตปัจจุบันไม่ได้ถูกเข้ารหัสในช่วงการประมวลผล SPP บิตข้อมูลในตำแหน่งนั้นจะถูกเข้ารหัสในช่วงการประมวลผลนี้

เส้นทางที่ 3 - CUP (Cleanup Pass): เป็นการตรวจสอบและเข้ารหัสบิตข้อมูลตำแหน่งใดที่ไม่ถูกเข้ารหัสในช่วงการประมวลผล SPP และ MRP ข้อมูลตำแหน่งดังกล่าวจะถูกเข้ารหัสในช่วงการประมวลผลนี้

3.3.3 คำอธิบายของศัพท์ที่เกี่ยวข้องในการเข้ารหัสระนาบบิต

Code-block (y): คืออาร์เรย์ 2 มิติของข้อมูลจำนวนเต็ม โดยแต่ละ Code-block ถูกกำหนดขนาดของความกว้างและความสูง ค่าของข้อมูลจำนวนเต็มสามารถเป็นไปได้ทั้งค่าบวก ค่า

ศูนย์ และค่าลบ ซึ่งค่านี้เป็นค่าของสัมประสิทธิ์เวฟเลทที่รับเข้ามาประมวลผลในการเข้ารหัสระนาบบิต

Sign Array (x): คืออาร์เรย์ 2 มิติของข้อมูลที่แสดงเครื่องหมาย ของค่าสัมประสิทธิ์เวฟเลท โดยค่าข้อมูลส่วนนี้นำไปใช้ในการเข้ารหัสระนาบบิต สำหรับค่าข้อมูลส่วนนี้สามารถหาได้จากสมการด้านล่าง

$$x[m,n]=\begin{cases}1 & if\ y[m,n]<0\\ 0 & otherwise\end{cases}$$

ในขณะที่อ้างอิงถึงข้อมูลในตำแหน่งที่นอกเหนือจากขอบเขตของ Code-block ข้อมูลที่แสดงเครื่องหมายนี้จะถูกกำหนดให้มีค่าเป็น '0' หรือเครื่องหมายของข้อมูลที่อยู่นอกขอบเขตของ Code-block มีค่าเป็นเครื่องหมายบวกนั่นเอง

Magnitude Array (v): เป็นอาร์เรย์ 2 มิติที่แสดงค่าของข้อมูลสัมประสิทธิ์เวฟเลทที่ไม่มีการคิดเครื่องหมายของข้อมูล หรือเฉพาะค่าของขนาด (Magnitude) ของสัมประสิทธิ์เวฟเลทนั่นเอง ค่าของขนาดนี้ สามารถหาได้จาก $v[m,n]=|y[m,n]|$ หากมีการอ้างถึงข้อมูลในส่วนนี้ที่อยู่นอกขอบเขตของ Code-block ค่าข้อมูลในตำแหน่งที่อยู่นอกขอบเขตนั้นจะมีค่าเป็น '0' สำหรับค่าของ $v^P[m,n]$ ใช้แสดงค่าของข้อมูลในตำแหน่งระนาบบิตที่ P ของค่าข้อมูล $v[m,n]$

ค่าตัวแปรสถานะของการเข้ารหัสระนาบบิต σ, σ' และ η : ค่าตัวแปรสถานะของการเข้ารหัสระนาบบิตที่เกี่ยวข้องในการเข้ารหัส ประกอบไปด้วยค่าตัวแปรสถานะ 3 ตัวแปร โดยแต่ละตัวแปรเป็นอาร์เรย์ 2 มิติที่มีขนาดเท่ากับขนาดของ Code-block ซึ่งค่าตัวแปรสถานะของการเข้ารหัสแต่ละตัวเป็นไปตามนี้

ค่า σ (Significant State): ค่าตัวแปรสถานะตำแหน่งที่มีนัยสำคัญของการเข้ารหัส ค่าของข้อมูลสถานะนัยสำคัญนี้มีขนาดเท่ากับขนาดของ Code-block และมีค่าเริ่มต้นเป็น '0' คือข้อมูลทุกตำแหน่งใน Code-block ยังไม่แสดงค่านัยสำคัญ ณ จุดเริ่มต้นของการเข้ารหัส โดยค่า $\sigma[m,n]=1$ หรือสถานะของการเข้ารหัสในตำแหน่งนี้แสดงการเกิดค่านัยสำคัญ จะเกิดขึ้นเมื่อข้อมูลที่ทำการเข้ารหัสมีค่าไม่เป็น '0' ในระนาบบิตแรกของข้อมูล $v[m,n]$ หากการเข้ารหัสมีการอ้างถึงข้อมูลสถานะนี้ในตำแหน่งที่อยู่นอกขอบเขตของ Code-block ค่าตัวแปรสถานะของการเข้ารหัสที่แสดงนัยสำคัญจะมีค่าเป็น '0' ค่าตัวแปรสถานะนัยสำคัญนี้จะทำการเก็บสถานะนัยสำคัญสำหรับการเข้ารหัสในทุกระนาบบิต และจะมีการล้างค่าเมื่อทำการเข้ารหัสระนาบบิตทุกระนาบใน Code-block เสร็จสิ้นแล้ว

ค่า σ' (Magnitude Refinement State): ค่าตัวแปรสถานะของการเข้ารหัสแบบ Magnitude Refinement มีขนาดเท่ากับขนาดของ Code-block และมีค่าเริ่มต้นเป็น '0' โดยข้อมูลใน

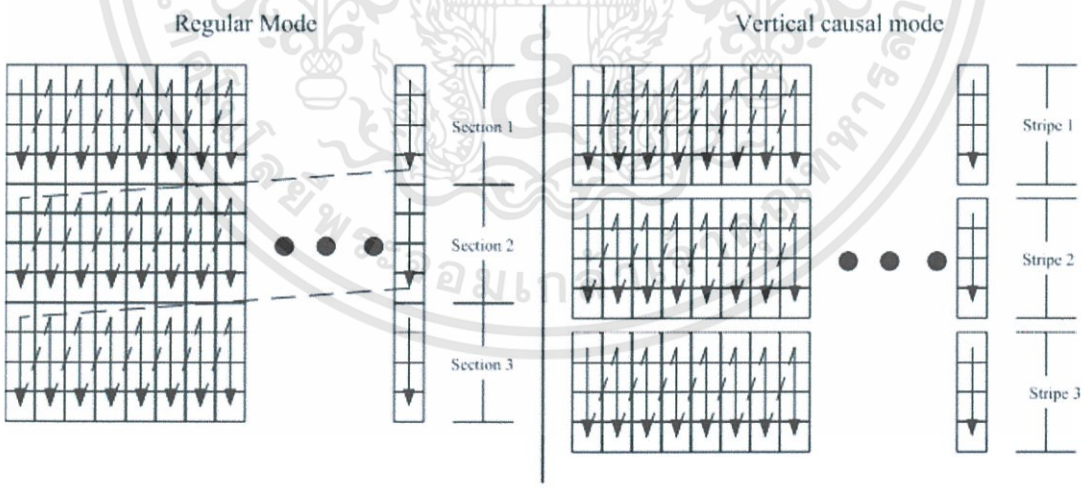
ตำแหน่งใดที่ผ่านการประมวลผลจากตัวดำเนินการพื้นฐานการเข้ารหัสแบบ Magnitude Refinement แล้วจะทำให้ค่าสถานะ $\sigma'[m,n]=1$ ค่าตัวแปรสถานะของการเข้ารหัสแบบ Magnitude Refinement นี้จะทำการเก็บสถานะของการเข้ารหัสแบบ Magnitude Refinement สำหรับการเข้ารหัสในทุกระนาบบิต และจะมีการล้างค่าเมื่อทำการเข้ารหัสระนาบบิตทุกระนาบใน Code-block เสร็จสิ้นแล้ว เช่นเดียวกับค่าตัวแปรสถานะนัยสำคัญ

ค่า η (Coding State): ค่าตัวแปรสถานะของการเข้ารหัส ใช้เป็นตัวบ่งบอกว่าข้อมูลตำแหน่งใดในระนาบบิตได้ถูกเข้ารหัสไปแล้วบ้าง หรือข้อมูลในตำแหน่งใดยังไม่ถูกเข้ารหัส ตัวแปรสถานะของการเข้ารหัสนี้จะเก็บสถานะเฉพาะแต่ละระนาบบิต เมื่อทำการประมวลผลระนาบบิตใดเสร็จสิ้นแล้ว ค่าตัวแปรสถานะของการเข้ารหัสนี้จะถูกล้างค่าก่อนที่จะนำไปติดตามสถานะของการเข้ารหัสของระนาบบิตต่อไป

รูปแบบการไล่ลำดับการเข้ารหัสของข้อมูล: ตามมาตรฐานการบีบอัดสัญญาณรูปภาพ JPEG2000 จะแบ่งรูปแบบการไล่ลำดับของการเข้ารหัสข้อมูลทั้ง Code-block ออกเป็น 2 รูปแบบ รูปแบบการไล่ลำดับการเข้ารหัสของข้อมูลสามารถแสดงได้ดังรูปที่ 3.9

<u>รูปแบบแรก</u> เป็นการแบ่งข้อมูลออกเป็น ส่วนๆ (Section) โดยที่แต่ละส่วนมีจำนวน 4 แถว และมีการประมวลผลแบบต่อเนื่อง

<u>รูปแบบที่ 2</u> เป็นการแบ่งข้อมูลออกเป็น Stripe ซึ่งแต่ละ Stripe จะแยกกันประมวลผลแบบไม่ต่อเนื่อง



รูปที่ 3.9 แสดงรูปแบบการไล่ลำดับการเข้ารหัสของข้อมูล

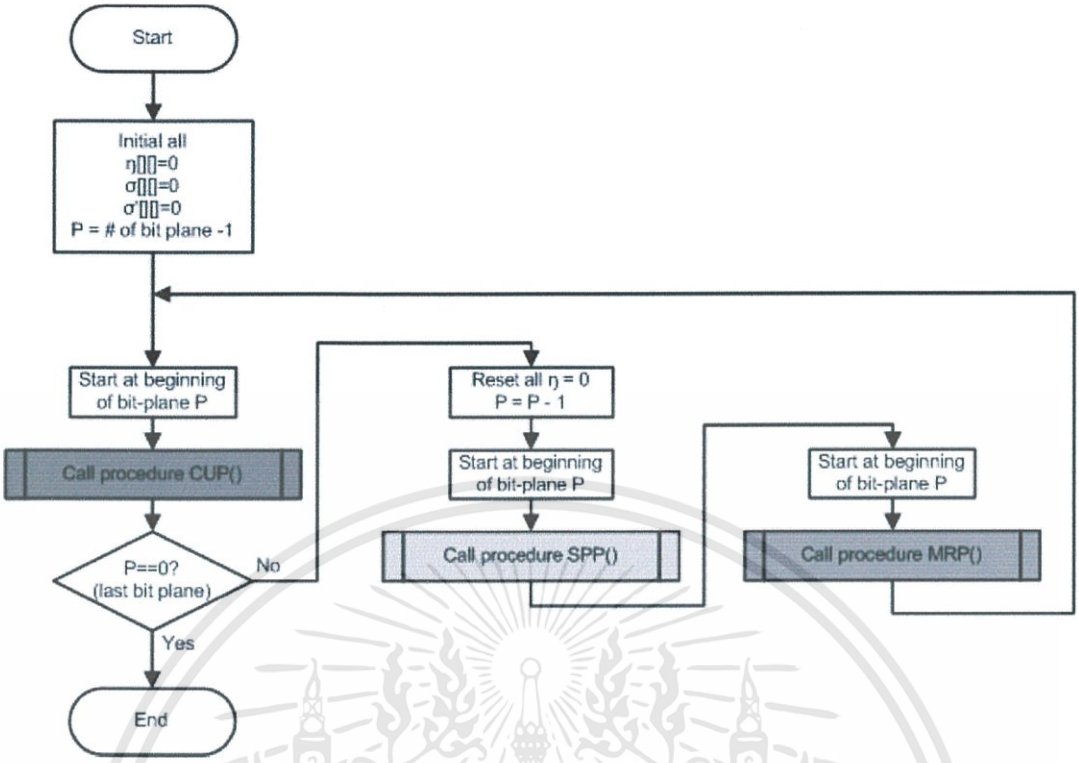
3.3.4 การเข้ารหัสในแต่ละพาส

ในการประมวลผลการเข้ารหัสระนาบบิต จะประกอบไปด้วยการเข้ารหัสที่แยกออกเป็นส่วนย่อยๆ 3 ส่วน โดยส่วนย่อยๆ เหล่านี้ถูกเรียกว่าพาสซึ่งข้อมูลในตำแหน่งใดก็ตามถูกเข้ารหัสหนึ่งในสามพาสแล้ว ข้อมูลในตำแหน่งดังกล่าวจะไม่ถูกเข้ารหัสซ้ำในพาสอื่นอีก พาสการทำงานทั้ง 3 ที่อยู่ในการเข้ารหัสระนาบบิต มีดังต่อไปในคือ SPP, MRP และ CUP

จากรูปที่ 3.10 แสดงผังการทำงานของการเข้ารหัสระนาบบิต เมื่อเริ่มต้นการประมวลผลข้อมูลของ Code-block ค่าสถานะของการเข้ารหัสอันได้แก่ ตัวแปรสถานะนัยสำคัญ (σ), ตัวแปรสถานะ Magnitude Refinement (σ') และ ตัวแปรสถานะของการเข้ารหัส (η) จะถูกตั้งค่าเริ่มต้นให้มีค่าเป็น 'ศูนย์' (0) ซึ่งตัวแปรสถานะของการเข้ารหัสทั้ง 3 จะมีขนาดพื้นที่ที่ใช้ในการเก็บสถานะเท่ากับขนาดของ Code-block ที่กำลังถูกประมวลผล นั่นคือโดยทั่วไปอยู่ที่ 64 แถว x 64 คอลัมน์ ในส่วนของความกว้างของข้อมูลสถานะในแต่ละตำแหน่งมีความกว้างเพียง 1 บิต หรือมีค่าเพียง 'ศูนย์' (0) กับ 'หนึ่ง' (1) เพื่อบ่งบอกสถานะของการเข้ารหัส

ข้อมูลในแต่ละระนาบบิตจะถูกประมวลผลในพาสทั้ง 3 เริ่มต้นจาก SPP ต่อมาเป็น MRP และสุดท้ายของการประมวลผลข้อมูลจะถูกประมวลผลที่ CUP แต่จากรูปที่ 3.10 จะเห็นได้ว่าข้อมูลในส่วนของระนาบบิตที่มีนัยสำคัญมากที่สุดกลับถูกประมวลผลในส่วนของ CUP เพียงพาสเดียวแทนที่จะเป็น 3 พาสเนื่องจากว่าค่าตัวแปรสถานะของการเข้ารหัสในตอนเริ่มต้นการประมวลผลมีค่าเป็นศูนย์ทั้งหมด จึงไม่มีตำแหน่งใดที่ค่าสถานะนัยสำคัญ (σ) และ ค่าสถานะ Magnitude Refinement (σ') เป็น 1 ทำให้ไม่ตรงกับเงื่อนไขของการประมวลผลใน SPP และ MRP ทำให้ข้อมูลในระนาบบิตที่มีนัยสำคัญมากที่สุดของแต่ละ Code-block ถูกประมวลผลใน CUP เพียงอย่างเดียว

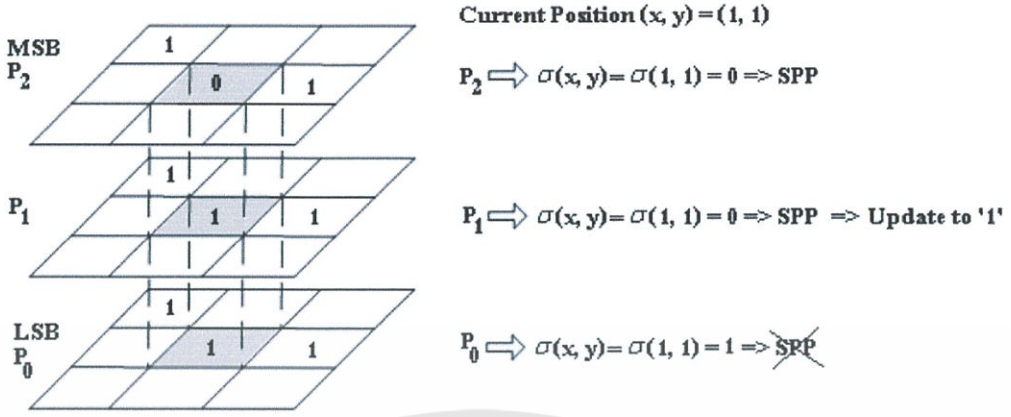
เมื่อข้อมูลในทุกตำแหน่งได้ผ่านการประมวลผลครบทั้งระนาบบิตแล้ว ข้อมูลในระนาบบิตถัดไปจะถูกส่งเข้ามาประมวลผลเรียงลำดับจาก SPP, MRP, และ ปิดท้ายด้วย CUP จนกว่าระนาบบิตของข้อมูลในระนาบบิตสุดท้ายถูกประมวลผล ก็เป็นอันสิ้นสุดการเข้ารหัสระนาบบิตของข้อมูลใน Code-block นั้น จากนั้นค่าตัวแปรสถานะต่างๆจะถูกล้างค่า เพื่อเริ่มต้นกระบวนการเข้ารหัสใหม่เมื่อข้อมูลชุดใหม่ถูกส่งเข้ามาประมวลผล



รูปที่ 3.10 แผนผังการทำงานโดยรวมของการประมวลผลการเข้ารหัสระนาบบิต

3.3.4.1 การเข้ารหัสในช่วง SPP (Significant Propagation Pass)

การเข้ารหัสในช่วง SPP เป็นการเข้ารหัสที่จะคอยตรวจสอบค่าสถานะนัยสำคัญของตำแหน่งปัจจุบันที่ประมวลผลและตำแหน่งรอบๆ ข้างตำแหน่งที่กำลังประมวลผล โดยถ้าค่าสถานะนัยสำคัญของตำแหน่งที่กำลังถูกประมวลผลมีค่าเป็น '0' และค่าสถานะนัยสำคัญของตำแหน่งรอบข้างตำแหน่งใดตำแหน่งหนึ่งมีค่าเป็น '1' นั่นหมายความว่าสถานะนัยสำคัญของตำแหน่งปัจจุบันมีความสำคัญน้อยกว่าตำแหน่งรอบข้าง เมื่อเกิดเงื่อนไขดังนี้ขึ้น ถือว่าข้อมูลในตำแหน่งปัจจุบันนี้จะถูกประมวลผลใน SPP และเมื่อใดก็ตามที่ค่าสถานะนัยสำคัญของตำแหน่งปัจจุบันที่กำลังประมวลผลในระนาบบิตที่ต่ำกว่าทำให้เกิดการเปลี่ยนแปลงค่าสถานะนัยสำคัญมีค่าเป็น '1' หรือค่าสถานะนัยสำคัญของตำแหน่งที่กำลังประมวลผลมีความสำคัญเทียบเท่ากับตำแหน่งรอบข้าง จะถือว่าตำแหน่งปัจจุบันนี้จะถูกเข้ารหัสใน SPP เป็นระนาบบิตสุดท้าย ตำแหน่งเดียวกันกับตำแหน่งปัจจุบันในระนาบบิตถัดต่อไปด้านล่าง ไปจนถึงระนาบบิต LSB (Least Significant Bit) จะไม่ถูกประมวลผลในช่วง SPP อีกต่อไป หรืออาจจะกล่าวได้ว่า การประมวลผลของการเข้ารหัสในช่วง SPP จะประมวลผลข้อมูลในตำแหน่งเดิมไปเรื่อยๆ จนกว่าจะมีระนาบบิตใดทำให้ค่าสถานะนัยสำคัญของตำแหน่งดังกล่าวมีความสำคัญเทียบเท่าตำแหน่งรอบข้าง การประมวลผลใน SPP ก็จะสิ้นสุดลง สามารถแสดงได้ดังรูปที่ 3.11

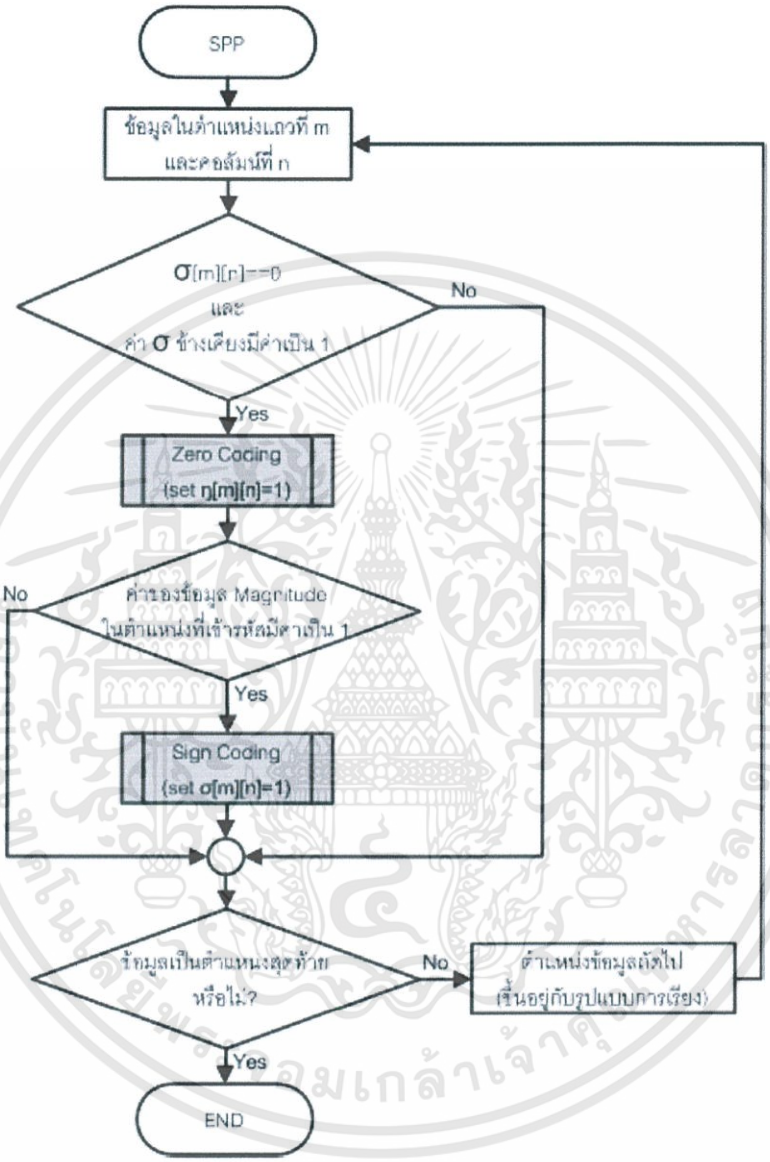


รูปที่ 3.11 ตัวอย่างค่าสถานะนัยสำคัญของข้อมูลที่ประมวลผลใน SPP ของข้อมูลในตำแหน่งเดียวกันในแต่ละระนาบบิต

จากรูปที่ 3.11 จะเห็นได้ว่าค่าสถานะนัยสำคัญของข้อมูลตำแหน่ง $(x, y) = (1, 1)$ ของระนาบบิต P_2 ซึ่งเป็นระนาบบิต MSB (Most Significant Bit)มีค่าเป็น '0' ข้อมูลในระนาบบิตนี้จะถูกประมวลผลในช่วง SPP เนื่องจากค่าสถานะนัยสำคัญของข้อมูลตำแหน่งปัจจุบันที่กำลังประมวลผลมีค่าเป็น '0' ในขณะที่ค่าสถานะนัยสำคัญของข้อมูลตำแหน่งรอบๆ ข้างมีค่าเป็น '1' เงื่อนไขของข้อมูลในตำแหน่งปัจจุบันนี้ไม่ทำให้เกิดการกำหนดค่าใหม่ให้กับค่าสถานะนัยสำคัญหลังจากการประมวลผลใน SPP เสร็จแล้ว ทำให้สถานะนัยสำคัญหลังจากประมวลผล P_2 แล้วยังคงมีค่าเป็น $\sigma(1,1) = 0$ จากนั้นเมื่อทำการประมวลผลเลื่อนไปยังระนาบบิต P_1 พบว่าค่าสถานะนัยสำคัญของข้อมูลตำแหน่งปัจจุบัน ในระนาบบิต P_1 มีค่าเป็น '0' ในขณะที่ค่าสถานะนัยสำคัญของข้อมูลตำแหน่งรอบๆ ข้างมีค่าเป็น '1' จึงยังคงถูกประมวลผลในช่วง SPP แต่เงื่อนไขของข้อมูลตำแหน่งปัจจุบันในระนาบบิต P_1 นี้ส่งผลให้เกิดการกำหนดค่าให้กับตัวแปรสถานะนัยสำคัญใหม่หลังจากการประมวลผลในช่วง SPP แล้วเสร็จ เป็น $\sigma(1,1) = 1$ และเมื่อค่าสถานะนัยสำคัญของข้อมูลในตำแหน่งปัจจุบันของระนาบบิต P_0 เข้ามาในช่วง SPP จะเห็นได้ว่าค่าสถานะนัยสำคัญของตำแหน่งที่กำลังประมวลผลมีค่าความสำคัญเทียบเท่ากับตำแหน่งรอบข้าง ดังนั้นข้อมูลในตำแหน่งที่กำลังประมวลผลนี้จึงไม่ถูกเข้ารหัสในช่วง SPP

สำหรับรูปที่ 3.12 แสดงแผนผังการทำงานของการเข้ารหัสในช่วง SPP โดยเริ่มต้นจากการตรวจสอบค่าสถานะนัยสำคัญของตำแหน่งที่กำลังถูกประมวลผลพร้อมทั้งตำแหน่งรอบข้าง ว่าตรงตามเงื่อนไขของการเข้ารหัส SPP หรือไม่ หากเป็นไปตามเงื่อนไข ข้อมูลในตำแหน่งที่กำลังถูกประมวลผลจะเข้ารหัสข้อมูลด้วยตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์ (Zero Coding) ภายหลังจากนั้นข้อมูลขนาดที่ได้เข้ารหัสศูนย์ไปแล้วจะถูกตรวจสอบว่ามีค่าเป็น '1' หรือไม่ หากมีค่าเป็น

'1' แล้วตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมาย (Sign Coding) จะถูกนำมาประมวลผลกับข้อมูลตำแหน่งนั้นด้วย สำหรับค่าสถานะนัยสำคัญจะถูกปรับค่าในช่วงการเข้ารหัสเครื่องหมายในช่วง SPP



รูปที่ 3.12 แผนผังแสดงการประมวลผลในช่วง SPP

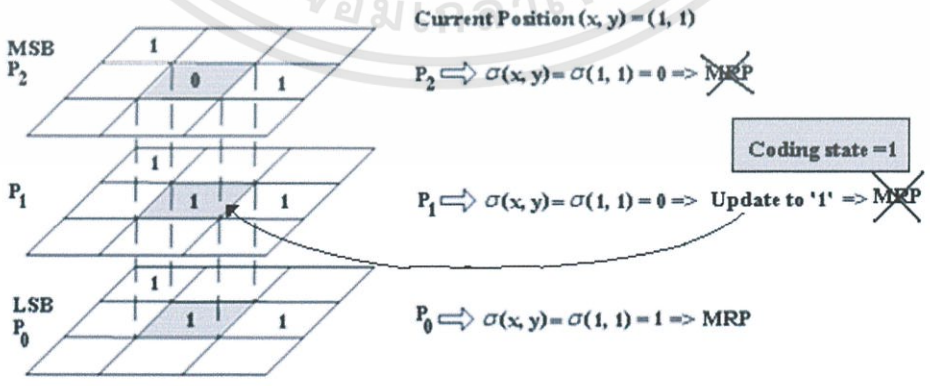
3.3.4.2 การเข้ารหัสในช่วง MRP (Magnitude Refinement Pass)

การเข้ารหัสในช่วง MRP เป็นการเข้ารหัสลำดับที่สองต่อจากการเข้ารหัสในช่วง SPP การเข้ารหัสในช่วง MRP จะเป็นการเข้ารหัสโดยการกรองเอาเฉพาะข้อมูลขนาด ในตำแหน่งที่กำลัง

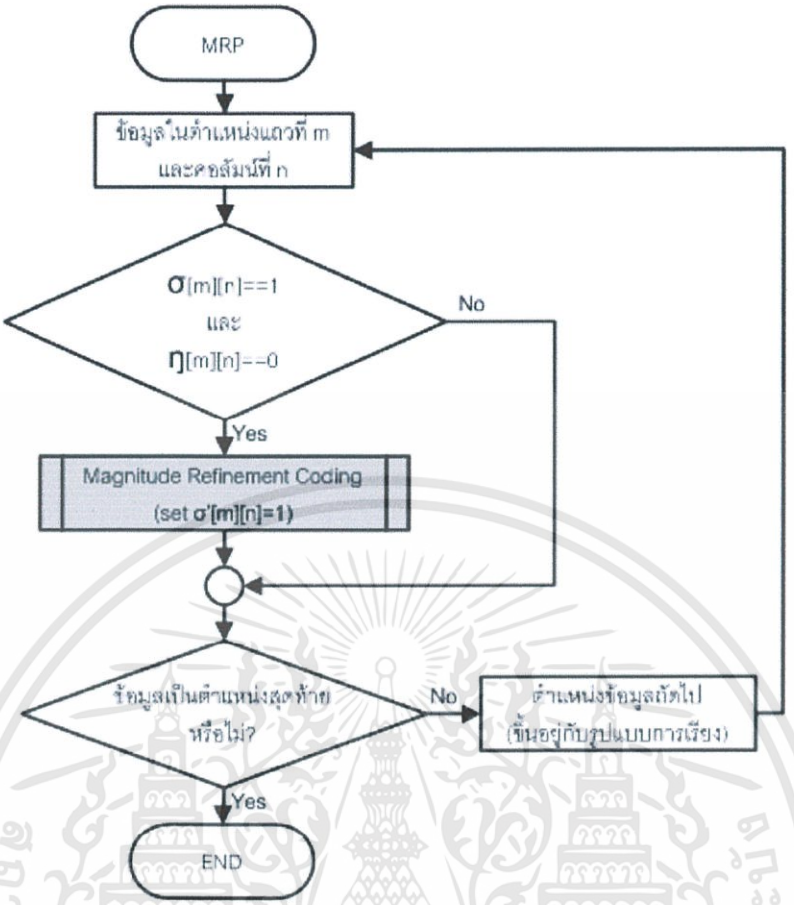
ถูกประมวลผลส่งออกเป็นผลลัพธ์ ซึ่งเงื่อนไขในการเข้ารหัส MRP เมื่อค่าสถานะนัยสำคัญในตำแหน่งปัจจุบันที่กำลังจะถูกประมวลผลจะมีค่าเป็นหนึ่ง หรือข้อมูลในตำแหน่งที่กำลังจะถูกประมวลผลมีนัยความสำคัญ

ในรูปที่ 3.13 แสดงตัวอย่างของการเข้ารหัสในช่วง MRP ของข้อมูลในตำแหน่งเดียวกันในทุกระนาบบิต จะเห็นว่าค่าสถานะนัยสำคัญของข้อมูลในระนาบบิต P_2 มีค่าเป็น '0' ในตำแหน่งที่เป็นสีเทา ซึ่งจะไม่ถูกประมวลผลการเข้ารหัสในช่วง MRP เช่นเดียวกับระนาบบิต P_1 เนื่องจากว่าค่าสถานะนัยสำคัญในตำแหน่งที่เป็นสีเทามีค่าเป็น '0' จึงทำให้ตำแหน่งนี้ถูกประมวลผลในช่วง SPP แต่ภายหลังจากการเข้ารหัสในช่วง SPP แล้ว ค่าสถานะนัยสำคัญของข้อมูลระนาบบิต P_1 จะถูกปรับค่าให้เป็น '1' แต่ข้อมูลในตำแหน่งดังกล่าวจะไม่ถูกประมวลผลใน MRP เนื่องจากว่าข้อมูลในตำแหน่งนั้นจะไม่ถูกประมวลผลซ้ำซ้อน (Non-overlap) และค่าจากการที่ค่าสถานการณ์เข้ารหัสมีค่าเป็น '1' นั่นเอง ดังนั้นจึงส่งผลให้ข้อมูลในระนาบบิต P_0 ถูกประมวลผลในช่วง MRP เนื่องจากค่าสถานะนัยสำคัญของข้อมูลเป็น '1' หรือ ข้อมูลตำแหน่งนั้นมีนัยสำคัญเกิดขึ้นในระนาบบิตก่อนหน้าแล้ว ซึ่งตรงตามเงื่อนไขของการเข้ารหัสในช่วง MRP ที่กำหนด

ในส่วนรูปที่ 3.14 แสดงผังงานในการทำงานของการเข้ารหัสในช่วง MRP จะเห็นว่าในการเข้ารหัสในช่วง MRP นี้ จะมีตัวดำเนินการพื้นฐานที่เกี่ยวข้องเพียงตัวเดียวคือ ตัวดำเนินการพื้นฐานแบบ Magnitude Refinement โดยเมื่อเริ่มการประมวลผลในการเข้ารหัสในช่วง MRP ค่าสถานะนัยสำคัญของข้อมูลที่กำลังถูกประมวลผลจะถูกตรวจสอบว่าเป็นไปตามเงื่อนไขของการเข้ารหัส MRP หรือไม่ หากค่าสถานะนัยสำคัญของตำแหน่งดังกล่าวมีค่าเป็นหนึ่งและยังไม่ได้เข้ารหัสในส่วนของ SPP ก่อนหน้า ข้อมูลในตำแหน่งดังกล่าวจะถูกประมวลผลด้วยตัวดำเนินการพื้นฐานแบบ Magnitude Refinement และทำการกำหนดค่าสถานะของการเข้ารหัสแบบ Magnitude Refinement ให้เป็น '1' จึงถือเป็นการสิ้นสุดกระบวนการในการเข้ารหัสในช่วง MRP ในระนาบบิตนั้นๆ



รูปที่ 3.13 แสดงตัวอย่างของข้อมูลสถานะนัยสำคัญที่ตรงตามเงื่อนไขการเข้ารหัสในช่วง MRP



รูปที่ 3.14 แผนผังแสดงการประมวลผลในช่วง MRP

3.3.4.3 การเข้ารหัสในช่วง Cleanup Pass (CUP)

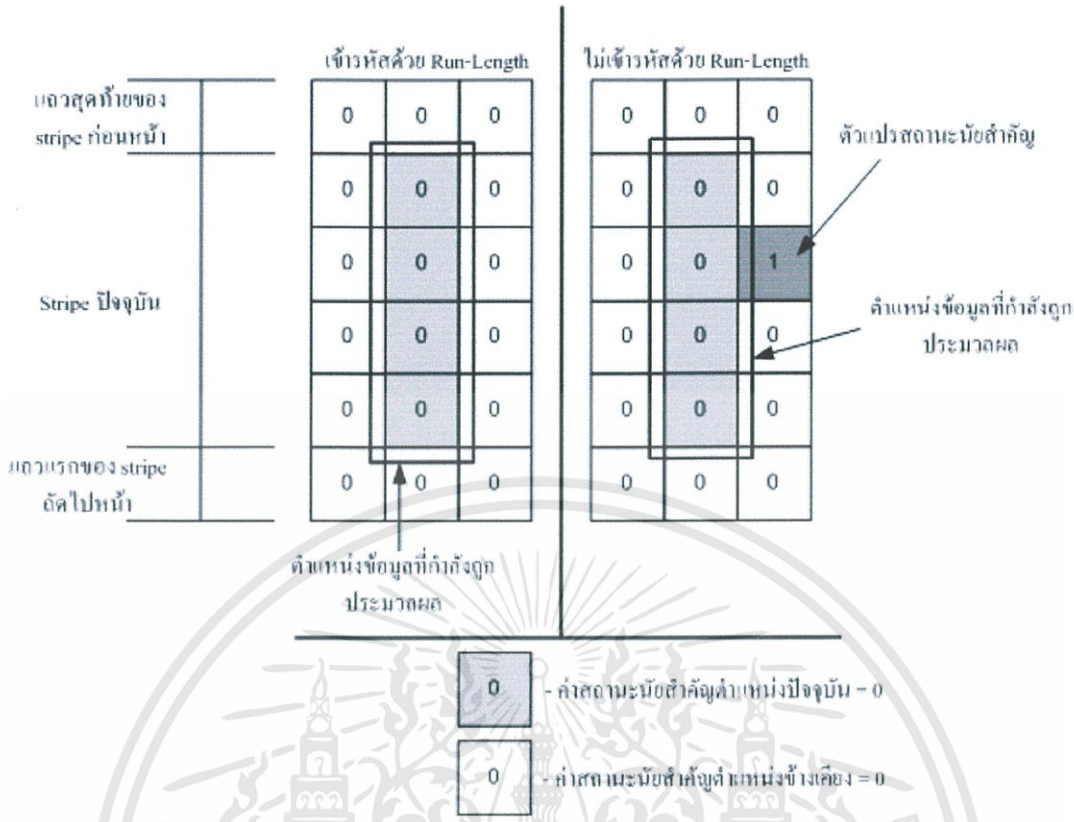
สำหรับการเข้ารหัสระนาบบิตในช่วง CUP เป็นการเข้ารหัสในลำดับสุดท้ายของการเข้ารหัสระนาบบิต ซึ่งหากข้อมูลในตำแหน่งใดไม่ได้ถูกประมวลผลใน SPP หรือ MRP ข้อมูลในตำแหน่งเหล่านั้นจะถูกประมวลผลใน CUP เพื่อเป็นการรับประกันว่าข้อมูลในทุกตำแหน่งของข้อมูลในระนาบบิตได้ถูกเข้ารหัส จะเห็นว่าข้อมูลในตำแหน่งที่จะถูกประมวลผลในช่วง CUP นี้จะเป็นข้อมูลในตำแหน่งที่ไม่มีนัยสำคัญทั้งที่ตำแหน่งของตัวเองและตำแหน่งรอบข้าง การเข้ารหัสในช่วง CUP จะเห็นได้ว่าตัวดำเนินการพื้นฐานการเข้ารหัสมีเหมือนกับการเข้ารหัสในช่วง SPP แต่ในการเข้ารหัส CUP จะมีตัวดำเนินการพื้นฐานการเข้ารหัสแบบ Run-length เพิ่มเข้ามา

เมื่อเริ่มการประมวลผล ค่าตัวแปรสถานะของการเข้ารหัสจะถูกตรวจสอบเพื่อพิจารณาว่าข้อมูลในตำแหน่งดังกล่าวถูกประมวลผลในช่วงก่อนหน้าหรือยัง ถ้ายังไม่ได้ถูกประมวลผลข้อมูลในตำแหน่งนั้นจะถูกประมวลผลการเข้ารหัสในช่วง CUP นี้ หลังจากนั้นตำแหน่งของข้อมูลจะถูกตรวจสอบว่า ตำแหน่งที่กำลังถูกประมวลผลเป็นตำแหน่งของแถวแรกในคอลัมน์ของ Stripe

หรือไม่ หากตำแหน่งของข้อมูลเป็นแถวแรก และค่าสถานะนัยสำคัญของทั้งคอลัมน์ของ Stripe พร้อมทั้งในตำแหน่งคอลัมน์รอบข้างที่กำลังถูกประมวลผล มีค่าเป็นศูนย์หรือไม่มีนัยสำคัญ ตัวดำเนินการเข้ารหัสพื้นฐานแบบ Run-length จะถูกนำมาประมวลผลข้อมูลทั้งคอลัมน์ Stripe เนื่องจากถือว่าเกิดการวิ่งต่อเนื่องของค่าศูนย์ (Run Length of Zeros) ในคอลัมน์ที่กำลังเข้ารหัสและคอลัมน์ข้างเคียง หากเงื่อนไขในการเลือกใช้ตัวดำเนินการพื้นฐานแบบ Run-length ไม่สมบูรณ์ คือ ค่าสถานะนัยสำคัญตำแหน่งแถวแรกของคอลัมน์เป็นศูนย์จริง แต่มีค่าสถานะนัยสำคัญของตำแหน่งใดตำแหน่งหนึ่งในคอลัมน์ หรือ ตำแหน่งข้างเคียงไม่เป็น '0' ข้อมูลในคอลัมน์นี้จะถูกเข้ารหัสด้วยตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์ (ZC) และหลังจากทำการเข้ารหัสด้วยตัวดำเนินการพื้นฐานแบบ Run-length หรือการเข้ารหัสศูนย์ แล้วข้อมูลจะถูกเข้ารหัสด้วยตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมายต่อไป สำหรับตัวอย่างของค่าสถานะนัยสำคัญในการตรวจสอบเงื่อนไขของตัวดำเนินการพื้นฐานการเข้ารหัสแบบ Run-length สามารถแสดงได้ดังรูปที่ 3.15

3.3.5 ตัวดำเนินการขั้นพื้นฐานในการเข้ารหัส

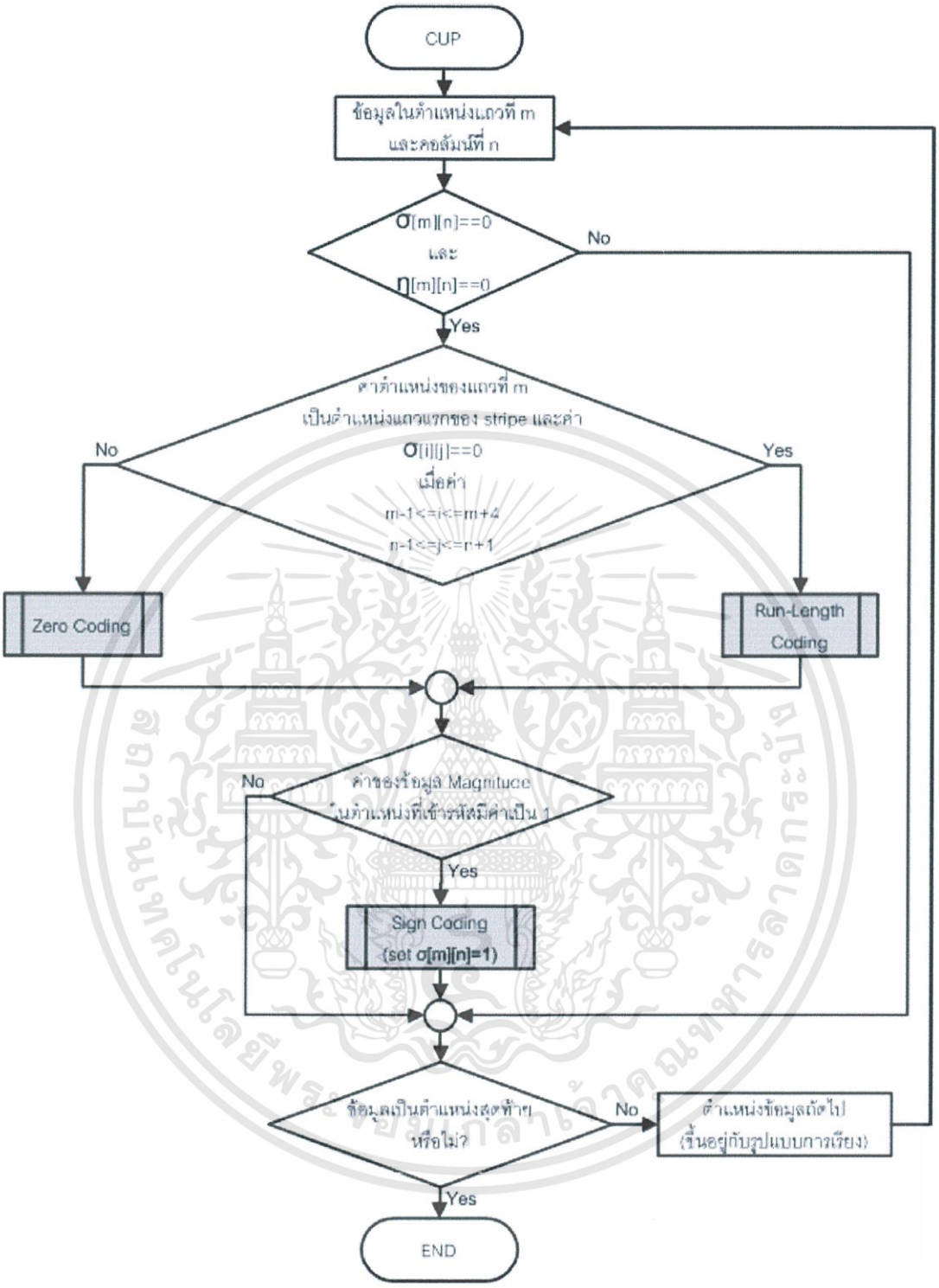
ในการเข้ารหัสระนาบบิตของกระบวนการบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 มีตัวดำเนินการพื้นฐานที่ใช้ในการเข้ารหัสทั้งหมด 4 ตัว ได้แก่ ตัวดำเนินการเข้ารหัสศูนย์ (Zero Coding) ตัวดำเนินการเข้ารหัสค่าเครื่องหมาย (Sign Coding) ตัวดำเนินการเข้ารหัส Magnitude Refinement (Magnitude Refinement Coding) และตัวดำเนินการเข้ารหัสการวิ่งต่อเนื่องของศูนย์ (Run-length Coding) โดยจากการเข้ารหัสด้วยตัวดำเนินการพื้นฐานต่างๆ เหล่านี้จะได้ผลลัพธ์ในรูปของคู่อันดับสัญลักษณ์ Context (CX) กับ Decision (D) เพื่อนำไปเข้ารหัสเพื่อบีบอัดต่อไปด้วยการเข้ารหัสเลขคณิต ค่า CX เป็นค่าจำนวนเต็มที่ไม่มีการคิดเครื่องหมายที่ใช้บ่งบอกสภาพแวดล้อมของข้อมูลในตำแหน่งที่กำลังประมวลผล ส่วนค่า D เป็นค่าที่มีค่าเพียง 0 กับ 1 ความแตกต่างของคู่อันดับสัญลักษณ์มีจำนวนทั้งสิ้น 19 ค่า (0-18) ซึ่งตัวดำเนินการพื้นฐานในการเข้ารหัส มีส่วนรายละเอียดดังต่อไปนี้



รูปที่ 3.15 แสดงลักษณะของตัวแปรสถานะนัยสำคัญในตำแหน่งคอลัมน์ที่กำลังประมวลผล

3.3.5.1 ตัวดำเนินการเข้ารหัสศูนย์ (Zero Coding)

ในส่วนการเข้ารหัสศูนย์ (ZC: Zero Coding) เป็นหนึ่งในตัวดำเนินการพื้นฐานในการเข้ารหัส โดยผลลัพธ์ของการเข้ารหัสจะได้ค่า D มีค่าเท่ากับค่าข้อมูลในตำแหน่งที่เข้ารหัสปัจจุบัน คือ ตำแหน่งแถวที่ m คอลัมน์ที่ n บนระนาบบิต P ($v^P[m,n]$) และค่าของ CX จะเป็นค่าที่สัมพันธ์กับค่าผลรวมของค่าสถานะนัยสำคัญในตำแหน่ง 8 ตำแหน่งรอบข้างตำแหน่งที่เข้ารหัส ตามทิศแนวตั้ง แนวนอน และแนวทแยงมุม การหาค่าผลรวมค่าสถานะนัยสำคัญในทิศทางต่างๆ สามารถแสดงได้ดังรูปที่ 3.17 ค่า CX ที่เป็นผลลัพธ์จากการเข้ารหัสศูนย์ จะมีค่าอยู่ในช่วง 0 – 8 โดยแบ่งพิจารณาค่าตามข้อมูลแต่ละ Subband (LL, LH, LH หรือ HH) ที่กำลังประมวลผล ซึ่งค่าของ CX ในส่วนของการเข้ารหัสศูนย์ ของแต่ละตารางแบ่งได้ทั้งหมด 9 รูปแบบ ดังสามารถแสดงความสัมพันธ์ของ CX กับค่าผลรวมของค่าสถานะนัยสำคัญ ได้ตามตารางที่ 3.1 และตารางที่ 3.2 สำหรับค่า X ในตารางทั้งสามบ่งบอกถึงค่าข้อมูลที่ไม่ต้องสนใจ (Don't Care)



รูปที่ 3.16 แผนผังแสดงการประมวลผลในช่วง CUP

สำหรับการเข้ารหัสศูนย์เป็นการเข้ารหัสที่ดูสภาพแวดล้อมของค่าผลรวมค่าสถานะนัยสำคัญรอบข้างตำแหน่งข้อมูลที่กำลังถูกเข้ารหัสในทิศทางแนวนอน แนวตั้ง และทางด้านแนว

