การประเมินการถอดรหัส LDPC โดยใช้อุปกรณ์ฮาร์ดแวร์

Evaluation of Hardware-based LDPC decoder

หัวหน้าโครงการวิจัย รศ. ดร. สมศักดิ์ ชุมช่วย

โครงการวิจัยคณะวิศวกรรมศาสตร์

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2554

คำนำ

ในการออกแบบโครงสร้างตัวถอดรหัส LDPC โดยใช้วิธีการผลรวมต่ำสุด (MS) และวิธีการผลรวมต่ำสุดดัดแปลง (MMS) เป็นตัวถอดรหัส ในส่วนของขั้นตอนการประมวลผล ได้ใช้เทคนิคการสลับลำดับเมตริกซ์พาริตี้เช็ค (reorder parity check matrix) จึงส่งผลให้มีความสามารถในการประมวลผลซ้อนทับกัน (overlap)เพื่อเพิ่มความเร็วในการถอดรหัสอย่างไรก็ตามสำหรับการบ่งบอกถึงประสิทธิภาพของการถอดรหัสของรายงานฉบับนี้ จะบ่งบอกถึงการใช้พื้นที่ของฮาร์ดแวร์ว่ามีขนาดเล็กเพียงใดเป็นหลัก และเทคนิคในการทำงานซ้อนทับกันของหน่วยประมวลผลจะมีความเหมาะสมเมื่อเรานำไปใช้ในเมตริกซ์พาริตี้เช็คบนมาตราฐานIEEE 802.11n ซึ่งมีรูปแบบเมตริกซ์มาตราฐาน ที่สามารถทำตามเงื่อนไขพื้นฐานของวิธีการที่เราได้นำไปใช้

สารบัญ

หน้า

คำนำ I

สารบัญ II

สารบัญตาราง IV

สารบัญรูป II

บทที่ 1 บทนำ 1

1.1 ความเป็นมาและความสำคัญของปัญหา 1

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา 2

1.3 สมมติฐานของการศึกษา 3

1.4 ขอบเขตของการศึกษา 3

1.5 ขั้นตอนของการศึกษา 3

1.6 โครงสร้างรายงาน 4

บทที่ 2 ทฤษฎีพื้นฐานและงานวิจัยที่เกี่ยวข้อง 6

2.1 ระบบการสื่อสารพื้นฐาน 6

2.2 รหัสบล็อกเชิงเส้น (Linear Block Codes) 7

2.3 รหัสพาริตี้เช็คความหนาแน่นต่ำ 14

2.4 งานวิจัยที่เกี่ยวข้อง 28

บทที่ 3 การออกแบบการถอดรหัสแอลดีพีซี 31

3.1 การสลับลำดับเมตริกซ์พาริตี้เช็ค (Reordering Parity Check Matrix) 31

3.2 การทำงานซ้อนกันของ VNU และ CNU(Overlapped Opreations of VNUs and CNUs) 34

บทที่ 4 การออกแบบโครงสร้างภายในเอฟพีจีเอ 36

4.1 หน่วยประมวลผลเช็คโหนด (CNUs Unit) 37

4.2 หน่วยประมวลผลบิตโหนด (VNUs Unit) 38

4.3 หน่วยอินพุตบัฟเฟอร์ (Input Buffer Unit) 40

4.4 หน่วยเอาต์พุตบัฟเฟอร์ (Output Buffer Unit) 40

4.5 หน่วยความจำ (Memmory Unit) 41

4.6 หน่วยควบคุม (Control Unit) 41

บทที่ 5 การทดลองระบบถอดรหัสแอลดีพีซี 43

5.1 ผลการทดลองของขั้นตอนการถอดรหัสแอลดีพีซี 43

5.2 ผลการทดลองจากการสังเคราะห์ของเอฟพีจีเอ 48
บทที่ 6 สรุปผลการวิจัย และข้อเสนอแนะ 49
6.1 สรุปผลการวิจัย 49
6.2 ปัญหาที่เกิดขึ้นและข้อเสนอแนะ 49
เอกสารอ้างอิง 51
ภาคผนวก 52
ดัชนี 53

สารบัญตาราง

ตารางที่ หน้า
ตารางที่ 3.1เปรียบเทียบการใช้เวลา (cycle)ของการทำงานแต่หละรูปแบบ..... 35
ตารางที่ 5.1เปรียบเทียบการใช้พื้นที่และความถี่ของฮาร์ดแวร์ 48

สารบัญรูป

รูปที่ หน้า
รูปที่ 2.1 แผนผังการเข้ารหัสข้อมูลของระบบสื่อสาร 6
รูปที่ 2.2 โครงสร้างของรหัสบล็อกเชิงเส้น 7
รูปที่ 2.3 กราฟแทนเนอร์ 20
รูปที่ 2.4 แสดงการส่งผ่านข้อมูลระหว่าง บิตโหนด และ เช็คโหนด 22
รูปที่ 2.5 แสดงค่า $L(qij)$ที่ส่งจาก บิตโหนด i ไปยัง เช็คโหนด j 23
รูปที่ 2.6แสดงแผนภาพการคำนวณค่า$L(r_{44})$ 24
รูปที่ 2.7แสดงแผนภาพการปรับปรุงข่าวสารของ$L(q_{45})$ 24
รูปที่ 2.8แสดงแผนภาพการหาค่าซอฟต์เอาต์พุตของการถอดรหัส 25
รูปที่ 2.9กราฟการคำนวณหาค่า $\phi(x)$ 26
รูปที่ 2.10เมตริกซ์พาริตี้เช็คอัตรารหัส 1/2 ตามมาตราฐาน WiMAXIEEE 802.16e [x]. 29
รูปที่ 2.11บล็อกไดอะแกรมหน่วยประมวลผล VNU ด้วยวิธีการหาค่าต่ำสุด 30
รูปที่ 2.12บล็อกไดอะแกรมหน่วยประมวลผล VNU ด้วยวิธีการหาค่าต่ำสุดดัดแปลง 30
รูปที่ 3.1 เมตริกซ์พาริตี้เช็คขนาด 648 bits อัตรารหัส 1/2 ตามมาตราฐาน IEEE 802.11n 32
รูปที่ 3.2 การสลับลำดับเมตริกซ์พาริตี้เช็ค 32
รูปที่ 3.3 เมตริกซ์พาริตี้เช็คH ที่ถูกสลับลำดับแล้ว 33
รูปที่ 3.4 เมตริกซ์พาริตี้เช็คH ที่ถูกสลับลำดับแล้วในรูปแบบที่เราสำเสนอ 33
รูปที่ 3.5รูปแบบการทำงานของหน่วยประมวลแต่ละแบบ 34
รูปที่ 4.1โครงสร้างระบบการถอดรหัสแอลดีพีซี 36
รูปที่ 4.2โครงสร้างของหน่วยประมวลผล CNU 37
รูปที่ 4.3 เส้นทางในการเปรียบเทียบเพื่อหาค่าต่ำสุด 37
รูปที่ 4.4โครงสร้างเพื่อการคำนวณหาบิตเครื่องหมาย (sign bit) 38
รูปที่ 4.5โครงสร้างของหน่วยประมวลผล VNUs min-sum 38
รูปที่ 4.6โครงสร้างของหน่วยประมวลผล VNUs modified min-sum 39
รูปที่ 4.7โครงสร้างของหน่วยประมวลผล VNUs modified min-sum ที่ถูกปรับปรุงแล้ว 39
รูปที่ 4.8โครงสร้างการสลับตำแหน่งคำรหัสของอินพุทบัฟเฟอร์ 40
รูปที่ 4.9โครงสร้างการสลับตำแหน่งคำรหัสของเอาต์พุตบัฟเฟอร์ 40
รูปที่ 4.10 เมตริกซ์พาริตี้เช็คที่ใช้ในการถอดรหัส 41
รูปที่ 4.11 แสดงรูปแบบของหน่วยความจำ 41

สารบัญรูป(ต่อ)

รูปที่ 5.1 ผลจำลองการทำงานของหน่วยอินพุตบัฟเฟอร์ก่อนสลับตำแหน่ง 43
รูปที่ 5.2 ผลจำลองการทำงานของหน่วยอินพุตบัฟเฟอร์หลังสลับตำแหน่ง 43
รูปที่ 5.3 ผลจำลองการทำงานของหน่วยประมวลผล CNU ค่า magnitude bit และ Sign bit 44
รูปที่ 5.4 ผลจำลองการทำงานของหน่วยประมวลผล CNU ค่าเอาต์พุตต่ำสุด 45
รูปที่ 5.5 ผลจำลองการทำงานของหน่วยประมวลผล VNU ด้วยวิธีการหาค่าต่ำสุด (MS) 45
รูปที่ 5.6 ผลจำลองการทำงานของหน่วยประมวลผล VNU ด้วยวิธีการหาค่าต่ำสุดดัดแปลง (MMS) 46
รูปที่ 5.7 ผลจำลองการทำงานของขั้นตอนการตัดสินใจ 47
รูปที่ 5.8 ผลจำลองการทำงานของหน่วยเอาต์พุตบัฟเฟอร์ 47

บทที่ 1

บทนำ

ในบทนี้จะกล่าวถึงความเป็นมาและความสำคัญของปัญหา ความมุ่งหมายและวัตถุประสงค์ของการศึกษา สมมติฐานของการศึกษา ทฤษฎีหรือแนวความคิดที่ใช้ในงานวิจัย ขอบเขตของงานวิจัยและโครงสร้างของรายงาน

1.1 ความเป็นมาและความสำคัญของปัญหา

การพัฒนาระบบสื่อสารเป็นสิ่งที่จำเป็นในปัจจุบันทั้งนี้ เพราะการสื่อสารเข้ามามีบทบาทสำคัญในการดำรงชีวิตประจำวันมากขึ้น เนื่องจากการเทคโนโลยีการสื่อสารทำให้เกิดความสะดวกสบายมากขึ้น เช่น การติดต่อประสานงานทำได้รวดเร็วขึ้น และช่วยลดค่าใช้จ่ายสำหรับการเดินทาง ดังนั้นจะ เห็นได้ว่า การพัฒนาเทคโนโลยีการติดต่อสื่อสารจึงเป็นสิ่งสำคัญโดยเฉพาะอย่างยิ่ง เทคโนโลยีการสื่อสารในเชิงดิจิตอล

ในระบบสื่อสารเชิงดิจิตอลซึ่งเกี่ยวข้องการส่งข้อมูลเชิงดิจิตอลจากจุดหนึ่งไปสู่อีกจุดหนึ่ง ผ่านช่องสื่อสารแบบต่างๆ ปัญหาคุณภาพของสัญญาณที่รับได้ที่ภาครับ โดยมีสาเหตุหลายประการ เช่น สัญญาณรบกวนที่ เกิดขึ้นภายในช่องสื่อสาร และสัญญาณรบกวนจากภายนอกหรือคุณลักษณะที่ไม่อุดมคติของช่องสื่อสาร จากประเด็นปัญหาดังกล่าวเหล่านี้ Shannon (1948) ได้ค้นพบทฤษฎีการส่งข้อมูลผ่านช่องสื่อสาร โดยแสดงให้เห็นว่า ข้อมูลดิจิตอลสามารถถูกส่งผ่านช่องสื่อสารได้อย่างมีประสิทธิภาพและมีความผิดพลาดน้อยมากถ้าอัตราการส่งข้อมูล (Data rate) ไม่เกินความจุของช่องสื่อสาร (Channel capacity) และมีการเข้ารหัสข้อมูลที่ต้นทาง (Data encoding) จากนั้นก็มีการถอดรหัสข้อมูลที่ปลายทาง (Data decoding) ที่เหมาะสม จากนั้นมานักวิจัยทั่วโลกก็ได้คิดค้นเทคนิคต่างๆ มาอย่างต่อเนื่องเพื่อที่จะให้ได้ประสิทธิภาพของการส่งข้อมูลสูงสุดตามทฤษฎีดังกล่าว นอกจากนั้นในปัจจุบัน ซึ่งเป็นยุคของข้อมูลข่าวสาร ความต้องการของผู้ใช้บริการระบบสื่อสารมีมากขึ้นเรื่อยๆ ไม่ว่าจะเป็นความเร็วของการส่งข้อมูล ประสิทธิภาพของข้อมูลที่รับได้จะต้องมีความถูกต้องแม่นยำสูงเทคนิคหนึ่งที่มีบทบาทมากในระบบสื่อสารเพื่อตอบสนองต่อความต้องการต่างๆเหล่านี้คือ เทคนิคการเข้ารหัสเพื่อแก้ไขความผิดพลาดล่วงหน้า (Forward error correction, FEC)

โดยรหัสควบคุมความผิดพลาด (ECC: Error Control Codes) เป็นแขนงหนึ่งในการประมวลสัญญาณ ที่ใช้อย่างแพร่หลายในการสื่อสารสมัยใหม่ การกระทำเป็นการอนุมานว่าความผิดพลาดจะเกิดขึ้นอย่างแน่นอน และมีการเตรียมพร้อมที่จะแก้ไขความผิดพลาดนั้น ซึ่งเป็นหนึ่งในเทคนิคของวิธีการแก้ไขความผิดพลาดไว้ล่วงหน้า (FEC: Forward Error Correction Code)

รหัสควบคุมความผิดพลาดนั้น แบ่งเป็น 2 แบบ คือ แบบที่ตรวจสอบได้ว่า เกิดความผิดพลาดขึ้น (Error detection) และแบบที่แก้ไขความผิดพลาดได้ด้วย (Error correction)

- Error Detection Code ที่ใช้กัน แพร่หลาย คือ Parity Check Code และ Cyclic Redundant Code (CRC)

-Error Correction Code มีใช้กันแพร่หลาย 2 กลุ่ม คือ กลุ่มรหัสบล็อค (Block Code) เช่น รหัสแฮมมิ่ง รหัส บีซีเอช รหัสรีด-โซโลมอน และรหัส แอลดีพีซี ในกลุ่มรหัสสังวัฏณาการ (Convolutional Code) เช่น รหัสสังวัฏณาการ และรหัสเทอร์โบ

เนื่องจากรหัสแอลดีพีซีมีประสิทธิภาพในการแก้ไขข้อผิดพลาดได้ดีและมีความซับซ้อนน้อยกว่าของกลุ่มในรหัสเดียวกัน ทำให้นักวิจัยต่างสนใจกันอย่างมากโดยการวิจัยจะมุ่งเน้นไปที่การวิเคราะห์และปรับปรุงรหัสให้ดีขึ้น และรหัสแอลดีพีซีได้รับการนำเสนอไปใช้ร่วมกับระบบต่างๆ มากมายในปัจจุบันเช่น นำไปใช้ร่วมกับระบบการสื่อสารไร้สาย ระบบการสื่อสารดาวเทียมและ ระบบฮาร์ดดิสก์ไดร์ฟ นอกจากนี้รหัสแอลดีพีซีได้รับการนำเสนอใช้ในยุคที่สี่ (4G) และที่สำคัญรหัสแอลดีพีซีไม่มี สิทธิบัตรคุ้มครองสามารถนำพัฒนาและผลิตโดยไม่เสียค่าใช้สิทธิแต่อย่างใด

จากประเด็นดังกล่าวทำให้เรานำรหัสแอลดีพีซีมาวิจัยและพัฒนาต่อเพื่อให้มีประสิทธิภาพดีที่ยิ่งขึ้นและลดความซับซ้อนเมื่อนำรหัสแอลดีพีซีไปประยุกต์ใช้ในฮาร์ดแวร์

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

จุดประสงค์ของงานวิจัยสามารถสรุปได้ดังนี้

1.2.1 เพื่อศึกษาการประยุกต์ใช้วิธีการเพิ่มประสิทธิภาพในการสื่อสาร ในช่องสัญญาณ โดยใช้รหัสควบคุมความผิดพลาด (ECC) เพื่อลดผลกระทบจากสัญญาณรบกวนและสามารถแก้ไขข้อมูลที่ผิดพลาดในการรับส่งข้อมูล

1.2.2 เพื่อศึกษา และประยุกต์ใช้รหัสควบคุมความผิดพลาด ด้วยรหัสแอลดีพีซีโดยการเข้ารหัสและการถอดรหัสแอลดีพีซีในระดับซอฟแวร์

1.2.3 เพื่อศึกษา การออกแบบโครงสร้างในการถอดรหัสแอลดีพีซีและนำไปประยุกต์ใช้ ในระดับฮาร์ดแวร์

1.2.4 เพื่อศึกษา และประยุกต์ใช้ การถอดรหัสแอลดีพีซี โดยพัฒนาให้รหัสมีความเร็วในการประมวลผลมากขึ้นและลดขนาดของฮาร์ดแวร์ให้น้อยลง

1.3 สมมติฐานของการศึกษา

จากการส่งผ่านข้อมูลผ่านไปยังอีกจุดหนึ่ง ตามปกติข้อมูลที่ได้รับจะคงรูปเดิมเหมือนฝั่งส่ง แต่เนื่องจากภายในช่องสัญญาณจริงนั้นมีสัญญานรบกวนแทรกอยู่จึงทำให้ข้อมูลที่ได้รับนั้นผิดเพี้ยนไปจากเดิม โดยอัตราการผิดเพี้ยนของข้อมูลขึ้นอยู่กับระดับความแรงของสัญญาณในการส่งและระดับความแรงของสัญญาณรบกวน ด้วยเหตุนี้จึงมีการคิดค้นวิธีการป้องกันสัญญาณรบกวนโดยใช้วิธีการเข้ารหัส ซึ่งรหัสแอลดีพีซี (LDPC) มีประสิทธิภาพสูงและมีความซับซ้อนน้อยเมื่อนำไปพัฒนาบนฮาร์ดแวร์

ในส่วนอัลกอริทึมของการถอดรหัสแอลดีพีซีนั้น ได้ถูกพัฒนามาหลายวิธีการจนกระทั่งในปัจจุบันวิธีการที่นิยมนำไปใช้บนฮาร์ดแวร์มากที่สุดคือวิธีการหาค่าต่ำสุด (Min-Sum Algorithm) และวิธีการหาค่าต่ำสุดดัดแปลง (Modified Min-Sum Algorithm) ด้วยเหตุผลที่กล่าวข้างต้นจึงนำเอาวิธีการดังกล่าวมาพัฒนา เพื่อนำไปสร้างเป็นต้นแบบของการถอดรหัสด้วยฮาร์ดแวร์ซึ่งใช้ในการรับส่งข้อมูลผ่านช่องสัญญาณ

1.4 ขอบเขตของการศึกษา

รายงานฉบับนี้เป็นการศึกษาเกี่ยวกับวิธีการถอดรหัสแอลดีพีซีด้วยวิธีการหาค่าต่ำสุดและวิธีการหาค่าต่ำสุดดัดแปลงโดยระบบการทำงานนั้นจะนำคำรหัสที่ได้จากเข้ารหัสมาผ่านช่องสัญญานที่มีสัญญาณรบกวนและผ่านขบวนการถอดรหัสด้วยวิธีการที่นำเสนอเพื่อให้ได้คำรหัสถูกต้อง

