การออกแบบการเข้ารหัสและถอดรหัสพาริตีเช็คความหนาแน่นต่ำแบบกึ่งไซไซคลิก

ENCODER AND DECODER DESIGN OF QUASI-CYCLIC LOW-DENSITY PARITY-CHECK CODES

ประพันธ์ ลีกุล

PRAPAN LEEKUL

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2551

KMITL-2008-EN-M-010-139

การออกแบบการเข้ารหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก

ENCODER AND DECODER DESIGN OF QUASI – CYCLIC LOW-DENSITY PARITY-CHECK CODES

ประพันธ์ ลี้กุล

PRAPAN LEEKUL

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2551

KMITL-2008-EN-M-010-139

ENCODER AND DECODER DESIGN OF QUASI – CYCLIC LOW-DENSITY PARITY-CHECK CODES

PRAPAN LEEKUL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN TELECOMMUNICATIONS ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2008

KMITL-2008-EN-M-010-139

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การออกแบบการเข้ารหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก

Encoder and Decoder Design of Quasi-Cyclic Low-Density Parity-Check Codes

นักศึกษา นายประพันธ์ ลี้กุล

รหัสประจำตัว 48060944

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมโทรคมนาคม

อาจารย์ที่ปรึกษาวิทยานิพนธ์ ผศ.ดร.พรชัย ทรัพย์นิธิ

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
รศ.ดร.ปราโมทย์	วาดเขียน	
รศ.ดร.สุวิพล	สิทธิชีวภาค	
ผศ.ดร.เผ่าภัค	ศิริสุข	
ผศ.ดร.สมเกียรติ	ฤกษ์วรัญญู	
ผศ.ดร.พรชัย	ทรัพย์นิธิ	

วัน / เดือน/ ปี ที่สอบ 2 พฤษภาคม 2551 เวลา 14.00-16.00 น.

สถานที่สอบ ณ ห้องประชุม 3 ชั้น 5 อาคาร A

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.รวีวรรณ ชินะตระกูล)

คณบดีบัณฑิตวิทยาลัย

วันที่.....23.....**เดือน**.....พฤษภาคม.....**พ.ศ**.....2551.....

หัวข้อวิทยานิพนธ์	การออกแบบการเข้ารหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก
นักศึกษา	นายประพัน ลี้กุล
รหัสประจำตัว	48060944
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขา	วิศวกรรมโทรคมนาคม
พ.ศ.	2551
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ.ดร.พรชัย ทรัพย์นิธิ

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้ เสนอวิธีการออกแบบวงจรเข้ารหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก (Quasi – Cyclic Low-Density Parity-Check Codes : QC – LDPC) ที่ใช้แก้ไขข้อผิดพลาดในระบบสื่อสารเชิงดิจิตอล การเข้ารหัสจะเน้นที่อัตรารหัสที่สูง (High Rate) ในกรณีที่บล็อกข้อมูลในการเข้ารหัสมีขนาดยาวขึ้น ทรัพยากรที่ต้องการและความซับซ้อนเพิ่มขึ้นในลักษณะเชิงเส้น ในส่วนการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำได้นำอัลกอลิทึมมการแพร่กระจายความน่าเชื่อถือ (Belief propagation : BP) มาใช้ร่วมกับแทนเนอร์กราฟ (Tanner graph) ในโดเมนลอการิทึม (Log-Domain Algorithm) การลดความซับซ้อนในวงจรถอดรหัส อาศัยการประมาณค่า $\log((e^{x}+1)/(e^{x}-1))$ นอกจากนี้ทำการเปรียบเทียบวิธีการถอดรหัส ด้วยเลขจำนวนจริง (floating-point) กับการถอดรหัสด้วยเลขตายตัว (fixed-point) ในช่องสัญญาณรบกวนเกาส์แบบขาว เพื่อจำลองสมรรถนะเปรียบเทียบอัตราบิตผิดพลาดของข้อมูลโดยการจำลอง

Thesis Title	Encoder and Decoder Design of Quasi – Cyclic Low-Density Parity-Check Codes
Student	Mr. Prapan Leekul
Student ID.	48060944
Degree	Master of Engineering
Program	Telecommunication Engineering
Year	2008
Thesis	Asst. Prof. Dr. Pornchai Supnithi

ABSTRACT

This thesis presents encoder and decoder design of high-rate Quasi-Cyclic Low-Density Parity-Check (QC-LDPC). The advantage of the proposed encoding schemes is the linearity of resources and complexity as the input block size increases. The decoder employs the Belief Propagation algorithm together with Tanner graph of the parity-check matrix in the log-domain. The function $\log((e^{x}+1)/(e^{x}-1))$ equation is approximated in order to reduce the complexity in the decoding process. Performance comparison of floating-point decoding and fixed-point decoding in AWGN is in forms of the bit error rate.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความกรุณาจาก ผศ.ดร. พรชัย ทรัพย์นิธิ ที่ให้ความช่วยเหลือ แนะนำ แก้ไขปัญหา และคอยดูแลเอาใจใส่ติดตามงาน ชี้แนะข้อบกพร่อง รวมทั้งฝึกฝนให้ข้าพเจ้ามีแนวคิดและทักษะในการนำเสนอและดำเนินงานซึ่งมีประโยชน์อย่างยิ่งต่อตัวข้าพเจ้า

ขอขอบพระคุณ ดร. เกียรติศักดิ์ ศรีพิมานวัฒน์ ฝ่ายวิจัยและพัฒนาเทคโนโลยีโทรคมนาคมและเครือข่ายคอมพิวเตอร์ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ ที่ให้คำแนะนำทางวิชาการตลอดจนความดูแลเอาใจใส่ และความปรารถนาดีที่มีต่อข้าพเจ้า

ขอขอบพระคุณ ผศ. นิภา ฉลาฉุจี ผศ. ณรงค์ เหมกรณ์ อ. ศรวัฒน์ ชิวปรีชา ที่ให้คำแนะนำทางวิชาการตลอดจนความดูแลเอาใจใส่ และความปรารถนาดีที่มีต่อข้าพเจ้า

ขอขอบคุณสมาชิกในห้องปฏิบัติการสื่อสารดาวเทียม (Satellite Communication Laboratory) ทุกท่าน สำหรับความช่วยเหลือ มิตรภาพ และความจริงใจที่มีให้แก่กันตลอดระยะเวลาที่ข้าพเจ้าปฏิบัติงานในห้องวิจัยแห่งนี้

ขอขอบคุณสมาชิกในห้องปฏิบัติการสื่อสารเชิงแสงและควอนตัมทุกท่าน สำหรับความช่วยเหลือ มิตรภาพ และความจริงใจที่มีให้แก่กันตลอดระยะเวลาที่ข้าพเจ้าปฏิบัติงาน

คุณความดีอันใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ขอมอบแด่บิดาและมารดาของข้าพเจ้า ที่ได้ให้ชีวิต โอกาสทางการศึกษา กำลังใจและความห่วงใยมาโดยตลอด และญาติของข้าพเจ้าที่คอยสนับสนุน ช่วยเหลือและให้แนะนำปรึกษาที่ดี ตลอดจนครูและอาจารย์ทุกท่านที่กรุณาประสิทธิ์ประสาทความรู้ให้แก่ข้าพเจ้า

ประพัน ลี้กุล

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....I

บทคัดย่อภาษาอังกฤษ.....II

กิตติกรรมประกาศ.....III

สารบัญ.....IV

สารบัญตาราง.....VIII

สารบัญรูป.....IX

บทที่ 1 บทนำ.....1

1.1 ความเป็นมาและความสำคัญของปัญหา.....1

1.2 วัตถุประสงค์และขอบเขตวิทยานิพนธ์.....4

1.3 ขอบเขตการวิจัย.....5

1.4 ขั้นตอนการศึกษา.....5

บทที่ 2 ทฤษฎีและหลักการรหัสช่องสัญญาณ.....6

2.1 หลักการรหัสช่องสัญญาณ.....6

2.1.1 หลักการเข้ารหัสแบบบล็อก (Block Codes).....6

2.1.2 เมตริกซ์กำเนิด (Generator Matrix).....9

2.1.3 เมตริกซ์พาริตี้เช็ค (Parity-check matrix).....11

2.1.4 รหัสแฮมมิง (Hamming Code).....11

2.1.5 การถอดรหัสบล็อกเชิงเส้น.....12

2.1.6 การถอดรหัสด้วยซินโดรม (Syndrome Decoding).....12

2.2 เอสเอนอาร์ของสัญลักษณ์รหัส.....12

2.3 เกนของรหัส (Coding Gain).....14

2.4 หลักการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำ.....15

2.4.1 การเข้ารหัสด้วยเมตริกซ์กำเนิด G.....15

2.4.2 การเข้ารหัสด้วยเมตริกซ์พาริตี้เช็ค H.....16

2.5 หลักการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ.....18

2.5.1 แทนเนอร์กราฟ (Tanner Graph).....18

2.5.2 วิธีคำนวณการถอดรหัสโดยใช้แทนเนอร์กราฟ (Tanner Graph).....20

สารบัญ (ต่อ)

หน้า

2.5.2.1 ขั้นตอนการถอดรหัสในรูปแบบของความน่าจะเป็น.....21
2.5.2.2 ขั้นตอนการถอดรหัสในรูปแบบของลอการิทึม.....23

บทที่ 3 เอฟพีจีเอพื้นฐานและการออกแบบ.....25
3.1 ประเภทวงจรรวมแบบดิจิตอล.....25
3.2 ไอซีมาตรฐานและวงจรรวมแบบมาตรฐาน.....26
3.3 วงจรรวมประเภทเอสิค (ASIC).....27
3.3.1 วงจรรวมแบบเกตอาเรย์.....27
3.3.2 วงจรรวมแบบฟูลัสตัม.....28
3.3.3 วงจรรวมแบบเซลล์มาตรฐาน (Standard Cell).....28
3.4 วงจรรวมแบบพีแอลดี.....28
3.4.1 อุปกรณ์เอสพีแอลดี.....29
3.4.1.1 พรอม (Programmable Read-Only Memory: PROM).....30
3.4.1.2 พีแอลเอ (Programmable Logic Array: PLA).....31
3.4.1.3 พีเอแอล (Programmable Array Logic: PAL).....32
3.4.1.4 อีพีแอลดี (Erasable Programmable Logic Device EPLD).....32
3.4.2 อุปกรณ์ซีพีแอลดี (Complex Programmable Logic Device: CPLD).....33
3.4.3 อุปกรณ์เอฟพีจีเอ (Field Programmable Gate Array: FPGA).....33
3.5 ภาษาวีเอชดีแอล (VHSIC Hardware Description Language: VHDL).....34
3.5.1 โครงสร้างพื้นฐานของภาษาวีเอชแอล.....35
3.5.2 ความสามารถของภาษาวีเอชดีแอล (Capability).....37
3.5.3 องค์ประกอบของภาษาวีเอชดีแอล.....38
3.5.3.1 การกำหนดการเชื่อมต่อ.....38
3.5.3.2 การกำหนดรูปแบบของการบรรยาย.....39
3.6 การออกวงจรดิจิตอลด้วยเอฟพีจีเอ.....40
3.6.1 การออกแบบ.....41
3.6.2 การติดตั้ง.....43

สารบัญ (ต่อ)

หน้า

3.6.3 การโปรแกรมข้อมูลลงชิพเอฟพีจีเอ.....44

บทที่ 4 การออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก.....45

4.1 ทฤษฎีการออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก.....45

4.1.1 รหัสพาริตี้เช็คความหนาแน่นต่ำประเภทอาร์เรย์.....46

4.1.2 รหัสพาริตี้เช็คความหนาแน่นต่ำแบบอาร์เรย์คุณสมบัติของ Cycle.....46

4.2 การออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก.....49

4.2.1 วงจรเข้ารหัสแบบอนุกรม.....51

4.2.2 วงจรเข้ารหัสแบบกึ่งขนาน.....52

4.2.3 วงจรเข้ารหัสแบบขนาน.....53

4.3 การออกแบบการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก.....57

4.3.1 โครงสร้างชุดการหาค่าบิตโนดเริ่มต้น ($L(c_j)$).....58

4.3.2 โครงสร้างชุดการหาค่าเช็คโนด ($L(r_{ji})$).....58

4.3.3 โครงสร้างชุดการหาค่าบิตโนด ($L(q_{ij})$).....60

4.3.4 โครงสร้างชุดการหาค่าการตัดสินใจแบบละเอียด ($L(Q_j)$).....60

4.3.5 โครงสร้างชุดการหาค่าการตัดสินใจแบบหยาบ (c_j).....61

บทที่ 5 ผลการออกแบบและทดสอบการทำงาน.....63

5.1 ผลการออกแบบและจำลองการทำงาน.....63

5.1.1 โครงสร้างและการออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำ.....64

5.1.1.1 โครงสร้างชุดดำเนินการเลื่อนบิตแบบวนกลับ.....64

5.1.1.2 โครงสร้างชุดการบวกลักษณะมอดูโล – 2.....66

5.1.2 โครงสร้างและการออกแบบการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ.....67

5.1.2.1 โครงสร้างชุดหน่วยความจำชั่วคราว (Ram).....67

5.1.2.2 โครงสร้างชุดตัวดำเนินการคูณ (Multiply).....69

5.1.2.3 โครงสร้างชุดการแยกสัญลักษณ์ (ABS).....70

5.1.2.4 โครงสร้างชุดตัวดำเนินการบวกแบบไม่คิดเครื่องหมาย.....70

5.1.2.5 โครงสร้างชุดหน่วยความจำถาวรที่ใช้ในการเปิดตาราง (LUT).....72

สารบัญ (ต่อ)

หน้า

5.1.2.6 โครงสร้างชุดการรวมบิตสัญลักษณ์กับบิตข้อมูล (Sign) 76
5.1.2.7 โครงสร้างชุดตัวดำเนินการบวกแบบคิดเครื่องหมาย (+, -).....77
5.1.2.8 โครงสร้างชุดตัวดำเนินการในการตัดสินใจแบบหยาบ.....78
5.2 ผลการเปรียบเทียบประสิทธิภาพ.....80
5.2.1 การวิเคราะห์ผลการใช้ทรัพยากร.....80
5.2.2 การเปรียบเทียบประสิทธิภาพ.....80
บทที่ 6 สรุปผลและข้อเสนอแนะ.....81
6.1 สรุปผลการวิจัย.....81
6.2 ปัญหาและข้อเสนอแนะ.....82

บรรณานุกรม.....86

ภาคผนวก.....88
ภาคผนวก ก. โปรแกรมจำลองการเข้ารหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ.....88
ภาคผนวก ข. ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่.....93

ประวัติผู้เขียน.....99

สารบัญตาราง

ตารางที่		หน้า
2.1	การเข้ารหัสแบบบล็อก ที่ $k = 4, n = 7$ และ $R = 4 / 7$	7
5.1	ตารางแทนค่า $\log((e^x + 1)/(e^x-1))$	75
5.2	การใช้จำนวน Logic Elements	81

สารบัญรูป

รูปที่		หน้า
1.1	แผนผังส่วนประกอบหลักในระบบสื่อสารเชิงดิจิตอล	3
2.1	โครงสร้างของรหัสบล็อกเชิงเส้น	7
2.2	ช่องสัญญาณสมมาตรไบนารี	13
2.3	กราฟแสดงความน่าจะเป็นของบิตผิดพลาดในกรณีที่มีการเข้ารหัสช่องสัญญาณ	14
2.4	การรับส่งข้อมูลในระบบสื่อสารดิจิตอล	18
2.5	แทนเนอร์กราฟ (Tanner Graph)	19
2.6	สัญลักษณ์ในแทนเนอร์กราฟ (Tanner Graph)	20
3.1	ไดอะแกรมการออกแบบวงจรดิจิตอล	25
3.2	ไอซีซีมอส (CMOS)	26
3.3	วงจรรวมทีทีแอล (TTL)	26
3.4	ไดอะแกรมของวงจรรวมเอสิก	27
3.5	ไดอะแกรมของวงจรรวมแบบพีแอลดี	28
3.6	วงจรพื้นฐานของพีแอลดีในรูปของผลคูณร่วมบวก	29
3.7	ไดอะแกรมแอสพีแอลดี	30
3.8	วงจรพื้นฐานภายในของพรอม	30
3.9	วงจรพื้นฐานภายในของพีแอลเอ	31
3.10	วงจรพื้นฐานภายในของพีเอแอล	32
3.11	โครงสร้างของซีพีแอลดี	33
3.12	โครงสร้างทั่วไปของเอฟพีจีเอ	34
3.13	ระดับของการออกแบบวงจรดิจิตอลในภาษาวีเอชดีแอล	36
3.14	การเชื่อมต่อและสถาปัตยกรรมของภาษาวีเอชดีแอล	38
3.15	บล็อกไดอะแกรมการบรรยายการเชื่อมต่อขององค์ประกอบ	39
3.16	การบรรยายเชิงพฤติกรรมขององค์ประกอบ	39
3.17	ขั้นตอนการออกแบบวงจรดิจิตอลบนเอฟพีจีเอ	40
3.18	การออกแบบงวจรด้วยภาษาวีเอชดีแอล	41
3.19	การออกแบบวงจรด้วยวิธีการวาดผังวงจร (Schematic)	42
3.20	การออกแบบด้วยการเขียนขั้นตอนการทำงาน (State diagram)	42

สารบัญรูป (ต่อ)

รูปที่		หน้า
3.21	สัญญาณที่ได้จากการจำลองการทำงานในรูปแบบ Waveform	43
3.22	การโปรแกรมลงในชิพเอฟพีจีเอ	44
4.1	วงจรการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบ XOR Network	49
4.2	บล็อกไดอะแกรมของชุดเข้ารหัส QC-LDPC	50
4.3	วงจรการเข้ารหัสแบบอนุกรม	51
4.4	วงจรการเข้ารหัสแบบกึ่งขนาน	52
4.5	วงจรการเข้ารหัสแบบขนาน	54
4.6	ขั้นตอนการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ	57
4.7	โครงสร้างการหาค่าบิตโนดเริ่มต้น ($L(c_i)$)	58
4.8	โครงสร้างการหาค่าเช็คโนด ($L(r_{ji})$)	59
4.9	โครงสร้างการหาค่าบิตโนด ($L(q_{ij})$)	60
4.10	โครงสร้างการหาค่าการตัดสินใจแบบละเอียด ($L(Q_i)$)	61
4.11	โครงสร้างการหาค่าการตัดสินใจแบบหยาบ (c_i)	61
5.1	ผังวงจรระดับ RTL ของชุดดำเนินการเลื่อนบิตแบบวนกลับ	65
5.2	ผลการจำลองการทำงานของตัวดำเนินการเลื่อนบิตแบบวนกลับ	66
5.3	ผังวงจรระดับ RTL ของชุดการบวกลักษณะมอดูโล-2	66
5.4	ผลการจำลองการทำงานตัวดำเนินการบวกแบบมอดูโล-2	67
5.5	ผังวงจรระดับ RTL ของชุดหน่วยความจำชั่วคราว (Ram)	68
5.6	ผลการจำลองการทำงานหน่วยความจำชั่วคราว	69
5.7	ผังวงจรระดับ RTL ของชุดตัวดำเนินการคูณ (Multiply)	69
5.8	การจำลองการทำงานของตัวดำเนินการคูณ (Multiply)	69
5.9	ผังวงจรระดับ RTL ของชุดการแยกเครื่องหมาย (ABS)	70
5.10	ผลการจำลองการทำงานของบล็อกการแยกเครื่องหมาย (ABS)	70
5.11	ผังวงจรระดับ RTL ของชุดดำเนินการบวกแบบไม่คิดเครื่องหมาย (+)	71
5.12	ผลการจำลองการทำงานของตัวดำเนินการบวกแบบไม่คิดเครื่องหมาย (+)	72
5.13	ผังวงจรระดับ RTL ของชุดหน่วยความจำถาวรที่ใช้ในการเปิดตาราง (LUT)	73
5.14	การแบ่งบิตข้อมูล	74
5.15	ผลการจำลองการทำงานของหน่วยความจำถาวรที่ใช้ในการเปิดตาราง (LUT)	75

สารบัญรูป (ต่อ)

รูปที่		หน้า
5.16	ผังวงจรระดับ RTL ของชุดดำเนินการรวมบิตสัญลักษณ์กับบิตข้อมูล (Sign)	76
5.17	ผลการจำลองการทำงานของตัวดำเนินการรวมบิตสัญลักษณ์กับบิตข้อมูล (Sign)	76
5.18	ผังวงจรระดับ RTL ของชุดดำเนินการบวกแบบคิดเครื่องหมาย (+, -)	77
5.19	ผลการจำลองการทำงานของตัวดำเนินการบวกแบบคิดเครื่องหมาย (+, -)	78
5.20	ผังวงจรระดับ RTL ของชุดดำเนินการในการตัดสินใจ (Hard Decision)	79
5.21	ผลการจำลองการทำงานของตัวดำเนินการในการตัดสินใจ (Hard Decision)	80
5.22	การใช้จำนวน Logic Elements	80
5.23	การประมาณค่าลอการิทึมด้วยเลขจำนวนจริง	82
5.24	การประมาณค่าลอการิทึมด้วยเลขตายตัว	82
5.25	สมรรถนะอัตราบิตผิดพลาดของการคำนวณของเลขตายตัวและเลขจำนวนจริง	83

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

รหัสพาริตี้เช็คความหนาแน่นต่ำได้รับการคิดค้นขึ้นในปี ค.ศ. 1960 โดย นาย R. G. Gallager [1] แห่งสถาบัน MIT ในประเทศสหรัฐอเมริกา และต่อมาในปี ค.ศ. 1962 บทความของเขาได้รับการตีพิมพ์ โดยใช้ชื่อว่า Low-density parity-check codes ในวารสาร IRE Transactions on Information Theory แต่ในขณะนั้นแทบจะไม่ได้รับความสนใจ เนื่องจากหลายๆ เหตุผลเช่น ปัญหาเรื่องฮาร์ดแวร์ ซึ่งในสมัยนั้นฮาร์ดแวร์ยังมีความสามารถที่ต่ำไม่เหมาะกับรหัสที่มีความซับซ้อน จึงทำให้รหัสพาริตี้เช็คความหนาแน่นต่ำเงียบหายไปนาน จนกระทั่งในปี ค.ศ.1996 D. J. C. MacKay และ R. M. Neal [2][3] ได้นำเสนอการค้นพบที่เกี่ยวข้องกับรหัสพาริตี้เช็คความหนาแน่นต่ำ (LDPC) โดยใช้วิธีการประมาณด้วยขั้นตอนของ Belief Propagation จึงทำให้รหัสพาริตี้เช็คความหนาแน่นต่ำได้รับการนำกลับมาใช้ใหม่ S. Y. Chung และคณะ [4] ได้แสดงให้เห็นถึงสมรรถนะของรหัสพาริตี้เช็คความหนาแน่นต่ำที่เข้าใกล้ขีดจำกัดของแชนนอน โดยห่างกัน 0.0045 dB เท่านั้น เนื่องจากมีประสิทธิภาพในการแก้ไขข้อผิดพลาดได้ดีเยี่ยม ทำให้นักวิจัยต่างๆ สนใจกันอย่างมาก โดยการวิจัยจะมุ่งเน้นไปที่การวิเคราะห์และปรับปรุงรหัสให้ดีขึ้น

รหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก (QC-LDPC) ได้รับการสร้างจากเมตริกซ์สลับที่สามารถให้ระยะห่างของ Girth เท่ากับ 12 และโครงสร้างในลักษณะนี้มีผลอีกอย่างคือ ทำให้ระยะห่างต่ำสุด (Minimum Hamming Distant) ซึ่งเป็นผลดีต่อการเข้ารหัส [6] การเข้ารหัสของรหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิกได้รับการนำเสนอการออกแบบการเข้ารหัสแบบ VLSI Implementation โดยพิจารณาถึงการลดความซับซ้อนในการเข้ารหัส จึงได้นำเสนอ Pivoting และ Bit-Reverse (PABR) อัลกอริทึม [7] LDPC ได้รับการออกแบบที่ Block length 648 ,1296 ,1944 Bit ที่อัตรารหัส 1/2 ,2/3 ,3/4 ,5/6 ตามมาตรฐาน IEEE 802.11n และได้มีการออกแบบโครงสร้างและการเข้ารหัส-ถอดรหัสเพื่อฮาร์ดแวร์ [8]

ในปัจจุบันเทคโนโลยีการสื่อสารได้เข้ามามีบทบาทในชีวิตประจำวันมากขึ้น เช่น ระบบโทรศัพท์เคลื่อนที่ (Mobile) ระบบสื่อสารไร้สาย (Wireless) การกระจายภาพด้วยระบบดิจิตอล (Digital Video Broadcast : DVB-S2) ระบบการสื่อสารดาวเทียม (Satellite Communication) ระบบการบันทึกข้อมูลด้วยแถบแม่เหล็ก (Hard-Disk Drive) เทคโนโลยีต่าง ๆ เหล่านี้ทำให้เกิดความสะดวกสบายมากขึ้น และช่วยให้การทำงานหรือติดต่อประสานงานสามารถทำได้อย่างรวดเร็ว วัตถุประสงค์ของระบบสื่อสารเชิงดิจิตอลคือ การรับ-ส่งข้อมูลข่าวสารจากแหล่งกำเนิดสัญญาณไป

ยังภาครับ โดยผ่านช่องสัญญาณสื่อสารที่ประกอบด้วยสัญญาณรบกวนที่มีผลกระทบทำให้ข้อมูลข่าวสารผิดพลาด ให้ปริมาณมาก รวดเร็ว และมีสมรรถนะในระดับที่ยอมรับได้ การสื่อสารในแต่ละระบบมักจะมีเกิดความผิดพลาดในการรับ-ส่งข้อมูลเนื่องมาจากสัญญาณรบกวน (Noise) ซึ่งเป็นอุปสรรคสำคัญที่เกิดขึ้น มีหลายสาเหตุ อาจเกิดขึ้นจากการกระทำของมนุษย์ หรือเกิดขึ้นเองตามธรรมชาติ และการแทรกสอดสัญลักษณ์ ความผิดพลาดด้านการซิงค์เวลาและการแทรกสอดระหว่างผู้ใช้ เป็นต้น ในส่วนของสัญญาณรบกวนภายในตัวอุปกรณ์เครื่องรับ (Internal Noise) แบ่งได้ 2 ประเภท

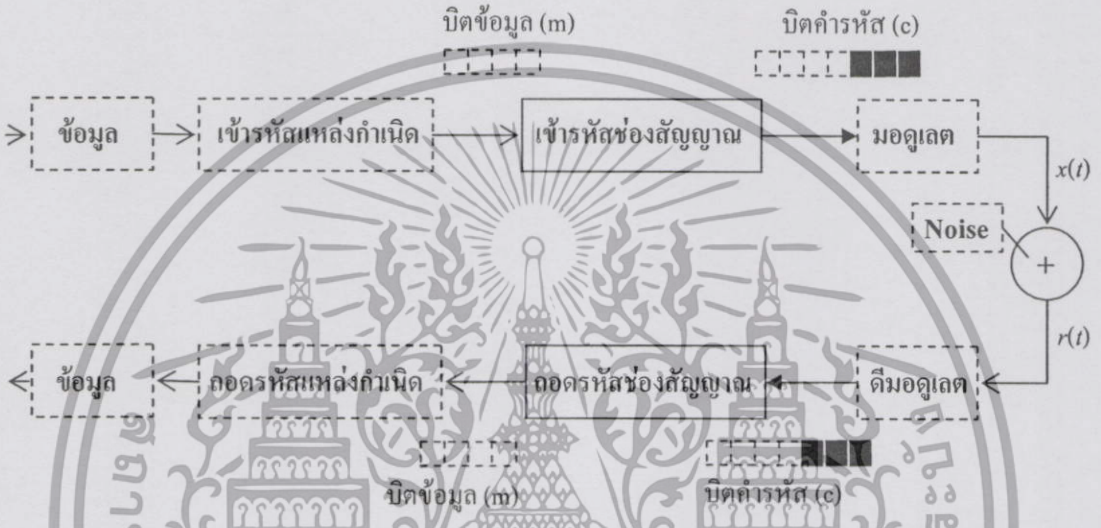
1. สัญญาณรบกวนจากอุณหภูมิ (Thermal Noise) เกิดจากการเคลื่อนที่ของอิเล็กตรอนภายในตัวอุปกรณ์
2. สัญญาณรบกวนที่เกิดจากการรวมตัวของอิเล็กตรอนกับโฮล (Hole)

ซึ่งหากสัญญาณที่รับเข้ามาในระบบเป็นสัญญาณที่ไม่ต้องการ จะจัดว่าสัญญาณเหล่านั้นเป็นสัญญาณรบกวน และเมื่ออุณหภูมิที่ตัวรับสัญญาณสูงจะทำให้กำลังของสัญญาณรบกวนจะสูงขึ้นทำให้ระดับเอสเอนอาร์ (Signal to Noise Ratio : SNR) ต่ำลงและมีอัตราบิตผิดพลาด (Bit Error Rate : BER) สูงขึ้น จากสาเหตุที่ได้กล่าวมาจะเห็นว่าในการรับ-ส่งข้อมูลข่าวสารจะเกิดความผิดพลาดขึ้นเนื่องจากสัญญาณรบกวนหลายรูปแบบซึ่งไม่สามารถคาดเดาได้แน่นอนที่อาจจะเข้ามาทำลายข้อมูลข่าวสาร การแก้ไขบิตผิดพลาดในการรับ-ส่งข้อมูลสามารถทำได้หลายวิธี เช่น การเพิ่มคุณภาพของสัญญาณหรือการเพิ่มกำลังเครื่องส่งเพื่อให้สัญญาณชัดเจนขึ้น (Signal to Noise Ratio : SNR) การเพิ่มกำลังเครื่องส่งสัญญาณจะนำมาซึ่งค่าใช้จ่ายที่เพิ่มมากขึ้นจึงเป็นวิธีที่ไม่เหมาะสม แต่มีวิธีที่นิยมใช้คือเทคนิคการเข้ารหัสช่องสัญญาณ (Channel Coding) การเข้ารหัสช่องสัญญาณนั้นใช้วิธีการเพิ่มบิตพิเศษหรือเรียกว่าบิตพาริตี้เข้ามาในบิตข้อมูลข่าวสาร โดยบิตพาริตี้ที่เพิ่มเข้ามาจะช่วยให้ภาครับสามารถตรวจจับบิตผิดพลาดได้ (Error Detection) หรือในกรณีที่บิตพาริตี้มีจำนวนมากพอสำหรับใช้แก้ไขบิตผิดพลาด ภาครับก็อาจจะแก้ไขบิตผิดพลาด (Error Correction) ของข้อมูลได้ การเข้ารหัสช่องสัญญาณแบ่งได้ 2 รูปแบบคือ

1. การตรวจสอบและขอให้ภาคส่ง ส่งข้อมูลมาใหม่ (Automatic Repeat Request : ARQ)
2. การตรวจสอบและแก้ไขบิตผิดพลาดด้วยตัวเอง (Forward Error Correction : FCC)

สำหรับวิธีแรก ARQ เมื่อภาครับได้รับบิตข้อมูลที่ได้เข้ารหัส จะทำการตรวจสอบผิดพลาด ในกรณีที่พบบิตข้อมูลผิดพลาด ภาครับจะส่งสัญญาณกลับไปที่ภาคส่งเพื่อให้ส่งข้อมูลชุดเดิมกลับมาใหม่ และในวิธีที่สอง FCC เมื่อภาครับได้รับบิตข้อมูลที่ได้เข้ารหัส จะทำการตรวจจับหาบิตผิดพลาด ในกรณีที่พบบิตผิดพลาดในชุดข้อมูลที่รับมา ภาครับจะทำการแก้ไขบิตผิดพลาดดังกล่าวให้ถูกต้องเอง การเข้ารหัสช่องสัญญาณจะช่วยเพิ่มสมรรถนะของระบบโดยรวม เมื่อวัดในรูปแบบของอัตราบิต

ผิดพลาด (Bit Error Rate : BER) ที่ต่ำลง ณ ระดับเอสเอนอาร์ (SNR) เท่าเดิม ซึ่งเทคนิคการเข้ารหัสช่องสัญญาณนี้ได้รับความนิยมอย่างแพร่หลาย โดยภาคส่งจะทำการเข้ารหัส (Encoder) ทำได้โดยเพิ่มบิตพาริตี้เข้าไปในชุดข้อมูล จะได้บิตข้อมูลชุดใหม่ขึ้นมา เรียกว่า คำรหัส (Codeword) จากนั้นคำรหัสจะถูกส่งออกไปในช่องสัญญาณที่เต็มไปด้วยสัญญาณรบกวน (Noise) เมื่อมาถึงภาครับจะทำการถอดรหัส (Decoder) โดยใช้บิตพาริตี้เป็นตัวตรวจเช็คหาบิตข้อมูลที่ผิดพลาด และถ้าบิตพาริตี้มีมากพอภาครับก็จะทำการแก้ไขบิตผิดพลาดนั้นให้ถูกต้องเอง



รูปที่ 1.1 แผนผังส่วนประกอบหลักในระบบสื่อสารเชิงดิจิตอล

ในรูปที่ 1.1 แสดงให้เห็นถึงวิธีการเพิ่มบิตพาริตี้เข้าไปในชุดข้อมูล หรือเรียกว่าวิธีการเข้ารหัสช่องสัญญาณ สำหรับรหัสช่องสัญญาณ (Channel Coding) เริ่มต้นตั้งแต่ยุคแรกๆ เป็นการเข้ารหัสบล็อกเชิงเส้น (Linear Block Code) ได้แก่รหัสไซคลิก (Cyclic Code) และรหัสรีดโซโลมอน (Reed Solomon code : RS) จากนั้นได้พัฒนาเป็นรหัสคอนโวลูชั่น (Convolution Code) เป็นการเข้ารหัสโดยอาศัยเทรลิสและในยุคแรก ๆ มีการถอดรหัสที่ให้เอาต์พุตแบบหยาบที่มีสมรรถนะในการรับ-ส่งปริมาณข้อมูล (Throughput) สูงขึ้น ต่อมามีการพัฒนาเป็นรหัสเทอร์โบ (Turbo Code) ซึ่งมีการถอดรหัสที่ให้เอาต์พุตแบบละเอียดและมีการประยุกต์ใช้เทคนิคการวนลูป ซึ่งแก้ไขบิตผิดพลาดได้สูงขึ้น และถูกนำไปใช้อย่างแพร่หลายในยุคที่สาม (3G) รหัสพาริตี้เช็คความหนาแน่นต่ำ (Low Low-Density Parity-Check : LDPC) จัดเป็นรหัสบล็อกเชิงเส้น (Linear Block Codes) ชนิดหนึ่ง คุณสมบัติที่น่าสนใจของรหัสพาริตี้เช็คความหนาแน่นต่ำก็คือ เป็นรหัสที่คล้ายกับรหัสเทอร์โบรุ่นใหม่ ซึ่งให้สมรรถนะแก้ไขบิตผิดพลาดเข้าใกล้ขอบเขตของแชนนอน (Shannon's Limit) จากทฤษฎีข่าวสารของแชนนอน (Shannon's Information Theory) [5] และได้รับการนำเสนอไปใช้ร่วมกับระบบสื่อสารต่าง ๆ เช่น ระบบฮาร์ดดิสก์ไดร์ ระบบสื่อสารไร้สาย ระบบสื่อสารดาวเทียม

และยังได้รับการนำเสนอให้นำมาใช้ในยุคที่สี่ (4G) [9] รหัสพาริตี้เช็คความหนาแน่นต่ำแบ่งออกเป็น 2 ประเภทคือ

1. รหัสแบบสม่ำเสมอ (Regular)
2. รหัสแบบไม่สม่ำเสมอ (Irregular)

ในการเข้ารหัสที่ขนาดบล็อกข้อมูล (Message) ขนาดใหญ่ การเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบเก่ามีปัญหาเรื่องการออกแบบวงจร เนื่องจากต้องการใช้ทรัพยากรสูง ในกรณีที่เพิ่มขนาดบล็อกบิตข้อมูลความซับซ้อนของวงจรเพิ่มขึ้นมากอย่างไม่เป็นเชิงเส้น และมีปัญหาเรื่องความเร็ว เนื่องจากวงจรที่ใช้อยู่ในลักษณะตรรกะ (Boolean) การทำงานของวงจรไม่สามารถเพิ่มหรือลดความเร็วได้ เนื่องจากไม่ได้ทำงานภายใต้สัญญาณความถี่ (clock) การถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบ่งได้ 2 รูปแบบคือ

1. รูปแบบของแลกเปลี่ยนข้อมูล (Message Passing Algorithm) การออกแบบวงจรถอดรหัสในลักษณะนี้จะมีความซับซ้อนมากกว่าแบบลอการิทึม
2. รูปแบบของลอการิทึม (Log-Domain Algorithm) การออกแบบวงจรถอดรหัสในลักษณะนี้จะมีความซับซ้อนน้อยกว่าแบบแลกเปลี่ยนข้อมูล และการหาค่า $\log((e^x + 1)/(e^x-1))$ ในส่วนถอดรหัสจะใช้วิธีประมาณ

1.2 วัตถุประสงค์และขอบเขตวิทยานิพนธ์

เนื่องจากการเข้ารหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำด้วยซอฟต์แวร์ทำงานได้ช้ากว่า การเข้ารหัสและถอดรหัสด้วยฮาร์ดแวร์ ดังนั้นจึงได้ออกแบบการเข้ารหัสและถอดรหัสเพื่อสังเคราะห์เป็นวงจรสำหรับทำงานบนอุปกรณ์ลอจิกแบบโปรแกรม (Field Programmable Gate Array : FPGA) แต่วงจรเข้ารหัสในแบบเก่าจะใช้ในรูปแบบ XOR Network การเข้ารหัสในลักษณะนี้จะสิ้นเปลืองทรัพยากรเป็นอย่างมาก โดยเฉพาะเมื่อบล็อกข้อมูล (Message) มีขนาดใหญ่ขึ้น และความซับซ้อนสูงขึ้น ความเร็วในการเข้ารหัสจะลดลง เนื่องจากวงจรอยู่ในลักษณะตรรกะ (Boolean) การทำงานของวงจรในลักษณะนี้จะไม่ทำงานภายสัญญาณความถี่ (Clock) เมื่อความถี่สูงขึ้นความเร็วของวงจรก็จะทำงานเท่าเดิมไม่เพิ่มความเร็วตามความถี่ ดังนั้นการหาวิธีลดความซับซ้อนในวงเข้ารหัสที่บิตอินพุตขนาดยาว การเพิ่มการรับส่งข้อมูล (Throughput) ให้สูงขึ้นและการใช้ทรัพยากรให้เหมาะสมจึงเป็นสิ่งที่ควรพิจารณา ในส่วนของการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ สิ่งที่ควรพิจารณาคือส่วนของการประมาณค่า $\log((e^x + 1)/(e^x-1))$ ให้เหมาะสมกับเข้ารหัสที่บิตอินพุตขนาดยาวที่อัตราเข้ารหัสที่สูง (High Rate)

1.3 ขอบเขตการวิจัย

ขอบเขตของวิทยานิพนธ์ฉบับนี้คือ การออกแบบวงจรเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำในลักษณะอนุกรม ลักษณะกึ่งขนาน และลักษณะขนาน โดยเน้นที่อัตรารหัสสูง (High Rate) เพื่อรองรับบล็อกข้อมูลที่มีขนาดเพิ่มขึ้น แต่ทรัพยากรที่ต้องการใช้และความซับซ้อนยังคงเพิ่มขึ้นอย่างเป็นเชิงเส้น การถอดรหัสพาริตี้เช็คความหนาแน่นต่ำจะอยู่ในรูปแบบของลอการิทึม (Log-Domain Algorithm) ขั้นตอนการหาค่า $L(r_{ji})$ หาได้จากการประมาณค่า $\log((e^x + 1)/(e^x-1))$ โดยค่าการประมาณนี้จะถูกเก็บในหน่วยความจำ (Ram) ซึ่งค่าการประมาณสามารถเปลี่ยนแปลงได้ในกรณีที่ต้องการลดการใช้ทรัพยากร หรือต้องความละเอียดในการประมาณมากกว่าเดิม และเปรียบเทียบการจำลองสมรรถนะการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำด้วยเลขจำนวนจริง (Floating Point) กับการถอดรหัสด้วยเลขตายตัว (Fixed Point) ในช่องสัญญาณรบกวนเกาส์แบบขาว (Additive White Gaussian Noise : AWGN)

1.4 ขั้นตอนของการศึกษา

วิทยานิพนธ์ฉบับนี้อธิบายถึงขั้นตอนและรายละเอียดต่าง ๆ ในการออกแบบวงจรเข้ารหัสและถอดรหัส โดยแบ่งเนื้อหาออกเป็น 6 บทดังนี้

บทที่ 1 อธิบายถึงความเป็นมา และปัญหาของงานวิจัย วัตถุประสงค์ และขอบเขตของงานวิจัยการศึกษา

บทที่ 2 กล่าวถึงทฤษฎีและหลักการ และองค์ประกอบของรหัสพาริตี้เช็คความหนาแน่นต่ำ การถอดรหัสในรูปแบบของแลกเปลี่ยนข้อมูล (Message Passing Algorithm) การถอดรหัสในรูปแบบลอการิทึม (Log-Domain Algorithm)