ทแยงมุม ว่ามีลักษณะเป็นแบบใด ซึ่งค่าผลรวมค่าสถานะนัยสำคัญรอบข้างที่เกี่ยวข้องกับข้อมูลที่กำลังเข้ารหัสจะมีการให้ความสำคัญของรูปแบบไม่เท่ากันในแต่ละทิศทาง โดยในกรณีที่ค่าผลรวมของค่าสถานะนัยสำคัญมีค่ามากในทิศทางที่สัมพันธ์กับทิศของข้อมูลสัมประสิทธิ์เวฟเลท ค่า CX ผลลัพธ์จะมีค่ามาก แต่ในทางกลับกันถ้าค่าผลรวมค่าสถานะนัยสำคัญมีค่าน้อย ทิศที่สัมพันธ์กับทิศของข้อมูลสัมประสิทธิ์เวฟเลท ค่า CX ผลลัพธ์จะมีค่าน้อยลงตามลำดับความสำคัญ เช่น หากค่าข้อมูลสัมประสิทธิ์เวฟเลทที่เข้ามาประมวลผลเป็นข้อมูลใน Subband LL หรือ LH ซึ่งข้อมูลในส่วนนี้จะมีความเด่นชัดข้อมูลในทิศทางแนวนอนมากกว่าในทิศทางอื่น ดังนั้นค่าของสถานะนัยสำคัญรอบข้างของข้อมูลที่กำลังประมวลผลใน Subband LL หรือ LH จึงให้ความสำคัญกับทิศทางแนวนอน โดยค่าของ CX จะขึ้นกับค่าผลรวมของค่าสถานะนัยสำคัญในทิศแนวนอนเป็นสำคัญ ดังแสดงในตารางที่ 3.1

ตัวอย่างเช่น

กรณีที่ 1: ค่าผลรวมค่าสถานะนัยสำคัญในทิศแนวนอนมีค่าเท่ากับ ‘2’ คือ มีนัยสำคัญทั้ง 2 ตำแหน่งในแนวนอนรอบตำแหน่งที่เข้ารหัส แสดงว่าความสำคัญทิศในแนวนอนเด่นชัดกว่าทิศอื่นอย่างเห็นได้ชัด ดังนั้นค่า CX จึงมีค่าสูงสุด คือ CX = 8

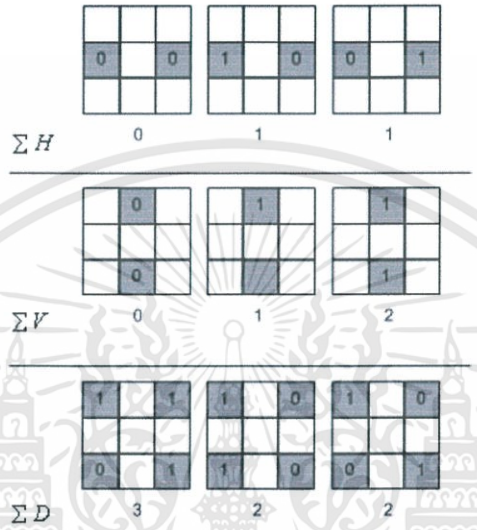
กรณีที่ 2: ค่าผลรวมค่าสถานะนัยสำคัญในทิศแนวนอนมีค่าลดลงเป็น ‘1’ ส่งผลให้ค่า CX จะมีค่าลดลงอยู่ในช่วง 5 – 7 ส่วนจะลดลงเป็นเท่าใด ต้องดูค่าผลรวมค่าสถานะนัยสำคัญในทิศแนวตั้งและแนวทแยงมุมประกอบด้วย เช่น

- ถ้าผลรวมค่าสถานะนัยสำคัญในทิศแนวตั้งมีค่ามากกว่าหรือเท่ากับ 1 แสดงว่ามีความเด่นชัดในทิศแนวตั้ง ค่า CX จะลดลงเป็น CX = 7
- ถ้าผลรวมค่าสถานะนัยสำคัญในทิศแนวตั้งมีค่าเท่ากับ ‘0’ ส่วนทิศแนวทแยงมุมมีค่ามากกว่าหรือเท่ากับ ‘1’ แสดงว่ามีความเด่นชัดในทิศแนวทแยงมุม ค่า CX จะลดลงเป็น CX = 6

กรณีที่ 3: ค่าผลรวมค่าสถานะนัยสำคัญในทิศแนวนอนมีค่าลดลงเป็น ‘0’ ส่งผลให้ค่า CX จะมีค่าลดลงอยู่ในช่วง 0 – 4 ส่วนจะลดลงเป็นเท่าใด ต้องดูค่าผลรวมค่าสถานะนัยสำคัญในทิศแนวตั้งและแนวทแยงมุมประกอบด้วย ซึ่งแนวทางของค่า CX ที่ลดลงเป็นเช่นเดียวกับในกรณีที่ 2

ในส่วนของข้อมูลสัมประสิทธิ์เวฟเลทของ Subband HL ข้อมูลที่จะเกิดเด่นชัดใน Subband นี้ จะเป็นข้อมูลที่อยู่ในทิศทางแนวตั้ง ดังนั้นทิศทางของค่าผลรวมค่าสถานะนัยสำคัญรอบ

ข้างจึงให้ความสำคัญกับทิศทางแนวตั้ง ส่วน Subband HH ซึ่งเป็น Subband สุดท้ายของข้อมูลสัมประสิทธิ์เวฟเลทจะมีข้อมูลเด่นชัดในทิศทางแนวทแยงมุม ค่าของสถานะนัยสำคัญของการเข้ารหัสศูนย์ก็ให้ความสำคัญในทิศทางแนวทแยงมุมเช่นเดียวกัน การเปลี่ยนแปลงของค่า CX ผลลัพธ์นั้นขึ้นกับความสำคัญของผลรวมค่าสถานะนัยสำคัญในทิศที่สัมพันธ์กับข้อมูลเช่นเดียวกับกรณีของข้อมูลสัมประสิทธิ์เวฟเลทของ Subband LL และ LH



รูปที่ 3.17 แสดงการหาค่าสถานะนัยสำคัญในแต่ละทิศทางที่เกี่ยวข้องกับการเข้ารหัสรหัสศูนย์

ตารางที่ 3.1 ตารางความสัมพันธ์ของค่า CX กับค่าสถานะนัยสำคัญในส่วนของ Subband LLและ LH

LL and LH Sub Bands			Context Label
$\sum H$	$\sum V$	$\sum D$	CX
2	X	X	8
1	≥ 1	X	7
1	0	≥ 1	6
1	0	0	5
0	2	X	4
0	1	X	3
0	0	≥ 2	2
0	0	1	1
0	0	0	0

ตารางที่ 3.2 แสดงตารางความสัมพันธ์ของค่า CX กับค่าสถานะนัยสำคัญในส่วนของ Subband HL

HL Sub Bands			Context Label
$\sum V$	$\sum H$	$\sum D$	CX
2	X	1	8
1	≥ 1	X	7
1	0	≥ 1	6
1	0	0	5
0	2	X	4
0	1	X	3
0	0	≥ 2	2
0	0	1	1
0	0	0	0

ตารางที่ 3.3 แสดงตารางความสัมพันธ์ของค่า CX กับค่าสถานะนัยสำคัญในส่วนของ Subband HH

HH Sub Bands		Context Label
$\sum D$	$\sum(H+V)$	CX
≥ 3	X	8
2	≥ 1	7
2	0	6
1	≥ 2	5
1	1	4
1	0	3
0	≥ 2	2
0	1	1
0	0	0

3.3.5.2 ตัวดำเนินการเข้ารหัสค่าเครื่องหมาย (Sign Coding)

สำหรับตัวดำเนินการพื้นฐานในส่วนนี้เป็นการเข้ารหัสเครื่องหมายที่บ่งบอกสภาพแวดล้อมข้างเคียงของตำแหน่งที่กำลังประมวลผลว่าข้อมูลมีเครื่องหมายใดและนำผลของสภาพแวดล้อมข้างเคียงเหล่านั้นมาทดสอบเครื่องหมายของข้อมูลตำแหน่งที่กำลังประมวลผล ซึ่งค่า

ของสภาพแวดล้อมข้างเคียงของเครื่องหมายจะแบ่งความสัมพันธ์ในทิศทางแนวตั้งและทิศทางแนวนอน รวมไปถึงค่าสถานะนัยสำคัญของข้อมูลข้างเคียงที่มีความสัมพันธ์เหล่านั้นด้วย

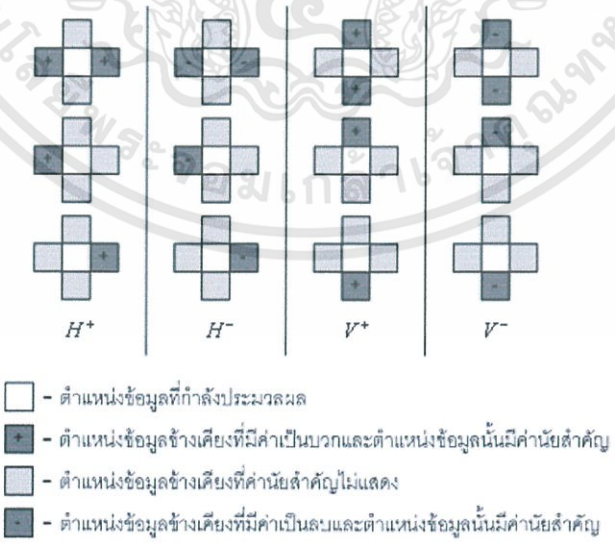
ค่าสถานะเครื่องหมายในตำแหน่งข้างเคียงที่แทนทิศทางแนวนอน คือ 'H' ส่วนค่าสถานะเครื่องหมายในตำแหน่งข้างเคียงในแนวทิศทางแนวตั้งแทนได้ด้วย 'V' ซึ่งค่าสถานะเครื่องหมายในตำแหน่งข้างเคียงในการเข้ารหัสเครื่องหมายสามารถมีค่าสถานะได้ 3 ค่าคือ สถานะบวก '+', สถานะลบ '-' และสถานะกลางคือไม่เป็นทั้งค่าบวกและลบ '0'

ดังนั้นค่าสถานะเครื่องหมายในการเข้ารหัสเครื่องหมายสามารถแสดงได้ดังนี้ H^+ หรือ V^+ แทนค่าสถานะเครื่องหมายที่เป็นค่าบวก, H^- หรือ V^- แทนค่าสถานะเครื่องหมายที่เป็นค่าลบ และ H^0 หรือ V^0 แทนค่าสถานะเครื่องหมายเป็นกลางที่ไม่เป็นทั้งบวกและลบ

ค่าสถานะเครื่องหมายมีความสัมพันธ์กับค่าสถานะนัยสำคัญในทิศที่พิจารณา และค่าเครื่องหมายในตำแหน่งที่สัมพันธ์กัน ดังแสดงความสัมพันธ์ในรูปที่ 3.18โดย

ค่าสถานะเครื่องหมายเป็นสถานะบวก แสดงว่ามีค่าสถานะนัยสำคัญในทิศที่พิจารณาเป็น '1' อย่างน้อย 1 ตำแหน่ง และค่าเครื่องหมายที่สัมพันธ์กับตำแหน่งที่ค่าสถานะนัยสำคัญเป็น '1' มีค่าเป็นบวกหรือมีค่าเป็น '0' ซึ่งจะทำให้มีค่าสถานะเครื่องหมายเป็นสถานะบวกอย่างน้อยหนึ่งตำแหน่งในทิศที่พิจารณา

ค่าสถานะเครื่องหมายจะเป็นสถานะลบ ในกรณีที่มีค่าสถานะนัยสำคัญในทิศที่พิจารณาเป็น '1' อย่างน้อย 1 ตำแหน่ง และค่าเครื่องหมายที่สัมพันธ์กับตำแหน่งที่ค่าสถานะนัยสำคัญเป็น '1' มีค่าเป็นลบหรือมีค่าเป็น '1' ซึ่งจะทำให้มีค่าสถานะเครื่องหมายเป็นสถานะลบอย่างน้อยหนึ่งตำแหน่งในทิศที่พิจารณา



รูปที่ 3.18 แสดงรูปแบบของค่าสถานะเครื่องหมายในตำแหน่งข้างเคียง

ดังนั้นการหาค่าสถานะเครื่องหมายในทิศทางแนวนอน พิจารณาความสัมพันธ์ของค่าสถานะนัยสำคัญในแนวนอนในตำแหน่งข้างเคียงตำแหน่งที่เข้ารหัส ในตำแหน่ง σ_{10} และ σ_{12} และค่าเครื่องหมายในตำแหน่งที่สัมพันธ์กัน χ_{10} และ χ_{12} ตามความสัมพันธ์ดังนี้

ค่าสถานะเครื่องหมายในทิศแนวนอน (H)

สถานะบวก

$$H^{+} = \sigma_{10} \bullet \sigma_{12} \bullet \overline{\chi}_{10} \bullet \overline{\chi}_{12} + \overline{\sigma}_{10} \bullet \sigma_{12} \bullet \overline{\chi}_{12} + \sigma_{10} \bullet \overline{\sigma}_{12} \bullet \overline{\chi}_{10}$$

สถานะลบ

$$H^{-} = \sigma_{10} \bullet \sigma_{12} \bullet \chi_{10} \bullet \chi_{12} + \overline{\sigma}_{10} \bullet \sigma_{12} \bullet \chi_{12} + \sigma_{10} \bullet \overline{\sigma}_{12} \bullet \chi_{10}$$

สถานะกลาง

$$H^{0} = \overline{H}^{+} \bullet \overline{H}^{-}$$

การหาค่าสถานะเครื่องหมายในทิศทางแนวตั้ง พิจารณาความสัมพันธ์ของค่าสถานะนัยสำคัญในแนวตั้งในตำแหน่งข้างเคียงตำแหน่งที่เข้ารหัส ในตำแหน่ง σ_{01} และ σ_{21} และค่าเครื่องหมายในตำแหน่งที่สัมพันธ์กัน χ_{01} และ χ_{21} ตามความสัมพันธ์ดังนี้

ค่าสถานะเครื่องหมายในทิศแนวตั้ง (V)

สถานะบวก

$$V^{+} = \sigma_{01} \bullet \sigma_{21} \bullet \overline{\chi}_{01} \bullet \overline{\chi}_{21} + \overline{\sigma}_{01} \bullet \sigma_{21} \bullet \overline{\chi}_{21} + \sigma_{01} \bullet \overline{\sigma}_{21} \bullet \overline{\chi}_{01}$$

สถานะลบ

$$V^{-} = \sigma_{01} \bullet \sigma_{21} \bullet \chi_{01} \bullet \chi_{21} + \overline{\sigma}_{01} \bullet \sigma_{21} \bullet \chi_{21} + \sigma_{01} \bullet \overline{\sigma}_{21} \bullet \chi_{01}$$

สถานะกลาง

$$V^{0} = \overline{V}^{+} \bullet \overline{V}^{-}$$

จากรูปที่ 3.18 แสดงรูปแบบที่เป็นไปได้ของการบ่งบอกค่าเครื่องหมายในตำแหน่งข้อมูลข้างเคียงว่ามีค่าสถานะเครื่องหมายเป็นค่าบวก ค่าลบ หรือค่าสถานะเครื่องหมายเป็นกลางที่ไม่เป็นทั้งบวกและลบ ซึ่งตามมาตรฐานการบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 ในส่วนของการเข้ารหัสเครื่องหมายแสดงได้ดังตารางที่ 3.4 สำหรับค่าสถานะเครื่องหมายในคอลัมน์ H และV ใช้บ่งบอกค่าสถานะเครื่องหมายของข้อมูลข้างเคียงในทิศแนวนอนและแนวตั้ง ซึ่งค่า ‘1’ แทนค่าสถานะเครื่องหมายในทิศทางแนวที่พิจารณา ที่มีค่าสถานะบวก, ‘0’ แทนค่าสถานะเครื่องหมายในทิศทางแนวที่พิจารณา ที่มีค่าสถานะเป็นกลาง ส่วนค่า -1 แทนค่าเครื่องหมายในทิศทางแนวที่พิจารณา ที่มีค่าสถานะเป็นลบ โดยค่าสถานะเครื่องหมายจะมีผลต่อค่าสัญลักษณ์ผลลัพธ์การ

เข้ารหัส (CX) นั่นคือค่า CX จะขึ้นอยู่กับทิศทางของค่าสถานะเครื่องหมาย เช่น กรณีที่ค่าสถานะเครื่องหมายในแนวตั้งมีสถานะเป็นกลาง (V^0) แต่ค่าสถานะเครื่องหมายในแนวนอนมีค่าเป็นบวก (H^+) หรือ เป็นลบ(H^-) จะทำให้ค่า CX = 12 หรือ กรณีที่ค่าสถานะเครื่องหมายในแนวตั้งมีสถานะไม่เป็นกลางคือมีสถานะเป็นบวก (V^+) หรือเป็นลบ (V^-) เช่นเดียวกับค่าสถานะเครื่องหมายในแนวนอนมีสถานะไม่เป็นกลางคือมีสถานะเป็นบวก (H^+) หรือเป็นลบ (H^-) จะทำให้ค่า CX = 11 เป็นต้น

ในส่วนของค่า Decision (D) ของการเข้ารหัสเครื่องหมายสามารถหาได้จากการนำค่าเครื่องหมายของข้อมูลในตำแหน่งที่กำลังประมวลผล (χ_{11}) มาทำการ XOR กับค่า $\hat{\chi}$ ที่ได้จากตารางที่ 3.4 สามารถเขียนได้ดังสมการด้านล่าง

$$D = \chi_{11} \oplus \hat{\chi}$$

ในส่วนของค่า $\hat{\chi}$ เป็นค่าที่มีความสัมพันธ์กับค่าสถานะเครื่องหมายของตำแหน่งรอบข้าง โดยสามารถเปิดค่าได้จากตารางที่อยู่ในตารางที่ 3.4 โดยค่า $\hat{\chi}$ จะเป็นไปในแนวทางที่ส่งผลให้ค่า D แสดงว่าค่าเครื่องหมายในตำแหน่งปัจจุบัน เป็นไปในแนวทางเดียวกับค่าสถานะเครื่องหมายของตำแหน่งรอบข้างหรือไม่ โดยที่ค่า D จะเป็น '0' ถ้าเครื่องหมายในตำแหน่งปัจจุบันมีค่าเหมือนกับค่าสถานะเครื่องหมายของตำแหน่งรอบข้าง ในทางตรงข้าม ค่า D จะเป็น '1' ถ้าเครื่องหมายในตำแหน่งปัจจุบันมีค่าแตกต่างกับค่าสถานะเครื่องหมายของตำแหน่งรอบข้าง ตัวอย่างเช่น ถ้าค่าสถานะเครื่องหมายในแนวตั้งมีสถานะเป็นกลาง (V^0) แต่ค่าสถานะเครื่องหมายในแนวนอนมีค่าเป็นบวก(H^+) กรณีนี้ จะทำให้ค่า CX = 12 ดังแสดงข้างต้น เมื่อเปิดตารางพบว่าเงื่อนไขนี้ เปิดค่า $\hat{\chi}$ = 0 ดังนั้น ถ้าค่าเครื่องหมายของตำแหน่งปัจจุบันเป็นบวก (χ_{11} = 0) เช่นเดียวกับค่าสถานะเครื่องหมายของตำแหน่งรอบข้าง จะทำให้ค่า D มีค่าเป็น '0' แต่ในทางกลับกัน ถ้าค่าเครื่องหมายของตำแหน่งปัจจุบันเป็นลบ (χ_{11} = 1) ซึ่งเป็นเครื่องหมายที่แตกต่างไปจากค่าสถานะเครื่องหมายของตำแหน่งรอบข้าง จะทำให้ค่า D มีค่าเป็น '1' เป็นต้น

3.3.5.3 ตัวดำเนินการพื้นฐานการเข้ารหัส Magnitude Refinement

สำหรับตัวดำเนินการพื้นฐานของการเข้ารหัส Magnitude Refinement จะเป็นตัวดำเนินการที่จะประมวลผลเฉพาะในช่วงการประมวลผล MRP เท่านั้น โดยค่า D เป็นค่าที่ได้จากค่าข้อมูลในตำแหน่งที่กำลังถูกประมวลผล คือตำแหน่งแถวที่ m คอลัมน์ที่ n บนระนาบบิต P ($\nu^P[m,n]$) ในส่วนของค่า CX เป็นค่าที่ขึ้นอยู่กับค่าของสถานะของการเข้ารหัส Magnitude Refinement $\sigma'[m,n]$ และผลรวมของค่าสถานะนัยสำคัญ 8 ตำแหน่ง ($\sigma[m-1,n-1]$,

$\sigma[m-1,n]$, $\sigma[m-1,n+1]$, $\sigma[m,n-1]$, $\sigma[m,n+1]$, $\sigma[m+1,n-1]$, $\sigma[m+1,n]$, $\sigma[m+1,n+1]$) ที่อยู่รอบตำแหน่งข้อมูลที่กำลังถูกประมวลผล โดยค่า CX ที่เป็นผลลัพธ์จากการเข้ารหัส Magnitude Refinement นั้นมีค่าอยู่ในช่วง 14 – 16 โดยค่าของ CX ที่สัมพันธ์กับเงื่อนไขของผลรวมค่าสถานะนัยสำคัญของตำแหน่งรอบข้างสามารถแสดงได้ดังตารางที่ 3.5

ตารางที่ 3.4 แสดงค่า CX ที่มีความสัมพันธ์กับค่าสถานะเครื่องหมายของข้อมูลตำแหน่งข้างเคียงตำแหน่งที่กำลังเข้ารหัสเครื่องหมาย

H	V	$\hat{\chi}$	CX	คำอธิบาย
0	0	0	9	ค่าสถานะเครื่องหมายทั้งในทิศทางแนวตั้งและทิศแนวนอนมีค่าเป็นกลาง ($CX = 9\ when\ H^0 \bullet V^0$)
0	1	0	10	ค่าสถานะเครื่องหมายในทิศทางแนวนอนมีค่าเป็นกลางและค่าสถานะเครื่องหมายในทิศทางแนวตั้งมีค่าเครื่องหมาย ($CX = 10\ when\ H^0 \bullet \lvert V \rvert$)
0	-1	1	10	
1	-1	0	11	ค่าสถานะเครื่องหมายในทิศทางแนวนอนและทิศทางแนวตั้งมีค่าเครื่องหมายที่ตรงข้ามกัน นั่นคือหากทิศทางแนวนอนมีค่าเป็นบวก ค่าเครื่องหมายในทิศทางแนวตั้งต้องเป็นเครื่องหมายลบ ($CX = 11\ when \begin{cases} H^+ \bullet V^- \\ H^- \bullet V^+ \end{cases}$)
-1	1	1	11	
1	0	0	12	ค่าสถานะเครื่องหมายในทิศทางแนวนอนมีค่าเครื่องหมาย ส่วนในทิศทางแนวตั้งมีค่าเครื่องหมายเป็นกลาง ($CX = 12\ when \lvert H \rvert \bullet V^0$)
-1	0	1	12	
1	1	0	13	ค่าสถานะเครื่องหมายของตำแหน่งรอบข้างในทั้ง 2 ทิศทาง (ทิศแนวนอนและทิศแนวตั้ง) มีค่าเครื่องหมายที่เหมือนกัน คือหากแนวนอนมีค่าเป็นบวก ค่าเครื่องหมายทางด้านแนวตั้งต้องมีค่าบวกด้วย ($CX = 13\ when \begin{cases} H^+ \bullet V^+ \\ H^- \bullet V^- \end{cases}$)
-1	-1	1	13	

ตารางที่ 3.5 แสดงค่า CX ของการเข้ารหัส Magnitude Refinement พร้อมทั้งเงื่อนไขที่เกี่ยวข้อง

$\sigma'[m,n]$	ค่าผลรวมค่าสถานะนัยสำคัญของตำแหน่งรอบข้าง	CX
1	X	16
0	≥ 1	15
0	0	14

ค่า CX ผลลัพธ์จะมีค่าสูงสุดคือ CX = 16 ในกรณีที่ตำแหน่งที่กำลังเข้ารหัส (m, n) นั้นเคยเข้ารหัสในช่วง MRP มาแล้วในระนาบบิตก่อนหน้า แต่ถ้าตำแหน่งที่กำลังเข้ารหัสยังไม่เคยเข้ารหัสในช่วง MRP มาก่อน ให้พิจารณาค่าผลรวมสถานะนัยสำคัญของตำแหน่งรอบข้างมีค่ามากกว่าหรือเท่ากับ '1' แสดงว่าตำแหน่งรอบข้างมีนัยสำคัญ จะให้ CX ผลลัพธ์ เป็น CX = 15 แต่ถ้าตำแหน่งรอบข้างไม่มีตำแหน่งใดมีนัยสำคัญเลย นั่นคือค่าผลรวมค่าสถานะนัยสำคัญมีค่าเป็น '0' จะส่งผลให้ได้ค่า CX ผลลัพธ์ค่าต่ำที่สุด คือ CX = 14

3.3.5.4 ตัวดำเนินการเข้ารหัสการวิ่งต่อเนื่องของศูนย์ (Run-length Coding)

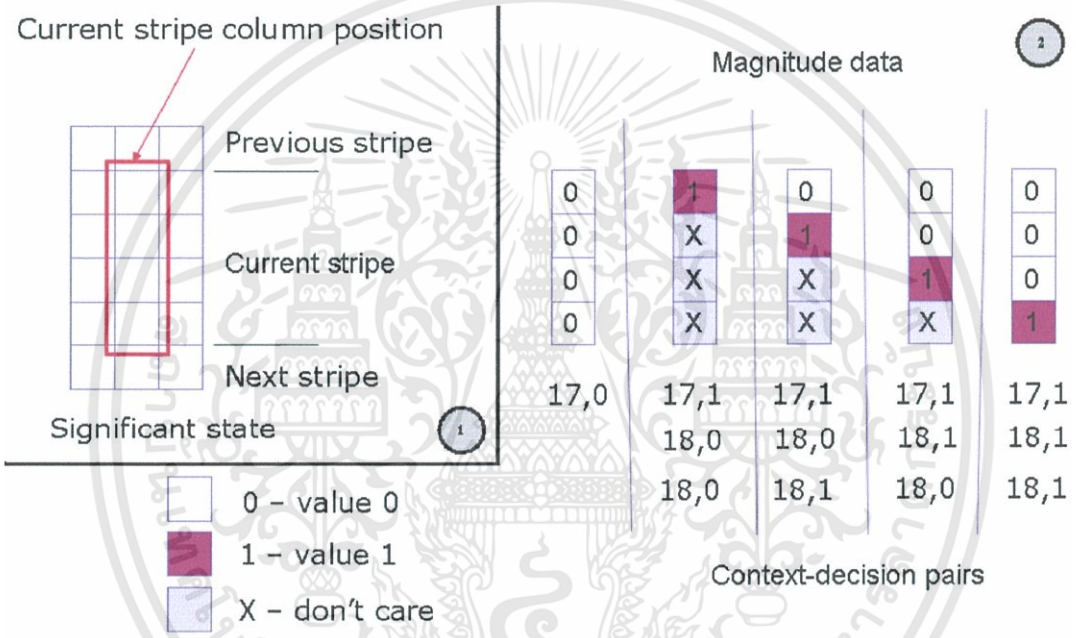
ในส่วนของตัวดำเนินการเข้ารหัสการวิ่งต่อเนื่องของศูนย์ (Run-length Coding) จะถูกประมวลผลเฉพาะในช่วง CUP เท่านั้น โดยตัวดำเนินการพื้นฐานแบบการเข้ารหัสการวิ่งต่อเนื่องของศูนย์ จะพิจารณาค่าสถานะนัยสำคัญของตำแหน่งข้อมูลในแนวคอลัมน์ที่กำลังเข้ารหัสและคอลัมน์ตำแหน่งรอบๆ ต้องมีค่าเป็น '0' ทั้งคอลัมน์ หรือค่าสถานะนัยสำคัญไม่ปรากฏ ในคอลัมน์ที่เข้ารหัส และในคอลัมน์ข้างเคียงทั้งทางด้านซ้ายและด้านขวา ถ้าคอลัมน์ข้างเคียงอยู่นอกขอบเขตของภาพ ให้ถือว่าคอลัมน์ข้างเคียงนั้นเป็นศูนย์ทั้งคอลัมน์ เมื่อพบคอลัมน์ที่ไม่มีนัยสำคัญทั้งในคอลัมน์ที่เข้ารหัสและคอลัมน์รอบข้างแล้ว คอลัมน์ที่เข้ารหัสนั้น จะถูกเข้ารหัสด้วยตัวดำเนินการเข้ารหัสการวิ่งต่อเนื่องของศูนย์ ดังแสดงในรูปที่ 3.15

สำหรับในส่วนของคู่อันดับสัญลักษณ์ CX, D ที่ถูกสร้างออกมาจากตัวดำเนินการเข้ารหัสการวิ่งต่อเนื่องของศูนย์ นี้ จะมีค่า CX เป็น 17 หรือ 18 โดยค่า CX จะบ่งบอกว่ามีการวิ่งต่อเนื่องของค่าศูนย์ของค่าข้อมูลในคอลัมน์ที่เข้ารหัสหรือไม่ และถ้ามีการเกิดการขัดจังหวะการวิ่งต่อเนื่องของค่าศูนย์ของข้อมูลจะเกิดขึ้นที่แถวใดของคอลัมน์ (แถว 0 – 3) โดยหมายเลขของ CX = 18 ในส่วนของ Decision จะใช้เป็นตัวบ่งบอกตำแหน่งแถวที่ทำให้เกิดการขัดจังหวะการวิ่งต่อเนื่องของค่าศูนย์ของข้อมูลเป็นครั้งแรก

(CX, D) = (17, 0) บ่งบอกสถานะในการวิ่งของค่าศูนย์ของข้อมูลว่า ข้อมูลมีค่าเป็น '0' ต่อเนื่องทั้งคอลัมน์ที่เข้ารหัส ไม่เกิดการขัดจังหวะการวิ่งของค่าศูนย์

(CX, D) = (17, 1) บ่งบอกสถานะในการวิ่งของค่าศูนย์ของข้อมูลว่าเกิดการขัดจังหวะของการวิ่งต่อเนื่องของค่าศูนย์ของข้อมูล และตามด้วย (CX, D) = [(18, B_1), (18, B_0)] ดังแสดงในรูปที่ 3.19

เช่นตัวอย่างแรกเกิดมีค่าข้อมูลเป็น 1 ที่ตำแหน่งแถวที่ 0 ของคอลัมน์ที่เข้ารหัส แสดงว่าเกิดการขัดจังหวะของการวิ่งต่อเนื่องของค่าศูนย์ของข้อมูล ดังนั้นจึงส่งผลลัพธ์ (CX, D) = (17, 1) ตามด้วย (CX, D) ที่ระบุตำแหน่งการเกิดค่าข้อมูลเป็น 1 คือแถวที่ 0 (“00” = 0_{10}) ทำให้ได้ (CX, D) ผลลัพธ์ตามมาอีก 2 ชุด คือ (18, 0), (18, 0) หรือ ถ้าค่าข้อมูลเป็น 1 ที่ตำแหน่งแถวที่ (“01” = 1_{10}) เป็นตำแหน่งแรก ค่า (CX, D) ผลลัพธ์ จะได้เป็นลำดับคือ (17, 1), (18, 0), (18, 1)



รูปที่ 3.19 แสดงคู่อันดับสัญลักษณ์ CX, D ของตัวดำเนินการเข้ารหัสการวิ่งต่อเนื่องของศูนย์

(1) ค่าสถานะนัยสำคัญที่เกี่ยวข้องกับตัวดำเนินการเข้ารหัสการวิ่งต่อเนื่องของศูนย์

(2) รูปแบบที่เป็นไปได้ของคู่อันดับสัญลักษณ์ CX, D ที่เกี่ยวข้องกับตัวดำเนินการเข้ารหัสการวิ่งต่อเนื่องของศูนย์

จากรูปที่ 3.19 (1) แสดงลักษณะของค่าสถานะนัยสำคัญที่ทำให้ตัวดำเนินการเข้ารหัสการวิ่งต่อเนื่องของศูนย์ประมวลผลข้อมูล จะเห็นได้ว่าค่าสถานะนัยสำคัญของตำแหน่งคอลัมน์ที่กำลังเข้ารหัสปัจจุบันทุกแถวมีค่าเป็น ‘0’ รวมทั้งค่าสถานะนัยสำคัญรอบข้างก็มีค่าเป็น ‘0’ เช่นกัน สำหรับในส่วนของรูปที่ 3.19 (2) แสดงรูปแบบของการสร้างคู่อันดับสัญลักษณ์ CX, D ในทุกรูปแบบข้อมูลที่เป็นไปได้ ของการเข้ารหัสของตัวดำเนินการเข้ารหัสการวิ่งต่อเนื่องของศูนย์ ทั้งกรณีที่ไม่มีและมีการขัดจังหวะการวิ่งต่อเนื่องของค่าศูนย์ของข้อมูลเกิดขึ้น

3.3.6 ตัวอย่างการเข้ารหัสระนาบบิต

ในส่วนหัวข้อนี้จะนำเสนอตัวอย่างการเข้ารหัสระนาบบิตโดยเป็นลำดับไป ซึ่งในที่นี้จะเป็นการเข้ารหัสระนาบบิตขนาด 4x4 จุดภาพ

ข้อมูลตัวอย่างเป็นข้อมูลขนาด 4 บิต รวมบิตที่เป็นบิตเครื่องหมาย ที่ใช้เป็นตัวอย่างในการแสดงลำดับการเข้ารหัสระนาบบิต ดังแสดงในรูปที่ 3.20 (ก) ในส่วนของค่าข้อมูลที่ไม่คิดเครื่องหมายแสดง ดังรูปที่ 3.20 (ข) และ อาร์เรย์ข้อมูลที่แสดงเครื่องหมาย แสดงดังรูปที่ 3.20 (ค)

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

(ก)

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

(ข)

0	0	0	0
1	0	0	0
1	1	1	0
0	0	0	0

(ค)

รูปที่ 3.20 แสดงข้อมูลตัวอย่างที่ใช้เข้ารหัส

(ก) ข้อมูลสัมประสิทธิ์เวฟเลท

(ข) ข้อมูลสัมประสิทธิ์เวฟเลทในส่วนที่เป็น Magnitude (ขนาดของข้อมูลที่ไม่คิดเครื่องหมาย; v)

(ค) ข้อมูลส่วนเครื่องหมายของข้อมูลสัมประสิทธิ์เวฟเลท (χ)

อาร์เรย์ข้อมูลที่ถูกแยกตามระนาบบิตที่เรียงจากระนาบที่มีนัยสำคัญมากไปจนถึงข้อมูลที่มีนัยสำคัญน้อย

0	0	0	1
0	1	0	0
1	0	0	0
0	1	0	0

v^2 (MSB)

1	0	0	0
1	1	1	0
0	0	1	1
0	1	0	1

v^1

1	0	0	1
1	1	0	1
0	1	0	1
0	0	0	0

v^0 (LSB)

รูปที่ 3.21 แสดงตัวอย่างข้อมูลขนาดที่แยกตามแต่ละระนาบบิต

ลำดับการเข้ารหัสระนาบบิตและผลลัพธ์ที่ได้จากการเข้ารหัสจะถูกแสดงโดยมีความหมายของรูปแบบในการแสดงดังนี้

<u>ตำแหน่งของข้อมูลที่ถูกเข้ารหัส</u> จะถูกแสดงในวงเล็บก้ามปู (‘[‘, ‘]’) โดยแสดงตำแหน่งแถวหลักดังนี้คือ [แถวที่, หลักที่]

<u>ผลลัพธ์คู่อันดับ</u> จะถูกแสดงในส่วนของวงเล็บปีกกา (‘{‘, ‘}’) โดยแสดงผลลัพธ์คู่อันดับดังนี้คือ {Context, Decision}

สำหรับตัวดำเนินการขั้นพื้นฐานแสดงได้ดังนี้คือ RLC (Run Length Coding), SC (Sign Coding), ZC (Zero Coding), และ MRC (Magnitude Refinement Coding)

<u>การเข้ารหัสระนาบบิตที่ 2</u>

ค่าสถานะเริ่มต้นก่อนการเข้ารหัสระนาบบิตสามารถแสดงได้ดังรูปข้างล่าง

0	0	0	1
0	1	0	0
1	0	0	0
0	1	0	0

v^2

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

σ

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

η

0	0	0	0
1	0	0	0
1	1	1	0
0	0	0	0

χ

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

σ'

การเข้ารหัสระนาบบิตในช่วง CUP ของระนาบบิตที่ 2 (v^2)

1. RLC for [0, 0] : { 17, 1}
2. RLC for [0, 0] : { 18, 1}
3. RLC for [0, 0] : { 18, 0}
4. SC for [2, 0] : { 9, 1}
5. ZC for [3, 0] : { 3, 0}
6. ZC for [0, 1] : { 0, 0}
7. ZC for [1, 1] : { 1, 1}
8. SC for [1, 1] : { 9, 0}
9. ZC for [2, 1] : { 7, 0}
10. ZC for [3, 1] : { 1, 1}
11. SC for [3, 1] : { 9, 0}
12. ZC for [0, 2] : { 1, 0}
13. ZC for [1, 2] : { 5, 0}
14. ZC for [2, 2] : { 2, 0}
15. ZC for [3, 2] : { 5, 0}

16. RLC for [0, 3] : { 17, 1}
17. RLC for [0, 3] : { 18, 0}
18. RLC for [0, 3] : { 18, 0}
19. SC for [0, 3] : { 9, 0}
20. ZC for [1, 3] : { 3, 0}
21. ZC for [2, 3] : { 0, 0}
22. ZC for [3, 3] : { 0, 0}

ค่าสถานะในการเข้ารหัสระนาบบิตภายหลังการเข้ารหัสในระนาบบิตที่ 2 ในช่วง CUP

0	0	0	1
0	1	0	0
1	0	0	0
0	1	0	0

v^2

0	0	0	1
0	1	0	0
1	0	0	0
0	1	0	0

σ

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

η

0	0	0	0
1	0	0	0
1	1	1	0
0	0	0	0

χ

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

σ'

<u>การเข้ารหัสระนาบบิตที่ 1</u>

ค่าสถานะก่อนเข้ารหัสระนาบบิตที่ 1

1	0	0	0
1	1	1	0
0	0	1	1
0	1	0	1

v^1

0	0	0	1
0	1	0	0
1	0	0	0
0	1	0	0

σ

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

η

0	0	0	0
1	0	0	0
1	1	1	0
0	0	0	0

χ

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

σ'

การเข้ารหัสระนาบบิตในช่วง SPP ของระนาบบิตที่ 1

23. ZC for [0, 0] : { 1, 1}
24. SC for [0, 0] : { 9, 0}
25. ZC for [1 ,0] : { 7, 1}
26. SC for [1, 0] : { 12, 1}
27. ZC for [3, 0] : { 7, 0}
28. ZC for [0, 1] : { 7, 0}

29. ZC for [2, 1] : { 7, 0}
30. ZC for [0, 2] : { 6, 0}
31. ZC for [1, 2] : { 6, 1}
32. SC for [1, 2] : { 12, 0}
33. ZC for [2, 2] : { 3, 1}
34. SC for [2, 2] : { 10, 1}
35. ZC for [3, 2] : { 7, 0}
36. ZC for [1, 3] : { 7, 0}
37. ZC for [2, 3] : { 6, 1}
38. SC for [2, 3] : { 12, 1}
39. ZC for [3, 3] : { 3, 1}
40. SC for [3, 3] : { 10, 0}

ค่าสถานะในการเข้ารหัสระนาบบิตภายหลังการเข้ารหัสในระนาบบิตที่ 1 ในช่วง SPP

1	0	0	0
1	1	1	0
0	0	1	1
0	1	0	1

v^1

1	0	0	1
1	1	1	0
1	0	1	1
0	1	0	1

σ

1	1	1	0
1	0	1	1
0	1	1	1
1	0	1	1

η

0	0	0	0
1	0	0	0
1	1	1	0
0	0	0	0

χ

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

σ'

การเข้ารหัสระนาบบิตในช่วง MRP ของระนาบบิตที่ 1

41. MRC for [2, 0] : { 15, 0}
42. MRC for [1, 1] : { 15, 1}
43. MRC for [3, 1] : { 15, 1}
44. MRC for [0, 3] : { 15, 0}

ค่าสถานะในการเข้ารหัสระนาบบิตภายหลังการเข้ารหัสในระนาบบิตที่ 1 ในช่วง MRP

1	0	0	0
1	1	1	0
0	0	1	1
0	1	0	1

v^1

1	0	0	1
1	1	1	0
1	0	1	1
0	1	0	1

σ

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

η

0	0	0	0
1	0	0	0
1	1	1	0
0	0	0	0

χ

0	0	0	1
0	1	0	0
1	0	0	0
0	1	0	0

σ'

การเข้ารหัสระนาบบิตในช่วง CUP ของระนาบบิตที่ 1 ไม่มีการเข้ารหัสในช่วงนี้ เนื่องจากข้อมูลถูกเข้ารหัสหมดทั้งระนาบบิตนี้แล้ว

<u>การเข้ารหัสระนาบบิตที่ 0</u>

ค่าสถานะก่อนเข้ารหัสระนาบบิตที่ 0

1	0	0	1
1	1	0	1
0	1	0	1
0	0	0	0

v^0

1	0	0	1
1	1	1	0
1	0	1	1
0	1	0	1

σ

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

η

0	0	0	0
1	0	0	0
1	1	1	0
0	0	0	0

χ

0	0	0	1
0	1	0	0
1	0	0	0
0	1	0	0

σ'

การเข้ารหัสระนาบบิตในช่วง SPP ของระนาบบิตที่ 0

45. ZC for [3, 0] : { 7, 0}
46. ZC for [0, 1] : { 7, 0}
47. ZC for [2, 1] : { 8, 1}
48. SC for [2, 1] : { 11, 0}
49. ZC for [0, 2] : { 7, 0}
50. ZC for [3, 2] : { 8, 0}
51. ZC for [1, 3] : { 7, 1}
52. SC for [1, 3] : { 13, 0}

ค่าสถานะในการเข้ารหัสระนาบบิตภายหลังการเข้ารหัสในระนาบบิตที่ 0 ในช่วง SPP

1	0	0	1
1	1	0	1
0	1	0	1
0	0	0	0

v^0

1	0	0	1
1	1	1	1
1	1	1	1
0	1	0	1

σ

0	1	1	0
0	0	0	1
0	1	0	0
1	0	1	0

η

0	0	0	0
1	0	0	0
1	1	1	0
0	0	0	0

χ

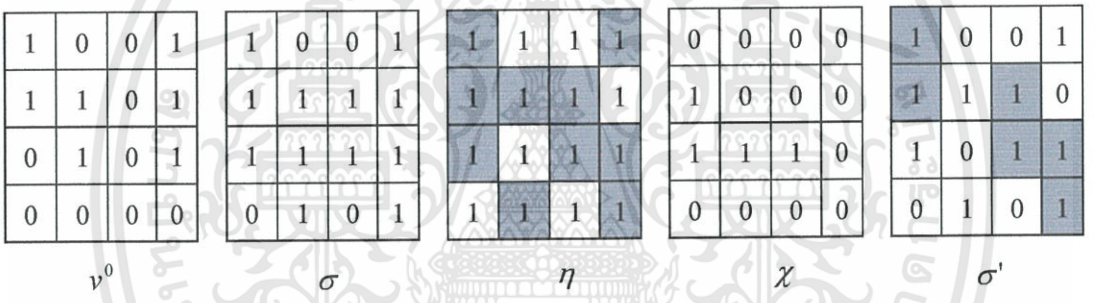
0	0	0	1
0	1	0	0
1	0	0	0
0	1	0	0

σ'

การเข้ารหัสระนาบบิตในช่วง MRP ของระนาบบิตที่ 0

53. MRC for [0, 0] : { 15, 1}
54. MRC for [1, 0] : { 15, 1}
55. MRC for [2, 0] : { 16, 0}
56. MRC for [1, 1] : { 16, 1}
57. MRC for [3, 1] : { 16, 0}
58. MRC for [1, 2] : { 15, 0}
59. MRC for [2, 2] : { 15, 0}
60. MRC for [0, 3] : { 16, 1}
61. MRC for [2, 3] : { 15, 1}
62. MRC for [3, 3] : { 15, 0}

ค่าสถานะในการเข้ารหัสระนาบบิตภายหลังการเข้ารหัสระนาบบิตที่ 0 ในช่วง MRP



การเข้ารหัสระนาบบิตในช่วง CUP ของระนาบบิตที่ 0 ไม่มีการเข้ารหัสในช่วงนี้ เนื่องจากข้อมูลถูกเข้ารหัสหมดทั้งระนาบบิตนี้แล้ว

หลังจากการเข้ารหัสระนาบบิตในระนาบสุท้ายแล้วจึงเป็นการสิ้นสุดการเข้ารหัสระนาบบิตซึ่งผลลัพธ์คู่อันดับ ได้ทั้งหมด 62 คู่อันดับ ดังที่ได้แสดงด้านบน

จากตัวอย่างผลลัพธ์การเข้ารหัสระนาบบิตแสดงให้เห็นว่าถ้าข้อมูลมีค่าเป็น A = [3, -3, -4, 0, 0, 7, -1, 6, 0, 2, -2, 0, 5, 1, 3, 2] (3.20ก) เมื่อได้ทำการหาค่า Entropy (A) แล้วจะได้ค่า Entropy = 1 บิตต่อสัญลักษณ์ เมื่อนำข้อมูลในชุด A มาผ่านการประมวลผลด้วยการเข้ารหัสระนาบบิตที่เป็น 19 CX และ 2 D (Decision) ได้จำนวนผลลัพธ์คู่อันดับสัญลักษณ์ทั้งสิ้น 62 คู่อันดับ เมื่อนำไปหาค่า Entropy ใหม่ ทำให้ได้ค่าของ Entropy = 0.2829 บิตต่อสัญลักษณ์ ซึ่งสามารถลดจำนวนบิตที่ใช้แทนข้อมูลใน 1 สัญลักษณ์ ได้ถึง 3.5348 เท่าของการนำข้อมูลสัมประสิทธิ์เวฟเลทไปใช้โดยตรง

3.4 การทำงานแบบส่งต่อ (Pipeline)

ในการคำนวณการทำงานแบบส่งต่อ คือ กลุ่มของข้อมูลที่จะถูกประมวลผลที่มีความสัมพันธ์กัน โดยผลลัพธ์จากการประมวลผลของข้อมูลกลุ่มหนึ่งกลายเป็นข้อมูลสำหรับใช้ประมวลผลของกลุ่มข้อมูลในกลุ่มต่อไป ในแต่ละส่วนของการทำงานแบบส่งต่อสามารถที่จะประมวลผลขนานกันไป หรือแบ่งเป็นช่วงของเวลาในการทำงาน ด้วยสาเหตุดังกล่าวการทำงานในแต่ละส่วนของการทำงานแบบส่งต่อจึงใช้หน่วยเก็บข้อมูลเชื่อมต่อคั่นกลางในแต่ละส่วนของการทำงานแบบส่งต่อ

การทำงานแบบส่งต่อเป็นแนวคิดทางธรรมชาติที่อยู่ในชีวิตประจำวัน ตัวอย่างเช่น กระบวนการในการซักผ้าของร้านรับซักรีด ซึ่งจะมีขั้นตอนซึ่งจะยกมาอย่างง่ายดังต่อไปนี้

ขั้นตอนที่ 1. นำเสื้อผ้าเข้าเครื่องซักผ้าและให้เครื่องซักผ้าทำงาน

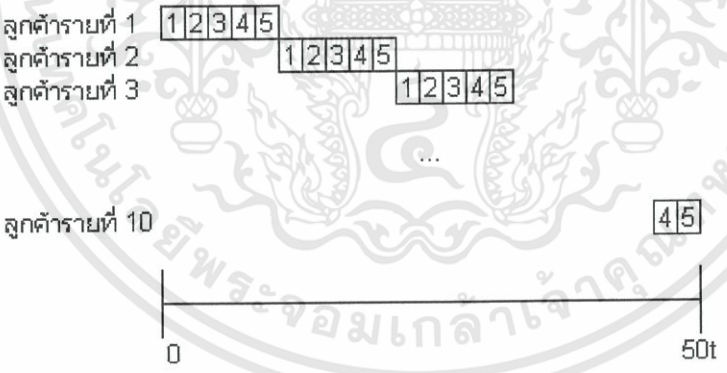
ขั้นตอนที่ 2. นำเสื้อผ้าที่ซักเสร็จเรียบร้อยออกจากเครื่องซักผ้าและนำมาเข้าเครื่องอบ