การทดลองในระดับซอฟแวร์จะอาศัยโปรแกรม Matlabในส่วนของการประมวลผลของการถอดรหัสแอลดีพีซี โดยใช้อัลกอลิทึมการหาค่าต่ำสุด (Min-Sum algorithm) และอัลกอลิทึมการหาค่าต่ำสุดดัดแปลง (Modified Min-Sum algorithm)การออกแบบโครงสร้างการถอดรหัสแอลดีพีซีในระดับฮาร์ดแวร์จะอาศัยโปรแกรม ISE ของ Xilinx ในการออกแบบระบบการถอดรหัสของทั้งสองอัลกอลิทึม เพื่อนำไปโปรแกรมลงบอร์ด FPGA และทดสอบประสิทธิภาพในการทำงานจริง

1.5 ขั้นตอนของการศึกษา

ในการวิจัยเพื่อที่จะนำเสนอเทคนิคการออกแบบโครงสร้างของการถอดรหัสแอลดีพีซีด้วยวิธีการหาค่าต่ำสุดและวิธีการหาค่าต่ำสุดดัดแปลง ได้กำหนดขั้นตอนต่างๆ ของการวิจัยไว้ตามลำดับดังนี้

1.5.1 ศึกษาและรวบรวมข้อมูลของรหัสควบคุมความผิดพลาดเพื่อกำหนดความเป็นไปได้ของงานวิจัย

1.5.2 ศึกษาและรวบรวมข้อมูลการทำงานของ FPGA และการเขียนโปรแกรมด้วยภาษา Verilog

1.5.3 จำลองการทำงานของระบบรหัสควบคุมความผิดพลาด ด้วยรหัสแอลดีพีซีบนโปรแกรม MATLAB เพื่อตรวจสอบความถูกต้องของระบบการทำงาน และขั้นตอนในการถอดรหัสของ

อัลกอริทึม

1.5.4 ออกแบบโครงสร้างการถอดรหัสแอลดีพีซีตามขั้นตอนอัลกอริทึมด้วยภาษา Verilog เพื่อนำไปโปรแกรมลงบนชิพFPGA ของ Xilinx

1.5.5 ทดสอบการทำงานของการถอดรหัสและปรับปรุงแก้ไขเพื่อให้ระบบทำงานตามขั้นตอนอย่างถูกต้อง

1.5.6 วิเคราะห์ปัญหาและวิจารณ์ผลการทดลอง พร้อมด้วยการสรุปการวิจัย

1.6 โครงสร้างรายงาน

รายงานฉบับนี้ได้นำเสนอวิธีการถอดรหัสแอลดีพีซีด้วยวิธีการหาค่าต่ำสุดและวิธีการหาค่าต่ำสุดดัดแปลง โดยให้รายละเอียดของทฤษฎี การออกแบบ และการทดสอบ ซึ่งรายงานฉบับนี้ได้แบ่งเนื้อหาออกเป็น6 บท

บทที่ 1บทนำ

กล่าวถึงความเป็นมาและความสำคัญของปัญหา ความมุ่งหมายและวัตถุประสงค์ของการศึกษา สมมติฐานของการศึกษา ทฤษฎีหรือแนวความคิดที่ใช้ในงานวิจัย ขอบเขตของงานวิจัย ขั้นตอนการศึกษา และโครงสร้างของรายงาน

บทที่ 2ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

อธิบายถึงระบบสื่อสารพื้นฐาน การเข้ารหัสช่องสัญญาณ โดยรหัสแอลดีพีซีจะเป็นรหัสแบบบล็อกเชิงเส้น คุณสมบัติของบล็อกเชิงเส้น รูปแบบและระบบการทำงานของรหัสแอลดีพีซี และงานวิจัยที่เกี่ยวข้อง

บทที่ 3การออกแบบตัวถอดรหัสแอลดีพีซี

กล่าวถึงขั้นตอนการออกแบบตัวถอดรหัสแอลดีพีซี เริ่มจากการออกแบบเมตริกซ์พาริตี้เช็ค เพื่อนำไปใช้สำหรับการออกแบบวิธีการประมวลผลบนฮาร์ดแวร์

บทที่ 4 การออกแบบโครงสร้างภายในแอฟพีจีเอ

กล่าวถึงวิธีการออกแบบของระบบการถอดรหัสแอลดีพีซีในฮาร์ดแวร์ โดยการออกแบบหน่วยประมวลผลหลัก จะแบ่งเป็นสองส่วนด้วยกันคือ หน่วยประมวลผลของแต่ละแถว CNUและหน่วย

ประมวลผลของแต่หละหลัก VNUนอกจากนั้นได้ทำการออกแบบโครงสร้างของส่วนประกอบที่สำคัญคือ หน่วยอินพุตบัฟเฟอร์ เอาต์พุตบัพเฟอร์ และหน่วยควบคุมระบบ

บทที่ 5 การทดลองระบบถอดรหัสแอลดีพีซี

ในบทนี้จะแสดงการทดลองในการถอดรหัสแอลดีพีซี ทั้งด้วยการจำลองการทำงานเพื่อทดสอบประสิทธิภาพและขั้นตอนทั้งวิธีการผลรวมต่ำสุด (MS) และวิธีการผลรวมต่ำสุดดัดแปลง(MMS)และทดสอบการสังเคราะห์โครงสร้างที่ได้ออกแบบไว้เพื่อจำลองการทำงานบน FPGA ด้วยโปรแกรมของ Xilinx

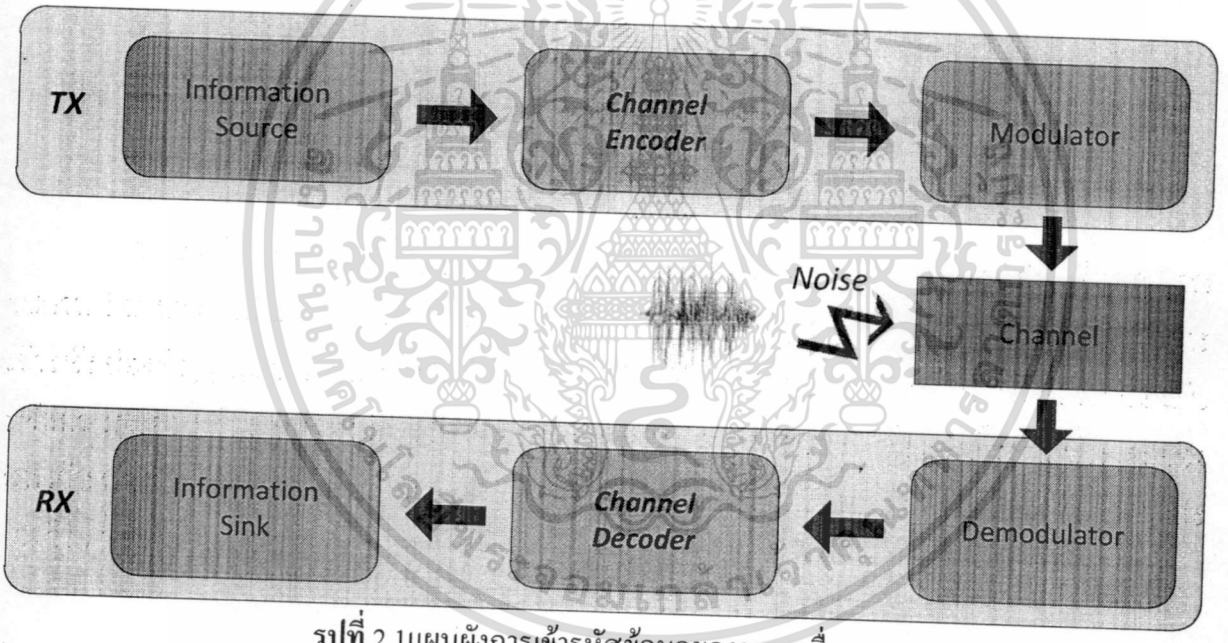
บทที่ 6 สรุปผลการวิจัยและข้อเสนอแนะ

ในส่วนนี้กล่าวถึงสรุปผลที่ได้จากการทดลอง ข้อจำกัดของโครงสร้างที่ได้ออกแบบไว้รวมทั้งประสิทธิภาพในการถอดรหัส พร้อมทั้งเสนอแนวทางแก้ไขเพื่อพัฒนางานวิจัยต่อไป

บทที่ 2
ทฤษฎีพื้นฐานและงานวิจัยที่เกี่ยวข้อง

การส่งข้อมูลผ่านช่องสัญญาณใดสัญญาณหนึ่ง ปัญหาสำคัญที่หลีกเลี่ยงไม่ได้นั้นคือสัญญาณรบกวน ซึ่งผลกระทบที่เกิดขึ้นอาจทำให้ข้อมูลเสียหายได้ วิธีการแก้ปัญหาอาจมีได้หลายรูปแบบ โดยวิธีการเข้ารหัสช่องสัญญาณเป็นวิธีการหนึ่งเพื่อแก้ปัญหาดังกล่าว ในบทนี้จะกล่าวถึงทฤษฎีที่เกี่ยวข้องกับรหัสควบคุมความผิดพลาดแอลดีพีซีโดยเนื้อหาจะประกอบไปด้วยระบบการสื่อสารพื้นฐาน ประเภทของการเข้ารหัสช่องสัญญาณรูปแบบการเข้ารหัสและขั้นตอนการถอดรหัสแอลดีพีซีด้วยอัลกอริทึมต่างๆและงานวิจัยที่เกี่ยวข้อง

2.1 ระบบการสื่อสารพื้นฐาน



รูปที่ 2.1แผนผังการเข้ารหัสข้อมูลของระบบสื่อสาร

ระบบการสื่อสารดิจิตอลดังแสดงในรูปที่ 2.1 เมื่อมีข้อมูลจากแหล่งกำเนิดข้อมูลที่ต้องการส่งไปยังจุดรับข้อมูล จะทำการเข้ารหัสด้วยวงเข้ารหัส (Encoder) เพื่อลดอัตราบิตผิดพลาด (Bit Error Rate) ที่อาจเกิดขึ้นเนื่องจากสัญญาณรบกวน จากนั้นข้อมูลที่ทำการเข้ารหัสแล้วจะผ่านไปยังวงจรมอดูเลเตอร์ (Modulator) ซึ่งเป็นวงจรที่ช่วยในการย้ายย่านความถี่ของสัญญาณข้อมูลโดยอาศัยคลื่นพาห์เป็นตัวช่วยเพื่อให้เกิดความเหมาะสมในการส่งสัญญาณผ่านตัวกลางหรือช่องสัญญาณไป เมื่อส่งผ่านตัวกลาง แล้วในด้านของเครื่องรับจะมีวงจรที่มีหน้าที่ตรงกันข้ามกับเครื่องส่ง เพื่อทำการแปลงข้อมูลที่รับมาได้ให้เป็นข้อมูลเดียวกับข้อมูลจากแหล่งกำเนิดข้อมูล โดยต้องผ่านวงจรดีมอดูเลเตอร์ซึ่งจะทำหน้าที่ในการแยกเอาสัญญาณ

ข้อมูลออกจากคลื่นพาห์ที่รับมาได้ทางเครื่องรับ และทำการถอดรหัสให้ได้ข้อมูลเดิมโดยใช้วงจรถอดรหัส ในส่วนของวงจรนี้ยังมีการตรวจสอบความผิดพลาดของข้อมูล และยังสามารถแก้ไขบิตที่ผิดพลาดของข้อมูลได้ด้วย

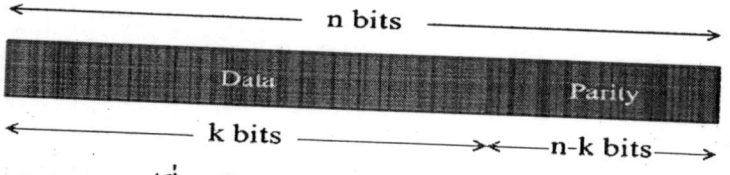
การสื่อสารในระบบต่างๆ สามารถเกิดความผิดพลาดได้ และวิธีลดความผิดพลาดที่เกิดนั้นสามารถทำได้หลายวิธี เช่น วิธีการเพิ่มกำลังส่งของเครื่องส่งในการส่งข้อมูลและวิธีการเข้ารหัส ซึ่งวิธีการเข้ารหัสเพื่อแก้ไขความผิดพลาดของข้อมูล (Error Control Coding) เป็นเทคนิคที่ได้นำไปใช้ สำหรับการรับส่งข้อมูลในระบบการสื่อสารดิจิตอล และความผิดพลาดของข้อมูลในระบบนั้นเกิดขึ้นจากสาเหตุหลายประการและไม่สามารถจะคาดคะเนได้ว่าจะเกิดขึ้นเวลาใด ซึ่งสาเหตุสำคัญประการหนึ่งที่ทำให้ข้อมูลผิดพลาดคือ สัญญาณข้อมูลถูกรบกวนจากสัญญาณรบกวน (Noise) ดังนั้นจึงทำให้การ เข้ารหัสข้อมูลจำเป็นสำหรับการสื่อสารดิจิตอล

การเข้ารหัสช่องสัญญาณสามารถแบ่งออกเป็นสองประเภทได้แก่

1) รหัสคอนโวลูชัน (Convolution code)
2) รหัสบล็อกเชิงเส้น (Linear block code)

รหัสสองชนิดนี้มีความแตกต่างกันที่ขั้นตอนในการเข้ารหัส กล่าวคือ รหัสคอนโวลูชันเป็นรหัสช่องเส้นเชิงเส้นชนิดหนึ่งซึ่งจะทำการเข้ารหัสข้อมูลที่ละบิต และเอาต์พุตของตัวเข้ารหัสที่ได้แต่ละครั้งจะขึ้นกับข่าวสารของอินพุตปัจจุบันและข่าวสารที่ผ่านมาด้วย รหัสคอนโวลูชันที่มีชื่อเสียงและเป็นที่รู้จักกันอย่างแพร่หลายคือ รหัสเทอร์โบ (Turbo codes) ซึ่งมีสมรรถนะการทำงานที่เข้าใกล้ลิมิตของแชนนอน (Shannon) ในขณะที่รหัสบล็อกเชิงเส้นจะทำการเข้ารหัสข้อมูลทีละบล็อก โดยที่เอาต์พุตของตัวเข้ารหัสที่ได้แต่ละครั้งจะไม่ขึ้นกับข่าวสารของอินพุตปัจจุบันและข่าวสารที่ผ่านมาด้วย โดยการเข้ารหัสจะใช้เมตริกส์กำเนิด (Generator matrix) หรือบางครั้งสามารถใช้เมตริกส์อีกประเภทหนึ่งที่เรียกว่าเมตริกส์พาริตี้เช็ค (Parity check matrix) มีงานวิจัยพบว่ารหัสบล็อกเชิงเส้นชนิดหนึ่งที่เรียกว่า รหัสแอลดีพีซี มีสมรรถนะการทำงานที่เข้าใกล้ลิมิตของแชนนอลได้เช่นเดียวกับ รหัสเทอร์โบ

2.2 รหัสบล็อกเชิงเส้น(Linear Block Codes)



รูปที่ 2.2โครงสร้างของรหัสบล็อกเชิงเส้น

ระบบรหัสบล็อกเชิงเส้นนั้นข้อมูลที่ใช้สำหรับทำการเข้ารหัสถูกแบ่งออกเป็นบล็อกข้อมูลที่มีขนาดเท่ากันจำนวนkบิตซึ่งแทนด้วย$m_0,m_1,...m_{k-1}$ซึ่งในการเข้ารหัสนั้นจะนำบล็อกข้อมูลทั้งkบิตไปใช้ในการสร้างบิตพาริตี้จำนวน n-k บิตซึ่งเขียนแทนด้วย$b_0,b_1,...,b_{n-k-1}$และเมื่อนำบิตข้อมูลและบิตพาริตี้มาประกอบกันจะได้เป็นคำรหัส$c_0,c_1,...,c_n$ซึ่งถ้าแสดงในรูปของสมการความสัมพันธ์จะได้สมการดังนี้คือ

$$c_i = \begin{cases} b_i, & i = 0,1,...,n-k-1 \\ m_{i+k+n}, & i = n-k, n-k+1,...,n-1 \end{cases} \tag{1.1}$$

จะเห็นว่ากระบวนการเข้ารหัสบล็อกเชิงเส้นจึงเหมือนการแปลงบิตข้อมูลจำนวน k บิต ให้ได้เป็นคำรหัสที่มีขนาดเพิ่มขึ้นเป็น n บิต นั่นเอง ซึ่งหากพิจารณาในเบื้องต้นดูเหมือนว่าเป็นกระบวนการที่ไม่ซับซ้อนยุ่งยาก แต่ถ้าพิจารณาให้ดีจะพบว่า เนื่องจากการเข้ารหัสจะต้องมีการพิจารณาบิตข้อมูลครั้งละจำนวน k บิต ซึ่งมีรูปแบบที่เป็นไปได้ทั้งหมดมากถึง 2k รูปแบบ เมื่อต้องบรรจุรูปแบบทั้งหมดไว้ในหน่วยความจำเพื่อแปลงให้ได้เป็นคำรหัสที่เหมาะสม ที่มีขนาดความยาว n บิต จะต้องใช้วงจรที่ซับซ้อนและหน่วยความจำที่มีขนาดใหญ่มาก โดยเฉพาะอย่างยิ่งถ้า k มีขนาดที่ใหญ่ขึ้น ความซับซ้อนของวงจรสร้างรหัสนี้เอง ที่เป็นปัญหาหลักในการพัฒนาและเป็นเหตุผลสำคัญที่ทำให้การพัฒนาแบบบล็อกแทบทั้งหมดจึงมุ่งเน้นไปในกลุ่มรหัสบล็อกที่มีคุณสมบัติพิเศษที่เรียกว่า คุณสมบัติเชิงเส้น (linear property) เป็นหลัก

จากที่กล่าวมานั้น จะเห็นได้ว่าในการเข้ารหัสอยู่ที่การคำนวณค่าบิตพาริตี้นั่นเอง ในกรณีของรหัสบล็อกเชิงเส้น ค่าของพาริตี้จะคำนวณจากบิตข้อมูลในรูปของการบวกเชิงเส้นในรูปแบบ ดังนี้

$$b_i = p_{i0}m_0 + p_{i1}m_1 + p_{i2}m_2 + ... + p_{i,k-1}m_{k-1} \qquad i = 0,1,2,...,n-k-1 \tag{1.2}$$