บทที่ 3 กล่าวถึงการออกแบบวงจรสำหรับชิปเอฟพีจีเอ (Field Programmable Gate Array : FPGA) และวิธีการออกแบบวงจรด้วยภาษาวีเอชดีแอล (VHSIC Hardware Description Language : VHDL)

บทที่ 4 อธิบายถึงการออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก (QC – LDPC) และเทคนิคที่ใช้ในการออกแบบวงจรเข้ารหัสในลักษณะต่าง ๆ และการออกแบบการถอดรหัส

บทที่ 5 อธิบายถึงการทดลองและผลการทดลอง

บทที่ 6 เป็นสรุปผลการวิจัยและข้อเสนอแนะ

บทที่ 2
ทฤษฎีและหลักการรหัสช่องสัญญาณ

เนื้อหาในบทนี้จะกล่าวถึงทฤษฎีพื้นฐานต่างๆ ที่ใช้ในการแก้ไขบิตผิดพลาดในระบบสื่อสารเชิงดิจิตอล เช่น การเข้ารหัสช่องสัญญาณในลักษณะรหัสบล็อกเชิงเส้น อธิบายถึงการเข้ารหัสด้วยเมตริกซ์กำเนิด **G** เทคนิคที่ใช้ในการออกแบบเมตริกซ์พาริตี้เช็ค **H** และการเข้ารหัสด้วยเมตริกซ์พาริตี้เช็ค **H**

2.1 หลักการรหัสช่องสัญญาณ

ระบบสื่อสารดิจิตอลประกอบด้วยภาคส่งและภาครับสัญญาณข่าวสาร ซึ่งการส่งสัญญาณข่าวสารอาจเกิดความผิดเพี้ยน หรือไม่ชัดเจนของรูปสัญญาณ เนื่องจากผลกระทบของสัญญาณรบกวนภายนอกในรูปแบบต่างๆ ที่มีคุณลักษณะแบบสุ่มไม่สามารถคาดเดาได้แน่นอน ดังนั้นเมื่อต้องการแก้ไขบิตหรือสัญลักษณ์ที่ผิดพลาดเพื่อให้อัตราบิตผิดพลาดในระบบต่ำลง เทคนิคการเข้ารหัสช่องสัญญาณ (Channel coding) เป็นวิธีหนึ่งที่นิยมนำไปใช้ โดยรหัสช่องสัญญาณแบ่งออกเป็น 2 ประเภทใหญ่ คือ

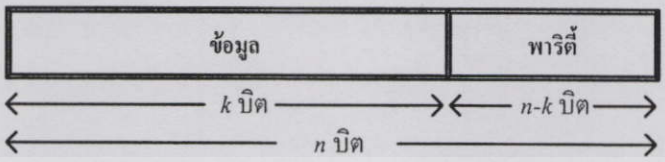
1. การเข้ารหัสแบบบล็อก (Block codes)
2. การเข้ารหัสคอนโวลูชัน (Convolution codes)

วิทยานิพนธ์ฉบับนี้จะมุ่งเน้นไปที่การแก้ไขความผิดพลาด (Error Correcting Code: ECC) ในลักษณะการเข้ารหัสแบบบล็อก โดยมีวิธีการเข้ารหัสข้อมูลข่าวสารคือการเพิ่มบิตพิเศษหรือบิตพาริตี้ (Parity bit) เข้าไปในบิตข้อมูลข่าวสาร บิตพาริตี้ที่เพิ่มเข้าไปนั้นมีไว้สำหรับตรวจจับบิตผิดพลาด และในกรณีที่บิตพาริตี้มีมาก ภาครับอาจทำการแก้ไขบิตผิดพลาดในชุดข้อมูลข่าวสารที่ส่งไปได้เอง สังเกตเห็นว่า การเข้ารหัสช่องสัญญาณนั้นจะเพิ่มบิตพาริตี้เข้าไปในข้อมูลข่าวสาร ทำให้ข้อมูลที่ส่งไปมีขนาดใหญ่กว่าเดิม และทำให้แบนด์วิดท์กว้างตามไปด้วย

2.1.1 การเข้ารหัสแบบบล็อก (Block Codes)

การเข้ารหัสแบบบล็อกจะแบ่งบิตข้อมูลข่าวสารที่จะทำการเข้ารหัสออกเป็นกลุ่มๆ หรือบล็อก เรียกอีกอย่างหนึ่งว่า รหัสบล็อกเชิงเส้น (Linear Block Codes) และจะมีบล็อกข้อมูลข่าวสารขนาด k บิต ในแต่ละบล็อก โดยในแต่ละบล็อกข้อมูลข่าวสารจะถูกแปลงให้เป็นคำรหัส (Codeword : **c**) ที่มีความยาวเท่ากับ n บิต โดยที่ $n > k$ โดยมีระยะห่างต่ำสุดของคำรหัสเท่ากับ d_{min} ให้ขนาดข้อมูล k บิต m = $[m_1\ m_2\ m_3\ m_4, \ldots, m_k]$ ตัวเข้ารหัสบล็อกเชิงเส้นทำการสร้างคำรหัสขนาด

n บิต $\mathbf{c} = [c_1\ c_2\ c_3\ c_4\ \ldots\ c_n]$ข้อมูลเรียกการเข้ารหัสนี้ว่า รหัส $\mathbf{C}(n, k, d_{min})$ ในกรณีของรหัสแบบมีโครงสร้าง (Systematic) คำรหัสที่ได้จากการเข้ารหัสนั้นจะประกอบด้วย 2 ส่วน ได้แก่ ส่วนของบิตข้อมูลเดิม k บิต และบิตเช็ค (Check Bit) ที่เพิ่มเข้าไป n-k บิต ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 โครงสร้างของรหัสบล็อกเชิงเส้น

ในการส่งข้อมูลที่ส่งผ่านช่องสัญญาณ พาริตี้มีไว้สำหรับตรวจสอบและแก้ไขข้อผิดพลาดที่เกิดขึ้นเมื่อถึงภาครับ

การเข้า-ถอดรหัสจะทำครั้งละบล็อก โดยขนาดของบล็อกข้อมูลจะขึ้นอยู่กับระบบสื่อสารที่ใช้ และสิ่งบ่งชี้ถึงซึ่งประสิทธิภาพของรหัส สามารถหาได้จากอัตราส่วนของจำนวนบิตข้อมูลข่าวสารต่อจำนวนบิตของคำรหัส หรือเรียกว่า อัตรารหัส (Code Rate: R) โดยที่ $0 < R \leq 1$ ในกรณีที่ R ยิ่งเข้าใกล้ 1 รหัสที่ได้จะมีประสิทธิภาพสูงขึ้น แต่ในกรณีที่ $R = 1$ แสดงว่าไม่มีการเข้ารหัส

$$R = k / n \tag{2.1}$$

ตารางที่ 2.1 เป็นตัวอย่างของการเข้ารหัสแบบบล็อก ที่มี $n = 7, k = 4$ และอัตรารหัส $R = 4 / 7$

ตารางที่ 2.1 การเข้ารหัสแบบบล็อก ที่ $k = 4$, $n = 7$ และ $R = 4 / 7$

ข้อมูล ($\mathbf{m} = m_1, m_2, m_3, m_4$)	คำรหัส ($\mathbf{c} = c_1, c_2, c_3, c_4, c_5, c_6, c_7$)
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 0)	(0 1 1 0 1 0 0)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

จากการเข้ารหัสในตารางที่ 2.1 สามารถพิจารณาถึงความสามารถในการตรวจจับ หรือแก้ไขบิตผิดพลาดได้ โดยเมื่อกำหนดให้คำรหัส $\mathbf{c} = [c_1\, c_2\, c_3\, c_4 \ldots c_n]$ สามารถนิยาม น้ำหนักแฮมมิ่ง (Hamming Weight) หรือ w_H ของคำรหัส $\mathbf{c}$ จาก

$$w_H(\mathbf{c}) = \text{จำนวนบิตในคำรหัส } \mathbf{c} \text{ ที่มีค่าเท่ากับ } 1$$

เช่น $w_H([1\ 1\ 0\ 1\ 0\ 0\ 0]) = 3$ เป็นต้น และนิยาม ระยะห่างแฮมมิง (Hamming Distance) ระหว่าง $\mathbf{C}_1$ และ $\mathbf{C}_2$ หรือ $d_H(\mathbf{C}_1, \mathbf{C}_2)$ แสดงได้ดังสมการที่ 2.2

$$\begin{aligned} d_H(\mathbf{C}_1, \mathbf{C}_2) &= \sum_{i=1}^{n} (c_{1,i} \neq c_{2,i}) \\ &= \sum_{i=1}^{n} ((c_{1,i} - c_{2,i}) \neq 0) \\ &= w_H(\mathbf{C}_1 - \mathbf{C}_2) \end{aligned} \tag{2.2}$$

ค่าระยะห่างแฮมมิงคือ จำนวนบิตที่แตกต่างกันของคำรหัส ซึ่งในตารางที่ 2.1 มีระยะห่างแฮมมิง $d_H[(0\ 0\ 1\ 0\ 1\ 1\ 1), (1\ 1\ 1\ 1\ 1\ 1\ 1)] = 3$ ให้รหัส $\mathbf{C}$ มีคำรหัสทั้งหมด 2^k คำรหัส และค่าระยะห่างแฮมมิ่งระหว่างคำรหัสที่ต่ำสุดเรียกว่า ระยะห่างต่ำสุดของรหัส (Minimum Distance) หรือ d_{min}

$$d_{min} = \min_{i \neq j} \{ d_H(\mathbf{C}_i, \mathbf{C}_j) \} \tag{2.3}$$

โดย $i, j = 1, 2, \ldots, 2^k$ ในกรณีที่มี $d_{min} = 3$ จำนวนบิตที่รหัสสามารถแก้ไขความผิดพลาด t หาได้ดังนี้

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor \tag{2.4}$$

และจำนวนบิตที่รหัสสามารถตรวจจับความผิดพลาด e หาได้จาก

$$e = d_{min} - 1 \tag{2.5}$$

ตัวอย่างที่ 2.1 การหาจำนวนบิตที่ผิดพลาดสามารถหาได้จาก d_{min} คือ จำนวนบิตที่สามารถแก้ไขได้

$$t = \frac{[d_{\min} - 1]}{2}$$

$$t = \left\lfloor \frac{3-1}{2} \right\rfloor$$

$$t = 1$$

จำนวนบิตที่ผิดพลาดและสามารถตรวจจับได้

$$e = d_{min} - 1$$

$$e = 3 - 1$$

$$e = 2$$

ดังนั้น จะเห็นได้ว่า ที่ $d_{min} = 3$ สามารถแก้ไขบิตผิดพลาดได้ 1 บิต และตรวจจับบิตที่ผิดพลาดได้ 2 บิต

รหัสที่ได้กล่าวมาข้างต้นสามารถตรวจจับบิตผิดพลาดได้ $e = 2$ บิต และแก้ไขบิตผิดพลาดได้ $t = 1$ บิต สังเกตได้ว่าจำนวนอินพุตเท่ากับ 2^k และจำนวนคำรหัสเท่ากับ 2^k เช่นกัน ในการใช้งานจริงขนาดของอินพุต k มีค่าสูง ให้คำรหัสจำนวนมาก กรณีเข้ารหัสอาจไม่เหมาะสม เช่น $k = 2048$ ให้จำนวนคำรหัสเท่ากับ 2^{2048} คำรหัส

2.1.2 เมตริกซ์กำเนิด (Generator Matrix)

การเข้ารหัสสามารถทำได้หลายวิธี หากจำนวนคำรหัสมีจำนวนน้อยการใช้ตารางดังตารางที่ 2.1 สามารถทำได้แต่ในกรณีที่จำนวนคำรหัสมีจำนวนมาก การเข้ารหัสจึงต้องอาศัยเทคนิคอื่น เช่น การใช้เมตริกซ์กำเนิด โดยเมตริกซ์กำเนิด **G** มีมิติขนาด kxn และการเข้ารหัสบล็อกเชิงเส้น (n, k, d_{min}) สามารถทำได้โดยการนำข้อมูลข่าวสาร (**m**) คูณกับเมตริกซ์กำเนิด **G** โดยข้อมูลที่ได้รับการเข้ารหัสแล้วจะได้เป็นคำรหัส (**c**) ดังแสดงในสมการที่ 2.9

เมตริกซ์กำเนิด **G** คือ

$$\mathbf{G} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,(n-k)} & 1 & 0 & \cdots & 0 \\ p_{21} & p_{22} & \cdots & p_{2,(n-k)} & 0 & 1 & \cdots & 0 \\ \cdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & 0 \\ p_{k,1} & p_{k,2} & \cdots & p_{k,(n-k)} & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{2.6}$$

$$\mathbf{G} = [\mathbf{P} : \mathbf{I}_k] \tag{2.7}$$

ข้อมูลข่าวสาร $\mathbf{m} = [m_1\ m_2\ m_3\ \ldots\ m_k]$ มีมิติขนาด $1 \times k$ หาได้จาก

$$\mathbf{m} = [m_1\ m_2\ m_3\ \ldots\ m_k] \tag{2.8}$$

คำรหัส $\mathbf{C} = [c_1\ c_2\ c_3\ c_4\ \ldots\ c_n]$ มีมิติขนาด $1 \times n$ หาได้จาก

$$\mathbf{c} = [c_1\ c_2\ c_3\ \ldots\ c_n] \tag{2.9}$$

การเข้ารหัสคือนำข้อมูล **m** คูณด้วยเมตริกซ์กำเนิด **G** จะได้ $\mathbf{c} = \mathbf{mG}$ โดยคำรหัส **c** ที่ได้นั้นมีมิติเท่ากับ $1 \times n$ ดังแสดงในสมการที่ 2.10

$$\mathbf{c} = \mathbf{mG} \tag{2.10}$$

โดยคำรหัสหาได้จาก

$$\mathbf{c} = [p_1\ p_2\ p_3\ \ldots\ p_{n-k}\ |\ m_1\ m_2\ m_3\ \ldots\ m_k] \tag{2.11}$$

ซึ่งการเข้ารหัสในลักษณะสมการที่ 2.11 จะมีข้อมูลเดิมอยู่ด้วย นั่นคือ $m_1\ m_2\ m_3\ \ldots\ m_k$ โดยข้อดีสำหรับการเข้ารหัสในลักษณะนี้คือ เมื่อทำการถอดรหัสสามารถที่นำข้อมูลเดิมกลับมาได้โดยง่าย รหัสประเภทนี้จัดเป็นรหัสเชิงระบบ (Systematic Code) โดยรหัสเชิงระบบคือการเข้ารหัสและมีข้อมูล **(m)** อยู่ในคำรหัส **(c)** รหัสในตารางที่ 2.1 ข้างต้น มีเมตริกซ์กำเนิด ได้แก่

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

โดยคำรหัส (c) หาได้จากการนำข้อมูลข่าวสาร m_1, m_2, m_3, m_4 มาคูณด้วยเมตริกซ์กำเนิด **G**

$$\begin{aligned}\mathbf{C} &= \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} m_1 + m_3 & m_1 + m_2 + m_4 & m_2 + m_3 & m_1 & m_2 & m_3 & m_4 \end{bmatrix}\end{aligned} \tag{2.12}$$

โดยการบวกข้างต้นเป็นการบวกแบบมอดูโล-2 หรือการ XOR

2.1.3 เมตริกซ์พาริตี้เช็ค (Parity-check matrix)

เมตริกซ์พาริตี้เช็ค **H** มีมิติขนาด $(n\text{-}k)xn$ สามารถสร้างได้จากเมตริกซ์กำเนิด **G** จากความสัมพันธ์ของเมตริกซ์กำเนิดและเมตริกซ์พาริตี้เช็คคือ ในแต่ละแถวของเมตริกซ์กำเนิด **G** จะตั้งฉากกับเมตริกซ์พาริตี้เช็ค **H** ในแต่ละหลัก กล่าวคือ $\mathbf{GH}^T = \mathbf{0}_{kx(n-k)}$ ดังนั้นการหาเมตริกซ์พาริตี้เช็ค **H** โดยใช้เมตริกซ์กำเนิด **G** ทำได้ดังนี้คือ

เมื่อกำหนดให้

$$\mathbf{H} = [\mathbf{I}_{n-k} : \mathbf{P}^T] \tag{2.13}$$

เมื่อสมการที่ 2.13 ได้รับการทรานสโพสและคูณด้วยเมตริกซ์กำเนิด **G** และบวกแบบมอดูโล-2 จะได้ดังนี้คือ

$$\mathbf{GH}^T = [\mathbf{P} : \mathbf{I}_k][\mathbf{I}_{n-k} : \mathbf{P}^T]^T = \mathbf{P} + \mathbf{P} = \mathbf{0} \tag{2.14}$$

การคำนวณหาระยะห่างต่ำสุด d_{min} ของรหัสสามารถทำได้ 2 วิธี ได้แก่

1. น้ำหนักต่ำสุดของแถวของเมตริกซ์กำเนิด **G**
2. จำนวนหลักจำนวนน้อยที่สุดที่บวกกันได้ศูนย์ของเมตริกซ์ **H**

2.1.4 รหัสแฮมมิง (Hamming Code)

รหัสแฮมมิงเป็นรหัสแก้ไขบิตผิดพลาดชนิดหนึ่งที่ทีพารามิเตอร์ดังนี้

$$(n, k) = (2^m - 1, 2^m - 1 - m) \tag{2.15}$$

ซึ่งค่า $m \geq 2$ ระยะห่างต่ำสุดของรหัสแฮมมิงเท่ากับ 3 สามารถแก้ไขบิตผิดพลาดได้ 1 บิต

2.1.5 การถอดรหัสบล็อกเชิงเส้น

ให้ลำดับสัญญาณข่าวสารที่ได้รับ $\mathbf{r} = \mathbf{C} + \mathbf{e}$ กำหนดให้ $\mathbf{e} = [e_1\ e_2\ e_3\ \dots\ e_n]$ คือเวกเตอร์ความผิดพลาดโดย $e_i = 1$ ทำให้บิต i ของคำรหัสเป็นบิตที่ผิดพลาด

2.1.6 การถอดรหัสด้วยซินโดรม (Syndrome Decoding)

การถอดรหัสด้วยซินโดรมจัดเป็นการถอดรหัสแบบหยาบประเภทหนึ่ง ซึ่งจะตัดสินใจให้สัญญาณที่ได้รับอยู่ในรูปบิต และ ก่อนการถอดรหัส ซินโดรม $\mathbf{S}$ ลำดับสัญญาณที่ได้รับ $\mathbf{r}$ หาได้จาก

$$\mathbf{S} = \mathbf{r}\mathbf{H}^T = (\mathbf{c} + \mathbf{e})\mathbf{H}^T = \mathbf{C}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T \tag{2.16}$$

ดังนั้นเวกเตอร์หรือลำดับความผิดพลาดที่แตกต่างกันจะให้ซินโดมที่แตกต่างกัน

ขั้นตอนการถอดรหัสด้วยซินโดรม

1. สร้างตารางแพทเทิร์นของบิตผิดพลาดตามความสามารถในการแก้ไขบิตผิดพลาดของรหัส และซินโดรมที่เกี่ยวข้อง
2. คำนวณซินโดรมสำหรับลำดับสัญญาณที่ได้รับ $\mathbf{r}$ โดยคำนวณ $\mathbf{S} = \mathbf{r}\mathbf{H}^T$
3. ค้นหาแพทเทิร์นของบิตผิดพลาดที่มีซินโดรมจากข้อ 2 ให้เป็น $\mathbf{e}$
4. ทำการแก้ไขบิตผิดพลาด โดยการบวกสัญญาณที่ได้รับ $\mathbf{r}$ กับ $\mathbf{e}$ ให้ได้คำรหัสที่ทำการตรวจสอบ $\hat{\mathbf{c}} = \mathbf{r} + \mathbf{e}$

2.2 เอสเอนอาร์ของสัญลักษณ์รหัส

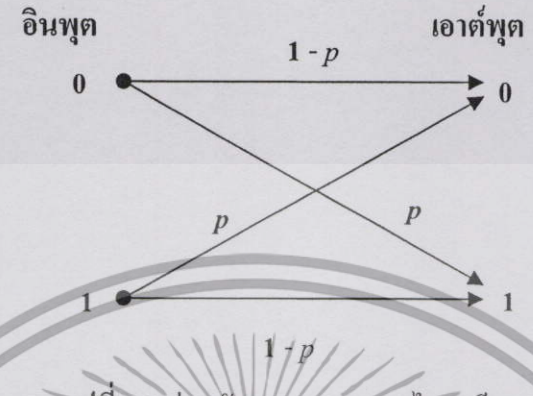
ให้ E_c เป็นพลังงานเฉลี่ยของบิตรหัส E_b เป็นพลังงานเฉลี่ยของบิตข้อมูล เอสเอนอาร์ (SNR) ของรหัสเขียนได้ดังในสมการที่ 2.17

$$\frac{E_c}{N_0} = R\frac{E_b}{N_0} \tag{2.17}$$

ดังนั้นเอสเอนอาร์สำหรับบิตข้อมูลคือ

$$\frac{E_b}{N_0} = \frac{1}{R}\frac{E_c}{N_0} \tag{2.18}$$

โดย R คืออัตรารหัส พิจารณาช่องสัญญาณสมมาตรไบนารี (Binary Symmetric channel) ที่มีความน่าจะเป็นของบิตผิดพลาด $p = P(0|1) = P(1|0)$ และความน่าจะเป็นของบิตถูกต้อง $P(0|0)$ $P(1|1) = 1 - p$ ดังแสดงในรูปที่ 2.2



รูปที่ 2.2 ช่องสัญญาณสมมาตรไบนารี

ในระบบบีพีเอสเค (BPSK) อัตราบิตผิดพลาดจากช่องสัญญาณ p คือ

$$p = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{2.19}$$

และเมื่อได้รับการเข้ารหัส อัตราบิตผิดพลาดจากช่องสัญญาณคือ

$$p = Q\left(\sqrt{\frac{2E_c}{N_0}}\right) \tag{2.20}$$

อัตราบิตผิดพลาดเฉลี่ยหลังจากที่ได้รับการถอดรหัสต้องคำนวณในระดับบล็อก กล่าวคือจำนวนบิตผิดพลาดน้อยกว่าหรือเท่ากับ t จะถูกแก้ไขหลังจากการถอดรหัส ความน่าจะเป็นของบิตผิดพลาดเท่ากับ j จาก n บิต $P(n, j)$ คือ

$$P(n, j) = \binom{n}{j} p^j (1-p)^{n-j} \tag{2.21}$$

ดังนั้นอัตราบิตผิดพลาดเฉลี่ย P_B หาได้จากความน่าจะเป็นของจำนวนบิตผิดพลาดมากกว่า t

$$P_B \approx \frac{1}{n} \sum_{j=t+1}^{n} \binom{n}{j} p^j (1-p)^{n-j} \tag{2.22}$$

สำหรับรหัสแฮมมิงจำนวนบิตที่แก้ไขได้เท่ากับ 1 บิตดังนั้นความน่าจะเป็นเฉลี่ยของบิตผิดพลาดเท่ากับ

$$P_B \approx \frac{1}{n}\sum_{j=2+1}^{n}\binom{n}{j}p^j(1-p)^{n-j} \tag{2.23}$$

และทำการลดรูป

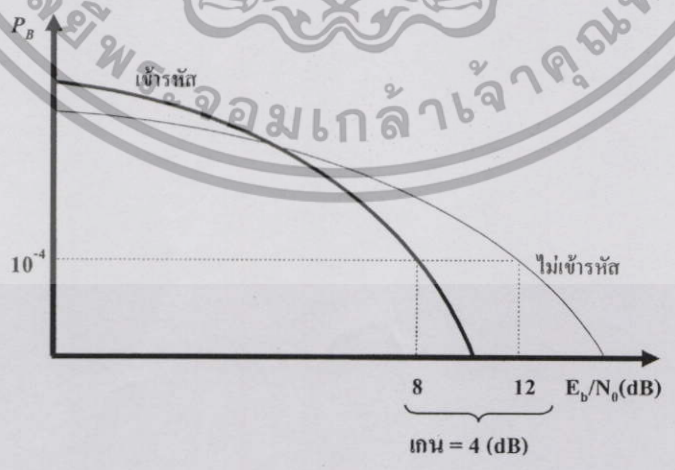
$$P_B \approx p - p(1-p)^{n-1} \tag{2.24}$$

2.3 เกนของรหัส (Coding Gain)

เกนของรหัส G หาได้จาก ผลต่างของเอสเอนอาร์ของกรณีที่ไม่มีการเข้ารหัส และกรณีที่มีการเข้ารหัส ณ ระดับความน่าจะเป็นบิตผิดพลาดของระบบที่สนใจ

$$G = \left(\frac{E_b}{N_0}\right)_u (dB) - \left(\frac{E_b}{N_0}\right)_c (dB) \tag{2.25}$$

โดย $\left(\frac{E_b}{N_0}\right)_u$ คือเอสเอนอาร์ในกรณีที่ไม่มีการเข้ารหัส และ $\left(\frac{E_b}{N_0}\right)_c$ คือเอสเอนอาร์ในกรณีที่มีการเข้ารหัส



รูปที่ 2.3 กราฟแสดงความน่าจะเป็นของบิตผิดพลาดในกรณีที่มีการเข้ารหัสช่องสัญญาณ

2.4. หลักการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำ

ประเภทของรหัสพาริตี้เช็คความหนาแน่นต่ำแบ่งออกได้เป็น 2 ประเภทคือ รหัสแบบสม่ำเสมอ (Regular) และรหัสแบบไม่สม่ำเสมอ (Irregular) การเข้ารหัสสามารถทำได้โดยการนำข้อมูล (Message) มาต่อกับพาริตี้ (**p**) และการหาค่าพาริตี้นั้นทำได้หลายวิธี ซึ่งวิธีที่นำเสนอนี้เป็นการหาจากความสัมพันธ์ของคำรหัส (Codeword: **c**) กับเมตริกซ์พาริตี้เช็ค **H** การเข้ารหัสสามารถทำได้หลายวิธีดังที่ได้กล่าวมาแล้ว การเข้ารหัสเริ่มแรกเป็นการเข้ารหัสด้วยเมตริกซ์กำเนิด **G** จากนั้นได้พัฒนามาเป็นการเข้ารหัสด้วยเมตริกซ์พาริตี้เช็ค **H** โดยตรง

เมื่อกำหนดให้

H	คือ	เมตริกซ์พาริตี้เช็ค
P	คือ	เพอร์มิวเทชั่นเมตริกซ์ (Permutation matrix)
I	คือ	เมตริกซ์เอกลักษณ์ (Identity matrix)
0	คือ	เมตริกซ์ 0
m	คือ	บิตข้อมูลข่าวสาร
c	คือ	บิตคำรหัส
p	คือ	บิตพิเศษหรือบิตพาริตี้เช็ค
L	คือ	ขนาดของเมตริกซ์ ***P***
j	คือ	การขยายเมตริกซ์ ***P*** ออกไปทางด้านหลัก
k	คือ	การขยายเมตริกซ์ ***P*** ออกไปทางด้านแถว

2.4.1 การเข้ารหัสด้วยเมตริกซ์กำเนิด G

รหัสพาริตี้เช็คความหนาแน่นต่ำเป็นบล็อครหัสเชิงเส้นที่มีจำนวนเลข 1 น้อยมากเมื่อเทียบกับขนาดของเมตริกซ์พาริตี้เช็ค **H** ในกรณีที่ต้องการเข้ารหัสหรือสร้างคำรหัสสามารถหาได้โดยอาศัยการคูณเมตริกซ์เมตริกซ์กำเนิด **G** กับเวกเตอร์บิตข้อมูล (**m**) ดังแสดงในสมการที่

$$\mathbf{c} = \mathbf{mG} \tag{2.26}$$

โดยที่ขนาดของเวกเตอร์ข้อมูล **m** เมตริกซ์กำเนิด **G** และเวกเตอร์คำรหัส **c** มีขนาดเท่ากับ 1xk, kxn และ 1xn ตามลำดับ การบวกทั้งหมดจะอยู่ในแบบของมอดูโล 2 นั่นคือ 0+0=0, 0+1=1+0=1, 1+1=0 แต่ในส่วนของการถอดรหัสจะใช้เมตริกซ์พาริตี้เช็ค **H** ในการถอดรหัส เมตริกซ์กำเนิด **G** และเมตริกซ์พาริตี้เช็ค **H** จึงมีความสัมพันธ์กันและมีขนาดเท่ากับ (n-k)xn

$$\mathbf{G}\mathbf{H}^T = \mathbf{0} \tag{2.27}$$

แต่การสร้างรหัสแอลดีพีซีเริ่มต้นโดยการสร้างเมตริกซ์พาริตี้เช็ค **H** ที่มีจำนวนเลข 1 น้อยมาก เพื่อให้ค่าระยะห่างต่ำสุดของรหัส (D_{min}) มีค่าสูงที่สุดเท่าที่จะเป็นไปได้ แต่เมตริกซ์พาริตี้เช็ค **H** ที่สร้างขึ้นอาจไม่อยู่ในรูปของ Systematic กล่าวคือมีเมตริกซ์เอกลักษณ์เป็นส่วนหนึ่งของเมตริกซ์พาริตี้เช็ค **H** ดังนั้นการสร้างเมตริกซ์กำเนิด **G** เพื่อใช้ในการสร้างคำรหัสจึงต้องอาศัยเทคนิคของ Gaussian elimination และถึงแม้ทำได้ก็จะทำให้โครงสร้างของรหัสเปลี่ยนไป เนื่องจากการถอดรหัสนั้นต้องอิงกับโครงสร้างของเมตริกซ์พาริตี้เช็ค **H** ด้วยเหตุผลดังกล่าว การเข้ารหัสด้วยเมตริกซ์พาริตี้เช็ค **H** โดยตรงจึงเป็นทางเลือกที่นิยมใช้

2.4.2 การเข้ารหัสด้วยเมตริกซ์พาริตี้เช็ค H

การเข้ารหัสด้วยเมตริกซ์พาริตี้เช็ค **H** โดยตรงเป็นวิธีที่ได้รับความนิยม เนื่องมาจากไม่มีปัญหาเรื่องโครงสร้างในการเปลี่ยนจากเมตริกซ์พาริตี้เช็ค **H** ไปเป็นเมตริกซ์กำเนิด **G** การเข้ารหัสด้วยเมตริกซ์พาริตี้เช็ค จะใช้ความสัมพันธ์ของเมตริกซ์พาริตี้เช็ค **H** และเวกเตอร์คำรหัส **c** เพื่อหาบิตพิเศษ แสดงได้ดังสมการที่ 2.27

$$\mathbf{c}\mathbf{H}^T = \mathbf{0} \tag{2.28}$$

โดยเมตริกซ์ศูนย์ **0** นี้มีขนาดเท่ากับ $1x(n\text{-}k)$

เมื่อเริ่มต้น กำหนดให้ข้อมูลเป็นส่วนหนึ่งของคำรหัส เพื่อความสะดวกในภาครับ รูปแบบหนึ่งของเวกเตอร์คำรหัส **c** คือ $\mathbf{c} = [c_1\ c_2\ c_3\ \ldots\ c_n] = [\ m_1\ m_2\ m_3\ \ldots\ m_k \mid p_1\ p_2\ p_3\ \ldots\ p_{n-k}\ ] = [\mathbf{m} \mid \mathbf{p}]$ เขียนเมตริกซ์พาริตี้เช็ค **H** ในรูป

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \mid \mathbf{H}_2 \end{bmatrix}$$
$$\mathbf{H}^T = \begin{bmatrix} \mathbf{H}_1^T \\ \mathbf{H}_2^T \end{bmatrix} \tag{2.29}$$

โดย $\mathbf{H}_1$ และ $\mathbf{H}_2$ มีขนาดเท่ากับ $(n\text{-}k)xk$ และ $(n\text{-}k)x(n\text{-}k)$ ตามลำดับ ดังนั้นเมื่อจัดรูปสมการแล้วสามารถแสดงได้ดังสมการที่ 2.29 ซึ่งเป็นการบวกแบบมอดูโล-2

$$\begin{aligned} \mathbf{c}\mathbf{H}^T &= \begin{bmatrix} \mathbf{m} \mid \mathbf{p} \end{bmatrix} \begin{bmatrix} \mathbf{H}_1^T \\ \mathbf{H}_2^T \end{bmatrix} = \mathbf{0} \\ &= \mathbf{m}\mathbf{H}_1^T + \mathbf{p}\mathbf{H}_2^T = \mathbf{0} \end{aligned} \tag{2.30}$$

เมตริกซ์บิตพาริตี้เช็คทั้งหมดหาได้จาก

$$\mathbf{p} = \mathbf{m}\mathbf{H}_1^T \left(\mathbf{H}_2^T\right)^{-1} \tag{2.31}$$

สังเกตว่าเมตริกซ์ $\mathbf{H}_2$ เป็นเมตริกซ์จัตุรัสขนาดเท่ากับ $(n\text{-}k)$x$(n\text{-}k)$ ทำให้สมารถหาส่วนกลับของเมตริกซ์และทำการเข้ารหัสได้ แต่สำหรับการออกแบบฮาร์ดแวร์การหาส่วนกลับของเมตริกซ์นั้นเป็นเรื่องที่ได้ได้ยากเนื่องจากปัญหาความซับซ้อนวิธีนี้จึงไม่เหมาะสม

ตัวอย่างที่ 2.2 กำหนดให้เมตริกซ์พาริตี้เช็ค $\mathbf{H}$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

กำหนดให้บิตข้อมูลข่าวสาร $\mathbf{m} = [1\ 0\ 1\ 1]$

เนื่องด้วยค่า $k = 4$ และ $n = 7$ เขียนเมตริกซ์พาริตี้เช็ค $\mathbf{H}$ ใหม่ในรูปของ

$$\mathbf{H} = [\mathbf{H}_1 \mid \mathbf{H}_2] = \left[\begin{array}{ccc:ccc:c} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right]$$

$$\mathbf{p} = \mathbf{m}\mathbf{H}_1^T \left(\mathbf{H}_2^T\right)^{-1} = [1\ \ 0\ \ 0]$$

ดังนั้นเมื่อทำการหาค่าเวกเตอร์คำรหัสจะได้เท่ากับ

$$\mathbf{c} = [\mathbf{m} \mid \mathbf{p}] = [1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0]$$

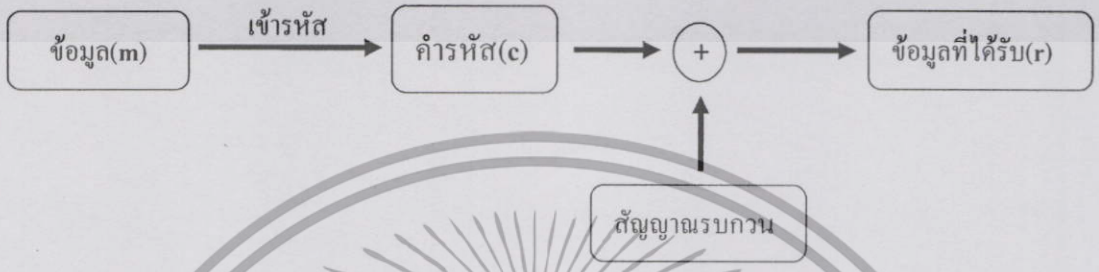
ทำการตรวจสอบว่าเวกเตอร์คำรหัส $\mathbf{c}$ เป็นคำรหัสที่ถูกต้องหรือไม่ ทำได้โดยการนำเวกเตอร์คำรหัสไปคูณกับเมตริกซ์พาริตี้เช็ค $\mathbf{H}$ จะได้

$$\mathbf{c}\mathbf{H}^T = [1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \mathbf{0}$$

กรณีที่คำตอบได้เท่ากับศูนย์ แสดงว่าเวกเตอร์คำรหัส $\mathbf{c}$ เป็นคำรหัสที่ถูกต้อง

2.5 หลักการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ

การถอดรหัสเป็นวิธีที่กู้บิตข้อมูลเดิมกลับมาใช้งาน ซึ่งเมื่อเริ่มแรกบิตข้อมูลจะได้รับการเข้ารหัส (Encoder) จากนั้นถูกส่งผ่านช่องสัญญาณรบกวน (Noise) และมาถึงภาครับข้อมูลอาจได้รับความเสียหาย ภาครับจะทำการการถอดรหัสเพื่อนำบิตข้อมูลที่ถูกต้องกลับมาใช้งาน



รูปที่ 2.4 การรับส่งข้อมูลในระบบสื่อสารดิจิตอล

ซึ่งอัลกอลิทึมในการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำนั้นมีชื่อเรียกหลากหลาย sum-product algorithm (SPA), belief propagation algorithm (BPA) และ message passing algorithm (MPA) ซึ่งเป็นขั้นตอนในการตัดสินอย่างละเอียด (Soft Decision) และมีการวนซ้ำเพื่อให้อัตราบิตผิดพลาด (Bit Error Rate: BER) ลดลงและทำให้สมรรถนะของระบบดีขึ้นเหมือนกับรหัสเทอร์โบรุ่นใหม่ แต่สิ่งที่รหัสพาริตี้เช็คความหนาแน่นต่ำแตกต่างและดีกว่ารหัสเทอร์โบนั้นคือ ในขั้นตอนการถอดรหัสสามารถทำงานในลักษณะขนานได้ เพราะในกรณีที่เพิ่มจำนวนการวนซ้ำในจำนวนที่เพิ่มขึ้นเพื่อลดอัตราบิตผิดพลาด รหัสพาริตี้เช็คความหนาแน่นต่ำยังคงให้ความซับซ้อนในการถอดรหัสอยู่ในลักษณะเชิงเส้น และการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำสามารถแสดงได้ 2 ลักษณะคือ อธิบายด้วยเมตริกซ์พาริตี้เช็ค **H** และอธิบายด้วยสัญลักษณ์ในลักษณะภาพที่เรียกว่าแทนเนอร์กราฟ (Tanner Graph)

2.5.1 แทนเนอร์กราฟ (Tanner Graph)

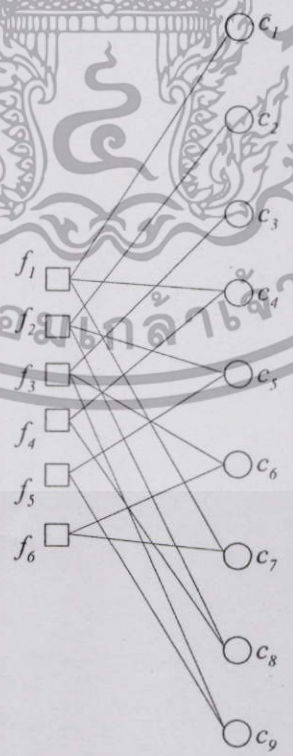
แทนเนอร์กราฟได้รับการคิดค้นโดย นาย R. M. Tanner ในปีค.ศ. 1981 การค้นพบนี้มีส่วนสำคัญต่อรหัสพาริตี้เช็คความหนาแน่นต่ำอย่างยิ่ง เนื่องจากแทนเนอร์กราฟนั้นได้แสดงให้เห็นถึงวิธีการและขั้นตอนการถอดรหัสได้อย่างชัดเจน และวิธีการสร้างแทนเนอร์กราฟนั้น สามารถหาได้จากจำนวนเลข 1 ในเมตริกซ์พาริตี้เช็ค **H** ซึ่งเป็นส่วนที่แสดงถึงการเชื่อมต่อในการแลกเปลี่ยนข้อมูลในการถอดรหัส

$$H = \left[\begin{array}{ccc:ccc:ccc} 1 & & & 1 & & & 1 & & \\ & 1 & & & 1 & & & 1 & \\ & & 1 & & & 1 & & & 1 \\ \hdashline & & & 1 & & & 1 & & \\ & & & & 1 & & & 1 & \\ \hdashline & & & & & 1 & 1 & & \end{array}\right] \tag{2.32}$$

จากสมการเมตริกซ์พาริตี้เช็ค **H** สามารถนำมาใช้อ้างอิงในการสร้างแทนเนอร์กราฟเพื่อใช้อธิบายวิธีการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ โดยแทนเนอร์กราฟนี้จะประกอบไปด้วย 3 ส่วน คือ

1. บิตโนด (Bit Nodes) มีจำนวนเท่ากับจำนวนหลักของเมตริกซ์พาริตี้เช็ค **H**
2. เช็คโนด (Check Nodes) มีจำนวนเท่ากับจำนวนแถวของเมตริกซ์พาริตี้เช็ค **H**
3. เส้นการเชื่อมต่อระหว่างบิตโนดและเช็คโนด มีจำนวนเท่ากับจำนวนเลข 1 ทั้งหมดในเมตริกซ์พาริตี้เช็ค **H**

จากสมการที่ 2.31 สามารถเขียนเมตริกซ์พาริตี้เช็ค **H** ออกมาในรูปของแทนเนอร์ (Tanner Graph) ซึ่งจะมีจำนวนบิตโนดเท่ากับ 9 และ จำนวนเช็คโนดเท่ากับ 6 และจำนวนเส้นในการเชื่อมต่อระหว่างบิตโนดและเช็คโนดเท่ากับ 15 ดังแสดงในรูปที่ 2.5



รูปที่ 2.5 แทนเนอร์กราฟ (Tanner Graph)