ขั้นตอนที่ 3. นำเสื้อผ้าที่ผ่านการอบออกจากเครื่องและนำมารีด

ขั้นตอนที่ 4. พับผ้าที่ผ่านจากการรีด

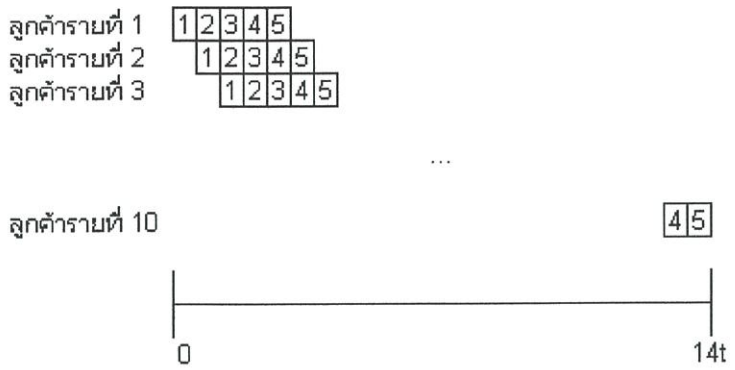
และขั้นตอนสุดท้าย นำผ้าที่พับเสร็จเรียบร้อยมาจัดเรียงเพื่อส่งมอบให้กับลูกค้าแต่ละราย

ถ้าหากไม่ได้มีการทำงานแบบส่งต่อ และสมมุติว่ามีลูกค้าทั้งหมด 10 ราย ลำดับในการทำงานของการซักผ้านี้จะเป็นดังรูปที่ 3.22



รูปที่ 3.22 แสดงกระบวนการในการซักผ้าที่ไม่ได้ใช้แนวคิดในเรื่องการทำงานแบบส่งต่อ

หากการทำงานดังที่ได้กล่าวข้างต้นได้นำหลักการทำงานแบบส่งต่อมาใช้งานสามารถแสดงกระบวนการในการซักผ้าได้ดังรูปที่ 3.23



รูปที่ 3.23 แสดงกระบวนการในการซักผ้าที่ใช้แนวคิดในเรื่องการทำงานแบบส่งต่อ

จากรูปที่ 3.23 จะเห็นได้ว่า ในขณะที่ทำงานในขั้นตอนที่ 2 ของลูกค้าคนที่ 1 ซึ่งเครื่องซักผ้าว่างอยู่ ทำให้สามารถนำเสื้อผ้าของลูกค้าคนที่ 2 นำไปเข้าเครื่องซักผ้าและปล่อยให้เครื่องซักผ้าทำงานในช่วงเวลาดังกล่าวได้ ด้วยความสามารถของการทำงานแบบส่งต่อนี้ทำให้ลดเวลาในกระบวนการซักผ้านี้ได้

สำหรับรูปที่ 3.24 แสดงกระบวนการทำงานของ CPU ที่ทำงานแบบ 5 Stage โดยในตัวอย่างได้ยกคำสั่งที่จะส่งให้ CPU ประมวลผล 2 คำสั่งคือ การลบ (SUB) และการนำค่ามาทำการ AND

SUB r3, r4 -> r5

AND r1, r2 -> r6

	Cycle 0	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5
Fetch	SUB	AND				
Decode		SUB	AND			
Execute			SUB	AND		
Access				SUB	AND	
Writeback					SUB	AND

รูปที่ 3.24 แสดงตัวอย่างการทำงานแบบส่งต่อของชุดคำสั่งที่ใช้ในคอมพิวเตอร์แบบ 5 ขั้นตอน

บทที่ 4
โครงสร้างระบบที่นำเสนอและการออกแบบ

4.1 บทนำ

จากความรู้พื้นฐานของกระบวนการในการบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 ที่ได้กล่าวไว้ในบทที่ 3 ซึ่งเวลาในการประมวลผลส่วนใหญ่อยู่ในส่วนของการเข้ารหัสระนาบบิต และการบีบอัดข้อมูลด้วยวิธีการทางคณิตศาสตร์ ทำให้เกิดการทำงานที่ล่าช้าในการบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000

ในบทนี้ จึงนำเสนอทฤษฎีและแนวคิดที่ใช้สร้างระบบโครงสร้างการเข้ารหัสระนาบบิตที่ได้นำเสนอ ทั้งในด้านแนวคิดและการออกแบบ โดยจะกล่าวนำตั้งแต่แนวคิดที่ได้ จนไปถึงการออกแบบโครงสร้างของระบบโดยรวมที่นำเสนอ และอธิบายโครงสร้างย่อยในแต่ละส่วน โดยแบ่งส่วนย่อยๆ ได้ดังนี้คือ ส่วนที่ใช้ในการเชื่อมต่อข้อมูลทางด้านอินพุตกับระบบการเข้ารหัสระนาบบิตที่นำเสนอ (Input Interface Module), ส่วนที่ใช้ในการจัดเก็บและการเข้าถึงข้อมูลแบบคู่ (Dual Memories Module), ส่วนที่ใช้ในการควบคุมการทำงานหลักของระบบ (Control Module), ส่วนที่ใช้ในการสร้างสัญญาณแสดงสถานะในการเข้ารหัส (State Signal Generator Module) และส่วนที่ใช้ในการดำเนินการสำหรับการเข้ารหัสขั้นพื้นฐาน (Primitive Operator Generator Module) อันเป็นส่วนสุดท้ายที่กล่าวถึง

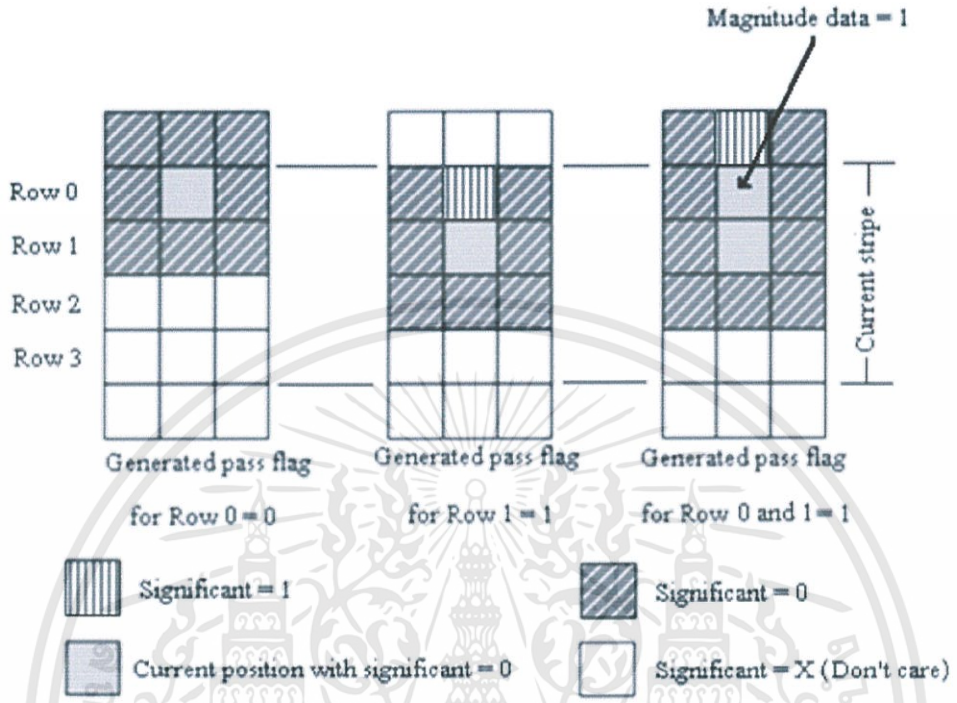
4.2 แนวคิดในการออกแบบ

จากงานวิจัย [9] ซึ่งนำเสนอแนวคิดโครงสร้างการประมวลผลการเข้ารหัสระนาบบิต ที่แบ่งการทำงานออกเป็น 3 ส่วนหลัก คือ ส่วนการจัดเก็บและการเข้าถึงข้อมูล ส่วนการเตรียมข้อมูลและสร้างสัญญาณสถานะต่างๆ ที่เกี่ยวข้องในการเข้ารหัส และส่วนการเข้ารหัสข้อมูลระนาบบิต การทำงานในแต่ละส่วนมีจุดเด่นและข้อจำกัดดังต่อไปนี้

1) ส่วนการจัดเก็บและเข้าถึงข้อมูลในงานวิจัย [9] ถูกออกแบบให้ใช้หน่วยความจำเพียงชุดเดียว ทำให้ข้อมูลที่ถูกนำมาประมวลผลที่มีปริมาณมากนั้น ไม่สามารถถูกนำมาประมวลผลใช้งานได้อย่างต่อเนื่อง เมื่อใช้งานข้อมูลที่ถูกนำเก็บไว้หมดและจำเป็นต้องรอการนำเข้าข้อมูลใหม่ในบล็อกถัดไปให้เสร็จสิ้นเสียก่อน จึงจะสามารถนำข้อมูลชุดใหม่ไปประมวลผลการเข้ารหัสได้

2) ส่วนการเตรียมข้อมูลและสร้างสัญญาณสถานะต่างๆ ที่เกี่ยวข้องในการเข้ารหัส ส่วนนี้จะคอยจัดการกับข้อมูลที่อยู่ในบริเวณขอบของรูปภาพ รวมถึงการสร้างสัญญาณสถานะต่างๆ ที่เกี่ยวข้องในการเข้ารหัส ได้แก่ สัญญาณสถานะของช่วงการทำงานในการเข้ารหัส สัญญาณสถานะที่บ่งบอกถึงตำแหน่งของข้อมูลที่ได้ถูกประมวลผลไปเป็นที่เรียบร้อยแล้ว เป็นต้น ซึ่งในส่วนนี้เป็น

จุดเด่นของงานวิจัย [9] เนื่องจากสามารถระบุตำแหน่งของข้อมูลที่จะถูกประมวลผลได้พร้อมๆ กัน ในแต่ละคอลัมน์ดังแสดงในรูปที่ 4.1



รูปที่ 4.1 แสดงตัวอย่างลักษณะการเกิดสัญญาณความเป็นสมาชิกของ SPP

จากรูปที่ 4.1 แสดงตัวอย่างการเกิดสัญญาณความเป็นสมาชิกของ SPP ที่สามารถระบุตำแหน่งได้พร้อมๆ กัน ในตัวอย่างนี้ลักษณะเงื่อนไขในการเกิดสัญญาณความเป็นสมาชิก (ตามที่ได้กล่าวไว้ในบทที่ 3) แบ่งได้เป็น 3 กรณีคือ

กรณีที่ 1 (รูปซ้าย): ในกรณีนี้สัญญาณสถานะที่แสดงนัยสำคัญของตำแหน่งแถวที่ 0 และตำแหน่งรอบข้างของตำแหน่งแถวที่ 0 มีค่าเป็น '0' (สัญญาณสถานะที่แสดงนัยสำคัญไม่ปรากฏ) ในส่วนเงื่อนไขนี้สามารถบ่งบอกได้ว่าข้อมูลที่กำลังจะถูกประมวลผลในแถวที่ 0 ไม่เป็นสมาชิกของ SPP

กรณีที่ 2 (รูปกลาง): ในกรณีนี้สัญญาณสถานะที่แสดงนัยสำคัญของตำแหน่งแถวที่ 0 มีค่าเป็นค่าบ่งชี้นัยสำคัญ (มีค่าเป็น '1') ส่วนสัญญาณสถานะที่แสดงนัยสำคัญรอบข้างของตำแหน่งแถวที่ 1 มีค่าเป็น '0' ในเงื่อนไขของกรณีนี้สามารถบ่งชี้ได้ว่าข้อมูลที่กำลังถูกประมวลผลในแถวที่ 1 เป็นสมาชิกของ SPP

กรณีที่ 3 (รูปขวา): ในกรณีนี้สัญญาณสถานะที่แสดงนัยสำคัญของตำแหน่งแถวที่ 0 และแถวที่ 1 มีค่านัยสำคัญเป็น '0' รวมไปถึงสัญญาณสถานะนัยสำคัญรอบข้างของตำแหน่งแถวที่ 0

และแถวที่ 1 มีค่าเป็น '0' ด้วย ยกเว้นแต่เพียงค่าสถานะนัยสำคัญของตำแหน่งที่อยู่บนตำแหน่งของแถวที่ 0 ที่มีค่าเป็น '1' ในกรณีนี้จะทำให้ข้อมูลที่กำลังจะถูกประมวลผลในตำแหน่งแถวที่ 0 เป็นสมาชิกของ SPP ซึ่งจากค่าของขนาด ในตำแหน่งแถวที่ 0 นี้มีค่าเป็น '1' จึงส่งผลให้ค่าสัญญาณสถานะนัยสำคัญของแถวที่ 0 ภายหลังการเข้ารหัสเกิดการเปลี่ยนแปลงค่าเป็น '1' ด้วยการเปลี่ยนแปลงค่านี้ทำให้ส่งผลกระทบต่อความเป็นสมาชิกของข้อมูลในตำแหน่งแถวที่ 1 ให้กลายเป็นว่าข้อมูลในตำแหน่งแถวที่ 1 ก็เป็นสมาชิกของ SPP เช่นเดียวกัน

จากตัวอย่างทั้ง 3 กรณีที่ได้กล่าวมา เมื่อนำมาประยุกต์กับข้อมูลที่กำลังจะถูกเข้ารหัสพร้อมกันทั้ง 4 แถวคือ ข้อมูลในแถวที่ 0, 1, 2 และ 3 จะทำให้สามารถระบุตำแหน่งของความเป็นสมาชิกในการเข้ารหัสของข้อมูลทั้ง 4 แถวได้อย่างพร้อมๆ กัน อีกทั้งยังสามารถสร้างสัญญาณที่ต้องถูกนำไปใช้ในการเข้ารหัสของข้อมูลในแต่ละตำแหน่งล่วงหน้าก่อนได้อีกด้วย

3) ส่วนการเข้ารหัสข้อมูลระนาบบิตเป็นส่วนที่ใช้ในการเข้ารหัสสัญญาณข้อมูลที่ได้เตรียมไว้เพื่อให้ได้ผลลัพธ์สุดท้าย โดยประสิทธิภาพการเข้ารหัสในส่วนนี้ขึ้นกับส่วนการเตรียมสัญญาณในส่วนก่อนหน้า ซึ่งถูกออกแบบให้สามารถระบุตำแหน่งของข้อมูลที่จะถูกประมวลผลได้พร้อมๆ กัน ทำให้ส่วนการเข้ารหัสนี้สามารถประมวลผลได้ด้วยความเร็วสูง

จากจุดเด่นของงานวิจัย [9] ที่สามารถบ่งบอกถึงตำแหน่งของข้อมูลที่จะถูกประมวลผลได้อย่างพร้อมๆ กันในแต่ละคอลัมน์ และสามารถสร้างสัญญาณที่จะนำไปใช้ในการประมวลผลของการเข้ารหัสล่วงหน้าได้ถูกนำมาประยุกต์ใช้เป็นพื้นฐานของโครงสร้างของระบบการเข้ารหัสระนาบบิตที่นำเสนอ

และจากจุดที่เป็นข้อจำกัดของงานวิจัย [9] คือ ความล่าช้าในการรอนำเข้าข้อมูลในชุดถัดไปเพื่อใช้ในการประมวลผล ดังนั้นในงานวิจัยที่นำเสนอนี้ได้ใช้วิธีการในการจัดเก็บและการเข้าถึงข้อมูลแบบคู่มาช่วยเพิ่มประสิทธิภาพในการประมวลผล นอกจากนี้ถึงแม้ว่างานวิจัย [9] สามารถบ่งบอกถึงตำแหน่งของข้อมูลที่จะถูกประมวลผลได้อย่างพร้อมๆ กันในแต่ละคอลัมน์ แต่การประมวลผลในแต่ละช่วงการทำงานของการเข้ารหัสยังเป็นการทำงานแบบประมวลผลไปทีละพาสการเข้ารหัส

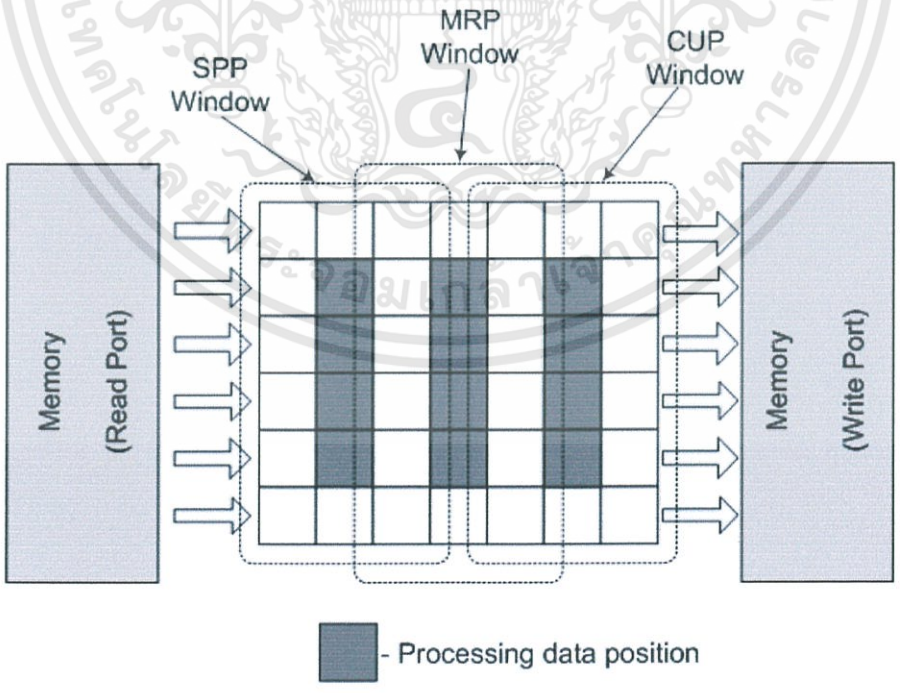
จากมาตรฐาน JPEG2000 จะมีการประมวลผลข้อมูลการเข้ารหัสระนาบบิต 3 พาสการเข้ารหัสด้วยกัน ทำให้งานวิจัย [9] ต้องทำงานรอการประมวลผลเสร็จทีละพาสการเข้ารหัสเรียงกันไป ทำให้การประมวลผลยังมีความล่าช้าอยู่ ด้วยปัญหาดังที่กล่าวมานี้ ในงานวิจัยที่นำเสนอนี้จึงได้ทำการรวมการทำงานทั้ง 3 พาสการเข้ารหัสให้ทำงานเพียงรอบเดียวโดยวิธีการนั้นก็คือ การประมวลผลของการเข้ารหัสทั้ง 3 พาสการเข้ารหัสมาจัดโครงสร้างให้ทำงานต่อเนื่องกันไปในคราวเดียวกันดังแสดงในรูปที่ 4.2

จากหลักการทำงานแบบส่งต่อกันไปที่สามารถแบ่งการทำงานใหญ่ๆ ออกเป็นงานเล็กๆ ที่ถูกประมวลผลต่อเนื่องกันไป โดยผลลัพธ์ที่ได้จากการทำงานก่อนหน้าจะกลายเป็นข้อมูลอินพุต

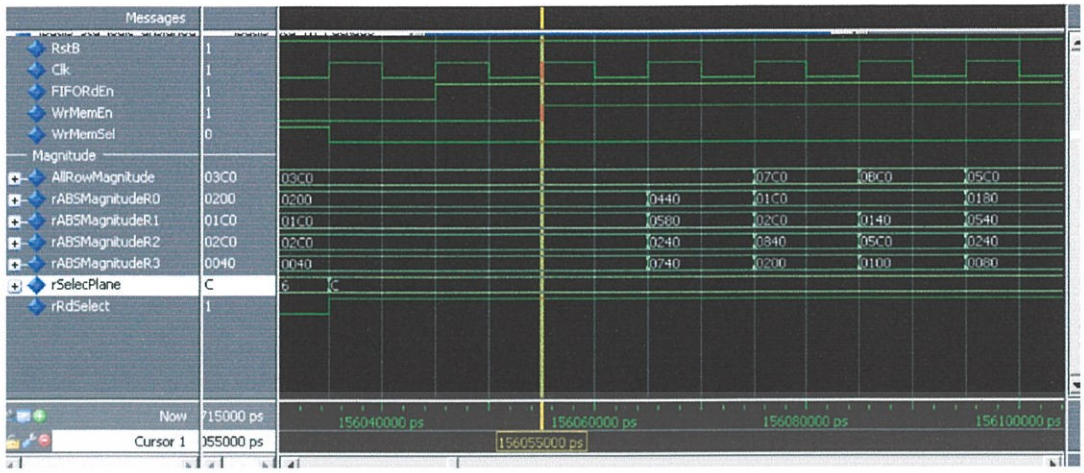
สำหรับการทำงานในช่วงถัดไปจึงมีความคล้ายคลึงกับการเข้ารหัสระนาบบิต งานเล็กๆ ที่ได้ถูกแบ่งออกมาในระบบโครงสร้างที่นำเสนอ คือ การเข้ารหัสในแต่ละช่วงพาสการเข้ารหัสนั่นเอง

จากรูปที่ 4.2 จะเห็นได้ว่าการทำงานในการเข้ารหัสระนาบบิตถูกแบ่งออกเป็น 3 พาส โดยเริ่มจากข้อมูลที่จะถูกประมวลผลถูกอ่านออกมาจากหน่วยความจำส่งต่อไปยังหน้าต่างการเข้ารหัสของช่วง SPP หลังจากได้ผ่านการประมวลผลเรียบร้อยในช่วง SPP แล้ว ข้อมูลที่ผ่านการประมวลผลจะส่งต่อไปเป็นอินพุตของช่วงการเข้ารหัส MRP ส่วนข้อมูลใหม่ก็เข้ามาถูกประมวลผลแทนที่ข้อมูลเก่าที่เลื่อนออกไป ในทำนองเดียวกันเมื่อข้อมูลเก่าถูกประมวลผลในช่วง MRP ข้อมูลนั้นก็จะกลายเป็นอินพุตให้กับการเข้ารหัสในช่วง CUP ต่อไป ภายหลังจากการประมวลผลการเข้ารหัสระนาบบิตเสร็จสิ้นในช่วง CUP แล้ว ข้อมูลค่าสัญญาณสถานะในการเข้ารหัสจะถูกเขียนกลับลงในตำแหน่งเดิมของหน่วยความจำเพื่อใช้เข้ารหัสระนาบบิตในระนาบถัดๆ ไป

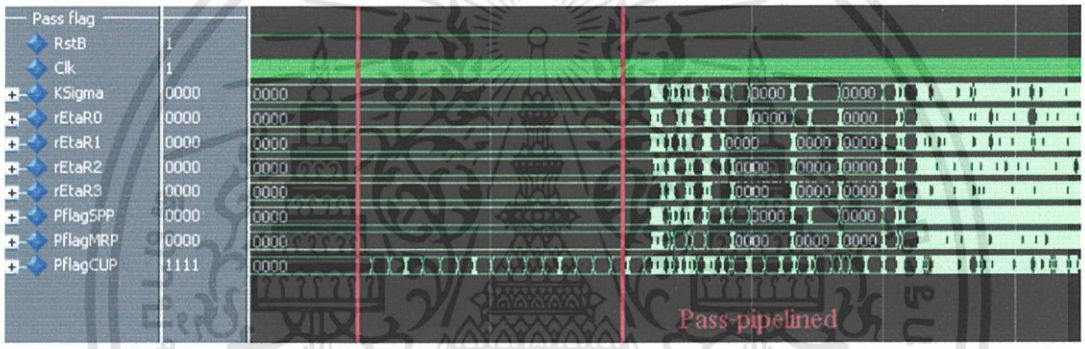
การออกแบบโดยใช้ความสามารถในการทำงานแบบส่งต่อกันไปที่ได้กล่าวมาทำให้สามารถลดระยะเวลาในการเข้าถึงข้อมูลในหน่วยความจำเทียบกับงานวิจัย [9] รวมไปถึงข้อมูลที่ถูกประมวลผลยังมีความต่อเนื่องกัน ตั้งแต่ SPP, MRP เรื่อยไปจนถึง CUP โดยไม่ต้องรอให้การประมวลผลข้อมูลต้องประมวลผลในช่วง SPP จนครบทั้งระนาบบิตก่อนถึงจะสามารถประมวลผลในช่วง MRP และ CUP ได้ ซึ่งจากจุดเด่นของการทำงานแบบส่งต่อกันไปนี้ สามารถลดข้อจำกัดของงานวิจัย [9] ในเรื่องของการประมวลผลซ้ำซ้อนข้อมูลที่ใช้ระยะเวลาในการทำงานในแต่ละพาสการเข้ารหัสที่ยาวนานลง



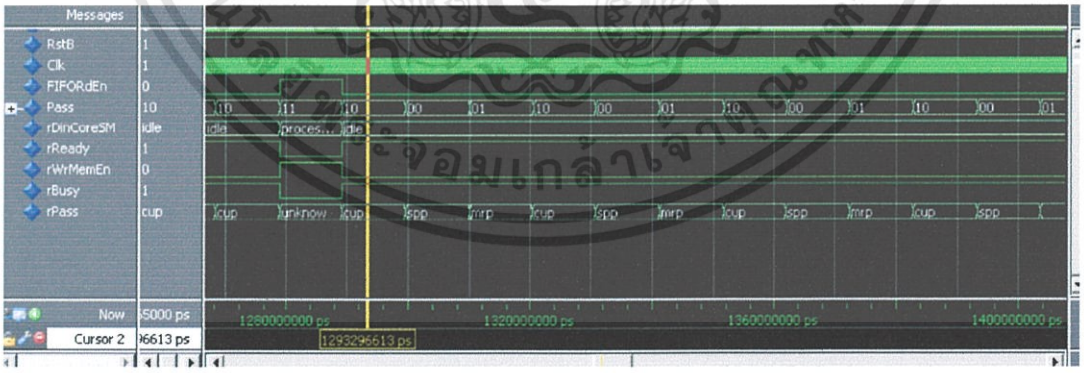
รูปที่ 4.2 แสดงกรอบหน้าต่างประมวลผลที่ใช้การทำงานแบบส่งต่อ



รูปที่ 4.3 แสดงสัญญาณในช่วงเวลาของการรับข้อมูล



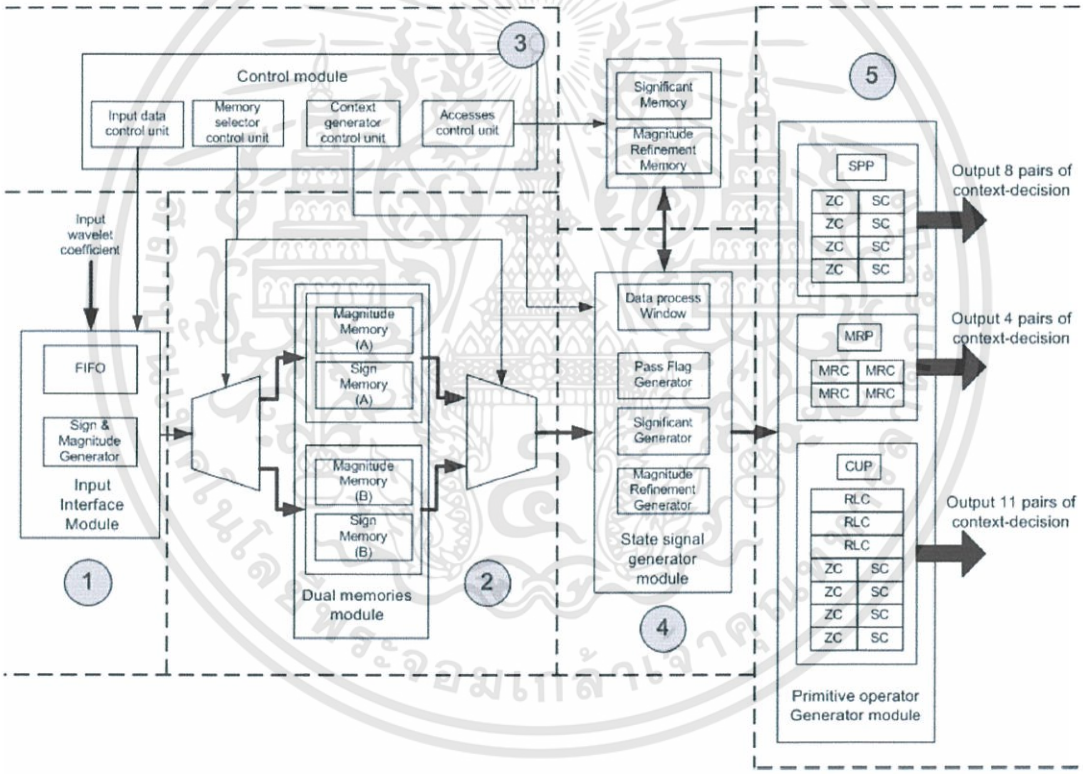
รูปที่ 4.4 แสดงเวลาในการประมวลผลแบบส่งต่อพาส



รูปที่ 4.5 แสดงเวลาในการประมวลผลแบบลำดับขั้นของงานวิจัย [9]

4.3 โครงสร้างระบบโดยรวมที่นำเสนอในงานวิจัยนี้

โครงสร้างระบบการเข้ารหัสระนาบบิตที่นำเสนอได้ถูกออกแบบโดยมีพื้นฐานจากจุดเด่นในงานวิจัย [9] ในส่วนของความสามารถในการระบุตำแหน่งของข้อมูลที่จะใช้ในการประมวลผลพร้อมกันทั้งคอลัมน์และการสร้างสัญญาณที่ต้องถูกนำไปใช้ในการเข้ารหัสของข้อมูลในแต่ละตำแหน่งล่วงหน้า นอกจากนี้ได้มีการเพิ่มประสิทธิภาพด้วยการเพิ่มชุดของการจัดเก็บและการเข้าถึงข้อมูลแบบคู่ เพื่อลดความล่าช้าในการนำเข้าข้อมูล ทำให้สามารถนำส่งข้อมูลเพื่อนำไปใช้ประมวลผลได้อย่างต่อเนื่อง และทำการปรับโครงสร้างการเข้ารหัสระนาบบิตให้สามารถประมวลผลทั้ง 3 พาสการเข้ารหัสได้พร้อมกัน ทำให้สามารถเข้ารหัสได้ผลลัพธ์สูงสุดถึง 22 สัญลักษณ์ภายในหนึ่งสัญญาณนาฬิกาและทำงานด้วยความเร็วของสัญญาณนาฬิกาที่สูงกว่าในงานวิจัย [9] สำหรับโครงสร้างระบบที่นำเสนอสามารถแสดงได้ดังรูปที่ 4.6



รูปที่ 4.6 แสดงโครงสร้างโดยรวมของระบบที่นำเสนอ

จากรูปที่ 4.6 แสดงโครงสร้างโดยรวมของระบบที่นำเสนอซึ่งถูกแบ่งส่วนการทำงานออกเป็นส่วนย่อย ดังนี้

1) <u>ส่วนการเชื่อมต่อข้อมูลทางด้านอินพุตกับระบบการเข้ารหัสระนาบบิตที่นำเสนอ (Input Interface Module)</u> ในส่วนของโมดูลนี้ ทำหน้าที่รับข้อมูลสัมประสิทธิ์เวฟเลทที่

เป็นข้อมูลที่มีการคิดเครื่องหมายเข้ามา และทำการแปลงข้อมูลสัมประสิทธิ์เวฟเลทที่รับเป็นข้อมูล 2 ส่วนคือ ข้อมูลในส่วนที่เป็นเครื่องหมายของค่าสัมประสิทธิ์เวฟเลท (Sign Data) และข้อมูลในส่วนที่เป็นค่าสัมบูรณ์ซึ่งไม่คิดเครื่องหมายของค่าสัมประสิทธิ์เวฟเลท (Magnitude Data)

2) ส่วนการจัดเก็บข้อมูลและการเข้าถึงข้อมูลแบบคู่ (Dual Memories Module) ในส่วนของโมดูลนี้ทำหน้าที่จัดเก็บข้อมูลค่าสมบูรณ์ของสัมประสิทธิ์เวฟเลท และข้อมูลเครื่องหมายของค่าสัมประสิทธิ์เวฟเลท โครงสร้างระบบหน่วยความจำแบบคู่ที่นำเสนอประกอบด้วย หน่วยความจำ 2 ชุด คือ ชุดหน่วยความจำ A และชุดหน่วยความจำ B โดยหน่วยความจำในแต่ละชุดจะสลับการทำงานเพื่อสลับกันนำเข้าข้อมูล ทำให้ส่วนการเข้ารหัสสามารถนำข้อมูลไปใช้ในการประมวลผลได้อย่างต่อเนื่อง

3) ส่วนการควบคุมการทำงานหลักของระบบ (Control Module) ในส่วนของโมดูลนี้ทำหน้าที่ในการควบคุมการทำงานทั้งหมดของระบบ แบ่งออกเป็นโมดูลย่อยๆ ได้แก่ โมดูลสร้างสัญญาณสำหรับควบคุมการรับข้อมูลทางด้านอินพุต (Input Data Control Unit), โมดูลสร้างสัญญาณสำหรับควบคุมการเลือกชุดของหน่วยความจำ (Memory Selector Control Unit), โมดูลสร้างสัญญาณสำหรับควบคุมการสร้างสัญญาณสถานะของการเข้ารหัส (Context Generator Control Unit) และโมดูลสร้างสัญญาณสำหรับควบคุมการเข้าถึงค่าสถานะของการเข้ารหัสในหน่วยความจำสถานะของการเข้ารหัส (Accesses Control Unit) สัญญาณควบคุมการทำงานที่ถูกสร้างออกมาใช้ควบคุมชุดของหน่วยความจำที่ใช้ในการเข้ารหัส ควบคุมการไหลของข้อมูลที่จะเข้ารหัส และควบคุมหน่วยความจำสถานะของการเข้ารหัสในแต่ละระนาบบิต การทำงานในส่วนการควบคุมนี้ จะทำงานสัมพันธ์กันไปทั้งระบบพร้อมๆ กับการเข้ารหัสข้อมูล

4) ส่วนการสร้างสัญญาณแสดงสถานะในการเข้ารหัส (State Signal Generator Module) การทำงานในส่วนของโมดูลนี้รับผิดชอบหน้าที่ในการเตรียมข้อมูลที่เกี่ยวข้องสำหรับใช้ในการเข้ารหัส โดยข้อมูลสัญญาณต่างๆ ที่ถูกสร้างจากส่วนการสร้างสัญญาณแสดงสถานะในการเข้ารหัสนี้จะถูกส่งต่อไปยังส่วนการดำเนินการเข้ารหัสขั้นพื้นฐานเพื่อใช้ในการเข้ารหัสต่อไป ในส่วนของการทำงานจะถูกแบ่งออกเป็นส่วนย่อยๆ ดังที่จะได้แสดงด้านล่างพร้อมทั้งหน้าที่รับผิดชอบของส่วนการทำงานย่อยๆ นั้นรับผิดชอบ

- ส่วนการทำงานย่อย Data Process Window: รับผิดชอบหน้าที่ในการเตรียมข้อมูลขนาด ที่จะใช้ในการเข้ารหัส โดยการดึงข้อมูลค่าสัมบูรณ์สัมประสิทธิ์เวฟเลทจากหน่วยความจำแบบคู่ โดยข้อมูลจะถูกดึงเข้ามาทีละระนาบบิต เริ่มจากระนาบบิตที่มีนัยสำคัญมากที่สุดเรื่อยไปจนถึงระนาบบิตที่มีนัยสำคัญน้อย

ที่สุด ซึ่งหลังจากที่เริ่มดึงข้อมูลเข้ามาแล้ว ในส่วนการทำงานย่อยนี้จะรวบรวมข้อมูลของขนาดในตำแหน่งปัจจุบันที่กำลังจะถูกประมวลผลและข้อมูลขนาดตำแหน่งบริเวณรอบข้างของตำแหน่งที่กำลังจะถูกประมวลผล จัดเป็นเซตของข้อมูลเตรียมไว้เพื่อให้ส่วนการเข้ารหัสสามารถนำไปใช้เข้ารหัสได้ในทันที โดยกรอบหน้าต่างของข้อมูลถูกออกแบบเป็นรีจิสเตอร์ทั้งหมด 7 คอลัมน์ซึ่งการไหลของข้อมูลในแต่ละคอลัมน์จะทำงานแบบส่งต่อกันไป ดังแสดงในรูปที่ 4.2

- ส่วนการทำงานย่อย Pass Flag Generator: ในส่วนการทำงานนี้ รับผิดชอบในส่วนของการสร้างสัญญาณ Pass ที่ใช้บ่งบอกว่าข้อมูลในแต่ละตำแหน่งจะถูกประมวลผลใน Pass ใดใน 3 Pass (SPP, MRP, CUP)
- ส่วนการทำงานย่อย Significant Generator: ในส่วนการทำงานย่อยนี้รับผิดชอบหน้าที่ในการสร้างสัญญาณสถานะนัยสำคัญซึ่งค่าสัญญาณสถานะนัยสำคัญนี้สามารถที่จะเกิดการเปลี่ยนค่าสถานะอย่างทันทีทันใดได้ในการเข้ารหัสที่ทำงานพร้อมๆกันทั้งคอลัมน์ ทำให้ส่วนการเข้ารหัสมีความซับซ้อนในการออกแบบ ดังนั้นในส่วนการทำงานย่อยนี้จะสร้างสัญญาณนัยสำคัญที่จำเป็นและเกี่ยวข้องกับข้อมูลในแต่ละตำแหน่งเพื่อเป็นการลดความซับซ้อนเหล่านั้น
- ส่วนการทำงานย่อย Magnitude Refinement Generator: ส่วนการทำงานย่อยนี้ มีหน้าที่รับผิดชอบในส่วนการสร้างสัญญาณสถานะ Magnitude Refinement ซึ่งมีส่วนเกี่ยวข้องกับการเข้ารหัส โดยในส่วนนี้จะเป็นการเตรียมข้อมูลสัญญาณสถานะ Magnitude Refinement ที่มีความสัมพันธ์กับข้อมูลในแต่ละตำแหน่งที่จะถูกประมวลผล ทำให้สามารถลดความซับซ้อนของส่วนการเข้ารหัสได้โดยข้อมูลสัญญาณต่างๆ ที่ได้กล่าวมานี้จะส่งไปยังในส่วนการเข้ารหัสต่อไป

5) ส่วนการดำเนินการเข้ารหัสขั้นพื้นฐาน (Primitive Operator Generator Module) ในส่วนของโมดูลนี้ทำหน้าที่ในการเข้ารหัสข้อมูลด้วยตัวดำเนินการขั้นพื้นฐานต่างๆได้ผลลัพธ์เป็นสัญลักษณ์ที่ถูกเข้ารหัส (Context-decision Pairs) ซึ่งจะถูกส่งต่อไปในส่วนการบีบอัดสัญญาณตามมาตรฐาน JPEG2000 ส่วนต่อไป โดยในส่วนของโมดูลนี้ประกอบไปด้วยตัวดำเนินการที่ใช้ในการเข้ารหัสทั้งหมด 4 แบบ คือ

- ตัวดำเนินการเข้ารหัสศูนย์ (ZC)
- ตัวดำเนินการเข้ารหัสแบบ Magnitude Refinement (MRC)
- ตัวดำเนินการเข้ารหัสเครื่องหมาย (SC)

- ตัวดำเนินการเข้ารหัสแบบ Run-length (RLC)

ซึ่งในแต่ละพาสของการเข้ารหัส จะมีตัวดำเนินการขั้นพื้นฐานเฉพาะตัวดำเนินการที่ใช้ในการเข้ารหัสที่เกี่ยวข้องกับพาสนั้น ส่วนตัวดำเนินการเข้ารหัสที่ไม่เกี่ยวข้องในการเข้ารหัสของพาสนั้นจะไม่ถูกรวมเข้ามาด้วย จากรูปที่ 4.6 จะเห็นได้ว่าในส่วนของ SPP ประกอบไปด้วย 4 ZC และ 4 SC ที่สามารถประมวลผลได้พร้อมๆ กัน ในส่วนของ MRP ประกอบไปด้วย 4 MRC ที่สามารถทำงานได้อย่างพร้อมกัน สำหรับส่วน CUP ประกอบไปด้วย 3 RLC, 4 ZC และ 4 SC โดยแต่ละตัวสามารถทำงานเป็นอิสระและสามารถทำงานได้พร้อมๆ กัน

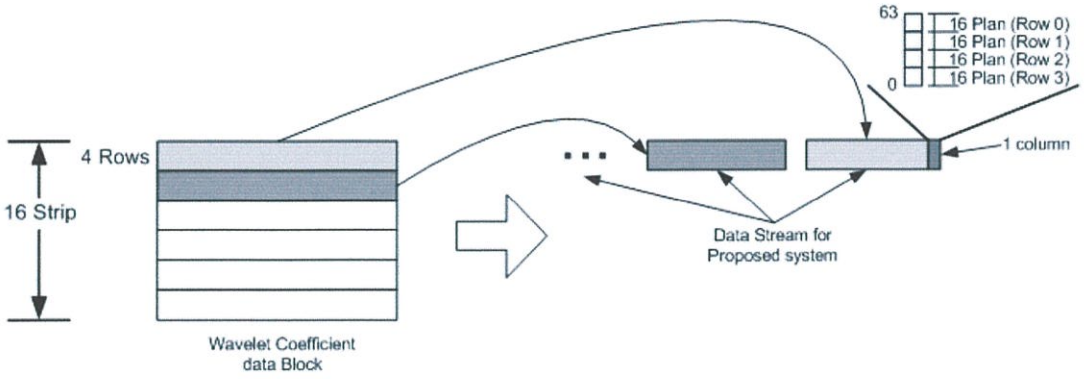
ซึ่งผลจากการออกแบบโครงสร้างของระบบที่นำเสนอ โดยแบ่งการทำงานออกเป็นส่วนย่อย ทำให้การทำงานสามารถที่จะทำงานส่งต่อเป็นทอดๆ ได้ ตั้งแต่การรับข้อมูลทางด้านอินพุตส่งต่อเรื่อยไปจนถึงได้ผลลัพธ์การเข้ารหัสออกทางเอาต์พุต นอกจากนี้ยังมีส่วนของการเตรียมข้อมูลที่สามารถแบ่งการทำงานของแต่ละพาส ให้ข้อมูลสามารถทำงานส่งต่อกันไปตั้งแต่ SPP MRP และสิ้นสุดที่ CUP ยังผลให้ประสิทธิภาพในการทำงานของระบบที่นำเสนอมีประสิทธิภาพที่สูงกว่างานวิจัย [9] อีกทั้งจุดด้อยในงานวิจัย [9] ในเรื่องของความล่าช้าในการนำเข้าของข้อมูลได้ถูกลดปัญหาลงจากการทำงานของระบบหน่วยความจำแบบคู่

4.3.1 โครงสร้างของส่วนการเชื่อมต่อข้อมูลทางด้านอินพุตกับระบบการเข้ารหัสระนาบบิตที่นำเสนอ (Input Interface Module)

ในส่วนของโมดูลการทำงานนี้ เป็นส่วนที่ใช้ในการเชื่อมต่อการทำงานของส่วนการแปลงเวฟเลท เพื่อนำเข้าค่าสัมประสิทธิ์เวฟเลทไปทำการเข้ารหัสด้วยระบบการเข้ารหัสระนาบบิตที่นำเสนอ โดยค่าสัมประสิทธิ์เวฟเลทที่นำเข้านี้เป็นข้อมูลที่มีการคิดเครื่องหมาย ขนาดข้อมูลที่รับเข้ามาในแต่ละบล็อก มีขนาด 64 x 64 จุด ซึ่งข้อมูลในแต่ละจุดมีจำนวนทั้งสิ้น 16 ระนาบบิต (Bit Plane)

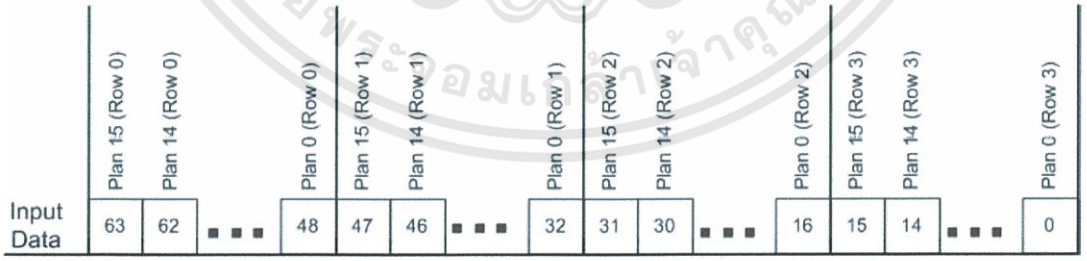
สำหรับข้อมูลที่รับเข้ามาจะใช้การเชื่อมต่อข้อมูลด้วยรูปแบบ FIFO (First In First Out) โดยในส่วนโมดูลภายนอกที่ต้องการส่งข้อมูลสามารถตรวจสอบสัญญาณความพร้อมของส่วนเชื่อมต่อทางด้านอินพุต หากสัญญาณที่ตรวจสอบบ่งบอกว่าระบบพร้อมรับข้อมูลทางด้านอินพุตโมดูลภายนอกสามารถเขียนข้อมูลเข้ามาให้กับระบบได้ทันที

จากรูปที่ 4.7 แสดงข้อมูลที่ป้อนให้กับส่วนการเชื่อมต่อข้อมูลทางด้านอินพุตเพื่อนำไปประมวลผลในระบบการเข้ารหัสระนาบบิตที่นำเสนอ โดยข้อมูลสัมประสิทธิ์เวฟเลทที่ถูกแบ่งออกเป็น Code-block ขนาด 64 x 64 จุด ซึ่งข้อมูลที่จะนำเข้าในแต่ละ Code-block ถูกแบ่งออกเป็นส่วนย่อย โดยส่วนย่อยเหล่านี้ถูกเรียกว่า “Stripe”



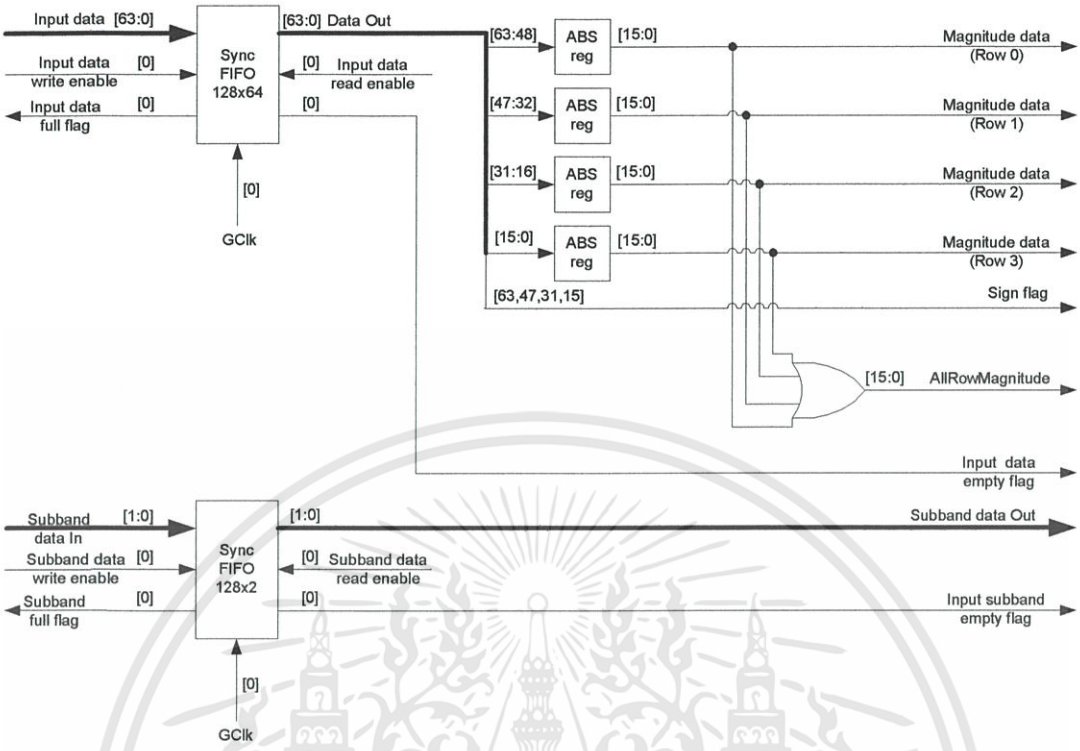
รูปที่ 4.7 แสดงข้อมูลที่ป้อนให้กับ Input Interface Module