โดยสัมประสิทธิ์p_{ij}จะมีค่าได้ 2 แบบเท่านั้นคือ 0 หรือ 1 ทั้งนี้ ค่าของ p_{ij}จะกำหนดให้สอดคล้องกับความต้องการที่จะให้บิตพาริตี้b_iมีความเกี่ยวพันกับบิตข้อมูลที่ m_jหรือไม่ ถ้าไม่ต้องการให้มีความสัมพันธ์หรือขึ้นแก่กันก็กำหนด $p_{ij} = 0$ เพราะฉะนั้นจุดสำคัญของการเข้ารหัสจึงอยู่ที่การกำหนด p_{ij}ที่เหมาะสมเพื่อให้ได้คุณสมบัติตามที่ต้องการ

โดยทั่วไปนั้น ในการศึกษาโครงสร้างวิธีการเข้าและถอดรหัสบล็อกเชิงเส้น จะแสดงค่าต่างๆ ที่กล่าวมานั้นในรูปของเมตริกซ์ เพื่อให้เกิดความกระชับ ได้ดังนี้

$$\mathrm{m} = [m_0, m_1, ..., m_{k-1}] \tag{1.3}$$

$$\mathrm{b} = [b_0, b_1, ..., b_{n-k-1}] \tag{1.4}$$

$$\mathrm{c} = [c_0, c_1, ..., c_{n-1}] \tag{1.5}$$

$$\mathrm{b} = \mathrm{mP} \tag{1.6}$$

โดย

$$P=\begin{bmatrix} p_{00} & p_{10} & \cdots & p_{n-k-1,0} \\ p_{01} & p_{11} & \cdots & p_{n-k-1,1} \\ \vdots & \vdots & & \vdots \\ p_{0,k-1} & p_{1,k-1} & \cdots & p_{n-k-1,k-1} \end{bmatrix} \tag{1.7}$$

สาหรบเมตริกซ์ c สามารถแสดงในรูปของ b และ m ได้ดังนี้

$$c=[b\,|\,m] \tag{1.8}$$

อาศัยความสัมพันธ์ตามสมการ จะได้ว่า

$$c=[mP\,|\,m]=m[P\,|\,I_k] \tag{1.9}$$

โดยI_k คือ เมตริกซ์เอกลักษณ์ขนาดkxk

$$I_k=\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{1.10}$$

ถ้ากำหนดให้

$$G=[P\,|\,I_k] \tag{1.11}$$

ซึ่งเป็นเมตริกซ์ขนาดn-kความสัมพันธ์ในสมการข้างบนสามารถทำให้กระชับขึ้นได้เป็น

$$c=mG \tag{1.12}$$

จากสมการจะเห็นได้ว่าคำรหัส**c**สามารถคำนวณได้จากการคูณชุดข้อมูล**m**โดยตรงกับเมตริกซ**G** ดังนั้นเราจึงเรียกเมตริกซ์**G** ว่า เมตริกซ์กำเนิด(generator matrix)

รหัสบล็อกเชิงเส้นมีคุณสมบัติที่น่าสนใจคือคุณสมบัติปิด(closure)นั้นคือถ้านำคำรหัสสองคำมา

รวมกันจะได้เป็นคำรหัสใหม่ที่เป็นสมาชิกของรหัสนั้นๆด้วยสามารถพิสูจน์คุณสมบัติข้อนี้ได้โดยงายสมมุติให้คำรหัสc_1 และc_2 ได้จากการเข้ารหัสข้อมูลm_1 และm_2 ตามลำดับและจะได้เป็น

$$\begin{aligned} c_1 + c_2 &= m_1G + m_2G \\ &= (m_1 + m_2)G \end{aligned} \tag{1.13}$$

เนื่องจาก$m_1 + m_2$จะได้เป็นข้อมูลชุดใหม่ซึ่งเมื่อนำไปคูณกับเมตริกซ์ตัวกำเนิดGก็จะได้คำรหัสที่เป็นสมาชิกหนึ่งของรหัสด้วย

สำหรับส่วนต่อไปนี้จะอธิบายถึงการสร้างความสัมพันธ์ที่เป็นประโยชน์กับกระบวนการถอดรหัส นิยามเมตริกส์H ที่มีขนาด (n-k) x n ขึ้นได้ดังนี้

$$H = \left[I_{n-k} \mid P^T \right] \tag{1.14}$$

โดยI_{n-k} คอเมตริกซ์เอกลักษณ์ขนาดn-kและPคือเมตริกซ์ทรานส์โพสของP(transposeofmatrixP) ซึ่งมีขนาดเท่ากับnถ้านำเมตริกซ์ทรานส์โพสของ**G** มาคูณทั้งสองด้านจะได้

$$\begin{aligned} HG^T &= \left[I_{n-k} \mid P^T \right] \begin{bmatrix} P^T \\ \hline I_k \end{bmatrix} \\ &= P^T + P^T \end{aligned} \tag{1.15}$$

จากคุณสมบัติของการบวกกันแบบมอดูโล2จะเห็นได้ว่า$P^T + P^T = 0$ โดยเมตริกซ์ที่คำนวณได้มีขนาดเท่ากับ(n-k)xkและมีสมาชิกเป็นศูนย์ทั้งหมดนั่นคือ

$$HG^T = 0 \tag{1.16}$$

หรือหากพิจารณาในอีกลักษณะหนึ่งจะได้ว่าGH^T =0ด้วยเช่นกัน สามารถใช้ประโยชน์จากความสัมพันธ์นี้ได้โดยนำทรานส์โพสของเมตริกส์H ไปคูณกับสมการที่ (2.12) ข้างต้นทั้งสองด้านนะได้ผลดังนี้

$$\begin{aligned} cH^T &= mGH^T \\ &= 0 \end{aligned} \tag{1.17}$$

สมการนี้มีประโยชน์กับกระบวนการถอดรหัสซึ่งจะอธิบายถึงวิธีการนำไปใช้งานในส่วนต่อไป

สำหรับเมตริกซ์**H** มีชื่อเรียกเฉพาะว่า เมตริกซ์พาริตี้เช็ค(parity-check matrix)

2.2.1 ค่าซินโดรม

เนื่องจากในการส่งคำรหัสที่ได้จากการเข้ารหัสบิตข้อมูลแต่ละชุดผ่านช่องสัญญาณสื่อสารจะได้รับผลกระทบจากปัญหาของสัญญาณรบกวนในแบบต่างๆ ซึ่งทำให้บิตบางบิตในคำรหัสที่รับได้ ณภาครับผิดเพี้ยนไปจากเดิมกำหนดให้สัญญาณที่รับได้ในรูปของเวกเตอร์**r** สามารถแสดงได้ในรูปของสัญญาณที่ส่งออกต้นทางในรูปของเวกเตอร์ c บวกกับสัญญาณรบกวนในรูปของเวกเตอร์ e ดังนี้

$$r = c + e \tag{1.18}$$

ค่าต่างๆในเวกเตอร์ **e**เป็นตัวกำหนดรูปแบบการผิดพลาดของบิตที่เกิดขึ้นเนื่องจากผลกระทบของสัญญาณรบกวนโดยจะมีค่าเป็น "0 "เมื่อบิตที่ตำแหน่งนั้นของสัญญาณ**c** ไม่มีความผิดพลาดเกิดขึ้นและมีค่าเป็น"1" เมื่อบิตของ**r** ที่ตำแหน่งดังกล่าวแตกต่างไปจาก **c**เนื่องจากได้รับผลกระทบของสัญญาณรบกวนนั่นเอง กล่าวคือ

$$e = \begin{cases} 0, & no\ error \\ 1, & error \end{cases} \tag{1.19}$$

นำสัญญาณที่รับได้ในรูปของเวกเตอร์**r** มาถอดรหัสเพื่อหาค่าซินโดรมจะได้ผลดังนี้

$$s = rH^T \tag{1.20}$$

อาศัยสมการที่ 18 และ 20 จะได้

$$\begin{aligned} s &= (c+e)H^T \\ &= cH^T + eH^T \\ &= eH^T \end{aligned} \tag{1.21}$$

พิจารณาในสมการจะเห็นได้ว่าค่าซินโดรมที่คำนวณได้ไม่ขึ้นกับรูปแบบของบิตข้อมูล**c**ที่ส่งออกเลยจะขึ้นก็เฉพาะกับรูปแบบของความผิดพลาด**e**เท่านั้น คุณลักษณะเช่นนี้มีความหมายว่าถ้ากำหนด รูปแบบของความผิดพลาดขึ้นมารูปแบบหนึ่งแล้วไม่ว่าจะส่งคำรหัสของชุดบิตข้อมูลรูปแบบใดออกสู่ช่องสัญญาณก็ตามค่าซินโดรมที่ภาครับคำนวณได้จะเป็นค่าเดียวกันเสมอซึ่งหมายความว่า จะมีชุด สัญญาณที่รับได้**r**ที่มี

รูปแบบแตกต่างอยู่ถึง2^kรูปแบบที่จะให้ค่าผลลัพธ์ของการคำนวณซินโดรม เป็นค่าเดียวกัน

ข้อสังเกตที่น่าสนใจอีกประการหนึ่งก็คือ ค่าซินโดรมsเป็นเวกเตอร์ขนาดn-kบิตเพราะฉะนั้นถ้านำผลลัพธ์ของค่าซินโดรมที่ได้นำมาใช้สำหรับการตัดสินใจวามความผิดพลาดใดเกิดขึ้นกับคำรหัสที่ส่งผ่านช่องสัญญาณจะมีรูปแบบของซินโดรมมค่าที่แตกต่างกันหมด2^k ค่านั้นคือสามารถบอกหรือแยกแยะความผิดพลาดออกได้ไม่เกิน 2^{n-k}รูปแบบ ฉะนั้นถ้าต้องการเพิ่มจำนวน บิตพาริตี้เช็คมากขึ้น ให้พิจารณาเงื่อนไข Hamming boundซึ่งมีใจความ ดังนี้ สำหรับรหัสบล็อกเชิงเส้น (n,k) จะสามารถแก้ไขความผิดพลาดได้มากขึ้นถึง t บิต เงื่อนไขความสัมพันธ์ต่อไปนี้ต้องเป็นจริง

$$2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i} \tag{1.22}$$

$$\binom{n}{i} = \frac{n!}{(n-i)!i!} \tag{1.23}$$

เงื่อนไขนี้สามารถพิสูจน์ได้ เนื่องจากค่าซินโดรม s เป็นเวกเตอร์ขนาด n-k บิต ฉะนั้น จึงมีรูปแบบของซินโดรมที่แตกต่างกันได้มากขึ้นถึง 2^{n-k}รูปแบบ เนื่องจากชุดรหัสในการแยกความแตกต่างหรือบ่งชี้รูปแบบความผิดพลาดได้มากถึง 2^{n-k}รูปแบบ เนื่องจากรหัสชุดนี้ที่มีความยาว n บิต สำหรับกรณีที่มีความผิดเกิดขึ้นถึง i บิตจะมีรูปแบบความผิดพลาดที่ต่างกันจำนวน (n/i) รูปแบบ ฉะนั้นเมื่อรวมรูปแบบความผิดพลาดที่เป็นไปได้ทั้งหมด ตั้งแต่ที่ไม่มีความผิดพลาดเลย มีความผิดพลาดหนึ่งบิต สองบิต สามบิตไปเรื่อยๆ จนกระทั่งถึงกรณีที่มีความผิดพลาด t บิต ชุดรหัสที่ใช้จะต้องมีซินโดรมจำนวนมากพอที่จะใช้แยกแยะความแตกต่างรูปแบบความผิดพลาดทั้งหมดได้ สำหรับชุดรหัสไบนารี่ที่มีจำนวนซินโดรมเท่ากับความผิดพลาดทั้งหมดพอดี นั่นคือ ได้ตามเงื่อนไข Hamming bound ด้วยการใช้เครื่องหมายเท่ากับ จัดได้ว่าเป็นรหัสที่สมบูรณ์ (perfect code)

โดยสรุปแล้ว เมื่อระบบได้รับสัญญาณ r รูปแบบหนึ่ง ให้คำนวณหาค่าซินโดรมโดยอาศัยความสัมพันธ์ $s = rH^T$เนื่องจากซินโดรมไม่สามารถระบุรูปแบบหนึ่งต่อหนึ่งได้ว่าเกิดจากความผิดพลาด e รูปแบบใด เพราะมีรูปแบบความผิดพลาด e ที่แตกต่างกันหลายรูปแบบ ซึ่งส่งผลกระทบกับชุดบิตข้อมูล c ที่ต่างกัน และให้ผลลัพธ์ของซินโดรมค่าเดียวกัน ฉะนั้น แนวทางในการตัดสินใจโดยทั่วไปที่จะต้องทำคือ ให้ตัดสินใจเลือกรูปแบบของความผิดพลาดที่มีความน่าจะเป็นในการเกิดมากที่สุด เช่น ถ้าความผิดพลาดที่เกิดขึ้นกับบิตที่ส่งผ่านช่องสัญญาณเป็นแบบแรนดอม ค่าความน่าจะเป็น ที่จะมีบิตผิดพลาดจำนวนน้อยบิตจะมีค่าสูงกว่าความน่าจะเป็นที่มีความผิดพลาดหลายบิต

2.2.2 ระยะห่างแฮมมิงของบล็อคเชิงเส้น

ในส่วนนี้จะเป็นการคำนวนระยะแฮมมิงสำหรับใช้การระบุขีดความสามารถของการตรวจจับหรือแก้ไขความผิดพลาดของรหัสบล็อกเชิงเส้น(n , k)ในลำดับแรกจะอธิบายคำจำกัดความของน้ำหนักแฮมมิง(Hamming weight) ของคำรหัสก่อนน้ำหนักแฮมมิงของคำรหัส $C = (C_0 , C_1 , ..., C_{n-1})$ ใดๆที่มีค่าเท่ากับจำนวนบิตที่มีค่าเป็น 1 ในคำรหัสนั้น เช่น คำรหัส (1101000) ทีค่าน้ำหนักแฮมมิงเท่ากับ 3 เป็นต้น ส่วนคำจำกัดความระยะแฮมมิงได้กล่าวไว้ ว่า ระยะแฮมมิงระหว่างคำรหัสสองชุด คือ จำนวนของบิตที่มีค่าต่างกัน เช่น ระยะแฮมมิงระหว่างคำรหัส (1101000) กับคำรหัส (0110100) ทีค่าเท่ากับ 4 เพราะมีจำนวนบิตที่ต่างกันเพียง 4 ตำแหน่งคือบิตแรก บิตที่สาม บิตที่สี่ และบิตที่ห้า กำหนดให้ v, w และ x เป็นคำรหัสของชุดรหัส (n, k) หนึ่ง ค่าระยะแฮมมิงมีคุณสมบัติที่น่าสนใจดังนี้

$$d(\mathrm{v},\mathrm{w}) + d(\mathrm{w},\mathrm{x}) \geq d(\mathrm{v},\mathrm{x}) \tag{1.24}$$

นอกจากนี้เรายังทราบอีกด้วยว่าค่าระยะแฮมมิงระหว่างคำรหัสคู่หนึ่งสามารถคำนวณได้จากความสัมพันธ์ต่อไปนี้

$$d(\mathrm{v},\mathrm{w}) = w(\mathrm{v},\mathrm{w}) \tag{1.25}$$

การที่ความสัมพันธ์นี้เป็นจริงได้ก็เพราะคุณสมบัติของการบวกกันแบบมอดูโล2นั่นเองกล่าวคือถ้าบิตแต่ละตำแหน่งของvและwมีค่าตรงกันผลบวกที่ได้ก็จะมีค่าเป็นศูนย์และถ้ามีค่าที่ต่างกันผลบวกที่ได้ก็จะเป็นหนึ่ง และเมื่อคำนวณหาค่าน้ำหนักแฮมมิงV + W จึงเท่ากับการหาค่าระยะแฮมมิง เช่น ให้ v = (1101000) และ w = (0110100) คำนวณหาผลบวกของคำรหัสทั้งสองจะได้ v + w = (1011100) และเมื่อหาค่าน้ำหนักแฮมมิงw(v + w) = 4 ซึ่งสอดคล้องตรงกับการหาค่าระยะแฮมมิงจากการคำนวณ d(v , w) โดยวิธีตรง

ในการระบุขีดความสามารถของรหัสบล็อกเชิงเส้นc(n,k)เราจะอาศัยค่าที่เรียกว่าระยะแฮมมิงต่ำสุด d_{min} ในการพิจารณาค่าดังกล่าวนี้มีนิยามดังนี้

$$\begin{aligned} \mathrm{d}_{\min} &= \min\{w(\mathrm{v}+\mathrm{w}) : \mathrm{v},\mathrm{w} \in \mathrm{C},\ \mathrm{v} \neq \mathrm{w}\} \\ &= \min\{w(\mathrm{x}) : \mathrm{v},\mathrm{x} \in \mathrm{C},\ \mathrm{x} \neq 0\} \\ &= \mathrm{w}_{\min} \end{aligned} \tag{1.26}$$

ค่า$\mathrm{W}_{\min}$ที่ได้นี้คือค่าน้ำหนักต่ำสุดของรหัสเชิงเส้นcสามารถนำมาสรุปเป็นทฤษฎีบทที่น่าสนใจ

ซึ่งมีประโยชน์ต่อการคำนวณหาระยะแฮมมิงต่ำสุดของรหัส **c** (n , k)คือแทนที่จะต้องคำนวณหาระยะแฮมมิงระหว่างคำรหัสทุกคู่ในชุดรหัส**c**และเปรียบเทียบค่าระยะแฮมมิงที่คำนวณได้ทั้งหมดเพื่อหาค่าที่ต่ำที่สุด ซึ่งผลลัพธ์ที่ได้คือ ระยะแฮมมิงต่ำสุดของรหัส **c** แต่ถ้าใช้ทฤษฎีบทในการคำนวณสิ่งที่ต้องทำคือคำนวณค่าน้ำหนักของคำรหัสทั้งหมดที่ไม่ใช่ศูนย์ และเปรียบเทียบน้ำหนักแฮมมิงแต่ละค่าที่ได้เพื่อหาค่าที่ต่ำที่สุดโดยผลลัพธ์ที่ได้จะเป็นค่าระยะแฮมมิงต่ำที่สุดของรหัส**c**ด้วย

2.3 รหัสพาริตี้เช็คความหนาแน่นต่ำ

รหัสแก้ไขความผิดพลาด (Error Correcting Code) ทำหน้าที่ลดจำนวนบิตผิดพลาดของข้อมูล ในระบบสื่อสารแบบดิจิตอล โดยยังคงระดับ SNR ในระดับที่เหมาะสม หรือ ณ ระดับอัตราบิตผิดพลาด เท่ากัน รหัสแอลดีพีซีให้เกนของ SNR เขาใกล้ขอบเขตของแชนนอน