2.5.2 วิธีคำนวณการถอดรหัสโดยใช้แทนเนอร์กราฟ (Tanner Graph)

การคำนวณการถอดรหัสในแต่ละบิตข้อมูล สามารถทำได้หลายวิธีตามแต่โครงสร้างของอัลกอริทึม และวิธีที่น่าสนใจ คือ แทนเนอร์กราฟ ในรูปแบบของ Log Domain SPA Algorithm โดยวิธีการถอดรหัสนั้นจะอาศัยการแลกเปลี่ยนข้อมูลระหว่างบิตโนดและเช็คโนด และใช้เทคนิคถอดรหัสแบบวนซ้ำ (soft iterative decoding) เข้าร่วมด้วย เพื่อทำให้อัตราบิตผิดพลาดต่ำลง แต่ใช้พลังงานในการส่งข้อมูลเท่าเดิม จากนั้นมาพิจารณาถึงการแลกเปลี่ยนข้อมูลข่าวสารระหว่างเช็คโนด j และบิตโนด i เพื่อให้เข้าใจมากขึ้นดังแสดงในรูปที่ 2.6

ข้อมูลข่าวสารที่ได้รับหรืออินพุตของการถอดรหัส จะจัดอยู่ในรูปแบบของอัตราส่วนความน่าจะเป็นแบบล็อก (Log Likelihood: LLRs) ของตัวแปลสุ่ม c_i โดยในแต่ละบิตโนด i จะส่งค่าความน่าจะเป็นของข่าวสารแบบละเอียด (Soft Decision) ไปที่เช็คโนด j ผ่านเส้นทางการเชื่อมต่อที่เชื่อมถึงกัน จากนั้นเช็คโนด j ก็จะทำการคำนวณค่าความน่าจะเป็นของข่าวสารที่ส่งจากบิตโนด i แล้วจึงส่งผลที่ได้ของข่าวสารแบบละเอียดไปให้บิตโนด i อีกครั้ง เพื่อนำข้อมูลที่ได้ไปคำนวณเพื่อตัดสินใจว่าบิตข้อมูลที่รับมาได้จะเป็น 0 หรือ 1

รูปที่ 2.6 สัญลักษณ์ในแทนเนอร์กราฟ (Tanner Graph)

โดยนิยามให้

q_{ij} คือ บิตข้อมูลข่าสารที่ถูกส่งจากบิตโนด c_i ไปยังเช็คโนด f_j

r_{ji} คือ บิตข้อมูลข่าวสารที่ส่งจากเช็คโนด f_j ไปยังบิตโนด c_i

$R_j = \{i : h_{ji} = 1\}$ คือ ชุดของตำแหน่งหลักที่เป็น 1 ในแถวที่ j เช่น $R_0 = \{0, 1, 2\}$

$R_{ij} = \{i' : h_{ji'} = 1\} \setminus \{i\}$ คือ ชุดของตำแหน่งหลักที่เป็น 1 ในแถวที่ j โดยไม่นับหลักที่ i เช่น $R_{0/1} = \{0, 2\}$

$C_j = \{i : h_{ji} = 1\}$ คือ ชุดของตำแหน่งหลักที่เป็น 1 ในหลักที่ j เช่น $C_0 = \{0, 1, 2\}$

$C_{ij} = \{i' : h_{j'i} = 1\} \setminus \{j\}$ คือ ชุดของตำแหน่งแถวที่เป็น 1 ในหลักที่ i โดยไม่นับหลักที่ j

$p_i = \Pr(c_i = 1|y_i)$ คือ การคำนวณค่าสำหรับ $\forall i, j$ โดยกำหนดให้ $h_{ij} = 1$

วิธีการถอดรหัสด้วยแทนเนอร์กราฟนั้นมีหลายวิธี และที่ได้รับความสนใจมี 2 วิธีนั่นคือ รูปแบบของความน่าจะเป็นในการคำนวณ (Message Passing Algorithm) และรูปแบบของลอการิทึม (Log-Domain Algorithm) การถอดรหัสทั้งสองรูปแบบนี้จะมีขั้นตอนในการคำนวณประกอบด้วย 5 ขั้นตอน

2.5.2.1 ขั้นตอนการถอดรหัสในรูปแบบของความน่าจะเป็น

ทฤษฎีการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำในรูปแบบความน่าจะเป็น

ขั้นตอนที่ 1 การกำหนดค่าเริ่มต้น

ในการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำเป็นการคำนวณหาค่าบิตโนดโดยใช้ในสมการ 2.33 และสมการ 2.34 ซึ่งจะทำการคำนวณในครั้งแรกและครั้งเดียวเท่านั้น นั่นคือการกำหนดค่าเริ่มต้น จากนั้นในการคำนวณหาค่าบิตโนดครั้งต่อๆ ไปจะใช้สมการ 2.37 และสมการ 2.38 แทน

การคำนวณค่าจากโนดบิต ในกรณีที่ผ่านช่องสัญญาณรบกวนแบบ AWGN

$$q_{ij}(0) = 1 - P_i = P_r\left(c_i = 0 / y_i\right) = \frac{1}{1 + e^{+2y_i/\sigma^2}} \tag{2.33}$$

$$q_{ij}(1) = P_i = \frac{1}{1 + e^{-2y_i/\sigma^2}} \tag{2.34}$$

นิยามให้

$q_{ij}(0)$, $q_{ij}(1)$ คือ ค่าคาดคะเนความน่าจะเป็นของบิต '0' และบิต '1' ที่โนดบิต i ส่งไปยังโนดเช็ค j

y_i คือ บิตข้อมูลข่าวสารที่ได้รับการเข้ารหัสและผ่านช่องสัญญาณ หรือค่าที่ได้รับของบิตรหัส i

σ^2 คือ ค่าความแปรปรวนของสัญญาณรบกวน

ขั้นตอนที่ 2 การคำนวณหาค่าเช็คโนด

$$r_{ji}(0) = 0.5 + 0.5 \prod_{i' \in R_{ji}} \left[1 - 2q_{i'j}(1)\right] \tag{2.35}$$

$$r_{ji}(1) = 1 - r_{ji}(0) \tag{2.36}$$

ขั้นตอนที่ 3 การคำนวณหาค่าบิตโนดในรอบที่สอง

$$q_{ij}(0) = K_{ij}(1-P_i)\prod_{j'\in c_{i/j}} r_{j'i}(0) \tag{2.37}$$

$$q_{ij}(1) = K_{ij}P_i\prod_{j'\in c_{i/j}} r_{j'i}(1) \tag{2.38}$$

โดยเลือกค่า K_{ij} เพื่อให้ $q_{ij}(0) + q_{ij}(1) = 1$

ขั้นตอนที่ 4 การตัดสินอย่างละเอียด

โอกาสและความน่าจะเป็นของค่าบิตโนด q ทุกตัว

$$Q_i(0) = K_{ij}(1-P_i)\prod_{j\in c_i} r_{ji}(0) \tag{2.39}$$

$$Q_i(1) = K_{ij}P_i\prod_{j\in c_i} r_{ji}(1) \tag{2.40}$$

โดยเลือกค่า K_{ij} เพื่อให้ $Q_i(0) + Q_i(1) = 1$

ขั้นตอนที่ 5 การตัดสินใจอย่างหยาบ

$$c_i \begin{cases} 1\ ; \text{ถ้า } Q_i(1) > Q_i(0) \\ 0\ ; \text{ในกรณีอื่นๆ} \end{cases} \tag{2.41}$$

ถ้า $\mathbf{cH}^T = \mathbf{0}$ หรือจำนวนของการวนซ้ำมากกว่าขอบเขตที่กำหนดแล้ว ให้หยุดและกลับไปทำงานใน**ขั้นตอนที่** 2

2.5.2.2 ขั้นตอนการถอดรหัสในรูปแบบของลอการิทึม

ทฤษฎีการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำในโดเมนของลอการิทึม

ขั้นตอนที่ 1. การนิยามค่าตัวแปล

$$L(q_{ij}) = \alpha_{ij}\beta_{ij}$$

α_{ij} คือ เครื่องหมายของบิตข้อมูลที่เป็นบวกหรือลบเท่านั้น

β_{ij} คือ ค่าของบิตข้อมูลที่เป็นบวกเท่านั้น

ขั้นตอนที่ 2. เริ่มต้นการถอดรหัสในรูปแบบของลอกโดเมน

2.1 การกำหนดค่าเริ่มต้น

$$L(c_i) = -8y_i \tag{2.42}$$

2.2 การคำนวณหาค่าเช็คโนด

$$L(r_{ji}) = \prod_{i' \in R_{ji}} \alpha_{i'j} \cdot \phi\left[\sum_{i' \in R_{ji}} \phi(\beta_{i'j})\right] \tag{2.43}$$

ขั้นตอนที่ 3. การคำนวณหาค่าบิตโนดในรอบที่สอง

$$L(q_{ij}) = L(c_i) = \sum_{j' \in C_{ij}} L(r_{j'i}) \tag{2.44}$$

ขั้นตอนที่ 4. การตัดสินใจแบบละเอียด

$$L(Q_i) = L(C_i) + \sum_{j \in C_i} L(r_{ji}) \tag{2.45}$$

ขั้นตอนที่ 5. การตัดสินใจแบบหยาบ

$$c_i \begin{cases} 1 \text{ ; ถ้า } L(Q_i) < 0 \\ 0 \text{ ; ในกรณีอื่นๆ} \end{cases} \tag{2.46}$$

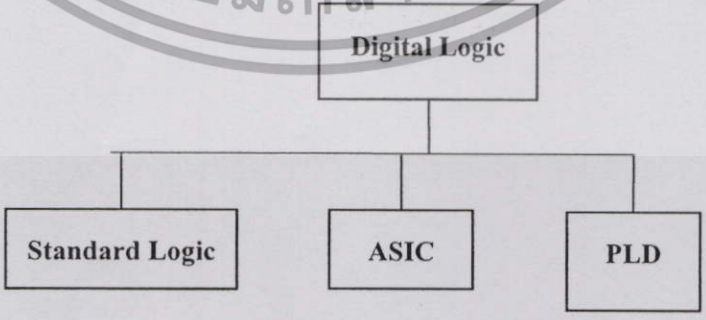
การถอดรหัสในรูปแบบของลอการิทึมเหมาะสำหรับ การนำมาออกแบบการถอดรหัสในแบบฮาร์ดแวร์มากกว่ารูปแบบของการอาศัยความน่าจะเป็น เนื่องจากในขั้นตอนของการเก็บค่า ในการแลกเปลี่ยนข้อมูลระหว่างบิตโนด i กับเช็คโนด j จะใช้หน่วยความจำชั่วคราว (RAM) น้อยกว่าการถอดรหัสแบบความน่าจะเป็น เนื่องจากการถอดรหัสแบบความน่าจะเป็นจะต้องเก็บถึง 2 ค่า คือ ค่าแรกคือโอกาสที่บิตข้อมูลจะเป็นศูนย์ และค่าที่สองคือโอกาสที่บิตข้อมูลจะเป็นหนึ่ง แต่การถอดรหัสในแบบลอการิทึมจะเก็บเพียงค่าเดียว นอกจากนั้นยังมีเรื่องของความซับซ้อนในการหาค่าเช็คโนด r_{ij} ตั้งแต่ในครั้งแรก การถอดรหัสแบบลอการิทึมจะใช้เพียงแค่การบวก แต่การถอดรหัสในแบบความน่าจะเป็นจะใช้การคูณ ถ้าในทางซอฟแวร์เรื่องแบบนี้อาจจะไม่แตกต่างกันมากนัก แต่ในทางฮาร์ดแวร์ จะเป็นปัญหาเนื่องจากการคูณจะใช้ทรัพยากรมากกว่าและทำงานได้ช้ากว่าการบวก ดังนั้นวิทยานิพนธ์ฉบับนี้จึงมุ่งเน้นที่การถอดรหัสในรูปแบบของโดเมนลอการิทึมเป็นหลัก

บทที่ 3
เอฟพีจีเอพื้นฐานและการออกแบบ

เทคโนโลยีการออกแบบฮาร์ดแวร์สำหรับระบบดิจิตอลได้รับการพัฒนาอย่างรวดเร็ว ซึ่งวงจรดิจิตอลจะมีความซับซ้อนและขนาดใหญ่ขึ้นเรื่อยๆ ในอดีตการออกแบบวงจรรวมหรือไอซี (Integration Circuit) จะต้องกำหนดประเภทของเกตหรืออุปกรณ์ต่างๆ มาโดยผู้ผลิต เช่น OR, AND, หรือ NAND นำมาทำการเชื่อมต่อเพื่อออกแบบวงจรบนบอร์ดทดลอง (Proto-Board) และเชื่อมต่อกับสวิตช์หรือตัวแสดงผลอื่นๆ เข้ากับลอจิกเกต บนบอร์ดทดลอง และวิเคราะห์ผลการออกแบบ ว่าเป็นไปตามทฤษฎีหรือไม่ การออกแบบวงจรรวมในลักษณะนี้ ต้องเสียเวลาในการเชื่อมต่อวงจรจริง ต้องใช้ลอจิกเกตหรืออุปกรณ์อื่นๆ จำนวนมาก และการแก้ไขก็จะทำได้ลำบาก ซึ่งต่างจากเอฟพีจีเอ (Field Programmable Gate Array : FPGA) การออกแบบวงจรด้วยเอฟพีจีเอสามารถทำได้ด้วยระยะเวลาอันสั้น เนื่องจากวงจรที่ออกแบบสามารถทำได้ด้วยคอมพิวเตอร์ส่วนบุคคล และทำการจำลองการทำงานเพื่อวิเคราะห์ผลที่ได้จากการออกแบบ จากนั้นทำการโปรแกรมวงจรที่ออกแบบลงบนชิพเอฟพีจีเอ โดยในปัจจุบันบอร์ดเอฟพีจีเอมีประสิทธิภาพสูงขึ้นมาก และจำนวนทรัพยากรในบอร์ดมีให้ใช้อย่างมากมาย แต่ราคากลับลดลงอย่างรวดเร็ว ดังนั้นการออกแบบวงจรรวมด้วยเอฟพีจีเอจึงเป็นสิ่งที่ได้รับความสนใจ

3.1 ประเภทของวงจรรวมแบบดิจิตอล

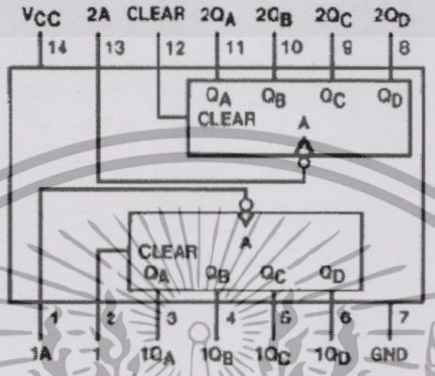
ประเภทของวงจรรวมแบบดิจิตอลแบ่งได้เป็น 3 ประเภท คือ ลอจิกมาตรฐาน (Standard Logic) เอสิก (Application Specific Integrated Circuit: ASIC) และพีแอลดี (Programmable Logic Device: PLD) ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 ไดอะแกรมการออกแบบวงจรดิจิตอล

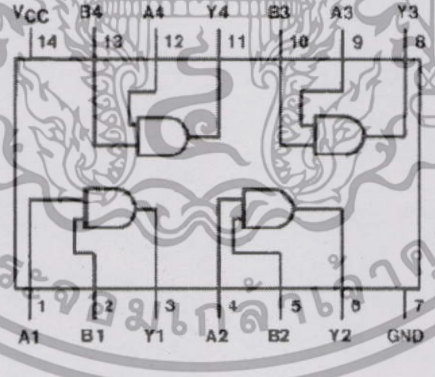
3.2 ไอซีมาตรฐานและวงจรรวมแบบมาตรฐาน

ไอซีมาตรฐานจะได้รับความนิยมนำไปใช้ในการออกแบบวงจรรวมดิจิตอล ไอซีมาตรฐานมีหลายประเภท เช่น ไอซีซีมอส (CMOS) ตระกูล 4000 และ 74HC00 เช่น ซีมอสเบอร์ 74HC393 จะประกอบไปด้วยวงจรนับเลขฐานสอง 4 บิต 2 ตัว ดังแสดงในรูปที่ 3.2



รูปที่ 3.2 ไอซีซีมอส (CMOS)

หรือวงจรรวมทีทีแอล (TTL) ตระกูล 74LS00 เช่น ไอซีทีทีแอลเบอร์ 74LS08 จะประกอบไปด้วยตัวแอนเกต 2 อินพุต 4 ตัว ดังแสดงในรูปที่ 3.3

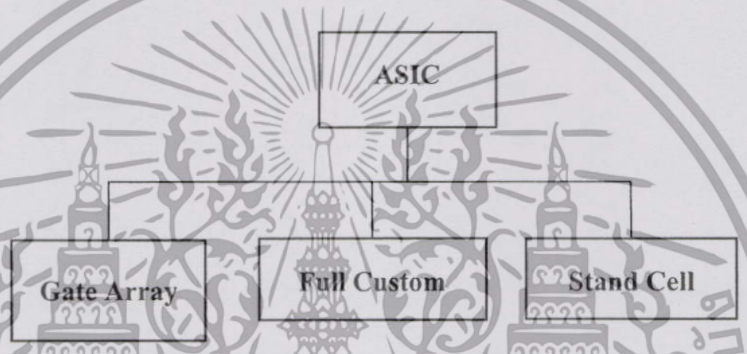


รูปที่ 3.3 วงจรรวมทีทีแอล (TTL)

รวมถึงไมโครคอนโทรลเลอร์ และหน่วยความจำต่างๆ เป็นต้น ไอซีสำเร็จรูปเหล่านี้จะมีการทำงานทางลอจิกแบบตายตัว จึงไม่เหมาะสมกับการออกแบบรวมขนาดใหญ่หรือความถี่สูงเนื่องจากเกิดเวลาล่าช้า (Delay) และวงจรรวมประเภทนี้ยังต้องการใช้พลังงานมาก จึงได้มีการคิดค้นชิพดิจิตอลอเนกประสงค์หรือไอซีขึ้นมาใช้แทน จึงทำให้ในซึ่งปัจจุบันการออกแบบวงจรดิจิตอลหันมา นิยมใช้อุปกรณ์พีแอลดีและเอสิกมากกว่าเนื่องจากมีขนาดเล็ก ประหยัดพลังงานและมีประสิทธิภาพสูง

3.3 วงจรรวมประเภทเอสิก (ASIC)

วงจรรวมประเภทเอสิก (Application Specific Integrated Circuit : ASIC) การออกแบบวงจรประเภทนี้ผู้ออกแบบจะใช้โปรแกรมจากบริษัทผู้ผลิต เพื่อออกแบบการทำงานทั้งหมดของวงจรเอง โดยจะเริ่มตั้งแต่การจัดเรียงตัวเซลลอจิก เช่น แอนด์เกท (AND) ออร์เกท (OR) ฟลิบฟล็อบ และมัลติเพล็กเซอร์ รวมไปถึงลักษณะ การจัดวางตัวอุปกรณ์บนไอซี การเจือสารและสร้างชั้นสาร (Mask) ต่างๆ เมื่อทำการจำลองการทำงานเสร็จแล้วก็สามารถส่งไฟล์ที่ได้จากการออกแบบนี้ไปให้บริษัทผู้ผลิตทำการเจือสารเพื่อสร้างวงจรรวมต่อไป ระยะเวลาตั้งแต่การออกแบบจนได้วงจรรวมนั้นใช้เวลานานถึงหลายเดือน และเสียค่าใช้จ่ายสูง



รูปที่ 3.4 ไดอะแกรมของวงจรรวมเอสิก

ดังนั้นการออกแบบประเภทเอสิกนี้จึงนิยมใช้ในเชิงพาณิชย์ เนื่องจากในกรณีที่จำนวนการผลิตจำนวนชิ้นงานมากจะทำให้มีต้นทุนที่ถูกกว่าแบบอื่นๆ และการออกแบบวงจรรวมประเภทเอสิกสามารถแบ่งออกได้เป็น 3 ประเภท ดังนี้

3.3.1 วงจรรวมแบบเกตอาเรย์

โดยทั่วไปแล้ววงจรเกตที่ใช้สร้างวงจรใช้พื้นที่ประมาณ 25-30 % ของพื้นที่ซิลิกอน [15] เนื่องจากส่วนพื้นที่ ที่เหลือนั้นใช้สำหรับเชื่อมต่อวงจรเกตเข้าด้วยกันกับขั้วต่อสายไฟ ซึ่งโครงสร้างวงจรรวมประเภทเกตอาเรย์ (Gate Array) นี้ ประกอบไปด้วยแถวลำดับของวงเกตกับขั้วต่อสายไฟ (Pad) สำหรับเชื่อมต่อกับวงจรภายนอก ซึ่งผู้ใช้ต้องออกแบบการเชื่อมต่อระหว่างเกตและขั้วต่อสายไฟให้ได้วงจรตามความต้องการ แล้วนำไฟล์ที่ได้จากการออกแบบไปทำการเจือสารที่บริษัทผู้ผลิต

3.3.2 วงจรรวมแบบฟูลัสตัม

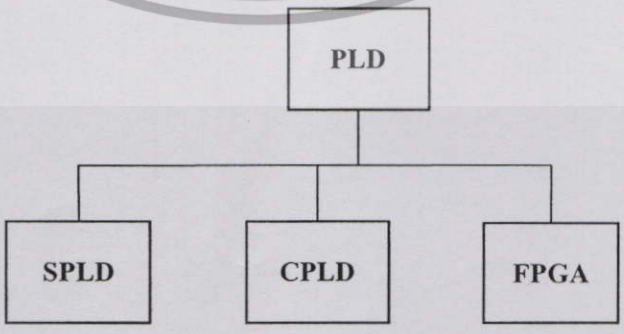
การออกแบบวงจรแบบฟูลัสตัม (Full Custom) จะเจือสารเป็นลักษณะที่รวบรวมการออกแบบหลายวงจรรวมลงบนแผ่นซิลิกอนแผ่นเดียวกัน เพื่อประหยัดค่าใช้จ่ายที่ใช้เจือสาร วงจรรวมประเภทฟูลัสตัม (Full Custom) นี้ ผู้ใช้เป็นผู้ออกเองแบบทั้งหมด แล้วจึงส่งไฟล์ที่ได้ทำการออกแล้ว ไปให้บริษัทผู้ผลิตเจือสาร ในรูปแบบของไฟล์ข้อมูลมาตรฐาน เป็นต้น

3.3.3 วงจรรวมแบบเซลล์มาตรฐาน (Standard Cell)

เนื่องจากผู้ใช้จำนวนมากมีความต้องการจะรวบรวมวงจรรวมมาตรฐาน เช่น วงจรตระกูล 7400 วงจรตระกูล 4000 จำนวนหลายวงจรเข้าด้วยกัน เพื่อออกแบบวงจรขนาดใหญ่ (Mega cell) ด้วยเหตุนี้ จึงทำให้เกิดปัญหาในการใช้พื้นที่ซิลิกอนสำหรับการเชื่อมโยงของวงจรรวมแบบเกตอาเรย์เกิดมากขึ้น ทำให้ไม่เหมาะสำหรับการนำไปใช้งานจริง และทำให้เกิดข้อจำกัดสำหรับการใช้งานกับวงจรที่มีความซับซ้อน จึงทำให้เกิดวงจรรวมเอสิกประเภทเซลล์มาตรฐาน (Standard Cell) ขึ้นมา ซึ่งวงจรรวมประเภทนี้ลูกค้าจะเป็นผู้กำหนดตำแหน่งของเซลล์มาตรฐาน การเชื่อมต่อภายในของแต่ละเซลล์จากแฟ้มข้อมูล (Library) ของบริษัทผู้ผลิตมาออกแบบวงจรรวม การผลิตวงจรรวมเซลล์มาตรฐานนั้นมีต้นทุนสูงกว่าวงจรรวมแบบเกตอาร์เรย์ และใช้ระยะเวลาการเจือสารนานกว่าสองถึงสามเท่า วงจรรวมประเภทนี้จึงเหมาะสมกับการใช้งานเชิงพาณิชย์ที่มีการผลิตจำนวนหมื่นตัวขึ้นไป

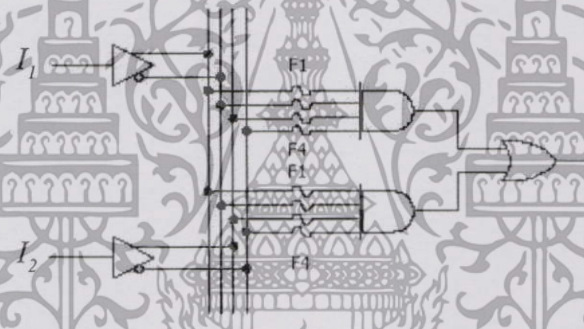
3.4 วงจรรวมแบบพีแอลดี

วงจรรวมแบบพีแอลดี (Programmable Logic Device: PLD) นี้แบ่งออกได้เป็น 3 ประเภท คือ เอสพีแอลดี (Simple Programmable Logic Device: SPLD) ซีพีแอลดี (Complex Programmable Logic Device: CPLD) และเอฟพีจีเอ (Field Programmable Gate Array: FPGA) ดังแสดงในรูปที่ 3.5



รูปที่ 3.5 ไดอะแกรมของวงจรรวมแบบพีแอลดี

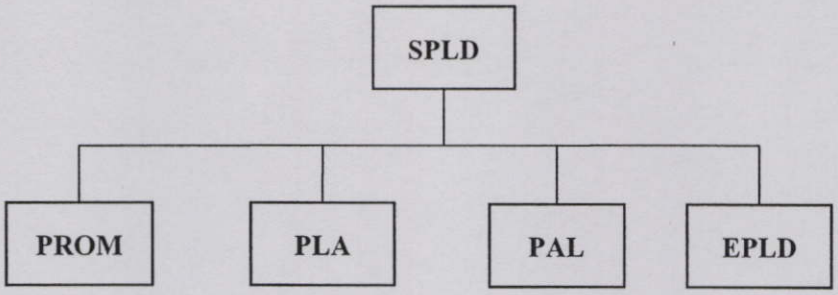
พีแอลดี (PLD) จะมีโครงสร้างพื้นฐานเหมือนกันด้านลอจิกต่อกันเป็นกลุ่มซึ่งมีทั้ง วงจรคอมบิเนชัน (Combination) และซีเควนเชียล (Sequential) ตามความเหมาะสมของแต่ละระบบ โดยไอซีพีแอลดีทุกชนิดจะมีหลักการของวงจรภายในที่เหมือนกันคือ จะอยู่ในรูปแบบของเป็นผลคูณร่วมบวก (Sum of product) ซึ่งจะประกอบไปด้วยชุดของแอนด์เกทต่อร่วมกับอาร์เกท และในการโปรแกรมจะเป็นการเลือกว่าอินพุตภายในของแอนด์เกทกับสัญญาณอินพุตใดบ้างที่จะต้องต่อถึงกัน ซึ่งจะมีทั้งจากภายนอกและสัญญาณป้อนกลับจากเอาต์พุทภายใน ซึ่งการออกแบบวงจรดิจิตอลด้วยอุปกรณ์พีแอลดีนั้นใช้เวลาน้อยกว่าการออกแบบด้วยวงจรรวมประเภทเอสิก การออกแบบวงจรรวมดิจิตอลประเภทพีแอลดีลูกค้าทำได้เองโดยคอมพิวเตอร์ส่วนตัว ซึ่งสามารถโปรแกรมเพื่อปรับเปลี่ยนวงจรในพีแอลดีได้หลายครั้ง ทำให้ผู้ใช้สามารถปรับปรุงความสามารถของอุปกรณ์ให้ดีขึ้นโดยไม่ต้องเปลี่ยนอุปกรณ์ใหม่ และมีต้นทุนที่ถูกกว่าเอสิกในกรณีที่ต้องการผลิตอุปกรณ์จำนวนน้อย เพราะในการออกแบบนั้นไม่ต้องเสียค่า NRE (Non-Recurring Engineering cost) ซึ่งเป็นค่าติดตั้งเครื่องจักร (Set-up Machine) ในการผลิตวงจรรวมซึ่งจะมีราคาแพงกว่า



รูปที่ 3.6 วงจรพื้นฐานของพีแอลดีในรูปของผลคูณร่วมบวก

3.4.1 อุปกรณ์เอสพีแอลดี

โครงสร้างเอสพีแอลดี(Simple Programmable Logic Device: SPLD) มีความซับซ้อนน้อยกว่าอุปกรณ์พีแอลดี เนื่องจากการสร้างเอสพีแอลดี (SPLD) จะใช้ออกแบบวงจรลอจิกเกตหลายๆ วงจรรวมเข้าด้วยกัน จึงทำให้ประหยัดเนื้อที่ของแผ่นวงจร (Printed Circuit Board) และโครงสร้างของอุปกรณ์เอสพีแอลดีนี้ประกอบไปด้วยแอนด์เกตและออร์เกต ซึ่งเรียงในรูปแบบของแถวลำดับ (Array) จึงทำให้สามารถโปรแกรมให้ลอจิกเกตเหล่านี้ เชื่อมต่อเข้าด้วยกันให้เป็นวงจรตามความต้องการ และอุปกรณ์เอสพีแอลดีมีด้วยกันหลายชนิด เช่น พรอม (Programmable Read Only Memory: PROM) พีแอลเอ (Programmable Logic Array: PLA) พีเอ-แอล (Programmable Array Logic: PAL) และอีพีแอลดี (Erasable Program Logic Device) ดังแสดงในรูปที่ 3.7

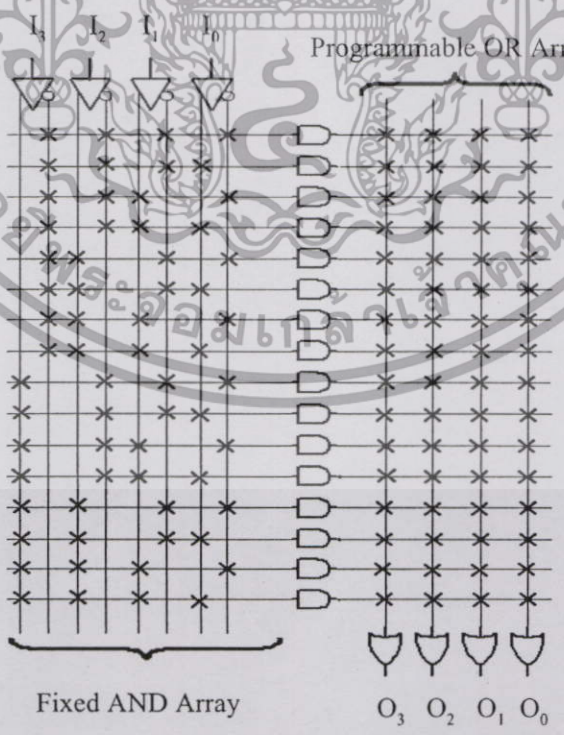


รูปที่ 3.7 ไดอะแกรมแอสพีแอลดี

สำหรับการโปรแกรมทางกายภาพของแอสพีแอลดีนั้นอินพุต ต่างๆ จะถูกต่อผ่านฟิวส์เข้ากับแหล่งสัญญาณ ซึ่งเมื่อจะทำการโปรแกรมก็จะทำการตัดหรือต่อฟิวส์ในตัวอุปกรณ์ เอสพีแอลดีบางชนิดใช้มอสทรานซิสเตอร์แทนฟิวส์ ดังนั้นจึงทำให้สามารถลบและโปรแกรมได้หลายครั้ง

3.4.1.1พรอม (Programmable Read-Only Memory: PROM)

พรอมเป็นอุปกรณ์ที่ใช้เป็นหน่วยความจำประเภทรอม (ROM) ชนิดหนึ่ง ซึ่งโครงสร้างประกอบด้วยแถวลำดับของแอนด์เกตและออร์เกต (And – Or Array) โดยเอาต์พุตของพรอมแสดงเป็นสมการฟังก์ชันของผลคูณร่วมบวก (Sum of Product) ของอินพุตแถวลำดับของแอนด์เกตและออร์เกตถูกเชื่อมต่อในลักษณะของวงจรถอดรหัส

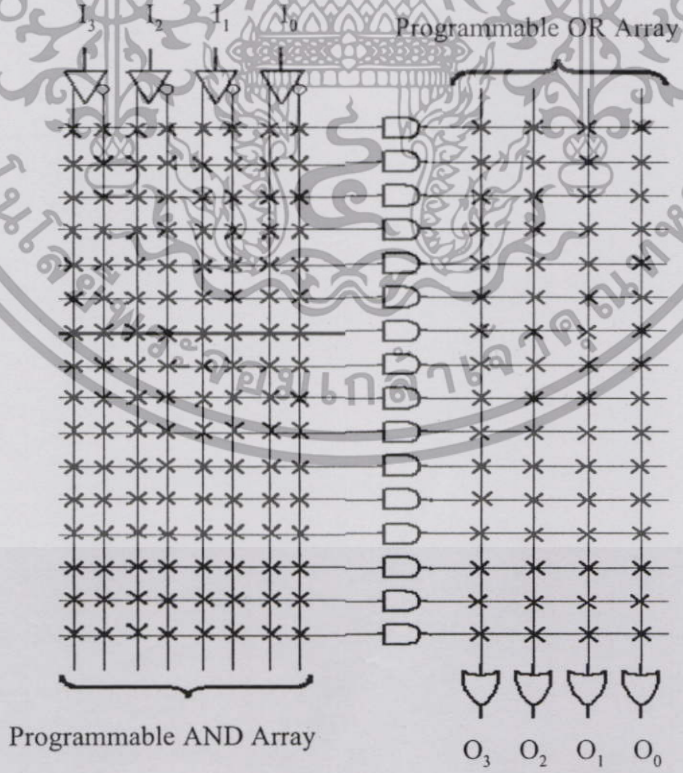


รูปที่ 3.8 วงจรพื้นฐานภายในของพรอม

รูปที่ 3.8 แสดงให้เห็นถึงการเชื่อมต่อของแอนด์เกตและออร์เกตของพรอม (PROM) ขนาด 16x4 บิต วงจรบนสุดด้านซ้ายคือแอนด์เกตจะให้ผลคูณ (Product) และเมื่อรับอินพุตเป็น 0000 แอนด์เกตที่อยู่ถัดลงมาจะเป็นผลคูณของกรณีที่อินพุตเป็น 0001, 0010, ..., 1111 เป็นต้น และในกรณีที่มีอินพุต n ตัว ค่าที่เป็นไปได้ทั้งหมดเท่ากับ 2^n และค่าเหล่านี้จะถูกจัดวางอยู่ในส่วนของแถวลำดับของแอนด์เกตซึ่งไม่สามารถแก้ไขได้ แต่ในส่วนของแถวลำดับอาร์เกตจะอนุญาตให้แก้ไขได้และให้ทำการโปรแกรมได้

3.4.1.2 พีแอลเอ (Programmable Logic Array: PLA)

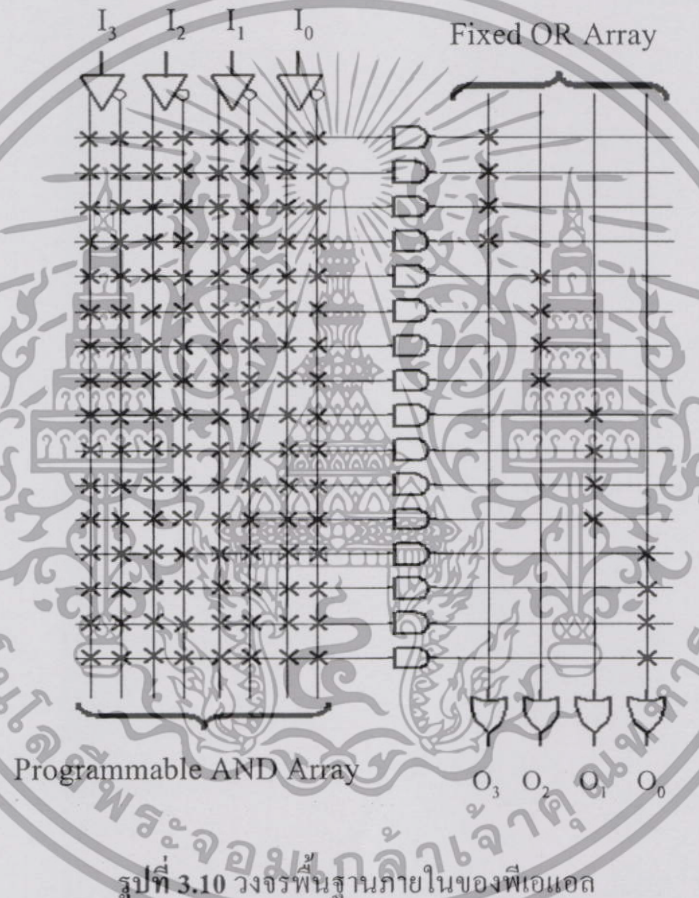
อุปกรณ์พีแอลเอได้รับการคิดค้นโดยบริษัท Philips ประมาณปี 1970 และมีลักษณะเด่นคือสามารถโปรแกรมการเชื่อมต่อได้ทั้งสองด้านคือทางด้านแอนด์เกตและด้านออร์เกต จึงทำให้ปรับเปลี่ยนรูปแบบการทำงานได้มากขึ้น และในส่วนของโครงสร้างของพีแอลเอจะประกอบด้วยแถวลำดับของแอนด์เกตและออร์เกต แถวลำดับของแอนด์เกตต่อกับอินพุตและแถวลำดับของออร์เกตต่อกับเอาต์พุต อุปกรณ์พีแอลเอนี้ผู้ใช้สามารถกำหนดการเชื่อมต่อได้ทั้งแถวลำดับของแอนด์เกตและออร์เกตได้เอง แต่พีแอลเอมีข้อเสียในเรื่องความยุ่งยากในการสร้างและคุณสมบัติทางด้านความเร็ว เนื่องจากสัญญาณจะต้องผ่านแถวลำดับของแอนด์เกตและออร์เกต ดังรูปที่ 3.9



รูปที่ 3.9 วงจรพื้นฐานภายในของพีแอลเอ

3.4.1.3 พีเอแอล (Programmable Array Logic: PAL)

อุปกรณ์พีเอแอลถูกพัฒนาขึ้นเพื่อลดข้อจำกัดของพีแอลเอจากการหน่วงของเวลาที่เกิดจากการเชื่อมต่อฟิวส์ในแถวลำดับ (Propagation Delay Time) พีเอแอลมีลักษณะโครงสร้างที่ใกล้เคียงกับพรอมและพีแอลเอ แต่สิ่งที่แตกต่างกันคือพีเอแอลจะทำการโปรแกรมได้เพียงด้านเดียว ซึ่งโครงสร้างของพีเอแอลจะประกอบด้วยแถวลำดับของแอนด์เกตที่สามารถโปรแกรมการเชื่อมต่อได้และแถวลำดับออร์เกตที่ถูกกำหนดการเชื่อมต่อไว้แล้ว โดยแถวลำดับของแอนด์เกตต่อกับอินพุตและแถวลำดับของออร์เกตต่อกับเอาต์พุต ซึ่งโครงสร้างของพีเอแอลแสดงดังรูปที่ 3.10



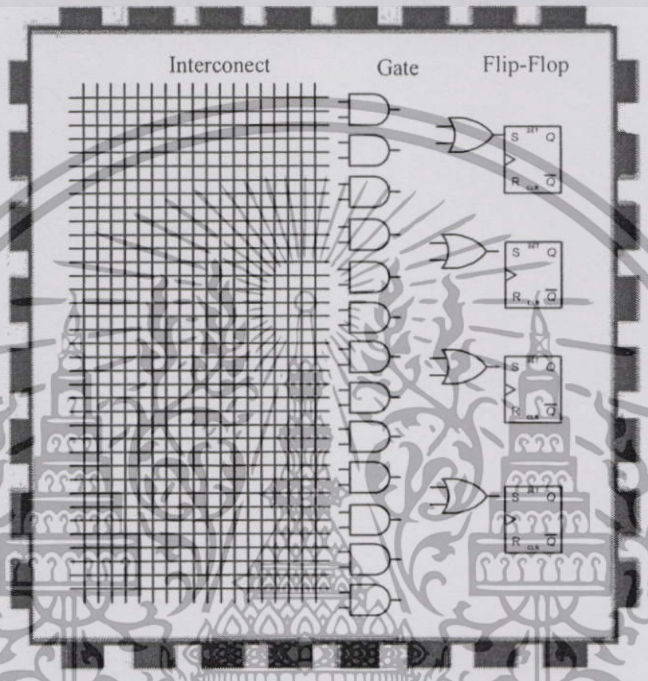
รูปที่ 3.10 วงจรพื้นฐานภายในของพีเอแอล

3.4.1.4 อีพีแอลดี (Erasable Programmable Logic Device EPLD)

อีพีแอลดี (EPLD) เป็นอุปกรณ์ที่สามารถทำการโปรแกรมได้หลายครั้ง ซึ่งเหมาะสำหรับการทำวงจรต้นแบบ สำหรับเทคโนโลยีที่ใช้ในการสร้างจะเหมือนกับซีมอส (CMOS) และอีพรอม (EPROM) คือ การใช้มอสทรานซิสเตอร์เชื่อมต่อระหว่าง สัญญาณอินพุตกับจุดที่ต้องการแทนการใช้ฟิวส์แบบเดิม ทำให้สามารถโปรแกรมการต่อวงจรภายในอุปกรณ์ด้วยการจ่าย ไฟฟ้าตามขนาดที่กำหนดไว้และลบได้โดยใช้แสงอัลตร้าไวโอเลทฉาย ผ่านช่อง หน้าต่างกระจกของตัวชิพ