ขนาดข้อมูลของแต่ละ Stripe มีขนาดสูง 4 แถว และความกว้างเท่ากับความกว้างของบล็อกข้อมูลที่จะนำเข้า ในที่นี้คือ 64 คอลัมน์ ดังนั้นจากข้อมูลที่จะนำเข้าขนาด 64 x 64 จุด สามารถแบ่งข้อมูลออกเป็น Stripe ได้ทั้งสิ้น 16 Stripe ขนาด 4 x 64 จุดสำหรับแต่ละ Stripe โดยในแต่ละจุดของข้อมูลมีจำนวนระนาบบิตทั้งสิ้น 16 ระนาบบิต เพื่อให้ข้อมูลที่จะนำเข้ามามีความรวดเร็ว ดังนั้นการนำข้อมูลเข้ามาในส่วนการเชื่อมต่อข้อมูลทางด้านอินพุตกับระบบการเข้ารหัสระนาบบิตที่นำเสนอจะนำเข้ามาที่ละ 1 คอลัมน์ของ Stripe ตั้งแต่คอลัมน์แรกของ Stripe แรกเรื่อยไปจนถึงคอลัมน์สุดท้ายของ Stripe แล้วจึงนำข้อมูลในคอลัมน์แรกของ Stripe ถัดมานำเข้ามาในระบบเรื่อยไปจนครบทั้ง 16 Stripe หรือครบสิ้นทั้งบล็อกข้อมูลที่จะนำเข้านั่นเอง จะเห็นได้ว่าข้อมูลที่จะนำเข้ามาในแต่ละสัญญาณนาฬิกา ข้อมูลที่นำเข้า 1 คอลัมน์จะมีขนาด 64 บิต ซึ่งประกอบด้วย 16 ระนาบบิตของแต่ละจุดภาพ x 4 แถว (1 คอลัมน์ใน 1 Stripe) ซึ่งมีรายละเอียดของข้อมูลในแต่ละระนาบบิตของข้อมูลที่นำเข้า แสดงได้ดังรูปที่ 4.8



รูปที่ 4.8 แสดงรายละเอียดข้อมูลที่นำเข้าในแต่ละบิตใน 1 คอลัมน์

โครงสร้างของส่วนการเชื่อมต่อข้อมูลทางด้านอินพุตกับระบบการเข้ารหัสระนาบบิตที่นำเสนอสามารถแสดงได้ดังรูปที่ 4.9 ซึ่งประกอบไปด้วยสัญญาณและโมดูลย่อยๆ ดังที่จะได้อธิบายต่อไปนี้



รูปที่ 4.9 แสดงโครงสร้างส่วนการเชื่อมต่อข้อมูลทางด้านอินพุตกับระบบที่นำเสนอ

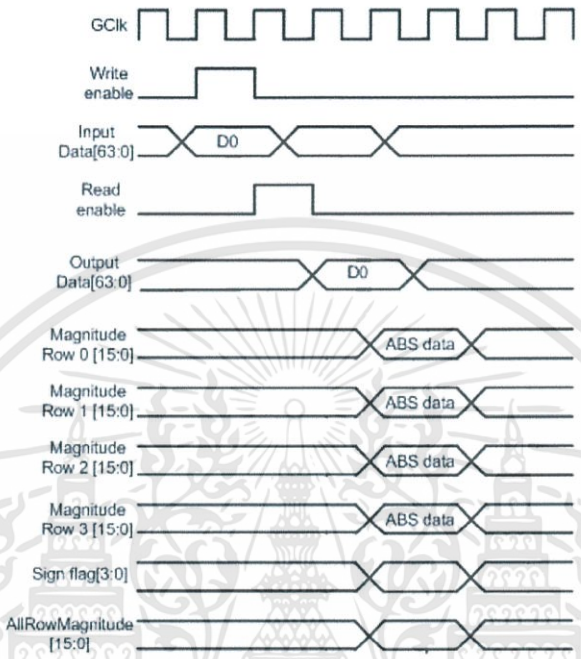
1. Input Data [63:0]: สัญญาณข้อมูลนำเข้าขนาด 64 บิต โดยข้อมูลแยกได้เป็น 16 บิต สำหรับแต่ละแถว ได้แก่ ข้อมูลในตำแหน่ง [63:48] สำหรับข้อมูลของแถวที่ 0 ใน Stripe มีขนาด 16 บิต, ข้อมูลในตำแหน่ง [47:32] สำหรับข้อมูลแถวที่ 1 ของ Stripe, ข้อมูลในตำแหน่ง [31:16] ข้อมูลแถวที่ 2 ของ Stripe, และ ข้อมูลในตำแหน่ง [15:0] สำหรับข้อมูลแถวที่ 3 แสดงได้ดังรูปที่ 4.8
2. Input Data Write Enable [0]: สัญญาณขาเข้าขนาด 1 บิต ใช้สั่งให้ FIFO เก็บข้อมูลขาเข้าของสัมประสิทธิ์เวฟเลท
3. Input Data Full Flag [0]: สัญญาณขาออกขนาด 1 บิต ใช้บ่งบอกสถานะปริมาณข้อมูลใน FIFO หากจำนวนข้อมูลใน FIFO เต็ม สัญญาณนี้จะให้ค่าเป็น '1'(Logic High) ส่วนในสถานะปกติค่านี้จะมีค่าเป็น '0'(Logic Low)
4. GClk [0]: สัญญาณนาฬิกาที่ใช้กระตุ้นให้ระบบทำงาน
5. Subband Data In [1:0]: สัญญาณข้อมูลที่ใช้บ่งบอก Subband ของสัญญาณข้อมูลค่าสัมประสิทธิ์เวฟเลทในแต่ละบล็อก ที่กำลังถูกนำเข้ามารอการประมวลผล ซึ่งแบ่งเป็น 4 แบบ คือ '00' ใช้แทน LL, '01' แทน LH, '10' แทน HL, และ '11' แทน HH

6. Subband Data Write Enable [0]: สัญญาณขาเข้าขนาด 1 บิต ใช้สั่งให้ FIFO เก็บข้อมูลที่ใช้บ่งบอก Subband ของข้อมูลค่าสัมประสิทธิ์เวฟเลท
7. Subband Full Flag [0]: สัญญาณบ่งบอกสถานะจำนวนข้อมูลใน FIFO เช่นเดียวกับข้อ 3. แต่ข้อมูลใน FIFO นี้เป็นข้อมูลของ Subband
8. Data Out [63:0]: สัญญาณข้อมูลเอาท์พุตของ FIFO ขนาด 64 บิต ซึ่งข้อมูลที่ออกจาก FIFO นี้เป็นข้อมูลของค่าสัมประสิทธิ์เวฟเลท
9. Input Data Read Enable: สัญญาณควบคุมการอ่านข้อมูลจาก FIFO ที่เก็บค่า Subband ของข้อมูลในแต่ละบล็อก ของค่าสัมประสิทธิ์เวฟเลท
10. Input Data Empty Flag: สัญญาณบ่งบอกสถานะข้อมูลของ FIFO ถ้าสัญญาณเป็น '1' แสดงว่าไม่มีข้อมูลใน FIFO แล้ว หากสัญญาณนี้เป็น '0' แสดงว่ายังมีข้อมูลใน FIFO
11. AllRowMagnitude: สัญญาณการ OR ข้อมูลของขนาดในแต่ละระนาบบิตของข้อมูลทั้ง Block เพื่อใช้ในการหาค่าของ MSB และ LSB ของข้อมูลแต่ละบล็อกซึ่งค่าข้อมูลที่ได้นี้จะนำไปเป็นตัวกำหนดตำแหน่งของระนาบบิตเริ่มต้นที่จะถูกเข้ารหัส โดยสามารถหาได้จากตำแหน่ง MSB ของสัญญาณข้อมูลที่ได้นี้ สำหรับค่าของตำแหน่ง LSB ของข้อมูลสัญญาณนี้จะใช้กำหนดตำแหน่งของระนาบบิตสุดท้ายที่จะถูกเข้ารหัส ซึ่งระนาบบิตที่ต่ำกว่าค่าของ LSB นี้จะไม่ถูกประมวลผล
12. Magnitude Data (Row 0 – 3): สัญญาณข้อมูลของแต่ละแถวที่ไม่มีการคิดเครื่องหมาย
13. Subband Data Out: สัญญาณเอาท์พุตของข้อมูลที่ใช้แทน Subband ของตำแหน่งของข้อมูลในแต่ละบล็อก ที่เข้ามา
14. Subband Data Read Enable: สัญญาณที่ใช้ควบคุม FIFO เช่นเดียวกับข้อ 9.
15. Input Subband Empty: สัญญาณที่ใช้บ่งบอกเช่นเดียวกับข้อ 10.

ขั้นตอนในการทำงานของส่วนที่ใช้ในการเชื่อมต่อข้อมูลทางด้านอินพุตกับระบบที่นำเสนอแสดงได้ดังรูปที่ 4.10

จากรูปที่ 4.10 แสดงผังเวลาลำดับขั้นตอนการทำงานของส่วนการเชื่อมต่อข้อมูลทางด้านอินพุต โดยในช่วงแรก สัญญาณข้อมูลทางด้านอินพุตจะถูกเขียนลงใน FIFO โดยมีสัญญาณการเขียนข้อมูลเป็นสัญญาณควบคุม (Write Enable) สัญญาณข้อมูลทางด้านอินพุตที่ถูกเขียนเข้ามาเป็นข้อมูลสัมประสิทธิ์เวฟเลทที่มีจำนวนระนาบทั้งสิ้น 16 ระนาบบิตจำนวน 64 แถว x 64 คอลัมน์ โดยที่ข้อมูลสัมประสิทธิ์ที่รับเป็นข้อมูลที่มีการคิดเครื่องหมายในรูปแบบ 2's Complement ในแต่ละครั้งของ 1 ลูกสัญญาณนาฬิกา ข้อมูลสัมประสิทธิ์เวฟเลทที่ถูกเขียนเข้ามาเป็นข้อมูลแต่ละหลักของแต่ละ Stripe ซึ่งมีขนาด 4 แถว x 1 คอลัมน์ หรือข้อมูลขนาด 4 แถว x 16 ระนาบบิตต่อตำแหน่งทำให้ข้อมูลที่ถูกเขียนเข้ามามีจำนวนทั้งสิ้น 64 บิตต่อการเขียน 1 ครั้ง การเขียนข้อมูลสัมประสิทธิ์เวฟ

เลทจะดำเนินต่อไปเรื่อยๆ จนกว่าข้อมูลที่ต้องการเขียนจะครบหมดทั้ง Code-block ในการดำเนินการเขียนข้อมูลจะต้องมีการหยุดรอหากตำแหน่งเก็บข้อมูลที่ว่างใน FIFO ถูกเขียนจนเต็มเมื่อตำแหน่งใน FIFO มีที่ว่างข้อมูลก็จะถูกเขียนลง



รูปที่ 4.10 แสดงผังเวลาลำดับขั้นการทำงานส่วนการเชื่อมต่อข้อมูลทางด้านอินพุตกับระบบที่นำเสนอ

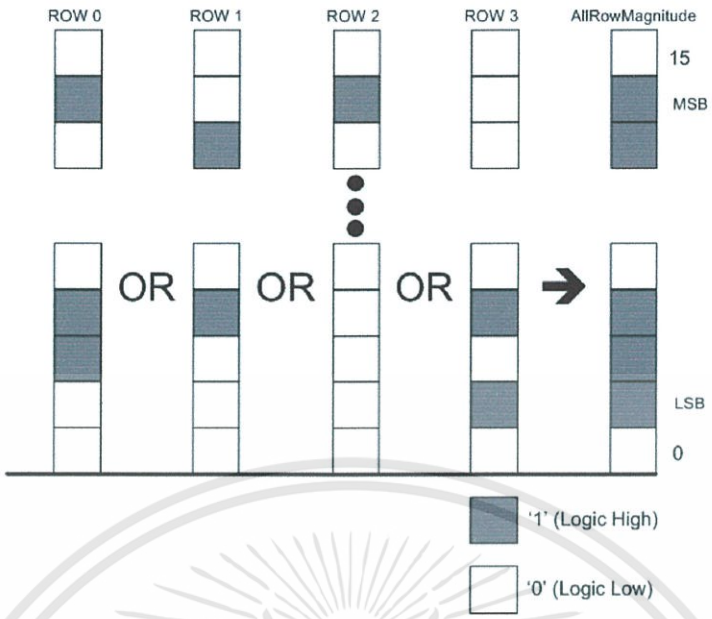
หลังจากที่เริ่มมีการเขียนข้อมูลสัมประสิทธิ์เวฟเลทลงใน FIFO แล้ว ส่วนควบคุมจะตรวจสอบว่าหน่วยความจำชุดใดในส่วนการจัดเก็บข้อมูลระหว่างชุด A หรือ B เป็นหน่วยความจำที่ว่าง เมื่อตรวจสอบได้แล้วจึงจะเริ่มดึงข้อมูลสัมประสิทธิ์เวฟเลทออกมาจาก FIFO (Output Data ในรูปที่ 4.10) ด้วยสัญญาณการอ่านข้อมูล (Read Enable) แล้วจึงทำการแยกข้อมูลสัมประสิทธิ์เวฟเลทที่อ่านออกมาจาก FIFO ออกเป็นข้อมูลค่าสัมบูรณ์ขนาด 16 ระนาบบิตกับสัญญาณข้อมูลเครื่องหมาย นำมาพักเก็บไว้ในรีจิสเตอร์ ABS (Magnitude Row 0 to 3) และรีจิสเตอร์เก็บเครื่องหมาย (Sign Flag) ตามลำดับ สำหรับรีจิสเตอร์ ABS แต่ละตัวมีขนาด 16 บิต โดยมีทั้งสิ้น 4 ตัวเท่ากับข้อมูลด้านอินพุตที่ถูกเขียนเข้ามาในแต่ละคอลัมน์ คอลัมน์ละ 4 แถว ในทำนองเดียวกันรีจิสเตอร์เก็บเครื่องหมาย มีขนาด 4 แถวโดยแต่ละแถวมีขนาดความกว้างเพียง 1 บิต

เพื่อให้ระบบสามารถกำหนดตำแหน่งของระนาบบิตที่เป็นระนาบของบิตที่มีนัยสำคัญมากที่สุด (MSB) กับระนาบของบิตที่มีนัยสำคัญน้อยที่สุด (LSB) ดังที่แสดงรูปที่ 4.11 สัญญาณข้อ

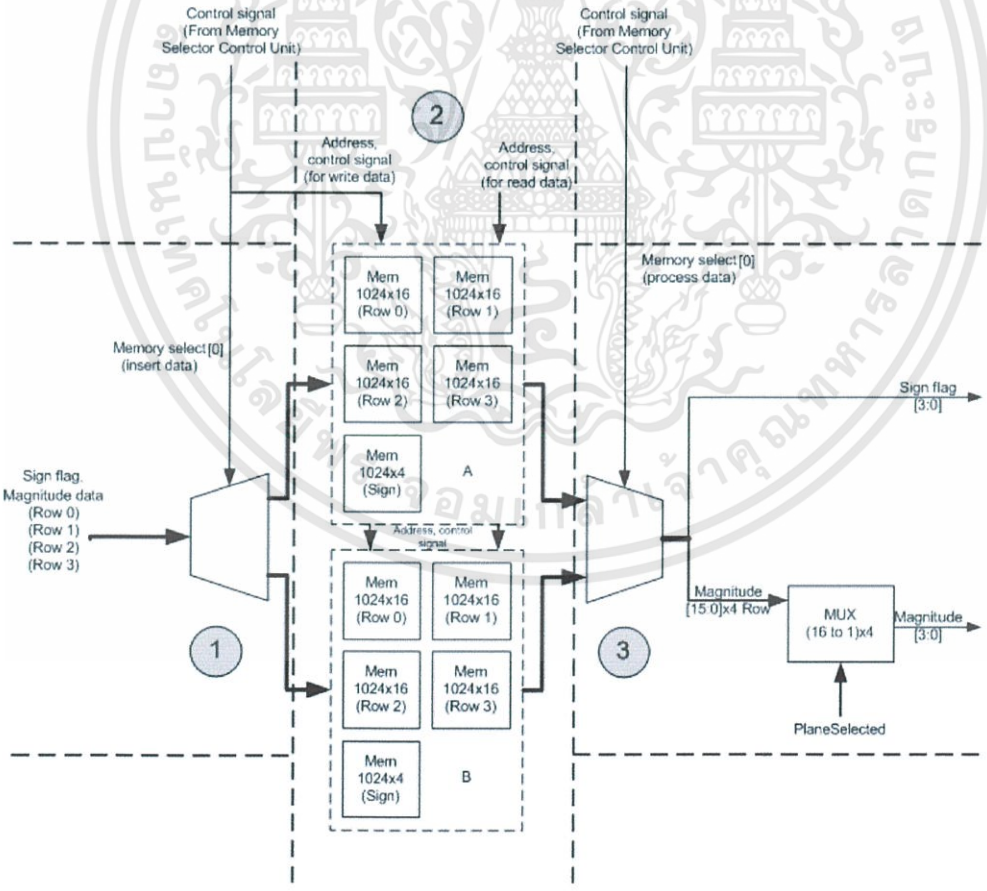
มูลค่าสัมบูรณ์ของแต่ละแถวจะถูกนำมารวมกันด้วยวิธีการ OR ข้อมูลในแต่ละระนาบบิต ซึ่งหลังจากการนำข้อมูลมารวมกันด้วยวิธีการทางตรรกะแล้วข้อมูลที่ได้จะมีขนาด 16 ระนาบบิต โดยรีจิสเตอร์ที่ใช้พักข้อมูลไว้ชั่วคราวนี้คือรีจิสเตอร์ค่าสัมบูรณ์รวมทุกแถว (AllRowMagnitude) โดยสัญญาณบิตที่เป็นตัวแทนของระนาบบิตบนสุดที่มีค่าเป็น '1' จะถือว่าเป็นระนาบบิตที่มีนัยสำคัญมากที่สุดและ ถ้าบิตที่เป็นตัวแทนของระนาบบิตต่ำสุดที่ยังมีค่าบิตเป็น '1' จะถือว่าเป็นระนาบบิตที่มีนัยสำคัญน้อยที่สุด จากตัวอย่างในรูปที่ 4.11 จะเห็นว่า ค่าสัมบูรณ์ที่เก็บในรีจิสเตอร์ AllRowMagnitude มีค่าเป็น '0' ที่บิตตัวแทนของระนาบบิตที่ 15 ดังนั้นจะถือว่าระนาบบิตนี้ไม่มีนัยสำคัญ จะไม่ทำการเข้ารหัส แต่จะมาเริ่มเข้ารหัสที่ระนาบบิตที่ 14 เนื่องจากค่าสัมบูรณ์ที่เก็บในรีจิสเตอร์ AllRowMagnitude มีค่าเป็น '1' แสดงว่าระนาบบิตที่ 14 มีนัยสำคัญ และถือว่าระนาบบิตที่ 14 นี้เป็นระนาบบิตที่มีนัยสำคัญมากที่สุด และระนาบบิตที่มีนัยสำคัญน้อยที่สุดในตัวอย่างนี้จะเป็นระนาบบิตที่ 1 เนื่องจากเป็นระนาบบิตที่ค่าสัมบูรณ์ที่เก็บในรีจิสเตอร์ AllRowMagnitude มีค่าเป็น '1' เป็นระนาบบิตสุดท้าย ระนาบบิตต่ำจากนี้ไป มีค่าสัมบูรณ์เป็น '0' ทั้งหมด ซึ่งแสดงให้เห็นว่า ระนาบบิตต่ำจากนี้เป็นระนาบบิตที่ไม่มีนัยสำคัญ จึงไม่จำเป็นต้องนำมาเข้ารหัส

4.3.2 โครงสร้างส่วนการจัดเก็บและการเข้าถึงข้อมูลแบบคู่ (Dual Memories Module)

สำหรับโครงสร้างส่วนการจัดเก็บและการเข้าถึงข้อมูลแบบคู่นี้ได้ถูกออกแบบให้ใช้หน่วยความจำ 2 ชุด แบ่งเป็นหน่วยความจำชุด A กับหน่วยความจำชุด B โดยหน่วยความจำในแต่ละชุดแบ่งพื้นที่ในการจัดเก็บออกเป็น พื้นที่ที่ใช้เก็บข้อมูลค่าสัมบูรณ์ (Magnitude Data) และพื้นที่ที่ใช้เก็บสัญญาณข้อมูลเครื่องหมาย เนื่องจากข้อมูลที่ระบบโครงสร้างที่นำเสนอต้องนำเข้ามาเก็บในหน่วยความจำถูกแบ่งออกเป็น Stripe จำนวนทั้งสิ้น 16 Stripe โดยในแต่ละ Stripe มีจำนวนคอลัมน์ทั้งสิ้น 64 คอลัมน์ (ในงานวิจัยนี้ Code-block ของข้อมูลที่จะถูกประมวลผลมีขนาด 64 แถว x 64 คอลัมน์) ซึ่งจะทำให้ข้อมูลที่เก็บนั้นจะมีขนาด 16 stripe x 64 คอลัมน์ นั่นคือจะต้องมีจำนวนตำแหน่งในหน่วยความจำที่ใช้เก็บข้อมูลทั้งสิ้น 1024 ตำแหน่งนั่นเอง



รูปที่ 4.11 แสดงสัญญาณที่เกิดจากการ OR ของขนาดทั้ง 4 แถว



รูปที่ 4.12 แสดงโครงสร้างส่วนการจัดเก็บและการเข้าถึงข้อมูลแบบคู่

ในรูปที่ 4.12 แสดงโครงสร้างภายในส่วนการจัดเก็บการเข้าถึงข้อมูลแบบคู่ โดยภายในของโมดูลนี้ประกอบด้วย

1) ส่วนเลือกชุดหน่วยความจำที่ต้องการเก็บข้อมูล ใช้เลือกชุดของหน่วยความจำที่จะเก็บข้อมูลที่มาจากส่วนการเชื่อมต่อข้อมูลทางด้านอินพุต ซึ่งสัญญาณที่ใช้เลือกชุดของหน่วยความจำในส่วนนี้ได้แก่สัญญาณ Memory Select โดยสัญญาณควบคุมนี้จะส่งไปยัง DEMUX เพื่อเลือกหน่วยความจำที่จะถูกเขียนข้อมูลลงไป สำหรับข้อมูลที่เข้ามาเป็นข้อมูลขนาดที่มีขนาด 16 ระนาบบิตและข้อมูลสัญญาณเครื่องหมาย ข้อมูลที่เข้ามาจะเข้ามาทีละ1 คอลัมน์ของ Stripe หรือทีละ 4 แถว (Row 0, Row 1, Row 2 และ Row 3 ของแต่ละ Stripe) โดยเริ่มตั้งแต่คอลัมน์แรกของ Stripe แรก เรื่อยไปจนถึงคอลัมน์สุดท้ายของ Stripe แรก แล้วต่อด้วยข้อมูลในคอลัมน์แรกของ Stripe ถัดไป เรื่อยไปจนถึงคอลัมน์สุดท้ายของ Stripe สุดท้าย ครบทั้ง Code-block
2) ชุดของหน่วยความจำ ประกอบไปด้วย หน่วยความจำที่ใช้เก็บข้อมูลเครื่องหมาย ขนาด 1024 x 4 บิต และหน่วยความจำที่ใช้เก็บข้อมูลค่าสัมบูรณ์ทั้ง 16 ระนาบบิต จำนวน 4 ตัว ซึ่งในแต่ละตัวของหน่วยความจำที่ใช้เก็บข้อมูลค่าสัมบูรณ์จะมีขนาด 1024 x 16 บิต
3) ส่วนมัลติเพลกซ์ข้อมูล (MUX) ใช้เลือกข้อมูลในแต่ละตำแหน่งเพื่อส่งให้กับส่วนประมวลผล ซึ่งในส่วนนี้ จะเลือกชุดของหน่วยความจำ A หรือ B ขึ้นอยู่กับสัญญาณควบคุมที่ส่งเข้ามา ข้อมูลที่ออกมาจากหน่วยความจำ A และ B ในส่วนของข้อมูลเครื่องหมายมีขนาด 4 บิตหรือ 4 แถว สำหรับข้อมูลค่าสัมบูรณ์ที่มาจากหน่วยความจำทั้ง A และ B มีขนาด 4 แถวโดยในแต่ละแถวข้อมูลจะมีระนาบบิตทั้งสิ้น 16 ระนาบบิต ข้อมูลที่จะถูกนำไปประมวลผลในส่วนของการเข้ารหัสนั้นจะเป็นข้อมูลเพียงระนาบบิตเดียว ดังนั้นจึงต้องมีการเลือกข้อมูลในระนาบบิตใดๆด้วยสัญญาณ PlaneSelected ทำให้ข้อมูลที่ผ่านจาก MUX ตัวที่สองมีขนาด 4 บิตซึ่งในแต่ละบิตเป็นข้อมูลที่มาจากแต่ละแถวของข้อมูล

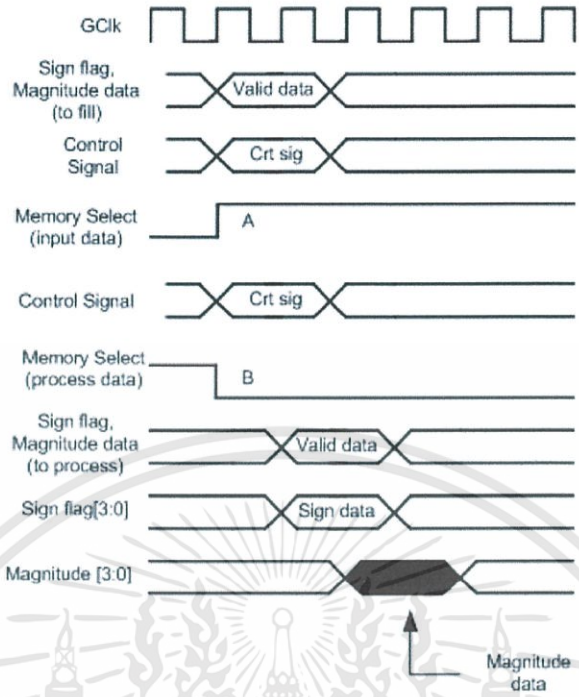
ในส่วนของพื้นที่ของหน่วยความจำที่ใช้เก็บข้อมูลค่าสัมบูรณ์ประกอบไปด้วยหน่วยความจำขนาด 1024 ตำแหน่ง x 16 บิต ทั้งสิ้น 4 ชุด (Mem 1024x16) ซึ่งในระบบโครงสร้างที่นำเสนอได้ใช้ระบบหน่วยความจำคู่ ดังนั้นในระบบโครงสร้างที่นำเสนอจะประกอบไปด้วยหน่วยความจำที่ใช้เก็บข้อมูลค่าสัมบูรณ์ขนาด 1024 ตำแหน่ง x 16 บิต ของหน่วยความจำชุด A ทั้งสิ้น 4 ตัว (สำหรับ 4 ค่า จากแต่ละแถวในคอลัมน์ที่ประมวลผล) และหน่วยความจำที่ใช้เก็บข้อมูลค่าสัมบูรณ์ชุด B ทั้งสิ้น 4 ตัว สำหรับหน่วยความจำที่ใช้เก็บข้อมูลเครื่องหมายขนาด 1024 ตำแหน่ง x 4 บิต ของหน่วยความจำชุด A จำนวน 1 ตัว และหน่วยความจำที่ใช้เก็บข้อมูลเครื่องหมายของหน่วยความจำชุด B จำนวน 1 ตัว

สำหรับกระบวนการในการเลือกชุดของหน่วยความจำจะมีสัญญาณควบคุมมาจากหน่วยควบคุม ซึ่งในหน่วยควบคุมการเข้ารหัสจะมีรีจิสเตอร์ที่ใช้จดจำว่าหน่วยความจำชุด A หรือ B ที่มีสถานะพร้อมสำหรับนำข้อมูลขาเข้ามาเก็บไว้และสัญญาณนี้มีความสัมพันธ์กับการเข้ารหัสด้วยเพื่อเลือกว่าหน่วยความจำชุด A หรือ B ที่พร้อมสำหรับการเข้ารหัส

สมมุติว่าในขณะที่หน่วยความจำชุด B พร้อมสำหรับการเข้ารหัส และในขณะเดียวกันหน่วยความจำชุด A มีสถานะพร้อมสำหรับนำข้อมูลขาเข้ามาเก็บ สัญญาณ Memory Select ของ Insert Data จะมีค่าเป็น '1' เพื่อเลือกหน่วยความจำชุด A สำหรับเก็บข้อมูลขาเข้าใหม่ ในส่วนของสัญญาณ Memory Select ของ Process Data จะมีค่าเป็น '0' เพื่อเลือกหน่วยความจำชุด B ไปใช้ในการเข้ารหัส ดังในรูปที่ 4.13

ในส่วนของตัว MUX (16 to 1) ใช้สำหรับใช้เลือกสัญญาณข้อมูลสัมบูรณ์ที่มาจากหน่วยความจำในขณะที่ทำการเข้ารหัสข้อมูล จากกระบวนการตามมาตรฐานการเข้ารหัสระนาบบิตการเข้ารหัสข้อมูลจะถูกเข้ารหัสตั้งแต่ระนาบบิตที่มีนัยสำคัญมากที่สุดเรื่อยไปจนถึงระนาบบิตที่มีนัยสำคัญน้อยที่สุดซึ่งจะเป็นการเลือกระนาบบิตไปทีละระนาบบิต ดังนั้นในส่วนของ MUX (16 to 1) ตัวนี้จะเป็นการเลือกข้อมูลตามระนาบบิตตามสัญญาณการเลือกระนาบบิตจากส่วนการเข้ารหัสส่งสัญญาณมาควบคุมระนาบบิตตามที่ต้องการ สัญญาณข้อมูลที่เข้าไปในตัว MUX จะมีขนาด 16 ระนาบบิต เมื่อผ่านส่วนของตัว MUX นี้จะได้สัญญาณออกมาเพียง 1 ระนาบบิต จึงทำให้ต้องมี MUX (16 to 1) ทั้งสิ้น 4 ตัวสำหรับข้อมูลใน 1 Stripe (4 แถว)

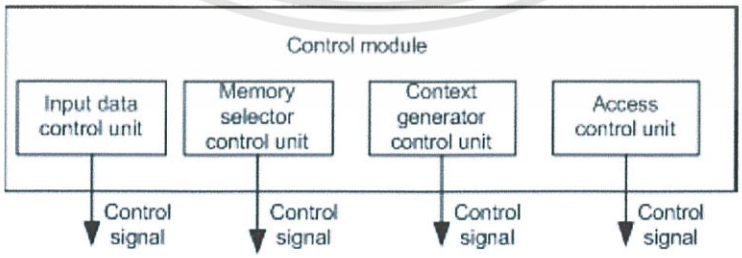
จากตัวอย่างสัญญาณในรูปที่ 4.13 แสดงขั้นตอนในการเขียนข้อมูลค่าสัมบูรณ์จากส่วนการเชื่อมต่อทางด้านอินพุตและการอ่านค่าจากหน่วยความจำไปใช้ในการเข้ารหัส การเขียนข้อมูลค่าสัมบูรณ์ลงในหน่วยความจำชุด A จะทำไปพร้อมกับกับการนำข้อมูลจากหน่วยความจำชุด B ไปประมวลผล จะเห็นได้ว่าในขณะที่ข้อมูลของ Code-block ปัจจุบันกำลังประมวลผลนั้น ข้อมูลสำหรับการประมวลผลชุดถัดไปก็ถูกเขียนเตรียมพร้อมไว้ในหน่วยความจำชุด A เรียบร้อยแล้ว ด้วยระบบการทำงานแบบระบบหน่วยความจำคู่นี้ สามารถลดความล่าช้าสำหรับรอข้อมูล ทำให้การประมวลผลข้อมูลมีประสิทธิภาพเพิ่มมากขึ้น



รูปที่ 4.13 แสดงผังเวลาลำดับขั้นการทำงานส่วนการจัดเก็บและการเข้าถึงข้อมูลแบบคู่

4.3.3 โครงสร้างส่วนการควบคุมการทำงานหลักของระบบ (Control Module)

โครงสร้างส่วนการควบคุมการทำงานหลักของระบบ เป็นโมดูลที่สร้างสัญญาณควบคุมการทำงานให้แก่โมดูลต่างๆ เพื่อให้ระบบสามารถทำงานสัมพันธ์กัน โดยมีทั้งส่วนสัญญาณในการควบคุมการทำงานของส่วนการเชื่อมต่อข้อมูลทางด้านอินพุต สัญญาณควบคุมการทำงานของหน่วยความจำแบบคู่ สัญญาณควบคุมการทำงานในการเข้ารหัสข้อมูล และส่วนการจัดเก็บและการเข้าถึงค่าสถานะในการเข้ารหัส



รูปที่ 4.14 แสดงโครงสร้างส่วนที่ใช้ในการควบคุมการทำงานหลักของระบบ

จากรูปที่ 4.14 แสดงส่วนที่ใช้สร้างสัญญาณควบคุมการทำงานหลักของระบบ ซึ่งภายในโมดูลประกอบด้วยส่วนต่างๆ ดังนี้

ส่วนสร้างสัญญาณสำหรับควบคุมกระบวนการรับข้อมูล (Input Data Control Unit): ในส่วนนี้ใช้สำหรับสร้างสัญญาณที่ใช้ควบคุมการทำงานของส่วนการเชื่อมต่อข้อมูลทางด้านอินพุต โดยสัญญาณที่สร้างขึ้นประกอบไปด้วย สัญญาณที่ใช้ในการอ่านข้อมูลสัมประสิทธิ์เวฟเลทจาก FIFO และข้อมูล Subband จาก FIFO เพื่อคอยควบคุมการทำงานในส่วนของกระบวนการในการนำเข้าข้อมูลให้มีความสัมพันธ์กับส่วนการทำงานต่างๆ ของระบบโครงสร้างที่นำเสนอ

ส่วนสร้างสัญญาณควบคุมการทำงานของส่วนการจัดเก็บและการเข้าถึงข้อมูลแบบคู่ (Memory Selector Control Unit): ในส่วนนี้ใช้สร้างสัญญาณสำหรับการควบคุมชุดหน่วยความจำ สัญญาณตำแหน่งสำหรับเก็บข้อมูลค่าสัมบูรณ์ สัญญาณสำหรับการเข้าถึงข้อมูลในหน่วยความจำ และสัญญาณควบคุมการเขียนข้อมูลลงหน่วยความจำ เป็นต้น ซึ่งสัญญาณที่สร้างจากส่วนนี้มีความสัมพันธ์กับการเข้ารหัสข้อมูลด้วย

ส่วนสร้างสัญญาณควบคุมการเข้ารหัสข้อมูล (Context Generator Control Unit) : ในส่วนนี้ใช้สร้างสัญญาณควบคุมที่บ่งชี้ถึงความถูกต้องของลำดับในการประมวลผลการเข้ารหัสข้อมูล รวมไปถึงการควบคุมการสร้างสัญญาณค่าสถานะในการเข้ารหัส โดยจะควบคุมให้ส่วนการเข้ารหัส เริ่มทำการเข้ารหัสตั้งแต่ระนาบบิตที่มีนัยสำคัญมากที่สุดเรื่อยไปจนถึงระนาบบิตที่มีนัยสำคัญน้อยที่สุด สัญญาณควบคุมนี้จะต้องทำการควบคุมให้การเข้ารหัสในแต่ละพาสเป็นไปตามลักษณะการประมวลผลแบบส่งต่อ โดยข้อมูลในระนาบบิตที่มีนัยสำคัญมากที่สุดจะเริ่มเข้ารหัสใน CUP ซึ่งเป็นส่วนของการทำงานแบบส่งต่อกันไป ส่วนสุดท้ายเพียงส่วนเดียว ส่วนข้อมูลในระนาบบิตอื่นๆ จะถูกเข้ารหัสตามลำดับเริ่มตั้งแต่ SPP, MRP และไปสิ้นสุดที่ CUP เพื่อเป็นไปตามลำดับในการเข้ารหัสของข้อมูลอย่าง

ส่วนสร้างสัญญาณควบคุมตำแหน่งสำหรับหน่วยความจำสถานะ (Access Control Unit): ในส่วนนี้ใช้สำหรับสร้างสัญญาณตำแหน่งของข้อมูลที่แสดงค่าสถานะในการเข้ารหัส (Significant state; σ , Magnitude refinement state; σ') เพื่อนำสัญญาณข้อมูลสถานะนั้น มาใช้ในการเข้ารหัสได้ตรงกับตำแหน่งของข้อมูลที่ถูกประมวลผลและจัดเก็บสัญญาณค่าสถานะในการเข้ารหัสที่ผ่านการประมวลลงในหน่วยความจำ

สำหรับสถานะในการสร้างสัญญาณควบคุมการทำงานของระบบที่นำเสนอแบ่งออกเป็น 3 สถานะ คือ สถานะช่วงประมวลผลว่าง (Idle State), สถานะช่วงประมวลผลของหน่วยความจำชุด A (Positive Process State), และสถานะช่วงประมวลผลของหน่วยความจำชุด B (Negative Process State) โดยค่าสถานะทั้ง 3 สถานะนี้จะมีด้วยกัน 2 ชุด โดยชุดแรกเพื่อใช้สำหรับส่วนการรับข้อมูลทางด้านอินพุต อีกชุดจะใช้สำหรับควบคุมการเข้ารหัส สถานะช่วงประมวลผลว่างจะเกิดขึ้นก็ต่อเมื่อระบบโครงสร้างที่นำเสนอไม่มีข้อมูลให้ประมวลผลเข้ารหัส เมื่อระบบตรวจพบว่ามีข้อมูล

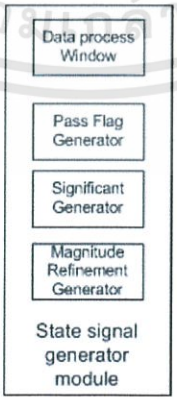
ถูกเขียนเข้ามาในระบบแล้ว สถานะของส่วนสร้างสัญญาณควบคุมจะเปลี่ยนไปสู่สถานะช่วงประมวลผลของหน่วยความจำชุด A เพื่อรองรับการเติมข้อมูลค่าสัมบูรณ์จนเสร็จสิ้นแล้วจึงเริ่มกระบวนการในการเข้ารหัสข้อมูลในหน่วยความจำชุด A ต่อ หากข้อมูลในบล็อกถัดไปถูกเขียนเข้ามาค่าสถานะของส่วนสร้างสัญญาณควบคุมส่วนการจัดเก็บข้อมูลและการเข้าถึงข้อมูลแบบคู่ในฝั่งการเขียนข้อมูลจะเปลี่ยนเป็นสถานะช่วงการประมวลผลของหน่วยความจำชุด B ในขณะที่ฝั่งการอ่านข้อมูลเพื่อนำไปประมวลผลการเข้ารหัสยังคงเป็นสถานะช่วงประมวลผลของหน่วยความจำ A

จะเห็นได้ว่าค่าสถานะในการสร้างสัญญาณควบคุมการทำงานของระบบจะสลับไปมาระหว่างที่ข้อมูลถูกเขียนเข้ามาและช่วงข้อมูลถูกนำไปประมวลผลการเข้ารหัส ทำให้ระบบที่นำเสนอนี้สามารถทำงานได้อย่างลื่นไหลไม่เกิดการติดขัดของการประมวลผลการเข้ารหัสข้อมูลแต่อย่างใด

4.3.4 โครงสร้างส่วนการสร้างสัญญาณแสดงสถานะในการเข้ารหัส (State Signal Generator Module)

ในส่วนของโมดูลนี้ใช้สำหรับเตรียมข้อมูลค่าสถานะที่จำเป็นต่อการเข้ารหัสไว้ล่วงหน้าก่อน โดยจะมีส่วนทำงานย่อยทั้งสิ้น 4 ส่วนด้วยกันคือ ส่วนเตรียมข้อมูลที่จะถูกเข้ารหัส (Data Process Window), ส่วนการสร้างสัญญาณบ่งบอกพาส (Pass Flag Generator), ส่วนสร้างสัญญาณค่าสถานะนัยสำคัญ (Significant Generator), และส่วนสร้างสัญญาณค่าสถานะ Magnitude refinement

เนื่องจากการทำงานในส่วนการเข้ารหัสมีความซับซ้อนและสัมพันธ์เกี่ยวเนื่องกันในแต่ละเส้นทางการเข้ารหัส การเตรียมข้อมูลค่าสถานะที่จำเป็นไว้ก่อนจะช่วยให้ลดความซับซ้อนในการเข้ารหัสลง ทำให้กระบวนการในการเข้ารหัสสามารถทำงานได้ง่ายและมีประสิทธิภาพสูงขึ้นได้

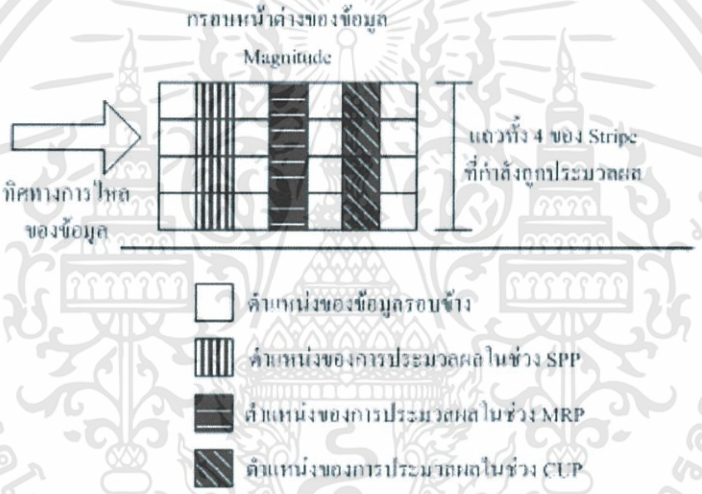


รูปที่ 4.15 แสดงโครงสร้างส่วนการสร้างสัญญาณแสดงสถานะในการเข้ารหัส

โครงสร้างของส่วนการสร้างสัญญาณแสดงสถานะในการเข้ารหัสดังที่ได้แสดงในรูปที่ 4.15 ในส่วนของรายละเอียดการทำงานและโครงสร้างภายใน จะกล่าวในหัวข้อถัดไป

4.3.4.1 โครงสร้างส่วนกรอบหน้าต่างการประมวลผล (Data Process Window)

โครงสร้างของส่วนกรอบหน้าต่างการประมวลผล (Data Process Window) เป็นโครงสร้างที่เป็นเมตริกซ์ของรีจิสเตอร์ขนาด 7 คอลัมน์ x 4 แถว เพื่อใช้สำหรับส่งต่อข้อมูลไปตามลำดับ ตั้งแต่ส่วนการประมวลผลของ SPP, MRP เรื่อยไปจนสิ้นสุดที่ CUP โดยข้อมูลที่จะถูกส่งต่อตามลำดับ คือ ข้อมูลขนาดหรือค่าสัมบูรณ์ของแต่ละระนาบบิตที่ได้มาจากการอ่านข้อมูลในหน่วยความจำ ดังแสดงในรูปที่ 4.16



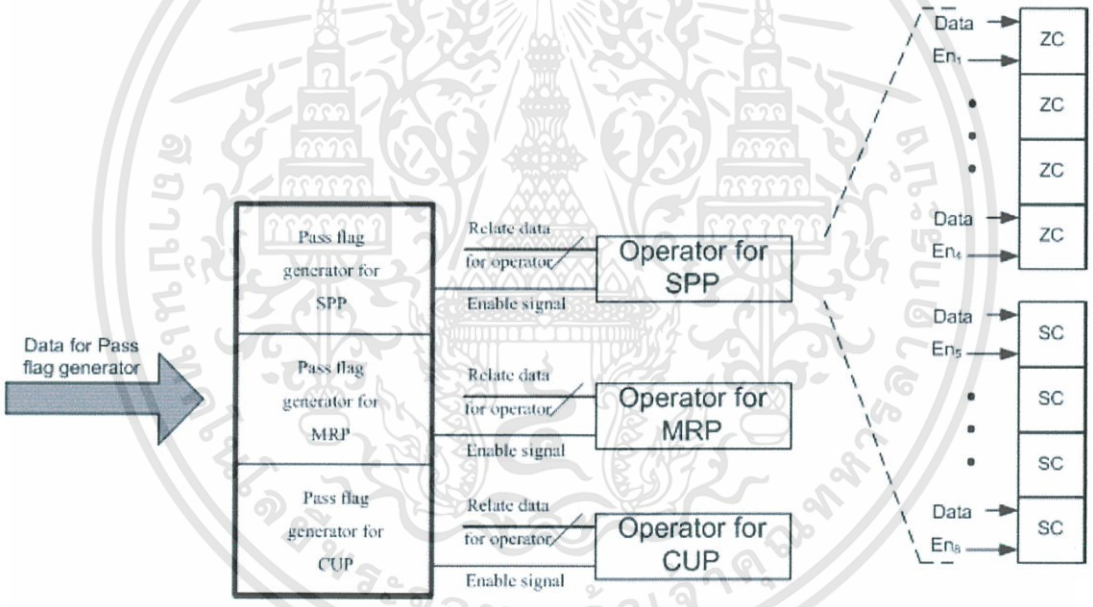
รูปที่ 4.16 แสดงกรอบหน้าต่างของการประมวลผลข้อมูล Magnitude

จากรูปที่ 4.16 จะเห็นได้ว่าข้อมูลที่อยู่ในกรอบการประมวลผลจะมีจำนวนทั้งสิ้น 7 คอลัมน์ แต่ข้อมูลที่จะถูกประมวลผลจริงในแต่ละพาสจะมีเพียง 3 คอลัมน์ (คอลัมน์ที่ถูกแรเงา) ส่วนในตำแหน่งของคอลัมน์ที่ไม่ได้แรเงานั้นจะไม่ถูกประมวลผล เนื่องจากว่าการทำงานในส่วนของการสร้างสัญญาณสถานะของการเข้ารหัสนั้นจะมีค่าของสถานะที่จะต้องถูกเปลี่ยนแปลงและค่าข้อมูลสถานะของการเข้ารหัสรอบข้างที่ต้องนำมาประมวลผล เพื่อให้ข้อมูลขนาดมีความสอดคล้องกับการสร้างและปรับเปลี่ยนค่าสถานะของการเข้ารหัส จึงทำให้การออกแบบหน้าต่างของการประมวลผลข้อมูลให้มีจำนวนของคอลัมน์และแถวมีขนาดเท่ากับหน้าต่างของค่าสถานะของการเข้ารหัส ในส่วนของกรอบหน้าต่างการประมวลผลข้อมูลขนาดนี้ ข้อมูลจะไม่ถูกเก็บลง

หน่วยความจำและข้อมูลทุกตำแหน่งจะไม่มีการถูกปรับเปลี่ยนค่าแต่อย่างใด เป็นเพียงการนำค่าข้อมูลขนาดมาใช้ในการเข้ารหัสอย่างเดียวเท่านั้น

4.3.4.2 โครงสร้างส่วนการสร้างสัญญาณบ่งบอกช่วงพาส (Pass Flag Generator)

สำหรับโครงสร้างในส่วนนี้ทำหน้าที่สร้างสัญญาณที่บ่งชี้พาสของข้อมูลในแต่ละตำแหน่งของกรอบหน้าต่างการประมวลผล เพื่อให้การเข้ารหัสข้อมูลในแต่ละตำแหน่งถูกต้องตามกระบวนการบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000 และให้ระบบมีประสิทธิภาพในการประมวลผลมากยิ่งขึ้น การสร้างสัญญาณในส่วนนี้จึงออกแบบให้สามารถทำงานพร้อมๆ กันในทุกๆ แถวของแต่ละพาสการเข้ารหัส พร้อมกันนั้นยังได้อาศัยหลักการในการทำงานแบบส่งต่อการประมวลผลร่วมด้วย