รหัสแอลดีพีซีสามารถแบ่งออกเป็นสองประเภทหลักคือ รหัสแอลดีพีซีที่มีการกระจายตัวของเลขหนึ่งในเมตริกส์พาริตี้เช็คเป็นแบบคงที่ (Regular LDPC codes) ซึ่งเป็นรหัสที่มีรูปแบบเดียวกับรหัสของ R.Gallagerและรหัสแอลดีพีซีอีกชนิดหนึ่งที่มีการกระจายตัวของเลขหนึ่งเป็นแบบไม่คงที่ (Irregular LDPC codes) ซึ่งเป็นรหัสที่พัฒนามาจาก Regular LDPC codes การเข้ารหัสจะเริ่มต้นจากการสร้างเมตริกส์พาริตี้เช็ค H ในหัวข้อถัดไปจึงจะอธิบายถึงโครงสร้างของเมตริกส์H จากนั้นจะกล่าวถึงขั้นตอนและวิธีการถอดรหัสแอลดีพีซี

2.3.1 รหัสแอลดีพีซีที่มีการกระจายตัวของเลขหนึ่งเป็นแบบคงที่ (Regular LDPC codes)

รหัสแอลดีพีซีที่มีการกระจายตัวของเลขหนึ่งเป็นแบบคงที่ (Regular LDPC codes) นั้นมีที่มา จากการที่จำนวนเลขหนึ่งในแต่ละแถว หรือแต่ละหลักของเมตริกส์พาริตี้เป็นค่าคงที่

นิยาม : Regular LDPC codes

รหัสแอลดีพีซีที่มีการกระจายตัวของเลขหนึ่งเป็นแบบคงที่$C(n, W_r, W_c)$ นิยามโดยเมตริกส์พาริตี้เช็คขนาด$m\times n$ เมื่อ

1) $\mathbf{H}_{m\times n}$ที่สร้างขึ้นเกิดจากการการสุ่มข้อมูลศูนย์หนึ่ง
2) W_cคือจำนวนเลขหนึ่งในหลักของเมตริกส์พาริตี้เช็คโดยที่$W_c<<m$
3) W_rคือจำนวนเลขหนึ่งในแถวของเมตริกสพ์าริตี้เช็คและ$W_r= W_c(n/k)$, $W_r<<n$
4) **H** ที่สร้างขึ้นจะต้องปราศจากไซเคิลขนาดเท่ากับ 4
5) อัตรารหัส$R = 1 - (W_c/W_r)$

ค่าความหนาแน่นของเลขหนึ่ง$\rho = (W_r / n) = (W_c/m)$ ทำให้ $m = (W_c / W_r)\cdot n$ และ$\lim_{n\to\infty}\rho = 0$ รหัสนี้มีความยาวข้อมูลอินพุตเท่ากับ$n - m$ บิตความยาวคำรหัสเท่ากับn และจำนวนพาริตี้เช็คเท่ากับm บิต

พิจารณา Regular LDPC ขนาด (10, 5) ซึ่งมีค่า$W_c = 2$ และ$W_r = W_c(n/k) = 2 \times (10/5) = 4$ เมตริกส์**H** ที่ได้คือ

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}_{(m=5,n=10)}$$

ตัวอย่างเมตริกส์พาริตี้เช็คที่ได้นั้นสอดคล้องกับเงื่อนไขทั้ง 4 ข้อกล่าวคือ$W_c = 2$, $W_r = 4$ รวมถึงปราศจากไซเคิลขนาดเท่ากับ 4 และเพื่อความเข้าใจคุณสมบัติของไซเคิลขนาดเท่ากับ 4 พิจารณานิยามของไซเคิลของรหัสแอลดีพีซี

นิยาม: ไซเคิลขนาดเท่ากับ 4

เมตริกส์พาริตี้เช็คใดๆจะมีไซเคิลขนาดเท่ากับ 4 เมื่อตำแหน่งของเลข 1 ใน**H** เกิดลูปปิดตามสมการนี้

$$\left(A_{i,j}\right),\left(A_{i,b}\right),\left(A_{a,b}\right),\left(A_{a,j}\right) \tag{1.27}$$

เมื่อA เป็นตำแหน่งของเลข 1 ใน**H** และค่าคงที่i, j, a, b เป็นค่าของแถวและหลักของ**H** โดยที่$i, a \leq m$ และ$j, b \leq n$ หรืออาจกล่าวได้ว่าไซเคิลขนาดเท่ากับ 4 ในเมตริกส์พาริตี้เช็คคือลูปปิดของเลขหนึ่งที่มีการใช้แถวและหลักร่วมกันเท่ากับ 2 แถวและ 2 หลัก

พิจารณา Regular LDPC ขนาด (10, 5) ซึ่งมีค่า$W_c = 2$ และ$W_r = W_c(n/k) = 4$ และมีไซเคิลขนาดเท่ากับ 4

$$H = \begin{bmatrix} \tilde{1} & 1 & 1 & \tilde{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \hat{1} & \hat{1} & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & \hat{1} & \hat{1} & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ \tilde{1} & 0 & 0 & \tilde{1} & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}_{(m=5,n=10)}$$

จากเมตริกส์H จะพบว่าเกิดไซเคิลขนาดเท่ากับ 4 จำนวนถึง 2 ไซเคิลกล่าวคือ
ไซเคิลที่ 1: ตำแหน่งของiกับลูปปิด $(A_{1,1}),(A_{1,4}),(A_{5,4}),(A_{5,1})$
ไซเคิลที่ 2:ตำแหน่งของiกับลูปปิด $(A_{2,5}),(A_{2,6}),(A_{3,6}),(A_{3,5})$

เหตุผลของไซเคิลขนาดเท่ากับ 4 ที่เป็นเรื่องต้องห้ามสำหรับรหัสแอลดีพีซีก็คืออัลกอริทึมสำหรับการถอดรหัสนั้นจะอาศัยหลักการของความน่าจะเป็นในการส่งผ่านข้อมูลโดยที่ความน่าจะเป็นของแต่ละเหตุการณ์นั้นเป็นอิสระต่อกันซึ่งผลของการมีไซเคิลนี้จะทำให้ความน่าจะเป็นในการส่งผ่านข้อมูลนี้ไม่เป็นอิสระต่อกันและจะส่งผลต่อสมรรถนะของการถอดรหัสเป็นอย่างมากผลกระทบของการมีไซเคิลขนาดเท่ากับ 4 นี้จะกล่าวโดยละเอียดอีกครั้งในเรื่องของการถอดรหัส

2.3.2 รหัสแอลดีพีซีที่มีการกระจายตัวของเลขหนึ่งเป็นแบบไม่คงที่ (Irregular LDPC codes)

รหัสแอลดีพีซีที่มีการกระจายตัวของเลขหนึ่งเป็นแบบไม่คงที่ (Irregular LDPC codes)นั้นมีที่มาจากการที่จำนวนเลขหนึ่งในแต่ละแถวหรือแต่ละหลักของเมตริกส์พาริตี้เช็คมีจำนวนไม่คงที่ในทุกแถวหรือทุกหลักของเมตริกส์พาริตี้เช็ค Irregular LDPC Codes ถูกพัฒนาขึ้นมาในปีค.ศ.2001(2544) โดย T.Richardsonและด้วยเหตุที่การกระจายตัวของเลขหนึ่งเป็นแบบไม่คงที่ค่าความหนาแน่นของเลขหนึ่งρที่นิยามไว้กับ Regular LDPC นั้นจึงเปลี่ยนไปดังนี้

นิยาม: Irregular LDPC codes คือรหัสแอลดีพีซีที่การกระจายตัวเลขหนึ่งไม่คงที่
1) ค่าความหนาแน่นของเลขหนึ่งของแต่ละแถวนิยามโดย$[\rho_2, \rho_3, \ldots, \rho_n]$
2) ค่าความหนาแน่นของเลขหนึ่งของแต่ละหลักนิยามโดย$[\lambda_2, \lambda_3, \ldots, \lambda_n]$
3) อัตรารหัส$R = 1 - \left[\left(\sum_{j=2}^{n} \rho_j / j\right) / \left(\sum_{j=2}^{n} \lambda_j / j\right)\right]$

ด้วยวิธีการของT.Richardsonนั้นสมรรถนะการทำงานของ Irregular LDPC Codes ดีกว่า Regular LDPCCodes ของ D.J.C. Mackey แต่ก็มีหลายงานวิจัยพบว่า Irregular LDPC ในแบบของT.Richardsonนั้นจะทำงานได้ดีสำหรับอัตรารหัส$R \leq 3/4$ และที่ความยาวคำรหัส$n \geq 5000$

2.3.3 การเข้ารหัสแอลดีพีซี

การเข้ารหัสของรหัสแอลดีพีซีมีลักษณะเป็นรหัสเชิงเส้นแบบบล็อกซึ่งมีวิธีการเข้ารหัสหลายวิธี เช่นการเข้ารหัสโดยใช้เมตริกซ์กำเนิดGซึ่งการเข้ารหัสโดยอาศัยเมตริกซ์กำเนิดGนั้นถ้าให้รหัส แอลดีพีซีมีอัตรารหัส (Rate) R = k/nจะหาคำรหัสได้โดย

$$c = mG \tag{1.28}$$

เมื่อ **m** คือ ข้อมูล

cคือ คำรหัส

G คือ เมตริกซ์กำเนิด

โดยที่ขนาดของเมตริกซ์ข้อมูล **m** กับเมตริกซ์กำเนิด **G** และเมตริกซ์คำรหัส **c** เท่ากับ $1 \times k$, $k \times n$, $1 \times n$ ตามลำดับ ซึ่งจะสังเกตเห็นว่าการบวกทั้งหมดนั้นเป็นการบวกแบบมอดูโล2 กล่าวคือ 0+0 = 0 , 1+0 =1 , 0+1 = 1 , 1+1 = 0 โดยที่เมตริกซ์กำเนิด **G** และเมตริกซ์พาริตี้เช็ค **H** มีค่าเท่ากับ (n-k) × n และมีความสัมพันธ์กันดังนี้

$$HG^T = 0 \tag{1.29}$$

เมื่อ **H** คือ เมตริกซ์พาริตี้เช็ค

0 คือ เมตริกซ์ศูนย์

แต่การสร้างเมตริกซ์กำเนิด**G**เพื่อนำไปใช้สร้างคำรหัส **c**ต้องอาศัยเทคนิคของGaussian elimination และถึงแม้จะสร้างได้แต่จะทำให้โครงสร้างของรหัสเปลี่ยนไป เนื่องมาจากการถอดรหัสของรหัสแอลดีพีซี นั้นจะต้องอ้างอิงกับโครงสร้างของเมตริกซ์พาริตี้เช็ค**H**ด้วยเหตุผลดังกล่าวจึงไม่นิยมที่จะทำการ เข้ารหัสโดยใช้เมตริกซ์กำเนิด**G**

การเข้ารหัสแอลดีพีซีวิธีที่ 2 ซึ่งเป็นที่นิยมใช้คือการเข้ารหัสด้วยเมตริกซ์พาริตี้เช็ค **H** โดยตรงจึงเป็นทางเลือกที่นิยมใช้

จาก (2.28) และ (2.29) เมตริกซ์คำรหัส c และเมตริกซ์พาริตี้เช็ค H มีความสัมพันธ์ได้แก่

$$cH^T = 0 \tag{1.30}$$

โดยเมตริกซ์ศูนย์**0** นี้มีขนาดเท่ากับ1x(n-k)

ให้ข้อมูลเริ่มต้นเป็นส่วนหนึ่งของคำรหัสเพื่อความสะดวกในภาครับรูปแบบหนึ่งของเมตริกซ์คำรหัส c คือ

$$\mathrm{c}=\begin{bmatrix}c_1 & c_2 & \cdots & c_n\end{bmatrix}=\begin{bmatrix}m_1 & m_2 & \cdots & m_k & p_1 & p_2 & \cdots & p_{n-k}\end{bmatrix}=[\mathrm{m}\,|\,\mathrm{p}] \qquad (1.31)$$

เขียนเมตริกซ์พาริตี้เช็ค**H** ในรูป

$$\mathrm{H}=[\mathrm{H}_1\,|\,\mathrm{H}_2]$$
$$\mathrm{H}^T=\begin{bmatrix}\mathrm{H}_1^T\\ \mathrm{H}_2^T\end{bmatrix} \qquad (1.32)$$

โดย $\mathbf{H}_1$และ$\mathbf{H}_2$มีขนาดเท่ากับ (n-k) x kและ(n-k) x (n-k)ตามลำดับ ดังนั้น

$$\mathrm{cH}^T=[\mathrm{m}\,|\,\mathrm{p}]\begin{bmatrix}\mathrm{H}_1^T\\ \mathrm{H}_2^T\end{bmatrix}=0$$
$$=\mathrm{mH}_1^T+\mathrm{pH}_2^T=0 \qquad (1.33)$$

ดังนั้นเมตริกซ์บิตพาริตี้เช็คทั้งหมดหาได้จาก

$$\mathrm{p}=\mathrm{mH}_1^T\left(\mathrm{H}_2^T\right)^{-1} \qquad (1.34)$$

โดยใช้คุณสมบัติการบวกกันแบบมอดูโล2

สังเกตว่าเมตริกซ์$\mathbf{H}_2$เป็นเมตริกซ์จตุรัสขนาดเท่ากับ (n-k)x(n-k)

2.3.4 การถอดรหัสแอลดีพีซี

การเข้ารหัสมีผลทำให้ข้อมูลแต่ละบิตมีความสัมพันธ์กันผ่านทางโครงสร้างของเมตริกส์พาริตี้เช็ค**H** ซึ่งการถอดรหัสก็จะอาศัยความสัมพันธ์เหล่านี้มาช่วยในการถอดรหัสอัลกอริทึมสำหรับการถอดรหัสแอลดีพีซีนั้นวิธีการเริ่มแรกมีชื่อเรียกว่าsum-product algorithm (SPA) ซึ่งเป็นรูปแบบการถอดรหัสที่เรียกว่า soft iterative decodingแต่เนื่องจากวิธีการข้างต้นเมื่อนำไปใช้บนฮาร์ดแวร์จะมีความซับซ้อนสูงดังนั้นจึงได้มีการพัฒนารหัสให้มีความซับซ้อนน้อยลงได้มาเป็นอัลกอรึทึมที่อยู่ในลอกโดเมนLog-Domain Algorithm แต่อย่างไรก็ตามอัลกอริทึมดังกล่าวยังคงมีความซับซ้อนอยู่เช่นกัน เมื่อนำไปใช้งานจริง จึงได้มีการพัฒนาอัลกอริทึมเพื่อลดความซับซ้อนลงไปอีก จนกระทั่งได้เป็นอัลกอริทึมการหาค่าต่ำสุด (Min-Sum Algorithm) และอัลกอริทึมการค่าต่ำสุดดัดแปลง (Modified Min-Sum Algorithm) ซึ่งจะมีความเหมาะสมมากกว่าเมื่อนำไปใช้งานบนฮาร์ดแวร์โดยขั้นตอนการถอดรหัสจะประกอบด้วยขั้นตอนหลักสองขั้นตอนคือสร้าง

สมการจากโครงสร้างของเมตริกส์พาริตี้เช็ค**H** แล้วจึงเขียนแผนภาพTanner Graph จากนั้นจึงทำการคำนวณหาค่าของข้อมูลแต่ละบิตตามโครงสร้างของอัลกอริทึมที่ใช้ในการถอดรหัส

2.3.4.1 สร้างสมการจากโครงสร้างของเมตริกส์พาริตี้เช็ค H และเขียนแผนภาพ TannerGraph

พิจารณา Regular LDPC ขนาด (10, 5) ซึ่งมีเมตริกส์พาริตี้เช็ค**H** ที่ได้คือ

$$H=\begin{bmatrix}1&1&1&1&0&0&0&0&0&0\\1&0&0&0&1&1&1&0&0&0\\0&1&0&0&1&0&0&1&1&0\\0&0&1&0&0&1&0&1&0&1\\0&0&0&1&0&0&1&0&1&1\end{bmatrix}_{(m=5,n=10)}$$

จากโครงสร้างของเมตริกส์พาริตี้เช็ค**H** จะได้ความสัมพันธ์เป็นสมการที่แสดงความสัมพันธ์ของแต่ละบิตโดยที่แต่ละแถวของเมตริกส์**H** จะเรียกว่าเช็คโหนด(Check Node)และแต่ละหลักจะเรียกว่าสัญลักษณ์โหนด หรือ บิตโหนด(Bit Node) สมการแสดงได้โดย

โหนดเช็ค ที่ 1: $c_1+c_2+c_3+c_4=0$

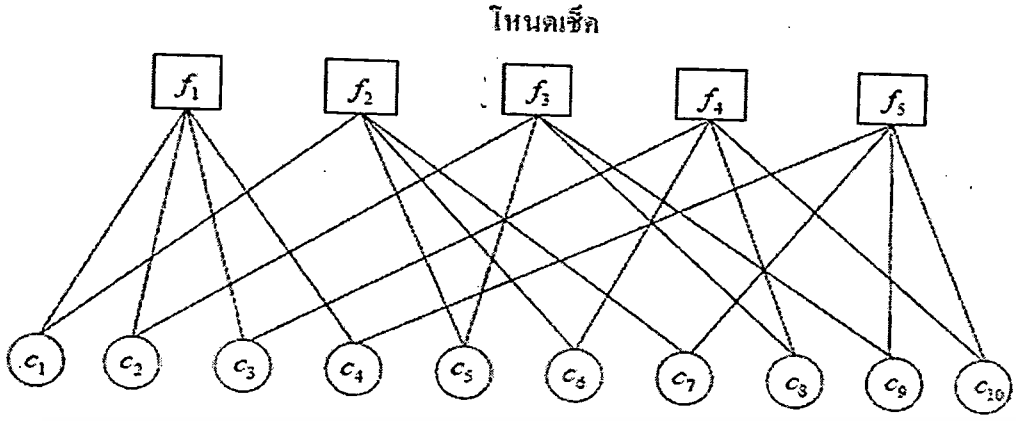
โหนดเช็ค ที่ 2: $c_1+c_5+c_6+c_7=0$

โหนดเช็ค ที่ 3: $c_2+c_5+c_8+c_9=0$

โหนดเช็ค ที่ 4: $c_3+c_6+c_8+c_{10}=0$

โหนดเช็ค ที่ 5: $c_4+c_7+c_9+c_{10}=0$

โดยc_i แทนตำแหน่งของเลขหนึ่งของแต่ละบิตโหนดในเช็คโหนดที่กำลังพิจารณาจากสมการข้างต้นสามารถสร้างกราฟ Tannerเพื่อช่วยในการถอดรหัสดังรูปที่ 2.3โดยเริ่มจากวาดรูปสี่เหลี่ยมตามจำนวนของเช็คโหนดจากนั้นวาดรูปวงกลมตามจำนวนของบิตโหนดในขั้นตอนสุดท้ายเป็นการลากเส้นตรงเชื่อมความสัมพันธ์ระหว่างเช็คโหนดและบิตโหนดตามความสัมพันธ์ในสมการ