3.4.2 อุปกรณ์ซีพีแอลดี (Complex Programmable Logic Device: CPLD)

ซีพีแอลดีเป็นอุปกรณ์ที่รวมไอซีประเภทเอชพีแอลดี (SPLD) หลายตัวหรือเรียกว่า ไมโครเซลล์ (Macro Cells) เข้ามาเป็นอุปกรณ์ตัวเดียวกัน ซึ่งทำให้ได้ซีพีแอลดีที่มีความจุของจำนวนเกตที่มากกว่าเอชพีแอลดี โดยโครงสร้างของซีพีแอลดีประกอบด้วย เกตต่างๆ ฟลิปฟลอป และส่วนที่ใช้เชื่อมต่อภายใน (Interconnect) และโครงสร้างของซีพีแอลดีแสดงได้ดังรูปที่ 3.11



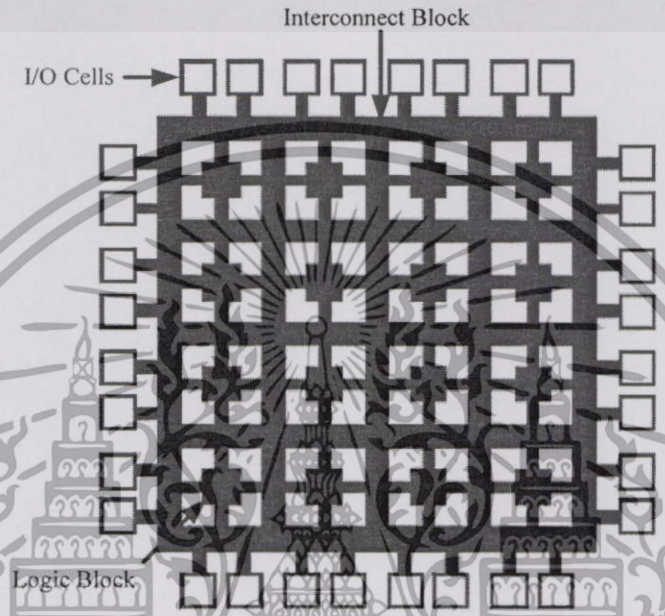
รูปที่ 3.11 โครงสร้างของซีพีแอลดี

3.4.3 อุปกรณ์เอฟพีจีเอ (Field Programmable Gate Array: FPGA)

เอฟพีจีเอ (Field Programmable Gate Array: FPGA) เป็นอุปกรณ์อิเล็กทรอนิกส์ที่สามารถโปรแกรมได้เพื่อให้มีฟังก์ชันการทำงานตามที่ต้องการ และมีอยู่ 4 ชนิด ได้แก่ Symmetrical Array, Row-Based, Hierarchical PLD และ Sea-of-Gates และเอฟพีจีเอแต่ละชนิด จะมีโครงสร้างภายในและลักษณะการเชื่อมต่อแตกต่างกันไป และได้รับการพัฒนาจากหลายบริษัท เช่น Xilinx และ Altera เอฟพีจีเอจะมีปริมาณความจุของเกตสูงกว่าอุปกรณ์ซีพีแอลดีมาก และราคาถูกกว่าในกรณีที่จำนวนเกตเท่ากัน และการออกแบบเอฟพีจีเอสะดวกสบายและรวดเร็ว เนื่องจากการออกแบบและการโปรแกรมเอฟพีจีเอสามารถทำได้โดยใช้คอมพิวเตอร์ส่วนบุคคล จึงทำให้การออกแบบวงจรดิจิตอลด้วยเอฟพีจีเอนั้นเป็นที่นิยมในปัจจุบัน และในส่วนของการเปลี่ยนแปลงแก้ไขทำได้ง่าย การออกแบบด้วยเอฟพีจีเอนั้นไม่ต้องเสียค่าใช้จ่ายในการแก้ไขวงจรและการเจือสาร ซึ่งแนวโน้มราคา

ของอุปกรณ์เอฟพีจีเอจะถูกลงเพราะมีการแข่งขันในด้านการตลาดของบริษัทผู้ผลิตเอฟพีจีเอ คือ Xilinx และ Altera

โครงสร้างของเอฟพีจีเอนั้นมีโครงสร้างที่ซับซ้อนมากกว่าซีพีแอลดี ซึ่งโครงสร้างภายในของเอฟพีจีเอจะประกอบไปด้วย 3 ส่วนคือ บล็อกลอจิก (Logic Block) บล็อกเชื่อมต่อภายใน (Interconnect Block) และไอโอเซลล์ (I/O Cell) ซึ่งแสดงดังรูปที่ 3.12



รูปที่ 3.12 โครงสร้างทั่วไปของเอฟพีจีเอ

ลอจิกบล็อกจะถูกจัดให้อยู่ในรูปเมตริกซ์ และภายในประกอบจะไปด้วย วงจรเกตต่างๆ รีจิสเตอร์ และหน่วยความจำ เพื่อใช้สร้างฟังก์ชันในการใช้งานโดยมีบล็อกเชื่อมต่อภายในทำหน้าที่เชื่อมต่อระหว่างลอจิกบล็อกและเชื่อมต่อกับไอโอเซลล์ (I/O Cell) ซึ่งทำหน้าที่เป็นอินพุตและเอาต์พุตเพื่อติดต่อกับวงจรภายนอก

3.5 ภาษาวีเอชดีแอล (VHSIC Hardware Description Language: VHDL)

วีเอชดีแอล (VHSIC Hardware Description Language: VHDL) ได้รับการพัฒนาโดยกระทรวงกลาโหมของสหรัฐอเมริกา (Department of Defense: DOD) ในช่วงปี ค.ศ. 1980 โดยมีจุดประสงค์ของโครงการอีเอชเอสไอซี (Very High Speed Integrated Circuits: VHSIC) คือ เพื่อพัฒนาการออกแบบวงจรรวมให้มีขีดความสามารถสูงขึ้น ทำการออกแบบวงจรรวมได้ง่ายขึ้น และมีเป้าหมายหลัก 2 ประการคือ

1. ต้องการภาษาที่รองรับการออกแบบวงจรที่ซับซ้อนได้
2. ต้องการภาษาที่เป็นมาตรฐานหรือเป็นภาษากลางที่สามารถแผยแพร่ผลงานการออกแบบภายในกลุ่มนักออกแบบด้วยกัน

ต่อมาในปี ค.ศ. 1986 ได้มีการปรับปรุงภาษาวีเอชดีแอล เพื่อให้สามารถกำหนดเป็นมาตรฐาน IEE (Institute of Electrical and Electronics Engineers) ซึ่งสามารถประกาศเป็นมาตรฐานได้ในเดือนธันวาคม ค.ศ. 1987 โดยอยู่ในหมวด IEE 1076-1987 หรือเรียกว่า VHDL'87 ภายหลังได้มีการปรับปรุงและพัฒนาอย่างต่อเนื่อง จึงได้มีการประกาศปรับปรุงอีกครั้งในปี ค.ศ. 1993 โดยอยู่ในหมวด IEE 1076-1993 หรือเรียกว่าสั้นๆ ว่า VHDL'93 โดยได้รับการปรับปรุงโดยการเพิ่มไวยากรณ์เพื่อให้ผู้ใช้สามารถใช้งานได้สะดวกยิ่งมากขึ้น

3.5.1 โครงสร้างพื้นฐานของภาษาวีเอชดีแอล

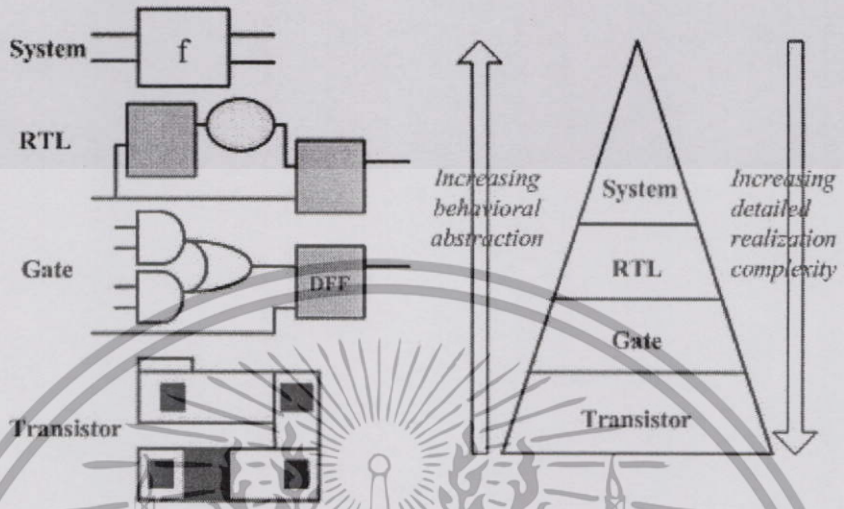
ภาษาวีเอชดีแอล (VHDL) เป็นภาษาที่ใช้สำหรับออกแบบวงจรดิจิตอล และมีโครงสร้างที่สามารถออกแบบวงจรได้หลายระดับ ตั้งแต่การออกแบบลอจิกเกต (Gate) ที่ไม่ซับซ้อน ง่ายต่อการเรียนรู้ ไปจนถึงการออกแบบระบบ (System) ที่ซับซ้อน รวมทั้งยังเป็นภาษาที่ผู้ใช้สามารถเข้าใจได้ง่าย และสิ่งที่สำคัญของภาษานี้คือ มีความเป็น Concurrent ซึ่งเป็นสิ่งที่ขาดไม่ได้สำหรับการออกแบบวงจรอิเล็กทรอนิกส์

ภาษาวีเอชดีแอลมีคุณสมบัติที่สามารถแสดงให้เห็นการออกแบบวงจรดิจิตอลได้หลายระดับ (Level of abstraction) ตั้งแต่ล่างสุดในระดับลอจิกเกต (Gate level) ไปจนถึงในระดับพฤติกรรมการทำงานของวงจร (Behavioral level) ซึ่งในแต่ละระดับจะมีรายละเอียดแตกต่างกัน กรณีที่ต้องการออกแบบวงจรคูณ โดยใช้ภาษาวีเอชดีแอลในการออกแบบสามารถทำได้หลายระดับ เช่น

1. อาจออกแบบวงจร โดยเขียนบรรยายพฤติกรรมหรือเรียกใช้โอเปอร์เรเตอร์บวก "+" ซึ่งในภาษาวีเอชดีแอล จะเขียนคล้ายกับซอฟต์แวร์ เช่น a <= b + c;
2. หรือออกแบบวงจรบวกในระดับลอจิก
3. หรืออาจออกแบบวงจรบวกในระดับล่างสุด คือระดับทรานซิสเตอร์ (Layout level)

จากที่ได้อธิบายมาแล้วนั้นแสดงให้เห็นว่า การออกแบบวงจรในลักษณะเดียวกัน การทำงานของฟังก์ชันการทำงานในวงจรเหมือนกัน แต่การออกแบบนั้นสามารถทำได้หลายระดับ ตั้งแต่ System,

RTL, Gate, ไปจนถึงระดับ Transistor ดังแสดงในรูปที่ 3.13 ขึ้นอยู่กับความเหมาะสมและทักษะความชำนาญของผู้ออกแบบเป็นหลัก



รูปที่ 3.13 ระดับของการออกแบบวงจรดิจิตอลในภาษาวีเอชดีแอล

โดยในแต่ละระดับของการออกแบบวงจรจะออกแบบต่างกันแต่การทำงานของวงจรจะเป็นแบบเดียวกัน เมื่อพิจารณาจากรูปพีระมิดมีรายละเอียดดังนี้

System Level จะเป็นการบรรยายพฤติกรรมการทำงานของระบบ หรือแบบอัลกอริทึม (Algorithm description) ของการทำงาน และจะคำนึงถึงข้อมูลเรื่องเวลา ซึ่งการออกแบบในลักษณะนี้จะจัดเป็นระดับบนสุดของการเขียนภาษาวีเอชดีแอล

RTL (Register Transfer Level) การออกแบบลักษณะนี้จะเป็นการบรรยายลงลึกในส่วนของวงจรในระดับ Architecture ในระดับนี้อาจเห็นรายละเอียดลึกลงไปถึงในวงจร เช่น Synchronous หรือ Asynchronous, State machines, Registers, Data path, ต่างๆ รวมถึงการไหลเข้าไหลออกของข้อมูลในแต่ละบล็อกของวงจรที่ออกแบบ

Gate Level เป็นการบรรยายการออกแบบลงลึกกว่าระดับ RTL ซึ่งจะมีรายละเอียดของวงจรระดับ ฟลิปฟลอป และลอจิกเกต ต่างๆ ที่เชื่อมต่อภายในวงจร

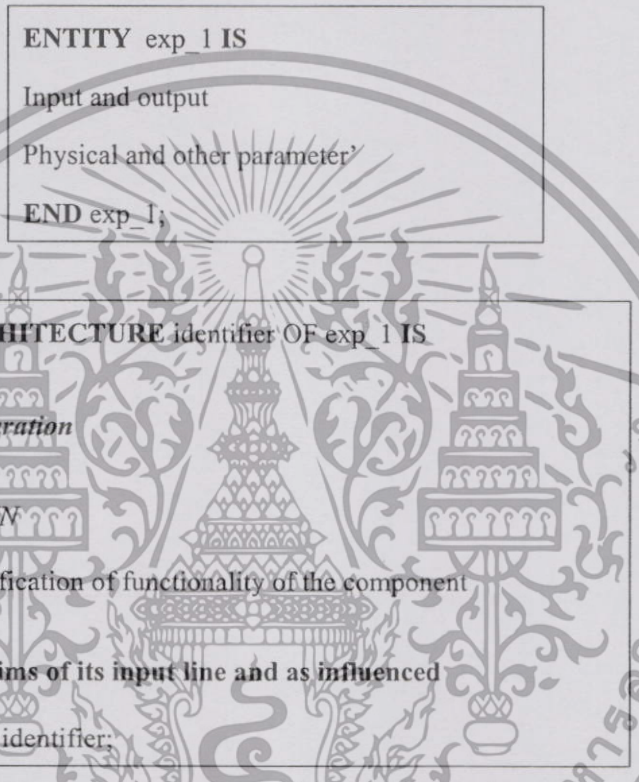
Transistor Level การออกแบบวงจรในระดับนี้จะเป็นระดับล่างสุดในระบบดิจิตอลหรือเรียกว่า Layout Level ในระดับนี้ภาษาวีเอชดีแอลไม่สามารถอธิบายได้ เนื่องจากข้อมูลจะเกี่ยวกับค่า Capacitances และ Resistances ของทรานซิสเตอร์เข้ามาเกี่ยวข้อง ซึ่งในระดับนี้ผู้ออกแบบต้องอาศัยซอฟต์แวร์ช่วยในการสังเคราะห์ในวงจรระดับทรานซิสเตอร์ (Layout synthesis) หรือโดยทั่วไปเรียกว่าซอฟต์แวร์ช่วยในการวางการเชื่อมต่อทรานซิสเตอร์ (Place and Route)

3.5.2 ความสามารถของภาษาวีเอชดีแอล (Capability)

1. ภาษาวีเอชดีแอลสามารถใช้เป็นสื่อกลางในการแลกเปลี่ยนระหว่างผู้ผลิตชิพกับ ผู้ออกแบบวงจรได้ (CAD Tools)
2. ใช้เป็นสื่อกลางในการแลกเปลี่ยนสื่อสารระหว่างซีเออี (CAE) และซีเอดีทูล (CAD Tool) เช่น ตัวภาษาซ้อสโค้ด (SOURCE CODE) ของวิเอชดีแอลสามารถคอมไพล์เลอร์ (Compiler) และซีมูเลเตอร์ (Simulators) ได้หลายตัวแตกต่างกัน
3. ภาษาวีเอชดีแอลสนับสนุนการออกแบบ แบบบนล่าง (Top Down Design) และแบบล่างขึ้นบน (Bottom Up Design) หรือผสมกันทั้งสองแบบในการออกแบบวงจร
4. ตัวภาษาวีเอชดีแอลเป็นภาษาแบบทั่วไป (Generic) ไม่อิงเทคโนโลยีใด ในขณะเดียวกันก็สนับสนุนหลายเทคโนโลยี
5. ภาษาวีเอชดีแอลสามารถอ่านและทำความเข้าใจโดยมนุษย์ (Human-readable)
6. สนับสนุนการออกแบบทั้งระบบแบบซิงโครนัส (Synchronous) และอะซิงโครนัส (Asynchronous)
7. ตัวภาษาวีเอชดีแอลเป็นภาษามาตรฐานรับรองโดยไออีอีเอ (IEEA) และเอเอนเอสไอ (ANSI) ทำให้โมเดลที่ออกแบบโดยภาษาวีเอชดีแอลสามารถเคลื่อนย้ายไปยังระบบใดก็ได้ และสามารถนำกลับมาใช้ได้ใหม่
8. สามารถเขียนโมเดลได้ขนาดไม่จำกัด ไม่มีข้อจำกัดในตัวภาษาเรื่องขนาดของโมเดล (ขึ้นอยู่กับซอฟแวร์)
9. ภาษาวีเอชดีแอลสนับสนุนการเขียนถึง 3 รูปแบบ ได้แก่ แบบบีเฮฟวีเออร์ (Behavioral Style) แบบสตรัคเจอรัล (Structural Style) แบบดาต้าโฟล์ว์ (Data Flow) หรือสามารถเขียนรวมกันได้ทั้ง 3 รูปแบบ
10. สนับสนุนการออกแบบวงจรขนาดใหญ่โดยใช้ความสามารถของส่วนประกอบ (Component) ฟังก์ชั่นโพรซีเดอร์ (Function Procedure) และแพคเกต (Package)
11. สามารถอธิบายตัวแปรที่เกี่ยวกับฟังก์ชั่นทางด้านเวลา เช่น Propagation delay, Min-Max delay, Setup, Holding Time สามารถอธิบายได้โดยตัวภาษา
12. ภาษาวีเอชดีแอลเป็นมาตรฐานที่ใช้โดยบริษัทและผู้ออกแบบหลายๆ แห่ง ฉะนั้นจึงง่ายที่จะทำความเข้าใจถึงแม้ว่าจะมาจากแหล่งต่างๆ
13. โมเดลที่สร้างขึ้นสามารถจำลองการทำงานได้ เพราะว่าตัวแปรภาษาได้ตรวจสอบตัวแปรทางซิมูเลชั่นซีเมนติกไว้ด้วย

3.5.3 องค์ประกอบของภาษาวีเอชดีแอล

ภาษาวีเอชดีแอลประกอบไปด้วย Entity, Architecture, Package, และ Configuration ซึ่งในการออกแบบวงจรดิจิตอลไม่ว่าจะเป็นวงจรที่ซับซ้อนหรือวงจรพื้นฐาน การออกแบบวงจรทุกวงจรจะต้องเขียนอยู่ในรูปส่วนองค์ประกอบ (Component) ของภาษาวีเอชดีแอลซึ่งเป็นพื้นฐานของการออกแบบ และภาษาวีเอชดีแอลนั้น จะประกอบไปด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และการกำหนดลักษณะเชิงสถาปัตยกรรม (Architecture) ดังแสดงในรูปที่ 3.14



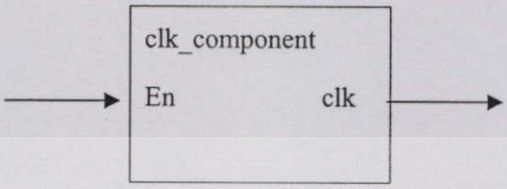
รูปที่ 3.14 การเชื่อมต่อและสถาปัตยกรรมของภาษาวีเอชดีแอล

การบรรยายเชื่อมต่อนี้ จะขึ้นต้นด้วยคำว่า ENTITY ตามด้วยชื่อขององค์ประกอบและคำว่า IS และภายในจะแสดงให้เห็นถึงพอร์ตการติดต่ออินพุตและเอาต์พุต พอร์ตขององค์ประกอบ โดยที่การทำงานขององค์ประกอบนั้นจะมีหน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาอินพุตและเอาต์พุต และพารามิเตอร์อื่นๆ ตามที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูป 3.14 และการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังคำว่า BEGIN

3.5.3.1 การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อ (Interface Description) จะเป็นระดับบนสุดของการออกแบบวงจร ซึ่งการออกแบบในระดับนี้จะต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบบล็อกการ

ออกแบบภายนอกอื่นๆ ดังแสดงในรูปที่ 3.15 ในบรรทัดแรกจะทำการกำหนดชื่อขององค์ประกอบ ซึ่งกำหนดให้เป็นชื่อ clock_component ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ในวงเล็บ IN และ OUT จากนั้นกำหนดโหมดของสัญญาณเป็นอินพุตหรือเอาต์พุต BIT เพื่อแสดงชนิดของข้อมูล




```
ENTITY clk_component IS
PORT ( en : IN BIT;
       clk : OUT BIT
       );
END clk_component;
```

รูปที่ 3.15 บล็อกไดอะแกรมการบรรยายการเชื่อมต่อขององค์ประกอบ

3.5.3.2 การกำหนดรูปแบบของการบรรยาย

หน้าที่การทำงานขององค์ประกอบ (Architecture Description) จะถูกบรรยายภายในส่วนนี้ การบรรยายสามารถกำหนดค่าของสัญญาณเอาต์พุตในเทอมของ clock_component ดังแสดงในรูปที่ 3.16

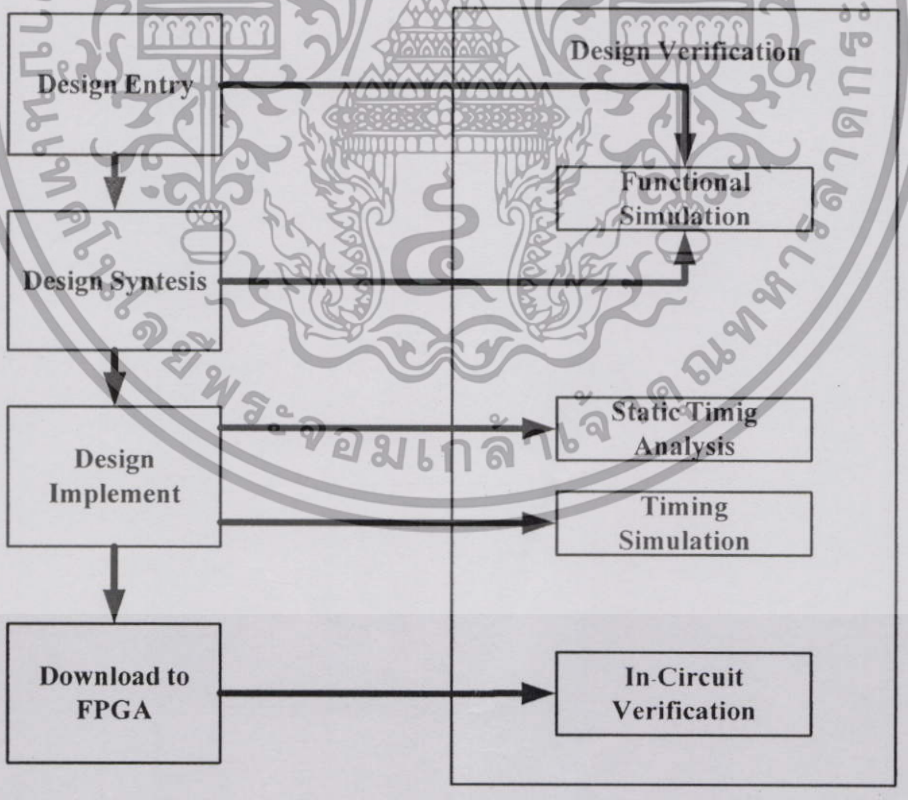
```
ARCHITECTURE behavioral OF clk_component IS
BEGIN
   PROCESS
      VARIABLE periodic : BIT : = '0';
   BEGIN

IF en = '1' THEN

clk <= periodic;
   END PDOCESS;
END clk_component;
```

รูปที่ 3.16 การบรรยายเชิงพฤติกรรมขององค์ประกอบ

ซึ่งเป็นการบรรยายในเชิงพฤติกรรมมี en เป็นอินพุต และมี clk เป็น Process เป็นค่าเริ่มต้น สำหรับการบรรยายในเชิงพฤติกรรมภายใน Process จะกำหนดให้ periodic เป็นตัวแปรที่มีค่าเริ่มต้นเป็น "0" ถ้าสัญญาณ en มีค่าเป็น "1" ค่าของ periodic ถูกคอมพลีเมนต์ และส่งค่าให้กับ clk ซึ่งจะได้เป็นสัญญาณเอาต์พุตและทำไปใช้งานหรือส่งต่อให้บล็อกอื่นทำงานต่อไป

3.6 การออกวงจรดิจิตอลด้วยเอฟพีจีเอ

การออกแบบวงจรดิจิตอลนั้นมีหลายขั้นตอนในการติดตั้งกับเอฟพีจีเอ โดยจะใช้ภาษาอธิบายฮาร์ดแวร์ (Hardware Description Language: HDL) เช่น ภาษาวีเอชดีแอล (VHDL) หรือภาษาเวริล็อก (Verilog HDL) สำหรับการออกแบบ และนำโค้ดที่ได้ไปสังเคราะห์ (Synthesis) ให้เป็นวงจร ซึ่งผลที่ได้จากการสังเคราะห์คือเน็ตลิสต์ในระดับเกต (Gate-level Netlist).จะได้เป็นรูปแบบมาตรฐานคือ EDIF (Electronic Data Interchange Format) หรือ XNF (Xilinx Netlist Format) ขั้นตอนการออกแบบและติดตั้งวงจรดิจิตอลบนเอฟพีจีเอ ดังแสดงในรูปที่ 3.17 ส่วนรายละเอียดขั้นตอนการออกแบบมีดังนี้



รูปที่ 3.17 ขั้นตอนการออกแบบวงจรดิจิตอลบนเอฟพีจีเอ

3.6.1 การออกแบบ

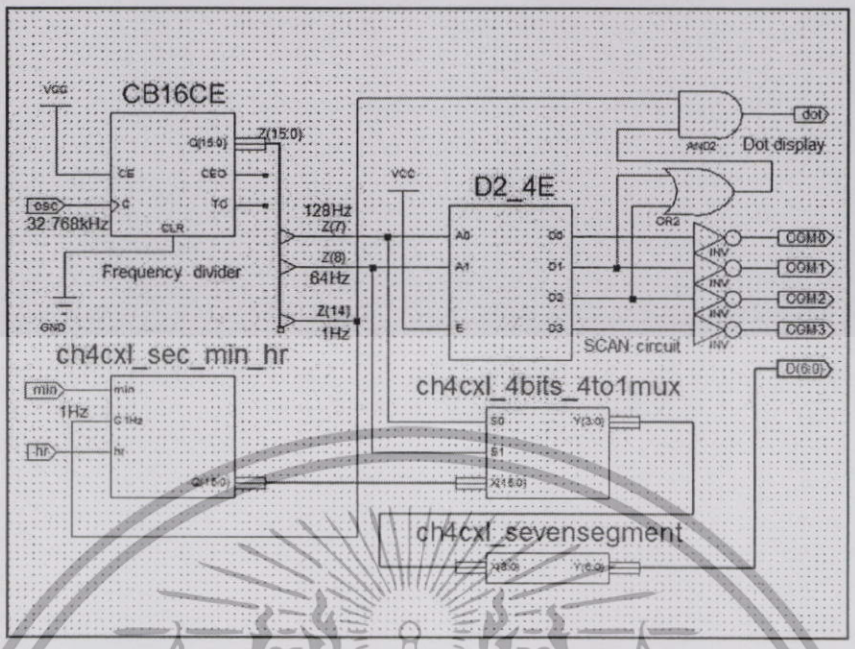
การออกแบบวงจร (Design entry) นั้นสามารถออกแบบได้ด้วยวิธีต่างๆ เช่น การออกแบบด้วยวิธีวาดผังวงจร (Schematic) การออกแบบด้วยการเขียนขั้นตอนการทำงาน (State diagram) และการออกแบบด้วยภาษาระดับสูงที่เรียกว่าเอชดีแอล (Hardware Description Language: HDL) โดยภาษาวีเอชดีแอลนั้น การออกแบบวงจรจะเริ่มจากการเข้าใจอัลกอริทึมการทำงานหรือมีการออกแบบวงจรไว้แล้ว และนำมาเขียนเป็นภาษาอธิบายฮาร์ดแวร์หรือวีเอชดีแอล สำหรับการอธิบายพฤติกรรม (Behavioral) การทำงานของวงจร ดังแสดงในรูปที่ 3.18

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity C100UP is
    Port ( OSC : in  STD_LOGIC;
           PB1 : in  STD_LOGIC;
           CLR : in  STD_LOGIC;
           DOT : out  STD_LOGIC;
           Y : out  STD_LOGIC_VECTOR (6 downto 0);
           COM : out  STD_LOGIC_VECTOR (1 downto 0));
end C100UP;