รูปที่ 4.17 แสดงสัญญาณพาสที่ใช้ควบคุมตัวดำเนินการพื้นฐานการเข้ารหัสในแต่ละพาส

จากรูปที่ 4.17 แสดงส่วนสร้างสัญญาณพาส เพื่อนำไปใช้ควบคุมการทำงานของตัวดำเนินการพื้นฐานของแต่ละพาสการเข้ารหัส เมื่อสัญญาณพาสใดถูกสร้างออกมาจะไปสั่งให้ตัวดำเนินการพื้นฐานการเข้ารหัสประมวลผลข้อมูลให้ได้ผลลัพธ์สัญญาณคู่อันดับสัญลักษณ์ออกมา หากตัวดำเนินการพื้นฐานการเข้ารหัสตัวใดไม่มีสัญญาณสั่งให้ทำงานมาจากส่วนสร้างสัญญาณบ่งบอกช่วงพาส ตัวดำเนินการพื้นฐานการเข้ารหัสเหล่านั้นจะไม่ประมวลผลข้อมูลให้เกิดผลลัพธ์สัญญาณคู่อันดับสัญลักษณ์ ซึ่งจากในรูปที่ 4.17 เป็นตัวอย่างของตัวดำเนินการพื้นฐานแต่ละตัว

สำหรับการเข้ารหัส SPP ซึ่งจะมีสัญญาณที่ใช้สำหรับสั่งการให้ตัวดำเนินการพื้นฐานทำงาน คือ สัญญาณ $En_1 - En_8$

จากงานวิจัย [9] ได้นำเสนออัลกอริธึมในการทดสอบความเป็นสมาชิกของพาสการเข้ารหัสได้ ดังนั้นในระบบที่นำเสนอนี้จึงได้นำอัลกอริธึมนั้นมาเป็นพื้นฐานในการออกแบบ และเพื่อแก้ไขปัญหาของการทำงานของการเข้ารหัสข้อมูลที่ต้องประมวลผลข้อมูลซ้ำซ้อน การทำงานแบบส่งต่อจึงเป็นวิธีการที่เพิ่มเข้าไปเพื่อเสริมการทำงานของการสร้างสัญญาณบ่งบอกช่วงการเข้ารหัส สำหรับคำสั่งเทียมที่งานวิจัย [9] ได้นำเสนอแสดงได้ดังรูปที่ 4.18

จากรูปที่ 4.18 แสดงคำสั่งเทียมของเงื่อนไขที่ใช้ในการตรวจสอบข้อมูลในตำแหน่งปัจจุบันเพื่อใช้สร้างสัญญาณบ่งชี้พาสการประมวลผล ถ้าข้อมูลตรงตามเงื่อนไขการประมวลผลของ SPP ค่าพาสที่ได้จากคำสั่งเทียมจะมีค่าเป็น SPP หากตรงเงื่อนไขการประมวลผลของ MRP ค่าพาสในคำสั่งเทียมจะเป็นค่า MRP ส่วน CUP คำสั่งเทียมจะให้ค่าพาสเป็น CUP

ซึ่งเห็นได้ว่าการตรวจสอบว่าข้อมูลในตำแหน่งปัจจุบันทั้งคอลัมน์ Stripe สามารถถูกตรวจสอบเงื่อนไขได้อย่างพร้อมๆ กัน แต่การทำงานของตัวดำเนินการพื้นฐานการเข้ารหัสใน SPP และ CUP ของงานวิจัย [9] ในส่วนของ ZC และ SC จะเป็นการใช้ร่วมกัน เพื่อเป็นการลดทรัพยากร ดังนั้นระบบโครงสร้างของงานวิจัย [9] จะต้องประมวลผลการเข้ารหัสไปทีละพาสให้เสร็จสิ้นก่อนทั้งระนาบบิตแล้วจึงจะเริ่มการประมวลผลในพาสถัดไปได้ อันเป็นสาเหตุให้ข้อมูลใน 1 ระนาบบิตจะถูกประมวลผลโดยใช้สัญญาณนาฬิกาเป็นจำนวนถึง 3 เท่าของข้อมูลที่ถูกนำไปประมวลผล เช่น ข้อมูลมีปริมาณ 64 จุด x 64 จุด ต้องใช้สัญญาณนาฬิกาประมาณ 64 จุด x 16 Stripe = 1024 ลูกสัญญาณนาฬิกา แต่ในงานวิจัย [9] จะต้องใช้สัญญาณนาฬิกาประมาณ 64 จุด x 16 Stripe x 3 พาส = 3072 ลูกสัญญาณนาฬิกา

แต่ในระบบโครงสร้างที่นำเสนอจะทำการประมวลผลขนานพร้อมกันทุกพาสและยังนำการทำงานแบบส่งต่อมาประยุกต์ใช้ร่วมกัน ทำให้สัญญาณนาฬิกาที่ใช้ในการประมวลผลของข้อมูลใน 1 ระนาบบิต จะใช้สัญญาณนาฬิกาประมาณ 1024 ลูกสัญญาณนาฬิกา เนื่องจากข้อมูลในตำแหน่งที่ยังไม่ถูกประมวลผลหรือสัญญาณบ่งชี้พาส ไม่ได้ถูกสร้างในช่วงพาสก่อนหน้า ในสัญญาณนาฬิกาถัดไปข้อมูลเหล่านั้นจะถูกส่งต่อไปประมวลผลในพาสถัดไปทันที ซึ่งในขณะเดียวกันนั้นข้อมูลใหม่ในคอลัมน์ถัดไปก็จะถูกประมวลผลในพาสก่อนหน้าด้วย เช่นจากรูปที่ 4.16 ในขณะที่ข้อมูลในคอลัมน์ที่ 5 กำลังประมวลผลใน CUP ข้อมูลในคอลัมน์ที่ 7 จะถูกประมวลผลใน MRP และข้อมูลในคอลัมน์ที่ 9 จะถูกประมวลผลใน SPP ขนานพร้อมกันไป จะเห็นได้ว่าในสัญญาณนาฬิกาเดียวกัน ข้อมูลในแต่ละคอลัมน์ของระนาบบิตเดียวกันจะถูกประมวลผลใน 3 พาสการทำงานพร้อมกัน และเมื่อสัญญาณนาฬิกาถัดไปข้อมูลจะถูกเลื่อนส่งต่อเนื่องไปเพื่อให้ทุกตำแหน่งถูกตรวจสอบการทำงานในแต่ละช่วงพาสเสมอ ทำให้สามารถลดความซ้ำซ้อนของการ

ประมวลผลที่ต้องประมวลผลให้เสร็จสิ้นทั้งระนาบบิตในพาสใดหนึ่งก่อนแล้วค่อยประมวลผลในพาสถัดไปได้

ตำแหน่งแถวที่ 0:

$If(\sigma[0])\ Pass = MRP;$

$Elseif(\vec{K}_{\sigma}[0])\ Pass = SPP;$

$Else\ Pass = CUP;$

ตำแหน่งแถวที่ 1:

$if(\sigma[1])\ Pass = MRP;$

$Elseif(\vec{K}_{\sigma}[1])\ Pass = SPP;$

$Elseif(\vec{K}_{\sigma}[0] \& v[0])\ Pass = SPP;$

$Else\ Pass = CUP;$

ตำแหน่งแถวที่ 2:

$If(\sigma[2])\ Pass = MRP;$

$Elseif(\vec{K}_{\sigma}[2])\ Pass = SPP;$

$Elseif(((\vec{K}_{\sigma}[0] \& v[0]) | \vec{K}_{\sigma}[1]) \& v[1])\ Pass = SPP;$

$Else\ Pass = CUP;$

ตำแหน่งแถวที่ 3:

$If(\sigma[3])\ Pass = MRP;$

$Elseif(\vec{K}\sigma[3] | \sigma[3] | \sigma[2])\ Pass = SPP;$

$Elseif(((((\vec{K}_{\sigma}[0] \& v[0]) | \vec{K}_{\sigma}[1]) \& v[1]) | \vec{K}_{\sigma}[2]) \& v[2])\ Pass = SPP;$

$Else\ Pass = CUP;$

รูปที่ 4.18 แสดงคำสั่งเทียมที่ใช้ในการสร้างสัญญาณบ่งชี้พาสจากงานวิจัย [9]

งานวิจัยนี้จึงทำการปรับสมการจากคำสั่งเทียมให้อยู่ในรูปแบบทางตรรกะเพื่อให้สามารถสร้างสัญญาณสมาชิกของพาสของทุกแถวในคอลัมน์ที่กำลังเข้ารหัสได้พร้อมกัน สามารถแสดงได้ดังด้านล่าง

SPP:

$$P_{flag}[0]^{t_n}_{SPP} = \bar{K}_{\sigma}[0] \bullet \bar{\sigma}[0]; \tag{4.1}$$

สมการที่ (4.1) ใช้ในการตรวจสอบข้อมูลในตำแหน่งแถวที่ 0 ว่าจะต้องถูกประมวลผลใน SPP หรือไม่ หากข้อมูลในตำแหน่งดังกล่าวต้องถูกประมวลผลใน SPP ค่าของ $P_{flag}[0]^{t_n}_{SPP}$ (สัญญาณบ่งบอกความเป็นสมาชิกของ SPP ณ. เวลาปัจจุบัน t_n) จะมีค่าเป็น 1 หากข้อมูลในแถวที่ 0 ไม่ถูกประมวลผลใน SPP ค่านี้จะเป็น '0' โดยค่า $P_{flag}[0]^{t_n}_{SPP}$ สามารถหาได้จากค่าสถานะนัยสำคัญรอบข้างของตำแหน่งแถวที่ 0 พิจารณาทั้ง 8 ทิศรอบตัว ($\bar{K}_{\sigma}[0]$) และค่าของสถานะนัยสำคัญของตำแหน่งแถวที่ 0 ($\bar{\sigma}[0]$) นั่นคือ ถ้าตำแหน่งรอบข้างมีความสำคัญคือมีค่าสถานะนัยสำคัญรอบข้าง $\bar{K}_{\sigma}[0] = 1$ แต่ตำแหน่งแถวที่ 0 ที่กำลังเข้ารหัสไม่มีความสำคัญคือ มีสถานะนัยสำคัญ $\bar{\sigma}[0]=0$ จะทำให้ค่า $P_{flag}[0]^{t_n}_{SPP} = 1$ คือ มีการประมวลผลใน SPP เป็นต้น

$$P_{flag}[i]^{t_n}_{SPP} = \left(\bar{K}_{\sigma}[i] \mid \left(P_{flag}[i-1]_{SPP} \bullet \nu[i-1]\right)\right) \bullet \bar{\sigma}[i] \qquad ; i \in \{1,2,3\} \tag{4.2}$$

ในสมการที่ (4.2) เป็นสมการที่ใช้สำหรับสร้างสัญญาณบ่งบอกว่าข้อมูลในตำแหน่งแถวที่ 1 ถึงแถวที่ 3 ในช่วง SPP ว่าข้อมูลในตำแหน่งดังกล่าวจะถูกเข้ารหัสหรือไม่ ซึ่งในสมการจะมีลักษณะที่ใกล้เคียงกับสมการที่ (4.1) เพียงแต่จะพิจารณาผลของสัญญาณบ่งบอกการเข้ารหัส SPP ของแถวก่อนหน้า ($P_{flag}[i-1]_{SPP}$) และค่าสัญญาณข้อมูลขนาด ในตำแหน่งของแถวก่อนหน้า ($\nu[i-1]$) ด้วย เนื่องจากการปรับค่าสถานะนัยสำคัญจากแถวก่อนหน้าจะส่งผลต่อค่าสถานะนัยสำคัญรอบข้างของตำแหน่งแถวที่กำลังเข้ารหัสด้วย

เนื่องจากการเข้ารหัสจะทำงานทุกแถวพร้อมกัน ถ้าค่าสถานะนัยสำคัญรอบข้างของข้อมูลในตำแหน่งแถวที่ 1 มีค่าเป็น $\bar{K}_{\sigma}[1] = 0$ แต่ว่าค่าข้อมูลในตำแหน่งแถวที่ 0 ถูกเข้ารหัสและมีการปรับค่าของสถานะนัยสำคัญในตำแหน่งแถวที่ 0 ($\sigma[0] =1$) จะส่งผลให้ $\bar{K}_{\sigma}[1]$ เปลี่ยนจากค่า '0' เป็น '1' ซึ่งจะทำให้ข้อมูลในตำแหน่งแถวที่ 1 นี้จะถูกเข้ารหัสใน SPP ด้วย ในทำนองเดียวกันกับแถวที่ 2 และแถวที่ 3 ซึ่งพจน์ที่ใช้พิจารณาค่าสถานะนัยสำคัญของแถวก่อนหน้า ที่อาจจะส่งผลกระทบต่อตำแหน่งแถวถัดไป นี้ได้แก่ พจน์ $P_{flag}[i-1]_{SPP} \bullet \nu[i-1]$ (รูปที่ 4.1 กรณีที่ 3)

MRP:

$$P_{flag}[i]^{t_n}_{MRP} = \sigma[i] \bullet \overline{\eta}[i]^{t_n}_{SPP} \qquad ; i \in \{0,1,2,3\} \tag{4.3}$$

ในสมการที่ (4.3) เป็นสมการที่ใช้สร้างสัญญาณบ่งบอกความเป็นสมาชิกของ MRP โดยสามารถหาได้จากการตรวจสอบค่าสถานะนัยสำคัญในตำแหน่งข้อมูลปัจจุบัน ($\sigma[i]$) กับค่าสถานะของการเข้ารหัสของ SPP (Coding State; $\overline{\eta}[i]^{t_n}_{SPP}$) ในแต่ละตำแหน่งแถวของข้อมูล หากข้อมูลยังไม่ถูกเข้ารหัสจาก SPP มาก่อน ค่าสถานะของการเข้ารหัส SPP มีค่าเป็น '0' พร้อมทั้งค่าสถานะนัยสำคัญของข้อมูลในแต่ละตำแหน่งมีค่าเป็น '1' ด้วยแล้ว ข้อมูลในตำแหน่งที่ตรงตามเงื่อนไขเหล่านั้นจะถูกประมวลผลใน MRP นั่นคือ $P_{flag}[i]^{t_n}_{MRP} = 1$

CUP:

$$P_{flag}[i]^{t_n}_{CUP} = \overline{\eta}[i]^{t_n}_{MRP} \qquad ; i \in \{0,1,2,3\} \tag{4.4}$$

สำหรับสมการที่ (4.4) เป็นสมการที่ใช้ในการสร้างสัญญาณบ่งบอกความเป็นสมาชิกของ CUP ซึ่งค่าของสัญญาณบ่งบอกความเป็นสมาชิกได้จากการตรวจสอบค่าสถานะของการเข้ารหัสว่าข้อมูลในแต่ละตำแหน่งได้ถูกประมวลผลผ่านมาแล้วหรือยัง หากยังไม่ถูกประมวลผลข้อมูลของพาสก่อนหน้า (MRP) ในตำแหน่งแถวนั้น ($\overline{\eta}[i]^{t_n}_{MRP}$) ข้อมูลในตำแหน่งดังกล่าวจะถูกประมวลผลใน CUP

จากสมการที่ (4.1) ถึงสมการที่ (4.4) สมการจะมีลักษณะการสร้างสัญญาณเหมือนกับงานวิจัย [9] เพียงแต่ว่าค่าสัญญาณที่ออกมาในสัญญาณนาฬิกา 1 ลูก จะไม่ใช่บ่งชี้เฉพาะว่าข้อมูลที่กำลังประมวลผลเป็นของช่วงการประมวลผลเฉพาะพาสนั้น แต่ภายในสัญญาณนาฬิกา 1 ลูก ข้อมูลใน 3 คอลัมน์ จะถูกตรวจสอบและประมวลผลในแต่ละพาสพร้อมกันในทีเดียว และด้วยเหตุนี้ทำให้ข้อมูลใน 1 ระนาบบิตจะถูกประมวลผลเสร็จสิ้นภายในจำนวนสัญญาณนาฬิกาประมาณ 1024 ลูกสัญญาณนาฬิกา

สมการที่ (4.1) ถึง (4.2) ใช้สำหรับสร้างสัญญาณบ่งชี้พาสของ SPP ซึ่งสมการที่สร้างสัญญาณพาส จะเป็นสัญญาณที่มีการส่งค่าสัญญาณต่อเนื่องกันไป นั่นคือค่าที่สร้างจากสมการที่ (4.1) จะถูกนำไปใช้งานในสมการที่ (4.2) เนื่องจากค่าของสัญญาณในแต่ละสมการมีความเกี่ยวข้องกัน และเพื่อเป็นการลดจำนวนทรัพยากรที่ถูกนำมาใช้ สัญญาณที่ได้จึงต้องมีการนำมาใช้ต่อๆกัน

สำหรับในส่วนของสมการที่ (4.3) เป็นสมการที่ใช้สร้างค่าสัญญาณบ่งชี้พาสของ MRP โดยสมการจะตรวจสอบค่าสถานะของการเข้ารหัสว่าข้อมูลในตำแหน่งเหล่านั้น ถูกประมวลผลใน SPP ก่อนหน้ามาหรือยัง หากข้อมูลในตำแหน่งใดยังไม่ถูกประมวลผลในพาสก่อนหน้าและค่าสถานะนัยสำคัญแสดงนัยสำคัญ ค่าสัญญาณบ่งชี้พาสของ MRP จะถูกสร้างขึ้น เมื่อเงื่อนไขเป็นไปตามข้อกำหนด

ส่วนสมการที่ใช้สร้างสัญญาณบ่งชี้พาสของ CUP คือสมการที่ (4.4) ซึ่งค่าของสัญญาณที่สร้างขึ้นมาเกิดจากการตรวจสอบสถานะของการเข้ารหัสของข้อมูลในแต่ละตำแหน่งของคอลัมน์ข้อมูล หากค่าสถานะของการเข้ารหัสข้อมูลในตำแหน่งใดยังไม่ได้ถูกประมวลผลข้อมูลใน SPP หรือ MRP ก่อนหน้า สัญญาณบ่งชี้พาสข้อมูลจะถูกสร้างในส่วนนี้

4.3.4.3 โครงสร้างส่วนสร้างสัญญาณสถานะบ่งบอกการเข้ารหัส (Coding State Generator)

โครงสร้างในส่วนนี้เป็นส่วนที่ใช้สร้างสัญญาณสถานะบ่งบอกการเข้ารหัส (Coding State; η) เพื่อเป็นการบ่งบอกว่าข้อมูลที่กำลังถูกเข้ารหัสในหน้าต่างการประมวลผลตำแหน่งใดได้ถูกเข้ารหัสเป็นที่เรียบร้อยแล้ว เนื่องจากระบบที่นำเสนอนี้ได้ออกแบบการประมวลผลโดยอาศัยกระบวนการทำงานแบบส่งต่อ ดังนั้นค่าสถานะที่ใช้บ่งบอกการเข้ารหัสจึงต้องออกแบบให้เป็นไปในทิศทางเดียวกันกับค่าสถานะของการเข้ารหัสตัวอื่น เพื่อให้ระบบสามารถทำงานสัมพันธ์กัน โดยในการติดตามการเปลี่ยนแปลงค่าสถานะของการเข้ารหัส ในงานวิจัยนี้ได้แบ่งการติดตามออกเป็น 2 ส่วน คือ การติดตามการเปลี่ยนแปลงของค่าสถานะของการเข้ารหัสที่เกิดขึ้นหลังจากการประมวลผลใน SPP และ MRP ($\hat{\eta}[i,j]^{t_n}_{SPP}, \hat{\eta}[i,j]^{t_n}_{MRP}$) และค่าสถานะของการเข้ารหัสหลักของระบบ ($\eta[i,j]^{t_{n+1}}$) ค่าสถานะของการเข้ารหัสที่ถูกปรับค่าหลังจากการประมวลผลใน SPP และ MRP ในช่วงเวลาการเข้ารหัส t_n จะถูกส่งไปปรับค่าสถานะของการเข้ารหัสหลักของระบบในเวลาถัดไป t_{n+1}

โดยการออกแบบส่วนสร้างสัญญาณสถานะบ่งบอกการเข้ารหัส จะออกแบบโดยการนำหลักการส่งต่อกันไปของข้อมูลมาใช้ ซึ่งในระบบโครงสร้างที่นำเสนอใช้รีจิสเตอร์ในการเก็บค่าสถานะของการเข้ารหัสที่มีขนาด 1 บิต เรียงต่อกันจำนวน 4 คอลัมน์ x 4 แถวสามารถแสดงได้ดังรูปที่ 4.19

SPP (ในช่วงเวลา t_n):

จากรูปที่ 4.19 ในส่วนของ $\hat{\eta}[i,j]^{t_n}_{SPP}$ เป็นตัวแปรแสดงค่าสถานะของการเข้ารหัสของ SPP (SPP Updated Coding State) ที่จะถูกปรับค่าเมื่อข้อมูลในตำแหน่งแถวที่ i คอลัมน์ที่ j หลังจากผ่านการประมวลผล SPP ของช่วงเวลาการประมวลผลปัจจุบัน (t_n) โดยค่าของ $\hat{\eta}[i,j]^{t_n}_{SPP}$ ได้มาจากสัญญาณบ่งบอกความเป็นสมาชิกของช่วง SPP ($P_{flag}[i]^{t_n}_{SPP}$) ของช่วงเวลาปัจจุบัน ดังนั้นอาจจะกล่าวได้ว่าค่าสถานะของการเข้ารหัสในเวลาถัดไปจะถูกปรับค่าด้วยค่าสัญญาณที่ได้จากสมการที่ (4.5)

$$\hat{\eta}[i,k]^{t_n}_{SPP} = P_{flag}[i]^{t_n}_{SPP} \qquad ;i \in \{0,1,2,3\}, k = j \tag{4.5}$$

ในส่วนของค่าที่บ่งบอกสถานะของการเข้ารหัสหลักของข้อมูลในตำแหน่งคอลัมน์ที่ j เมื่อเวลาถัดไปได้เกิดขึ้น (t_{n+1}) หรือค่า $\eta[i,j]^{t_{n+1}}$ จะถูกปรับเปลี่ยนค่าไปตามสมการที่ (4.6)

$$\eta[i,k]^{t_{n+1}} = \hat{\eta}[i,k]^{t_n}_{SPP} \qquad ;i \in \{0,1,2,3\}, k = j \tag{4.6}$$

อาจกล่าวได้ว่า ณ. เวลาถัดไปค่าสถานะของการเข้ารหัสหลักของคอลัมน์ที่ j ($\eta[i,j]^{t_{n+1}}$) จะถูกปรับค่าในเวลา t_{n+1} จะถูกเลื่อนไปทางขวามือ เพื่อรอประมวลผลใน MRP ในช่วงเวลา t_{n+2} ($\eta[i,j+1]^{t_{n+2}}_{MRP} = \eta[i,j+1]^{t_{n+2}}$) ถ้าค่าของสถานะของการเข้ารหัสมีค่าเป็น '1' ในตำแหน่งใด แสดงว่าข้อมูลในตำแหน่งดังกล่าวนั้นได้ถูกประมวลผลในช่วง SPP เป็นที่เรียบร้อยแล้ว

ในส่วนของสัญญาณสถานะของการเข้ารหัสในช่วง MRP ($\eta[i,k]^{t_n}_{MRP}$) จะถูกปรับค่าให้มีค่าเป็นไปตามสัญญาณสถานะของการเข้าของข้อมูลในตำแหน่งเดียวกันที่เวลา t_n หรืออาจจะกล่าวได้ว่าค่าสัญญาณสถานะของการเข้ารหัสในช่วง MRP เป็นค่าสัญญาณสถานะของข้อมูลในตำแหน่งเดียวกันที่ถูกเลื่อนมาให้ตรงตำแหน่งคอลัมน์ที่จะถูกประมวลผลในช่วง MRP ($\eta[i,j+1]^{t_{n+1}}_{MRP} = \eta[i,j+1]^{t_n}$)

MRP (ในช่วงเวลา t_n):

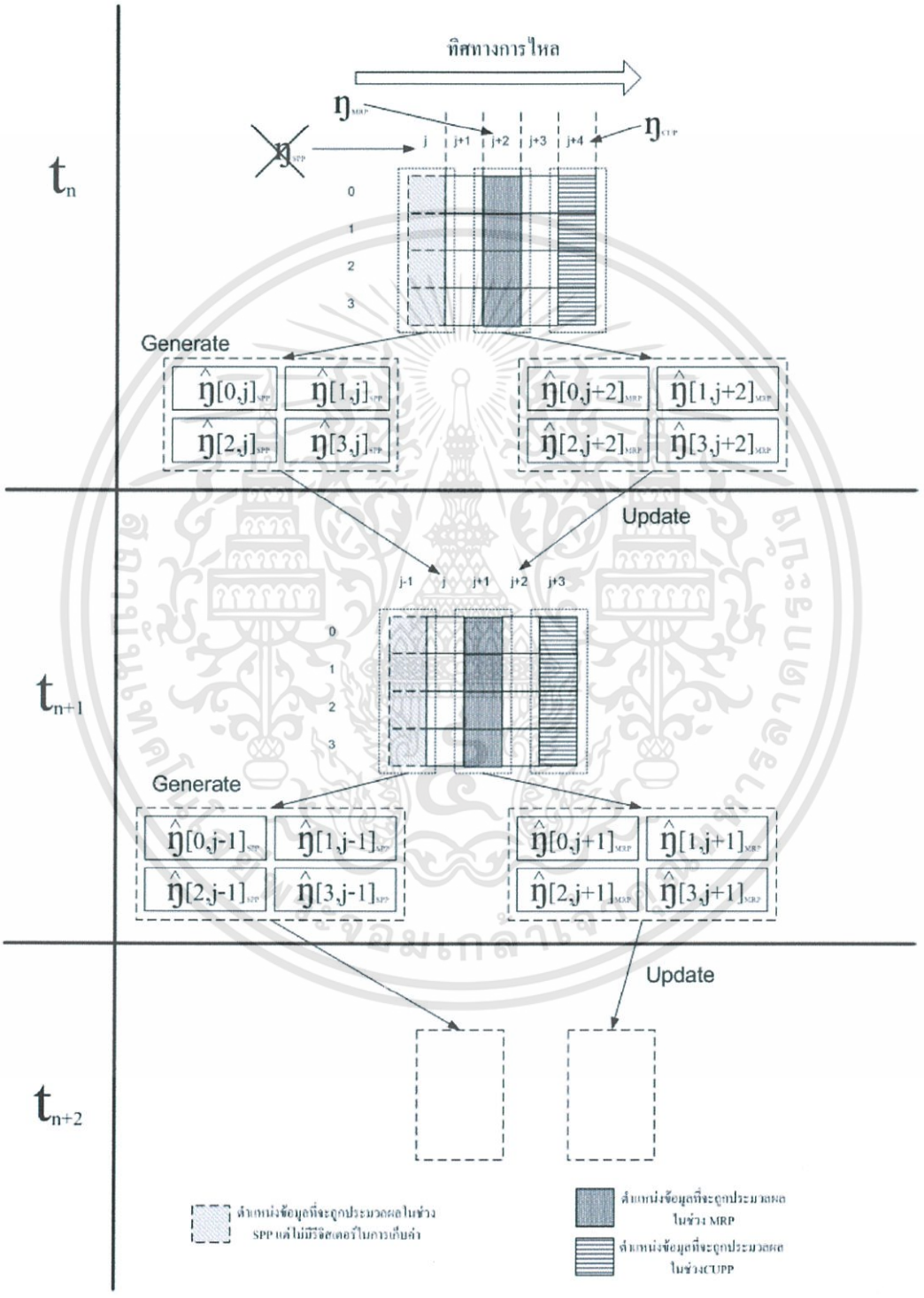
ส่วนของ $\hat{\eta}[i,j+2]^{t_n}_{MRP}$ เป็นค่าสถานะของการเข้ารหัสหลังผ่านช่วงการประมวลผลของ MRP ในช่วงเวลา t_n แล้ว โดยค่าสัญญาณนี้จะเป็นตัวที่ใช้ในการปรับค่าสถานะของการเข้ารหัสหลักในช่วงเวลาถัดไป t_{n+1} ซึ่งสามารถหาได้จากค่าสัญญาณสถานะของการเข้ารหัสที่ยังไม่ถูกปรับค่า ($\eta[i,j+2]^{t_n}_{MRP}$) และค่าสถานะบ่งบอกความเป็นสมาชิก ($P_{flag}[i]^{t_n}_{MRP}$) สามารถแสดงได้ดังสมการที่ (4.7)

$$\hat{\eta}[i,k]^{t_n}_{MRP} = P_{flag}[i]^{t_n}_{MRP} \mid \eta[i,k]^{t_n}_{MRP} \qquad ;i \in \{0,1,2,3\}, k = j+2 \tag{4.7}$$

นั่นคือ ค่า $\hat{\eta}[i,j+2]^{t_n}_{MRP}$ จะเป็น 1 จากเงื่อนไข 2 กรณีคือ

1. มีการเข้ารหัสเกิดขึ้นใน MRP ในช่วงเวลา t_n ที่ส่งผลให้ $P_{flag}[i]^{t_n}_{MRP} = 1$
2. มีการเข้ารหัสเกิดขึ้นใน SPP ในช่วงเวลา t_{n-2} ก่อนหน้า ที่ส่งผลต่อเนื่องให้ค่า $\eta[i,j+2]^{t_n}_{MRP} = 1$

ในทำนองเดียวกับสมการที่ (4.6) ค่าสถานะของการเข้ารหัสหลักของตำแหน่งคอลัมน์ที่ j+2 จะถูกปรับค่าในช่วงเวลา t_{n+1} คือ $\eta[i,j+2]^{t_{n+1}} = \hat{\eta}[i,j+2]^{t_n}_{MRP}$ เพื่อบ่งบอกว่าข้อมูลในตำแหน่งแถวใดในคอลัมน์ที่ j+2 ได้ถูกเข้ารหัสไปแล้วในช่วง MRP นี้บ้าง



รูปที่ 4.19 แสดงกรอบหน้าต่างของสัญญาณสถานะบ่งบอกการเข้ารหัส

CUP (ในช่วงเวลา t_n):

ส่วนค่าสัญญาณสถานะบ่งบอกการเข้ารหัสสำหรับข้อมูลในช่วง CUP ($\eta[i, j+4]^{t_n}_{CUP}$) ในช่วงเวลาต่อมา (t_n) จะได้ค่าสัญญาณบ่งบอกการเข้ารหัสหลัก ($\eta[i, j+4]^{t_n}$) สมการที่แสดงการสร้างสัญญาณสถานะบ่งบอกการเข้ารหัสสามารถแสดงได้ดังสมการที่ (4.8)

$$\eta[i,k]^{t_n}_{CUP} = \eta[i,k]^{t_{n-1}} \qquad ;i \in \{0,1,2,3\}, k = j+4 \tag{4.8}$$

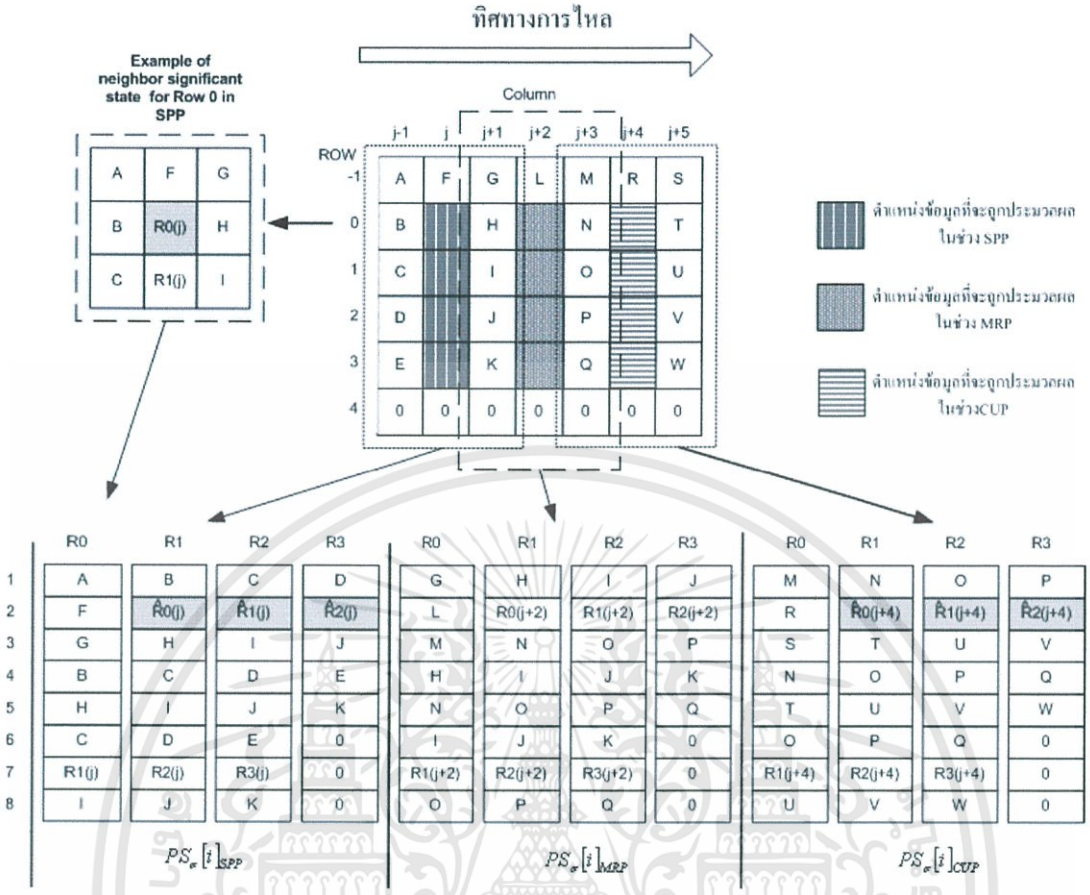
ค่าสถานะของการเข้ารหัสในช่วง CUP ($\eta[i, j+4]^{t_n}_{CUP}$) นี้จะถูกนำไปตรวจสอบ ถ้ามีค่าเป็น '1' แสดงว่าได้ถูกทำการเข้ารหัสจากช่วงก่อนหน้า (SPP หรือ MRP) แล้ว ดังนั้นจะไม่ถูกเข้ารหัสในช่วง CUP แต่ถ้าค่าสถานะของการเข้ารหัสในช่วง CUP มีค่าเป็น '0' แสดงว่ายังไม่ถูกเข้ารหัสจากช่วงก่อนหน้า ดังนั้นจะทำการเข้ารหัสในช่วง CUP นี้

เมื่อข้อมูลที่ถูกเข้ารหัสในช่วง CUP ซึ่งเป็นการเข้ารหัสพาสสุดท้าย ค่าสถานะของการเข้ารหัสจะไม่ถูกเขียนลงหน่วยความจำ เนื่องจากข้อมูลได้ถูกประมวลผลในทุกตำแหน่งเป็นที่เรียบร้อยแล้วต่างจากกระบวนการของงานวิจัย [9] ที่ยังต้องเก็บค่าสถานะของการเข้ารหัสไว้ในหน่วยความจำเนื่องจากการทำงานของงานวิจัย [9] จะประมวลผลข้อมูลไปทีละพาสทำให้ทรัพยากรของหน่วยความจำที่ต้องใช้เก็บค่าสถานะของการเข้ารหัสหลักไม่มีความจำเป็นอีกต่อไป

จากรูปที่ 4.19 จำนวนของคอลัมน์ที่ต้องใช้ในการเก็บค่าสถานะของการเข้ารหัสที่ต้องถูกส่งต่อๆกันไปนั้นจะมีเพียง 4 คอลัมน์เท่านั้น ในช่วงเวลาที่ t_n ค่าสถานะของการเข้ารหัสในช่วง SPP ($\eta[i, j]^{t_n}_{SPP}$) สำหรับในตำแหน่งคอลัมน์ที่ j ไม่มีความจำเป็นต้องใช้รีจิสเตอร์ที่ใช้ในการแสดงสถานะของการเข้ารหัสในการเก็บแต่อย่างใด (รีจิสเตอร์ของตัวแปลสถานะของการเข้ารหัสหลัก) เนื่องจากข้อมูลในส่วนนี้จะมีค่าสถานะของการเข้ารหัสเป็น '0' หรือยังไม่ถูกประมวลผลจากพาสใดๆ มาก่อนหน้าเลย จึงสามารถตัดรีจิสเตอร์ในส่วนของคอลัมน์นี้ออกได้

4.3.4.4 โครงสร้างของส่วนสร้างสัญญาณค่าสถานะนัยสำคัญ (Significant Generator)

ในส่วนโครงสร้างนี้ใช้สร้างสัญญาณค่าสถานะนัยสำคัญเนื่องจากการเข้ารหัสข้อมูลในแต่ละตำแหน่ง ค่าสถานะนัยสำคัญมีความสัมพันธ์กันระหว่างค่าสถานะนัยสำคัญเก่ากับค่าสถานะนัยสำคัญที่เกิดขึ้นในขณะที่ข้อมูลในตำแหน่งนั้นถูกประมวลผลในกรอบหน้าต่างประมวลผล ซึ่งในส่วนสร้างสัญญาณค่าสถานะนัยสำคัญนี้จะสร้างกลุ่มของค่าสถานะนัยสำคัญไว้ล่วงหน้าก่อนกลุ่มของค่าสัญญาณสถานะนัยสำคัญในแต่ละตำแหน่งสามารถแสดงได้ดังรูปที่ 4.20



รูปที่ 4.20 กลุ่มของค่าสถานะนัยสำคัญที่เป็นไปได้สำหรับการเข้ารหัสในแต่ละตำแหน่ง (PS_σ)

$$PS\sigma[0]_x^{t_n} = \begin{Bmatrix} \sigma[A]_x^{t_n} & \sigma[F]_x^{t_n} & \sigma[G]_x^{t_n} \\ \sigma[B]_x^{t_n} & & \sigma[H]_x^{t_n} \\ \sigma[C]_x^{t_n} & \sigma[R1(j)]_x^{t_n} & \sigma[I]_x^{t_n} \end{Bmatrix}$$

รูปที่ 4.21 แสดงตัวอย่างของกลุ่มของค่าสถานะนัยสำคัญที่เป็นไปได้สำหรับข้อมูลในแถวที่ 0

จากรูปที่ 4.20 แสดงกลุ่มของค่าสถานะนัยสำคัญที่เป็นไปได้สำหรับการเข้ารหัสในแต่ละตำแหน่ง (PS_σ) ซึ่งเป็นรีจิสเตอร์ที่ใช้เก็บกลุ่มของค่า σ (Significant) ที่เป็นไปได้สำหรับข้อมูลในแถวที่ i จะถูกส่งไปประมวลผลในส่วนของตัวดำเนินการพื้นฐานต่างๆ ของพาสใดๆ โดยค่าสถานะนัยสำคัญที่เรียงตั้งแต่ลำดับที่ 1 ไปถึง 8 แสดงได้ดังรูปที่ 4.21 ซึ่งเรียงจากซ้ายบนเรื่อยไปจนถึงขวาล่าง

ในส่วนของค่า $\hat{\sigma}$ (ส่วนที่แรเงาในรูปที่ 4.20) เป็นสถานะนัยสำคัญที่มีการปรับค่าอย่างทันทีทันใด ณ.เวลาประมวลผลปัจจุบัน อันเนื่องมาจากค่าสถานะนัยสำคัญในแถวที่อยู่ก่อนหน้า

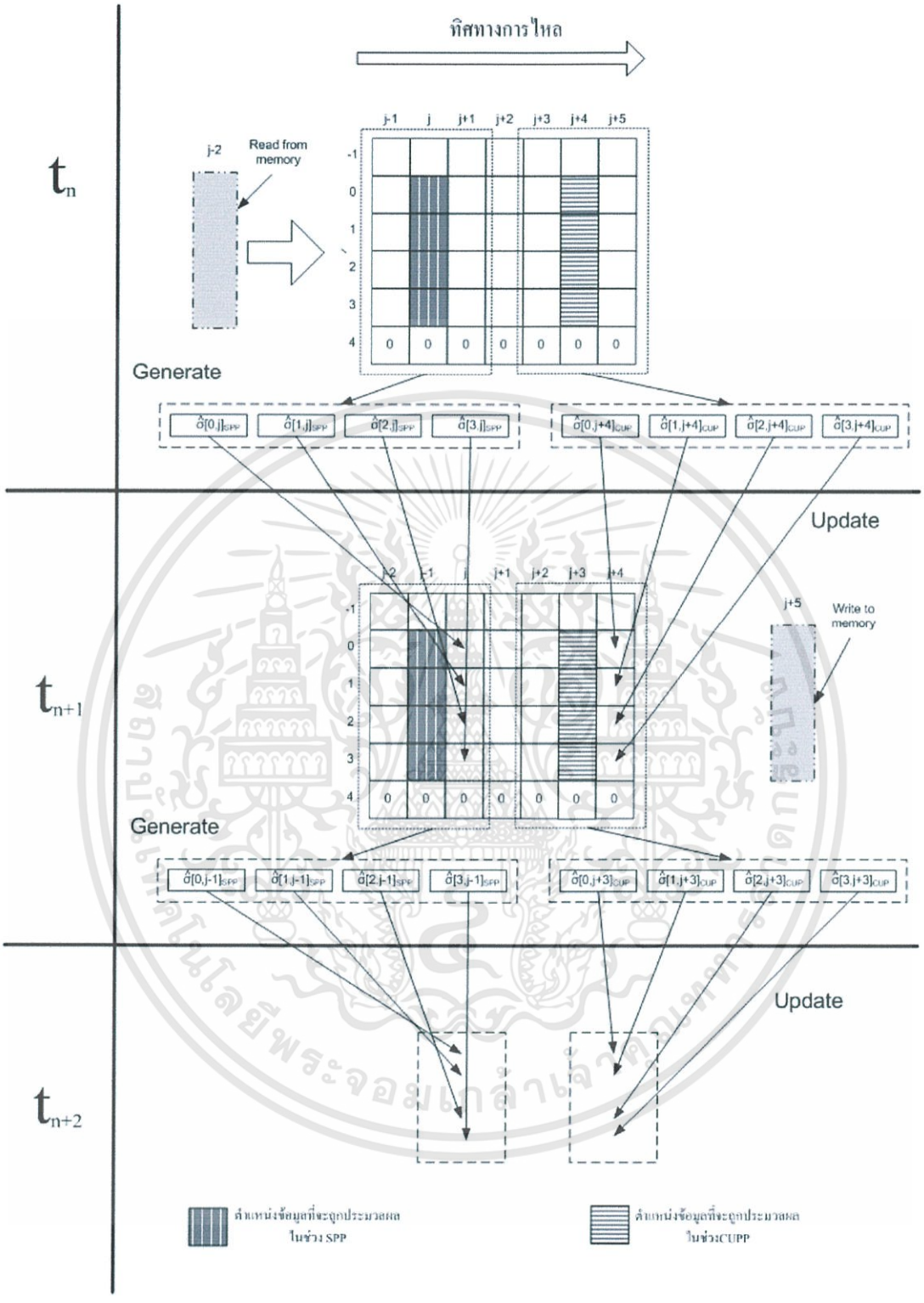
ในช่วง SPP และในช่วง CUP หากข้อมูลในตำแหน่งที่อยู่ในคอลัมน์ j และ $j+4$ ในแถวใดก็ตามมีการประมวลผลและค่าของข้อมูลในตำแหน่งดังกล่าวมีค่าเป็น '1' จะทำให้เกิดการปรับค่าสถานะนัยสำคัญที่ส่งผลกระทบไปยังแถวที่อยู่ถัดไป แต่ในระบบโครงสร้างที่นำเสนอ ส่วนของการประมวลผลจะประมวลผลเข้ารหัสข้อมูลไปทีละคอลัมน์ ทำให้ค่าสัญญาณดังกล่าวนี้จะต้องถูกปรับค่าอย่างทันทีในเวลาปัจจุบันและถูกนำไปใช้กับข้อมูลในแถวถัดๆ ไปโดยค่าดังกล่าวสามารถหาได้จากสมการที่ (4.9)

$$\hat{\sigma}[i,k]^{t_n}_{Pass} = \underset{i \in \{0,1,2\}}{\left(\left(P_{flag}[i]^{t_n}_{Pass} \;\&\; v[i,k]\right) \middle|\; \sigma[i,k]^{t_n}_{Pass}\right)} \quad k = \begin{cases} j & ;Pass = SPP \\ j+4 & ;Pass = CUP \end{cases} \tag{4.9}$$

จากรูปที่ 4.20 เป็นรูปที่แสดงการเตรียมกลุ่มของค่าสถานะนัยสำคัญของข้อมูลในแต่ตำแหน่งแถวให้มีความเหมาะสม เพื่อส่งกลุ่มของค่าสถานะนัยสำคัญที่ได้เตรียมไว้ให้กับตัวดำเนินการพื้นฐานต่อไป ทำให้ตัวดำเนินการพื้นฐานที่เกี่ยวของสามารถออกแบบได้ง่ายขึ้นและสามารถสร้างเป็นโมดูลย่อยๆ แล้วนำมาประกอบกันเพื่อให้เป็นระบบที่ใหญ่ขึ้นได้โดยง่าย จะเห็นได้ว่ากลุ่มของค่าสถานะนัยสำคัญที่เป็นตัวอย่างในรูปเมื่อเปรียบเทียบกับ

รูปที่ 4.21 คือค่าเดียวกันกับตัวอย่างในรูปที่ 4.20 กล่าวได้ว่ากลุ่มของค่าสถานะนัยสำคัญที่เป็นไปได้ในแต่ละตำแหน่งก็คือค่าของสถานะนัยสำคัญเพื่อนบ้านทั้ง 8 ทิศในแต่ละตำแหน่งนั่นเอง

จากรูปที่ 4.23 แสดงขั้นตอนในการส่งต่อค่าสถานะนัยสำคัญและค่าสถานะนัยสำคัญที่มีการปรับเปลี่ยนค่าเมื่อผ่านการประมวลผลใน SPP และ CUP โดยในช่วงเวลาที่ t_n ค่าสถานะนัยสำคัญของคอลัมน์ที่ j กับ $j+4$ จะถูกสร้างขึ้นมาเพื่อนำมาใช้กับข้อมูลในคอลัมน์ที่ j และ $j+4$ ในส่วนที่ค่าเหล่านั้นส่งผลกระทบ เมื่อเวลาผ่านไปที่ t_{n+1} ค่าสถานะนัยสำคัญในตำแหน่งคอลัมน์ j กับ $j+4$ ในช่วง t_n จะถูกส่งต่อไปยังคอลัมน์ที่ $j+1$ กับ $j+5$ (มองตำแหน่งดังกล่าวในช่วงเวลาที่ t_n) ส่วนค่าสถานะนัยสำคัญที่ $j+5$ จะถูกส่งต่อไปเก็บในหน่วยความจำค่าสถานะนัยสำคัญ



รูปที่ 4.22 แสดงสัญญาณสถานะนัยสำคัญที่ถูกส่งต่อไปในกรอบหน้าต่างของการประมวลผล

4.3.4.5 โครงสร้างของส่วนสร้างสัญญาณค่าสถานะ Magnitude Refinement (Magnitude Refinement Generator)