รูปที่ 2.3กราฟแทนเนอร์

จาก Tanner Graph จะสามารถถอดรหัสได้ด้วยหลายวิธีโดยขั้นการถอดรหัสแอลดีพีซีจะมีขั้นตอนในการคำนวณประกอบด้วย 5 ขั้นตอน

2.3.4.2 ขั้นตอนการถอดรหัสด้วยวิธีการของความน่าจะเป็น (Sum-Product Algorithm)

ในการถอดรหัสพาริตี้เช็คความหนาแน่นต่ำเป็นการคำนวณหาค่าบิตโหนดโดยใช้สมการ 2.35 และสมการ 2.36ซึ่งจะทำการคำนวณในครั้งแรกและครั้งเดียวเท่านั้น นั่นคือการคำนวณค่าเริ่มต้น จากนั้นในการคำนวณหาค่าบิตโหนดครั้งต่อๆ ไปจะใช้สมการ 2.39และสมการ 2.40แทน

ขั้นตอนที่ 1 การคำนวณค่าเริ่มต้น

การคำนวณค่าจากโหนดบิต ในกรณีที่ผ่านช่องสัญญาณรบกวนแบบ AWGN

$$q_{ij}(0) = 1 - P_i = P_r(c_i = 0 / y_i) = \frac{1}{1+e^{+2y_i/\sigma^2}} \tag{1.35}$$

$$q_{ij}(1) = P_i = \frac{1}{1+e^{+2y_i/\sigma^2}} \tag{1.36}$$

นิยามให้

$q_{ij}(0)$, $q_{ij}(1)$ คือ ค่าคาดคะเนความน่าจะเป็นของบิต '0' และบิต '1' ที่โหนดบิต iส่งไปยังโหนดเช็ค j

y_iคือ บิตข้อมูลข่าวสารที่ได้รับการเข้ารหัสและผ่านช่องสัญญาณ หรือค่าที่ได้รับของบิตรหัส i

σ^2คือ ค่าความแปรปรวนของสัญญาณรบกวน

ขั้นตอนที่ 2 การคำนวณหาค่าเช็คโหนด.

$$r_{ij}(0) = 0.5 + 0.5 \prod_{i \in R_{fv}} \left[1 - 2q_{ij}(1)\right] \tag{1.37}$$

$$r_{ij}(1) = 1 - r_{ji}(0) \tag{1.38}$$

ขั้นตอนที่ 3 การคำนวณหาค่าบิตโหนดในรอบที่สอง

$$q_{ij}(0) = K_{ij}(1 - P_i) \prod_{j \in c_{Nj}} r_{ji}(0) \tag{1.39}$$

$$q_{ij}(1) = K_{ij} P_i \prod_{j \in c_{Nj}} r_{ji}(1) \tag{1.40}$$

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

ขั้นตอนที่ 4 การตัดสินอย่างละเอียด

หาความน่าจะเป็นของค่าบิตโหนดqทุกตัว

$$Q_i(0) = K_{ij}(1 - P_i) \prod_{j \in c_i} r_{ji}(0) \tag{1.41}$$

$$Q_i(1) = K_{ij} P_i \prod_{j \in c_i} r_{ji}(1) \tag{1.42}$$

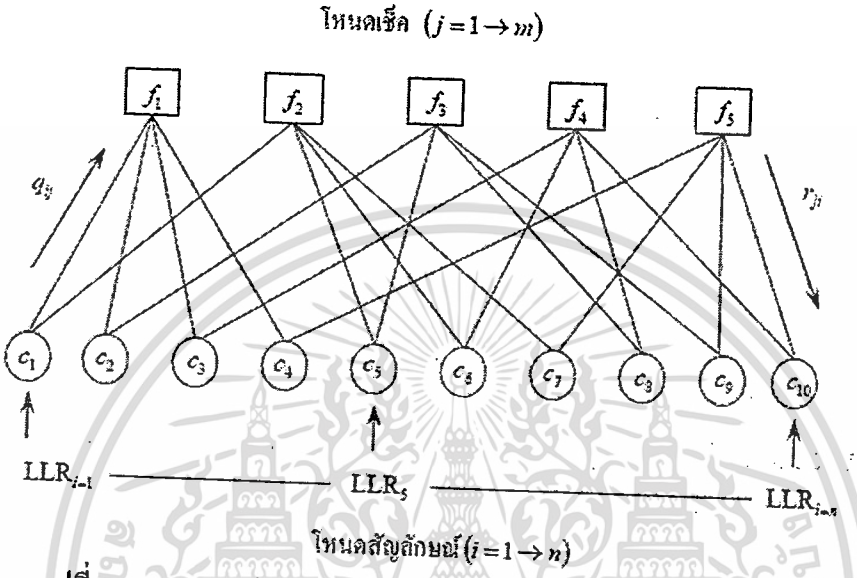
โดยเลือกค่า K_{ij}เพื่อให้ $Q_i(0) + Q_i(1) = 1$

ขั้นตอนที่ 5 การตัดสินใจอย่างหยาบ

$$c_i \begin{cases} 1 & if \quad Q_i(1) < Q_i(0) \\ 0 & otherwise \end{cases} \tag{1.43}$$

ถ้า $\mathbf{cH}^T = \mathbf{0}$ หรือจำนวนของการวนซ้ำมากกว่าขอบเขตที่กำหนดแล้ว ให้หยุดและกลับไปทำงานในขั้นตอนที่ 2

2.3.4.3 ขั้นตอนการถอดรหัสด้วยวิธีการของความน่าจะเป็นในรูปแบบของลอการิทึม (Log-Domain Algorithm)



รูปที่ 2.4แสดงการส่งผ่านข้อมูลระหว่าง บิตโหนด และ เช็คโหนด

โดยหลักการทำงานจะเป็นการแลกเปลี่ยนข่าวสารแบบซอฟต์ระหว่างบิตโหนดiและเช็คโหนดj พิจารณารูปที่2.4 อินพุตของการถอดรหัสจะอยู่ในรูปของอัตราส่วนความน่าเป็นจริงแบบล็อก (Log Likelihood Ratios : LLRs) ของตัวแปรสุมc_iตามสมการโดยในแต่ละบิตโหนดสัญลักษณ์iจะส่งค่าความน่าจะเป็นของข่าวสารแบบซอฟต์ไปที่โหนดเช็คjผ่านตามเส้นความสัมพันธ์ที่เชื่อมถึงกันและจากนั้นที่โหนดเช็คj ก็จะทำการคำนวณค่าความน่าจะเป็นของข่าวสารที่ส่งมาจากบิตโหนดI แล้วจึงส่งผลที่ได้ของข่าวสารแบบซอฟต์ไปให้บิตโหนดi อีกครั้งเพื่อนำข้อมูลที่ได้ไปใช้ในการตัดสินใจว่าควรจะเป็น 0 หรือ 1

ขั้นตอนที่ 1 การคำนวณค่าเริ่มต้น

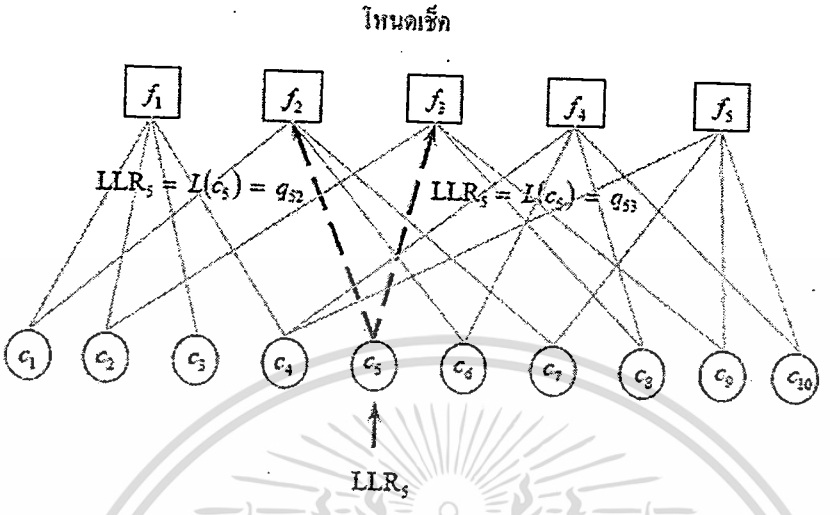
$$L(q_{ij}) = L(c_i) = 2y_i / \sigma^2 \tag{1.44}$$

$$L(c_i) = \mathrm{LLR}_i = log\left[P(c_i = 0 \mid y_i) / P(c_i = 1 \mid y_i)\right] \tag{1.45}$$

เมื่อ $L(c_i)$ คืออัตราส่วนความน่าเป็นจริงแบบล็อก

y_i คือสัญญาณที่ได้รับผ่านช่องสัญญาณ

σ^2 คือค่าเบี่ยงเบนของสัญญาณรบกวนเกาส์แบบขาว



รูปที่2.5แสดงค่า $L(qij)$ที่ส่งจาก บิตโหนดi ไปยัง เช็คโหนดj

ขั้นตอนที่ 2คำนวณค่า$L(r_{ji})$ที่ส่งจากเช็คโหนดj ไปที่บิตโหนดi ตามเส้นความสัมพันธ์ที่เชื่อมถึงกันผ่านสมการของในแต่ละบิตi ตั้งแต่บิตที่ 1 ถึงบิตที่n

$$L\left(r_{ji}\right)=\prod_{i'V_{j\backslash i}}\alpha_{i'j}\,\phi\left[\sum_{i'\in V_{j\backslash i}}\phi\left(\beta_{i'j}\right)\right] \tag{1.46}$$

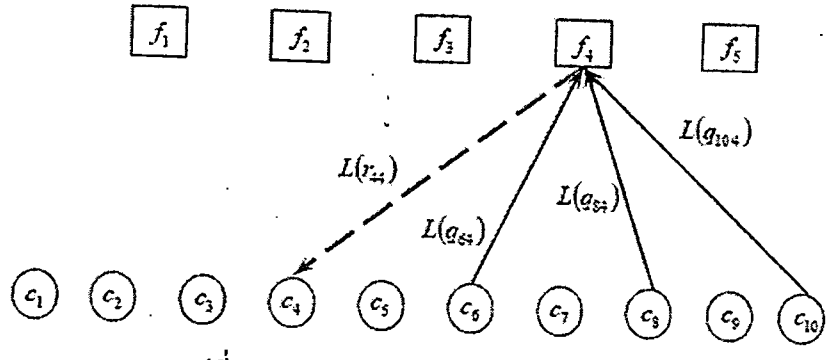
เมื่อ

$$\alpha_{ij}=sgn\left\{L\left(q_{ij}\right)\right\} \text{ และ } \beta_{ij}=\left|L\left(q_{ij}\right)\right|$$

นิยามให้

$$\phi(x)=\log\left\{\left(e^{x}+1\right)/\left(e^{x}-1\right)\right\} \tag{1.47}$$

$V_j\backslash i$ โดยแทนการพิจารณาข่าวสารจากทุกบิตโหนดที่เชื่อมต่อกับเช็คโหนดj ยกเว้นบิตโหนดที่กำลังพิจารณารูปที่2.6แสดงแผนภาพการคำนวณค่า$L(r_{44})$

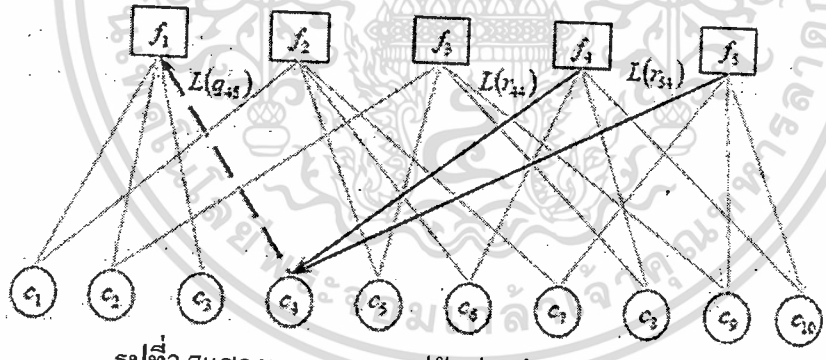


รูปที่2.6แสดงแผนภาพการคำนวณค่า$L(r_{44})$

ขั้นตอนที่ 3จะเป็นการปรับปรุงข่าวสารของ$L(qij)$เพื่อที่จะใช้เป็นอินพุตของการถอดรหัสแบบวนซ้ำที่ส่งจากบิตโหนดi ไปยังเช็คโหนดj ของในแต่ละบิตi ตั้งแต่บิตที่ 1 จนถึงบิตที่n ผ่านทางสมการที่ 3.21

$$L(q_{ij}) = L(c_i) + \sum_{j' \in C_i \setminus j} L(r_{j'i}) \tag{1.48}$$

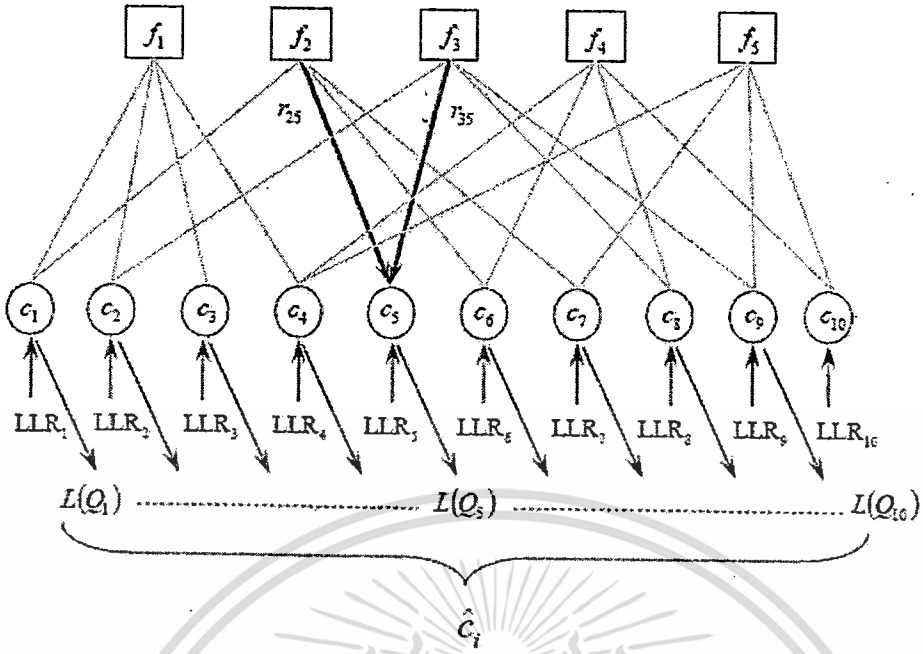
$C_i \setminus j$ แทนการพิจารณาผลรวมของข่าวสาร$L(r_{ji})$ จากทุกเช็คโหนดj ที่เชื่อมต่อกับบิตโหนดยกเว้นข่าวสารที่ใช้เส้นทางเดียวกับ$L(q_{ij})$ รูปที่7 แสดงแผนภาพการปรับปรุงข่าวสารของ$L(q_{ij})$



รูปที่2.7แสดงแผนภาพการปรับปรุงข่าวสารของ$L(q_{45})$

ขั้นตอนที่ 4เป็นการคำนวณหาค่าซอฟต์เอาต์พุตของการถอดรหัสของแต่ละบิตi ตั้งแต่บิตที่ 1ถึงบิตที่n ผ่านสมการที่ 2.49

$$L(Q_i) = L(c_i) + \sum_{j \in C_i} L(r_{ji}) \tag{1.49}$$



รูปที่2.8แสดงแผนภาพการหาค่าซอฟต์เอาต์พุตของการถอดรหัส

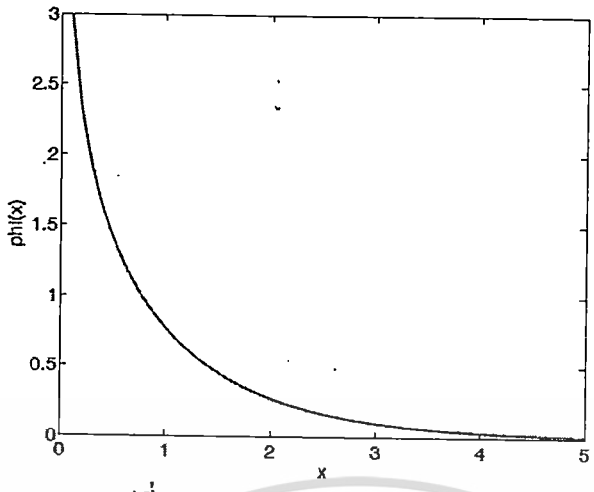
ขั้นตอนที่ 5เป็นการนำค่าซอฟต์เอาต์พุตที่ได้จากขั้นตอนที่ 4 ของแต่ละบิตมาทำการตัดสินใจแบบหยาบผ่านสมการที่ 2.50

$$c_i \begin{cases} 1 & if \quad L(Q_i) < 0 \\ 0 & otherwise \end{cases} \tag{1.50}$$

จากขั้นตอนที่ 1 – 5 ก็จะเป็นการเสร็จสิ้นขั้นตอนในการถอดรหัสหนึ่งลูปซึ่งจะพบว่าถ้าระบบที่ใช้การถอดรหัสแบบไม่มีการวนซ้ำขั้นตอนที่ 3 ก็สามารถยกเลิกและข้ามมายังขั้นตอนที่ 4 ได้เลยหรือในกรณีที่ใช้การวนซ้ำก็จะทำตามขั้นตอนที่ 1 – 5ตามจำนวนรอบการวนซ้ำที่ได้กำหนดไว้หรือใช้สมการ$\hat{c}\cdot\mathbf{H}^T = \mathbf{0}$ เป็นเงื่อนไขเสร็จสิ้นขั้นตอนในการถอดรหัส

2.3.4.4 ขั้นถอดรหัสแอลดีพีซีโดยใช้วิธีการผลรวมต่ำสุด (Min-Sum Algorithm)

วิธีการผลรวมต่ำสุด (MS) นั้นคล้ายวิธีการถอดรหัสด้วยวิธีการความน่าจะเป็น (Sum-Product) แต่ถูกนำมาแก้ไขในช่วงของการประมวลผลเช็คโหนดให้เป็นแบบประมาณค่า (approximation) ซึ่งจะทำให้มีข้อได้เปรียบในการนำมาใช้งานจริงมากกว่ารูปแบบความน่าจะเป็น