architecture Behavioral of C100UP is
    signal C,S,C_DB,CLR_DB : STD_LOGIC;
    signal Q_temp : STD_LOGIC_VECTOR (14 downto 0);
    signal Q0,Q1,A : STD_LOGIC_VECTOR (3 downto 0);
begin
-----------------------------------------------------
DB   : process(OSC,PB1,CLR_DB)
       begin
          if CLR_DB='0' then
             C <= '0';
          elsif C_DB'event and C_DB='1' then
             if PB1='0' then
                C <= '1';
             end if;
          end if;
    end process;
    C_DB <= not C and OSC;
-----------------------------------------------------
F_DIV : process (OSC)
       begin
          if OSC'event and OSC='1' then
```

รูปที่ 3.18 การออกแบบวงจรด้วยภาษาวีเอชดีแอล

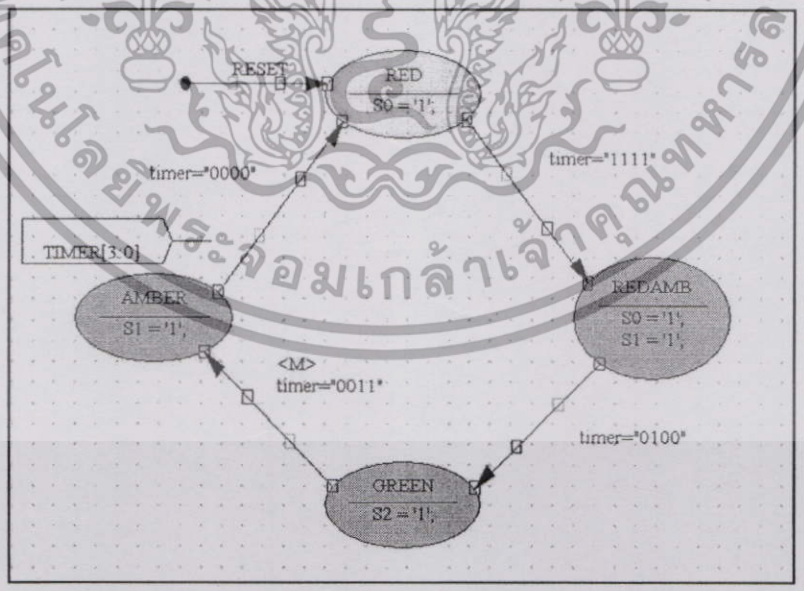
ซึ่งการออกแบบวงจรด้วยภาษาเอชดีแอลจะใช้โปรแกรม Text Editor หรือใช้ซอฟต์แวร์ที่สนับสนุนการเขียนภาษาเอชดีแอล ซึ่งทำให้ง่ายในการเขียนและสามารถตรวจสอบความถูกต้องตามหลักไวยากรณ์ได้ เช่น ModelSim ของบริษัท Model Technology เป็นต้น

การออกแบบวงจรด้วยวิธีการวาดผังวงจรเดียวกันการออกแบบวงจรในลักษณะ ผู้ออกแบบจะต้องเข้าใจหลักการทำงานของวงจรที่จะออกแบบไว้แล้ว จากนั้นจะนำอุปกรณ์ที่มีอยู่ในโปรแกรมมาทำการออกแบบวงจร แต่การออกแบบวงจรในลักษณะนี้จำเป็นจะต้องอ้างอิงเทคโนโลยีของบอร์ดทดลองที่จะใช้งาน เนื่องจากผังวงจรที่นำมาใช้นั้นต้องมีอยู่ในบอร์ดทดลองที่นำมาใช้งาน การออกแบบนั้นคล้ายกับการวาดรูป สามารถวาดผังวงจรได้ดังในรูปที่ 3.19



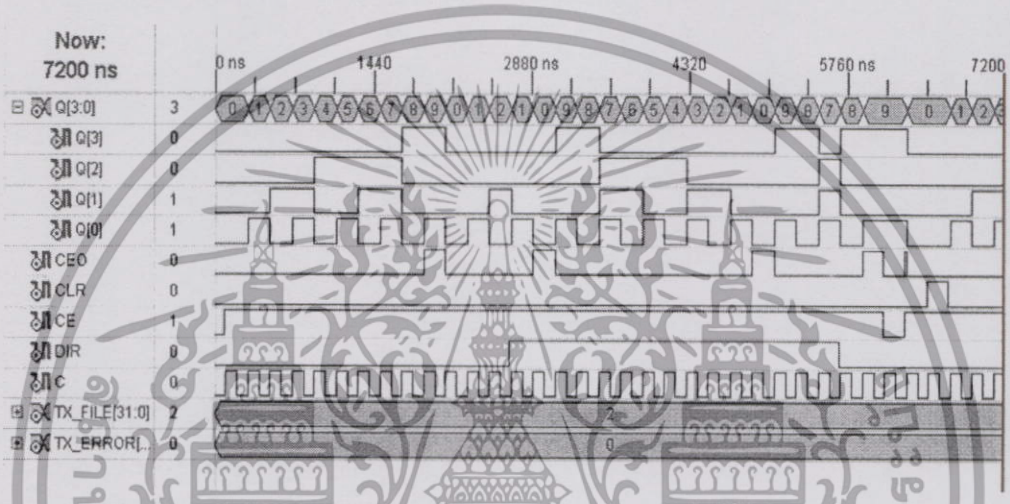
รูปที่ 3.19 การออกแบบวงจรด้วยวิธีการวาดผังวงจร (Schematic)

การออกแบบวงจรด้วยการเขียนขั้นตอนการทำงานเป็นอีกหนึ่งวิธีในการออกแบบวงจร การออกแบบนี้ผู้ออกแบบต้องเข้าใจถึงขั้นตอนการทำงานที่ต้องการออกแบบ การออกแบบแสดงได้ดังในรูปที่ 3.20



รูปที่ 3.20 การออกแบบด้วยการเขียนขั้นตอนการทำงาน (State diagram)

จากการออกแบบวงจร ขั้นตอนต่อไปคือการตรวจสอบว่าวงจรที่ออกด้วยแบบภาษาเอชดีแอลนั้นสามารถทำงานได้ตามที่ออกแบบหรือไม่ โดยสามารถตรวจสอบได้จากการจำลองการทำงานในระดับพฤติกรรม (Behavioral Simulation) ของวงจรแบ่งออกได้ 2 ลักษณะคือการเขียนภาษาเอชดีแอลเพื่อทดสอบวงจร ที่เรียกว่า "Testbench" และการจำลองการทำงานโดยใช้ซอฟต์แวร์ HDL Bencher โดยผู้ใช้สามารถป้อนสัญญาณอินพุต ให้ค่าเป็นไปตามที่ต้องการได้ในรูปแบบที่เรียกว่า Test bench waveform (.tbw) และเรียกใช้งาน ISE simulator เพื่อทดสอบและแสดงผลในรูปแบบ Waveform ดังแสดงในรูปที่ 3.21



รูปที่ 3.21 สัญญาณที่ได้จากการจำลองการทำงานในรูปแบบ Waveform

ซึ่งจากการทดสอบวงจรที่ออกแบบด้วยโปรแกรมซิมมูเลเตอร์ (Simulator) จะได้สัญญาณเอาต์พุตทำให้สามารถตรวจสอบว่าการทำงานของวงจรที่ถูกตรวจสอบ (Design under Test: DUT) ทำงานถูกต้องหรือไม่ ซึ่งถ้าไม่ถูกต้องก็สามารถกลับไปแก้ไขใหม่ได้ ซึ่งการแก้ไขสามารถทำได้ด้วยผู้ออกแบบเอง

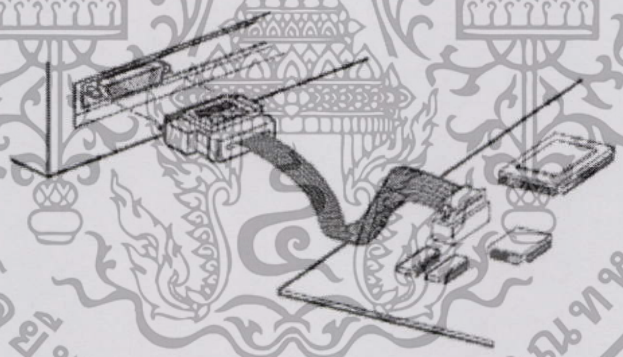
3.6.2 การติดตั้ง

ในขั้นตอนการติดตั้งนี้ จะนำผลที่ได้จากการสังเคราะห์ที่เป็นเน็ตลิสต์ (Design netlist) ที่ได้มาจาก Design Entry ที่เป็น Schematic หรือจากการสังเคราะห์วงจรในกรณีที่เป็น HDL มาทำการออพติไมซ์วงจรในระดับเกต จากนั้นทำการแปลงให้เป็นกลุ่มของลอจิก และทำการตรวจสอบว่าสามารถวางหรือบรรจุลงวงจรลงในชิพเบอร์ที่เรากำหนดได้หรือไม่ และขั้นตอนต่อไปจะทำการเมพ (Map) โดยจะต้องมีการคำนวณหาตำแหน่งที่เหมาะสมเพื่อทำการออกแบบ และเชื่อมเส้นทาง (Place & Route) ในเอฟพีจีเอด้วยซอฟต์แวร์ของผู้ผลิตเอฟพีจีเอ เช่น Xilinx ISE และ Quartus II เป็นต้น ซึ่งโปรแกรมที่ใช้จะแสดงให้เห็นถึงเวลาที่ใช้ในการทำงานของวงจร (Static Timing) และ

ตรวจสอบความถูกต้องจากการจำลองการทำงานระดับฐานเวลา (Timing Simulation) ด้วยโปรแกรมซิมมูเลเตอร์ การจำลองการทำงานระดับฐานเวลาถือว่าเป็นการจำลองการทำงานขั้นตอนสุดท้าย ซึ่งจะต้องทำหลังจากการออกแบบวงจรและเชื่อมต่อเส้นทางการทำงาน การจำลองการทำงานจะเกี่ยวกับเรื่องของเวลา เช่น ความล่าช้าของสัญญาณไฟฟ้าตามเส้นทางต่างๆ ภายในวงจรและระยะคาบของสัญญาณนาฬิกา (Clock Period) น้อยที่สุดที่สามารถใช้งานในการทำงานของวงจรได้ ซึ่งถ้าการทำงานไม่ถูกต้องก็สามารถกลับไปแก้ไขโค้ดภาษาเอชดีแอลใหม่ โดยไม่เสียค่าใช้จ่าย

3.6.3 การโปรแกรมข้อมูลลงชิพเอฟพีจีเอ

จากขั้นตอนการตรวจสอบด้วยการจำลองการทำงานระดับฐานเวลาแล้ว ขั้นตอนต่อไปคือการโปรแกรมข้อมูลวงจรลงชิพเอฟพีจีเอ (Device Programming) การโปรแกรมข้อมูลวงจรจะนำไฟล์ที่นามสกุล .bit ที่ได้จากขั้นตอนก่อนหน้าคือ ขั้นตอน Implementation หรือขั้นตอน Generate Programming File มาทำการดาวโหลดลงบอร์ดทดลองเอฟพีจีเอด้วยเครื่องโปรแกรมหรือดาวน์โหลดเคเบิลก็ได้ เป็นการตรวจสอบความถูกต้องในการทำงานในลำดับสุดท้าย สำหรับการตรวจสอบในวงจรที่นำไปใช้งานจริง (In-Circuit Verification)



รูปที่ 3.22 การโปรแกรมลงในชิพเอฟพีจีเอ

บทที่ 4
การออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก

บทนี้กล่าวถึงการออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำและหลักการทำงาน ชุดเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำ ซึ่งเป็นชุดเข้ารหัสที่ให้ประสิทธิภาพสูง เนื่องจากสามารถเข้ารหัสได้เร็วและให้ความซับซ้อนอยู่ในลักษณะเชิงเส้น ในส่วนแรกของบทนี้กล่าวถึงทฤษฎีและหลักการของวงจรเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก การออกแบบวงจรในลักษณะต่างๆ เช่น วงจรเข้ารหัสในลักษณะอนุกรม วงจรเข้ารหัสในลักษณะกึ่งขนาน และวงจรเข้ารหัสในลักษณะขนาน การออกแบบการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ วงจรที่ออกแบบในแต่ละแบบจะให้ประสิทธิภาพในด้านความเร็ว ในด้านการใช้ทรัพยากรแตกต่างกันขึ้นอยู่กับการนำไปประยุกต์ใช้งานและความเหมาะสม

4.1 ทฤษฎีการออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก

รหัสพาริตี้เช็คความหนาแน่นต่ำโดยทั่วไป สามารถเข้ารหัสได้สองวิธีคือ การเข้ารหัสด้วยเมตริกซ์กำเนิด **G** และเข้ารหัสด้วยเมตริกซ์พาริตี้เช็ค **H** ดังที่ได้กล่าวในบทที่ 2 สำหรับการออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก จัดเป็นการเข้ารหัสด้วยเมตริกซ์พาริตี้เช็ค **H** ซึ่งเมตริกซ์พาริตี้เช็ค **H** ในลักษณะควอไซไซคลิกจะได้รับการสร้างขึ้นในลักษณะของแถวลำดับ (Array) ซึ่งจะมีคุณสมบัติของไซคลิก (Cycle) ทำให้ความซับซ้อนในการเข้ารหัสลดลง การเข้ารหัสของรหัสพาริตี้เช็คที่มีความหนาแน่นต่ำทำได้โดยการนำบิตพาริตี้ต่อกับบิตข้อมูลข่าวสาร **(p|m)** จะเห็นว่าการนำบิตข้อมูลข่าวสาร **(m)** มาทำการต่อกับบิตพาริตี้ไม่ทำให้เกิดความซับซ้อน แต่ความความซับซ้อนจะอยู่ที่การนำข้อมูลข่าวสารมาทำการหาค่าบิตพิเศษหรือพาริตี้ **(p)** นั่นเอง และวิธีที่ทำให้ความซับซ้อนในการหาค่าบิตพาริตี้ลดลงได้คือการสร้างเมตริกซ์พาริตี้เช็คในลักษณะแถวลำดับ ซึ่งทำให้เมตริกซ์ที่สร้างขึ้นมีคุณสมบัติคือ เมื่อทำการเลื่อนบิตข้อมูลของคำรหัส **(c)** แบบวนกลับ ก็จะได้พาริตี้ **(p)** ที่เลื่อนบิตแบบวนกลับเช่นกัน

4.1.1 รหัสพาริตี้เช็คความหนาแน่นต่ำประเภทอาร์เรย์

รหัสพาริตี้เช็คความหนาแน่นต่ำประเภทอาร์เรย์หรือรหัสคิวซีแอลดีพีซี (QC – LDPC) [10] ได้รับการกล่าวถึงการเข้ารหัสและถอดรหัสในหลายวิธี เช่น การปรับปรุงเมตริกซ์ย่อย (Sub Matrix) เพื่อใช้สำหรับสร้างเมตริกซ์พาริตี้เช็ค **H** ที่รองรับอัตรารหัสสูง (Hi Rate) และเมื่อเปลี่ยนแปลงข้อมูลข่าวสารให้มีปริมาณเพิ่มขึ้น ความซับซ้อนในการเข้ารหัสยังคงอยู่ในลักษณะเชิงเส้นไม่เปลี่ยนแปลง [11]

การสร้างเมตริกซ์พาริตี้เช็ค **H** ในลักษณะแถวลำดับนั้น สามารถสร้างได้จากเมตริกซ์สลับตำแหน่ง (***P***) ที่มีมิติขนาด LxL และนำไปขยายเพื่อสร้างเมตริกซ์พาริตี้เช็ค **H**

$$P_{LxL} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & & 1 \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix} \tag{4.1}$$

เมตริกซ์พาริตี้เช็ค **H** สามารถสร้างจากสมการที่ 4.1 ซึ่งวิธีการสร้างนั้นทำได้โดยการนำเมตริกซ์สลับตำแหน่งขยายออกในด้านแถวและด้านหลักไปจำนวน j และ k เท่า สำหรับเมตริกซ์พาริตี้เช็ค **H** มีมิติขนาด $(n\text{-}k) \times n$

4.1.2 รหัสพาริตี้เช็คความหนาแน่นต่ำแบบอาร์เรย์คุณสมบัติของ Cycle

เมตริกซ์พาริตี้เช็คสำหรับรหัสอาร์เรย์ (Array Code) อธิบายได้ด้วยพารามิเตอร์ 3 ค่าได้แก่ จำนวนเฉพาะ L และจำนวนเต็ม j และ k โดยที่เมตริกซ์นี้มีขนาด $jLxkL$

$$\mathbf{H} = \begin{bmatrix} I & I & I & \dots & I \\ I & P & P^2 & \dots & P^{k-1} \\ I & P^2 & P^4 & \dots & P^{2(k-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I & P^{j-1} & P^{2(j-1)} & \dots & P^{(j-1)(k-1)} \end{bmatrix} \tag{4.2}$$

โดยที่ ***I*** คือเมตริกซ์เอกลักษณ์และ ***P*** คือเมตริกซ์สลับตำแหน่งขนาด LxL ซึ่งเกิดจากการเลื่อนทางซ้ายหรือขวาของ ***I*** กำลังของ **H** ในเมตริกซ์ คือจำนวนครั้งของการเลื่อน เพื่อให้การเข้ารหัสมี

ความซับซ้อนในระดับเชิงเส้นของรหัสพาริตี้เช็คความหนาแน่นต่ำข้างต้น ทำได้โดยการแปลงให้มีรูปร่างสามเหลี่ยมด้านล่างดังสมการที่ 4.3

$$\mathbf{H}(q,j,k)=\begin{bmatrix} I & I & I & \dots & I & \dots & I \\ 0 & I & P & \dots & P^{(j-2)} & \dots & P^{(k-2)} \\ 0 & 0 & I & \dots & P^{2(j-3)} & \dots & P^{2(k-3)} \\ \vdots & & & & \ddots & & \vdots \\ 0 & \dots & 0 & I & P^{(j-2)} & \dots & P^{(j-2)(k-j-1)} \\ 0 & 0 & \dots & 0 & I & \dots & P^{(j-1)(k-j)} \end{bmatrix} \tag{4.3}$$

j $\quad$ $k\text{-}j$

ซึ่งการขยายเมตริกซ์เมตริกซ์สลับตำแหน่งมีข้อจำกัดคือ การขยายในด้านแถวต้องขยายน้อยกว่าในด้านหลักและการขยายในด้านหลักต้องน้อยกว่าขนาดของเมตริกซ์ $\boldsymbol{P}$ คือ L แสดงได้ดังนี้ $q \le k \le j$ โดยอัตรารหัส (Code Rate) หาได้จากสมการที่ 4.4

$$R=\frac{k-j}{k}=1-\frac{j}{k} \tag{4.4}$$

การสร้างเมตริกซ์พาริตี้เช็ค $\mathbf{H}$ โดยใช้เมตริกซ์สลับตำแหน่ง ทำให้เมตริกซ์ที่สร้างขึ้นมีคุณสมบัติของควอไซไซคลิก (Quasi-Cyclic) ทำให้ความซับซ้อนในการออกแบบวงจรลดลง สามารถเข้ารหัสได้โดยที่กำหนดให้คำรหัสมีมิติเท่ากับ $1\text{x}n$ เมตริกซ์ $\mathbf{0}$ มีมิติขนาด $1\text{x}(n\text{-}k)$ และใช้เทคนิคการบวกแบบมอดูโล-2 หรือการทำ XOR จากความสัมพันธ์ของคำรหัสและเมตริกซ์พาริตี้เช็ค $\mathbf{H}$ แสดงได้ดังสมการที่ 4.5

$$\mathbf{cH}^T=\mathbf{0} \tag{4.5}$$

เมื่อทรานสโพส $\mathbf{cH}^T=\mathbf{0}$ จะได้

$$\mathbf{Hc}^T=\mathbf{0}^T \tag{4.6}$$

ซึ่งกำหนดให้คำรหัส (Codeword) คือ $\mathbf{c} = [p_1\ p_2\ p_3\ \ldots\ p_{n-k} \mid m_1\ m_2\ m_3\ \ldots\ m_k] = [\mathbf{p} \mid \mathbf{m}]$ จากนั้นทำการจัดรูปสมการในการหาค่าพาริตี้เช็ค (p) ได้จากสมการที่ 4.6 การหาค่าพาริตี้ (p) แสดงได้ดังนี้

$$\mathbf{c}^T = \begin{bmatrix} p_{n-k} \\ \hline m_k \end{bmatrix} \tag{4.7}$$

ตัวอย่างที่ 4.1 การออกแบบวงจรเข้ารหัสแบบ XOR Network

กำหนดให้เมตริกซ์พาริตี้เช็ค **H** คือ

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

ความสัมพันธ์ของของเมตริกซ์พาริตี้เช็ค **H** และคำรหัส **c** คือ $\mathbf{Hc}^T = \mathbf{0}^T$ เมื่อทำการแทนค่าตัวแปลจะได้ดังนี้

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \hline m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \end{bmatrix}$$

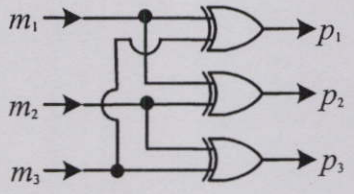
ทำการคูณเมตริกซ์พาริตี้เช็ค **H** ด้วยเวกเตอร์คำรหัสทรานสโพสจะได้สมการการหาค่าบิตพาริตี้ออกมาคือ

$$p_1 + m_1 + m_3 = 0$$
$$p_2 + m_2 + m_1 = 0$$
$$p_3 + m_3 + m_2 = 0$$

โดยการบวกในข้างต้นเป็นการบวกแบบมอดูโล-2 หรือการทำ XOR ดังนั้นการย้ายข้างของสมการจึงไม่ติดค่าลบ ทำให้ได้สมการการหาค่าพาริตี้บิต ดังนี้

$$p_1 = m_1 + m_3$$
$$p_2 = m_2 + m_1$$
$$p_3 = m_3 + m_2$$

จากสมการข้างต้น สามารถนำไปออกแบบเป็นวงจรการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำ ซึ่งแสดงดังรูปที่ 4.1



รูปที่ 4.1 วงจรการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบ XOR Network

การเข้ารหัสข้างต้น มีจำนวนอินพุตที่รับเข้ามาในจำนวนน้อย แต่ในระบบการใช้งานจริงจะมีจำนวนอินพุตที่รับเข้ามาเท่ากับ 2^k และขนาดของอินพุต k มีค่าสูง เช่น $k = 512,\ 1024$ ทำให้มีความซับซ้อน วิธีดังกล่าวจึงไม่เหมาะสมสำหรับการนำไปออกแบบฮาร์ดแวร์ แต่ในกรณีของวงจรเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำลักษณะอนุกรมและลักษณะกึ่งขนานที่ออกแบบ ใช้สัญญาณความถี่ควบคุมการทำงาน และเมื่อขนาดของอินพุต k มีค่าสูง ความต้องการทรัพยากรจะเพิ่มขึ้นอย่างเป็นเชิงเส้น ซึ่งเป็นวิธีที่เหมาะสมสำหรับระบบที่ต้องเข้ารหัสบล็อกข้อมูลขนาดใหญ่

4.2 การออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก

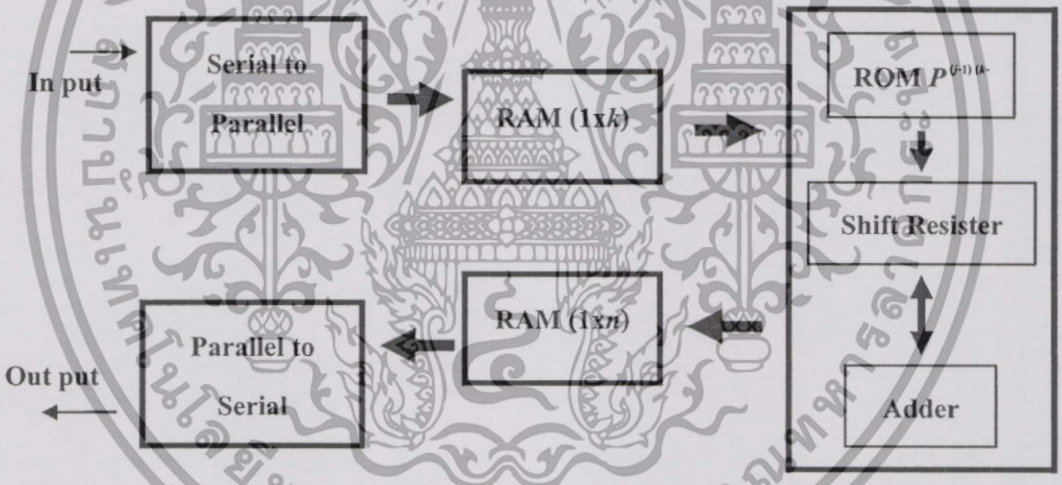
การสร้างเมตริกซ์พาริตี้เช็ค **H** สามารถทำได้โดยการใช้เมตริกซ์สลับตำแหน่ง P ซึ่งจะทำให้เมตริกซ์พาริตี้เช็ค **H** ที่สร้างขึ้นมีคุณสมบัติของควอไซไซคลิก (Quasi-Cyclic) [12] คุณสมบัติของควอไซไซคลิก คือจะทำให้เกิดความสัมพันธ์ระหว่างข้อมูลข่าวสาร (**m**) กับบิตพิเศษหรือพาริตี้เช็ค (**p**) นั่นคือเมื่อทำการเลื่อนบิตคำรหัส (**c**) ในแต่ละบล็อกข้อมูลย่อยแบบวนกลับหนึ่งครั้ง จะเกิดคำรหัสใหม่ และสมการที่ 4.8 เป็นการหาค่าพาริตี้เช็ค โดยใช้ความสัมพันธ์ของ $\mathbf{Hc}^T = \mathbf{0}$

$$\mathbf{Hc}^T = \begin{bmatrix} I & I & I & \cdots & I & \cdots & I \\ 0 & I & P & \cdots & P^{(j-2)} & \cdots & P^{(k-2)} \\ 0 & 0 & I & \cdots & P^{2(j-3)} & \cdots & P^{2(k-3)} \\ & & & & \ddots & & \\ 0 & \cdots & 0 & I & P^{(j-2)} & \cdots & P^{(j-2)(k-j-1)} \\ 0 & 0 & \cdots & 0 & I & \cdots & P^{(j-1)(k-j)} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_j \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_{k-j} \end{bmatrix} = \mathbf{0} \qquad (4.8)$$

จากสมการที่ 4.8 เมื่อได้รับการคำนวณจะอยู่ในรูปตามสมการที่ 4.9 ซึ่งจะได้ค่าพาริตี้ออกมานั่นคือ $\mathbf{p}_j, \mathbf{p}_{(j-1)}, \mathbf{p}_{(j-2)}, \ldots \mathbf{p}_l$ จากนั้นจึงนำไปออกแบบวงจรการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำ

$$
\begin{aligned}
\mathbf{p}_j &= \mathbf{m}_1 P^{(j-1)} + \mathbf{m}_2 P^{(j-1)2} + \mathbf{m}_3 P^{(j-1)3} + \ldots + \mathbf{m}_{k-j} P^{(j-1)(k-j)} \\
\mathbf{p}_{j-1} &= \mathbf{p}_j P^{(j-2)} + \mathbf{m}_1 P^{(j-2)2} + \mathbf{m}_2 P^{(j-2)3} + \ldots + \mathbf{m}_{k-j} P^{(j-2)(k-j+1)} \\
\mathbf{p}_{j-2} &= \mathbf{p}_{j-1} P^{(j-3)} + \mathbf{p}_j P^{(j-3)2} + \mathbf{m}_1 P^{(j-3)3} + \ldots + \mathbf{m}_{k-j} P^{(j-3)(k-j+2)} \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
\mathbf{p}_l &= \mathbf{p}_{l+1} P^{(l-1)} + \mathbf{p}_{l+2} P^{(l-1)2} + \ldots + \mathbf{p}_j P^{(l-1)(j-l)} \ldots + \mathbf{m}_{k-j} P^{(l-1)(k-l)}
\end{aligned}
\tag{4.9}
$$

จากสมาการที่ 4.9 สามารถนำมาออกแบบวงจรได้เมื่อ $\mathbf{m}$ คือข้อมูลข่าวสาร $P^{(j-1)(k-j)}$ คือจำนวนการเลื่อนบิตข้อมูลข่าวสาร $\mathbf{p}_j$ คือบิตพิเศษหรือพาริตี้ ในการออกแบบวงจรเข้ารหัสนั้นจะกำหนดให้การบวกเป็นการบวกแบบมอดูโล 2 และถ้าดังของเมตริกซ์สลับตำแหน่ง P เป็นจำนวนครั้งในการเลื่อนบิตแบบวนกลับ



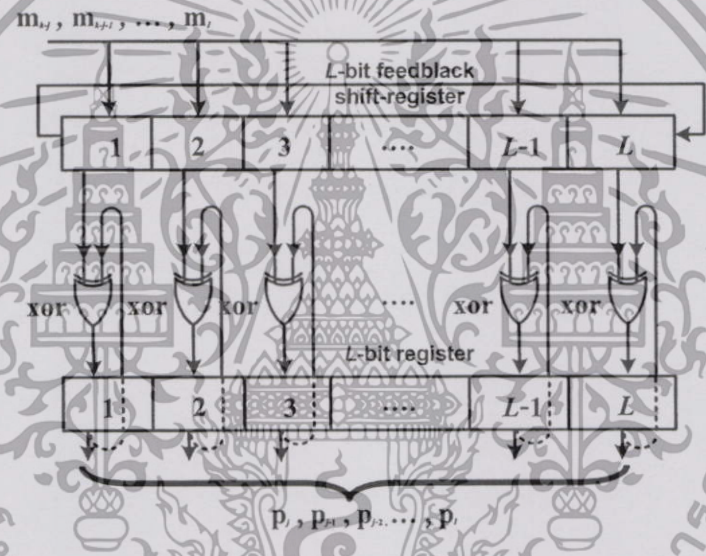
รูปที่ 4.2 บล็อกไดอะแกรมของชุดเข้ารหัส QC-LDPC

จากรูปที่ 4.2 เป็นการแสดงให้เห็นถึงโครงสร้างวิธีการออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำ โดยเริ่มตั้งแต่การรับข้อมูล (Input) ที่ต้องการเข้ารหัส จากนั้นทำการเปลี่ยนข้อมูลจากอนุกรมมาเป็นขนาน (Serial to Parallel) ข้อมูลจะถูกเก็บในหน่วยความจำชั่วคราว (RAM) โดยที่ในแต่ละที่อยู่ (Address) จะมีขนาด 1xk ต่อจากนั้นค่าที่ได้จะถูกส่งต่อไปที่ตัวดำเนินการเลื่อนแบบวนกลับ เมื่อทำการเลื่อนบิตแบบวนกลับ และบวกกับข้อมูลที่อยู่ก่อนหน้าครบทุกจำนวนแล้ว บิตข้อมูลจะถูกเก็บในหน่วยความจำชั่วคราว (RAM) ที่มีขนาด 1xn จากนั้นส่งบิตข้อมูลต่อไปเพื่อ

เปลี่ยนบิตข้อมูลจากขนานไปเป็นอนุกรม (Parallel to Serial) และเปลี่ยนระดับสัญญาณเพื่อส่งออกไปในช่องสัญญาณ

4.2.1 วงจรเข้ารหัสแบบอนุกรม

วงจรการเข้ารหัสแบบอนุกรม เป็นการนำบิตข้อมูลข่าวสาร (Message) มาทำการแบ่งออกในลักษณะบล็อกย่อย เพื่อใช้สำหรับการเข้ารหัสจำนวนครั้งละบล็อกย่อย (Sub block) จากนั้นก็จะทำตามขั้นตอนการเข้ารหัส เริ่มต้นข้อมูลจะได้รับการอ่านเข้าในตัวดำเนินการเลื่อน (Shift Register) และทำการบวกแบบมอดูโล 2 โดยค่าที่ได้รับจะถูกเก็บในรีจีสเตอร์ (Register) เพื่อรอการอ่านออกมาบวกแบบมอดูโล 2 ในครั้งต่อไป ดังแสดงในรูปที่ 4.3 โดยมีขั้นตอนการเข้ารหัสแบบอนุกรม ดังนี้



รูปที่ 4.3 วงจรการเข้ารหัสแบบอนุกรม

ขั้นตอนที่ 1 เมื่อเริ่มต้น ข้อมูลข่าวสาร (Message) ที่รับเข้ามาในครั้งแรก บล็อกข้อมูลเดิมจะมีขนาดเท่ากับ 1xk จากนั้นทำการแบ่งข้อมูลออกให้อยู่ในลักษณะบล็อกย่อยที่มีขนาดเท่ากับ 1xL ซึ่งจะมีจำนวนทั้งหมด $k-j$ บล็อก

ขั้นตอนที่ 2 นำข้อมูลในบล็อกแรก คือ $\mathbf{m}_{(k-j)}$ มาทำการเลื่อนบิตแบบวนกลับ จำนวนทั้งหมด $P^{(j-1)(k-j)}$ ครั้ง

ขั้นตอนที่ 3 นำข้อมูลที่ได้รับการเลื่อนบิตแบบวนกลับมาทำการ XOR กับข้อมูลเดิมที่อยู่ก่อนหน้าและค่าที่ได้จะค้างอยู่ใน หน่วยความจำ

ขั้นตอนที่ 4 นำข้อมูลชุดต่อไป คือ $\mathbf{m}_{(k-j-1)}$ มาทำการเลื่อนบิตแบบวนกลับจำนวน $P^{(j-1)3}$ ครั้ง

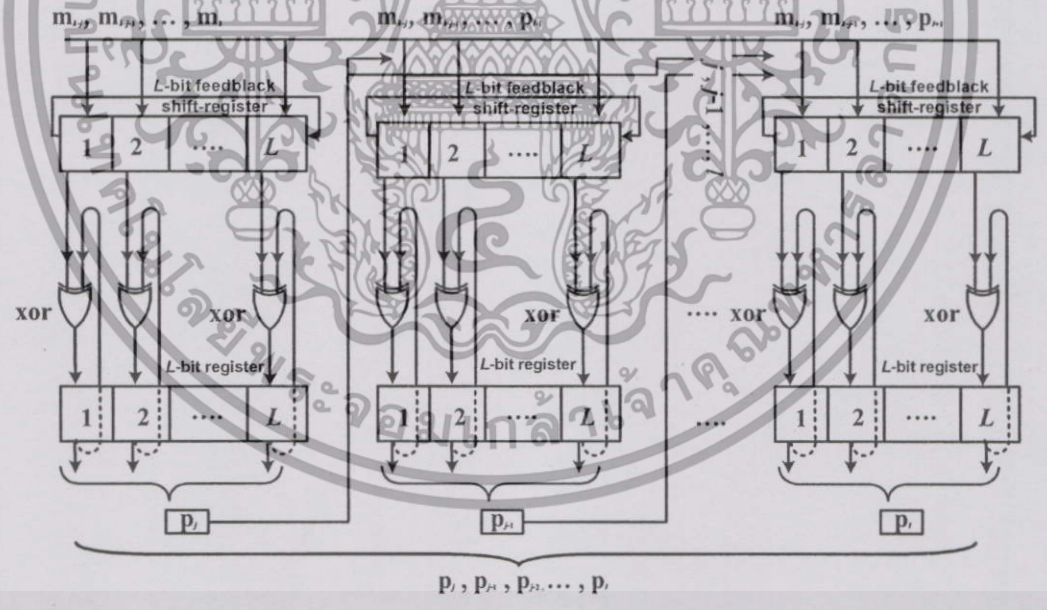
ขั้นตอนที่ 5 ทำการ XOR กับข้อมูลที่อยู่ก่อนหน้า คือ $\mathbf{m}_{(k-j)}$ ที่ถูก เลื่อนบิตแบบวนกลับจำนวน $P^{(j-1)(k-j)}$ ครั้ง

ขั้นตอนที่ 6 ทำขั้นตอนที่ 1 ถึง 5 จนครบทุกค่าของบล็อกข้อมูลย่อย คือ $\mathbf{m}_{(k-j)}, \mathbf{m}_{k-j-1}, \ldots, \mathbf{m}_1$ บล็อก ซึ่งจะได้ค่าพาริตี้ คือ $\mathbf{p}_j$ ออกมาในบล็อกแรก

ขั้นตอนที่ 7 ทำตามขั้นที่ 1-5 และค่าจำนวนของการ เลื่อนบิตแบบวนกลับ จะเปลี่ยนไปตามสมการที่ 4.6 คือ $P^{(j-2)(k-j+1)}$ ไปจนถึง $P^{(j-2)2}$ จากนั้นนำข้อมูลที่ได้มา XOR กับข้อมูลที่อยู่ก่อนหน้าก็จะได้ค่าพาริตี้ครบทุกค่า

4.2.2 วงจรเข้ารหัสแบบกึ่งขนาน

วงจรเข้ารหัสแบบกึ่งขนาน วิธีนี้สามารถเข้ารหัสในแต่ละบล็อกข้อมูลในการหาค่า $\mathbf{p}_j, \mathbf{p}_{j-1}, \ldots, \mathbf{p}_l$ ในเวลาที่ใกล้เคียงกัน ซึ่งมีความเร็วในการเข้ารหัสมากกว่าการเข้ารหัสด้วยวงจรแบบอนุกรมประมาณ j เท่า เริ่มต้นข้อมูลจะได้รับการอ่านเข้าในตัวดำเนินการเลื่อน (Shift Register) พร้อมกันทุกบล็อกย่อย จากนั้นทำการบวกแบบมอดูโล 2 และค่าที่ได้รับจะถูกเก็บในรีจีสเตอร์ (Register) ของบล็อกย่อยเพื่อรอการอ่านออกมาบวกแบบมอดูโล 2 ในครั้งต่อไปของแต่ละบล็อกย่อย ดังแสดงในรูปที่ 4.4 และมีขั้นตอนในการเข้ารหัสแบบกึ่งขนาน ดังนี้



รูปที่ 4.4 วงจรการเข้ารหัสแบบกึ่งขนาน

ขั้นตอนที่ 1 เมื่อเริ่มต้นรับข้อมูลข่าวสาร (Message) เข้ามา ข้อมูลจะมีขนาด 1xk จากนั้นทำการแบ่งข้อมูลออกเป็นบล็อกย่อยขนาด 1xL จำนวน $k-j$ บล็อก และสามารถหาค่า $\mathbf{p}_j, \mathbf{p}_{j-1}, \ldots, \mathbf{p}_l$ ที่เวลาใกล้เคียงกัน

ขั้นตอนที่ 2 ในบล็อกแรกจะทำการหาค่า $\mathbf{p}_j$ โดยจะนำข้อมูลในบล็อกที่ $\mathbf{m}_{k-j}$ มาทำการเลื่อนบิตแบบวนกลับ จำนวน $P^{(j-1)(k-j)}$ ครั้งและในบล็อกที่ 2 ในการหาค่า $\mathbf{p}_{j-1}$ จะทำพร้อมกัน คือ $\mathbf{m}_{k-j}$ ก็จะถูกทำการเลื่อนบิตแบบวนกลับ จำนวน $P^{(j-2)(k-j+1)}$ ครั้ง ทำในลักษณะนี้พร้อมกันจนครบทุกบล็อกไปถึงการหาค่า $\mathbf{p}_l$ นั่นคือ นำข้อมูล $\mathbf{m}_{(k-j)}$ มาทำการ เลื่อนบิตแบบวนกลับ จำนวน $P^{(l-1)(k-l)}$ ครั้ง

ขั้นตอนที่ 3 ข้อมูลในแต่ละบล็อกที่ทำการเลื่อนบิตแบบวนกลับจะนำมาทำการ XOR กับข้อมูลที่อยู่ก่อนหน้าพร้อมกันในแต่ละบล็อก ค่าที่ได้จะถูกเก็บไว้ใน หน่วยความจำ

ขั้นตอนที่ 4 นำค่า $\mathbf{m}_{k-j-1}$ มาทำการ เลื่อนบิตแบบวนกลับ ตามจำนวนการ เลื่อนบิตแบบวนกลับ ของแต่ละบล็อก จากนั้นนำค่าที่ได้ XOR กับข้อมูลที่อยู่ก่อนหน้า คือ $\mathbf{m}_{(k-j)}$ ที่ค้างอยู่ในหน่วยความจำ ทำในลักษณะนี้ทุกบล็อก ตั้งแต่ $\mathbf{m}_{(k-j)}$, $\mathbf{m}_{(k-j-1)}$, … , $\mathbf{m}_1$ ก็จะได้ค่า $\mathbf{p}_l$ ออกมา

ขั้นตอนที่ 5 จากขั้นตอนที่ 4 จะได้ค่า $\mathbf{p}_l$ ออกมา จากนั้นทำการส่งค่า $\mathbf{p}_l$ ให้แต่ละบล็อกข้อมูลที่ทำการหาค่า $\mathbf{p}_{j-1}$, $\mathbf{p}_{j-2}$, … , $\mathbf{p}_l$ จากนั้นนำค่า $\mathbf{p}_l$ ไปทำการ เลื่อนบิตแบบวนกลับ ตามจำนวนการ เลื่อนบิตแบบวนกลับ ของแต่ละบล็อกและทำการ XOR กับข้อมูลที่อยู่ก่อนหน้าก็จะได้ค่า $\mathbf{p}_{j-2}$ ออกมา

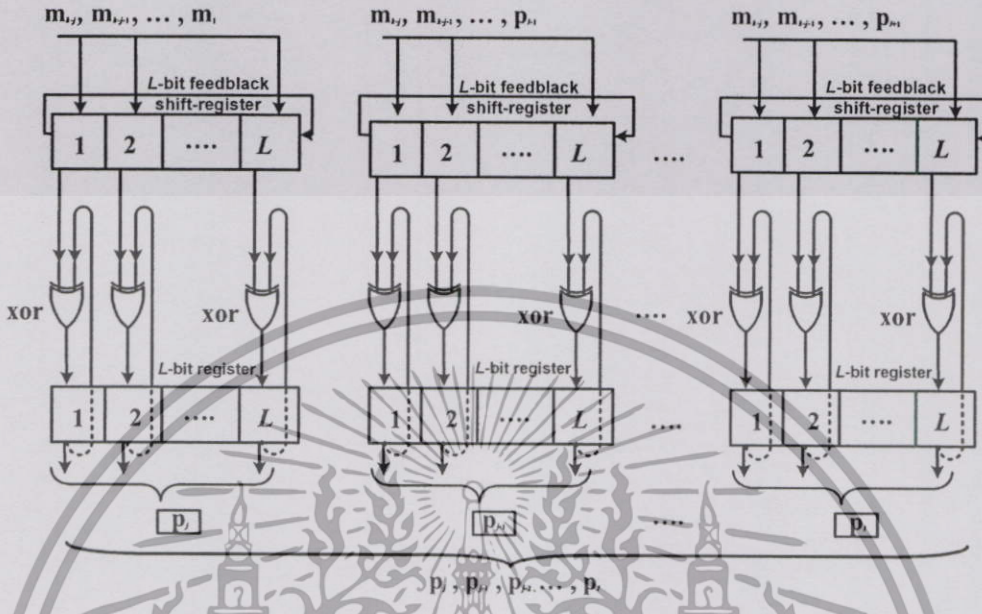
ขั้นตอนที่ 6 ทำการส่งข้อมูล $\mathbf{p}_{j-1}$ ไปในแต่ละบล็อกที่ทำการหาค่า $\mathbf{p}_{j-2}$, $\mathbf{p}_{j-3}$, …, $\mathbf{p}_l$ จากนั้นนำค่า $\mathbf{p}_{j-2}$ ไปทำการ เลื่อนบิตแบบวนกลับ ของแต่ละบล็อกและทำการ XOR กับข้อมูลที่อยู่ก่อนหน้าก็จะได้ค่า $\mathbf{p}_{j-3}$ ออกมา

ขั้นตอนที่ 7 ทำลักษณะแบบเดียวกันจนถึงบล็อกสุดท้าย คือ การหาค่า $\mathbf{p}_l$ ซึ่งจะนำค่าของ $\mathbf{p}_{l+1}$ มาทำการ เลื่อนบิตแบบวนกลับ จำนวน $P^{(l-1)}$ และนำค่าที่ได้ไป XOR กับข้อมูลที่อยู่ก่อนหน้า คือ $\mathbf{p}_{l+2}$ ที่ถูก เลื่อนบิตแบบวนกลับ $P^{(l+1)2}$ ครั้ง ก็จะได้พาริตี้ $\mathbf{p}_j$, $\mathbf{p}_{j-1}$, $\mathbf{p}_{j-2}$, … , $\mathbf{p}_l$ ครบทุกค่า การทำงานในลักษณะนี้เหมาะสำหรับการเข้ารหัสที่มีอัตรารหัสสูง (High Rate) เพราะเมื่ออัตราสูง การส่งค่าจาก $\mathbf{p}_j$ ไปยัง $\mathbf{p}_{j-1}$, $\mathbf{p}_{j-2}$, … , $\mathbf{p}_l$ ที่จะทำการน้อยครั้ง จึงทำให้ได้ค่าบิตพาริตี้ออกมาเร็วกว่าการทำงานในลักษณะอนุกรมและมีการใช้ทรัพยากรน้อยกว่าการเข้ารหัสแบบ XOR Network

4.2.3 วงจรเข้ารหัสแบบขนาน

วงจรเข้ารหัสแบบขนาน วิธีนี้สามารถเข้ารหัสในแต่ละบล็อกข้อมูลในการหาค่า $\mathbf{p}_j$, $\mathbf{p}_{j-1}$, $\mathbf{p}_{j-2}$, … , $\mathbf{p}_l$ ในเวลาเดียวกัน โดยมีความเร็วในการเข้ารหัสมากกว่าวงจรแบบอนุกรมประมาณ j เท่า และมีความซับซ้อนน้อยกว่าการเข้ารหัสแบบกึ่งขนาน สิ่งที่ทำให้การเข้ารหัสแบบขนานมีความซับซ้อนต่ำนั่นคือการลดรูปสมการในการหาค่าบิตพิเศษดังแสดงในตัวอย่างที่ 2 โดยจะทำให้การหาค่าบิตพิเศษจะหาจากข้อมูลข่าวสารเพียงอย่างเดียว ซึ่งต่างจากวงจรเข้ารหัสแบบกึ่งขนานและแบบอนุกรมต้องรอค่าบิตพาริตี้ก่อนหน้า โดยเริ่มต้นข้อมูลจะได้รับการอ่านเข้าในตัวดำเนินการเลื่อน (Shift Register) พร้อมกันทุกบล็อกย่อย จากนั้นทำการบวกแบบมอดูโล 2 ค่าที่ได้รับการคำนวณจะถูกเก็บในรีจีสเตอร์ (Register) ของบล็อกย่อยเพื่อรอการอ่านออกมาบวกแบบมอดูโล 2 ในครั้ง