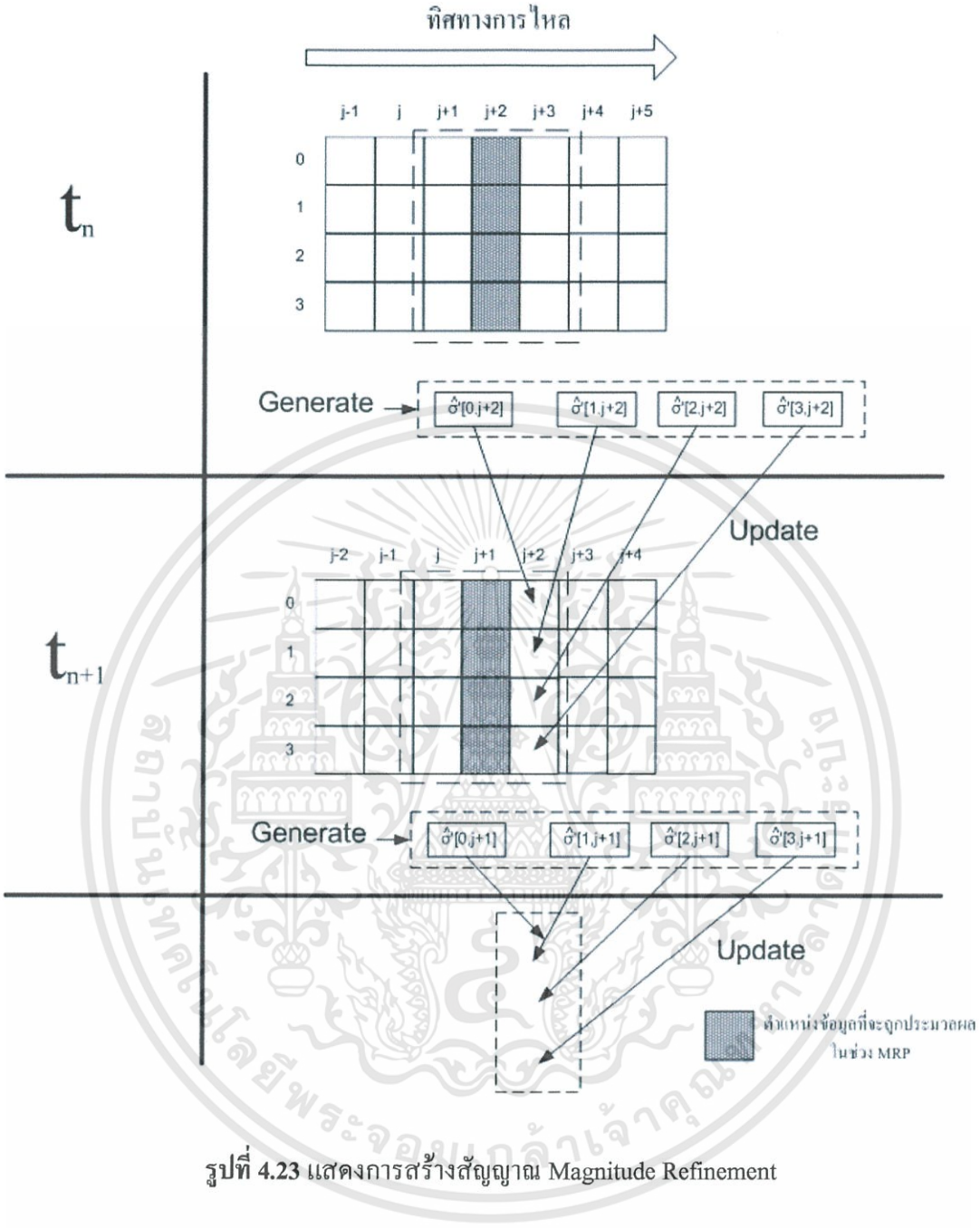
สำหรับโครงสร้างในส่วนนี้ ทำหน้าที่สร้างสัญญาณค่าสถานะ Magnitude Refinement ซึ่งค่าสัญญาณสถานะนี้จะถูกใช้ในการเข้ารหัสของช่วงการเข้ารหัส MRP เพื่อให้ระบบสามารถทำงานได้อย่างเต็มประสิทธิภาพ

จากรูปที่ 4.23 แสดงขั้นตอนในการสร้างสัญญาณสถานะ Magnitude Refinement โดย ณ. ตำแหน่งเวลาปัจจุบัน (t_n) ข้อมูลในตำแหน่งคอลัมน์ที่ $j+2$ จะถูกประมวลผลในช่วง MRP หากข้อมูลในตำแหน่งใดถูกเข้ารหัสในช่วง MRP นี้ ค่าสัญญาณสถานะ Magnitude Refinement จะถูกสร้างขึ้น ($\hat{\sigma}'$) ซึ่งค่าสัญญาณที่สร้างขึ้นจะมีค่าเป็น '1' หากข้อมูลถูกเข้ารหัสในช่วง MRP หากข้อมูลในตำแหน่งดังกล่าวไม่ถูกประมวลผลในช่วง MRP ค่าของ $\hat{\sigma}'$ จะมีค่าเป็น '0' ซึ่งค่าสัญญาณที่สร้างขึ้นมานี้จะเป็นค่าที่ได้จากสัญญาณความเป็นสมาชิกของ MRP ($P_{flag}[i]^{t'_n}_{MRP}$) กับค่าสัญญาณสถานะ Magnitude Refinement เดิม โดยในตัวอย่างที่ได้จากรูปคือค่า $\sigma'[i, j+2]^{t'_n}_{MRP}$ ของข้อมูลที่ถูกเข้าในตำแหน่งคอลัมน์ที่ $j+2$ ที่ได้แสดงในรูปที่ 4.23 สำหรับสมการที่ใช้สร้างสัญญาณสถานะ Magnitude Refinement สามารถหาได้จากสมการที่ (4.10)

$$\hat{\sigma}'[i,k]^{t'_n}_{MRP} = \left(P_{flag}[i]^{t'_n}_{MRP} \mid \sigma'[i,k]^{t'_n}_{MRP}\right) \quad i \in \{0,1,2\} \qquad k = j+2 \tag{4.10}$$

4.3.5 โครงสร้างส่วนการดำเนินการสำหรับการเข้ารหัสขั้นพื้นฐาน (Primitive Operator Generator Module)

ในส่วนของตัวดำเนินการพื้นฐานประกอบไปด้วย 4 ส่วน คือ ตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์ (ZC), ตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมาย (SC), ตัวเนินการพื้นฐานการเข้ารหัสแบบ Magnitude Refinement (MRC), ตัวดำเนินการพื้นฐานการเข้ารหัสแบบการวิ่งต่อเนื่องของศูนย์ (RLC) โดยส่วนของตัวดำเนินการพื้นฐานของแต่ละโมดูลจะถูกประกอบรวมกันให้เหมาะสมกับการทำงานในแต่ละพาส



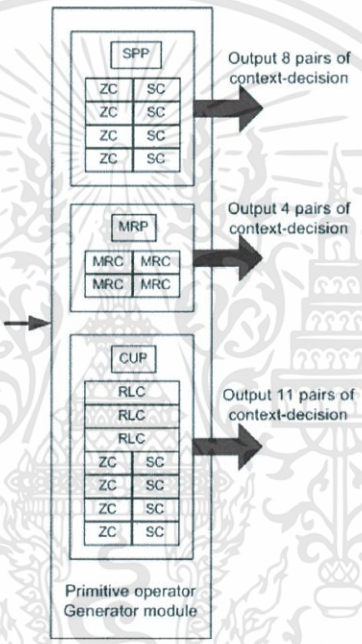
รูปที่ 4.23 แสดงการสร้างสัญญาณ Magnitude Refinement

โดยจำนวนของตัวดำเนินการพื้นฐานการเข้ารหัสชนิดต่างๆเป็นดังนี้

ตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์	4 ตัว
เพื่อให้สามารถประมวลผลข้อมูลพร้อมกันทั้ง 4 แถว	
ตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมาย	4 ตัว
ตัวเนินการพื้นฐานการเข้ารหัสแบบ Magnitude Refinement	4 ตัว
ตัวดำเนินการพื้นฐานการเข้ารหัสแบบการวิ่งต่อเนื่องของศูนย์	
ใช้สร้างสัญญาณ CX = 17	1 ตัว

ใช้สร้างสัญญาณ CX = 18	2 ตัว

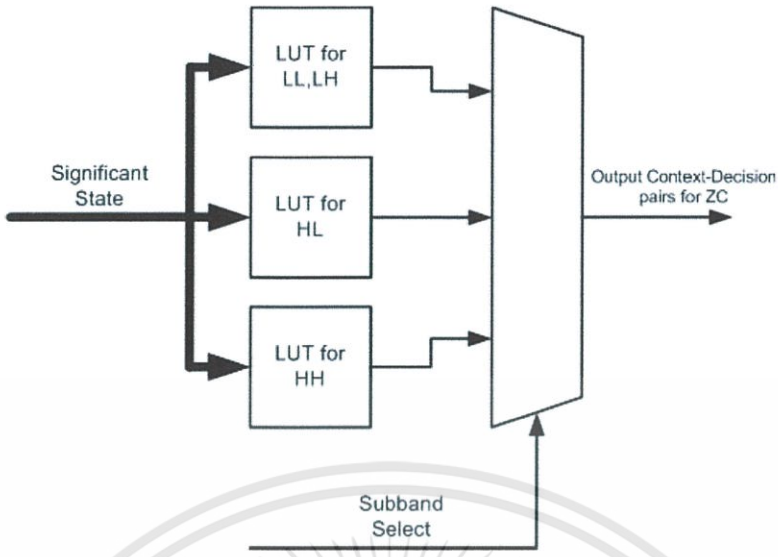
ซึ่งในช่วงการทำงาน SPP จะประกอบไปด้วยส่วนของตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์ และตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมาย สำหรับการทำงาน MRP ประกอบด้วยตัวดำเนินการพื้นฐาน การเข้ารหัสแบบ Magnitude Refinement ในส่วนการทำงาน CUP อันเป็นส่วนสุดท้ายประกอบด้วยตัวตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์ ตัวดำเนินการพื้นฐานการเข้ารหัสแบบการวิ่งต่อเนื่องของศูนย์ และตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมาย สามารถแสดงได้ดังรูปที่ 4.24



รูปที่ 4.24 แสดงโครงสร้างของตัวดำเนินการพื้นฐานในแต่ละพาสการทำงาน

4.3.5.1 ตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์ (Zero Coding Operator)

โครงสร้างในส่วนนี้ เป็นส่วนที่ใช้สร้างสัญญาณคู่อันดับของ Context-decision ซึ่งค่าคู่อันดับที่สร้างขึ้นมานี้เป็นไปตามกระบวนการของตัวดำเนินการขั้นพื้นฐานการเข้ารหัสศูนย์ ในการบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG2000



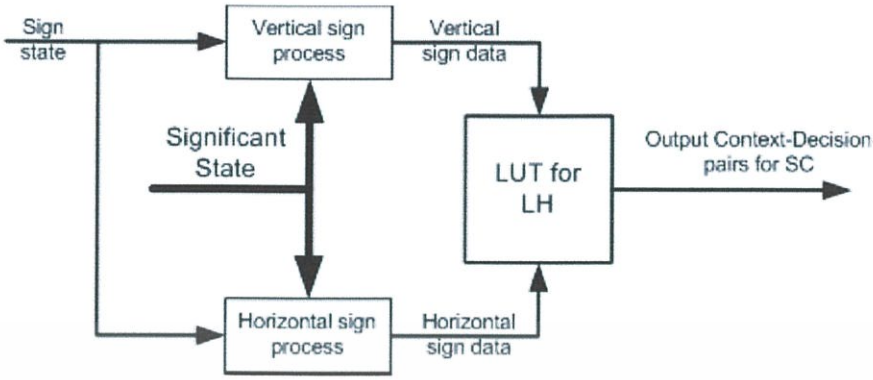
รูปที่ 4.25 แสดงโครงสร้างของตัวดำเนินการพื้นฐาน Zero Coding

จากรูปที่ 4.25 จะเห็นได้ว่า โครงสร้างของตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์ ซึ่งผลลัพธ์การเข้ารหัสศูนย์นี้จะมีค่าสัมพันธ์กับค่าผลรวมของค่าสถานะนัยสำคัญในตำแหน่ง 8 ตำแหน่งรอบข้างตำแหน่งที่เข้ารหัส ตามทิศแนวตั้ง แนวนอน และแนวทแยงมุม ซึ่งสามารถออกแบบการทำงานโดยใช้การเทียบค่าตาราง (LUT: Look Up Table) และเนื่องจากความสัมพันธ์ของผลรวมของค่าสถานะนัยสำคัญ มีความแตกต่างกันตามค่าสัมประสิทธิ์เวฟเลทที่นำมาเข้ารหัส จึงต้องใช้ตารางค่าความสัมพันธ์ของผลรวมของค่าสถานะนัยสำคัญแยกสำหรับแต่ละ Subband ของสัมประสิทธิ์เวฟเลทที่นำมาเข้ารหัส ซึ่งใช้ทั้งหมด 3 ตารางสำหรับข้อมูลที่เป็น LL, LH, HL, และ HH สำหรับการเทียบค่าตารางจะเป็นไปตามตารางของการดำเนินการพื้นฐานการเข้ารหัสศูนย์ของมาตรฐานการบีบอัดสัญญาณรูปภาพ JPEG2000 (ดังกล่าวในหัวข้อ 3.3.5.1) ดังนั้นในการออกแบบของตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์ในงานวิจัยนี้ จึงใช้วิธีการเทียบค่าตารางทั้ง 3 ตารางเตรียมไว้ล่วงหน้า แล้วจึงส่งสัญญาณมาเลือกผลลัพธ์ใน Subband ที่ต้องการด้วยมัลติเพลกซ์

สัญญาณที่ใช้ในการเทียบค่าตารางของตัวดำเนินการขั้นพื้นฐานการเข้ารหัสศูนย์ จะใช้สัญญาณค่าสถานะนัยสำคัญในช่วงการประมวลผลของ SPP และ CUP ซึ่งสัญญาณค่าสถานะนัยสำคัญนี้เป็นไปตามรูปที่ 4.22

4.3.5.2 ตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมาย (Sign Coding Operator)

สำหรับโครงสร้างของตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมาย เป็นไปดังรูปที่ 4.26

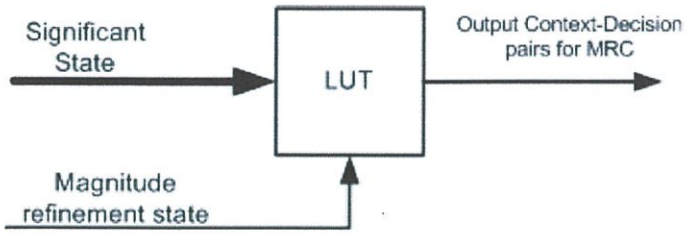


รูปที่ 4.26 แสดงโครงสร้างของตัวดำเนินการพื้นฐาน Sign Coding

ลักษณะของโครงสร้างของตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมาย เป็นการเข้ารหัสเครื่องหมายที่บ่งบอกสภาพแวดล้อมข้างเคียงของตำแหน่งที่กำลังประมวลผลว่าข้อมูลมีเครื่องหมายใดและนำสถานะเครื่องหมายของสภาพแวดล้อมข้างเคียงเหล่านั้นมาทดสอบเครื่องหมายของข้อมูลตำแหน่งที่กำลังประมวลผล ซึ่งจะแบ่งความสัมพันธ์ไปในทิศทางแนวตั้งและทิศทางแนวนอน รวมไปถึงค่าสถานะนัยสำคัญของข้อมูลข้างเคียงที่มีความสัมพันธ์เหล่านั้นด้วย ดังนั้นในส่วนของตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมายจะรับค่าเครื่องหมายและค่าสถานะนัยสำคัญจากตำแหน่งที่เข้ารหัสและตำแหน่งข้างเคียง มาพิจารณาสถานะเครื่องหมายในทิศแนวตั้งเพื่อประมวลผลสถานะเครื่องหมายทิศแนวตั้งและ พิจารณาสถานะเครื่องหมายในทิศแนวนอน เพื่อประมวลผลสถานะเครื่องหมายทิศแนวนอนค่าสถานะเครื่องหมายทั้งสองทิศที่ประมวลผลได้จะถูกนำไปเปิดตารางจึงได้ผลลัพธ์คู่อันดับสัญลักษณ์ (ดังกล่าวในหัวข้อ 3.3.5.2) จะเห็นได้ว่าลักษณะโครงสร้างตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมายมีลักษณะโครงสร้างคล้ายกับตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์ ซึ่งเป็นการทำงานแบบ LUT แต่ใช้ข้อมูลในการอ่านค่าตารางต่างกันเท่านั้น

4.3.5.3 ตัวเนินการพื้นฐานการเข้ารหัสแบบ Magnitude Refinement (Magnitude Refinement Coding Operator)

โครงสร้างในส่วนของโมดูลตัวเนินการพื้นฐานการเข้ารหัสแบบ Magnitude Refinement สามารถแสดงได้ดังรูปที่ 4.27



รูปที่ 4.27 แสดงโครงสร้างตัวเนินการพื้นฐานการเข้ารหัสแบบ Magnitude Refinement

จากรูปที่ 4.27 จะเห็นได้ว่า ค่าคู่อันดับสัญลักษณ์ เป็นค่าที่ขึ้นอยู่กับค่าของสถานะของการเข้ารหัส Magnitude Refinement และผลรวมของค่าสถานะนัยสำคัญ 8 ตำแหน่งที่อยู่รอบตำแหน่งข้อมูลที่เข้ารหัส ดังนั้นค่าทั้งสองนี้จะถูกนำไปเลือกค่าคู่อันดับผลลัพธ์จากตาราง ที่แสดงในหัวข้อ 3.3.5.3 กลุ่มของสัญญาณสถานะนัยสำคัญที่นำไปใช้ในการประมวลผล เป็นไปตามรูปที่ 4.22

4.3.5.4 ตัวดำเนินการพื้นฐานการเข้ารหัสแบบการวิ่งต่อเนื่องของศูนย์ (RLC: Run-Length Coding Operator)

ในส่วนโมดูลสุดท้ายที่นำเสนอนี้เป็นโมดูลของตัวตัวดำเนินการพื้นฐานการเข้ารหัสแบบการวิ่งต่อเนื่องของศูนย์ ซึ่งแสดงดังรูปที่ 4.28

รูปที่ 4.28 แสดงโครงสร้างตัวดำเนินการพื้นฐานการเข้ารหัสแบบการวิ่งต่อเนื่องของศูนย์

สำหรับโครงสร้างตัวดำเนินการพื้นฐานการเข้ารหัสแบบการวิ่งต่อเนื่องของศูนย์นี้ จะประมวลผลเฉพาะในช่วง CUP โดยภายในโมดูลประกอบไปด้วยส่วนที่คอยตรวจสอบว่ากลุ่มของค่าสถานะนัยสำคัญรอบข้างของทั้งคอลัมน์ว่ามีค่าที่เป็นไปตามเงื่อนไขหรือไม่ หากค่าสถานะนัยสำคัญรอบข้างเป็นไปตามเงื่อนไขโมดูลนี้จะทำการประมวลผลของข้อมูลในขณะนั้น ซึ่งส่วนของตัวดำเนินการพื้นฐานจะแบ่งออกเป็น 2 ตัว คือตัวดำเนินการพื้นฐานการเข้ารหัสการวิ่งต่อเนื่องของศูนย์ที่ใช้บ่งบอกถึงความต่อเนื่องของค่าศูนย์ในคอลัมน์ที่เข้ารหัสและคอลัมน์ข้างเคียง ซึ่งจะได้คู่อันดับผลลัพธ์เป็น CX = 17 และตัวดำเนินการที่ใช้บ่งชี้ตำแหน่งของข้อมูลที่ทำให้การวิ่งต่อเนื่อง

ของศูนย์เกิดการชะงัก ซึ่งจะได้คู่อันดับผลลัพธ์เป็น CX = 18 ทำให้ตัวดำเนินการพื้นฐานการเข้ารหัสการวิ่งต่อเนื่องของศูนย์ มีด้วยกันทั้งสิ้น 3 ตัว คือ ตัวที่ใช้สร้าง CX หมายเลข 17, ตัวบ่งชี้ตำแหน่งที่ทำให้เกิดการชะงักของการวิ่ง 2 ตัว โดยตัวแรกบอกค่า MSB ของตำแหน่งที่ทำให้เกิดชะงัก ส่วนอีกตัวใช้บอกค่า LSB ของตำแหน่งดังกล่าว

เนื่องจากการทำงานของระบบโครงสร้างที่นำเสนอประมวลผลพร้อมกันทั้ง 3 พาสทำให้ตัวดำเนินการพื้นฐานการเข้ารหัสจะต้องแยกออกมาเป็นส่วนๆ โดยส่วนการทำงานช่วง SPP ประกอบด้วย ตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์จำนวน 4 ตัว และ ตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมายจำนวน 4 ตัวที่ประมวลผลพร้อมๆ กันทำให้ได้ผลการเข้ารหัสพร้อมกันถึง 8 สัญลักษณ์ สำหรับในช่วง MRP ประกอบด้วยส่วนของตัวเนินการพื้นฐานการเข้ารหัสแบบ Magnitude Refinement จำนวน 4 ตัว ที่ประมวลผลพร้อมกัน เป็นสาเหตุให้การเข้ารหัสในช่วง MRP สามารถให้ผลการเข้ารหัสได้ถึง 4 สัญลักษณ์ และในช่วงการทำงาน CUP ประกอบไปด้วยตัวดำเนินการพื้นฐานการเข้ารหัสแบบการวิ่งต่อเนื่องของศูนย์ จำนวน 3 ตัว, ตัวดำเนินการพื้นฐานการเข้ารหัสศูนย์จำนวน 4 ตัว และ ตัวดำเนินการพื้นฐานการเข้ารหัสเครื่องหมายจำนวน 4 ตัว ซึ่งในส่วนการประมวลผลของช่วงการเข้ารหัสนี้ สามารถเข้ารหัสได้มากถึง 10 สัญลักษณ์พร้อมกัน ซึ่งในช่วงที่การทำงานสูงสุดของการเข้ารหัสที่เป็นไปตามเงื่อนไข ทำให้โครงสร้างในงานวิจัยนี้สามารถประมวลผลในการเข้ารหัสได้มากที่สุดถึง 22 สัญลักษณ์พร้อมกันใน 1 สัญญาณนาฬิกา

บทที่ 5

การทดลองและผลการทดลอง

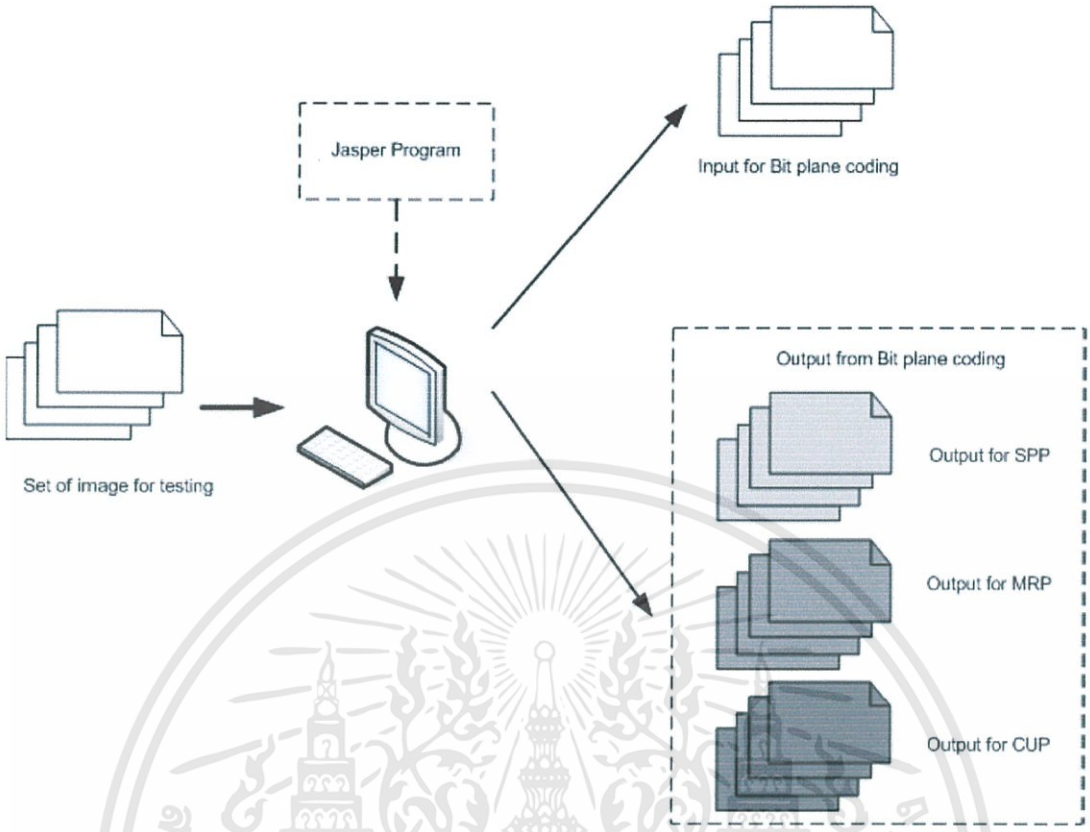
5.1 บทนำ

ในบทที่ 4 ได้เสนอระบบโครงสร้างของงานวิจัย ส่วนในบทนี้จะเป็นการนำเสนอถึงกระบวนการในการทดสอบประสิทธิภาพของระบบโครงสร้างได้นำเสนอและเปรียบเทียบกับระบบโครงสร้างของงานวิจัย [9] โดยระบบโครงสร้างในส่วนของระบบโครงสร้างที่นำเสนอและระบบโครงสร้างของงานวิจัย [9] จะถูกทดสอบตั้งแต่การวัดขนาดของทรัพยากรที่ใช้ใน FPGA (Cost), การวัดประสิทธิภาพความสามารถทางด้านความเร็วของแต่ระบบ (Clock Frequency), การวัดประสิทธิภาพในการตอบสนองการทำงานตั้งแต่ข้อมูลถูกนำเข้าจนประมวลผลเสร็จสิ้น (Processing Time), การวัดประสิทธิภาพปริมาณข้อมูลผลลัพธ์ที่สามารถประมวลผลได้ (Throughput) และการทดสอบความคุ้มค่าของระบบโครงสร้างในแต่ละระบบ (Throughput/Cost) การทดสอบระบบโครงสร้างทั้งหมดจะถูกทดสอบด้วยการจำลองการทำงานบนโปรแกรม Modelsim ซึ่งใช้สำหรับจำลองการทำงานของระบบโครงสร้างทางฮาร์ดแวร์และเพื่อใช้ตรวจสอบความถูกต้องของระบบโครงสร้างทั้งหมด

5.2 การเตรียมข้อมูลที่ใช้ในการทดสอบระบบโครงสร้างทั้งหมด

ในหัวข้อนี้จะเป็นการกล่าวถึงวิธีการในการเตรียมข้อมูลที่จะถูกนำมาใช้ในการทดสอบระบบโครงสร้างทั้งหมด เพื่อนำมาใช้ทดสอบความถูกต้องของผลลัพธ์ที่ได้จากระบบโครงสร้างต่างๆ รวมไปถึงเพื่อนำไปเป็นตัวอ้างอิงในการวัดประสิทธิภาพต่างๆในการเปรียบเทียบผลการทำงาน

ในการเตรียมข้อมูลที่จะนำมาใช้ทดสอบ จะใช้โปรแกรมที่เป็นมาตรฐานสำหรับการบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 ซึ่งมีอยู่ด้วยกัน 2 โปรแกรม ที่มาตรฐานได้รองรับไว้ คือ โปรแกรม Jasper และ JJ2000 ในส่วนของการเตรียมข้อมูลนี้จะใช้โปรแกรม Jasper มาเป็นตัวสร้างข้อมูลที่จะถูกนำไปทดสอบพร้อมทั้งผลลัพธ์ที่ได้จากการเข้ารหัสระนาบบิต สามารถแสดงการสร้างข้อมูลที่เป็นอินพุตสำหรับการเข้ารหัสระนาบบิตและผลลัพธ์ที่ได้จากการสร้างดังรูปที่ 5.1



รูปที่ 5.1 แสดงการสร้างสัญญาณข้อมูลที่ใช้เป็นตัวทดสอบการทำงาน

จากรูปที่ 5.1 แสดงวิธีการในการสร้างสัญญาณข้อมูลในส่วนอินพุตของการเข้ารหัสระนาบบิต และส่วนของผลลัพธ์ที่ได้จากการเข้ารหัสระนาบบิต โดยเริ่มแรกข้อมูลที่เป็นรูปภาพจะถูกส่งเข้าไปประมวลผลในโปรแกรม Jasper ซึ่งโปรแกรม Jasper จะทำการบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 เมื่อการประมวลผลมาถึงช่วงการเข้ารหัสระนาบบิต ข้อมูลที่เป็นส่วนอินพุตจะถูกเขียนลงไฟล์ เพื่อนำมาใช้เป็นอินพุตสำหรับการทดสอบ เมื่อการเข้ารหัสระนาบบิตในโปรแกรม Jasper ประมวลผลเสร็จสิ้นจะได้ผลลัพธ์คู่อันดับสัญลักษณ์ ผลลัพธ์ที่ได้นั้นจะถูกเขียนลงไฟล์เพื่อนำมาใช้เป็นผลลัพธ์ในการตรวจสอบความถูกต้องของผลลัพธ์กับระบบโครงสร้างที่จะนำมาทดสอบ

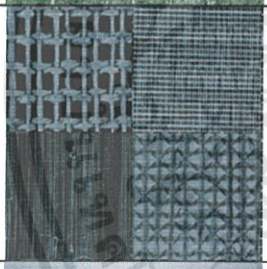
โดยข้อมูลผลลัพธ์จะถูกแบ่งออกเป็น 3 ไฟล์ ไฟล์แรกเป็นข้อมูลผลลัพธ์สำหรับการประมวลผลช่วง SPP ไฟล์ที่สองเป็นของการประมวลผลช่วง MRP และส่วนไฟล์สุดท้ายเป็นผลลัพธ์สำหรับการประมวลผลช่วง CUP โดยรวมแล้วไฟล์ที่ได้จากโปรแกรม Jasper จะมีด้วยกันทั้งสิ้น 4 ไฟล์ (ผลลัพธ์ 3 ไฟล์และข้อมูลอินพุต 1 ไฟล์) สำหรับรูปภาพที่ใช้เป็นข้อมูลอินพุตในการเตรียมข้อมูลสำหรับการเข้ารหัสระนาบบิตจะมีความแตกต่างกันใน 3 ลักษณะด้วยกัน คือ

- ขนาดของรูปภาพที่ใช้ในการเข้ารหัสระนาบบิตที่มีขนาดแตกต่างกัน

- รูปภาพที่เป็นรูปสีและรูปภาพระดับขาวดำ
- รูปภาพที่มีความละเอียดของภาพที่แตกต่างกัน

รูปที่ใช้ในการทดสอบทั้งหมดสามารถแสดงได้ดังข้างล่าง

ตารางที่ 5.1 แสดงรายละเอียดของรูปภาพที่ถูกนำมาใช้ในการทดสอบ

รูปภาพที่ใช้ในการทดสอบ	หมายเลขภาพ	ขนาด (จุด x จุด)	จำนวนผลลัพธ์คู่อันดับสัญลักษณ์จากโปรแกรม Jasper (คู่อันดับสัญลักษณ์)
	1	256 x 256	667245
	2	256 x 256	480879
	3	256 x 256	355642
	4	512 x 512	1089597
	5	256 x 256	903923
	6	512 x 512	2802085

ตารางที่ 5.1 (ต่อ) แสดงรายละเอียดของรูปภาพที่ถูกนำมาใช้ในการทดสอบ

รูปภาพที่ใช้ในการทดสอบ	หมายเลขภาพ	ขนาด (จุด x จุด)	จำนวนผลลัพธ์คู่อันดับสัญลักษณ์จากโปรแกรม Jasper (คู่อันดับสัญลักษณ์)
	7	512 x 512	1337570
	8	512 x 512	1424199
	9	512 x 512	3701804
	10	256 x 256	445296
	11	512 x 512	1448574
	12	512 x 512	2635122

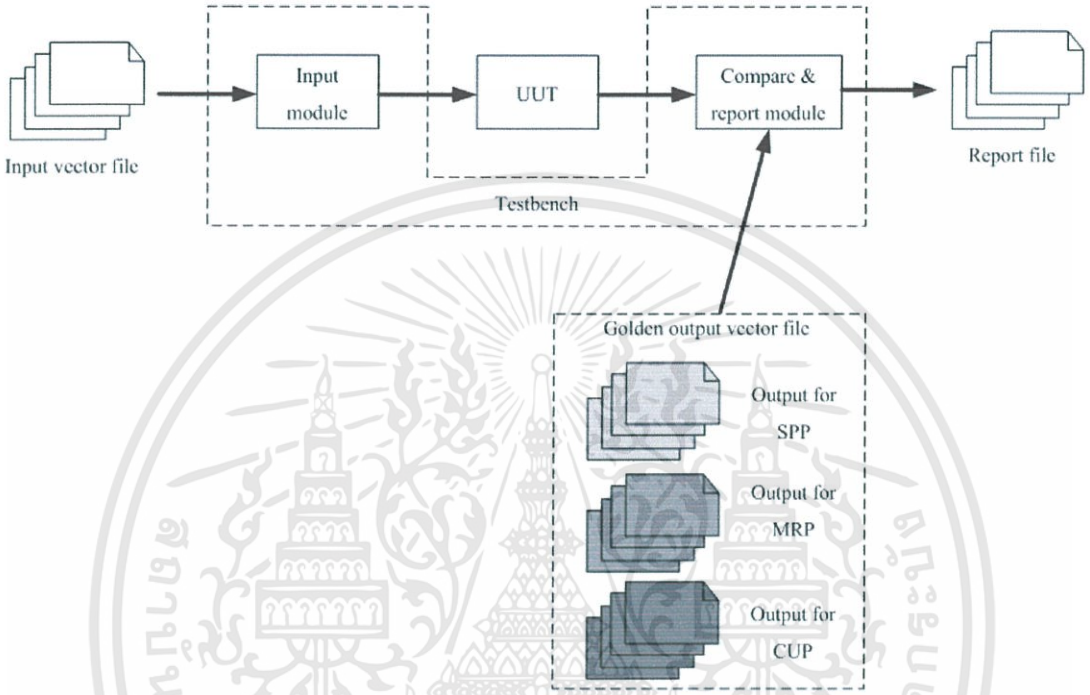
ตารางที่ 5.1 (ต่อ) แสดงรายละเอียดของรูปภาพที่ถูกนำมาใช้ในการทดสอบ

รูปภาพที่ใช้ในการทดสอบ	หมายเลขภาพ	ขนาด (จุด x จุด)	จำนวนผลลัพธ์คู่อันดับสัญลักษณ์จากโปรแกรม Jasper (คู่อันดับสัญลักษณ์)
	13	512 x 512	1705287
	14	512 x 512	1414563
	15	512 x 512	1127936
	16	512 x 512	1420524
	17	512 x 512	951016

โดยความละเอียดของรูปภาพแต่ละรูปจะมีความสัมพันธ์กับปริมาณผลลัพธ์คู่อันดับสัญลักษณ์ หากรูปภาพมีปริมาณผลลัพธ์คู่อันดับสัญลักษณ์ที่มากแสดงว่ารูปภาพนั้นมีความละเอียดหรือเนื้อข้อมูลของภาพที่มากตามไปด้วย

5.3 ระบบโครงสร้างที่ใช้ทดสอบประสิทธิภาพ

ในส่วนของหัวข้อนี้จะเป็นการกล่าวถึงระบบโครงสร้างที่ใช้ในการทดสอบโมดูลของระบบโครงสร้างที่นำเสนอ และโมดูลระบบโครงสร้างของงานวิจัยต่างๆ เพื่อทดสอบความถูกต้องและประสิทธิภาพของระบบโครงสร้างของงานวิจัยต่างๆ



รูปที่ 5.2 แสดงระบบโครงสร้างที่ใช้ในการทดสอบโมดูล

จากรูปที่ 5.2 แสดงโครงสร้างของระบบที่ใช้ทดสอบการทำงาน โดยระบบนี้จะจำลองการทำงานของระบบโครงสร้างทางฮาร์ดแวร์ที่ได้นำเสนอ ซึ่งแบ่งส่วนที่เกี่ยวข้องออกได้เป็น 3 ส่วนคือ

- ส่วนที่ใช้อ่านข้อมูลอินพุตจากไฟล์ข้อมูลอินพุตแล้วทำการส่งข้อมูลต่อไปยังโมดูลที่จะใช้ทดสอบ (Input Module)
- ส่วนของโมดูลที่จะถูกทดสอบ (UUT: Unit Under Test)
- ส่วนสุดท้ายเป็นส่วนที่ใช้สำหรับรับค่าสัญญาณผลลัพธ์ที่ได้จากการจำลองการทำงานของโมดูลที่ถูกทดสอบ และอ่านค่าผลลัพธ์ที่ได้สร้างมาจากโปรแกรมบีบอัดสัญญาณรูปภาพมาตรฐาน JPEG200 หรือ Jasper เพื่อนำมาเปรียบเทียบความถูกต้องของการทำงาน รวมทั้งยังออกรายงานผลการทดสอบด้วย

ดังที่ได้กล่าวมาข้างต้นจะเห็นได้ว่า การทำงานของระบบโครงสร้างทางฮาร์ดแวร์จะถูกตรวจสอบความถูกต้องของข้อมูลเทียบกับโปรแกรมที่ได้ตามมาตรฐานการบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 ซึ่งหากข้อมูลของผลลัพธ์ที่ได้จากการจำลองการทำงานกับผลลัพธ์ที่ได้จากโปรแกรมตามมาตรฐานไม่สัมพันธ์กัน ในส่วนของโมดูลสุดท้ายที่ใช้เปรียบเทียบจะรายงานสิ่งผิดปกติเหล่านั้นลงไปในไฟล์รายงาน เพื่อบ่งชี้ความไม่ถูกต้องของการทำงานไว้เป็นหลักสำคัญ อีกทั้งในส่วนของการรายงานยังรายงานผลของประสิทธิภาพในการทำงานของระบบโครงสร้างต่างๆ ไว้ด้วย ทั้งในส่วนของจำนวนสัญญาณนาฬิกาที่ระบบใช้ในการประมวลผลและจำนวนสัญญาณผลลัพธ์คู่อันดับสัญลักษณ์ที่ได้

5.3.1 การทดสอบวัดขนาดทรัพยากรของ FPGA

ในส่วนของหัวข้อนี้จะเป็นการทดสอบวัดขนาดทรัพยากรของระบบโครงสร้างที่นำเสนอกับระบบโครงสร้างของงานวิจัย [9] โดยการนำระบบโครงสร้างทั้งสองไปผ่านการสังเคราะห์ ด้วยโปรแกรมสำหรับใช้สังเคราะห์ของบริษัท xilinx ที่มีชื่อว่า ISE Webpack ในการทดสอบจะใช้ชิพ FPGA Spatan3 xc3s1500-4fg676 สำหรับผลที่ได้จากการสังเคราะห์สามารถแสดงได้ดังตารางที่ 5.2 และตารางที่ 5.3

ตารางที่ 5.2 แสดงทรัพยากรที่ใช้ในการออกแบบระบบโครงสร้างที่นำเสนอ

	จำนวนทั้งหมด	จำนวนที่ใช้ไป	เปอร์เซ็นต์
Slices	13,312	786	5%
Flip Flops	26,624	573	2%
4 input LUTs	26,624	1,174	4%
Block RAMs	32	15	46%
GCLKs	8	1	12%
Total equivalent gate count for design			997,762

ตารางที่ 5.3 แสดงทรัพยากรที่ใช้ในการออกแบบระบบโครงสร้างจากงานวิจัย [9]

	จำนวนทั้งหมด	จำนวนที่ใช้ไป	เปอร์เซ็นต์
Slices	13,312	978	7%
Flip Flops	26,624	441	1%
4 input LUTs	26,624	1,178	4%
Block RAMs	32	10	31%
GCLKs	8	1	12%
Total equivalent gate count for design			701,519

5.3.2 การทดสอบประสิทธิภาพของระบบโครงสร้างและผลการทดสอบ

ในหัวข้อนี้เป็นการนำเสนอการทดสอบประสิทธิภาพการทำงานของระบบโครงสร้างที่นำเสนอเปรียบเทียบกับระบบโครงสร้างจากงานวิจัย [9] พร้อมทั้งผลการทดสอบประสิทธิภาพในแต่ละการทดสอบ ในการทดสอบประสิทธิภาพนี้จะแบ่งการทดสอบออกเป็น

- การทดสอบทางด้านความเร็วของระบบโครงสร้างที่นำเสนอและระบบโครงสร้างของงานวิจัย [9] (Clock Frequency)
- การทดสอบการตอบสนองทางด้านเวลาในการทำงานของระบบโครงสร้างที่นำเสนอกับระบบโครงสร้างของงานวิจัย [9] (Processing Time)
- การทดสอบปริมาณผลลัพธ์ต่อเวลาของระบบโครงสร้างที่นำเสนอและระบบโครงสร้างของงานวิจัย [9] (Throughput)
- การทดสอบประสิทธิภาพระบบหน่วยความจำคู่ของระบบที่นำเสนอกับระบบหน่วยความจำเดี่ยวของงานวิจัย [9]

5.3.3 การทดสอบทางด้านความเร็วของระบบโครงสร้างที่นำเสนอและระบบโครงสร้างของงานวิจัย [9]

ในส่วนของการทดสอบของส่วนนี้จะเป็นการทดสอบความสามารถของระบบโครงสร้างที่นำเสนอและระบบโครงสร้างของงานวิจัย [9] ในด้านความเร็วของการประมวลผล ซึ่งการทดสอบนี้จะใช้โปรแกรมในการสังเคราะห์ภาษา VHDL ให้กลายเป็นโครงสร้างที่ไปขึ้นอยู่กับฮาร์ดแวร์จริง โดยตัวโปรแกรมสังเคราะห์ที่ใช้ในการทดสอบหัวข้อนี้ จะใช้ตัวซอฟต์แวร์สังเคราะห์ที่ชื่อว่า ISE Webpack ของบริษัท xilinx เป็นตัวสังเคราะห์ระบบโครงสร้างที่นำเสนอ

และระบบโครงสร้างจากงานวิจัย [9] ผลที่ได้จากการสังเคราะห์ระบบโครงสร้างที่นำเสนอและระบบโครงสร้างจากงานวิจัย [9] สามารถแสดงได้ดังตารางที่ 5.4

ตารางที่ 5.4 ผลจากการสังเคราะห์ระบบโครงสร้างที่นำเสนอและระบบโครงสร้างจากงานวิจัย [9]

ระบบโครงสร้าง	ความเร็วสูงสุดที่สามารถทำงานได้ (Maximum Clock Frequency; MHz)
ระบบโครงสร้างที่นำเสนอ	100.371
ระบบโครงสร้างจากงานวิจัย [9]	100.160

จากตารางที่ 5.4 จะเห็นได้ว่าในแต่ละระบบโครงสร้าง ความเร็วที่สามารถทำงานได้จะมีความเร็วที่ใกล้เคียงกัน อันเนื่องมาจากระบบโครงสร้างการทำงานทั้งสองระบบใช้การออกแบบด้วยการทำงานแบบส่งต่อและการประมวลผลข้อมูลแบบขนาน แต่การประมวลผลการเข้ารหัสระนาบบิตของงานวิจัย [9] จะประมวลผลไปทีละพาส ต่างจากระบบโครงสร้างที่นำเสนอซึ่งสามารถประมวลผลการเข้ารหัสระนาบบิตทั้ง 3 พาส ในสัญญาณนาฬิกาลูกเดียวกันโดยตำแหน่งของข้อมูลที่ถูกประมวลผลจะต่างตำแหน่งกัน และอีกเหตุผลหนึ่งที่ความเร็วในการประมวลผลของระบบโครงสร้างทั้งสองมีค่าใกล้เคียงกัน เนื่องจากระบบโครงสร้างทั้งสองถูกออกแบบระบบโครงสร้างบนเทคโนโลยีเดียวกัน (บน FPGA เหมือนกัน) นั่นเอง

5.3.4 การทดสอบการตอบสนองทางด้านเวลาในการทำงานของระบบโครงสร้างที่นำเสนอกับระบบโครงสร้างของงานวิจัย [9]

ในการวัดประสิทธิภาพการประมวลผลของระบบโครงสร้างทางฮาร์ดแวร์ ไม่เพียงแต่ความเร็วในการประมวลผลเท่านั้นที่เป็นตัวบอกประสิทธิภาพของระบบ แต่ยังต้องสามารถตอบสนองต่อการสั่งการได้อย่างรวดเร็วด้วย เช่น หากระบบที่มีความเร็วในการทำงานสูง 100 เมกะเฮิร์ต แต่เมื่อสั่งการทำงานไปแล้วต้องรอการประมวลผลให้เสร็จสิ้นเป็นเวลา 1 วินาที กับอีกระบบหนึ่งที่มีความเร็วในการทำงานสูงเพียง 90 เมกะเฮิร์ต แต่สามารถประมวลผลข้อมูลเดียวกันให้เสร็จสิ้นในเวลา 0.48 วินาที จะเห็นได้ว่าระบบที่ประมวลผลด้วยความเร็ว 90 เมกะเฮิร์ต แต่มีความสามารถในการตอบสนองต่อการสั่งงานจนทำงานเสร็จสิ้นเร็วกว่าระบบโครงสร้างที่ประมวลผลความเร็วในการทำงานสูง 100 เมกะเฮิร์ต

ในการวัดประสิทธิภาพของการตอบสนองทางด้านเวลาต่อการสั่งงานนี้จะใช้วิธีการนับจำนวนลูกสัญญาณนาฬิกาตั้งแต่เริ่มต้นการประมวลผลข้อมูลจนสิ้นสุดการประมวลผล แล้วนำมาคำนวณหาค่าเวลาในการทำงานเฉลี่ยของระบบโครงสร้างของแต่ละระบบโครงสร้าง ในการ

ทดสอบจะวัดประสิทธิภาพของทั้งระบบโครงสร้างที่นำเสนอและระบบโครงสร้างของงานวิจัย [9] โดยแบ่งวัดไปตามแต่ละรูปภาพที่ใช้ในการทดสอบ แล้วนำผลทั้งหมดมาเฉลี่ยเพื่อเป็นค่ากลางในการบ่งชี้ประสิทธิภาพ ซึ่งผลที่ได้จากการทดสอบประสิทธิภาพการประมวลผลของแต่ละระบบโครงสร้างสามารถแสดงได้ดังตารางที่ 5.5

สมการที่ใช้วัดประสิทธิภาพระยะเวลาในการตอบสนองต่อการสั่งการที่ใช้สามารถหาค่าได้จากสมการที่ (5.1)

$$T_{res} = \frac{1}{F} \times Clock\ cycle \tag{5.1}$$

- ระยะเวลาในการตอบสนอง (T_{res}) มีหน่วยเป็นวินาที
- ความเร็วในการประมวลผล (F; Clock Frequency) มีหน่วยเป็นเมกะเฮิร์ต
- จำนวนลูกสัญญาณนาฬิกา (Clock Cycle) มีหน่วยเป็นลูกสัญญาณนาฬิกา

จากตารางที่ 5.6 แสดงให้เห็นว่าระบบโครงสร้างการทำงานของทั้งสองระบบสามารถตอบสนองต่อการสั่งงานได้รวดเร็วเพียงใด โดยหากระบบโครงสร้างทั้งสองประมวลผลข้อมูลชุดเดียวกัน ในระบบโครงสร้างที่นำเสนอ เมื่อสั่งการไปแล้วระบบโครงสร้างที่นำเสนอจะใช้เวลาประมาณ 4.491 มิลลิวินาที จึงประมวลผลข้อมูลเสร็จสิ้น หากข้อมูลเดียวกันนี้ถูกส่งให้ระบบโครงสร้างจากงานวิจัย [9] ประมวลผล การประมวลผลการเข้ารหัสจากงานวิจัย [9] จะใช้เวลาทั้งสิ้นประมาณ 12.673 มิลลิวินาที ซึ่งใช้เวลายาวนานกว่าระบบโครงสร้างที่นำเสนอ

5.3.5 การทดสอบปริมาณผลลัพธ์ต่อเวลาของระบบโครงสร้างที่นำเสนอและระบบโครงสร้างของงานวิจัย [9] (Throughput: Symbols / Sec)

สำหรับในการทดสอบประสิทธิภาพของส่วนการทดสอบนี้จะเป็นการทดสอบในเรื่องของปริมาณผลลัพธ์ที่ได้จากการประมวลผลของระบบโครงสร้างที่นำเสนอและระบบโครงสร้างของงานวิจัย [9] ต่อเวลาที่ใช้ในการประมวลผล (Throughput: Symbols/Sec) เพื่อหาขีดความสามารถของประสิทธิภาพในการประมวลผลของระบบโครงสร้างดังกล่าว ในการประมวลผลการเข้ารหัสข้อมูลที่เป็นรูปภาพในแต่ละรูป ผลลัพธ์ที่ได้จะไม่เท่ากัน ดังนั้นในการทดสอบนี้จะใช้การหาค่าปริมาณผลลัพธ์ต่อเวลาในแต่ละรูป และการหาค่าเฉลี่ยทั้งหมดเพื่อเป็นตัวแทนของปริมาณผลลัพธ์ต่อเวลาของแต่ละระบบโครงสร้าง ผลที่ได้จากการทดสอบสำหรับระบบโครงสร้างที่นำเสนอและระบบโครงสร้างงานวิจัย [9] แสดงดังตารางที่ 5.7 สำหรับตารางที่ 5.8 แสดงผลเฉลี่ยของประสิทธิภาพในเรื่องปริมาณผลลัพธ์ต่อเวลาแยกออกเป็นแต่ละระบบโครงสร้าง