รูปที่2.9กราฟการคำนวณหาค่า $\phi(x)$

จากรูปที่ 2.9เป็นการคำนวณหาค่า $\phi(x)$ ในขั้นตอนที่ 2 จะเห็นได้ว่าเมื่อค่า x มีค่าน้อย ผลลัพธ์จะยิ่งมีค่ามาก ทำให้สรุปได้ว่าค่าต่ำสุดจะมีผลมากที่สุด ดังนั้นจึงนำมาปรับปรุงสมการของ Lijใน log-domain ได้ดังนี้

จากสมการเดิม

$$L(r_{ji}) = \prod_{i' V_{j\backslash i}} \alpha_{i'j} \phi\left[\sum_{i' \in V_{j\backslash i}} \phi(\beta_{i'j})\right]$$

ปรับปรุงฟังก์ชั่น $\phi(x)$

$$\phi\left[\sum_{i' \in V_{j\backslash i}} \phi(\beta_{i'j})\right] = \phi\left[\phi\left(\min_{i'} \beta_{i'j}\right)\right] = \min_{i'} \beta_{i'j} \qquad (1.51)$$

ดังนั้นสมการหาค่าเช็คโหนด2.46 ในรูปแบบความน่าจะเป็นในลอกโดเมน (log-domain) จะถูกแทนที่ด้วยสมการที่ 2.51 เพื่อให้อยู่ในวิธีการหาค่าต่ำสุด (Min-Sum) ดังนี้

เมื่อ

$$\alpha_{ij} = sgn\{L(q_{ij})\} \text{ และ } \beta_{ij} = |L(q_{ij})|$$

ปรับปรุงได้เป็น

$$L(r_{ji}) = \left(\prod_{i' \in R(i)\backslash j} \text{sign}(L(q_{i'j}))\right) \min_{i' \in R(i)\backslash j} |L(q_{i'j})|$$

เมื่อสรุปขั้นตอนทั้งหมดของวิธีการหาค่าต่ำสุดจะแสดงให้เห็นได้ดังนี้

ขั้นตอนที่ 1 การคำนวณค่าเริ่มต้น เนื่องจากเป็นขั้นตอนที่มาจากการประมาณค่า ค่า AWGN จึงไม่มีผลต่อค่าเริ่มต้นจึงปรับปรุงสมการได้ดังนี้

$$L(q_{ji}) = L(c_i) = y_i \tag{1.52}$$

ขั้นตอนที่ 2คำนวณหาค่าที่เช็คโหนด (CNU)

$$L(r_{ji}) = \left(\prod_{i' \in R(i) \setminus j} \text{sign}(L(q_{i'j})) \right) \min_{i' \in R(i) \setminus j} \left| L(q_{i'j}) \right| \tag{1.53}$$

ขั้นตอนที่ 3คำนวณหาค่าที่บิตโหนด (VNU)

$$L(q_{ij}) = L(c_i) + \sum_{j' \in C_i \setminus j} L(r_{j'i}) \tag{1.54}$$

ขั้นตอนที่ 4ตัดสินใจแบบละเอียด

$$L(Q_i) = L(c_i) + \sum_{j \in C_i} L(r_{ji}) \tag{1.55}$$

ขั้นตอนที่ 5ตัดสินใจแบบหยาบ

$$\hat{c}_i = \begin{cases} 1 & if \quad L(Q_i) < 0 \\ 0 & otherwise \end{cases} \tag{1.56}$$

เมื่อทำครบทั้งหมดครบ 5 ขั้นตอน จะเป็นการคำนวณครบหนึ่งรอบจากนั้นจะมาหาคำนวณพาริตี้เช็ค $H \cdot (\hat{c}_1, \hat{c}_2, ..., \hat{c}_n)^T = 0$ โดยเมื่อผลลัพธ์เป็นไปตามเงื่อนไข การถอดรหัสถือว่าเสร็จสิ้น แต่ถ้าไม่เป็นไปตามเงื่อนไขจะกลับไปทำการคำนวณที่ขั้นตอนที่ 2 – 5 จนกว่าจะเป็นไปตามเงื่อนไข หรือจนกว่าจะคำนวณครบตามจำนวนการวนซ้ำที่กำหนดไว้

2.3.4.5 ขั้นถอดรหัสแอลดีพีซีโดยใช้วิธีการผลรวมต่ำสุดดัดแปลง (Modified Min-Sum Algorithm)

เป็นการนำเอาวิธีการหาค่าผลรวมต่ำสุด (MS) มาปรับปรุงโดยจะให้ประสิทธิภาพที่ดีกว่าเดิม เนื่องจากวิธีการผลรวมต่ำสุดนั้นได้ทำการลดความซับซ้อนของการประมวลผลด้วยวิธีการประมาณค่า ทำให้ประสิทธิภาพในการถอดรหัสลดลง โดยการปรับปรุงนั้นจะเป็นการปรับปรุงเพียงเล็กน้อยเพื่อให้การคำนวณไม่ซับซ้อนมากนัก เพียงแค่ชดเชยประสิทธิภาพที่หายไปเท่านั้น ดังนั้นจะสามารถประยุกต์ได้โดยการนำเสนอตามบทความที่[5] ด้วยการปรับปรุงในสมการที่ 2.54 โดยนำมาใส่ค่า scaling factor(γ)สามารถเขียนออกมาเป็นสมการได้ดังนี้

$$L(q_{ij}) = \left(L(c_i) + \sum_{j' \in C_i \setminus j} L(r_{j'i}) \right) * \gamma \qquad (1.57)$$

ซึ่งค่าที่เหมาะสมจะมีอยู่ระหว่าง $\gamma = 0.6\text{-}0.8$ สำหรับค่าที่เหมาะสมที่จะนำไปใช้ในการออกแบบฮาร์ดแวร์จะใช้ค่า $\gamma = 0.75$ ซึ่งเราใช้ในการวิจัยครั้งนี้

2.4 งานวิจัยที่เกี่ยวข้อง

ในช่วงเวลาหลายปีที่ผ่านมา ได้มีการค้นคว้าเกี่ยวกับการวิเคราะห์สัญญาณไฟฟ้ากล้ามเนื้อแขน ซึ่งจะมีทั้งการวิเคราะห์ความผิดปรกติของกล้ามเนื้อ และการจำแนกลักษณะเฉพาะของกล้ามเนื้อ ดังนั้นผู้จัดทำจึงนำเอางานวิจัยเกี่ยวข้องกับการจำแนกลักษณะเด่นของกล้ามเนื้อแขนในปีที่ผ่านมาสามารถกล่าวโดยสรุปได้ดังนี้

รหัสแอลดีพีซีได้รับการคิดค้นขึ้นมาโดย Robert Gallagerแห่งสถาบันMITในปี1963โดยรหัสนี้มีประสิทธิภาพเข้าใกล้แชนนอนลิมิตแต่ในขณะนั้นแทบจะไม่ได้รับความสนใจแต่อย่างใด เนื่องจากการนำไปใช้งานยังมีความซับซ้อนสูง และ 20 ปีถัดจากนั้นMichaelTanner [2] ได้นำเสนอกราฟของรหัสแอลดีพีซีเพื่อให้เข้าใจง่ายขึ้น โดยเรียกว่าbi-partiteกราฟ ถึงอย่างไรก็ตามสมการของรหัสแอลดีพีซียังคงไม่ได้ถูกปรับปรุงแต่อย่างใดจึงทำให้การนำไปใช้งานจริงยังคงซ้ำซ้อนเหมือนเดิม

จนกระทั่งในปี 1996 David MacKay [2]ได้นำรหัสแอลดีพีซี (LDPC) กลับมาใช้ใหม่โดย ได้รับการค้นคว้าวิจัยและพัฒนาให้ดีขึ้นเพื่อทำให้การนำไปใช้งานได้ง่ายยิ่งขึ้น และเนื่องจากรหัสแอลดีพีซีประสิทธิภาพในการแก้ไขข้อผิดพลาดได้ดี

เนื่องจากเราต้องการที่จะทำให้การถอดรหัสข้อมูลเสร็จเร็วมากยิ่งขึ้นหรือทำให้การใช้พื้นที่ของฮาร์ดแวร์มีขนาดเล็กลง โดยจะนำไปใช้บนมาตราฐานของการสื่อสารไร้สาย เราจึงได้นำวิธีการของ[4]ที่มี

ความสามารถในการถอดรหัสให้ได้ข้อมูลเร็วยิ่งขึ้น Xin-Yu Shih และคณะ ได้นำเสนอวิธีการสลับลำดับเมตริกซ์พาริตี้เช็ค (reorder parity check matrix)ที่ใช้บนมาตราฐานไร้สาย จึงทำให้มีความสามารถในการทำงานซ้อนทับกัน (overlap)โดยนำไปใช้บนมาตรฐานไร้สาย IEEE802.16e ซึ่งเมตริกซ์พาริตี้เช็คพื้นฐานจะแสดงดังรูปที่ 2.10

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	-	94	73	-	-	-	-	-	55	83	-	-	7	0	-	-	-	-	-	-	-	-	-	-
2	-	27	-	-	-	22	79	9	-	-	-	12	-	0	0	-	-	-	-	-	-	-	-	-
3	-	-	-	24	22	81	-	33	-	-	-	0	-	-	0	0	-	-	-	-	-	-	-	-
4	61	-	47	-	-	-	-	-	65	25	-	-	-	-	-	0	0	-	-	-	-	-	-	-
5	-	-	39	-	-	-	84	-	-	41	72	-	-	-	-	-	0	0	-	-	-	-	-	-
6	-	-	-	-	46	40	-	82	-	-	-	79	-	-	-	-	-	0	0	-	-	-	-	-
7	-	-	95	53	-	-	-	-	-	14	18	-	0	-	-	-	-	-	0	0	-	-	-	-
8	-	11	73	-	-	-	2	-	-	47	-	-	-	-	-	-	-	-	-	0	0	-	-	-
9	12	-	-	-	83	24	-	43	-	-	-	51	-	-	-	-	-	-	-	-	0	0	-	-
10	-	-	-	-	-	94	-	59	-	-	70	72	-	-	-	-	-	-	-	-	-	0	0	-
11	-	-	7	65	-	-	-	-	39	49	-	-	-	-	-	-	-	-	-	-	-	-	0	0
12	43	-	-	-	-	66	-	41	-	-	-	26	1	-	-	-	-	-	-	-	-	-	-	0

รูปที่2.10เมตริกซ์พาริตี้เช็คอัตรารหัส 1/2 ตามมาตราฐานWiMAXIEEE 802.16e [x].

สังเกตุได้ว่าช่องว่าง (-) ของเมตริกซ์ หรือบล็อคที่ไม่มีการคำนวณมีจำนวนมากพอที่จะทำการสับเปลี่ยนตำแหน่ง แต่การสับเปลี่ยนตำแหน่งจะต้องไม่ส่งผลกระทบต่อประสิทธิภาพการถอดรหัสดังนั้นกฎการสลับตำแหน่งจะแสดงให้เห็นได้รูปแบบด้านล่าง

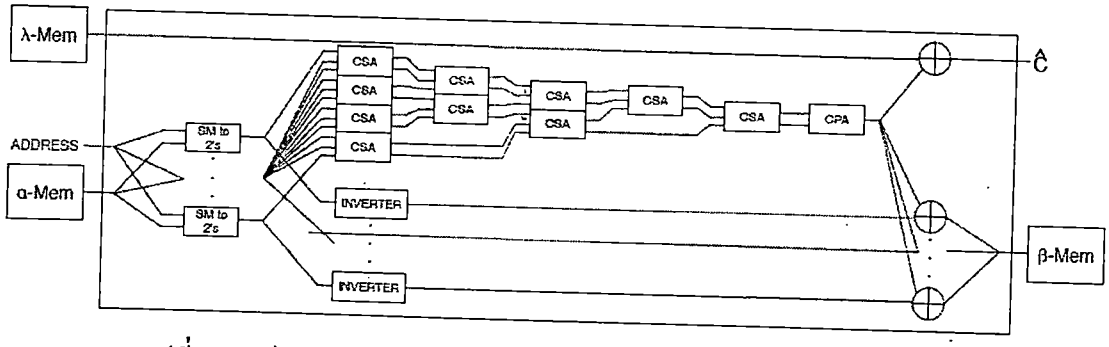
$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \\ b_1 & b_2 & b_3 & b_4 & b_5 \\ c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix} \bullet \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix} = 0 \quad \Rightarrow \quad \begin{bmatrix} b_3 & b_2 & b_1 & b_5 & b_4 \\ a_3 & a_2 & a_1 & a_5 & a_4 \\ c_3 & c_2 & c_1 & c_5 & c_4 \end{bmatrix} \bullet \begin{bmatrix} v_3 \\ v_2 \\ v_1 \\ v_5 \\ v_4 \end{bmatrix} = 0$$

เมตริกซ์มาตราฐาน

เมตริกซ์ที่ถูกสลับตำแหน่งแล้ว

โดยรูปแบบด้านบนจะแสดงให้เห็นว่าไม่ส่งผลกระทบต่อประสิทธิภาพของการถอดรหัสเพราะการคูณกันของเมตริกซ์$\mathbf{Hc}^T = \mathbf{0}$ยังส่งผลให้ผลลัพธ์ที่ได้ยังคงมีค่าเหมือนเดิม

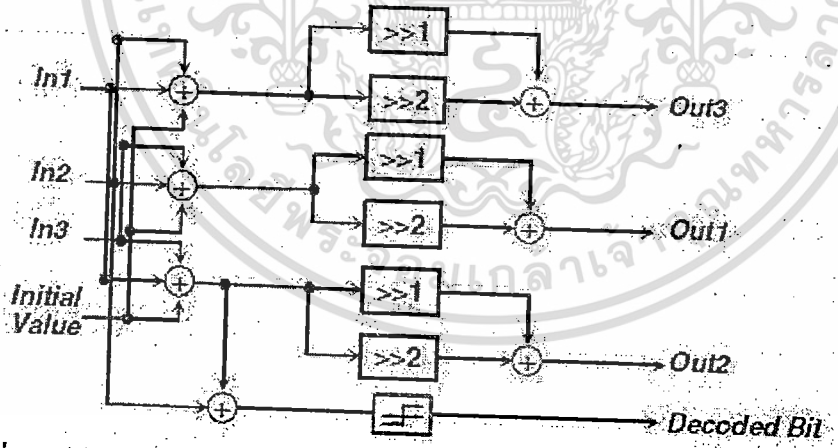
ในส่วนของหน่วยประมวลผลเราได้นำโครงสร้างของ Hiroyuki SHIMAJIRIและคณะ [6]ที่ออกแบบสำหรับการสื่อสารไร้สายตามมาตราฐานIEEE 802.11n โดยเรานำไปใช้ออกแบบฮาร์ดแวร์ในการถอดรหัสด้วยวิธีการหาค่าต่ำสุด (Min-Sum Algorithm)



รูปที่2.11บล็อกไดอะแกรมหน่วยประมวลผล VNU ด้วยวิธีการหาค่าต่ำสุด

จากบล็อกไดอะแกรมตามรูปที่ 2.11 เราจะนำเฉพาะหน่วยประมวลผลVNU มาเป็นต้นแบบเพื่อนำมาออกแบบในโครงสร้างการถอดรหัส ตามโครงสร้างด้านบนจะใช้ Carry save adder (CSA) และ Carry Propagate Adder (CPA) เป็นตัวกระทำการบวกโดยการคำนวณทั้งหมดจะคำนวณในระบบตัวเลข2's complement

เนื่องจากเราต้องการที่จะออกแบบหน่วยประมวลผลในการถอดรหัสสองวิธีด้วยกัน ซึ่งในวิธีการแรกเราได้นำเสนอที่มาไปแล้ว ส่วนในวิธีการถอดรหัสในแบบที่สองนั้นจะเป็นวิธีการถอดรหัสด้วยวิธีการหาค่าต่ำสุดดัดแปลง(Modified Min-Sum Algorithm)โดย MarjanKarkootiและคณะ [5] ได้ทำการออกแบบโครงสร้าง VNU ที่มีการเพิ่มในส่วนของสเกลลิ่งเฟคเตอร์ดังแสดงให้เห็นดังรูปที่2.12



รูปที่2.12บล็อกไดอะแกรมหน่วยประมวลผล VNU ด้วยวิธีการหาค่าต่ำสุดดัดแปลง

บทที่ 3

การออกแบบการถอดรหัสแอลดีพีซี

ในบทนี้จะกล่าวถึงหลักวิธีการออกแบบตัวถอดรหัสแอลดีพีซีในการออกแบบนั้นจะมีพื้นฐานการออกแบบอยู่หลายรูปแบบแล้วแต่ปัจจัยหลักของผู้ออกแบบในส่วนการออกแบบพื้นฐานของเรานั้นจะมีปัจจัยหลักคือ ความเร็วในการถอดรหัสและพลังงานที่ใช้ โดยการใช้FPGAจำนวนเกตต่ำๆ หรือใช้เวลาในการทำงานโดยรวมน้อย จะส่งผลทำให้ระบบมีการบริโภคพลังงานต่ำลงไปด้วย โดยการออกแบบของเรานั้นพยายามให้ระบบมีการบริโภคพลังงานที่ต่ำที่สุดและขนาดของฮาร์ดแวร์ที่มีขนาดเล็กที่สุด แต่การออกแบบนั้นต้องคำนึงถึงอัตราข้อมูลและประสิทธิภาพที่ได้โดยผลลัพธ์ที่ได้ต้องเป็นไปตามความต้องการของระบบหรือมาตราฐานในส่วนนี้เราได้นำเสนอเทคนิคการออกแบบเพื่อเพิ่มความเร็วในการถอดรหัสให้เสร็จเร็วยิ่งขึ้น และยังคงจำกัดจำนวนหน่วยประมวลผลไม่ให้มีมากจนเกินไป โดยผลลัพธ์ของการออกแบบนี้จะทำให้หน่วยประมวลผลของ CNU และ VNU ทำงานซ้อนกันในช่วงขณะหนึ่ง

ส่วนสำคัญของการถอดรหัสแอลดีพีซีคือ เมตริกซ์พาริตี้เช็ค H โดยประสิทธิภาพการถอดรหัสขึ้นอยู่กับเมตริกซ์นี้ ดังนั้นการจะทำให้ฮาร์ดแวร์มีความซับซ้อนน้อยลงหรือเพิ่มความเร็วในการทำงานมากยิ่งขึ้นจะต้องนำเอาเมตริกซ์พาริตี้เช็คไปปรับปรุง แต่เนื่องจากการถอดรหัสแอลดีพีซีของเรานั้นจะนำไปใช้บนการสื่อสารไร้สายในพื้นฐานของ IEEE 802.11nทำให้ไม่สามารถแก้ไขเมตริกซ์ได้เพราะได้มีการกำหนดเมตริกซ์พื้นฐานสำหรับการเข้ารหัสไว้แล้ว ถึงอย่างไรก็ตาม ยังมีการนำเสนอวิธีการปรับปรุงเมตริกซ์โดยไม่ส่งผลต่อการเข้ารหัสแอลดีพีซีและประสิทธิในการถอดรหัส หนึ่งในวิธีนั้นก็คือวิธีการสลับลำเมตริกซ์พาริตี้เช็ค