ต่อไปของแต่ละบล็อกย่อย ดังแสดงในรูปที่ 4.5 ซึ่งมีตัวอย่างในการลดรูปสมการ และขั้นตอนในการเข้ารหัสแบบขนาน ดังนี้



รูปที่ 4.5 วงจรการเข้ารหัสแบบขนาน

ขั้นตอนที่ 1 เมื่อเริ่มต้นรับข้อมูล (Message) เข้ามา ข้อมูลจะมีขนาด 1xk จากนั้นทำการแบ่งข้อมูลออกเป็นขนาด 1xL และมีจำนวนบล็อกย่อย k-j บล็อก โดยจะทำการหาค่า $\mathbf{p}_j$, $\mathbf{p}_{j-1}$, $\mathbf{p}_{j-2}$, … , $\mathbf{p}_1$ ในเวลาเดียวกัน

ขั้นตอนที่ 2 ในบล็อกแรกจะทำการหาค่า $\mathbf{p}_j$ และนำข้อมูลในบล็อกที่ $\mathbf{m}_{(k-j)}$ มาทำการเลื่อนบิตแบบวนกลับจำนวน $P^{(j-1)\,(k-j)}$ ครั้งและในบล็อกที่ 2 ในการหาค่า $\mathbf{p}_{j-1}$ จะทำพร้อมกัน คือ $\mathbf{m}_{(k-j)}$ ก็จะถูกทำการเลื่อนบิตแบบวนกลับ จำนวน $P^{(j-2)\,(k-j+1)}$ ครั้ง ทำในลักษณะนี้พร้อมกันจนครบทุกบล็อกไปถึงการหาค่า $\mathbf{p}_1$ นั่นคือ นำข้อมูล $\mathbf{m}_{(k-j)}$ มาทำการ เลื่อนบิตแบบวนกลับ จำนวน $P^{(l-1)\,(k-l)}$ ครั้ง

ขั้นตอนที่ 3 ข้อมูลในแต่ละบล็อกที่ทำการเลื่อนบิตแบบวนกลับจะนำมาทำการ XOR กับข้อมูลที่อยู่ก่อนหน้าพร้อมกันในแต่ละบล็อกเข้ารหัส ค่าที่ได้จะถูกเก็บไว้ใน หน่วยความจำ

ขั้นตอนที่ 4 นำค่า $\mathbf{m}_{(k-j-1)}$ มาทำการ เลื่อนบิตแบบวนกลับ ตามจำนวนการ เลื่อนบิตแบบวนกลับ แต่ละบล็อก จากนั้นนำไป XOR กับข้อมูลที่อยู่ก่อนหน้า คือ $\mathbf{m}_{(k-j)}$ ที่ค้างอยู่ในหน่วยความจำ ทำในลักษณะนี้ทุกบล็อก ตั้งแต่ $\mathbf{m}_{(k-j)}$, $\mathbf{m}_{(k-j-1)}$, … , $\mathbf{m}_1$ ก็จะได้ค่า $\mathbf{p}_j$, $\mathbf{p}_{j-1}$, $\mathbf{p}_{j-2}$,… , $\mathbf{p}_1$ ครบทุกค่าออกมา ทำลักษณะแบบเดียวกันจนถึงบล็อกสุดท้าย จึงทำให้ได้ค่าบิตพาริตี้ออกมาเร็วกว่าการทำงานในลักษณะอนุกรมและมีจำนวนการใช้ทรัพยากรน้อยกว่าการเข้ารหัสแบบ XOR Network

ตัวอย่างที่ 2 สร้างสมการเพื่อใช้สำหรับออกแบบวงจรเข้ารหัสในลักษณะขนาน

โดยใช้เมตริกซ์สลับที่ (Permutation Matrix) ขนาด 5x5 และขยายในด้านหลัก 5 เท่าและขยายทางด้านแถว 3 เท่า ในการสร้างเมตริกซ์พาริตี้เช็ค H ที่อัตรารหัส 2/5

$L = 5, j = 3, k = 5, R = 2/5$

$$P^0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, P^1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}, P^2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \ldots, P^5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

เมตริกซ์พาริตี้เช็ค $\mathbf{H}$ ได้สร้างจากเมตริกซ์เอกลักษณ์และเมตริกซ์สลับที่ดังสมการที่ 4.10 คุณสมบัติของเมตริกซ์สลับที่ยกกำลังศูนย์นั้น จะเท่ากับคุณสมบัติของเมตริกซ์เอกลักษณ์นั่นคือ $P^0 = I$

$$\mathbf{H} = \begin{bmatrix} I & I & I & I & I \\ 0 & I & P^1 & P^2 & P^3 \\ 0 & 0 & I & P^2 & P^4 \end{bmatrix} \tag{4.10}$$

ความสัมพันธ์ของ $\mathbf{cH}^T = \mathbf{0}$ เมื่อทำการทรานโพสสมการจะได้ $\mathbf{Hc}^T = \mathbf{0}^T$ จากนั้นแทนค่าเมตริกซ์สลับที่และเมตริกซ์เอกลักษณ์ หาผลลัพธ์ที่ได้จากการคูณของเมตริกซ์พาริตี้เช็ค $\mathbf{H}$ กับคำรหัส $\mathbf{c}$ ได้ดังสมการที่ 4.11

$$\begin{bmatrix} I & I & I & I & I \\ 0 & I & P^1 & P^2 & P^3 \\ 0 & 0 & I & P^2 & P^4 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ m_1 \\ m_2 \end{bmatrix} = \begin{matrix} Ip_1 + Ip_2 + Ip_3 + Im_1 + Im_2 \\ Ip_2 + P^1 p_3 + P^2 m_1 + P^3 m_2 \\ Ip_3 + P^2 m_1 + P^4 m_2 \end{matrix} \tag{4.11}$$

จากสมการที่ 4.11 จะได้ผลลัพธ์คือ สมการการหาค่าบิตพาริตี้เท่ากับเมตริกซ์ศูนย์ทรานโพส ซึ่งในสมการการหาค่าพาริตี้ประกอบไปเมตริกซ์เอกลักษณ์คูณกับข้อมูลข่าวสาร Im_i และเมตริกซ์สลับที่คูณกับข้อมูลข่าวสาร Pm_i ด้วยดังแสดงในสมการที่ 4.12

$$\begin{aligned} Ip_1 + Ip_2 + Ip_3 + Im_1 + Im_2 &= 0 \\ Ip_2 + P^1 p_3 + P^2 m_1 + P^3 m_2 &= 0 \\ Ip_3 + P^2 m_1 + P^4 m_2 &= 0 \end{aligned} \tag{4.12}$$

ทำการย้ายข้างเพื่อหาค่าพาริตี้ p_i ซึ่งการบวกเป็นการบวกในลักษณะมอดูโล 2 ดังนั้นการย้ายข้างเพื่อหาค่าพาริตี้จึงไม่ติดค่าลบ และคูณเมตริกซ์เอกลักษณ์ I เข้ากับพาริตี้ p_i และข้อมูล m_i จะได้สมการ

ดังแสดงในสมการที่ 4.13 ซึ่งคุณสมบัติของเมตริกซ์เอกลักษณ์เมื่อคูณกับเวกเตอร์หรือเมตริกซ์ใดๆ จะได้เวกเตอร์หรือเมตริกซ์ตัวเดิม

$$\begin{aligned} p_1 &= p_2 + p_3 + m_1 + m_2 \\ p_2 &= P^1 p_3 + P^2 m_1 + P^3 m_2 \\ p_3 &= P^2 m_1 + P^4 m_2 \end{aligned} \tag{4.13}$$

คุณสมบัติของเมตริกซ์สลับที่คือ เมื่อคูณเมตริกซ์หรือเวกเตอร์ใดๆ จะทำให้เมตริกซ์หรือเวกเตอร์นั้นๆ สลับที่ไปตามจำนวนการยกกำลังของเมตริกซ์สลับที่ คูณเมตริกซ์สลับที่เข้ากับตัวแปรในสมการ 4.13 ทำให้พาริตี้ p_i และข้อมูล m_i สลับที่ตามจำนวนการยกกำลัง ดังแสดงในสมการที่ 4.14

$$\begin{aligned} p_1 &= p_2 + p_3 + m_1 + m_2 \\ p_2 &= (p_3)^1 + (m_1)^2 + (m_2)^3 \\ p_3 &= (m_1)^2 + (m_2)^4 \end{aligned} \tag{4.14}$$

ทำการลดรูปสมการคือ แทนค่า p_3 ในสมการ p_2 แทนค่า p_3 และ p_2 ในสมการ p_1 จากนั้นทำการรวมจำนวนการยกกำลังหรือการเลื่อนบิตแบบวนกลับในตัวแปลที่ได้แทนค่า ดังแสดงในสมการ 4.15

$$\begin{aligned} p_1 &= (m_1)^3 + (m_2)^5 + (m_1)^2 + (m_2)^3 + (m_1)^2 + (m_2)^4 + m_1 + m_2 \\ p_2 &= (m_1)^3 + (m_2)^5 + (m_1)^2 + (m_2)^3 \\ p_3 &= (m_1)^2 + (m_2)^4 \end{aligned} \tag{4.15}$$

เลขยกกำลังแสดงถึงการเลื่อนบิตแบบวนกลับ จากสมการที่ 4.15 การหาค่าพาริตี้ p_1 มีตัวแปลซ้ำเนื่องด้วยการบวกแบบมอดูโล 2 ตัวแปลแบบเดียวกันบวกกันจะได้ศูนย์ ดังนั้นจึงลดรูปข้อมูล $(m_1)^2$ และ $(m_2)^5$ และการเลื่อนบิตแบบวนกลับ 5 ครั้งจะเท่ากับการเลื่อนบิตแบบวนกลับ 0 ครั้ง ทำการลดรูปในแต่ละสมการ แสดงได้ดังสมการที่ 4.16

$$\begin{aligned} p_1 &= m_1 + (m_1)^3 + (m_2)^3 + (m_2)^4 \\ p_2 &= (m_1)^2 + (m_1)^3 + m_2 + (m_2)^3 \\ p_3 &= (m_1)^2 + (m_2)^4 \end{aligned} \tag{4.16}$$

สมการที่คิดขึ้นใหม่นั้นเมื่อนำไปออกแบบวงจร จะใช้ข้อมูลข่าวสารเพียงอย่างเดียวในการหาค่าบิตพิเศษ ซึ่งต่างจากวงจรเข้ารหัสแบบกึ่งขนานคือ ในกรณีของการเข้ารหัสแบบกึ่งขนานนั่นคือ เมื่อบล็อกย่อยแรกหาค่าพาริตี้ได้ ก็จะทำการส่งให้บล็อกย่อยถัดไปนำไปหาค่าพาริตี้ตัวต่อไป แต่วงจรเข้ารหัสแบบขนานนั้นจะให้ข้อมูลข่าวสารเพียงอย่างเดียวจึงไม่ใช้เวลาในการรอบิตพาริตี้

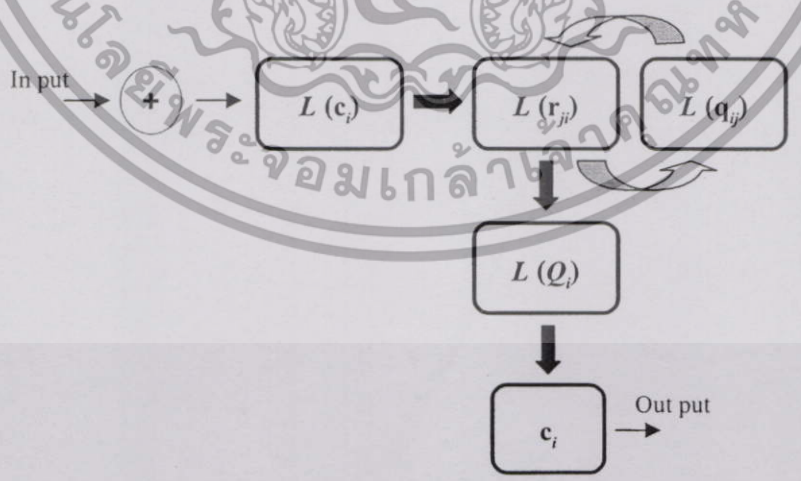
4.3 การออกแบบการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก

เมื่อได้ทำการเข้ารหัสเป็นที่เรียบร้อยแล้ว ข้อมูลก็จะถูกบรรจุอยู่ในคำรหัส (c) แล้วส่งผ่านตัวกลางหรือช่องสัญญาณไปยังด้านรับ เมื่อด้านรับได้รับสัญญาณข่าวสารแล้ว จะทำการถอดรหัสสำหรับสัญญาณที่รับมาได้ เพื่อให้ได้ข้อมูลเดิมออกมาและนำไปใช้งานต่อไป ซึ่งในขั้นตอนของการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ จะนำโครงสร้างของเมตริกซ์ **H** มาใช้ในการสร้างแทน-เนอร์กราฟ (Tanner graph) ในโดเมนลอการิทึม (Log-Domain Algorithm) เนื่องจากการถอดรหัสในรูปแบบของลอการิทึม จะทำให้การใช้จำนวนทรัพยากรน้อยลง และความซับซ้อนที่เพิ่มขึ้นอย่างเป็นเชิงเส้น

โครงสร้างของชุดถอดรหัสพาริตี้เช็คความหนาแน่นต่ำนั้น สามารถแบ่งได้เป็น 5 ส่วน ตามลักษณะการทำงานในการถอดรหัสในรูปแบบของโดเมนลอการิทึม ได้แก่

1. โครงสร้างการหาค่าบิตโนดเริ่มต้น ($L(c_i)$)
2. โครงสร้างการหาค่าเช็คโนด ($L(r_{ji})$)
3. โครงสร้างการหาค่าบิตโนด ($L(q_{ij})$)
4. โครงสร้างการหาค่าการตัดสินใจแบบละเอียด ($L(Q_i)$)
5. โครงสร้างการหาค่าการตัดสินใจแบบหยาบ (c_i)

โดยบล็อกไดอะแกรมแสดงการทำงานของชุดถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ แสดงดังในรูปที่ 4.6

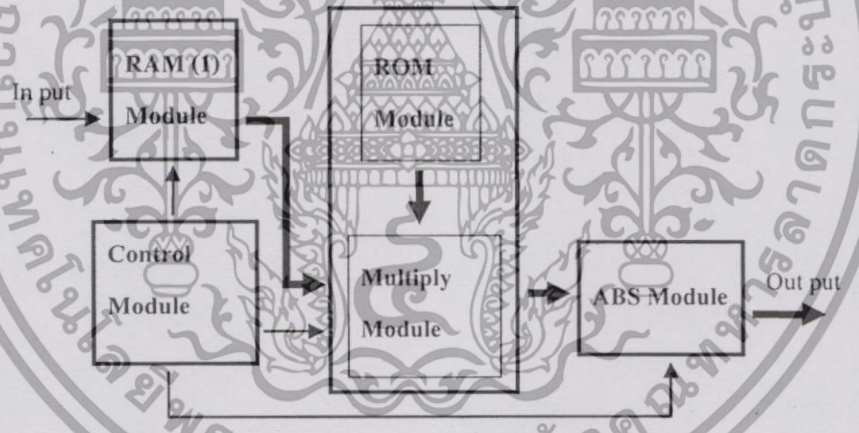


รูปที่ 4.6 ขั้นตอนการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ

โครงสร้างและการทำงานในส่วนต่างๆ ของการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ สามารถอธิบายได้ดังต่อไปนี้

4.3.1 โครงสร้างชุดการหาค่าบิตโนดเริ่มต้น ($L(c_i)$)

การถอดรหัสพาริตี้เช็คความหนาแน่นต่ำนั้นสร้างได้จากสมการที่ 2.42 โดยมีขั้นตอนการทำงานดังนี้ ขั้นตอนที่ 1 เมื่อเริ่มต้นจะทำการรับข้อมูลข่าวสารในลักษณะอนุกรมมาเก็บไว้ในหน่วยความจำชั่วคราว (RAM (1) Module) ขั้นตอนที่ 2 ในขั้นตอนนี้ ข้อมูลข่าวสารจะได้รับการอ่านออกมาในลักษณะขนาน เพื่อเข้าสู่บล็อกการคูณ (Multiply Module) และจะได้รับคูณด้วยค่าคงที่ ที่ได้รับการเก็บไว้ในหน่วยความจำถาวร (ROM Module) จากนั้นจึงทำการส่งออกยังบล็อกต่อไป ขั้นตอนที่ 3 ต่อมาในบล็อกสุดท้ายจะเป็นบล็อกแยกเครื่องหมายบวกและลบ (ABS Module) ออกจากข้อมูลข่าวสาร จากนั้นจะทำการส่งบิตข้อมูลข่าวสารและบิตเครื่องหมายออกไปเพื่อใช้งานต่อไป และการทำงานในแต่ละบล็อกทั้งหมด จะทำงานภายใต้บล็อกควบคุม (Control Module) เพื่อให้ระบบทำงานได้อย่างต่อเนื่องและไม่มีความผิดพลาดทางสัญญาณนาฬิกา และโครงสร้างชุดถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ สามารถแสดงได้ดังในรูปที่ 4.7



รูปที่ 4.7 โครงสร้างการหาค่าบิตโนดเริ่มต้น ($L(c_i)$)

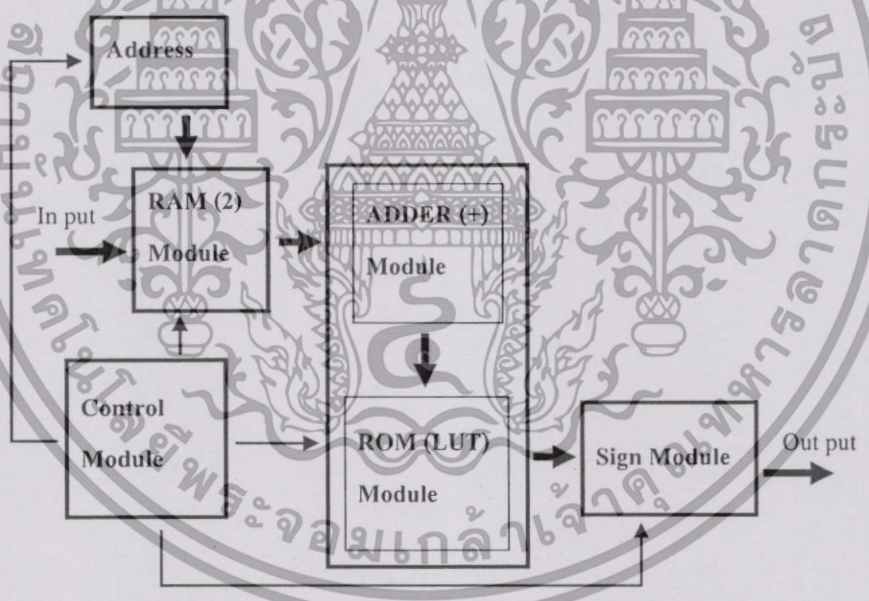
4.3.2 โครงสร้างชุดการหาค่าเช็คโนด ($L(r_{ji})$)

ในส่วนนี้จะเป็นการหาค่าเช็คโนด เพื่อใช้สำหรับช่วยแก้ไขบิตข้อมูลข่าวสารที่อาจเกิดความผิดพลาดในการเดินทางผ่านช่องสัญญาณ ในบล็อกการหาค่าเช็คโนดนี้ จะรับบิตข้อมูลต่อมาจากบล็อกการหาค่าบิตโนดเริ่มต้น โดยการทำงานในการหาค่าเช็คโนดได้แบ่งออกเป็นขั้นตอนการทำงานดังนี้ ขั้นตอนที่ 1 การเริ่มต้นการทำงานของบล็อกการหาค่าเช็คโนดจะรับข้อมูลที่ได้รับการคำนวณมาแล้วต่อจากบล็อกการหาค่าบิตโนดเริ่มต้น และจะนำค่าที่ได้รับมานั้นเก็บไว้ในหน่วยความจำชั่วคราว (RAM (2) Module) ขั้นตอนที่ 2 บิตข้อมูลข่าวสารที่ได้รับการเก็บไว้ใน

หน่วยความจำชั่วคราวจะได้รับการอ่านออกมาเพื่อทำการบวก (ADDER (+) Module) ซึ่งการอ่านค่าในแต่ละที่อยู่ (Address) ออกมานั้นจะถูกอ่านตามโครงสร้างของเมตริกซ์พาริตี้เช็ค **H** ขั้นตอนที่ 3 นำค่าผลลัพธ์ที่ได้จากการบวกมาทำการหาค่าด้วยวิธีการการเปิดตาราง (LUT) ซึ่งค่าที่เก็บไว้ในหน่วยความจำถาวร (ROM) นั้นสามารถหาได้จากสมการที่ 4.17 โดยตัวแปล x คือค่าที่รับเข้ามาคำนวณหรือค่าที่รับจากด้านบิตโนด และฟังก์ชัน $\phi(x)$ คือผลลัพธ์ที่ได้หรือค่าที่ในการคำนวณหาค่าเช็คโนด

$$\phi(x) = \log((e^{x} + 1)/(e^{x}-1)) \qquad (4.17)$$

ขั้นตอนที่ 4 บิตข้อมูลที่ได้รับการคำนวณโดยการเปิดตารางมาแล้ว ในขั้นต้อนนี้จะทำการรวมเครื่องหมายบวกและลบ (Sign Module) เข้ากับบิตข้อมูลข่าวสาร การทำงานของบล็อกต่างๆ ในโครงสร้างการหาค่าเช็คโนดนั้นจะทำงานภายใต้การควบคุมจากบล็อกควบคุม (Control Module) และสามารถแสดงโครงสร้างการหาค่าเช็คโนดได้ดังในรูปที่ 4.8

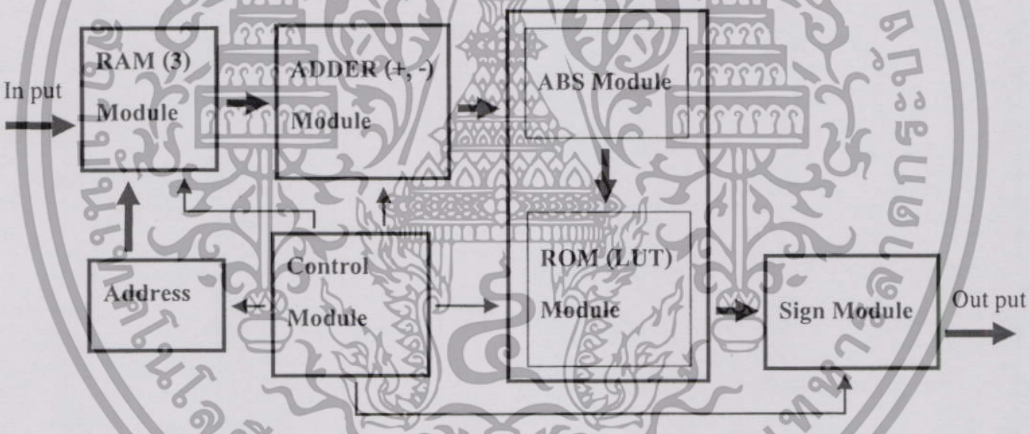


รูปที่ 4.8 โครงสร้างการหาค่าเช็คโนด ($L(r_{ji})$)

ในการหาค่าเช็คโนดนั้นการจัดลำดับการอ่านค่าบิตข้อมูลข่าวสารนั้นถือเป็นหัวใจของการถอดรหัส จึงทำให้การออกแบบในส่วนนี้เป็นสิ่งสำคัญ เนื่องจากในกรณีที่ต้องการเปลี่ยนแปลงอัตราการเข้ารหัส หรือเปลี่ยนแปลงขนาดบล็อกข้อมูล การลำดับการอ่านค่าออกจากหน่วยความจำชั่วคราวจะเปลี่ยนแปลงอยู่ตลอดเวลา และต้องเข้าใจโครงสร้างเมตริกซ์พาริตี้เช็ค **H** เป็นอย่างดี

4.3.3 โครงสร้างชุดการหาค่าบิตโนด ($L(q_{ij})$)

ขั้นตอนการหาค่าบิตโนดส่วนนี้จะเห็นว่าเป็นการหาค่าบิตโนดเป็นครั้งที่ 2 การหาค่าบิตโนดในครั้งนี้ เป็นการหาเพื่อหาค่าความน่าเชื่อถือว่าบิตข้อมูลข่าวสารที่ได้รับมาและนำมาทำการถอดรหัสนั้นควรจะมีค่าเป็น 0 หรือ 1 การทำงานในลักษณะนี้เป็นการหาค่าแบบวนซ้ำเพื่อเป็นการพิจารณาและทำการตัดสินใจต่อไป ชุดโครงสร้างการหาค่าบิตโนดมีวิธีการดังนี้ ขั้นตอนที่ 1 บิตข้อมูลที่ส่งมาจากบล็อกการหาค่าเช็คโนดจะได้รับการเก็บไว้ในหน่วยความจำชั่วคราว (RAM (3) Module) ขั้นตอนที่ 2 ค่าที่เก็บไว้ในหน่วยความจำชั่วคราว จะได้รับการอ่านออกมาเพื่อทำการบวก (ADDER (+, -) Module) โดยการบวกนี้จะมีทั้งค่าบวกและค่าลบ ขั้นตอนที่ 3 ผลรวมของบล็อกที่ผ่านมาบิตข้อมูลจะมีทั้งค่าบวกและค่าลบ ซึ่งในบล็อกแยกเครื่องสัญลักษณ์ (ABS Module) จะทำการแยกสัญลักษณ์ออกจากบิตข้อมูล จากนั้นจะทำการส่งบิตข้อมูลต่อไปยังหน่วยความจำถาวร (ROM (LUT) Module) เพื่อเปิดตารางหาค่าลอการิทึมและส่งบิตข้อมูลไปยังบล็อกถัดไป ขั้นตอนที่ 4 เป็นการรวมบิตสัญลักษณ์เข้ากับบิตข้อมูล ซึ่งจากบล็อกข้อมูลในชุดการหาค่าบิตโนดจะทำงานภายใต้การควบคุมของบล็อกควบคุม ดังแสดงในรูปที่ 4.9



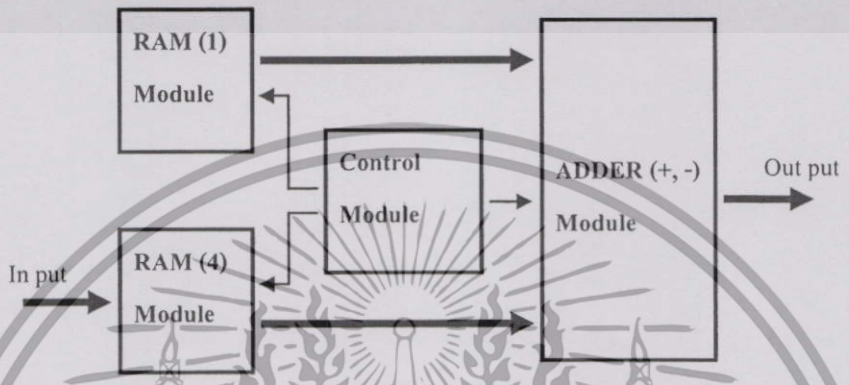
รูปที่ 4.9 โครงสร้างการหาค่าบิตโนด ($L(q_{ij})$)

สาเหตุในการแยกบิตสัญลักษณ์ออกจากบิตข้อมูลนั้น เนื่องมาจากค่าหรือตัวแปลที่จะนำมาเปิดตารางนั้นจะต้องเป็นเฉพาะสัญลักษณ์บวก เพราะค่าที่นำมาเก็บไว้ในตารางนั้นหาได้จากสมการลอการิทึมในสมการที่ 4.10 ดังนั้นค่าที่จะนำมาหาค่าที่เปิดจากตางนั้นจึงมีได้เฉพาะสัญลักษณ์บวกเท่านั้น

4.3.4 โครงสร้างชุดการหาค่าการตัดสินใจแบบละเอียด ($L(Q_i)$)

ในส่วนของการตัดสินใจในแบบละเอียดนี้ จะนำค่าจากหน่วยความจำในชุดแรก (RAM (1) Module) และหน่วยความจำชั่วคราวในชุดที่ 4 (RAM (4) Module) มาทำการคำนวณรวมกัน ซึ่งมี

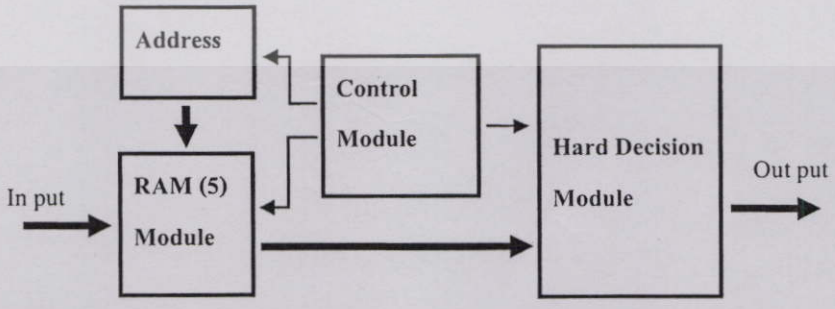
ขั้นตอนดังนี้ ขั้นตอนที่ 1 จะได้รับการอ่านบิตข้อมูลออกมาจากหน่วยความจำชั่วคราวจากชุดที่ 1 และชุดที่ 4 ออกมาพร้อมกัน ขั้นตอนที่ 2 นำบิตข้อมูลที่อ่านออกมาจากหน่วยความจำมาทำการบวกกันในบล็อกการบวก (ADDER (+, -) Module) ซึ่งการบวกในที่นี้เป็นการบวกโดยมีสัญลักษณ์บวกและลบ ซึ่งการทำงานทั้งหมดจะได้รับการควบคุมจากบล็อกควบคุม (Control Module) ดังแสดงในรูปที่ 4.10



รูปที่ 4.10 โครงสร้างการหาค่าการตัดสินใจแบบละเอียด ($L(Q_i)$)

4.3.5 โครงสร้างชุดการหาค่าการตัดสินใจแบบหยาบ (c_i)

ขั้นตอนการตัดสินแบบหยาบเป็นขั้นตอนสุดท้าย ในการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ ในขั้นตอนนี้จะเป็นการตัดสินใจ ในกรณีที่บิตข้อมูลน้อยกว่าศูนย์ค่าที่ได้จะเป็น 1 หรือในกรณีที่บิตข้อมูลมากกว่าศูนย์ค่าที่ได้จะเป็น 0 ซึ่งขั้นตอนในการตัดสินใจแบบหยาบมีขั้นตอนดังนี้ ขั้นตอนที่ 1 บิตข้อมูลที่ได้รับจากบล็อกก่อนหน้าจะได้รับการเก็บเข้าในหน่วยความจำชั่วคราว (RAM (5) Module) และจะทำการอ่านค่าออกตามที่อยู่ (Address) ที่กำหนด ขั้นตอนที่ 2 บิตข้อมูลที่อ่านออกมาจากหน่วยความจำชั่วคราวจะได้รับการตัดสินใจว่าเป็น 0 หรือ 1 จากบล็อกการตัดสินใจแบบหยาบ (Hard Decision Module) การทำงานทั้งหมดในการหาค่าการตัดสินใจแบบหยาบจะทำงานภายใต้การควบคุมของบล็อกควบคุม (Control Module) ดังแสดงในรูปที่ 4.11



รูปที่ 4.11 โครงสร้างการหาค่าการตัดสินใจแบบหยาบ (c_i)

การหาค่าบิตโนดเริ่มต้นนั้น ในขั้นตอนแรกจะรับค่าเข้ามาเก็บไว้ในหน่วยความจำชั่วคราว (RAM Module) โดยการเก็บค่าและการอ่านค่าในหน่วยความจำชั่วคราวจะถูกควบคุมด้วยจากชุดควบคุม (Control Module) ให้ทำการเก็บค่าในรูปแบบอนุกรมและอ่านค่าออกในลักษณะขนาน ขั้นตอนที่ 2 ชุดควบคุมจะทำการควบคุมให้ทำการคูณค่าที่อ่านออกมาจากหน่วยความจำชั่วคราวกับค่าคงที่ ที่ได้รับการเก็บไว้ในส่วนของหน่วยความจำถาวร (ROM Module)

บทที่ 5

ผลการออกแบบและทดสอบการทำงาน

รหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิกได้รับการอธิบายในส่วนของทฤษฎีในบทที่ 2 และได้รับการพัฒนาต่อมาในบทที่ 4 ว่าด้วยเรื่องการออกแบบโครงสร้างต่างๆ ทั้งในส่วนของการออกแบบโครงสร้างการเข้ารหัส และโครงสร้างในส่วนของการถอดรหัส ดังนั้นในบทที่ 5 จึงเป็นผลการออกแบบและทดสอบการทำงาน สำหรับวิทยานิพนธ์นี้ใช้ภาษาวีเอชดีแอล (VHDL) ในการบรรยายพฤติกรรมการทำงานของวงจรชุดเข้ารหัสและถอดรหัส โดยใช้โปรแกรม Xilinx ISE 9.1 ในการออกแบบและการสังเคราะห์วงจร (Synthesis) และใช้โปรแกรม ModelSim ในส่วนของการจำลองการทำงานและการตรวจสอบ

ในส่วนแรกแสดงผลการออกแบบชุดเข้ารหัสของวงจรที่ออกแบบในแต่ละชุดการทำงาน จำนวนการใช้อุปกรณ์ภายในเอฟพีจีเอ และผลการการจำลองการทำงานของชุดเข้ารหัสโดยแบ่งตามภาคต่างๆ ในส่วนที่สองแสดงผลการออกแบบชุดถอดรหัสในแต่ละชุด จำนวนการใช้ทรัพยากร และทดสอบประสิทธิภาพของรหัสพาริตี้เช็คความหนาแน่นต่ำในช่องสัญญาณรบกวนแบบขาว (Additive White Gaussian Noise: AWGN) เป็นการเปรียบเทียบวิธีการถอดรหัส ด้วยเลขจำนวนจริง (Floating-point) กับการถอดรหัสด้วยเลขตายตัว (Fixed-point) และในส่วนสุดท้ายกล่าวถึงการวิเคราะห์ประสิทธิภาพและลักษณะเด่นของชุดถอดรหัส

5.1 ผลการออกแบบและจำลองการทำงาน

ภาษาวีเอชดีแอลได้รับการนำเสนอมาใช้เพื่อออกแบบวงจร ในการเข้ารหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ ผลที่ได้จากการออกแบบทั้งเข้ารหัสและถอดรหัสโดยภาษาวีเอชดีแอล เมื่อนำไปสังเคราะห์วงจรโดยใช้โปรแกรม Xilinx ISE จะได้เป็นกลุ่มของเน็ตลิสต์ (Netlist) ในระดับเกต จากนั้นตรวจสอบความถูกต้องโดยการจำลองการทำงานโดยโปแกรม ModelSim การออกแบบในส่วนนี้แสดงให้เห็นถึงผังวงจรในระดับ RTL และผลการใช้ทรัพยากรในอุปกรณ์ภายในเอฟพีจีเอ

ชุดเข้ารหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบ่งเป็น 2 ส่วน

- ส่วนแรกได้แก่ ชุดเข้ารหัสประกอบไปด้วย ชุดดำเนินการเลื่อนบิตแบบวนกลับ และชุดการบวกแบบมอดูโล - 2

- ส่วนที่สองได้แก่ ชุดถอดรหัสประกอบไปด้วย หน่วยความจำชั่วคราว (RAM) ตัวดำเนินการคูณ (Multiply) ตัวดำเนินการแยกบิตสัญลักษณ์ (ABS) ตัวดำเนินการบวกแบบไม่รวมบิตสัญลักษณ์ ตัวดำเนินการบวกแบบรวมบิตสัญลักษณ์ หน่วยความจำถาวรที่ในการเก็บค่า $\log((e^x+1)/(e^x-1))$ ตัวดำเนินการรวมบิตสัญลักษณ์ (Sign) และในส่วนของชุดควบคุม (Controller)

ในภาษาวีเอชดีแอลการออกแบบในลักษณะข้างต้น จะทำการออกแบบแยกในแต่ละชุด โดยในแต่ละชุดเรียกว่า "มอดูล" ซึ่งสะดวกในการออกแบบ ในส่วนของการทดสอบจะทำการจำลองการทำงานในแต่ละมอดูล และก็จึงนำมอดูลย่อยต่างๆ มาประกอบกันเป็นมอดูลหลัก (Top Module) ซึ่งผลการออกแบบและการจำลองการทำงานชุดเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำและ ชุดถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ แบ่งตามโครงสร้างต่างๆ มีดังนี้

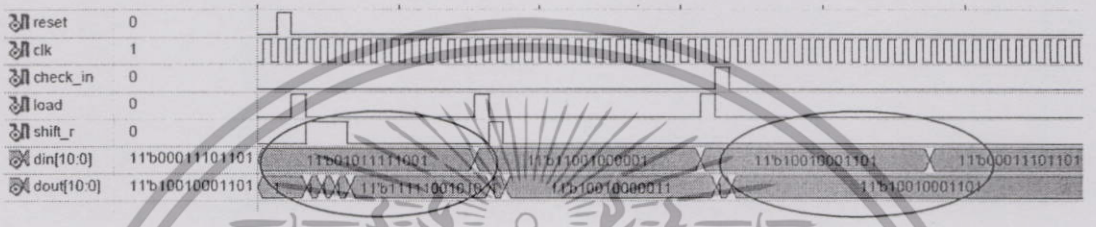
5.1.1 โครงสร้างและการออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำ

5.1.1.1 โครงสร้างชุดดำเนินการเลื่อนบิตแบบวนกลับ

มอดูลชุดดำเนินการเลื่อนบิตแบบวนกลับจัดอยู่ในส่วนของชุดโครงสร้างการเข้ารหัส โดยมีหน้าที่รับบิตอินพุต (Input) มาทำการเลื่อนบิตแบบวนกลับ การเลื่อนบิตนั้นจะเลื่อนตามจำนวนครั้งที่กำหนด หรือตามการกำลังของเมตริกซ์สลับที่ในสมการที่ 4.9 เมื่อนำมาออกแบบการทำงานและสังเคราะห์ชุดเข้ารหัสด้วยโปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ในระดับเกต ซึ่งแสดงให้เห็นถึงการใช้ทรัพยากรดังในรูปที่ 5.1

รูปที่ 5.1 ผังวงจรระดับ RTL ของชุดดำเนินการเลื่อนบิตแบบวนกลับ

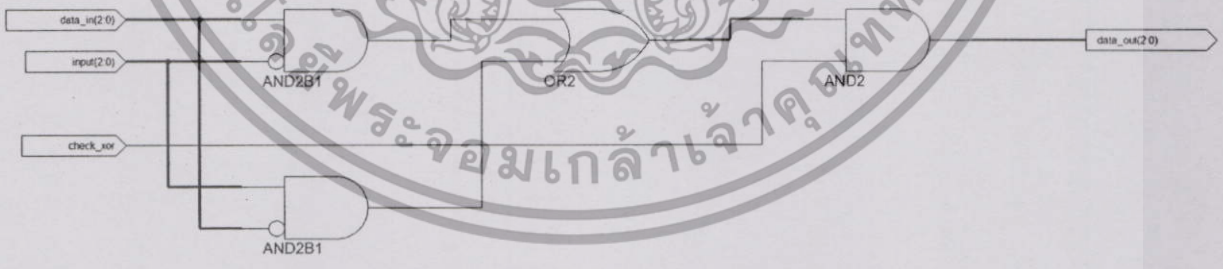
การจำลองการทำงานของชุดการเลื่อนบิตแบบวนกลับ ซึ่งประกอบไปด้วยสัญญาณรีเซต สัญญาณนาฬิกา สัญญาณตรวจสอบ สัญญาณโหลดข้อมูล สัญญาณจำนวนการเลื่อนบิตแบบวนกลับ และบิตข้อมูลที่ต้องการทำการเลื่อนบิตแบบวนกลับ เมื่อเริ่มต้นการรับบิตข้อมูลเข้ามาในระบบ ระบบจะทำการโหลดข้อมูล จากนั้นจะทำการเลื่อนบิตข้อมูลแบบวนกลับ โดยข้อมูลจะเลื่อนบิตแบบวนกลับจำนวน 2 ครั้งตามการควบคุมของสัญญาณจำนวนการเลื่อนบิตแบบวนกลับ จากบิตข้อมูล 01011111001 เป็น 11111001010 และในกรณีที่ไม่มีสัญญาณควบคุมจำนวนการเลื่อนบิตแบบวนกลับ บิตข้อมูลก็จะไม่ได้รับการเลื่อนดังแสดงในรูปที่ 5.2



รูปที่ 5.2 ผลการจำลองการทำงานของตัวดำเนินการเลื่อนบิตแบบวนกลับ

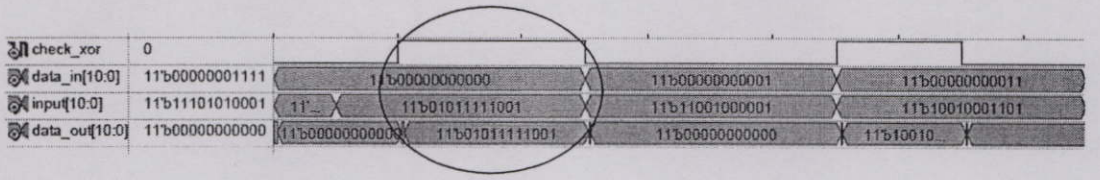
5.1.1.2 โครงสร้างชุดการบวกลักษณะมอดูโล-2

โครงสร้างชุดการบวกลักษณะมอดูโล-2 นี้ อยู่ในส่วนของชุดเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำ มีหน้าที่ในการรับบิตข้อมูลต่อจากชุดดำเนินการเลื่อนบิตแบบวนกลับ มาทำการบวกในลักษณะมอดูโล-2 กับบิตข้อมูลที่อยู่ก่อนหน้า ภายใต้การควบคุมของตัวควบคุม เมื่อนำมาออกแบบการทำงานและสังเคราะห์ชุดเข้ารหัสด้วยโปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ในระดับเกต ดังแสดงในรูปที่ 5.3



รูปที่ 5.3 ผังวงจรระดับ RTL ของชุดการบวกลักษณะมอดูโล-2

การจำลองการทำงานของชุดการการบวกในลักษณะมอดูโล-2 ซึ่งประกอบไปด้วยสัญญาณควบคุม บิตข้อมูลที่ต้องการบวกในลักษณะมอดูโล-2 และบิตข้อมูลที่อยู่ก่อนหน้า ขั้นตอนแรกในการทำงานของระบบคือรับบิตข้อมูลเข้ามาในระบบ และชุดควบคุมจะเป็นทำการควบคุมบิตข้อมูลให้ทำการบวกกันในลักษณะมอดูโล-2 ดังแสดงในรูปที่ 5.4

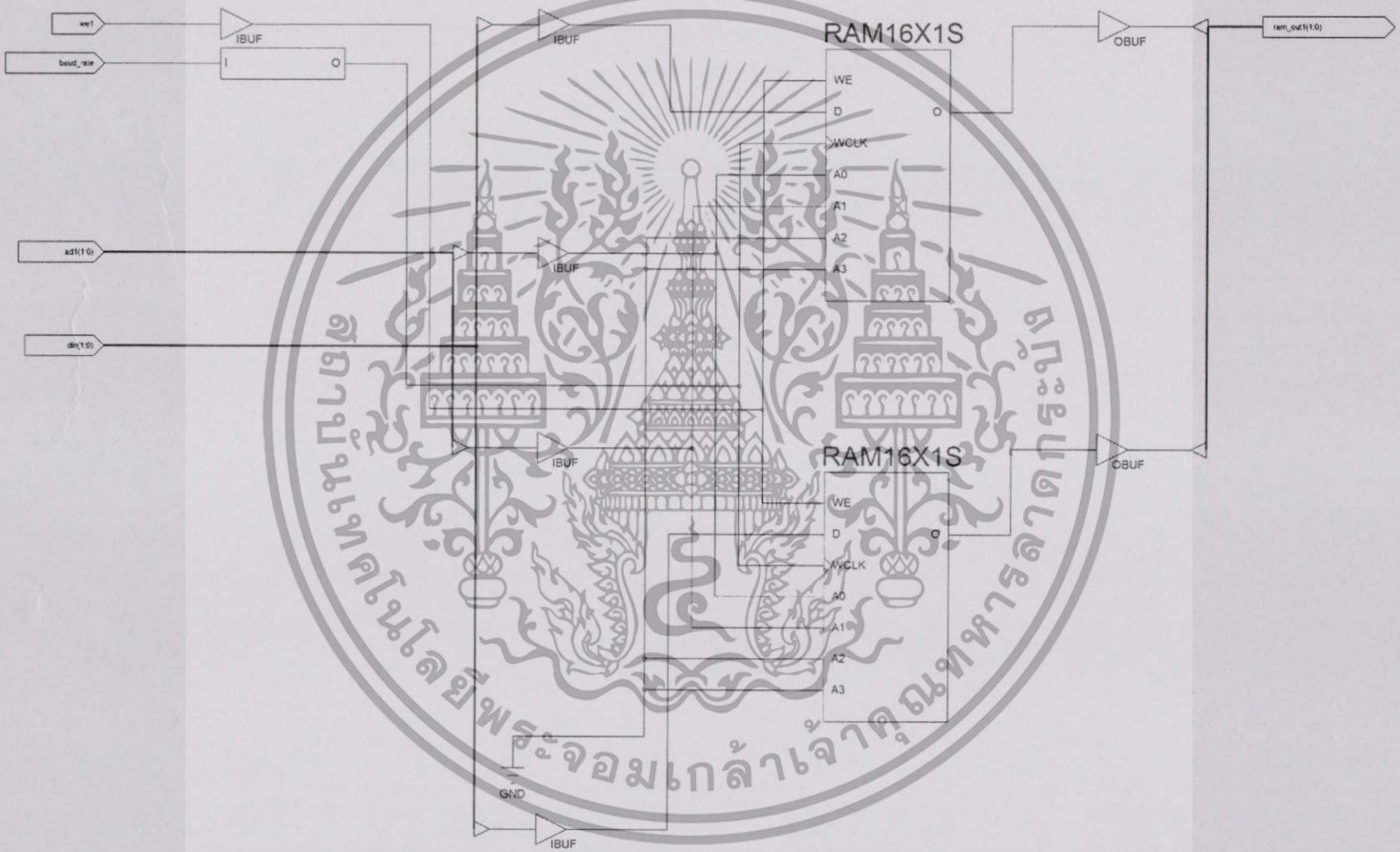


รูปที่ 5.4 ผลการจำลองการทำงานตัวดำเนินการบวกแบบมอดูโล-2

5.1.2 โครงสร้างและการออกแบบการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ

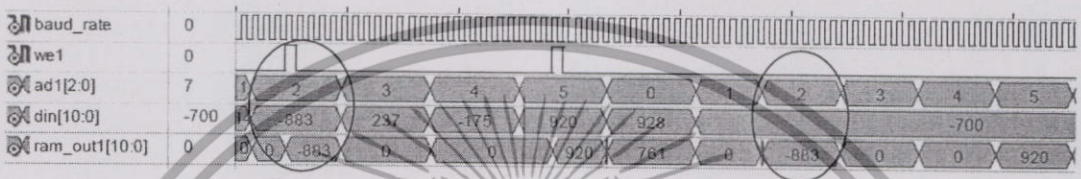
5.1.2.1 โครงสร้างชุดหน่วยความจำชั่วคราว (RAM)

หน่วยความจำชั่วคราวเป็นสิ่งจำเป็นสำหรับการออกแบบการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ เนื่องจากการอ่านค่าบิตข้อมูลในการคำนวณหาค่าบิตโนด และหาค่าเช็คโนดนั้นไม่อ่านตามลำดับของที่อยู่ (Address) เพราะการอ่านค่าในแต่ละที่อยู่ของหน่วยความจำชั่วคราวนั้นจะขึ้นกับโครงสร้างของเมตริกซ์พาริตี้เช็ค H การทำงานของหน่วยความจำชั่วคราวนี้จะทำภายใต้การควบคุมของสัญญาณนาฬิกา และได้รับการควบคุมทั้งในการอ่านข้อมูลและเขียนข้อมูล เมื่อนำมาออกแบบการทำงานและสังเคราะห์วงจรชุดหน่วยความจำชั่วคราวโดยโปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ในระดับเกต ดังแสดงในรูปที่ 5.5



รูปที่ 5.5 ผังวงจรระดับ RTL ของชุดหน่วยความจำชั่วคราว (RAM)

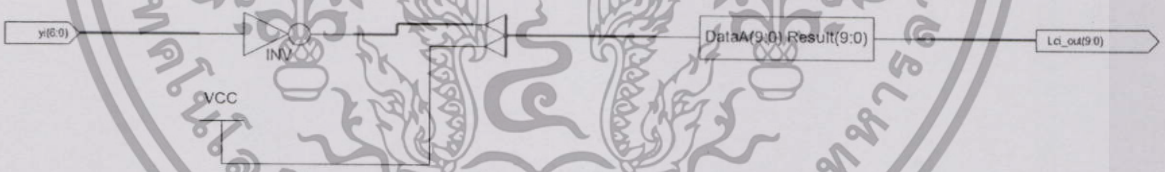
การจำลองการทำงานของชุดหน่วยความจำชั่วคราว ซึ่งประกอบไปด้วยสัญญาณสัญญาณนาฬิกา สัญญาณการเก็บบิตข้อมูลและอ่านบิตข้อมูล บิตข้อมูลที่อยู่ของบิตข้อมูล การทำงานของหน่วยความจำชั่วคราว เริ่มต้นด้วยการเก็บบิตข้อมูลเข้าหน่วยความจำ การเก็บบิตข้อมูลเข้าในหน่วยความจำนั้นควบคุมโดยสัญญาณการเก็บบิตข้อมูลและอ่านบิตข้อมูล ถ้าสัญญาณการเก็บบิตข้อมูลและอ่านบิตข้อมูลเท่ากับ 1 จะทำการเก็บค่าและถ้าเท่ากับ 0 จะทำการอ่านค่าออกเพื่อนำไปใช้ต่อไป และในแต่ละหน่วยความจำต้องระบุด้วยที่อยู่ของบิตข้อมูลที่เก็บไว้ในหน่วยความจำ การทำงานทั้งหมดสามารถแสดงได้ดังในรูปที่ 5.6



รูปที่ 5.6 ผลการจำลองการทำงานของหน่วยความจำชั่วคราว

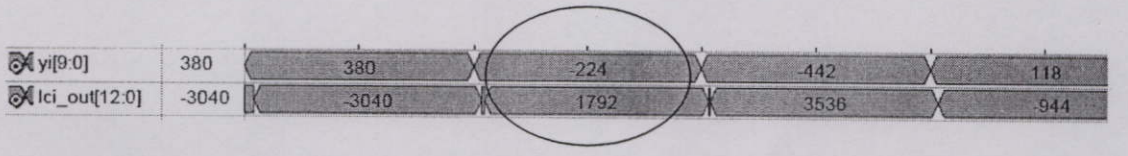
5.1.2.2 โครงสร้างชุดตัวดำเนินการคูณ (Multiply)