ตารางที่ 5.5 แสดงผลการประมวลผลของระบบโครงสร้างที่นำเสนอแบ่งตามแต่ละรูปภาพ

หมายเลขรูปภาพที่ใช้ทดสอบ	จำนวนลูกสัญญาณนาฬิกา		ระยะเวลาในการตอบสนอง	
	งานวิจัย [9]	งานวิจัยที่นำเสนอ	งานวิจัย [9]	งานวิจัยที่นำเสนอ
1	623673	225686	6.226767173	2.248517998
2	365977	128964	3.653923722	1.284873121
3	366605	128964	3.66019369	1.284873121
4	1025001	365046	10.23363618	3.636966853
5	851429	301602	8.500688898	3.004871925
6	2600741	934964	25.96586462	9.315081049
7	1151881	407684	11.50040935	4.06177083
8	1268331	446164	12.66304912	4.445148499
9	3034425	1079526	30.29577676	10.75535762
10	360736	126884	3.601597444	1.264150003
11	1115459	395204	11.13677117	3.937432127
12	2644425	949526	26.40200679	9.460162796
13	1677047	582404	16.74368011	5.802512678
14	1265179	445124	12.63157947	4.43478694
15	1018752	362964	10.17124601	3.61622381
16	1337001	469046	13.34865216	4.673122715
17	872121	314086	8.707278355	3.129250481
รวม	21578783	7663838	215.443121	76.35510257

ตารางที่ 5.6 แสดงผลเฉลี่ยประสิทธิภาพของการตอบสนองต่อการสั่งงานในแต่ละระบบโครงสร้าง

ระบบโครงสร้าง	ผลเฉลี่ยจำนวนลูกสัญญาณนาฬิกา (Clock Cycle)	ผลเฉลี่ยระยะเวลาในการตอบสนอง (มิลลิวินาที)
งานวิจัย [9]	1,269,340.176	12.67312477
ระบบโครงสร้างที่นำเสนอ	450,814	4.491476622

ในการหาค่าปริมาณผลลัพธ์คู่อันดับสัญลักษณ์ต่อเวลานั้น สามารถคำนวณได้ดังสมการที่ (5.2)

$$Throughput = \frac{Number\ of\ symbols}{T_{res}} \tag{5.2}$$

- Throughput คือค่าของปริมาณผลลัพธ์คู่อันดับสัญลักษณ์ต่อเวลา (Symbols/ Sec)
- Number of Symbols เป็นจำนวนผลลัพธ์คู่อันดับสัญลักษณ์ในแต่ละรูปภาพที่ทดสอบ
- T_{res} เป็นเวลาที่ใช้ในการประมวลผลการเข้ารหัสของแต่ละรูปภาพที่ทดสอบ

จากตารางที่ 5.8 แสดงผลเฉลี่ยของปริมาณผลลัพธ์คู่อันดับสัญลักษณ์ต่อเวลา ซึ่งในระบบโครงสร้างที่นำเสนอสามารถประมวลผลได้ผลลัพธ์ที่มากถึง 316,132,900.5 คู่อันดับสัญลักษณ์ต่อวินาที ส่วนระบบโครงสร้างของงานวิจัย [9] สามารถประมวลผลได้ที่ 111,852,747.3 คู่อันดับสัญลักษณ์ต่อวินาที จะเห็นได้ว่าระบบโครงสร้างที่นำเสนอมีความสามารถที่จะประมวลผลได้ปริมาณผลลัพธ์คู่อันดับสัญลักษณ์ต่อเวลามากกว่า เนื่องจากว่าระบบโครงสร้างที่นำเสนอในขณะที่การทำงานสามารถประมวลผลข้อมูลแบบส่งต่อเต็มรูปแบบ (Full Pipeline) ข้อมูลทั้ง 3 พาสที่ต้องประมวลผลจะเสมือนว่าข้อมูลจะถูกประมวลผลขนานกันในเวลา 1 ลูกสัญญาณนาฬิกา แต่ระบบโครงสร้างของงานวิจัย [9] จะยังคงเป็นการประมวลผลเพียง 1 พาสใน 1 ลูกสัญญาณนาฬิกา

ส่วนในอีกนัยหนึ่งที่ทำให้ระบบโครงสร้างที่นำเสนอสามารถประมวลผลได้ปริมาณผลลัพธ์คู่อันดับสัญลักษณ์ต่อเวลามีค่าสูง เนื่องจากว่าในช่วงที่การทำงานแบบส่งต่อเต็มรูปแบบ ตัวดำเนินการพื้นฐานการเข้ารหัสในแต่ละตัวประมวลผลแยกกัน ทำให้ภายใต้เงื่อนไขรูปแบบหนึ่งปริมาณผลลัพธ์คู่อันดับสัญลักษณ์สามารถมีจำนวนมากถึง 22 คู่อันดับสัญลักษณ์ในเวลา 1 ลูกสัญญาณนาฬิกา

ตารางที่ 5.7 แสดงผลของปริมาณผลลัพธ์ต่อเวลาสำหรับแต่ละระบบโครงสร้าง

หมายเลขรูปภาพที่ใช้ทดสอบ	จำนวนผลลัพธ์คู่อันดับต่อเวลา (Symbols/Sec)	
	ระบบโครงสร้างงานวิจัย [9]	ระบบโครงสร้างที่นำเสนอ
1	107157531.6	296748792.1
2	131606195.6	374261856.9
3	97164803.32	276791532.4
4	106472125.9	299589477.7
5	106335264.2	300819143.9
6	107914180.5	300811660.7
7	116306294.8	329307106.7
8	112468883.8	320394020.6
9	122188780	344182325.7
10	123638470.7	352249336.5
11	130071272.8	367898151.2
12	99807640.42	278549329.1
13	101846606.5	293887681.9
14	111986232.8	318969776.7
15	110894574.7	311909898.1
16	106417036.2	303977465.8
17	109220810.6	303911753.3
รวม	1,901,496,704	5,374,259,309

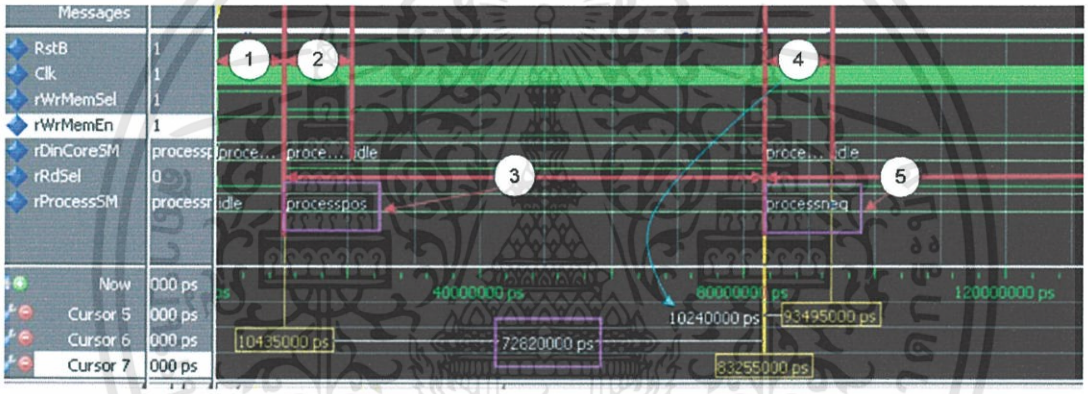
ตารางที่ 5.8 แสดงผลเฉลี่ยของปริมาณผลลัพธ์ต่อเวลาในแต่ละระบบโครงสร้าง

ระบบโครงสร้าง	ผลเฉลี่ยของปริมาณผลลัพธ์คู่อันดับสัญลักษณ์ต่อเวลา (Symbols/Sec)
ระบบโครงสร้างของงานวิจัย [9]	111,852,747.3
ระบบโครงสร้างที่นำเสนอ	316,132,900.5

5.3.6 การทดสอบประสิทธิภาพระบบหน่วยความจำคู่ของระบบโครงสร้างที่นำเสนอกับระบบหน่วยความจำเดี่ยวของงานวิจัย [9]

ในการทดสอบหัวข้อนี้เป็นการทดสอบเปรียบเทียบความสามารถของระบบโครงสร้างที่ใช้กระบวนการในการจัดเก็บข้อมูลในหน่วยความจำแบบคู่กับระบบโครงสร้างที่ใช้การจัดเก็บข้อมูลแบบหน่วยความจำเดี่ยว เพื่อเป็นการเปรียบเทียบความสามารถที่ส่งผลต่อประสิทธิภาพการประมวลผล

ในการจำลองการทำงานของระบบโครงสร้างที่นำเสนอและระบบโครงสร้างของงานวิจัย [9] จะใช้สัญญาณนาฬิกาที่ความถี่ 100 เมกะเฮิร์ต หรือ คาบเวลาที่ 10 นาโนวินาที (nano second; 10^{-9} second) เพื่อให้สามารถเปรียบเทียบการทำงานของทั้งสองระบบได้ สำหรับผลจากการทดสอบการทำงานของระบบหน่วยความจำแบบคู่แสดงได้ดังรูปด้านล่าง



รูปที่ 5.3 แสดงสัญญาณการนำเข้าข้อมูลลงหน่วยความจำแบบคู่ของระบบโครงสร้างที่นำเสนอ

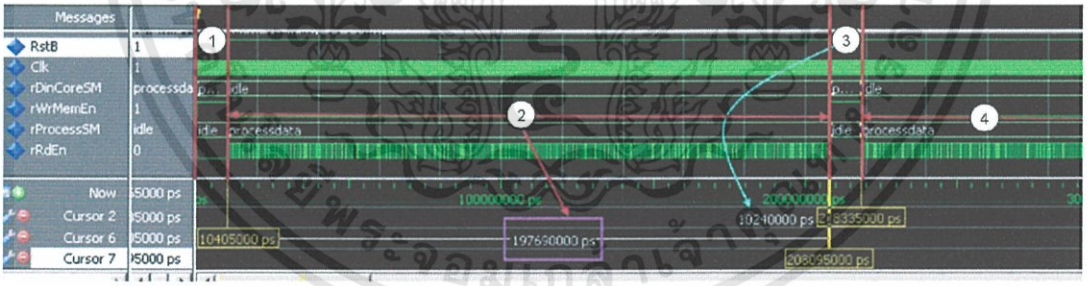
จากรูปที่ 5.3 แสดงสัญญาณการนำเข้าข้อมูลลงในหน่วยความจำแบบคู่สามารถอธิบายการทำงานได้ดังนี้

- ข้อมูลจะถูกนำเข้าสู่หน่วยความจำชุด A (rDinCoreSM = 'proccsspos')ในขณะนี้ส่วนการเข้ารหัสจะยังไม่ทำงาน (rProcessSM = 'idle')
- หลังจากข้อมูลถูกนำเข้าสู่หน่วยความจำชุด A เรียบร้อยแล้ว ส่วนการเข้ารหัสจะเริ่มประมวลผลข้อมูลจากหน่วยความจำชุด A (rProcessSM = 'processpos') ในขณะเดียวกันส่วนการนำเข้าข้อมูลจะนำข้อมูลใหม่เขียนลงหน่วยความจำชุด B (rDinCoreSM = 'processneg') ไปพร้อมกันด้วย

- ช่วงที่ส่วนการเข้ารหัสประมวลผลข้อมูลในหน่วยความจำชุด A ซึ่งเริ่มทำทันทีที่นำเข้าข้อมูลในหน่วยความจำชุด A เสร็จ และพร้อมกับการนำเข้าข้อมูลในหน่วยความจำชุด B
- เมื่อข้อมูลในหน่วยความจำชุด A ถูกประมวลผลข้อมูลจนเสร็จสิ้น ส่วนการนำเข้าข้อมูลจะนำข้อมูลชุดใหม่เพื่อเขียนลงในหน่วยความจำชุด A (rDinCoreSM = 'processpos') ต่อไป
- ส่วนของการประมวลผลการเข้ารหัสจะประมวลผลข้อมูลในหน่วยความจำชุด B ที่ได้เขียนไว้ก่อนหน้านั้นแล้ว (rProcessSM = 'processpos') ได้ทันที และพร้อมกับทำการนำเข้านำเข้าข้อมูลในหน่วยความจำชุด A

จากการสังเกต ข้อมูลใหม่ที่ถูกนำเข้าในหน่วยความจำชุดหนึ่ง ในช่วงที่กำลังประมวลผลข้อมูลที่ได้เตรียมไว้ก่อนหน้าในหน่วยความจำอีกชุด (เช่นกำลังประมวลผลข้อมูลในหน่วยความจำชุด A ข้อมูลใหม่จะถูกเขียนลงหน่วยความจำชุด B) โดยที่เวลาในการนำเข้าข้อมูลมีคาบเวลาน้อยกว่าการประมวลผลการเข้ารหัสมาก (ใช้เวลาในการนำเข้า 10240 นาโนวินาที ส่วนเวลาในการประมวลผลใช้เวลา 72820 นาโนวินาที ระบบจำลองการทำงานที่คาบสัญญาณนาฬิกาที่ 10 นาโนวินาที)

ในส่วนของสัญญาณของงานวิจัย [9] สามารถแสดงได้ดังรูปที่ 5.4



รูปที่ 5.4 แสดงสัญญาณการนำเข้าข้อมูลลงหน่วยความจำของงานวิจัย [9]

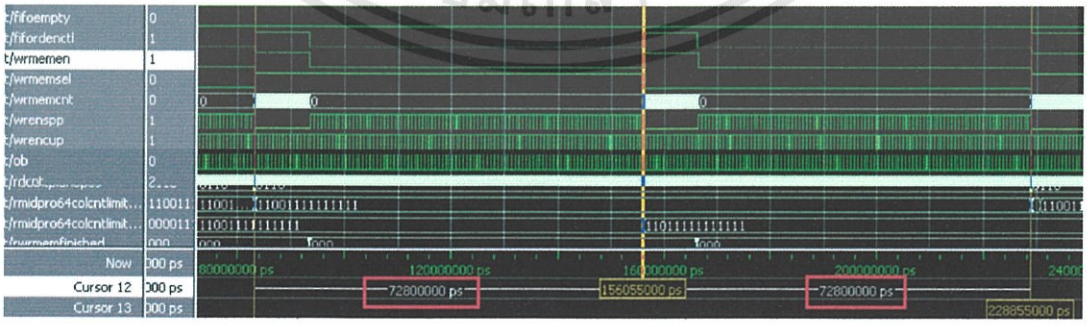
จากรูปที่ 5.4 แสดงสัญญาณการนำเข้าข้อมูลลงหน่วยความจำในระบบโครงสร้างของงานวิจัย [9] โดยการทำงานจะสามารถอธิบายได้ดังนี้

- ส่วนการนำเข้าข้อมูลลงหน่วยความจำ (rDinCoreSM = 'processdata') ในขณะนี้ส่วนการประมวลผลไม่สามารถประมวลผลการเข้ารหัสข้อมูลได้ (rProcessSM = 'idle')
- เมื่อข้อมูลนำเข้าเสร็จสิ้นเป็นที่เรียบร้อย ส่วนการประมวลผลการเข้ารหัสจะประมวลผลข้อมูลในหน่วยความจำ (rProcessSM = 'processdata') เนื่องจากระบบ

โครงสร้างนี้ใช้หน่วยความจำเดี่ยวดังนั้นส่วนการนำเข้าข้อมูลจะไม่ทำงานใดๆทั้งสิ้น (รอจนกว่าส่วนการประมวลผลการเข้ารหัสเสร็จนั่นคือ rDinCoreSM = 'idle') เวลาในการประมวลผลการเข้ารหัสจะใช้เวลาทั้งสิ้น 197690 นาโนวินาที

- ส่วนการประมวลผลการเข้ารหัสเสร็จสิ้นข้อมูลชุดใหม่จะถูกนำเข้าลงหน่วยความจำ (rDinCoreSM = 'processdata') ในจังหวะนี้ส่วนการประมวลผลการเข้ารหัสจะไม่สามารถเข้ารหัสข้อมูลใหม่ได้จึงต้องรอจนกว่าข้อมูลจะถูกนำเข้าเสร็จสิ้น (rProcessData = 'idle') เวลาที่ใช้ในการเขียนข้อมูลใหม่ลงหน่วยความจำจะใช้เวลาทั้งสิ้น 10240 นาโนวินาที
- ส่วนการประมวลผลการเข้ารหัสกลับมาทำงานอีกครั้ง (rProcessSM = 'processdata') ส่วนของการนำเข้าข้อมูลต้องรอคอยจนกว่าการเข้ารหัสจะเสร็จสิ้น (rDinCoreSM = 'idle')

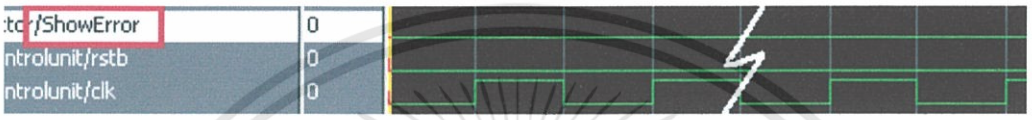
ดังนั้นเวลาที่ใช้ในการประมวลผลการเข้ารหัสข้อมูล 1 Code-block ของระบบโครงสร้างของงานวิจัย [9] ซึ่งจากรูปที่ 5.4 จะเห็นว่าเวลาในการเขียนข้อมูลลงหน่วยความจำจะใช้เวลา 10240 นาโนวินาที ส่วนเวลาที่ใช้ในการเข้ารหัสจะใช้เวลา 197690 นาโนวินาที หากปรับค่าเวลาในการประมวลผลการเข้ารหัสให้มีค่าเท่ากันกับระบบโครงสร้างที่นำเสนอ เวลาที่ได้จากระบบโครงสร้างของงานวิจัย [9] ก็ยังมีการใช้เวลามากกว่าอยู่ 10240 นาโนวินาที อันเนื่องมาจากต้องมีการรอคอยการนำเข้าข้อมูลถัดไป ทำให้ต้องเสียเวลารอข้อมูลเขียนลงหน่วยความจำเสร็จจึงจะประมวลผลใน Code-block ถัดไปได้ ต่างจากระบบโครงสร้างที่นำเสนอซึ่งแสดงในรูปที่ 5.5 จะเห็นว่าเวลาที่ใช้ในการเขียนข้อมูลสำหรับ Code-block ถัดไปจะอยู่ในช่วงที่ข้อมูลของ Code-block ปัจจุบันกำลังถูกประมวลผล ทำให้ประสิทธิภาพในการทำงานของระบบดีขึ้น โดยที่ไม่ต้องสูญเสียเวลาในการรอคอยข้อมูลใน Code-block ถัดไป



รูปที่ 5.5 แสดงเวลาของการประมวลผลข้อมูล 2 Code-block ในระบบโครงสร้างที่นำเสนอ

5.4 การทดสอบความถูกต้องของระบบโครงสร้างที่นำเสนอ

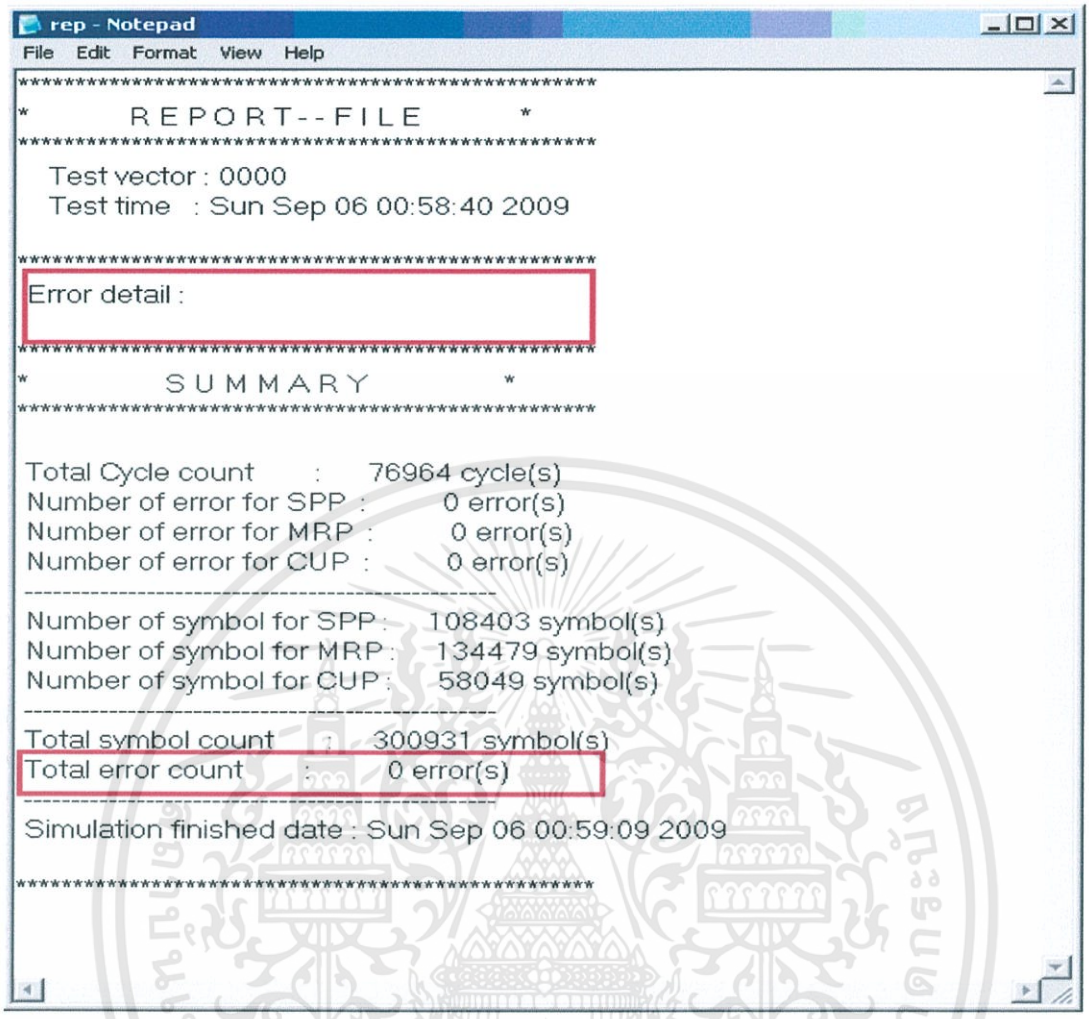
สำหรับในส่วนของหัวข้อนี้เป็นการทดสอบความถูกต้องของระบบโครงสร้างที่นำเสนอ โดยข้อมูลที่ใช้ทดสอบจะถูกสร้างมาจากตัวโปรแกรมที่ใช้เป็นมาตรฐานอ้างอิง ข้อมูลทั้งที่เป็นอินพุตและเอาต์พุตที่ได้จากโปรแกรม จะถูกนำมาใช้เป็นตัวทดสอบความถูกต้อง หากข้อมูลที่ผ่านการประมวลผลจากการจำลองระบบโครงสร้างทางฮาร์ดแวร์มีความถูกต้อง ข้อมูลที่รายงานออกมาจะไม่แสดงข้อผิดพลาด แต่ถ้าหากว่ามีผลที่ผิดพลาดเกิดขึ้นในการประมวลผลการเข้ารหัส ข้อมูลที่รายงานจะแสดงข้อผิดพลาดเหล่านั้นออกมา



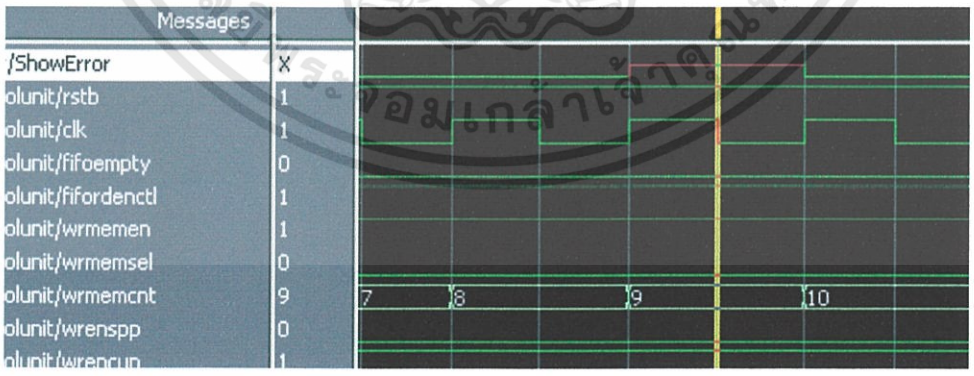
รูปที่ 5.6 สัญญาณที่ใช้แสดงข้อผิดพลาดของการเข้ารหัสที่ไม่ปรากฏความผิดพลาดใดๆ

จากรูปที่ 5.6 แสดงสัญญาณที่ใช้บ่งบอกข้อผิดพลาดของการเข้ารหัส โดยสัญญาณที่แสดงในรูปดังกล่าว แจ้งว่าการเข้ารหัสระนาบบิตของระบบโครงสร้างที่นำเสนอ ผลลัพธ์ที่ได้จากการจำลองการทำงานและผลลัพธ์ที่ได้จากตัวโปรแกรมอ้างอิงตามมาตรฐานมีค่าเหมือนกัน ซึ่งความถูกต้องที่ได้แสดงดังกล่าวสามารถตรวจสอบได้อีกทางหนึ่งจากการบันทึกรายงานการเปรียบเทียบค่าผลลัพธ์ที่ได้จากการจำลองกับโปรแกรมอ้างอิงมาตรฐาน สามารถแสดงได้ดังรูปที่ 5.7

จากรูปที่ 5.7 จะเห็นได้ว่าข้อมูลที่รายงานออกมาจะรายงานว่าไม่มีข้อผิดพลาดใดๆ เกิดขึ้นเลย หากมีผลลัพธ์ที่ไม่ตรงกัน ระบบจำลองการทำงานที่ใช้ตรวจสอบความผิดพลาด จะแสดงผลได้ดังรูปที่ 5.8 โดยค่าของสัญญาณที่เกิดความผิดพลาดจะแสดงเป็นสีแดง หรือมีค่าสัญญาณเป็น ‘X’ (Unknown value)

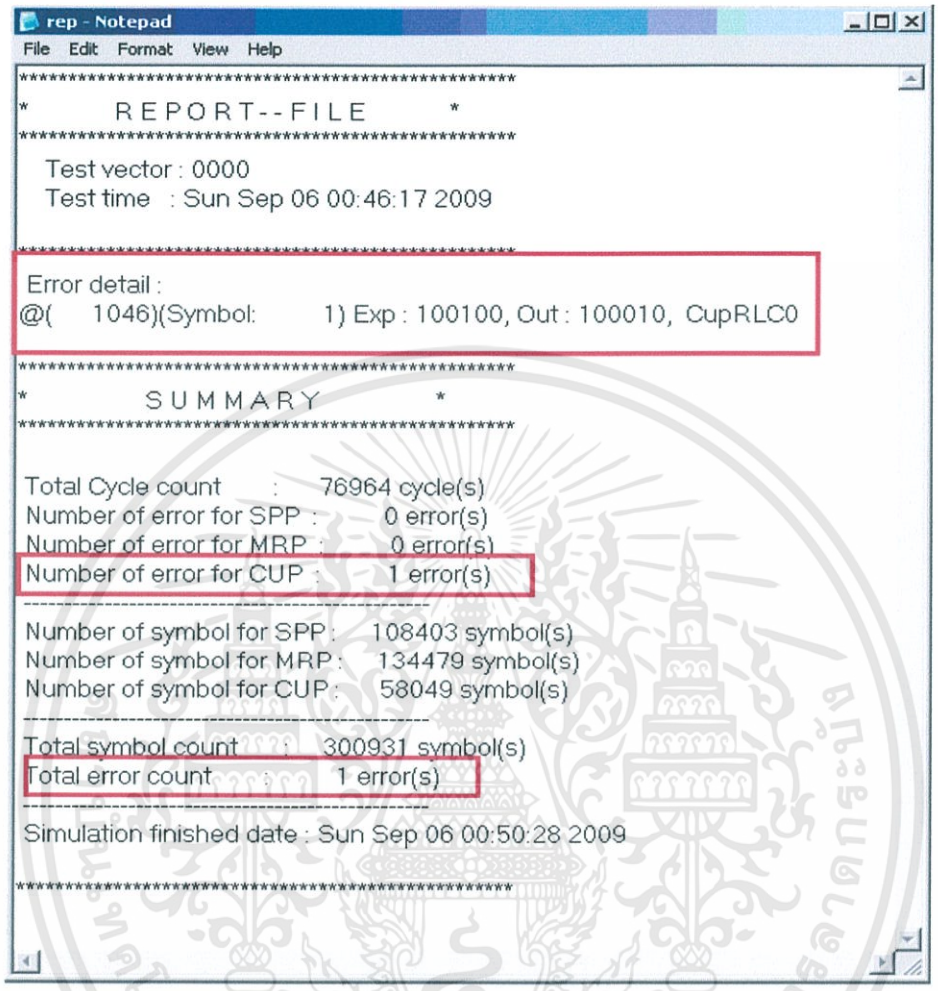


รูปที่ 5.7 ตัวอย่างไฟล์รายงานของการจำลองการทำงานของระบบโครงสร้างที่นำเสนอ



รูปที่ 5.8 แสดงสัญญาณที่บ่งบอกความผิดพลาดที่เกิดขึ้นในระบบโครงสร้างที่นำเสนอ

ในส่วนของไฟล์รายงานที่บักทึกความผิดพลาดสามารถแสดงได้ดังรูปที่ 5.9



รูปที่ 5.9 แสดงตัวอย่างของไฟล์รายงานที่แสดงถึงความผิดพลาดที่เกิดในระบบโครงสร้างที่นำเสนอ

5.5 การเปรียบเทียบผลการทดสอบประสิทธิภาพของแต่ละระบบโครงสร้าง

ในหัวข้อที่ได้กล่าวมาในแต่ละหัวข้อ จะเป็นผลที่ได้จากการทดสอบประสิทธิภาพของระบบโครงสร้างในแต่ละระบบ สำหรับในหัวข้อนี้จะเป็นการนำผลการทดสอบที่ได้มาเปรียบเทียบให้เห็นประสิทธิภาพในด้านต่างๆ เพื่อหาจุดเด่นของแต่ละผลการทดสอบในแต่ละระบบโครงสร้าง

การเปรียบเทียบอย่างแรกเป็นการเปรียบเทียบผลทางด้านทรัพยากรที่แต่ละระบบโครงสร้างใช้ โดยการเปรียบเทียบค่านี้จะใช้ผลที่ได้จากจำนวนทรัพยากรของวงจรสมมูล (Equivalent Gate Count) มาใช้ในการเปรียบเทียบ ซึ่งจากค่าที่ได้ในตารางที่ 5.2 และ ตารางที่ 5.3 จะเห็นว่าปริมาณของทรัพยากรที่ใช้ในระบบโครงสร้างที่นำเสนอมีค่าที่สูงกว่าถึง 1.422 เท่า หาก

นำระบบโครงสร้างที่นำเสนอไปใช้งานที่มีข้อจำกัดทางด้านทรัพยากรอาจจะเป็นปัญหาในการออกแบบได้ โดยขนาดของทรัพยากรที่ใหญ่เพิ่มมากขึ้น อันเนื่องมาจากการเพิ่มหน่วยความจำเป็นแบบหน่วยความจำคู่รวมทั้งการทำงานแบบส่งต่อพาส

การเปรียบเทียบอย่างที่สองเป็นการเปรียบเทียบผลทางด้านความเร็วที่ระบบโครงสร้างแต่ละระบบสามารถทำงานได้ ซึ่งผลที่ได้แสดงดังตารางที่ 5.4 จากผลที่ได้จะเห็นได้ว่าระบบโครงสร้างที่นำเสนอมีความเร็วมากกว่าระบบโครงสร้างของงานวิจัย [9] อยู่ถึง 1.002 เท่า (100.371 เมกะเฮิร์ต/ 100.160 เมกะเฮิร์ต) สาเหตุที่ความเร็วในการประมวลผลแตกต่างกันเล็กน้อยเนื่องจากว่าระบบโครงสร้างทั้งคู่ถูกออกแบบให้ทำงานแบบส่งต่อ

การเปรียบเทียบต่อมาเป็นการเปรียบเทียบผลการตอบสนองทางด้านเวลาในการสั่งการ (Processing Time) เนื่องจากการเข้ารหัสระนาบบิตในแต่ละรูปภาพมีค่าผลการตอบสนองที่ไม่เท่ากัน ดังนั้นจึงใช้ค่าที่ได้เฉลี่ยในแต่ละระบบโครงสร้างมาเปรียบเทียบกัน ซึ่งได้มาจากตารางที่ 5.6 เมื่อเปรียบเทียบแล้วจะได้ประสิทธิภาพในด้านการตอบสนองทางด้านเวลาในการสั่งการ โดยระบบโครงสร้างที่นำเสนอมีประสิทธิภาพที่ดีกว่าระบบโครงสร้างจากงานวิจัย [9] ในด้านนี้อยู่ถึง 2.822 เท่า (12.67312477 มิลลิวินาที/ 4.491476622 มิลลิวินาที) เนื่องจากระบบโครงสร้างที่นำเสนอออกแบบการทำงานแบบส่งต่อพาสและการทำงานของหน่วยความจำคู่

การเปรียบเทียบต่อมาเป็นการเปรียบเทียบผลทางด้านปริมาณผลลัพธ์ต่อเวลา (Throughput) โดยในการเปรียบเทียบนี้จะใช้ค่าที่ได้ของผลเฉลี่ยจากตารางที่ 5.8 ซึ่งเห็นได้ว่าระบบโครงสร้างที่นำเสนอมีผลทางด้านปริมาณผลลัพธ์ต่อเวลาที่ดีกว่าระบบโครงสร้างจากงานวิจัย [9] อยู่ถึง 2.826 เท่า เพราะว่าระบบโครงสร้างสามารถประมวลผลการทำงานทั้ง 3 พาสได้ภายในเวลา 1 ลูกสัญญาณนาฬิกาที่การทำงานแบบส่งต่อเต็มรูปแบบ ทำให้ดูเสมือนว่าระบบโครงสร้างทำงานทั้ง 3 พาสขนานกัน

ในส่วนการเปรียบเทียบถัดมาเป็นการเปรียบเทียบผลทางด้านความคุ้มค่าในการใช้งาน ซึ่งค่าที่คำนวณนี้ จะหาจากปริมาณผลลัพธ์ต่อเวลาต่อปริมาณของทรัพยากรที่ใช้ไป (Throughput/Gate) โดยเมื่อคำนวณออกมาแล้วผลที่ได้สามารถแสดงได้ดังตารางที่ 5.9

ตารางที่ 5.9 แสดงผลการคำนวณทางด้านความคุ้มค่าในแต่ละระบบโครงสร้าง

ระบบโครงสร้าง	ความคุ้มค่าของแต่ละระบบโครงสร้าง (Throughput / Gate)
ระบบโครงสร้างของงานวิจัย [9]	159.444
ระบบโครงสร้างที่นำเสนอ	316.842

จากตารางที่ 5.9 ความคุ้มค่าของแต่ละระบบโครงสร้างในการหาค่าจะสามารถหาได้จากสมการที่ (5.3)

$$Throughput\ per Gate = \frac{Avg.Throughput}{Gate} \tag{5.3}$$

ซึ่งโดยค่านี้จะบ่งบอกปริมาณผลลัพธ์คู่อันดับสัญลักษณ์ต่อทรัพยากรที่ใช้ในระยะเวลา 1 วินาที จะเห็นได้ว่าความคุ้มค่าที่ได้จากระบบโครงสร้างที่นำเสนอมีความคุ้มค่าที่ดีกว่าระบบโครงสร้างจากงานวิจัย [9] อยู่ถึง 1.987 เท่า หากนำระบบโครงสร้างที่นำเสนอไปออกแบบบนฮาร์ดแวร์ที่มีความเกี่ยวข้องกับด้านเศรษฐกิจแล้ว ระบบโครงสร้างที่นำเสนอจะมีความคุ้มค่าที่ดีกว่าเนื่องจากความสามารถในการประมวลผลสัญลักษณ์เฉลี่ยต่อเกทในเวลา 1 วินาที สูงกว่ามาก

ในการเปรียบเทียบสุดท้ายเป็นการเปรียบเทียบระบบโครงสร้างที่นำเสนอกับงานวิจัย [7] และงานวิจัย [8] เนื่องจากผลการทดลองของแต่ละระบบโครงสร้างที่ได้มานั้นจะใช้เทคโนโลยีในการออกแบบที่ต่างกัน แต่เพื่อใช้เป็นแนวทางในการดูภาพรวมของระบบโครงสร้างที่แตกต่างกันยังประโยชน์เพื่อเป็นแนวทางในการนำระบบโครงสร้างที่นำเสนอไปออกแบบด้วยเทคโนโลยีที่แตกต่างออกไป ในงานวิจัย [7] จะทำการเปรียบเทียบกับระบบโครงสร้างแบบขนาน โดยใช้ผลจากการจำลองการทำงานด้วยโปรแกรม MATLAB ของระบบโครงสร้างขนานออกมา ส่วนในงานวิจัย [8] จะออกแบบระบบโครงสร้างบนเทคโนโลยี TSMC 0.25 um สำหรับตารางที่ใช้ในการเปรียบเทียบระบบโครงสร้างสามารถแสดงได้ดังตารางที่ 5.11

ในงานวิจัย [7] จะใช้การจำลองการทำงานของระบบโครงสร้างขนาน ซึ่งผลการทำงานจะใช้การคำนวณตามสมการด้านล่าง

$$Column\ encoding\ time = P_2 * 2 + P_3 * 3 + P_4 * 4$$

โดยที่ค่า P_2, P_3 และ P_4 เป็นค่าของความน่าจะเป็นในการที่แต่ละคอลัมน์ต้องใช้เวลา 2, 3 และ 4 สัญญาณนาฬิกาในการประมวลผล เมื่อนำมาคำนวณแล้วจะได้ค่าดังนี้

$$Column\ encoding\ time = 0.18 * 2 + 0.21 * 3 + 0.61 * 4 = 3.43\ cycle$$

สำหรับการคำนวณหาค่าของเวลาในการประมวลผลต่อคอลัมน์ของระบบโครงสร้างที่นำเสนอ จะสามารถหาได้ดังนี้

- ข้อมูล 1 Code-block จะมีจำนวนคอลัมน์ที่จะถูกประมวลผลทั้งสิ้น 64 คอลัมน์ x 16 Stripe ดังนั้นจะได้จำนวนคอลัมน์ใน 1 Code-block เท่ากับ 1024 คอลัมน์
- เนื่องจากรูปภาพที่ใช้ในการทดสอบมีจำนวนทั้งสิ้น 17 รูปมีจำนวน Code-block ทั้งสิ้น 1296 Code-block จะได้จำนวนคอลัมน์ที่ต้องประมวลผลทั้งสิ้น 1327104 คอลัมน์
- เวลาในการประมวลผลของระบบโครงสร้างที่นำเสนอที่ใช้ประมวลผลของรูปทั้ง 17 รูปใช้เวลาทั้งหมด 7663838 สัญญาณนาฬิกา
- ดังนั้นเวลาเฉลี่ยที่ใช้ในการประมวลผลของระบบโครงสร้างที่นำเสนอ 7663838 สัญญาณนาฬิกา / 1327104 คอลัมน์ จะเท่ากับ 5.775 สัญญาณนาฬิกาต่อคอลัมน์

เพื่อเป็นการเปรียบเทียบกับงานวิจัย [8] ระบบโครงสร้างที่นำเสนอจะใช้การคำนวณความสามารถในการประมวลผลข้อมูลเป็นตำแหน่งจุดข้อมูล จึงจะสามารถเปรียบเทียบผลได้ โดยผลจากการคำนวณสามารถแสดงได้ดังตารางที่ 5.10 โดยผลจากตารางที่ 5.10 ประสิทธิภาพในการเข้ารหัสระนาบบิตที่นำเสนอสามารถประมวลผลข้อมูลได้ 66,247,891.22 จุดภาพต่อวินาทีที่ความเร็ว 100.371 เมกะเฮิร์ต แต่ในงานวิจัย [8] สามารถประมวลผลข้อมูลได้ 4.2 ล้านจุดภาพต่อวินาทีที่ความเร็ว 40 เมกะเฮิร์ต

ถึงแม้จำนวนสัญญาณลูกนาฬิกาที่ใช้ในระบบโครงสร้างที่นำเสนอใช้จำนวนลูกสัญญาณนาฬิกาที่มากกว่า แต่ว่าค่าที่ได้มาจากผลการจำลองทางฮาร์ดแวร์และเป็นการใช้รูปภาพในการทดสอบจริง แต่ว่าค่าที่ได้จากงานวิจัย [7] นั้นเป็นค่าที่ได้จากความน่าจะเป็นและคำนวณผลที่ได้ออกมาจากการเปรียบเทียบกับระบบโครงสร้างขนาน อีกทั้งไม่สามารถที่จะพิจารณาในเรื่องของความคุ้มค่าของระบบที่นำเสนอได้ อาจจะเป็นไปได้ที่ว่าระบบจากงานวิจัย [7] มีความคุ้มค่าที่น้อยกว่าหรือมากกว่าระบบโครงสร้างที่นำเสนอก็เป็นได้ ส่วนทรัพยากรที่ใช้ในงานวิจัย [8] มีการใช้ทรัพยากรที่น้อยกว่า แต่เป็นผลแสดงได้มาจากส่วนของการทำงานอย่างเดียว ไม่ได้นำส่วนของหน่วยความจำมาคิดด้วย โดยในงานวิจัย [8] ไม่ได้กล่าวว่าเมื่อนำหน่วยความจำมาสร้างด้วยโลจิกเกตจะใช้ทรัพยากรเท่าใด โดยบอกเพียงว่าใช้หน่วยความจำ 256 ตำแหน่ง x 22 บิตกับหน่วยความจำ 64 ตำแหน่ง x 10 บิต หากนำมาสร้างด้วยโลจิกเกอาจจะมีค่าที่น้อยกว่าหรือมากกว่าก็เป็นไปได้ เพราะเนื่องจากระบบโครงสร้างที่นำเสนอได้ใช้วิธีการเพิ่มจำนวนหน่วยความจำเป็นแบบระบบหน่วยความจำแบบคู่มาเพิ่มประสิทธิภาพในการประมวลผลการเข้ารหัสนั่นเอง

ตารางที่ 5.10 แสดงปริมาณจำนวนจุดภาพที่สามารถประมวลผลได้ใน 1 วินาทีของระบบโครงสร้างที่นำเสนอ

หมายเลขรูปภาพ	จำนวน Code-block ในแต่ละรูป	จำนวนจุดภาพที่คำนวณได้ (64 แถว x 64 คอลัมน์ x จำนวน Code-block)	ปริมาณจำนวนจุดภาพที่สามารถประมวลผลได้ใน 1 วินาที
1	48	196608	87438926.52
2	16	65536	51005814.45
3	16	65536	51005814.45
4	64	262144	72077643.43
5	48	196608	65429743.73
6	192	786432	84425674.44
7	64	262144	64539337.88
8	64	262144	58973057.94
9	192	786432	73120023.32
10	16	65536	51841949.01
11	64	262144	66577401.6
12	192	786432	83130916.13
13	64	262144	45177669.49
14	64	262144	59110844.23
15	64	262144	72491088.43
16	64	262144	56096108.75
17	64	262144	83772137
รวม	1296	5308416	1,126,214,151
เฉลี่ย	76.23529412	312,259.7647	66,247,891.22

ตารางที่ 5.11 แสดงการเปรียบเทียบระบบโครงสร้างที่นำเสนอกับงานวิจัย [7 - 8]

ระบบโครงสร้าง	เทคโนโลยีที่ใช้	ความเร็วที่สามารถทำงานได้	ทรัพยากรที่ใช้
ระบบโครงสร้างที่นำเสนอ	Xilinx Spatan3 xc3s1500-4fg676	100.371 MHz (5.775 Cycles/Column) (66,247.9 kpixels/s)	997,762 gate
งานวิจัย [7]	-	6,348 kpixels/s ≤ 3.43 *Cycles / Column*	-
งานวิจัย [8]	TSMC 0.25 um	40 MHz (312.259 kpixels/s)	105.9 kgates + memory (256 x 22 bits, 64 x 10 bits)

บทที่ 6
สรุปผลการทดลอง

6.1 บทนำ

ในส่วนของบทสุดท้ายนี้จะกล่าวถึงบทสรุปที่ได้จากการทดลองของระบบโครงสร้างที่นำเสนอ ซึ่งได้ใช้ความสามารถทางฮาร์ดแวร์มาเพิ่มประสิทธิภาพในการเข้ารหัสระนาบบิต ในส่วนของวิธีการที่ได้ถูกนำมาใช้ในการออกแบบระบบโครงสร้างที่นำเสนอ อันได้แก่ การประมวลผลการเข้ารหัสข้อมูลแบบส่งต่อ และการทำงานขนานกันในแต่ละช่วงพาส (Pass-pipelined) รวมไปถึงการใช้ระบบหน่วยความจำแบบคู่ อันทำให้ประสิทธิภาพในการเข้ารหัสระนาบบิตเพิ่มขึ้น ทั้งในแง่ของ ความเร็วในการทำงานของระบบโครงสร้างที่นำเสนอ ผลการตอบสนองด้านเวลาในการสั่งการ และปริมาณของผลลัพธ์คู่อันดับสัญลักษณ์

ถึงแม้ประสิทธิภาพที่ได้จะสูงขึ้นแต่ทรัพยากรที่ใช้ก็สูงขึ้นตาม แม้ว่าจำนวนทรัพยากรที่เพิ่มมากขึ้นแต่ความคุ้มค่าของระบบโครงสร้างที่นำเสนอมีผลที่ดี ซึ่งสามารถวัดได้จากปริมาณของผลลัพธ์คู่อันดับสัญลักษณ์ต่อทรัพยากรที่ใช้ต่อเวลา (Symbols/Gate ในเวลา 1 วินาที)

6.2 บทสรุปการวิเคราะห์ผลที่ได้จากระบบโครงสร้างที่นำเสนอ

จากผลการทดลองในบทที่ 5 จะเห็นได้ว่า ระบบโครงสร้างที่นำเสนอมีประสิทธิภาพในด้านต่างๆ สามารถสรุปได้ดังนี้

- ด้านความเร็วที่ระบบโครงสร้างสามารถทำงานได้ 100.371 เมกะเฮิร์ต หรือเป็น 1.002 เท่าของงานวิจัย [9]
- ด้านการตอบสนองด้านเวลาในการสั่งงาน (Processing Time) เวลาเฉลี่ยที่ระบบโครงสร้างที่นำเสนอสามารถตอบสนองได้อยู่ที่ 4.491 มิลลิวินาที ซึ่งสามารถตอบสนองได้เร็วกว่าระบบโครงสร้างของงานวิจัย [9] เป็น 2.822 เท่า
- ด้านปริมาณของผลลัพธ์จากการเข้ารหัสต่อเวลา (Throughput) ผลเฉลี่ยของปริมาณผลลัพธ์ต่อเวลาอยู่ที่ 316,132,900.5 คู่อันดับสัญลักษณ์ใน 1 วินาที (สัญลักษณ์ต่อวินาที)

ประสิทธิภาพที่ดีดังได้กล่าวมาข้างต้นนั้น เป็นผลอันเนื่องมาจากการออกแบบระบบโครงสร้างที่ใช้การทำงานแบบส่งต่อพาส (Pass-pipelined Architecture) เข้ามาใช้งานทำให้สามารถลดเวลาในการเข้ารหัสระนาบบิตได้เป็นอย่างดี รวมไปถึงการใช้วิธีการเพิ่มหน่วยความจำจากระบบ

หน่วยความจำเดี่ยวมาเป็นระบบหน่วยความจำแบบคู่ ยังสามารถช่วยลดเวลาในการรอคอยการนำเข้าข้อมูลจากส่วนภายนอกระบบโครงสร้างการเข้ารหัสระนาบบิต

แต่อย่างไรก็ตามประสิทธิภาพที่ได้มา ย่อมต้องแลกกับทรัพยากรที่เพิ่มมากขึ้น โดยทรัพยากรที่ระบบโครงสร้างที่นำเสนอใช้นั้นมีจำนวนเทียบเท่าการออกแบบโดยใช้โลจิกเกตทั้งสิ้น 997,762 Gate หากนำไปเปรียบเทียบกับงานวิจัย [9] แล้วจะมีการใช้ทรัพยากรสูงถึง 1.422 เท่า ถึงแม้ทรัพยากรที่ใช้จะสูงขึ้นแต่เมื่อนำไปคำนวณหาค่าความคุ้มค่าที่ได้จากปริมาณผลลัพธ์ที่ได้ต่อทรัพยากรที่ใช้ในเวลา 1 วินาทีแล้ว กลับพบว่า ในระยะเวลาการประมวลผลการเข้ารหัสระนาบบิตของระบบโครงสร้างที่นำเสนอ สามารถทำงานและให้ผลลัพธ์คู่อันดับสัญลักษณ์สูงถึง 316.842 คู่อันดับสัญลักษณ์ต่อ 1 โลจิกเกตในเวลา 1 วินาที และเมื่อนำไปเปรียบเทียบกับระบบโครงสร้างของงานวิจัย [9] ที่ใช้ทรัพยากรน้อย กลับยิ่งแสดงให้เห็นว่าระบบโครงสร้างที่นำเสนอมีความคุ้มค่ามากกว่าถึง 1.987 เท่า

ดังนั้นอาจจะกล่าวได้ว่าระบบโครงสร้างที่นำเสนอมีประสิทธิภาพที่สูงขึ้น ด้วยความสามารถทางฮาร์ดแวร์ที่ช่วยให้การทำงานเป็นแบบขนาน การทำงานแบบส่งต่อพาสที่ช่วยลดเวลาในการประมวลผลการเข้ารหัส และสุดท้าย ระบบหน่วยความจำแบบคู่ที่ช่วยลดระยะเวลาในการนำเข้าข้อมูลจากภายนอกระบบโครงสร้าง ยังผลให้ระบบโครงสร้างที่นำเสนอมีประสิทธิภาพที่ดี แม้ว่าจะมีข้อด้อยในเรื่องของการใช้ทรัพยากรที่สูงขึ้น หากนำระบบโครงสร้างที่นำเสนอนี้ไปใช้งานโดยมุ่งเน้นไปที่ทรัพยากรเป็นหลัก ระบบโครงสร้างที่นำเสนอนี้อาจจะไม่เหมาะสม แต่ถ้าไม่ได้มุ่งเน้นในเรื่องของทรัพยากรแล้ว ประสิทธิภาพที่เพิ่มมากขึ้นในหลายๆ ด้านก็เป็นตัวบ่งชี้ได้ว่าระบบโครงสร้างที่นำเสนอนี้ สามารถเพิ่มประสิทธิภาพต่อการบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 ได้มากยิ่งขึ้น พร้อมทั้งยังมีผลของความคุ้มค่าในการนำระบบโครงสร้างที่นำเสนอนี้ไปประยุกต์และออกแบบบนฮาร์ดแวร์ เพื่อประสิทธิภาพในการบีบอัดอีกด้วย

6.3 แนวทางในการพัฒนาต่อ

ถึงแม้ว่าประสิทธิภาพของระบบโครงสร้างที่นำเสนอจะมีประสิทธิภาพในการประมวลผลการเข้ารหัสที่ดี แต่ในยุคของโลกปัจจุบันนี้ เรื่องของการใช้งานทางด้านพลังงานในการประมวลผลก็เป็นประเด็นที่สำคัญอีกอย่างที่ไม่น้อยไปกว่าประสิทธิภาพที่ได้มาเช่นกัน ดังนั้นหากผู้ที่ได้ทำการศึกษาและวิจัยการเพิ่มประสิทธิภาพในการเข้ารหัสระนาบบิตของมาตรฐานการบีบอัดสัญญาณรูปภาพตามมาตรฐาน JPEG2000 ควรที่น่าจะพิจารณาในส่วนนี้ด้วย ซึ่งอาจจะใช้เทคนิคในการออกแบบอย่างอื่นเข้ามาใช้ร่วมด้วยในการลดปริมาณของพลังงานที่ใช้

ในอีกแนวทางหนึ่งของการพัฒนาต่อยอดของระบบโครงสร้างที่นำเสนอ นั่นคือในระบบโครงสร้างที่ได้นำเสนอจะเห็นได้ว่าในส่วนของโมดูลสุดท้ายที่ใช้สร้างผลลัพธ์คู่อันดับสัญลักษณ์นั้น ผลลัพธ์ที่ได้ออกมาจะเป็นแบบขนาน แต่ในการเข้ารหัสเลขคณิต (Arithmetic

Coding) จะใช้การนำเข้าข้อมูลที่เป็นแบบลำดับขั้น (Sequential) ทำให้อาจจะเกิดปัญหาที่ตามมาของการเชื่อมต่อระหว่างระบบโครงสร้างทั้งสอง (Bit-Plane กับ Arithmetic Coding) ซึ่งก็คือปัญหาคอขวดของการไหลของข้อมูล (Bottle Neck Problem)

ส่วนแนวทางสุดท้ายที่ผู้วิจัยขอเสนอแนะแนวทางในการพัฒนาก็คือ การนำระบบโครงสร้างที่นำเสนอนี้ ไปออกแบบบนวงจรรวมที่เป็นแบบเฉพาะขึ้นมา (ASIC: Application-Specific Integrated Circuit) โดยใช้ความสามารถในการวางผังของระบบโครงสร้าง (การวาง Layout) ซึ่งน่าจะเพิ่มประสิทธิภาพในด้านความเร็ว หรือด้านการใช้พลังงานที่สามารถลดลงได้

เอกสารอ้างอิง

[1] JPEG 2000 Part I Final Committee Draft Version 1.0, ISO/IEC JTC1/SC29 WG1 N1646R, March 2000.