3.1 การสลับลำดับเมตริกซ์พาริตี้เช็ค(Reordering Parity Check Matrix)

จากพื้นฐานเมตริกซ์พาริตี้เช็ค Hของ IEEE 802.11n ที่มีขนาดของคำรหัส 648 bits อัตรารหัส 0.5 ตามรูปที่ 3.1จะประกอบไปด้วยบล็อกช่องว่าง บล็อกเลขศูนย์และบล็อกเลขจำนวนเต็มที่ไม่ใช่ศูนย์ ซึ่งจะมีทั้งหมด 12 แถว 24 หลัก ในบล็อกช่องว่าง (-) จะสามารถขยายองค์ประกอบได้ว่ามีขนาดของเมตริกซ์ศูนย์เท่ากับ $s*s$ ในองค์ประกอบของบล็อกศูนย์จะเป็นรูปแบบของเมตริกซ์ที่ถูกเลื่อนวนขนาด $s*s$ ส่วนองค์ประกอบของบล็อกเลขจำนวนเต็มที่ไม่ใช่ศูนย์จะเป็นเมตริกซ์ที่ถูกเลื่อนวนโดยถูกระบุตำแหน่งเริ่มเลื่อนบิตด้วยเลขจำนวนเต็มที่แสดงอยู่มีขนาด $s*s$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	-	-	-	0	0	-	-	0	-	-	0	1	0	-	-	-	-	-	-	-	-	-	-
2	22	0	-	-	17	-	0	0	12	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-
3	6	-	0	-	10	-	-	-	24	-	0	-	-	-	0	0	-	-	-	-	-	-	-	-
4	2	-	-	0	20	-	-	-	25	0	-	-	-	-	-	0	0	-	-	-	-	-	-	-
5	23	-	-	-	3	-	-	-	0	-	9	11	-	-	-	-	0	0	-	-	-	-	-	-
6	24	-	23	1	17	-	3	-	10	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-
7	25	-	-	-	8	-	-	-	7	18	-	-	0	-	-	-	-	-	0	0	-	-	-	-
8	13	24	-	-	0	-	8	-	6	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-
9	7	20	-	16	22	10	-	-	23	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-
10	11	-	-	-	19	-	-	-	13	-	3	17	-	-	-	-	-	-	-	-	-	0	0	-
11	25	-	8	-	23	18	-	14	9	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
12	3	-	-	-	16	-	-	2	25	5	-	-	1	-	-	-	-	-	-	-	-	-	-	0

รูปที่3.1เมตริกซ์พาริตี้เช็คขนาด 648 bits อัตรารหัส 1/2 ตามมาตราฐาน IEEE 802.11n

ในการประมวลผลของการถอดรหัส LDPC การทำงานของบิตโหนดของหลักแรกจะต้องรอการทำงานของเช็คโหนดเสร็จเสียก่อน หรืออีกนัยหนึ่งก็คือ การทำงานของเช็คโหนดในแถวแรกจะต้องรอการทำงานของบิตโหนดเสร็จก่อน เพื่อเป็นการลดเวลาว่างในการทำงานระหว่างเช็คโหนดกับบิตโหนด จะสามารถปรับปรุงได้โดยสลับลำดับแถวและหลักของเมตริกซ์ H [4] ซึ่งการสลับลำดับแถวและหลักจะไม่ส่งผลกระทบให้ประสิทธิภาพในการถอดรหัสลดลง คำรหัสที่ถูกสลับลำดับแล้วจะ ไม่ทำให้ตำแหน่งของแถวเกิดการเปลี่ยนแปลงแต่จะเปลี่ยนเฉพาะตำแหน่งของหลักเท่านั้นที่ถูกสลับตำแหน่งซึ่งอยู่ในระดับบิต

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

(ก)

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

(ข)

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

(ค)

รูปที่3.2การสลับลำดับเมตริกซ์พาริตี้เช็ค

(ก) เมตริกซ์พื้นฐาน

(ข) สลับลำดับของแถว

(ค) สลับลำดับทั้งแถวและหลัก

ตัวอย่างการสลับลำดับเมตริกซ์พาริตี้เช็คจะแสดงให้ดังรูปที่ 3.2ในรูป (ก) จะเป็นตัวอย่างของเมตริกซ์พื้นฐาน รูป (ข) แถวของเมตริกซ์ได้ถูกสลับลำดับที่แถว 3 กับ 6รูป(ค) หลังจากสลับลับดับแถวจึงนำมาสลับลำดับของหลักโดยที่นี้จะเปลี่ยนตำแหน่งของหลักทั้งหมด

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	Old	7	2	15	16	20	21	17	18	9	8	5	12	1	14	3	11	13	10	4	6	19	22	23	24
1	5	-	-	-	-	-	-	0	0	0	-	3	11	23	-	-	9	-	-	-	-	-	-	-	-
2	2	0	0	0	-	-	-	-	-	12	0	17	-	22	0	-	-	-	-	-	-	-	-	-	-
3	3	-	-	0	0	-	-	-	-	24	-	10	-	6	-	0	0	-	-	-	-	-	-	-	-
4	8	8	24	-	-	0	0	-	-	6	-	0	-	13	-	-	-	-	-	-	-	-	-	-	-
5	4	-	-	-	0	-	-	0	-	25	-	20	-	2	-	-	-	-	0	0	-	-	-	-	-
6	6	3	-	-	-	-	-	-	0	10	-	17	-	24	-	23	-	-	-	1	-	0	-	-	-
7	7	-	-	-	-	0	-	-	-	7	-	8	-	25	-	-	-	0	18		-	0	-	-	-
8	9	-	20	-	-	-	0	-	-	23	-	22	-	7	-	-	-	-	-	16	10	-	0	-	-
9	1	-	-	-	-	-	-	-	-	0	-	0	0	0	0	-	-	1	-	-	0	-	-	-	-
10	10	-	-	-	-	-	-	-	-	13	-	19	17	11	-	-	3	-	-	-		-	0	0	-
11	11	-	-	-	-	-	-	-	-	9	14	23	-	25	-	8	-	-	-	-	18	-	-	0	0
12	12	-	-	-	-	-	-	-	-	25	2	16	-	3	-	-	-	1	5	-	-	-	-	-	0

รูปที่3.3เมตริกซ์พาริตี้เช็คH ที่ถูกสลับลำดับแล้ว

การสลับลำดับเมตริกซ์พาริตี้เช็ค H จะมีความเป็นได้ในการสลับออกมาได้หลายรูปแบบ แต่มีกฎสำคัญอยู่ว่าต้องย้ายทั้งแถวหรือทั้งหลักในเวลาเดียวกัน ในการนำเอาไปใช้งานบนฮาร์ดแวร์เราได้เพิ่มบัฟเฟอร์หลังจากรับคำรหัสมาจากช่องสัญญาณแล้ว จากนั้นจะทำการสลับตำแหน่งให้เข้ารูปแบบเมตริกซ์ที่ได้เราได้ออกแบบไว้ แล้วส่งไปยังหน่วยประมวลผลเพื่อทำการคำนวณตามอัลกอริทึม ทำนองเดียวกันเมื่อทำการถอดรหัสเสร็จแล้ว เราจำเป็นต้องสลับตำแหน่งของคำรหัสที่ได้ให้มีตำแหน่งเหมือนเดิม เพื่อให้ได้คำรหัสที่ถูกต้อง

จากการสลับลำดับเมตริกซ์ดังในรูปที่ 3.3จะเห็นว่าแถวที่ 9-12 ค่าผลลัพธ์ของการคำนวณจะไม่มีผลต่อหลักที่ 1-8 ซึ่งการทำงานที่ได้จากการสลับลำดับนี้จะส่งผลให้เกิดการทำงานที่ซ้อนกันหรือขนานกันได้ และการทำงานในหลักที่ 17-24 กับแถวที่ 1-4 ก็จะมีลักษณะเหมือนกัน โดยทั้งหมดนี้จะประมวลผลที่ 4 CNUs ต่อไซเคิลและ 8 VNUs ต่อไซเคิล และเมื่อมีความต้องการใช้จำนวนประมวลที่น้อยกว่านี้เราสามารถออกแบบให้ใช้หน่วยประมวลผลได้ที่ 3 CNUs และ 6 VNUs หรือสามารถใช้หน่วยประมวลที่น้อยที่สุดคือ 2 CNUs และ 4 VNUs โดยการสลับลำดับเมตริกซ์ทั้งหมดนี้ต้องเป็นไปกฎตามที่ระบุไว้ข้างต้น ดังแสดงให้เห็นดังรูปที่ 3.4 ซึ่งเป็นเมตริกซ์ที่เราออกแบบไว้โดยใช้หน่วยประมวลน้อยที่สุด

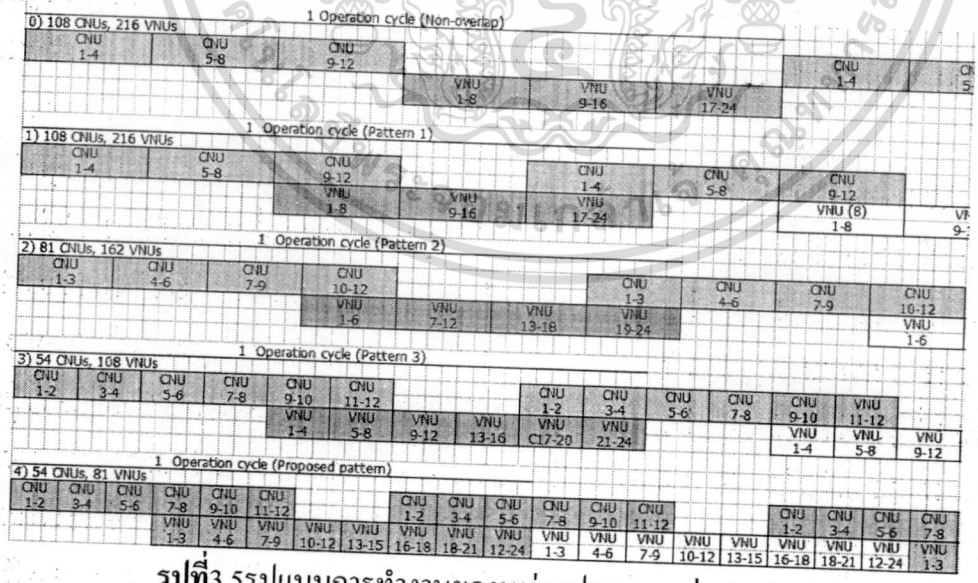
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	Old	15	17	7	16	20	21	2	18	14	12	5	8	1	9	11	3	19	10	4	6	13	22	23	24
1	8	-	-	8	-	0	0	24	-	-	-	0	-	13	6	-	-	-	-	-	-	-	-	-	-
2	2	0	-	0	-	-	-	0	-	0	-	17	0	22	12	-	-	-	-	-	-	-	-	-	-
3	3	0	-	-	0	-	-	-	-	-	-	10	-	6	24	0	0	-	-	-	-	-	-	-	-
4	5	-	0	-	-	-	-	-	0	-	11	3	-	23	0	9	-	-	-		-	-	-	-	-
5	4	-	0	-	0	-	-	-	-	-	-	20	-	2	25	-	-	-	0	0	-	-	-	-	-
6	6	-	-	3	-	-	-	-	0	-	-	17	-	24	10	-	23	0	-	1	-	-	-	-	-
7	7	-	-	-	-	0	-	-	-	-	-	8	-	25	7	-	-	0	18		-	0	-	-	-
8	9	-	-	-	-	-	0	20	-	-	-	22	-	7	23	-	-	-	-	16	10	-	0	-	-
9	1	-	-	-	-	-	-	-	-	0	0	0	-	0	0	-	-	-	-	-	0	1	-	-	-
10	10	-	-	-	-	-	-	-	-	-	17	19	-	11	13	3	-	-	-	-		-	0	0	-
11	11	-	-	-	-	-	-	-	-	-	-	23	14	25	9	-	8	-	-	-	18	-	-	0	0
12	12	-	-	-	-	-	-	-	-	-	-	16	2	3	25	-	-	-	5	-	-	1	-	-	0

รูปที่3.4เมตริกซ์พาริตี้เช็คH ที่ถูกสลับลำดับแล้วในรูปแบบที่เราสำเสนอ

3.2 การทำงานซ้อนกันของ VNU และ CNU(Overlapped Opreations of VNUs and CNUs)

จากส่วนที่ผ่านมา เราสามารถเปลี่ยนรูปแบบของเมตริกซ์พาริตี้เช็คโดยการสลับลำดับเมตริกซ์ ซึ่งสามารถจัดรูปให้เห็นถึงการทำงานที่ไม่มีการซ้อนกันและการทำงานซ้อนกันของแต่ละรูปแบบดังรูปที่ 3.5 โดยรูปแบบที่ 1-3 จะเป็นรูปร่างตามการออกแบบของบทความ [4] และการออกแบบของเราจะอยู่ในรูปแบบการทำงานที่ 4

ในรูปแบบการประมวลผลที่ 3 จะเห็นว่าหน่วยประมวลผล 8 VNUs จะมีช่วงเวลาที่ไม่สามารถทำงานซ้อนกันได้ หรืออีกนัยหนึ่งคือ จะมีช่วงเวลาว่างที่ไม่ทำงาน 8/24 หรือ 33% หน่วยประมวลผลของ CNU ในทำนองเดียวกัน CNU ก็จะมีลักษณะเหมือนกันคือจะมีช่วงว่างจำนวน 8 ช่วง ซึ่งอยู่ในช่วงเวลาการทำงานของ VNU ในการเปรียบเทียบเพื่อให้เห็นถึงความแตกต่าง สำหรับรูปแบบการประมวลแบบที่ 4 หน่วยประมวลผล VNU ทั้งหมดจะทำงานติดกันโดยไม่มีช่วงว่าง ขณะที่ CNU มีช่วงว่างอยู่ที่ 6 ช่วง ในช่วงเวลาการทำงานของ VNU ซึ่งจำนวนไซเคิลที่ซ้ำกันจะเป็นจำนวนของการซ้อนกัน โดยสามารถเลือกใช้ได้ตามความเหมาะสมดังที่เราออกแบบไว้ ดังเช่นกรณีของรูปแบบในกลุ่มการประมวลผลของรูปแบบที่ 3 โดยนำรูปแบบที่ 4 ซึ่งเราออกแบบไว้มาเทียบเคียงจะเห็นได้ว่าผลลัพธ์โดยรวมดีกว่า ดังนั้นผลลัพธ์ที่ได้ในการออกแบบการสลับลำดับเมตริกซ์นั้นจะมีหลายรูปแบบ ดังเช่นเมตริกซ์ของเราในรูปที่ 3.4



รูปที่3.5รูปแบบการทำงานของหน่วยประมวลแต่หละแบบ

จากการพิจารณาในรูปที่ 3.5 จะชี้ให้เห็นถึงการทำงานวนซ้ำ (iteration) ในหนึ่งรอบของการประมวลผลของยูนิตแถว (CNU) และยูนิตหลัก (VNU) โดยในการทำงานหนึ่งไซเคิลจะเป็นการคำนวณหนึ่งแถวหรือหนึ่งหลัก (สามารถทำในรูปแบบขนานได้) การวิเคราะห์ทางคณิตศาสตร์; ในรูปแบบทำงานที่ไม่ซ้อนกัน (pattern 0) จะถูกทำการประมวลผลทีหละ 4 แถวในหนึ่งไซเคิล (ต้องการ 108 CNUs) และจะทำการประมวลผลทีหละ 8 หลักในหนึ่งไซเคิล (ต้องการ 216 VNUs) โดยทั้งหมดจะทำงานที่ 20 รอบดังนั้นจะใช้ไซเคิลเท่ากับ 120 ไซเคิล ในรูปแบบที่ 1 (pattern 1) จะทำการประมวลผล 4 แถว 8 หลัก ต่อหนึ่งไซเคิล ใช้ 108 CNUs และ 216 VNUs เมื่อทำงานซ้อนกันไซเคิลจะใช้เท่ากับ [(4*Ite*)+1] และทำงานไม่ซ้อนกัน [6*Ite*] *Ite*จะบ่งบอกถึงจำนวนการวนซ้ำ ในทำนองเดียวกันของรูปแบบที่ 2, 3 และ 4 จะใช้วิธีคำนวณคล้ายกันข้างต้น ซึ่งผลลัพธ์จะแสดงอยู่ในตารางที่ 3.1โดยเราจะเน้นการเปรียบเทียบกันระหว่างรูปแบบที่ 3 กับรูปแบบที่ 4 ที่เราได้ออกแบบไว้ ซึ่งเราจะใช้ ยูนิตของ VNU เพียง 75% จากรูปแบบที่ 3 และยังคงสามารถทำงานซ้อนกันได้เหมือนเดิม จึงทำให้เห็นว่าการใช้หน่วยประมวลผลของ CNU ยังคงเดิมแต่หน่วยประมวลผลของ VNU น้อยลง

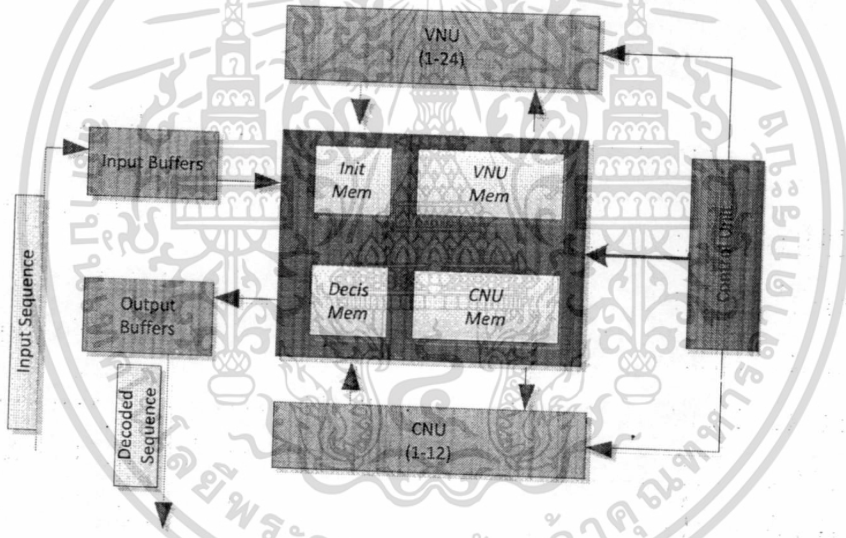
Pattern	CNU (Units)	VNU (Units)	Overlapped (Cycles)	Cycles Utilization	
				CNU	VNU
Typ. Cons	324	648	-	(324/972)	(648/972)
0	108	216	-	(3/6)	(3/6)
1, [5]	108	216	81; [(4*Ite*)+1]	(4/5)	(4/5)
2, [5]	81	162	121; [(6*Ite*)+1]	(5/7)	(5/7)
3, [5]	54	108	162; [(8*Ite*)+2]	(8/10)	(8/10)
4, [proposed]	54	81	163; [(8*Ite*)+3]	(9/11)	(11/11)