โครงสร้างชุดดำเนินการคูณ ได้สร้างมาจากสมการที่ 2.41 โครงสร้างส่วนนี้จะทำหน้าที่นำค่า y_i มาคำนวณหาค่าบิตโนดเริ่มต้น ($L(c_i)$) ด้วยวิธีการคูณซึ่งเป็นการคูณที่มีทั้งบวกและลบ จากนั้นนำมาออกแบบการทำงานและสังเคราะห์ชุดตัวดำเนินการคูณโปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ในระดับเกต ดังแสดงในรูปที่ 5.7



รูปที่ 5.7 ผังวงจรระดับ RTL ของชุดตัวดำเนินการคูณ (Multiply)

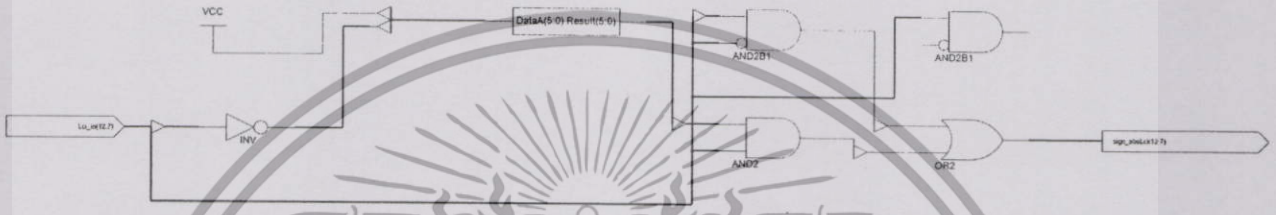
การจำลองการทำงานในการหาค่าบิตโนดเริ่มต้นประกอบด้วยบิตข้อมูลที่ต้องการนำมาหาค่าบิตโนดเริ่มต้น มีขั้นตอนการทำงานคือ เมื่อรับบิตข้อมูลเข้ามาในระบบจะนำบิตข้อมูลที่ได้คูณด้วย -8 ซึ่งเป็นค่าคงที่ ที่ได้รับการเก็บไว้ในหน่วยความจำถาวร และผลลัพธ์ที่ได้สามารถแสดงดังในรูปที่ 5.8



รูปที่ 5.8 การจำลองการทำงานของตัวดำเนินการคูณ (Multiply)

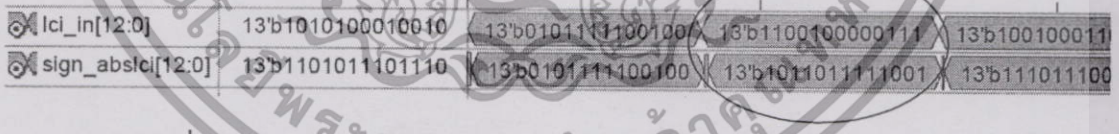
5.1.2.3 โครงสร้างชุดการแยกสัญลักษณ์ (ABS)

การแยกสัญลักษณ์หรือการแยกเครื่องหมายบวกและลบออกจากบิตข้อมูล เป็นสิ่งจำเป็นในการถอดรหัสของรหัสพาริตี้เช็คความหนาแน่นต่ำ เนื่องจากในขั้นตอนการหาค่า $\log((e^x+1)/(e^x-1))$ ในสมการที่ 4.6 นั้น ค่าที่จะนำมาคำนวณนั้นเป็นได้เฉพาะค่าที่เป็นบวก ดังนั้นการแยกสัญลักษณ์จึงเป็นสิ่งที่ต้องทำก่อนนำบิตข้อมูลมาทำการหาค่า $\log((e^x + 1)/(e^x - 1))$ ด้วยวิธีการผ่านตาราง เมื่อนำมาออกแบบการทำงานและสังเคราะห์วงจรชุดการแยกสัญลักษณ์ด้วยโปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ในระดับเกต แสดงได้ดังในรูปที่ 5.9



รูปที่ 5.9 ผังวงจรระดับ RTL ของชุดการแยกเครื่องหมาย (ABS)

การจำลองการทำงานของ การแยกบิตสัญลักษณ์นั้น จะรับข้อมูลเข้ามาทั้งหมด 12 บิต จากนั้นจะนำบิตที่ 12 มาทำการตัดสินใจว่าบิตข้อมูลที่รับเข้ามานั้นจะนำไปคำนวณต่อหรือไม่ กรณีที่ค่าเป็น 0 แสดงว่าบิตข้อมูลที่รับมาได้นั้นเป็นบวกระบบก็จะทำการส่งบิตข้อมูลชุดเดิมออกไป แต่ในกรณีที่เป็น 1 แสดงว่าบิตข้อมูลที่รับมาได้เป็นลบระบบจะทำการนำบิตข้อมูลที่เหลือทั้ง 11 บิตมาทำการ เอ็กซ์ออ (XOR) กับ 11111111111 และจากนั้นนำค่าที่ได้รับ มาบวกด้วย 1 จากนั้นจึงส่งขอมูลออกเพื่อนำไปใช้ต่อไป การทำงานสามารถแสดงในรูปที่ 5.10



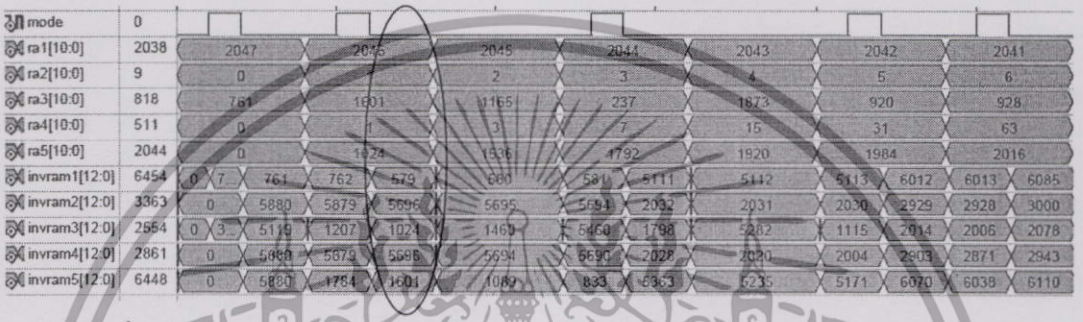
รูปที่ 5.10 ผลการจำลองการทำงานของบล็อกการแยกเครื่องหมาย (ABS)

5.1.2.4 โครงสร้างชุดตัวดำเนินการบวกแบบไม่คิดเครื่องหมาย

การบวกที่ออกแบบให้ใช้สำหรับการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำนั้นทำได้สองวิธี คือการบวกแบบคิดเครื่องหมายและการบวกแบบไม่คิดเครื่องหมาย ซึ่งการบวกแบบไม่คิดเครื่องหมายนี้จะอยู่ในขั้นตอนของการหาค่าเช็คโนด ($L(r_{ji})$) ซึ่งการบวกนี้จะรับบิตข้อมูลมาทำการคำนวณหาผลบวกรวม จากนั้นนำมาลบกับข้อมูลเดิมที่ต้องการหา เมื่อนำมาออกแบบการทำงานและสังเคราะห์วงจรชุดถอดรหัสด้วยโปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ในระดับเกต แสดงได้ดังในรูปที่ 5.11

รูปที่ 5.11 ผังวงจรระดับ RTL ของชุดดำเนินการบวกแบบไม่คิดเครื่องหมาย (+)

การจำลองการทำงานของตัวดำเนินการบวกแบบไม่คิดเครื่องหมาย ซึ่งประกอบไปด้วยสัญญาณควบคุมการบวก บิตข้อมูลชุดที่ 1 บิตข้อมูลชุดที่ 2 บิตข้อมูลชุดที่ 3 บิตข้อมูลชุดที่ 4 และบิตข้อมูลชุดที่ 5 และระบบจะทำงานดังนี้คือถ้าต้องการหาค่าของบิตข้อมูลชุดที่ 1 ระบบจะรับค่าบิตข้อมูลพร้อมกันทั้ง 5 ชุด จากนั้นจะทำการบวกรวม และเมื่อได้ผลบวกรวมแล้วจะนำผลบวกรวมที่ได้มาทำการลบด้วยบิตข้อมูลชุดที่ 1 อีกครั้ง ในกรณีที่ต้องการหาบิตข้อมูลชุดที่ 2 ก็เช่นกันคือ จะนำผลบวกรวมมาทำการลบด้วยบิตข้อมูลชุดที่ 2 จากนั้นก็จะทำการส่งข้อมูลต่อไปเพื่อนำไปใช้งานในบล็อกต่อไป การทำงานสามารถแสดงในรูปที่ 5.12



รูปที่ 5.12 ผลการจำลองการทำงานของตัวดำเนินการบวกแบบไม่คิดเครื่องหมาย (+)

5.1.2.5 โครงสร้างชุดหน่วยความจำถาวรที่ใช้ในการเปิดตาราง (LUT)

หน่วยความจำถาวรหรือตารางการเก็บค่าเพื่อใช้สำหรับการคำนวณค่าในสมการที่ 4.6 ซึ่งในการเก็บค่านี้สามารถเก็บให้ละเอียดหรือไม่ละเอียดนั้น ขึ้นอยู่กับความต้องการของผู้ออกแบบ ซึ่งการเก็บค่าให้ละเอียดนั้นอาจทำให้ค่าที่ประมาณใกล้เคียงกับค่าจำนวนจริงมากขึ้น และความผิดพลาดน้อย แต่การใช้ทรัพยากรในการสร้างวงจรก็จะเพิ่มขึ้น ถ้าในกรณีที่มีจำนวนทรัพยากรอย่างจำกัดการเก็บค่าไม่ละเอียดก็น่าจะเหมาะสมกว่าแต่ค่าที่ประมาณอาจผิดพลาดได้มากกว่า เมื่อนำมาออกแบบการทำงานและสังเคราะห์วงจรชุดถอดรหัสด้วยโปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ในระดับเกต แสดงได้ดังในรูปที่ 5.13

รูปที่ 5.13 ผังวงจรระดับ RTL ของชุดหน่วยความจำถาวรที่ใช้ในการเปิดตาราง (LUT)

จากการออกแบบวงจรที่ใช้สำหรับการหาค่าในสมการ log ((ex+1)/(ex-1)) และจึงนำมาทำการทดสอบผลการทำงานของระบบ การทำงานประกอบไปด้วยบิตข้อมูลด้านรับ และบิตข้อมูลด้านส่ง เมื่อรับข้อมูลเข้ามามีทั้งหมด 13 บิต จากนั้นจะทำการแบ่งบิตข้อมูลออกเป็น 3 ชุด ชุดแรกเป็น บิตที่ 12 ชุดที่สองเป็น บิตที่ 11 ถึง 10 และชุดสุดท้ายเป็นบิตที่ 9 ถึง 0 ดังเสดงในรูปที่ 5.14

1	11	1100101010

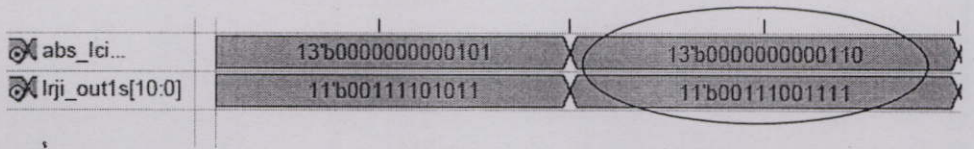
รูปที่ 5.14 การแบ่งบิตข้อมูล

โดยบิตข้อมูลชุดที่ 1 แทนสัญลักษณ์ ชุดที่สองแทนค่าที่เกินจาก 6 เนื่องจากการแบ่งระดับสัญญาณอยู่ในช่วง 5.1397 ถึง 0.0050 ในกรณีที่บิตใดในสองบิตเป็น 1 ค่าที่ได้จะเท่ากับ 0000000001 โดยไม่ต้องผ่านตางรางการแทนค่า log $((e^{x}+1)/(e^{x}-1))$ และชุดที่สามเป็นบิตข้อมูลที่นำมาทำการหาค่าในตารางที่ 5.1 ซึ่งค่าในตารางนี้หาได้จากการคำในสมการ 4.6 เป็นการเก็บค่าในลักษณะเลขฐานสอง (Binary)

ตารางที่ 5.1 ตารางแทนค่า $\log((e^x+1)/(e^x-1))$

$\phi(x)$	$\log((e^x + 1)/(e^x-1))$
0000000001	1010011101
0000000010	1001000100
0000000100	1000010000
0000000101	0111101011
0000000110	0111001111
0000001000	0110110111
0000001001	0110100100
0000001011	0110010010
0000010110	0100111010
0000100001	0100000110
0000101100	0011100010
0000110111	0011000110
0001000010	0010110000
0001001101	0010011101
0001011000	0010001101
0010110000	0001000010
0100001000	0000100000
0101100000	0000010000
0110111000	0000001000
1000010000	0000000100
1001101000	0000000010
1011000000	0000000001

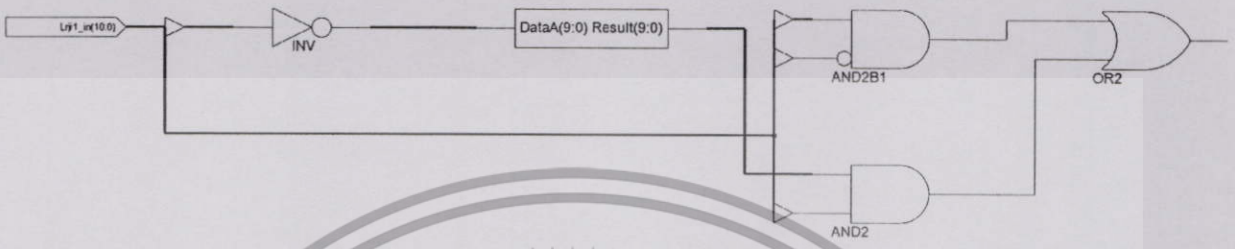
เมื่อทำการหาค่าบิตข้อมูลได้ทุกชุดแล้ว จะนำบิตข้อมูลชุดที่ 1 ที่เป็นบิตสัญลักษณ์ต่อกับบิตข้อมูลที่หาได้จากตาราง และนำบิตข้อมูลชุดที่ได้ส่งออกเพื่อนำไปใช้งานในบล็อกการทำงานต่อไป และแสดงผลการทดสอบการทำงานในรูปที่ 5.15



รูปที่ 5.15 ผลการจำลองการทำงานของหน่วยความจำถาวรที่ใช้ในการเปิดตาราง (LUT)

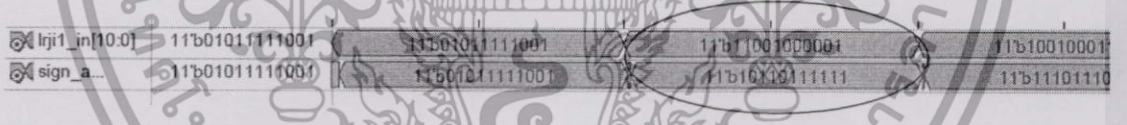
5.1.2.6 โครงสร้างชุดการรวมบิตสัญลักษณ์กับบิตข้อมูล (Sign)

การทำงานในส่วนนี้เป็นการเปลี่ยนบิตข้อมูลที่อยู่ในรูปแบบของค่าบวก มาทำการใส่สัญลักษณ์ทั้งบวกและลบเพื่อนำไปใช้ในขั้นตอนการหาค่าบิตโนด $L(Q_{ij})$ จากนั้นนำมาออกแบบการทำงานและสังเคราะห์วงจร ในระดับเกตสามารถแสดงได้ดังรูป 5.16



รูปที่ 5.16 ผังวงจรระดับ RTL ของชุดดำเนินการรวมบิตสัญลักษณ์กับบิตข้อมูล (Sign)

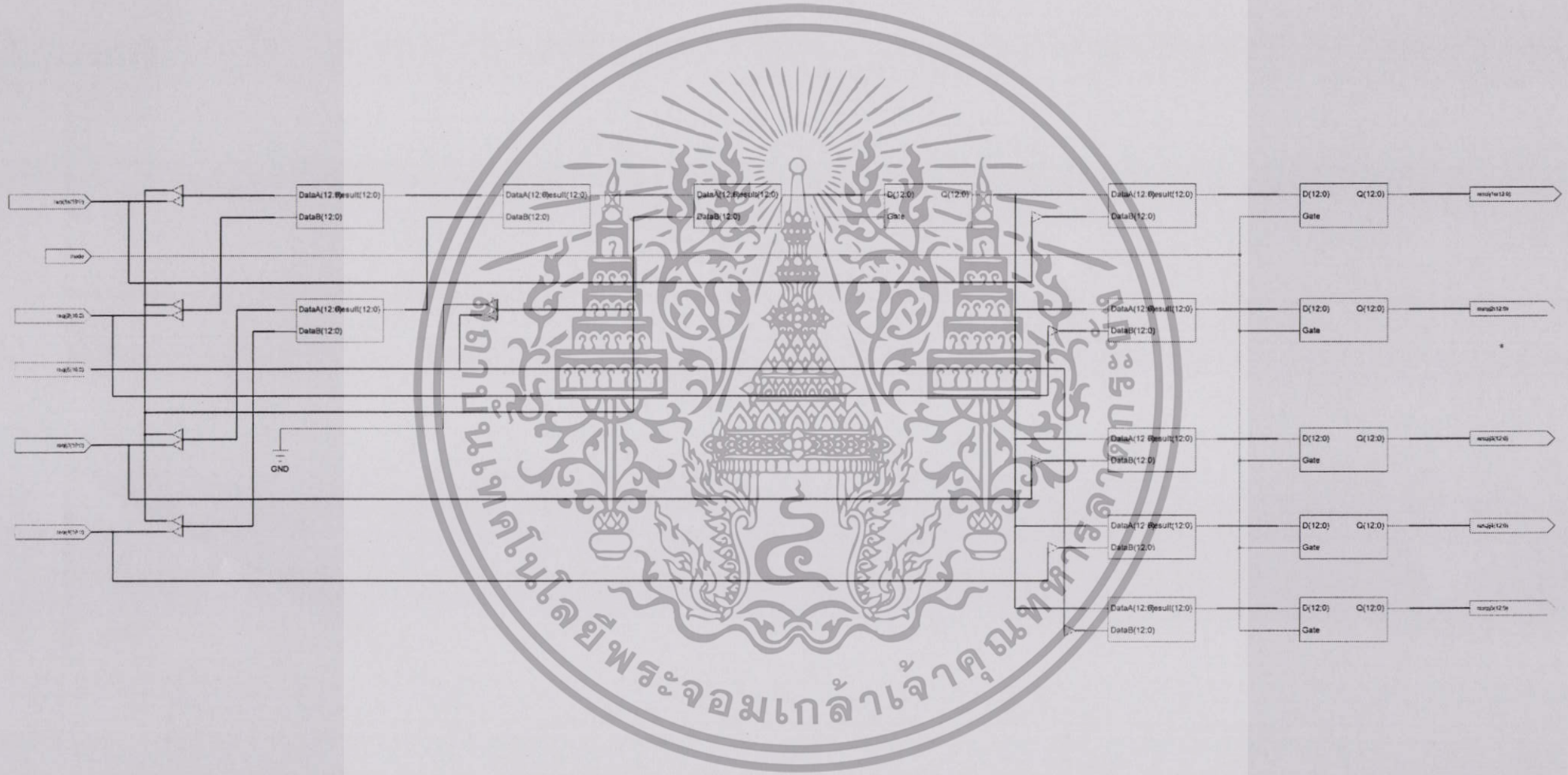
ผลการทดลองในการรวมบิตสัญลักษณ์ประกอบด้วยขั้นตอนการรับบิตข้อมูลและรับบิตข้อมูล เริ่มต้นจะรับข้อมูลทั้งหมด 11 บิต โดยบิตตัวที่ 11 จะเป็นตัวควบคุมการทำงาน ถ้าบิตตัวที่ 11 เป็น 0 บิตข้อมูลจะไม่ได้รับการเปลี่ยนแปลง แต่ถ้าบิตข้อมูลตัวที่ 11 เป็น 1 ข้อมูลจะได้รับการเอ็กซ์ออ (XOR) กับ 11111111111 จากนั้นนำผลที่ได้ไปทำการบวกกับ 1 และส่งบิตข้อมูลที่ผ่านขั้นตอนการรวมบิตสัญลักษณ์กับบิตข้อมูลเพื่อใช้งานในบล็อกต่อไปผลการจำลองการทำงานแสดงได้ดังในรูปที่ 5.17



รูปที่ 5.17 ผลการจำลองการทำงานของตัวดำเนินการรวมบิตสัญลักษณ์กับบิตข้อมูล (Sign)

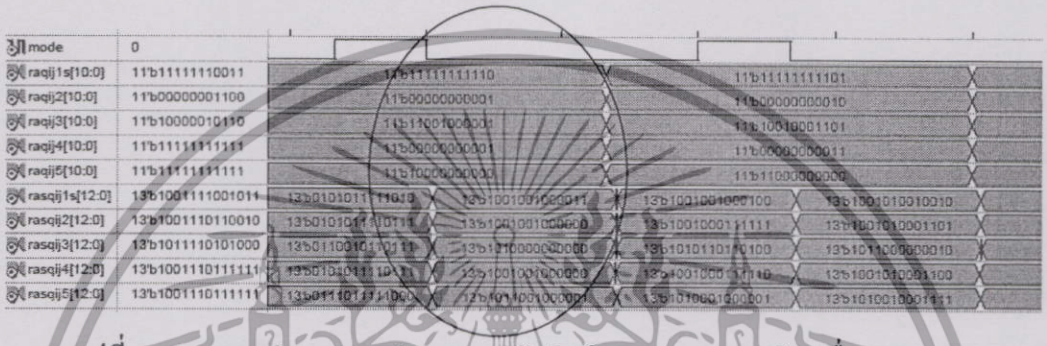
5.1.2.7 โครงสร้างชุดตัวดำเนินการบวกแบบคิดเครื่องหมาย (+, -)

การบวกที่ออกแบบให้ใช้สำหรับการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำนั้นทำได้สองวิธี คือการบวกแบบคิดเครื่องหมายและการบวกแบบไม่คิดเครื่องหมาย ซึ่งการบวกแบบคิดเครื่องหมายนี้จะอยู่ในขั้นตอนของการหาค่าบิตโนด ($L(q_{ij})$) ซึ่งการบวกนี้จะรับบิตข้อมูลมาทำการคำนวณหาผลบวกรวม จากนั้นนำมาลบกับข้อมูลเดิมที่ต้องการหา เมื่อนำมาออกแบบการทำงานและออกแบบวงจรด้วยโปรแกรม Xilinx ISE จะได้วงจรในระดับเกตดังแสดงในรูปที่ 5.18



รูปที่ 5.18 ผังวงจรระดับ RTL ของชุดดำเนินการบวกแบบคิดเครื่องหมาย (+, -)

การจำลองการทำงานของตัวดำเนินการบวกแบบไม่คิดเครื่องหมาย ซึ่งประกอบไปด้วยสัญญาณควบคุมการบวก บิตข้อมูลชุดที่ 1 บิตข้อมูลชุดที่ 2 บิตข้อมูลชุดที่ 3 บิตข้อมูลชุดที่ 4 และบิตข้อมูลชุดที่ 5 และระบบจะทำงานดังนี้คือถ้าต้องการหาค่าของบิตข้อมูลชุดที่ 1 ระบบจะรับค่าบิตข้อมูลพร้อมกันทั้ง 5 ชุด จากนั้นจะทำการบวกรวม และเมื่อได้ผลบวกรวมแล้วจะนำผลบวกรวมที่ได้มาทำการลบด้วยบิตข้อมูลชุดที่ 1 อีกครั้ง ในกรณีที่ต้องการหาบิตข้อมูลชุดที่ 2 ก็เช่นกันคือ จะนำผลบวกรวมมาทำการลบด้วยบิตข้อมูลชุดที่ 2 จากนั้นก็จะทำการส่งข้อมูลต่อไปเพื่อนำไปใช้งานในบล็อกต่อไป การทำงานสามารถแสดงในรูปที่ 5.19



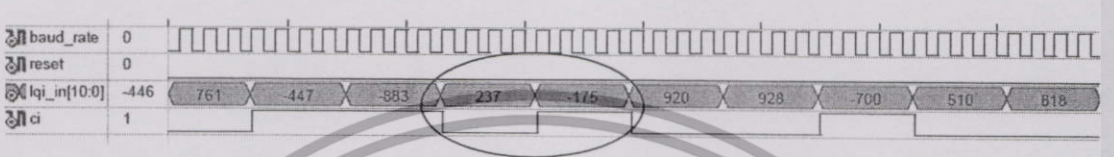
รูปที่ 5.19 ผลการจำลองการทำงานของตัวดำเนินการบวกแบบคิดเครื่องหมาย (+, -)

5.1.2.8 โครงสร้างชุดตัวดำเนินการในการตัดสินใจแบบหยาบ (Hard Decision)

การบวกที่ออกแบบให้ใช้สำหรับการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ การตัดสินใจแบบหยาบเป็นขั้นตอนสุดท้ายในการถอดรหัส เป็นการตัดสินใจว่าข้อมูลที่รับมาได้จากการถอดรหัสแบบละเอียดมาทำการตัดสินใจแบบหยาบในรูปแบบ 0 หรือ 1 เมื่อนำมาทำการออกแบบการทำงานและสังเคราะห์วงจรชุดการตัดสินใจแบบหยาบ ด้วยโปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ในระดับเกต แสดงได้ดังในรูปที่ 5.20

รูปที่ 5.20 ผังวงจรระดับ RTL ของชุดดำเนินการในการตัดสินใจ (Hard Decision)

การจำลองการทำงานของตัวดำเนินการในการตัดสินใจ ประกอบไปด้วยสัญญาณรีเซต สัญญาณนาฬิกา บิตข้อมูลอินพุต และสัญญาณเอาต์พุต (Output) ระบบจะเริ่มทำงานโดยการรับบิตอินพุต ขนาด 11 บิต มาทำการตัดสินใจว่าสัญญาณเอาต์พุตความจะเป็น 01 ในกรณีที่บิตข้อมูลทั้ง 11 ที่เป็นเลขฐานสองเปลี่ยนรูปมาอยู่ลักษณะของเลขฐานสิบ และมีค่าน้อยกว่าศูนย์ค่าที่ได้จะเป็น 1 แต่ถ้าค่าที่ได้จากการตัดสินใจมากกว่า 1 ค่าที่ได้จะเป็นศูนย์ จากนั้นส่งข้อมูลเพื่อนำไปใช้งานต่อไป และสามารถจำลองการทำงานได้ดังในรูปที่ 5.21

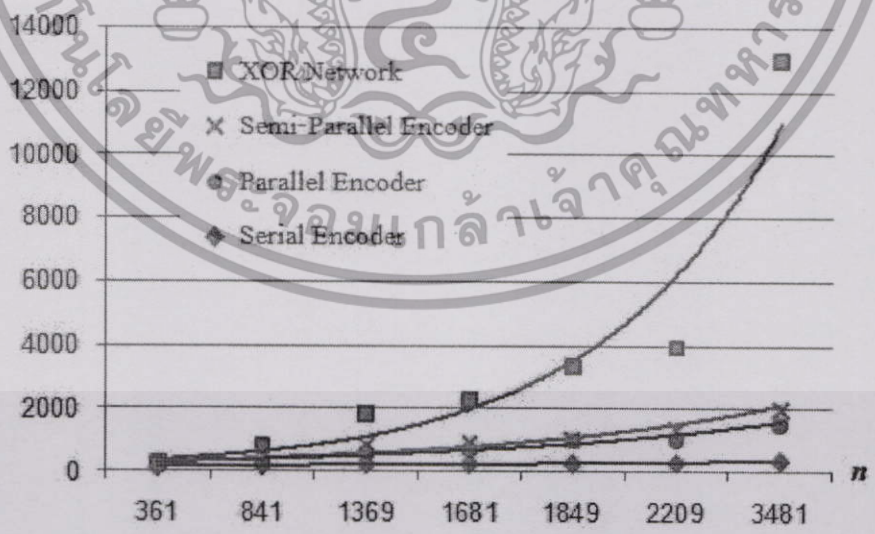


รูปที่ 5.21 ผลการจำลองการทำงานของตัวดำเนินการในการตัดสินใจ (Hard Decision)

5.2 การวิเคราะห์ผลและเปรียบเทียบสมรรถนะ

5.2.1 การวิเคราะห์ผลการใช้ทรัพยากร

จากวงจรเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำที่ทำงานแบบอนุกรมและแบบขนาน สามารถเปรียบเทียบทรัพยากรที่ต้องใช้ในการออกแบบได้ดังแสดงในรูปที่ 5.22 การใช้จำนวน Logic Elements ของวงจรเข้ารหัสแบบ XOR Network จะใช้จำนวน Logic Elements จำนวนสูงมากเมื่อบล็อกข้อมูลที่จะทำการเข้ารหัสมีขนาดใหญ่ขึ้น ในด้านของความเร็วในการทำงาน วงจรในลักษณะนี้ไม่สัญญาณนาฬิกาควบคุมการทำงาน



รูปที่ 5.22 การใช้จำนวน Logic Elements

การเข้ารหัสด้วยวงจรแบบอนุกรมมีสัญญาณนาฬิกาควบคุม เมื่อจำนวนบล็อกข้อมูลมีขนาดใหญ่ขึ้น แต่จำนวน Logic Elements ที่ถูกใช้งานจะน้อยมากเมื่อเทียบกับ XOR Network และการเข้ารหัสแบบขนาน การเข้ารหัสด้วยวงจรแบบขนานนั้นเมื่อเทียบในเรื่องของความเร็วในการทำงานนั้นทำงานได้เร็วกว่าทุกวิธีที่กล่าวมานั่นคือ วงจรมีสัญญาณนาฬิกาเป็นตัวควบคุมและสามารถเข้ารหัสได้พร้อมกันในครั้งเดียว ดังในรูปที่ 4.2 สามารถเข้ารหัสได้เร็วกว่าวงจรแบบอนุกรมถึง 3 เท่า

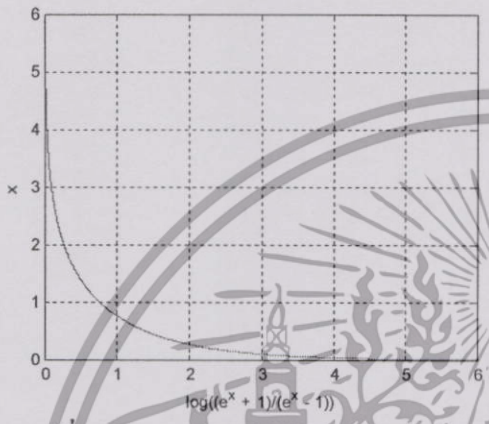
ตารางที่ 5.2 การใช้จำนวน Logic Elements

ประเภท		การเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก			
เมตริกซ์สลับที่ (P)	คำรหัส($\mathbf{c}$)	อนุกรม	ขนาน	กึ่งขนาน	โครงข่าย XOR
$L = 19$	$n = 361$	124	203	213	228
$L = 29$	$n = 841$	180	421	480	813
$L = 37$	$n = 1369$	236	598	849	1,801
$L = 41$	$n = 1681$	257	719	930	2,249
$L = 43$	$n = 1849$	269	1001	1,049	3,314
$L = 47$	$n = 2209$	286	1,027	1,342	3,925
$L = 59$	$n = 3481$	359	1,536	1,978	13,001

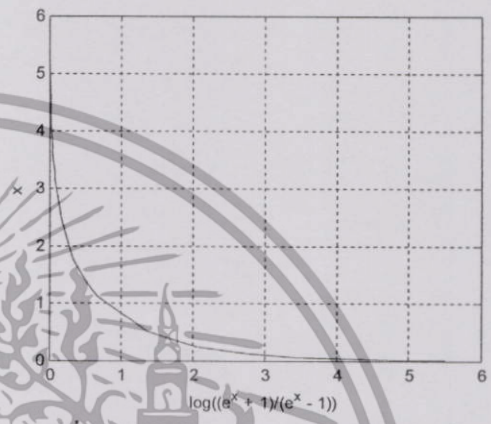
การเปรียบเทียบประสิทธิภาพการทำงานการเข้ารหัสที่อัตรารหัส (Rate) ประมาณ 0.88 – 0.9 ซึ่งได้นำการเข้ารหัสแบบดั้งเดิมคือ XOR Network มาเปรียบเทียบกับการเข้ารหัสแบบขนานและการเข้ารหัสแบบอนุกรม ซึ่งได้เปลี่ยนแปลงข้อมูลที่จะทำการเข้ารหัสด้วยตั้งแต่ 361 – 3481 บิต (Bit) จะเห็นได้ว่าการใช้ทรัพยากรหรือจำนวน Logic Elements จะเพิ่มขึ้นตามไปด้วย การใช้จำนวน Logic Elements มีค่าตั้งแต่ 124 – 13,001 Logic Element เริ่มต้นที่บล็อกข้อมูลขนาดเล็กคือที่ $\mathbf{c}$ = 361 บิตและ P = 19 การใช้จำนวน Logic Elements ความแตกต่างกันไม่มาก แต่เมื่อบล็อกข้อมูลมีขนาดใหญ่ขึ้นที่ P = 43 และ $\mathbf{c}$ = 1,849 บิต การใช้จำนวนทรัพยากรเริ่มแตกต่างกันคือ วงจรที่ออกแบบในลักษณะ XOR Network จะใช้จำนวน Logic Element 3,314 แต่ในขณะที่การเข้ารหัสแบบ Serial Encoder ใช้ 269 Logic Element ซึ่งน้อยมากเมื่อเทียบกับ XOR Network และจากกราฟในรูปที่ 5.22 จะเห็นได้ว่าเมื่อบล็อกข้อมูลเพิ่มขึ้นการเข้ารหัสแบบ XOR Network กราฟการใช้ทรัพยากรจะอยู่ในรูปของเอ็กซ์โปเนนเชียล แต่กราฟการใช้ทรัพยากรของ Serial Encoder จะอยู่ในลักษณะเชิงเส้น

5.2.2 การเปรียบเทียบประสิทธิภาพ

ในการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำในกรณีที่ต้องการออกแบบสำหรับฮาร์ดแวร์ การคำนวณค่าลอการิทึมในสมการ การหาค่าบิตโนดนั้นมีความซับซ้อน ดั้งนั้นการหาโดยการใช้เลขจำนวนจริงเป็นเรื่องที่ซับซ้อนและไม่เป็นที่นิยมสำหรับฮาร์ดแวร์ การประมาณค่าเป็นเป็นวิธีหนึ่งที่เหมาะสมในการหาค่าลอการิทึม



รูปที่ 5.23 การประมาณค่าลอการิทึมด้วยเลขจำนวนจริง

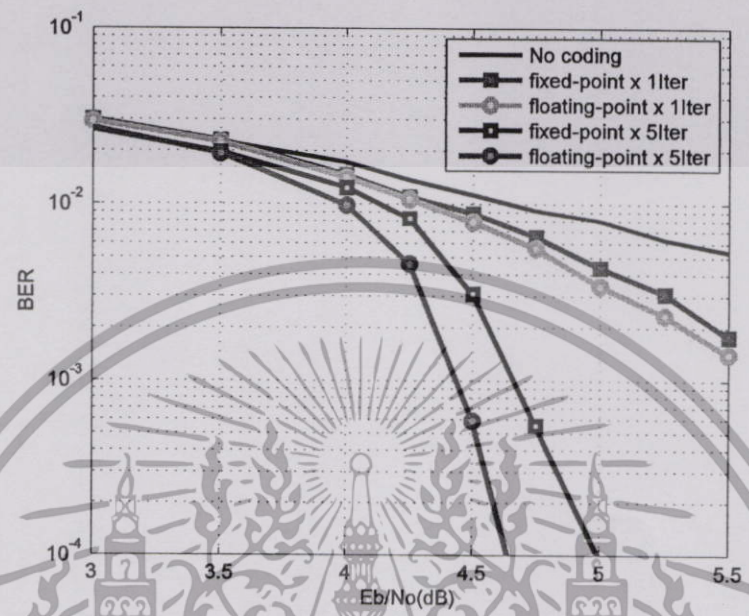


รูปที่ 5.24 การประมาณค่าลอการิทึมด้วยเลขตายตัว

จากกราฟในรูปที่ 5.23 แสดงให้เห็นถึงการคำนวณค่า $\log((e^x+1)/(e^x-1))$ โดยใช้เลขจำนวนจริง (floating-point) การใช้เลขจำนวนจริงในการคำนวณหาค่าบิตโนดย่อมให้ผลดีต่อสมรรถนะของระบบโดยรวมในการถอดรหัส แต่ในด้านฮาร์ดแวร์จะทำให้การจำนวนทรัพยากรเพิ่มขึ้นมาก การทำงานช้าลงเนื่องจากความซับซ้อนในการคำนวณค่าลอการิทึม จากกราฟในรูปที่ 5.24 เป็นการประมาณค่าลอการิทึมโดยใช้เลขตายตัว (fixed-point) การประมาณค่าโดยใช้เลขตายตัวอาจจะทำให้ประสิทธิภาพในการแก้ไขข้อผิดพลาดดีน้อยกว่าการใช้เลขจำนวนจริงในการคำนวณ แต่การคำนวณด้วยเลขจำนวนจริงมีผลดีกับในด้านการออกแบบฮาร์ดแวร์ เนื่องจากเลขจำนวนจริงจะได้รับการคำนวณและได้รับการเก็บไว้ในหน่วยความจำถาวร (ROM) เมื่อต้องการใช้งานค่าต่างๆ ที่ได้เก็บในหน่วยความจำถาวร ก็ทำการอ่านค่าออกมาใช้งาน การคำนวณค่าบิตโนดในลักษณะนี้ทำให้สามารถทำได้อย่างรวดเร็วเนื่องจากไม่ต้องใช้เวลาในการคำนวณค่าลอการิทึม

การทดสอบสมรรถนะของการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำในช่องสัญญาณรบกวนแบบขาว (AWGN) เป็นการเปรียบเทียบเพื่อจำลองสมรรถนะเปรียบเทียบอัตราบิตผิดพลาดของข้อมูล ในขั้นตอนการถอดรหัสในโดเมนลอการิทึม โดยใช้การประมาณค่า $\log((e^x+1)/(e^x-1))$ ทำการเปรียบเทียบการถอดรหัสด้วยเลขจำนวนจริง กับการประมาณค่าในการถอดรหัสด้วยเลขตายตัว แสดงให้เห็นสมรรถนะของระบบโดยรวมที่ความยาวคำรหัส 3481 บิต อัตรารหัส 0.9 โดย

ใช้จำนวนการวนลูป 1 และ 5 [15] ตามลำดับ พารามิเตอร์ที่แตกต่างกันคือ การใช้เลขจำนวนจริงในการหาค่าลอการิทึมกับการใช้เลขตายตัว



รูปที่ 5.25 สมรรถนะอัตราบิตผิดพลาดของการคำนวณของเลขตายตัวและเลขจำนวนจริง

จากรูปที่ 5.25 แสดงให้เห็นว่าในกรณีที่ไม่มีการเข้ารหัสช่องสัญญาณสมรรถนะการแก้ไขข้อผิดพลาดจะต่ำกว่าในกรณีที่มีการเข้ารหัสช่องสัญญาณ ในการเข้ารหัสช่องสัญญาณที่มีการวนลูป 1 ครั้ง ได้ทำการเปรียบเทียบการแก้ไขบิตผิดพลาดระหว่างการใช้เลขจำนวนจริงและเลขตายตัวนั้นจะต่างกัน ที่อัตราบิตผิดพลาด $3x10^{-3}$ ค่าของอัตราส่วนของสัญญาณต่อสัญญาณรบกวน (SNR) เมื่อเทียบในระดับบิต (Eb/No) จะมีค่าต่างกันประมาณ 0.1 เดซิเบล (dB) และต่อมาใช้การวนลูปในการถอดรหัส 5 ครั้ง การแก้ไขบิตผิดพลาดที่ใช้การคำนวณด้วยเลขตายตัวให้สมรรถนะดีกว่าการแก้ไขโดยใช้การคำนวณแบบเลขตายตัว ที่อัตราบิตผิดพลาด $1x10^{-3}$ ค่าของอัตราส่วนของสัญญาณต่อสัญญาณรบกวนเมื่อเทียบในระดับบิต จะมีค่าต่างกันประมาณ 0.3 เดซิเบล สำหรับระบบที่ต้องการให้ประสิทธิภาพในการถอดรหัสแบบเลขตายตัวให้มีสมรรถนะเพิ่มนั้นสามารถทำได้โดยการเพิ่มความละเอียดในการเก็บค่าเลขตายตัว และการเพิ่มความละเอียดในการแบ่งระดับในสมการที่ 4.17 โดยขึ้นอยู่กับความเหมาะสมของการประยุกต์ใช้งานในระบบสื่อสารแต่ละประเภท

บทที่ 6
สรุปผลและข้อเสนอแนะ

รหัสพาริตี้เช็คความหนาแน่นต่ำจัดเป็นเทคนิครหัสช่องสัญญาณในที่ใช้ในการแก้ไขบิตผิดพลาดในลักษณะวนซ้ำ ซึ่งมีสมรรถนะในการแก้ไขบิตผิดพลาดเข้าใกล้ขีดจำกัดของแชนนอล และมีงานวิจัยหลายฉบับแสดงให้เห็นถึงสมรรถนะของรหัสพาริตี้เช็คความหนาแน่นต่ำว่าทำงานได้ดีกว่ารหัสเทอร์โบ [16] จึงทำให้ได้รับความสนใจจากกลุ่มนักวิจัยต่างๆ โดยได้รับการพัฒนาในหลายด้าน ส่วนหนึ่งคือการพัฒนาเทคนิคการเข้ารหัสในรูปแบบของซอฟแวร์ และการออกแบบวงจรสำหรับฮาร์ดแวร์

6.1 สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้นำเสนอการออกแบบการเข้ารหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก ซึ่งการออกแบบได้ ในรูปแบบต่างๆ โดยพัฒนาจากวงจรเข้ารหัสแบบเก่าคือการเข้ารหัสแบบโครงข่าย XOR การเข้ารหัสแบบเก่านี้ เมื่อทำการเพิ่มการรับ-ส่งข้อมูลหรือการเพิ่มขนาดของคำรหัส (c) จะใช้ทรัพยากรเพิ่มมากขึ้นอย่างไม่เป็นเชิงเส้น และวงจรเข้ารหัสในลักษณะโครงข่ายเอ็กซ์ออนี้ไม่ทำงานภายใต้การควบคุมของสัญญาณนาฬิกา ดังนั้นการเพิ่มหรือลดความเร็วในการทำงานจึงไม่สามารถควบคุมได้ ดังนั้นจึงไม่เหมาะสมสำหรับระบบการเข้ารหัสที่อัตรารหัสสูง ดังนั้นวิทยานิพนธ์นี้จึงนำเสนอการออกแบบวงจรเข้ารหัสแบบใหม่คือการเข้ารหัส ในลักษณะอนุกรม การเข้ารหัสในลักษณะกึ่งขนาน และการเข้ารหัส ซึ่งวงจรที่ออกแบบสามารถทำงานภายใต้การควบคุมของสัญญาณนาฬิกา การทำงานสามารถทำงานภายใต้สัญญาณนาฬิกาที่ความถี่สูงได้ ซึ่งให้ประสิทธิภาพในการทำงานได้สูงกว่าวงจรเข้ารหัสแบบโครงข่าย XOR

การถอดรหัสพาริตี้เช็คความหนาแน่นต่ำเป็นทคสอบสมรรถนะของการออกแบบการถอดรหัส ซึ่งได้เปรียบเทียบการประมาณค่าในขั้นตอนการถอดรหัสด้วยเลขจำนวนจริงกับเลขตายตัว เพื่อเปรียบเทียบในกรณีใช้หน่วยความจำถาวรในการเก็บค่า $\log((e^x + 1)/(e^x - 1))$ เพื่อใช้ในการเปิดตารางหาค่าลอการิทึม การใช้เลขตายตัวในการถอดรหัสจะใช้ทรัพยากรน้อยกว่าการใช้เลขจำนวนจริง และเหมาะสมสำหรับการออกแบบฮาร์ดแวร์

6.2 ปัญหาและข้อเสนอแนะ

การถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิก ในขั้นตอนการเข้ารหัสนั้นในแบบขนานนั้นไม่อาจจะระบุการใช้ทรัพยากรได้ เนื่องจากการออกแบบวงจรเข้ารหัสในแต่ละครั้ง

ต้องได้รับการคิดสมการการออกแบบวงจรทุกครั้ง แต่ข้อดีของการเข้ารหัสแบบขนานคือการเข้ารหัสนั้นใช้เพียงแต่ บิตข้อมูลข่าวสารเท่านั้นซึ่งต่างจากการเข้ารหัสในแบบอนุกรมและกึ่งขนาน

จากผลการทดสอบสมรรถนะในการถอดรหัส จะเห็นว่าการแก้ไขบิตผิดพลาดของการการประมาณค่า $\log((e^x+1)/(e^x-1))$ โดยใช้เลขจำนวนจริงในกรณีที่ใช้พลังงานในการส่งเท่ากันจะแก้ไขบิตผิดพลาดได้ดีกว่าการประมาณโดยใช้เลขตายตัว แต่สำหรับระบบที่ต้องการ การถอดรหัสที่มีสมรรถนะดีขึ้นสามารถทำได้ 2 วิธีคือ

- แบ่งระดับการประมาณค่า $\log((e^x+1)/(e^x-1))$ ให้ละเอียดขึ้น
- เพิ่มบิตในการเก็บค่าให้ละเอียดขึ้น

การออกแบบการเข้ารหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบบควอไซไซคลิกสำหรับฮาร์ดแวร์นั้น ขึ้นอยู่กับการนำไปประยุกต์ใช้งาน

บรรณานุกรม

[1] R. G. Gallager, "**Low density parity check codes**," IRE Trans. Inform. Theory, vol. IT-8, pp. 21–28, Jan 1962.

[2] D. J. C. MacKay and R. M. Neal, "**Good codes based on very sparse matrices**," in Cryptography and Coding 5th IMA Conf., C. Boyd, Ed., Lecture Notes in Computer Science, no. 1025. Berlin, Germany: Springer, 1995, pp. 100–111.

[3] D. J. C. Mackay, "**Good Error-Correcting Codes Based on Very SparseMatrices**," IEEE Trans.Inform.Theory, vol.45, pp.339-431, Mar.1999.

[4] Chung, S.-Y., Forney, G. D., Jr., Richardson, T. J., and Urbanke, R. L., "**On the design of low-density parity-check codes within 0.0045 dB of the shannon limit**," Electron. Lett., vol. 5, pp. 58–60, Feb. 2001.

[5] C. E. SHANNON A Mathematical Theory of Communication

[6] M. P. C. Fossorier "**Quasi-Cyclic Low-Density Parity-Check Codes From Circulant Permutation Matrices**" IEEE Trans. Inf. Theory, vol. 50, no. 8, Aug. 2004

[7] S. Chae , Y. Park "**Low Complexity Encoding of Improved Regular LDPC Codes**" IEEE 2004.

[8] Y. Sun , M. Karkooti , J. R. Cavallaro "**High Throughput, Parallel, Scalable LDPC Encoder/Decoder Architecture for OFDM Systems**"

[9] Giuseppe C., S. Shamai and S. Verdu "**Noiseless Data Compression with Low-Density Parity-Check code**"

[10] S. Myung, K.Yang, J. Kim, "**quasi - cyclic ldpc codes for fast encoding**," IEE Trans. on Information Theory. vol. 51. no.8. Aug. 2005

[11] Z. Wang , Z. Cui "**A Memory Partially Parallel Decoder Architecture for QC-LDPC Codes**" IEEE 2005.

[12] Z. Li and B. V. K. V. Kumar, "**A class of good quasi-cyclic lowdensityparity check codes based on progressive edge growth graph**,"Thirty-Eighth Asilomar Conference on Signals, Systems andComputers, vol. 2, pp. 1990-1994, 2004.

[13] J. J. F. Cavanagh, "**Digital Computer Arithmetic Design and Implementation**," McGraw-Hill, Inc., New York, 1985.

[14] ผศ.ดร.พรชัย ทรัพย์นิธิ, "การสื่อสารดิจิตอล," คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ.2549

[15] J. L. Fan, **"Constrained coding and soft iterative decoding for storage,"** PhD thesis, Stanford University,1999.

[16] Richardson T.J., Shokrollahi MA, Urbank R.L. **"Design of capacity-approaching irregular low-density parity-check codes,"** Information Theory IEEE Trans.volume 47, pp.619-639, Feb 2001.

ภาคผนวก ก.

โปรแกรมจำลองการเข้ารหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำ

```
%=========================================================================
```

การทดสอบการเข้ารรหัสและถอดรหัสพาริตี้เช็คความหนาแน่นต่ำแบบ

ควอไซไซคลิกในรูปแบบของลอกโดเมน

```
%=========================================================================

clc; clear all;

%*************************************************************************%
%Set parameter of the parity-check matrix check
L=67;       %permutation matrix size
J=6;        %row weight of triangular matrix
K=61        %column weight of triangular matrix
%*************************************************************************%

H=zeros(J*L,K*L);                   %Parity-check matrix size
I=eye(L,L);                         %Set Identity be LxL
P=[I(2:L,:);I(1,:)];                %Set P be the LxL permutation matrix
blockz=(10^6)/(L*K)*1000
for loopR = 1:J
   if loopR == 1
        col = 1;
    for loopC = 1:K
        H(loopR:loopR+L-1,col:col+L-1) = I;     %Set identity matrix
        col = col+L;
    end
else
        x = 0;
        n = ((loopR - 1)*L) + 1;
        for nn = n:L:K*L
   H(n:n + L-1,nn:nn + L-1) = P^((loopR - 1)*x);%parity check matrix
   x = x + 1;
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [1] Find all index in H
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Check_node,Bit_node]=size(H);%find Check_node and Bit_node
[a,b]=find(H);                      %find position of "1" in H
H_t=H';                             %find H transpose
[c,d]=find(H_t);                    %find position of "1" in H transpose
s=size(c,1);                %find total number of "1" in H transpose

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [2] Define SNR Range
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dB=[0:2, 3:0.5:3.5, 4:0.25:5.5];    %range of SNR in dB
iter_no=[1 5 20];
SNR_bit=10.^(dB/10);                %Eb/No coversion from dB to decimal
No_uncode=1./SNR_bit;               %since Eb=1
R=1-(J/K)                           %code rate
No=No_uncode/R;
MaxBlockNum=blockz;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [3] Channel simulation and LDPC decoder
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for ii=1:length(iter_no)            %maximum number of decoder iterations
```

```
iteration=iter_no(ii);
BER(ii,:)=zeros(1,length(dB));      %array for channel error rate

for z=1:length(dB)      %loop for testing over range of SNR value
      tot_err=0;        %value for compute BER
      loop_num=0; %value for testing over range of codeword-errors
   while (loop_num<=MaxBlockNum)&(tot_err<=1000) %while loop
      m=randint(1,(L*(K-J))); %Generate message bit 0, 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Compute Parity bit
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        p=parity_cal(L,J,m,H);

        u=[p,m];          %Code word
        temp=2*u(:)-1;    %bpsk mod
        tx=temp';         %tx is codeword to be transmitted (bpsk)
        sigma=sqrt(No(z)/2);  %standare deviation
        rx=tx+sigma*randn(1,length(tx));  %rx is receive signal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % LDPC decoder%Compute error
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        Ci=logdomain(H,sigma,iteration,rx);
        tot_err=tot_err+sum(u~=Ci); %Compute error
        loop_num=loop_num+1;
    end %while loop
    BER(ii,z)=tot_err/(loop_num*Bit_node);      %Compute BER
    fprintf(1,'\n Simulation finished for SNR : %d \n',dB(z));
end %loop for testing over range of SNR values
    fprintf(1,'\n finished for iteration : %d \n',iter_no(ii));
end
semilogy(dB,BER(1,:),'b-o')
grid
hold on
semilogy(dB,BER(2,:),'r-o')
semilogy(dB,BER(3,:),'k-o')
legend('1 iteration','5 iteration','20 iteration')
title('Simulation result of LDPC Log-Approx Decoder')
ylabel('BER')
xlabel('Eb/No(dB)')

%=========================================================
            function Ci=logdomain(H,sigma,iteration,rx)
%=========================================================
Lci=(-8*rx);
[ci di]=size(H);

%==================Step 2 Update check nodes==============
Lqij=H.*repmat(Lci, ci, 1);
Lqija=Lqij;

%=======================================================
for n=1:iteration
    sLqij = sign(Lqij);
    aLqij = abs(Lqij);
for l=1:ci
    posi=find(Lqij(l,:));
    po=length(posi);
    apLqij=[];
    sLa=[];
    for li=1:po
```

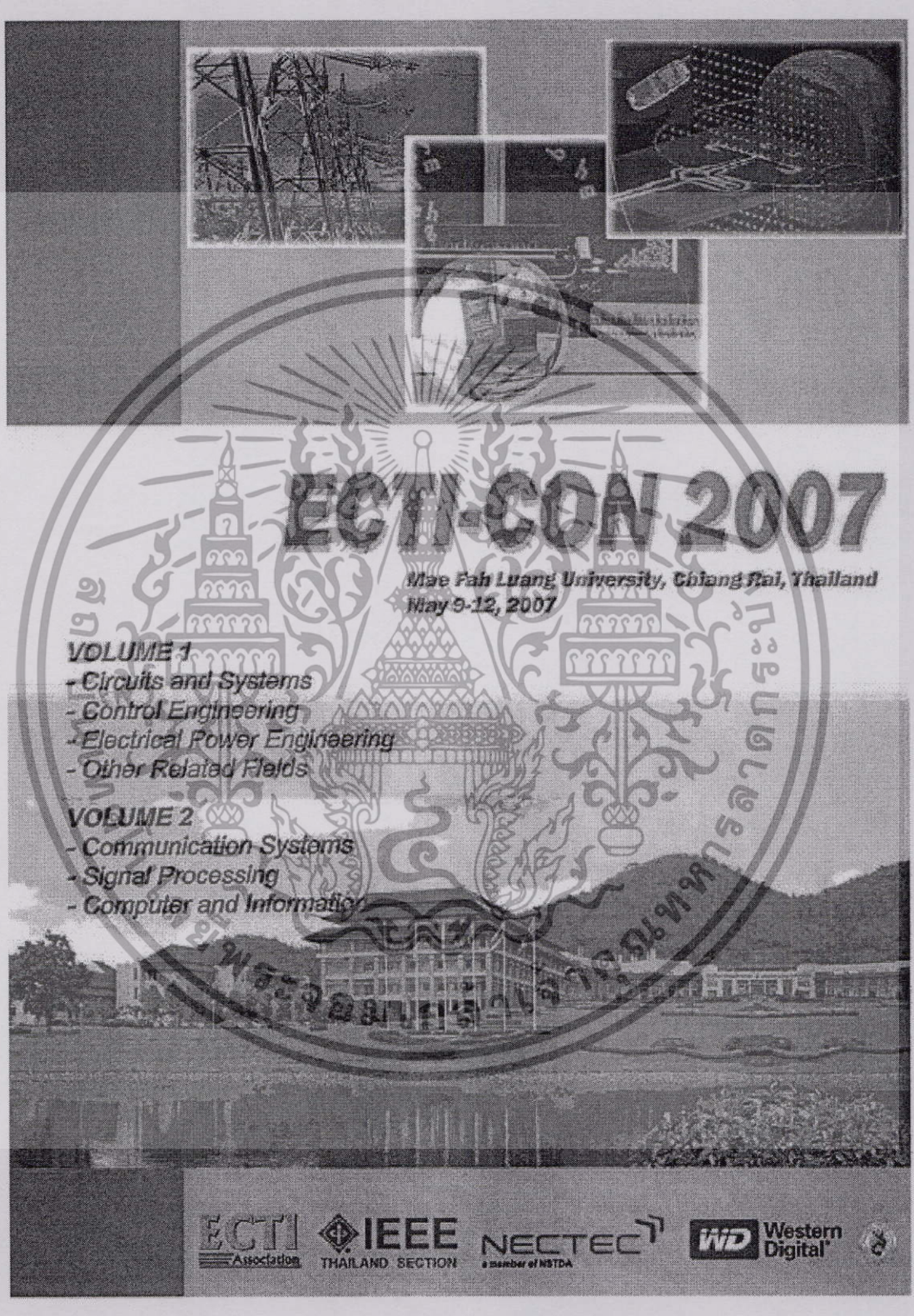
```
        La=aLqij(l,posi(li));
        apLqij(li)=log((exp(La) + 1)/(exp(La) - 1));%approx_exa(La)
        sLa(li)=sLqij(l,posi(li));%find signal
    end%li
    mLqij=sum(apLqij);
    poss=prod(sLa);
    for fi=1:po
        sLb=poss*sLa(fi);%sum of signal
        atLqij=mLqij-apLqij(fi);
        appLqij(l,posi(fi))=log((exp(atLqij) + 1)/(exp(atLqij) - 1));
%sLb*approx_exa(atLqij)
    end%fi
end%l
appLqij;

%=============================================================
apl=appLqij';
for r=1:di
    rab=[];
    posr=find(apl(r,:));
    pos=length(posr);
    rab=sum(apl(r,:));
    for lr=1:pos
        Lrji(r,posr(lr))=rab-apl(r,posr(lr));
    end%lr
end%r
Lqij=Lqija+Lrji';
end%n

%=============================================================
cl=appLqij;

%=====================Step 4 Soft decision====================
        for sof=1:di
        LQ(sof)=Lci(sof)+sum(cl(:,sof));
        End

%=====================Step 5 Hard decision====================
        for hard=1:di
            if LQ(hard) < 0
                Ci(hard)=1;
            else
                Ci(hard)=0;
            end
        end
%=============================================================
                    % END Step 5 Hard decision
%=============================================================

%=============================================================
                function p=parity_cal(L,J,m,H)
%=============================================================
%Initialrization
index_row=0;
%p=zeros(1,L*J);
for i=1:L*J
%find sub matrix
    [ai,bi]=find(H(L*J-index_row,:));
    ci=size(bi,2)-1;        %keep range of sub matrix
    p_temp=0;               %initialrize parity matrix
    for ii=1:ci             %Parity cal. loop
```

```
        if bi(ii+1)>(L*J)
           p_temp=xor(p_temp,m(bi(ii+1)-(L*J)));
        else
           p_temp=xor(p_temp,p(bi(ii+1)));
        end %end of if loop
     end %end of for loop
    p(L*J-index_row)=p_temp;
    index_row=index_row+1;
end %end of for loop

%=====================================================================
                          % END of Encoder
%=====================================================================
```

ภาคผนวก ข.

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

1. **P. Leekul**, S. Chiwapreecha, K. Sripimanwat, and P. Supnithi, **"Block-Type Encoder Design of Modified Array-Based Low-Density Parity-Check Codes,"** Proceedings of ECTI-CON 2007, Chiang Rai, Thailand, pp. 951-954, May 2007.



Subblock Encoder Design of Modified Array-Based Low-Density Parity-Check Codes

P. Leekul, S. Chivapreecha, †K. Sripimanwat, and P. Supnithi
Faculty of Engineering and Research Center for Communications and Information Technology
I/U CRC in Data Storage Technology and Applications
King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520
†National Electronics and Computer Technology (NECTEC), Thailand Science Park, Klongluang, Patumthani 12120
E-mail: s8060944@kmitl.ac.th, sorawat@telecom.kmitl.ac.th, ksripima@ieee.org, ksupornc@kmilt.ac.th

Abstract **- Modified array code is considered a Quasi-cyclic LDPC codes with simple encoding structure. The parity-check matrix is based on identity matrix and permutation matrix. In this paper, we present a new encoding method with encoder circuit designs presented. The encoding is done on one subblock at a time compared to the direct XOR network and does not need matrix inversion. Both Serial and Semi-Parallel circuit structures are constructed for high-rate applications. The circuit complexity in terms of number of logic elements are compared against direct XOR networks for high-rate codes suitable for magnetic recording systems. The direct XOR networks show exponential complexity, while the proposed encoding methods are much simpler and show linear complexity.**

Keywords**: Quasi-cyclic LDPC codes, FPGA, Modified array codes.**

I. INTRODUCTION

Low-density parity-check or LDPC code is a linear block code defined by a sparse parity-check matrix **H**. The sparseness results in large minimum distance, hence, a large number of correctable errors can be achieved. The codes when combined with iterative processing leads to a performance approaching the Shannon's limit in an additive white Gaussian noise channel. Low Density Parity Check (LDPC) codes was originally discovered in 1960 by Gallager [1] and in 1962, but the high-complexity hindered its usefulness. It was not until in 1996 when in [2], the author rediscovers LDPC and presents the decoding method using Belief Propagation. In [3], S. Y Chung et. al. showed that LDPC code reaches a Shannon limit within 0.0045 dB. LDPC code comes into two types; regular LDPC code and irregular LDPC code. The encoding problem is typically complex and has been studied by many researchers [4-6]. A type of LDPC codes called quasi-cyclic LPDC (QC-LDPC) code has been studied recently due to the quasi-cyclic structure that allows simpler encoding scheme. In [4], the encoding is based on dividing the parity-check matrix **H** into submatrices to accommodate high-rate application. The encoding is still complex as matrix inversion is required. In [5] specific construction of QC-LDPC codes is based on Permutation matrix **P** to achieve a girth size of 12. The structure allows simpler encoding. In [6], encoding based on generator matrix **G** rather than **H** is presented instead. It is still based on breaking the matrix **H** into submatrices and then translates into the generator matrix **G**. In this paper, we present a block-type encoding of QC-LDPC codes for high-rate application. The simplicity is due to systematic encoding nature and encoding is done on a subblock basis. There is no matrix inversion required and for high-rate application, the numbers of encoding equations are small. In Section II, we review LDPC codes and QC-LDPC codes and some properties. In Section III, encoding equations are derived and encoder circuits are presented. The analysis results and discussions are shown in Section IV and the conclusions are in Section V.

II. LOW-DENSITY PARITY-CHECK (LDPC) CODES

A. Array-based LDPC code

An array-based LDPC code or Array code is a type of non-systematic LDPC codes with quasi-cycle property. The parity-check matrix **H** has a structure that allows efficient encoding process. The encoding is done with **H** rather than generator matrix **G** [6]. The quasi-cyclic property dictates that shifted message bits correspond to shifted parity-check bits.

The parity-check matrix **H** has the dimension $(n\text{-}k) \times n$, where k is the message length and n is the code length. The matrix **H** can be built from the permutation matrix P with dimension size $L \times L$ given by

$$P_{L\times L} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & & 0 \end{bmatrix} \quad (1)$$

The matrix **P** is a shifted identity matrix **I** one time. The matrix **H** can be built from

$$\mathbf{H} = \begin{bmatrix} I & I & I & \cdots & I \\ I & P & P^2 & \cdots & P^{k-1} \\ I & P^2 & P^4 & \cdots & P^{2(k-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I & P^{j-1} & P^{2(j-1)} & \cdots & P^{(j-1)(k-1)} \end{bmatrix} \quad (2)$$

Basically, the matrix P expands rowwise by j times and columnwise by k times, respectively.

B. Modified array-based LDPC code

A modified array-based LDPC code or a Modified Array code [7] is an array code with modification so that the codeword is systematic. The parity-check matrix **H** for modified array codes can be described by three parameters (q, j, k), where q is prime and integer number j and k. The matrix **H** has dimension jL x kL, i.e.,

$$\mathrm{H}(q,j,k)=\begin{bmatrix} I & I & I & \cdots & I & \cdots & I \\ 0 & I & P & \cdots & P^{(J-2)} & \cdots & P^{(k-2)} \\ 0 & 0 & I & \cdots & P^{2(J-2)} & \cdots & P^{2(k-2)} \\ & & & & \ddots & & \\ 0 & \cdots & 0 & I & P^{(J-2)} & \cdots & P^{(k-2)(k-j-1)} \\ 0 & 0 & \cdots & 0 & I & \cdots & P^{(j-1)(k-j)} \end{bmatrix} \quad (3)$$

The encoding complexity of the modified array code is linear [5]. An expansion of **H** matrix using **P** has a constraint that the rowwise expansion factor must be less than the column wise expansion factor, which is also less than the size of permutation matrix, i.e., $q \geq k \geq j$. The code rate can be computed from

$$R=\frac{k-j}{k}=1-\frac{j}{k} \quad (4)$$

Modified array code possesses a quasi-cyclic property and has a linear encoding complexity.

III. Encoder Circuit of LDPC Codes

There are various ways to encode LDPC codes, but in this section, we review direct encoding method of LDPC code. The encoding complexity is linear. Encoding is transforming a message into a codeword. For systematic code, this means adding parity bits. A simple way is to exploit the relationship between coderword and the parity-check matrix **H**. Given a codeword **c** with dimension of 1xn and the corresponding matrix parity check **H** with dimension of (n-k)xn and **0** is a zero matrix with dimension of (n-k)xn. The encoding circuit uses to modulo-2 addition or XOR operation. Consider the relationship

$$\mathbf{c}\mathbf{H}^T=\mathbf{0} \quad (5)$$

Transpose both sides of Eq. (5) to obtain

$$\mathrm{H}\mathbf{c}^T=\mathbf{0}^T \quad (6)$$

Let a systematic codeword **c** be in the form of

$$\mathbf{c}=\left[p_1\, p_2 \ldots p_{n-k} \;\middle|\; m_1 m_2 \ldots m_k\right]=[\mathbf{p}\;\;\mathbf{m}] \quad (7)$$

where the parity bits are positioned at the front part and the message bits are at the back part. In this way, the encoding can be done efficiently. The transpose of codeword **c** in Eq. (6) can be written as

$$\mathbf{c}^T=\begin{bmatrix}\mathbf{p}\\ \mathbf{m}\end{bmatrix} \quad (8)$$

Substituting Eq.(8) into Eq.(6), the parity-check equation can be derived. For example, given that a parity-check matrix **H** of a (3, 6) code is

$$\mathbf{H}=\begin{bmatrix}1&0&0&1&0&1\\0&1&0&1&1&0\\0&0&1&0&1&1\end{bmatrix} \quad (9)$$

Rewrite Eq.(8) as

$$\mathrm{H}\mathbf{c}^T=\mathbf{0}^T=\begin{bmatrix}1&0&0&1&0&1\\0&1&0&1&1&0\\0&0&1&0&1&1\end{bmatrix}\begin{bmatrix}p_1\\p_2\\p_3\\m_1\\m_2\\m_3\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix} \quad (10)$$

Then the parity-check equation can be listed as

$$\begin{aligned} p_1 + m_1 + m_3 &= 0 \\ p_2 + m_2 + m_1 &= 0 \\ p_3 + m_3 + m_2 &= 0 \end{aligned} \quad (11)$$

where '+' is a modulo-2 addition or an XOR operation. The parity bits can then be found from

$$\begin{aligned} p_1 &= m_1 + m_3 \\ p_2 &= m_2 + m_1 \\ p_3 &= m_3 + m_2 \end{aligned} \quad (12)$$

A simple circuit diagram can thus be shown in Fig. 1.

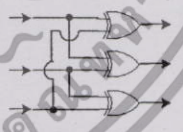


Fig. 1. Direct encoder of a (3, 6) code

Direct method is viable for the code with small k, but in practical applications, the message length can be up to 1024, for example. Given a code rate of ½, the codeword length is n = 2048. The number of parity-check equations is then 1024. The number of XOR operation can be very high and not suitable for encoder circuit design. In the next part, we propose a block-type encoder design.

A. Subblock-Type Encoder Design of Modified Array Codes

The encoder complexity can be reduced by encoding each subblock at a time. The matrix H consists of j subblocks rowwise and k subblocks columnwise. Consider a systematic codeword of length $N = kL$ with parity-check part at the front and the message part at the back, i.e.,

$$\mathbf{c} = \left[\mathbf{p}_1 \;\; \mathbf{p}_2 \;\; \mathbf{p}_3 \;\; \mathbf{p}_4 \;\; \cdots \;\; \mathbf{p}_j \;\; \mathbf{m}_1 \;\; \mathbf{m}_2 \;\; \cdots \;\; \mathbf{m}_{k-j-1} \;\; \mathbf{m}_{k-j}\right] \quad (13)$$

Consider the equation

$$\mathbf{H}\mathbf{c}^T = \begin{bmatrix} I & I & I & \cdots & I & \cdots & I \\ 0 & I & P & \cdots & P^{(j-2)} & \cdots & P^{(k-2)} \\ 0 & 0 & I & \cdots & P^{2(j-3)} & \cdots & P^{2(k-3)} \\ & & & \ddots & & & \\ 0 & \cdots & 0 & I & P^{(j-2)} & \cdots & P^{(j-2)(k-j+1)} \\ 0 & 0 & \cdots & 0 & I & \cdots & P^{(j-1)(k-j)} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_j \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_{k-j} \end{bmatrix} = \mathbf{0}^T \quad (14)$$

where $\mathbf{p}_i$, $i = 1,2,\ldots,j$ is a subblock of parity bits of size $L \times L$. Thus, each subsection of parity-check bits can be computed from two parts: the higher-indexed parity-check subsection and all the message subsections. The above equations can be written as a set of equations

$$\begin{aligned} \mathbf{p}_j &= \mathbf{m}_1 P^{(j-1)} + \mathbf{m}_2 P^{(j-1)2} + \mathbf{m}_3 P^{(j-1)3} + \cdots + \mathbf{m}_{k-j} P^{(j-1)(k-j)} \\ \mathbf{p}_{j-1} &= \mathbf{p}_j P^{(j-2)} + \mathbf{m}_1 P^{(j-2)2} + \mathbf{m}_2 P^{(j-2)3} + \cdots + \mathbf{m}_{k-j} P^{(j-2)(k-j+1)} \\ \mathbf{p}_{j-2} &= \mathbf{p}_{j-1} P^{(j-3)} + \mathbf{p}_j P^{(j-3)2} + \mathbf{m}_1 P^{(j-3)3} + \cdots + \mathbf{m}_{k-j} P^{(j-3)(k-j+2)} \\ &\vdots \\ \mathbf{p}_l &= \mathbf{p}_{l+1} P^{(l-1)} + \mathbf{p}_{l+2} P^{(l-1)2} + \cdots + \mathbf{p}_j P^{(l-1)(j-l)} \cdots + \mathbf{m}_{k-j} P^{(l-1)(k-j)} \\ &\vdots \\ \mathbf{p}_1 &= \mathbf{p}_2 + \mathbf{p}_3 + \cdots + \mathbf{p}_j + \mathbf{m}_1 + \mathbf{m}_2 \cdots + \mathbf{m}_{k-j} \end{aligned} \quad (15)$$

or in a simplified equation as

$$\begin{aligned} \mathbf{p}_j &= \sum_{i=1}^{k-j} \mathbf{m}_i P^{(j-1)i} \\ \mathbf{p}_l &= \sum_{i=1}^{j-l} \mathbf{p}_{l+i} P^{(l-1)i} + \sum_{i=1}^{k-j} \mathbf{m}_i P^{(l-1)(j-l+i)}, \; 1 \le l \le j-1 \end{aligned} \quad (16)$$

There is a total of j parity subblock equations. We start from the computation of $\mathbf{p}_j$ all the way to $\mathbf{p}_1$. Notice that subblock-type encoding does not require any real multiplication due to the fact that the result of a vector multiplying with permutation matrix is nothing but cyclic-shift the vector to the right by the power of the matrix $\mathbf{P}$.

B. Implementation of Subblock Encoder Circuit Using Serial Structure

The parity subblocks in Eq. (14) can be implemented in Serial or Semi-Parallel structure. Here, we discuss the serial circuit type. According to Eq.(14), the amount of shifts in each equation is determined by the power of $\mathbf{P}$ matrix and it needs to be stored in ROM. The encoder circuit consists of shift and modulo-2 addition operations as shown in Fig. 2. We first compute the parity subblock $\mathbf{p}_j$. The message subblock $\mathbf{m}_1$ is first read into the top registers, then shifted by the amount of $(j-1)$ times to the right. It is then placed at the bottom register. Next, the subblock $\mathbf{m}_2$ is read in and shifted by $(j-1)$ times to the right. The output is moved to the bottom and XOR with previous stored subblock. The entire process is repeated until $\mathbf{p}_j$ is fully computed. Each subblock of $\mathbf{p}_l$ is computed similarly except that the previously computed parity subblocks will be used together with message subblocks.

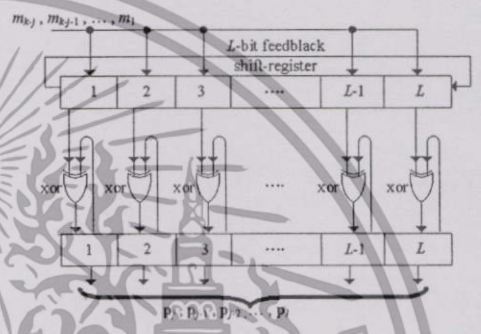


Fig. 2. Serial Encoder Circuit of Modified Array Codes

C. Implementation of Subblock Encoder Circuit Using Semi-Parallel Structure

To speed up the encoding process, and considering the encoding process in Eq. (15) and Eq. (16), the message subblocks are used in all equations, thus they can be shifted in advance and in Semi-Parallel, ready for the computation of higher-indexed parity subblocks as shown in Fig. 3.

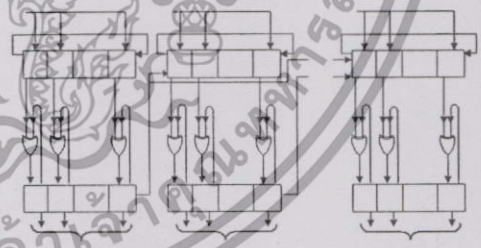


Fig. 3. Semi-Parallel Encoder Circuit of Modified Array Codes

IV. ANALYSIS RESULTS AND DISCUSSIONS

The circuit synthesis of serial and Semi-Parallel structure is performed on Quartus II with the utilized resources computed. The comparison of both designs against the direct XOR networks is illustrated in Table I.. The circuit analysis begins

with small message block size c = 361 bits and permutation matrix P of dimension 19x19. For this code, the complexity between Semi-Parallel circuit and direct XOR networks is similar, but for the block with large matrix P of dimension 43x43 and c = 1849 bits, XOR networks requires about 3 times logic elements than Semi-Parallel networks. Direct encoder circuits based on XOR Network requires much higher number of logic elements compared with subblock encoder for large message blocks. Larger message block size with larger permutation matrix size causes XOR networks to far exceed the complexity of subblock methods. In addition, direct XOR networks lack clock signal to synchronize the encoding process. For subblock encoding, serial circuits require less logic elements than the Semi-Parallel counterpart. The speed of Semi-Parallel circuit is of course faster. It is observed that Semi-Parallel encoding is 3 times faster than serial encoding circuits.

TABLE I. The Number of Required Logic Elements

Type		Quasi - Cyclic – LDPC Encoding		
Permutation Matrix	Block Size	Serial	Semi-Parallel	XOR Network
$P = 19$	c = 361	124	213	228
$P = 29$	c = 841	180	480	813
$P = 37$	c = 1369	236	849	1,801
$P = 41$	c = 1681	257	930	2,249
$P = 43$	c = 1849	269	1,049	3,314
$P = 47$	c = 2209	286	1,342	3,925
$P = 59$	c = 3481	359	1,978	13,001

The code rates of modified array codes analyzed here and listed in Table I ranges from 0.88 to 0.9. The high rate is suitable for many applications such as wireless/satellite communication under good conditions and high-density magnetic recording systems, for instance. The results from Table I is then plotted in Fig. 5. Notice the trend of required amount of logic elements. The direct XOR networks exhibit exponential increase as the block size goes up, while the serial subblock encoder shows linear complexity.

V. CONCLUSIONS

In this paper, we present a subblock encoder of modified array code, which is considered a type of quasi-cyclic LDPC codes suitable for high-rate applications such as magnetic recording systems and many other communication systems. The encoder equations are derived for each subblock at a time and then serial and Semi-Parallel circuit structures are presented. The circuit complexity in terms of number of logic

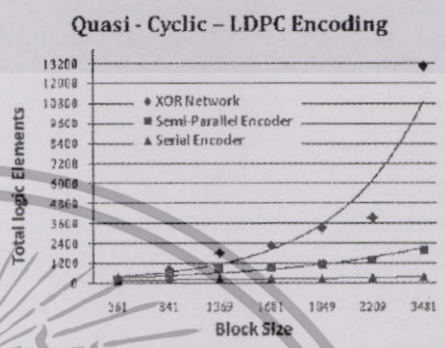


Fig. 5. The required number logic elements of direct XOR networks, Serial subblock encoder and Semi-Parallel subblock encoder

elements are compared for direct XOR networks, serial structure and Semi-Parallel structure. The direct XOR network exhibits exponential increase in complexity for large block size with lack of clock signal. The serial encoder has the lowest complexity for each message block size, but possess slower speed than the Semi-Parallel counterpart.

REFERENCES

[1] R. G. Gallager, "Low density parity check codes," IRE Trans. Inform. Theory, vol. IT-8, pp. 21–28, Jan 1962.

[2] D. J. C. MacKay and R. M. Neal, "Good codes based on very sparse matrices," in Cryptography and Coding 5th IMA Conf., C. Boyd, Ed., Lecture Notes in Computer Science, no. 1025. Berlin, Germany: Springer, 1995, pp. 100–111.

[3] Chung, S.-Y., Forney, G. D., Jr., Richardson, T. J., and Urbanke, R. L., "On the design of low-density parity-check codes within 0.0045 dB of the shannon limit," *Electron. Lett.*, vol. 5, pp. 58–60, Feb. 2001.

[4] Z. Wang , Z. Cui, "A Memory Partially Parallel Decoder Architecture for QC-LDPC Codes" *IEEE 2005*

[5] M. P. C. Fossorier, "Quasi-Cyclic Low-Density Parity-Check Codes From Circulant Permutation Matrices" *IEEE Trans. Information Theory, vol. 50, no. 8, Aug. 2004*

[6] Zongwang Li, et.al., "Efficient Encoding of Quasi–Cyclic Low-Density Parity-Check Codes" *IEEE Trans. on Information Theory, Vol. 54, no. 1, Jan. 2006*

[7] S. Myung, K.Yang, J. Kim, "quasi - cyclic ldpc codes for fast encoding," *IEEE Trans. on Information Theory, vol. 51, no.8, Aug. 2000.*

ประวัติผู้เขียน

นายประพัน ลี้กุล เกิดเมื่อวันที่ 26 มิถุนายน พ.ศ. 2523 ที่จังหวัดชลบุรี สำเร็จการศึกษาระดับปริญญาตรี วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมโทรคมนาคม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เมื่อปี พ.ศ. 2548

บทความที่ได้รับการตีพิมพ์

- P. Leekul, S. Chiwapreecha, K. Sripimanwat, and P. Supnithi , "Block-Type Encoder Design of Modified Array-Based Low-Density Parity-Check Codes," Proceedings of ECTI-CON 2007, Chiang Rai, Thailand, pp. 951-954, May 2007.