[2] C. Lian, K. Chen, H. Chen and L. Chen, "**Analysis and architecture design of block-coding engine for EBCOT in JPEG 2000**", IEEE Transactions on Circuits and Systems for Video Technology, Vol.13, pp. 219 – 230, March 2003.

[3] J. Chiang, Y. Lin and C. Hsieh, "**Efficient pass-parallel architecture for EBCOT in JPEG 2000**", IEEE International Symposium on Circuits and Systems, Vol.1, pp. I-773 – I-776, May 2002.

[4] J. Chiang, C. Chang, Y. Lin and C. Hsieh, "**High throughput rate EBCOT architecture for JPEG2000**", Proceedings of the 46th IEEE International Midwest Symposium on Circuits and Systems, Vol.2, pp. 610 – 613, December. 2003.

[5] J. Chiang, C. Chang, Y. Lin, C. Hsich and C. Hsia, "**High-speed EBCOT with dual context-modeling coding architecture for JPEG2000**", Proceedings of the 2004 International Symposium on Circuits and Systems, Vol.3, pp. III-865 – III-868, May 2004.

[6] J. Chiang, C. Hsieh, J. Liu and C. Chien, "**Concurrent bit-plane coding architecture for EBCOT in JPEG2000**", 2006 IEEE International Symposium on Circuits and Systems, May 2006.

[7] Y. Li, R.E. Aly, M.A. Bayoumi and S.A. Mashali, "**Parallel high-speed architecture for EBCOT in JPEG2000**", 2003 IEEE International Conference on Acoustics, Speech and Signal Processing, Vol.2, pp. II-481 – II-484, April 2003.

[8] T. Tsai and L. Tsai, "**JPEG2000 encoder architecture design with fast EBCOT algorithm**", 2005 IEEE VLSI-TSA International Symposium on VLSI Design, pp. 279 – 282, April 2005.

[9] A.K. Gupta, D. Taubman and S. Nooshabadi, "**High speed VLSI architecture for bit plane encoder of JPEG2000**", The 2004 47th Midwest Symposium on Circuits and Systems, Vol.2, pp. II-233 – II-236, July 2004.

ภาคผนวก

ภาคผนวก ก.
ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

1. K. Mathiang and O. Chitsobhuk, "**Efficient pass-pipelined VLSI architecture for context modeling of JPEG2000l**," 2007 Asia-Pacific Conference on Communications (APCC'2007), pp. 63-66, Bangkok, Thailand, October 18-20, 2007.
2. K. Mathiang and O. Chitsobhuk, "**Improved Context Modeling Architecture of JPEG2000 on FPGA**," Proceeding of 2008 International Computer Symposium (ICS'2008), Vol.II, pp. 132-135, Taipei County, Taiwan, November 13-15, 2008.

IEEE

APCC

APCC 2007

2007 Asia-Pacific Conference on Communications

October 18-20, 2007

Bangkok, Thailand

PROCEEDINGS

ECTI Association

IEEE THAILAND SECTION

NECTEC a member of NSTDA

IEEE COMMUNICATIONS SOCIETY THAILAND CHAPTER

IEEE CAS Society Thailand Chapter

IEEE MTT/AP/ED THAILAND CHAPTER

IEEE LEOS LASERS & ELECTRO-OPTICS SOCIETY Thailand Chapter

IEEE COMMUNICATIONS SOCIETY

Proceedings of Asia-Pacific Conference on Communications 2007 **TAM1-3**

Efficient pass-pipelined VLSI architecture for context modeling of JPEG2000

Khomkris Mathiang
Faculty of Engineering,
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
Email: ratiodetector@hotmail.com

Orachat Chitsobhuk
Faculty of Engineering,
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
Email: kcoracha@kmitl.ac.th

Abstract—**In this paper, a design of pass-pipelined architecture for context modeling implemented on FPGA is proposed. The architecture is separated into 4 pipelined stages. As a result, the processing time and the critical path delay can be reduced while the multiple symbol context pairs are allowed to be generated simultaneously. Moreover, dual memories and data multiplexer are employed in order to accelerate the memory access. The proposed pass-pipelined architecture can process with the speed greater than 100 MHz and can generate up to 22 context-data pairs in one clock cycle.**

I. Introduction

The developed JPEG2000 [9] standard based on Discrete Wavelet Transform (DWT) can perform for both lossless and lossy compression. The JPEG2000 standard will be effective in a variety of applications such as internet, digital photography, digital library, printing, scanning, medical image, and mobile multimedia communication.

To improve the performance of DWT and bit-plane coding algorithms for real-time applications, there is a requirement for the development of the JPEG2000 system on the custom spatial VLSI for high-performance computation.

An Embedded Block Coding with Optimized Truncation (EBCOT) is the most time consuming function in the JPEG2000 process. It requires nearly 70% of the total coding time. Therefore, several researchers have been proposed the VLSI architectures to help reducing this computational time. One of the proposed architecture employs a pass-parallel data path [2] - [5], [8]. In this designed architecture, the context modeling scheme merges the three coding passes of bit-plane coding into a single pass to improve the system performance. Another architecture was proposed using a pass-predicting and clean-up pass skipping structure [7].

This paper is organized as follows. In section II, the proposed context modeling system is discussed. The experimental results are illustrated in section III while section IV present a conclusion.

II. The Proposed Context Modeling System

The proposed architecture processes all stripe columns (shown in Fig. 1) in one clock cycle for each coding pass. From this process, up to 22 context-data pairs can be generated. The extreme case of 22 generated context-data pairs occurs when all data is non-zero and all locations in the stripe columns of context window are active. As a result, 8 context-data pairs are generated for Signi□cant Propagation Pass (SPP). 4 context-data pairs are produced in the Magnitude Re□nement Pass (MRP) while 10 context-data pairs are created in the Cleanup Pass (CUP).

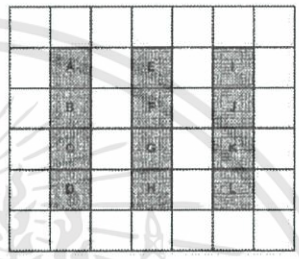


Fig. 1. stripe column of the context window.

In this paper, 3 state bits are used for signi□cant state (σ), magnitude re□nement state (σ'), and coding state (η) of all sample locations in the stripe columns of the context window. The signi□cant state variable (σ) is set to 0 for the □rst bit-plane of each code block. However, if the □rst non-zero bit has been coded, the σ is set to 1. In addition, the magnitude re□nement state variable (σ') is also initialized to 0. The σ' will be set to 1 when the magnitude bit of the location has been coded in MRP. The η is set to 0 at the beginning of each bit-plane. When the sample location has been coded, the η will be set to 1.

The context window of the proposed architecture is shown in Fig. 1. The shaded columns are current coding stripe column that will be processed for each coding pass. The sample location A, B, C, and D will be processed if the pass □ag of SPP is active. If the pass □ag of MRP is active, the sample location E, F, G, and H will be processed. However, if the pass □ag of CUP is active, the location I, J, K, and L will be processed.

1-4244-1373-7/07/$25.00 ©2007 IEEE

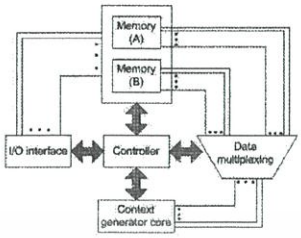


Fig. 2. Hardware structure.

The followings are terminologies used in this paper

i, j, k :location in the stripe column of the context window {0-6}.

$v[i,j]$:magnitude data at row i and column j in stripe column of the context window.

$\sigma[i,j]^{t_n}_{SPP}$:signi□cant data at row i and column j in the stripe column of the context window for SPP at time t_n

$\eta[i,j]^{t_n}_{SPP}$:coding state data at row i and column j in the stripe column of the context window for SPP at time t_n.

$\hat{\eta}[i,j]^{t_n}_{SPP}$:updated coding state data at row i and column j in the stripe column of the context window for SPP at time t_n.

$Pflag[i]^{t_n}_{SPP}$:pass □ag at row i in the stripe column of the context window of SPP at time t_n.

| :bit-wise OR operation.

& :bit-wise AND operation.

The pass concurrent membership testing methodology [1] is adopted to process all sample locations in the stripe columns.

A. The proposed architecture for context generating

Fig. 2 shows the hardware structure of the proposed JPEG2000 context modeling system. It includes the I/O interface, controller, memory, data multiplexing, and context generator core modules. The I/O interface contains FIFO, which is used to manage data coming to and sending from the proposed system. The controller module is responsible for interfacing to I/O interface module and managing the tasks of data multiplexer, memory, and context generator core module. In the proposed system, dual memories data multiplexing module are utilized to speed up the memory access time. Finally, the context generator core module generates context-data pairs using the proposed pipelined architecture.

B. The proposed context generator core module

1) The context generator process: The state diagram of the proposed context modeling architecture is shown in Fig. 3. The 4 stage pipelined is developed to optimize the critical path delay on FPGA. In stage 0, control unit generate control signals for reading, writing, and addressing to control the

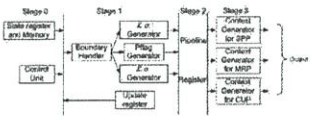


Fig. 3. The proposed pass-pipelined architecture block diagram.

memory units. This helps in managing and synchronizing the access to the memories. The stage variables are also initialized in this stage. In stage 1, the state, sign, and magnitude data will be processed in the boundary handler unit to manage the boundary of each code block by inserting zero for the □rst and the last columns. After boundary handling process, the data will be sent to $K\sigma'$, $K\sigma$, and $Pflag$ generators. The $K\sigma'$ generates the possible set of σ' bits, while the $K\sigma$ generates the possible set of σ bits. $Pflag$ produces the pass □ag. The update register unit updates the state and data back to the memory. The proposed pass-pipelined architecture is used in stage 2 to reduce the critical path delay. All the generated information from stage 1 will be registered. All the data from the pipelined register in stage 2 will be sent to the three passes of the context generators (SPP, MRP, and CUP). Each context generator contains several operation modules. The operation modules for the SPP are zero coding and sign coding modules. The magnitude re□nement coding module is used in the MRP. Finally, the zero coding, sign coding, and run-length coding modules are used for the CUP.

2) The pass-pipelined architecture: The pass-pipelined implementation is the main designed structure for the proposed context modeling system. The pipeline structure consists of 4 columns of the coding state shift register as illustrated in Fig. 4 for each coding pass. The η shift register is used to track the status of the input sample whether it has been process. The equations used to modify the values of all locations in the η shift register are shown as followed:

$$\hat{\eta}[i,j]^{t_n}_{SPP} = Pflag[i]^{t_{n-1}}_{SPP}: \quad i \in \{1,2,3,4\}, j \in \{2\} \tag{1}$$

$$\eta[i,j]^{t_n}_{MRP} = \hat{\eta}[i,k]^{t_{n-1}}_{SPP}: \quad i \in \{1,2,3,4\}, j \in \{3\}, k \in \{2\} \tag{2}$$

$$\hat{\eta}[i,j]^{t_n}_{MRP} = Pflag[i]^{t_{n-1}}_{MRP} | \eta[i,k]^{t_{n-1}}_{MRP}; \quad i \in \{1,2,3,4\}, j \in \{4\}, k \in \{3\} \tag{3}$$

$$\eta[i,j]^{t_n}_{CUP} = \hat{\eta}[i,k]^{t_{n-1}}_{MRP}; \quad i \in \{1,2,3,4\}, j \in \{5\}, k \in \{4\} \tag{4}$$

Only 4 columns (column 2 - 5) of the coding state shift register are chosen in the proposed architecture since the coding state columns of 0 and 1 are always zeros. The information in those columns has not been coding from any other passes. In addition, the coding state column 6 will not be processed any further. Therefore, it would be unnecessary to store in the memory and would be neglected. Reducing the

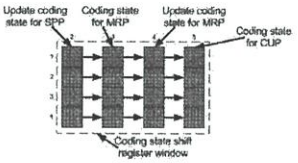


Fig. 4. η shift register window.

number of columns of the coding state shift register leads to hardware saving without any loss of information.

Equation 1 presents the update coding state, which relates to the pass flag value of the SPP. If some samples in the same stripe column of the context window has been coded in the SPP, the $\hat{\eta}$ state will be changed to the value of the represented sample at the location of the coded stripe column.

Equation 2 :The value of the coding state value of the MRP is changed based on the update coding state of the SPP before altered.

Equation 3 is used to assign the combination values of the pass flag of the MRP and the previous η state of the MRP to the $\hat{\eta}$ state of the MRP in the stripe column 4 from Fig. 4.

Equation 4 :The value of the coding state of the CUP is the same as the update coding state of the MRP before altered. If any location of the coding state of the CUP is zero, that location will be processed in the CUP.

C. Possible Set of σ Bits

A possible set of σ bits is a set of possible σ state in the stripe column of the context window. A σ can be changed immediately in the same stripe column that only happens in the SPP and CUP. The possible set of σ bits ($PS\sigma B$) can be generated as followed:

$$\begin{aligned} PS\sigma B[1]_x^{t_n} = \{&\sigma[0,j-1]_x^{t_n}, \sigma[0,j]_x^{t_n}, \\ &\sigma[0,j+1]_x^{t_n}, \sigma[1,j-1]_x^{t_n}, \\ &\sigma[1,j+1]_x^{t_n}, \sigma[2,j-1]_x^{t_n}, \\ &\sigma[2,j]_x^{t_n}, \sigma[2,j+1]_x^{t_n}\} \end{aligned} \quad (5)$$

$$\begin{aligned} PS\sigma B[2]_x^{t_n} = \{&\sigma[1,j-1]_x^{t_n}, \hat{\sigma}[1,j]_x^{t_n}, \\ &\sigma[1,j+1]_x^{t_n}, \sigma[2,j-1]_x^{t_n}, \\ &\sigma[2,j+1]_x^{t_n}, \sigma[3,j-1]_x^{t_n}, \\ &\sigma[3,j]_x^{t_n}, \sigma[3,j+1]_x^{t_n}\} \end{aligned} \quad (6)$$

$$\begin{aligned} PS\sigma B[3]_x^{t_n} = \{&\sigma[2,j-1]_x^{t_n}, \hat{\sigma}[2,j]_x^{t_n}, \\ &\sigma[2,j+1]_x^{t_n}, \sigma[3,j-1]_x^{t_n}, \\ &\sigma[3,j+1]_x^{t_n}, \sigma[4,j-1]_x^{t_n}, \\ &\sigma[4,j]_x^{t_n}, \sigma[4,j+1]_x^{t_n}\} \end{aligned} \quad (7)$$

$$\begin{aligned} PS\sigma B[4]_x^{t_n} = \{&\sigma[3,j-1]_x^{t_n}, \hat{\sigma}[3,j]_x^{t_n}, \\ &\sigma[3,j+1]_x^{t_n}, \sigma[4,j-1]_x^{t_n}, \\ &\sigma[4,j+1]_x^{t_n}, \sigma[5,j-1]_x^{t_n}, \\ &\sigma[5,j]_x^{t_n}, \sigma[5,j+1]_x^{t_n}\} \end{aligned} \quad (8)$$

The j is 1 if x is the SPP. Otherwise, the j is 5 for the CUP.

Where $\hat{\sigma}[i,j]_{SPP}^{t_n}$ for the SPP and can be generated as followed:

$$\hat{\sigma}[i,j]_{SPP}^{t_n} = ((Pflag[i]_{SPP}^{t_n} \& cv[i,j]) | \sigma[i,j]_{SPP}^{t_n}); \quad i \in \{1,2,3,4\}, j \in \{1\} \quad (9)$$

Where $\hat{\sigma}[i,j]_{CUP}^{t_n}$ for the CUP and can be generated as followed:

$$\hat{\sigma}[i,j]_{CUP}^{t_n} = ((Pflag[i]_{CUP}^{t_n} \& cv[i,j]) | \sigma[i,j]_{CUP}^{t_n}); \quad i \in \{1,2,3,4\}, j \in \{5\} \quad (10)$$

D. Possible Set of σ' Bits

A possible set of σ' bits is a set of possible σ' state in the stripe column of the context window. A σ' can be changed immediately in the same stripe column that only happens in the MRP. The possible set of σ' bits ($PS\sigma' B$) can be generated as followed:

$$\begin{aligned} PS\sigma' B[1]_{MRP}^{t_n} = \{&\sigma'[0,2]_{MRP}^{t_n}, \sigma'[0,3]_{MRP}^{t_n}, \\ &\sigma'[0,4]_{MRP}^{t_n}, \sigma'[1,2]_{MRP}^{t_n}, \\ &\sigma'[1,4]_{MRP}^{t_n}, \sigma'[2,2]_{MRP}^{t_n}, \\ &\sigma'[2,3]_{MRP}^{t_n}, \sigma'[2,4]_{MRP}^{t_n}\}; \end{aligned} \quad (11)$$

$$\begin{aligned} PS\sigma' B[2]_{MRP}^{t_n} = \{&\sigma'[1,2]_{MRP}^{t_n}, \hat{\sigma'}[1,3]_{MRP}^{t_n}, \\ &\sigma'[1,4]_{MRP}^{t_n}, \sigma'[2,2]_{MRP}^{t_n}, \\ &\sigma'[2,4]_{MRP}^{t_n}, \sigma'[3,2]_{MRP}^{t_n}, \\ &\sigma'[3,3]_{MRP}^{t_n}, \sigma'[3,4]_{MRP}^{t_n}\}; \end{aligned} \quad (12)$$

$$\begin{aligned} PS\sigma' B[3]_{MRP}^{t_n} = \{&\sigma'[2,2]_{MRP}^{t_n}, \hat{\sigma'}[2,3]_{MRP}^{t_n}, \\ &\sigma'[2,4]_{MRP}^{t_n}, \sigma'[3,2]_{MRP}^{t_n}, \\ &\sigma'[3,4]_{MRP}^{t_n}, \sigma'[4,2]_{MRP}^{t_n}, \\ &\sigma'[4,3]_{MRP}^{t_n}, \sigma'[4,4]_{MRP}^{t_n}\}; \end{aligned} \quad (13)$$

$$\begin{aligned} PS\sigma' B[4]_{MRP}^{t_n} = \{&\sigma'[3,2]_{MRP}^{t_n}, \hat{\sigma'}[3,3]_{MRP}^{t_n}, \\ &\sigma'[3,4]_{MRP}^{t_n}, \sigma'[4,2]_{MRP}^{t_n}, \\ &\sigma'[4,4]_{MRP}^{t_n}, \sigma'[5,2]_{MRP}^{t_n}, \\ &\sigma'[5,3]_{MRP}^{t_n}, \sigma'[5,4]_{MRP}^{t_n}\}; \end{aligned} \quad (14)$$

Where $\hat{\sigma'}[i,j]_{MRP}^{t_n}$ for the MRP and can be generated as followed:

$$\hat{\sigma'}[i,j]_{MRP}^{t_n} = (Pflag[i]_{MRP}^{t_n} | \sigma'[i,j]_{MRP}^{t_n}); \quad i \in \{1,2,3\}, j \in \{3\} \quad (15)$$

III. Experimental Results

The first step, we initialized the test vector data to testbench. After that, the simulated program will be processed the test vector data to testing the proposed system. For the test result, that show in TABLE.II presents a clock cycle count for the proposed architecture. In the proposed architecture, it consumes 15599 clock cycle per code block. For this mean, the 30 images for size 640x480 is process completely in a second. To get the hardware cost and system performance, we should the Xilinx ISE Webpack tool for synthesized the proposed

TABLE I
A HARDWARE COST FOR THE SYNTHESIZED PROPOSED ARCHITECTURE

Technology Library:	xc3s1500-4fg676
Number of Slices:	494
Number of Slice Flip Flops:	397
Number of 4 input LUTs:	826
Number of BRAMs:	14
Number of GClks:	1
Maximum Frequency (MHz):	110.461

TABLE II
A CLOCK CYCLE COUNT FOR THIS WORK

clock cycles/bit-plane	1040 cycles
clock cycles/code block	15599 cycles

architecture and selected Spartan-3 FPGA device technology. The results show in TABLE.I, present selected FPGA device technology, number of slices, flip flops, 4-input LUTs, and block rams (BRAMs). The maximum frequency from this FPGA device for the proposed architecture is 110.461 MHz.

IV. CONCLUSION

In this paper, it is presented the pipelined algorithm to process completed stripe column during each pass. The architecture is efficient design to enhance the clock rate. The results show the proposed architecture that has processing speed than 100 MHz and generated context-data between 0 to 22 context-data pairs a clock cycle. To reduce critical path delay, the proposed architecture used a 4-stage pipelined. For smoothly accessed memory, the dual buffer technique is used. The estimation hardware cost was synthesized on Spartan-3 platform with Xilinx ISE Webpack.

REFERENCES

[1] A.K. Gupta, D. Taubman, S. Nooshabadi, *High speed VLSI architecture for bit plane encoder of JPEG2000,* The 2004 47th Midwest Symposium on Circuits and Systems, vol. 2, pp. II-233 - II-236, July 2004.

[2] Jen-Shiun Chiang, Yu-Sen Lin, Chang-Yo Hsieh, *Efficient pass-parallel architecture for EBCOT in JPEG2000,* IEEE International Symposium on Circuits and Systems, vol. 1, pp. I-773 - I-776, May 2002.

[3] Jen-Shiun Chiang, Chang-Yo Hsieh, Jin-Chan Liu, Cheng-Chih Chien, *Concurrent bit-plane coding architecture for EBCOT in JPEG2000,* 2006 IEEE International Symposium on Circuits and Systems, May 2006.

[4] Jen-Shiun Chiang, Chun-Hau Chang, Yu-Sen Lin, Chang-You Hsieh, Chih-Hsieh Hsia, *High-speed EBCOT with dual context-modeling coding architecture for JPEG2000,* Proceedings of the 2004 International Symposium on Circuits and Systems, Vol. 3, pp. III-865 - III-868, May 2004.

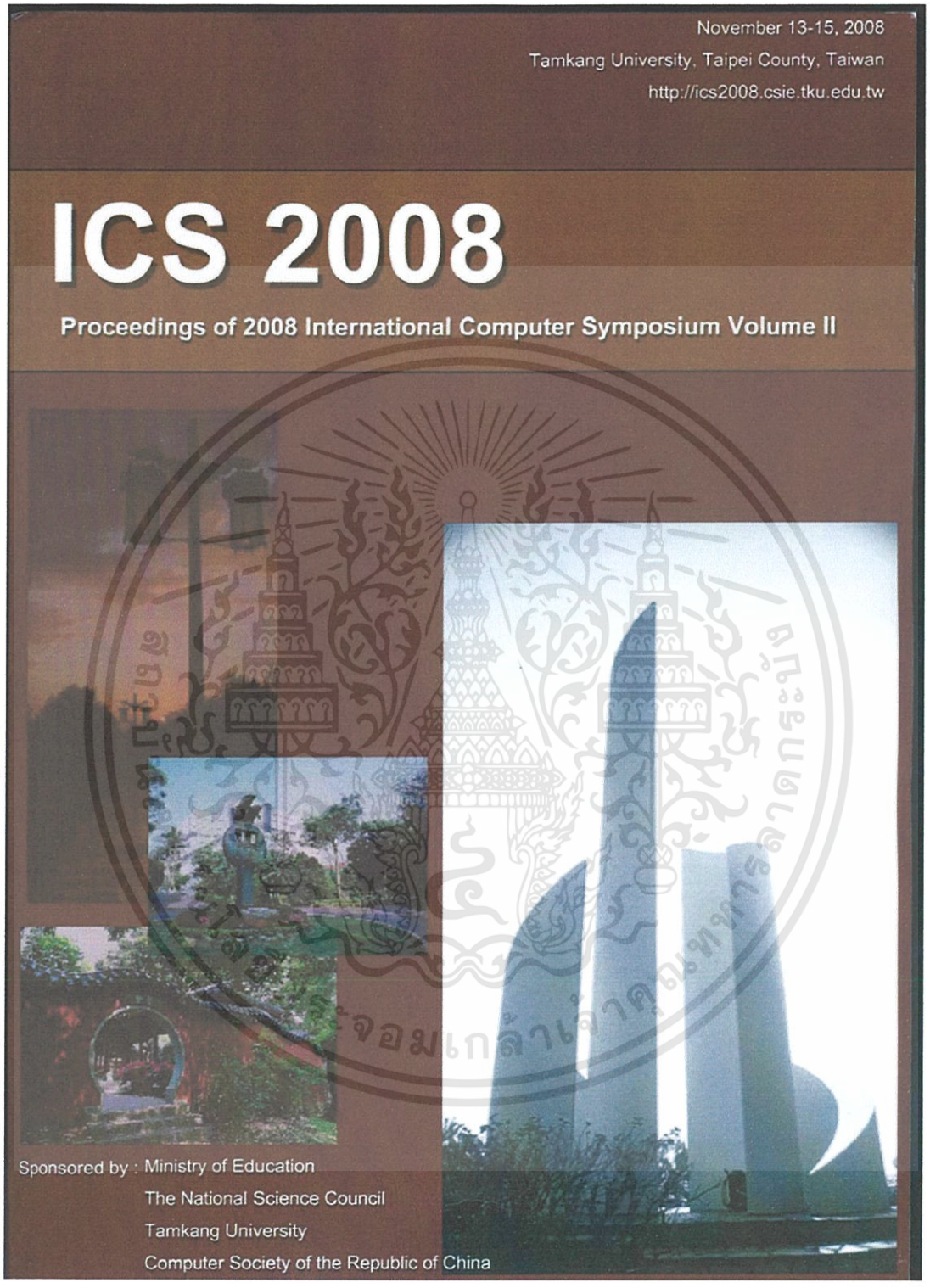
[5] Jen-Shiun Chiang, Chun-Hau Chang, Yu-Sen Lin, Chang-Yuo Hsieh, *High throughput rate EBCOT architecture for JPEG2000,* Proceedings of the 46th IEEE International Midwest Symposium on Circuits and Systems, Vol. 2, pp. 610 - 613, Dec. 2003.

[6] Chung-Jr Lian, Kuan-Fu Chen, Hong-Hui Chen, Liang-Gee Chen, *Analysis and architecture design of block-coding engine for EBCOT in JPEG 2000,* IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, pp. 219 - 230, March 2003.

[7] Tsung-Han Tsai, Lian-Tsung Tsai, *JPEG2000 encoder architecture design with fast EBCOT algorithm,* 2005 IEEE VLSI-TSA International Symposium on VLSI Design, pp. 279 - 282, April 2005.

[8] Yijun Li, R.E. Aly, M.A. Bayoumi, S.A. Mashali, *Parallel high-speed architecture for EBCOT in JPEG2000,* 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. II-481 - II-484, April 2003.

[9] *JPEG 2000 Part I Final Commitee Draft Version 1.0,* ISO/IEC JTC1/SC29 WG1 N1646R, March 2000.



Improved Context Modeling Architecture of JPEG2000 on FPGA

Khomkris Mathiang
Faculty of Engineering,
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
Email: ratiodetector@hotmail.com

Orachat Chitsobhuk
Faculty of Engineering,
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
Email: orachatc@yahoo.com

***Abstract*—In this paper, an improved context modeling architecture of JPEG2000 implemented on FPGA is proposed. The proposed architecture is based on a pass-pipelined structure with dual memories and data multiplexer. The proposed context modeling architecture allows multiple symbol context pairs to be generated simultaneously while the pass-pipelined structure helps to reduce the processing time and the critical path delay. Moreover, the dual memories and data multiplexer are employed in order to accelerate the memory access. The proposed pass-pipelined architecture can process with the speed greater than 100 MHz and can generate up to 22 context-data pairs in one clock cycle.**

I. Introduction

The developed JPEG2000 [9] standard based on Discrete Wavelet Transform (DWT) can perform for both lossless and lossy compression. The JPEG2000 standard will be effective in a variety of applications such as internet, digital photography, digital library, printing, scanning, medical image, and mobile multimedia communication.

To improve the performance of DWT and bit-plane coding algorithms for real-time applications, there is a requirement for the development of the JPEG2000 system on the custom spatial VLSI for high-performance computation.

An Embedded Block Coding with Optimized Truncation (EBCOT) is the most time consuming function in the JPEG2000 process. It requires nearly 70% of the total coding time. Therefore, several researchers have proposed the VLSI architectures to help reducing this computational time. One of the proposed architecture employs a pass-parallel data path [2] - [5], [8]. In this designed architecture, the context modeling scheme merges the three coding passes of bit-plane coding into a single pass to improve the system performance. Another architecture was proposed using a pass-predicting and clean-up pass skipping structure [7].

However, in this paper, a design of pass-pipelined architecture for context modeling implemented on FPGA is proposed. The architecture is separated into 4 pipelined stage. As a result, the processing time and the critical path delay can be reduced while the multiple symbol context pairs are allowed to be generated simultaneously.

This paper is organized as follows. In section II, the reference and proposed context modeling system is described.

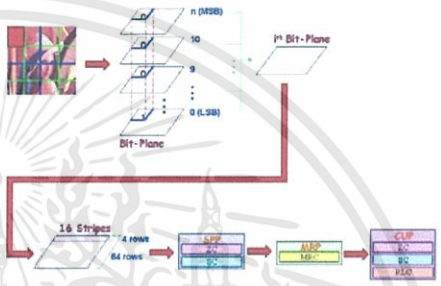


Fig. 1. Context modeling system block diagram

The experimental results are illustrated in section III followed by a conclusion in section IV.

II. The Context Modeling System

A. An overview of a context modeling system

In this section, an overview of context modeling system in bit-plane coding(BPC) process is described.

In BPC module, the quantized wavelet coefficients are encoded bit-plane by bit-plane, starting with the most significant bit-plane which is the sign data. The i^{th} bit-plane is separated into stripes. Each stripe consists of 4 rows, which is processed in 3 coding pass, Significant Propagation Pass (SPP), Magnitude Refinement Pass (MRP), and Cleanup Pass (CUP) respectively. There are four possible coding operations used for generating the value of context-data pair. Two operations of zero and sign coding are employed in the SPP while the magnitude refinement coding is operated in the MRP. Additionally, three operations of zero, sign, and run-length coding are implemented in the CUP.

B. A review on a context modeling system

There are several context modeling system are proposed in the literature [1]-[8]. This section provides a review of the context modeling system proposed in [1]. Fig.2 shows the

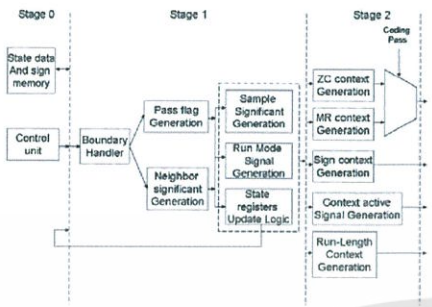


Fig. 2. The exciting context modeling system from [1].

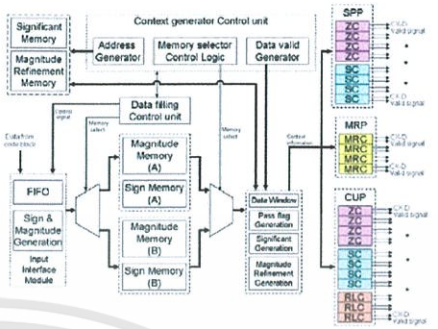


Fig. 3. The proposed pass-pipelined architecture block diagram.

hardware architecture of the context modeling from [1].

The context modeling architecture can be separated into three states. State 0 is the control unit providing the interface to the main control unit of the encoding system and controlling the memory read/write signal generation for data, sign, state bits, and coding pass information. State 1 is used to handle data and generate simple information for compression. The boundary handler in this state is used to monitor data at the boundary of the image. Finally, state 2 encodes information through three coding pass (SPP, MRP, and CUP) using sign, ZC, MR, and run-length coding operations in order to generate the context-data pairs. It also generates the context-active signal, depending on the current pass, run-mode signal, run-interrupt signal, pass flags, and data bits to identify the active context-data pairs. In addition, a multiplexer is required to select between the context-data pair generated from the ZC operation and that of MR operation depending on the current coding pass. The system can process a complete stripe column in a single clock cycle for each coding pass. Besides, up to 10 coding-data pairs can be produced in a single clock cycle. The extreme case of 10 context-data pairs happens only during the CUP when the run-mode condition is satisfied while a run interrupt occurs immediately at the first sample location in the stripe column. This system repeatedly computes for the context-data pairs three times for each bit-plane since the coding pass can be operated only one pass at a time.

C. The proposed pass-pipelined context modeling system

Fig.3 shows a block diagram of the proposed pass-pipelined context modeling system. The proposed system consists of five modules as followed.

1) Input Interface Module: : This module is used to exchange information between external data and the proposed context modeling system. First, input data is inserted into FIFO. Then, input data from FIFO are sent to the sign and magnitude generator block in order to separate the sign from the magnitude of the data. The last process of this module is to pass the magnitude and sign data to Dual Memories Module.

2) Dual Memories Module and data multiplexer: : In this module, dual memories of A and B are used to store both magnitude and sign memory. While memory A is used by the context generation module, the data for the next code block will be loaded into memory B. Each set of memory will be enabled by the main controller through the data multiplexer. This helps the context generation module to process input data continuously without an interruption of input interface delay.

3) Context Generator Control Unit: : This module is used to generate the control signals such as addressing the memory, selecting data, and control valid data for the context generation module.

4) Context Generation Module: : This module is used to generate pass flag, significant state, magnitude refinement state, and coding state. The pass flag equation is illustrated in equation (1) - (6):

$$Pflag[1]^{t_n}_{SPP} = \vec{K}Sigma(1)\&\neg SigmaSPP(1); \quad (1)$$

$$Pflag[2]^{t_n}_{SPP} = (\vec{K}Sigma(2)|(Pflag[1]^{t_n}_{SPP}\&v[1,1])) \&\neg SigmaSPP(2); \quad (2)$$

$$Pflag[3]^{t_n}_{SPP} = (\vec{K}Sigma(3)|(Pflag[2]^{t_n}_{SPP}\&v[2,1])) \&\neg SigmaSPP(3); \quad (3)$$

$$Pflag[4]^{t_n}_{SPP} = (\vec{K}Sigma(4)|(Pflag[3]^{t_n}_{SPP}\&v[3,1])) \&\neg SigmaSPP(4); \quad (4)$$

$$Pflag[i]^{t_n}_{MRP} = SigmaMRP(i)\&\neg\eta[i,3]^{t_n}_{MRP}; i \in 1,2,3,4 \quad (5)$$

$$Pflag[i]^{t_n}_{CUP} = \neg\eta[i,5]^{t_n}_{CUP}; i \in 1,2,3,4 \quad (6)$$

where $\vec{K}Sigma(i)$ is a neighbor significant state of row i. The $\eta[i,j]_x^{t_n}$ is a coding state for row i and column j at current time. Witch x, it is a set of $\{MRP, CUP\}$.

The possible set of significant bit show as below:

$$PS\sigma B[1]_x^{t_n} = \left\{ \begin{array}{ll} \sigma[0,j-1]_x^{t_n}, & \sigma[0,j]_x^{t_n}, \\ \sigma[0,j+1]_x^{t_n}, & \sigma[1,j-1]_x^{t_n}, \\ \sigma[1,j+1]_x^{t_n}, & \sigma[2,j-1]_x^{t_n}, \\ \sigma[2,j]_x^{t_n}, & \sigma[2,j+1]_x^{t_n} \end{array} \right\} \quad (7)$$

$$PS\sigma B[2]_x^{t_n} = \left\{ \begin{array}{ll} \sigma[1,j-1]_x^{t_n}, & \hat{\sigma}[1,j]_x^{t_n}, \\ \sigma[1,j+1]_x^{t_n}, & \sigma[2,j-1]_x^{t_n}, \\ \sigma[2,j+1]_x^{t_n}, & \sigma[3,j-1]_x^{t_n}, \\ \sigma[3,j]_x^{t_n}, & \sigma[3,j+1]_x^{t_n} \end{array} \right\} \quad (8)$$

$$PS\sigma B[3]_x^{t_n} = \left\{ \begin{array}{ll} \sigma[2,j-1]_x^{t_n}, & \hat{\sigma}[2,j]_x^{t_n}, \\ \sigma[2,j+1]_x^{t_n}, & \sigma[3,j-1]_x^{t_n}, \\ \sigma[3,j+1]_x^{t_n}, & \sigma[4,j-1]_x^{t_n}, \\ \sigma[4,j]_x^{t_n}, & \sigma[4,j+1]_x^{t_n} \end{array} \right\} \quad (9)$$

$$PS\sigma B[4]_x^{t_n} = \left\{ \begin{array}{ll} \sigma[3,j-1]_x^{t_n}, & \hat{\sigma}[3,j]_x^{t_n}, \\ \sigma[3,j+1]_x^{t_n}, & \sigma[4,j-1]_x^{t_n}, \\ \sigma[4,j+1]_x^{t_n}, & \sigma[4,j-1]_x^{t_n}, \\ \sigma[5,j]_x^{t_n}, & \sigma[4,j+1]_x^{t_n} \end{array} \right\} \quad (10)$$

The possible set of significant bit is a set of significant state that is neighbor position for all position in current processed column. This data may change the state immediately in current process time.

The $\hat{\sigma}[i,j]_x^{t_n}$ can be generated from equation (11), where i is the i^{th} row and j is the j^{th} column of process window. The t_n is a current time when x is a current pass to process. Equation (11) is used to generate immediately updated significant state for the significant propagation pass.

$$\hat{\sigma}[i,j]_{SPP}^{t_n} = \left(\left(Pflag[i]_{SPP}^{t_n} \& v[i,j] \right) | \sigma[i,j]_{SPP}^{t_n} \right); \quad i \in \{1,2,3,4\}, j \in \{1\} \quad (11)$$

Equation (12) is used to generate immediately updated significant state for the cleanup pass.

$$\hat{\sigma}[i,j]_{CUP}^{t_n} = \left(\left(Pflag[i]_{CUP}^{t_n} \& v[i,j] \right) | \sigma[i,j]_{CUP}^{t_n} \right); \quad i \in \{1,2,3,4\}, j \in \{5\} \quad (12)$$

The neighbor magnitude refinement state can be generated using equation (14) - (16). This is called possible set of magnitude refinement state bits.

$$PS\sigma' B[1]_{MRP}^{t_n} = \left\{ \begin{array}{ll} \sigma'[0,2]_{MRP}^{t_n}, & \sigma'[0,3]_{MRP}^{t_n}, \\ \sigma'[0,4]_{MRP}^{t_n}, & \sigma'[1,2]_{MRP}^{t_n}, \\ \sigma'[1,4]_{MRP}^{t_n}, & \sigma'[2,2]_{MRP}^{t_n}, \\ \sigma'[2,3]_{MRP}^{t_n}, & \sigma'[2,4]_{MRP}^{t_n} \end{array} \right\} \quad (13)$$

$$PS\sigma' B[2]_{MRP}^{t_n} = \left\{ \begin{array}{ll} \sigma'[1,2]_{MRP}^{t_n}, & \hat{\sigma}'[1,3]_{MRP}^{t_n}, \\ \sigma'[1,4]_{MRP}^{t_n}, & \sigma'[2,2]_{MRP}^{t_n}, \\ \sigma'[2,4]_{MRP}^{t_n}, & \sigma'[3,2]_{MRP}^{t_n}, \\ \sigma'[3,3]_{MRP}^{t_n}, & \sigma'[3,4]_{MRP}^{t_n} \end{array} \right\} \quad (14)$$

$$PS\sigma' B[3]_{MRP}^{t_n} = \left\{ \begin{array}{ll} \sigma'[2,2]_{MRP}^{t_n}, & \hat{\sigma}'[2,3]_{MRP}^{t_n}, \\ \sigma'[2,4]_{MRP}^{t_n}, & \sigma'[3,2]_{MRP}^{t_n}, \\ \sigma'[3,4]_{MRP}^{t_n}, & \sigma'[4,2]_{MRP}^{t_n}, \\ \sigma'[4,3]_{MRP}^{t_n}, & \sigma'[4,4]_{MRP}^{t_n} \end{array} \right\} \quad (15)$$

$$PS\sigma' B[4]_{MRP}^{t_n} = \left\{ \begin{array}{ll} \sigma'[3,2]_{MRP}^{t_n}, & \hat{\sigma}'[3,3]_{MRP}^{t_n}, \\ \sigma'[3,4]_{MRP}^{t_n}, & \sigma'[4,2]_{MRP}^{t_n}, \\ \sigma'[4,4]_{MRP}^{t_n}, & \sigma'[5,2]_{MRP}^{t_n}, \\ \sigma'[5,3]_{MRP}^{t_n}, & \sigma'[5,4]_{MRP}^{t_n} \end{array} \right\} \quad (16)$$

Because the magnitude refinement state can be immediately change its value in the current process column, its value must be immediately updated using equation (17).

$$\hat{\sigma}'[i,j]_{MRP}^{t_n} = \left(Pflag[i]_{MRP}^{t_n} | \sigma'[i,j]_{MRP}^{t_n} \right); \quad i \in \{1,2,3\}, j \in \{3\} \quad (17)$$

where $\sigma'[i,j]_{MRP}^{t_n}$ is a previous magnitude refinement state before immediately updated magnitude refinement state at current time.

5) Primitive Operator Generation Module: : The primitive operators in context modeling consist of 4 primitive operators: Zero, Sign, Magnitude Refinement, and Run-length Coding. The SPP employs the zero and sign coding operators. The MRP consists of only the magnitude refinement coding. In addition, the CUP utilizes all three operators: zero, sign, and run-length coding.

In the proposed system, all the three passes (SPP, MRP, and CUP) are fully processed in parallel. This can increase the performance of the proposed system upto 22 context-data pairs generated in a single clock cycle

III. Experimental Results

In this section, the proposed pass-pipelined context modeling architecture is implemented on FPGA. The designed architecture is compiled and synthesized using Xilinx ISE Webpack tool and ported onto Spartan-3 FPGA platform. Several standard images are selected from the database as test images such as camera man, Lena, Mandrill, and pirate. The performance comparison of the context-modeling system between [1] and the proposed system are presented in TABLE.I. and TABLE.II. TABLE.I presented a comparison of hardware cost and maximum clock speed while TABLE.II

ประวัติผู้เขียน

ชื่อ-นามสกุล	นาย คมกริช มาเที่ยง
วัน เดือน ปีเกิด	10 กันยายน 2524
ที่อยู่	190 หมู่ 12 ต. มะต้อง อ. พรหมพิราม จ. พิษณุโลก 65180

ประวัติการศึกษา

2548	เกียรตินิยมอันดับ 2 วิศวกรรมศาสตรบัณฑิต
	สาขาวิชาวิศวกรรมคอมพิวเตอร์
	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
2545	ประกาศนียบัตรวิชาชีพชั้นสูง
	สาขาเทคนิคคอมพิวเตอร์
	สถาบันเทคโนโลยีราชมงคล วิทยาเขตพระนครเหนือ

ประวัติการทำงาน

2551 - 2552	ตำแหน่ง Software Engineer
	บริษัท Toyota Tsusho electronics (Thailand) Co, Ltd.
2548 - 2549	ตำแหน่ง Hardware Engineer
	บริษัท Design-Gateway Co, Ltd.