ตารางที่ 3.1เปรียบเทียบการใช้เวลา (cycle)ของการทำงานแต่ละรูปแบบ

บทที่ 4

การออกแบบโครงสร้างภายในเอฟพีจีเอ

ในส่วนนี้ เราจะอธิบายถึงการออกแบบโครงสร้างภายในของการถอดรหัสแอลดีพีซี สำหรับการนำไปใช้งานบนแอฟพีจีเอการออกแบบสำหรับแอฟพีจีเอของเรานั้นจะเป็นการออกแบบในระดับ RTLหรือในระดับเกต เพื่อส่งผลให้การสังเคราะห์ของโปรแกรมมีการใช้งานที่เกตต่ำๆ โดยจะเริ่มออกแบบเป็นโมดูลย่อยแล้วจึงนำมารวมกันเป็นระบบ ขนาดของคำรหัสที่เรานำมาออกแบบมีขนาด 648 บิตอัตรารหัส ½ มีขนาดบิตที่ 6 บิต ส่วนประกอบหลักในการออกแบบเริ่มแรกคือหน่วยประมวล CNU และ VNU เมื่อทำการออกแบบจนได้โครงสร้างที่เหมาะสมแล้ว จากนั้นทำการออกแบบหน่วยความจำเพื่อใช้เก็บข้อมูลจากหน่วยประมวลผล ส่วนถัดมาคือหน่วยอินพุตบัฟเฟอร์และเอาต์พุตบัฟเฟอร์ เมื่อออกแบบโครงสร้างดังกล่าวแล้วจึงนำโครงสร้างทั้งหมดมารวมกันและทำการออกแบบหน่วยควบคุมเพื่อให้ระบบทำงานสอดคล้องกัน

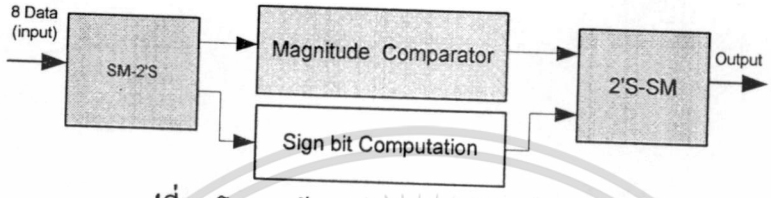


รูปที่4.1 โครงสร้างระบบการถอดรหัสแอลดีพีซี

ในรูปที่ 4.1จะแสดงถึงโครงสร้างการถอดรหัสของแอลดีพีซีโดยสามารถใช้ได้ทั้งวิธีการผลรวมต่ำสุด (MS)และวิธีการผลรวมต่ำสุดดัดแปลง (MMS)โดยจะมีโครงสร้างหลักอยู่ 4 โครงสร้างคือ หน่วยประมวลผลบิตโหนด (VNU) หน่วยประมวลผลเช็คโหนด (CNU) หน่วยความจำ (memory unit) และ หน่วยควบคุม (Control unit) โดยหน่วยความจำจะทำการเก็บข้อมูลที่คำนวณได้จากหน่วยประมวลผล บิตโหนดกับเช็คโหนด ซึ่งขนาดของหน่วยความจำจะขึ้นอยู่กับขนาดของเมตริกซ์พาริตี้เช็คที่นำมาใช้ ส่วนอินพุตบัฟเฟอร์ (input buffer unit) กับเอาต์พุตบัฟเฟอร์ (output buffer unit) จะทำหน้าที่สลับตำแหน่ง (permutation) ของคำรหัสที่เข้ามาและสลับตำแหน่งคืน (de-permutation) หลังจากทำการถอดรหัสเสร็จสิ้น

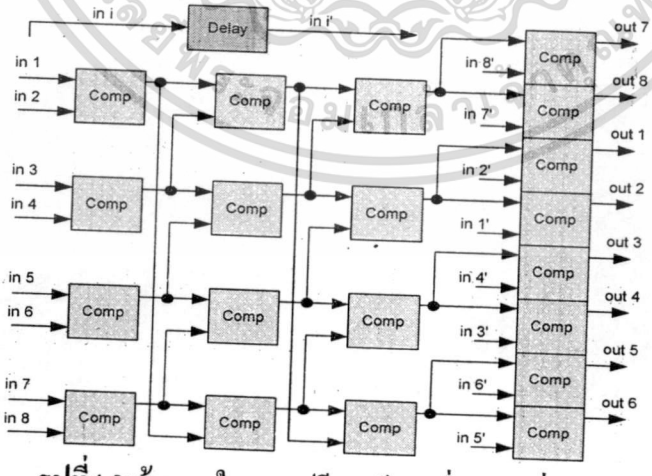
4.1 หน่วยประมวลผลเช็คโหนด (CNUs Unit)

หน่วย CNU จะทำหน้าที่ในการประมวลผลของแต่หละแถว ซึ่งเป็นไปตามสมการที่ 2.53โดยจะทำการหาค่าต่ำสุดจากทั้งหมด 24 ตำแหน่ง ในแถวนั้นๆ แต่อย่างไรก็ตามเนื่องจากในแถวจะประกอบไปด้วยค่าศูนย์กับค่าหนึ่ง ซึ่งในตำแหน่งที่มีค่าศูนย์จะไม่นำมาคิด จึงมีเพียง 7-8 ตำแหน่งต่อแถวเท่านั้นที่นำมาพิจารณา



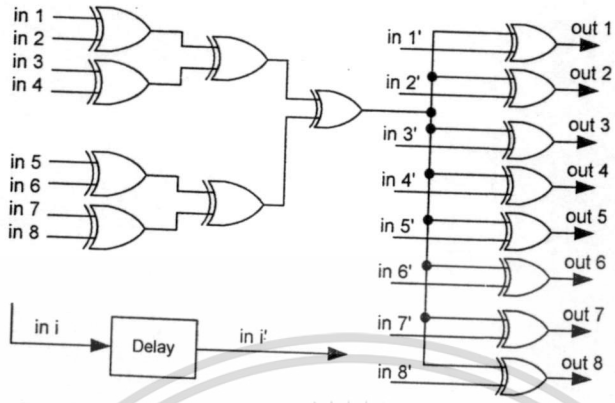
รูปที่4.2โครงสร้างของหน่วยประมวลผล CNU

ในแต่หละตำแหน่งจะมีขนาดของบิตอยู่ที่ 6 บิต ซึ่งรวมบิตเครื่องหมายด้วย โดยอินพุตที่เข้ามาจะเป็นระบบเลข 2's complement จากการแสดงในรูปที่ 4.2เมื่ออินพุตเข้ามาจะทำการเปลี่ยน 2's complement ให้แยกออกเป็นสองส่วนคือ ขนาดบิต (magnitude bits) กับ บิตเครื่องหมาย (sign bit) โดยขนาดบิตจะถูกนำไปเปรียบเทียบกันเพื่อหาค่าต่ำสุด รูปแบบของเส้นทางในการเปรียบเทียบสามารถดูได้จากรูปที่ 8 ซึ่งจำนวนของการเปรียบเทียบจะมีทั้งหมด 20 บล็อก แต่หละบล็อกจะเปรียบกัน 2 อินพุต และออก 1 เอาต์พุต สำหรับแถวที่มีค่าไม่ใช่ศูนย์อยู่ 7 ตำแหน่ง ในจำนวน 8 อินพุตจะมีหนึ่งอินพุตที่ถูกตั้งค่าให้เป็นค่าสูงสุดเพื่อละเลยในการเปรียบเทียบ



รูปที่4.3เส้นทางในการเปรียบเทียบเพื่อหาค่าต่ำสุด

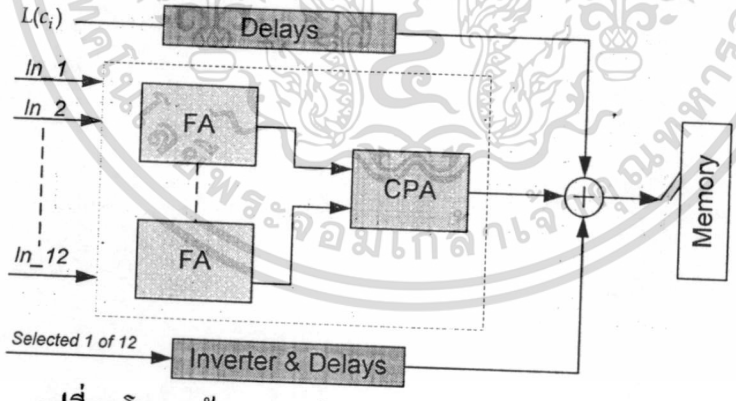
การคำนวณของบิตเครื่องหมายสามารถคำนวณได้โดยง่าย ซึ่งแสดงให้เห็นดังรูปที่ 4.4จะเป็นวงจรที่ประกอบไปด้วยเกต XORเท่านั้น



รูปที่4.4โครงสร้างเพื่อการคำนวณหาบิตเครื่องหมาย (sign bit)

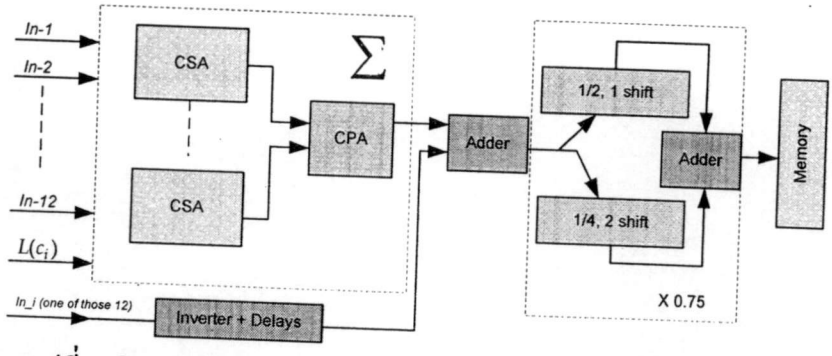
4.2 หน่วยประมวลผลบิตโหนด (VNUs Unit)

ในส่วนของโครงสร้างหน่วยประมวลผลบิตโหนดเราได้ออกแบบให้รองรับวิธีการถอดรหัสเป็นสองวิธีเพื่อที่จะสามารถใช้ได้ทั้งวิธีการผลรวมต่ำสุด (MS) และวิธีการผลรวมต่ำสุดดัดแปลง (MMS) และยังได้พัฒนาโครงสร้างของ VNUที่ใช้วิธีการ MMS เพื่อเป็นอีกทางเลือกหนึ่งในการใช้งาน



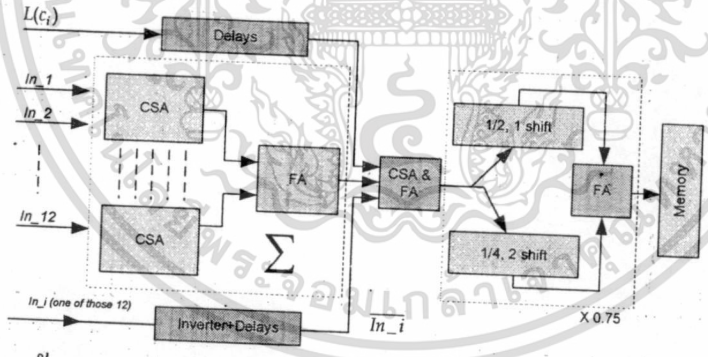
รูปที่4.5โครงสร้างของหน่วยประมวลผล VNUs min-sum

การถอดรหัสแอลดีพีซีด้วยวิธีการผลรวมต่ำสุดจะเป็นการคำนวณตามสมการที่ 2.54 โดยสามารถแสดงให้เห็นถึงโครงสร้างภายในได้ดังรูปที่ 4.5 ซึ่งการคำนวณจะเป็นการหาผลรวมของทั้ง 12อินพุต ที่อยู่ในแต่หละหลักมาคำนวณ เริ่มจากการนำเอาจำนวนอินพุตทั้งหมดนำมาบวกกันก่อน และในขณะเดียวกันอินพุตทั้งหมดจะถูกแยกไปผ่านอินเวอร์เตอร์เพื่อกลับเครื่องหมายจากนั้นนำไปบวกผลรวมที่ได้ก่อนหน้านี้และรวมกับค่า Lciซึ่งจะทำให้เราจะได้ผลลัพธ์ในตำแหน่งนั้นๆ



รูปที่4.6โครงสร้างของหน่วยประมวลผล VNUs modified min-sum

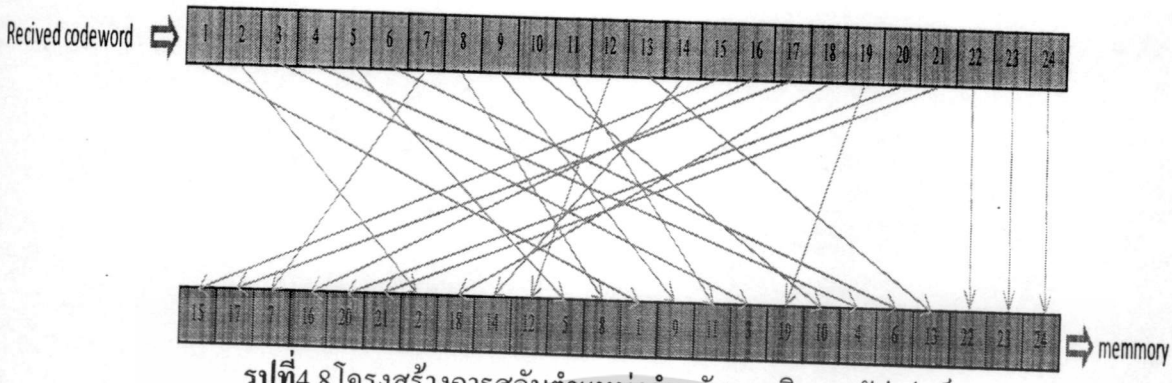
ในส่วนของวิธีการผลรวมต่ำสุดดัดแปลง (MMS) หน่วย VNU จะเป็นการนำสมการที่ 2.57 มาใช้งาน โดยสามารถแสดงให้เห็นถึงโครงสร้างภายในได้ดังรูปที่ 4.6 ซึ่งการหาผลรวมของทั้ง 12 ตำแหน่งในหลักมาคำนวณ โดยขั้นตอนแรกจะนำเอาจำนวนอินพุตทั้งหมดนำมาบวกกันทั้งหมดรวมทั้งค่า Lci อย่างไรก็ตามเมื่อเราพิจารณาหาค่าที่ตำแหน่งนั้นๆ จะนำค่าที่ตำแหน่งนั้นมาลบกับผลรวมในขั้นตอนแรกโดยการผ่านแอดเดอร์ซึ่งจะได้ผลลัพธ์ในตำแหน่งนั้นๆ วงจรในการรวมค่าทั้ง 12 อินพุตเราจะใช้ CSA 10 ยูนิต ตามด้วยยูนิตCPA 1 ยูนิต โดยผลลัพธ์ที่นำไปใช้งานจริงจะถูกคำนวณผ่าน scaling factor อีกหนึ่งขั้นตอน โดยมีค่าสเกลเท่ากับ 0.75 ด้วยวิธีการเลื่อนบิตและบวกกันโดยตรง



รูปที่4.7โครงสร้างของหน่วยประมวลผล VNUs modified min-sumที่ถูกปรับปรุงแล้ว

ความเร็วในการทำงานสูงสุดของหน่วย VNU ตามรูปที่ 4.6 จะมีความเร็วที่สูงกว่าหน่วย CNU ในทางทฤษฎีการทำงานของ CNU และ VNU จะทำงานในรูปแบบขนาน อย่างไรก็ตามสำหรับรูปแบบที่เราสร้างเป็นพิเศษเราจะต้องใช้หนึ่ง CNU และหนึ่ง VNU ในการทำงานแต่หละรอบ เราสามารถปรับความเร็วให้เหมาะสมกันโดยพิจารณาจากรูปแบบที่ 4 (pattern 4) ในรูปที่ 3.5 จะเห็นว่าการทำงานของ CNU ในบล็อกของแถวที่ 7 กับ 8 จะต้องทำงานพร้อมกันทั้งสองแถวในหนึ่ง CNU และหน่วย VNU จะทำงานที่หลักที่ 1-3 ขนานกันไปด้วย ดังนั้นการทำงานของ VNU ควรจะทำงานเร็วเป็น 1.5 เท่า ของหน่วย CNU เมื่อพิจารณาได้ดังนี้หน่วย VNU จึงถูกออกแบบใหม่เพื่อเพิ่มความเร็วให้ได้ตามความต้องดังแสดงในรูปที่ 4.7

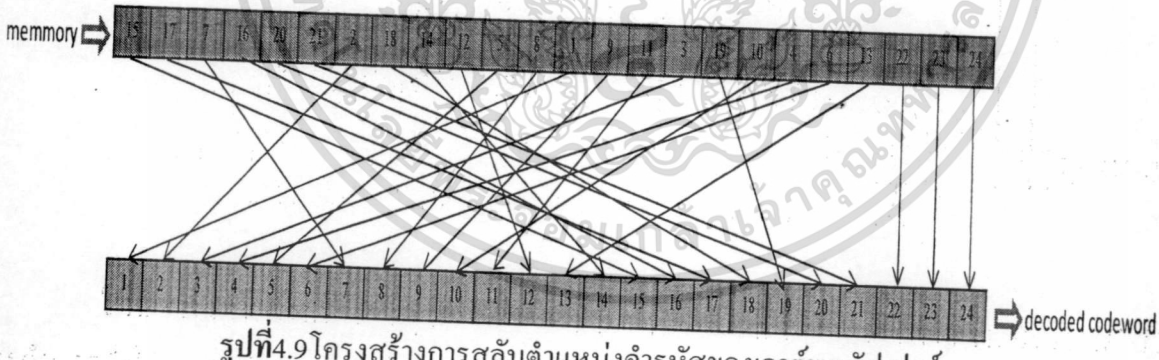
4.3 หน่วยอินพุตบัฟเฟอร์(Input Buffer Unit)



รูปที่4.8โครงสร้างการสลับตำแหน่งคำรหัสของอินพุทบัฟเฟอร์

ในส่วนของโครงสร้างอินพุตบัฟเฟอร์จะทำหน้าสลับตำแหน่งของคำรหัสที่รับเข้ามาเพื่อใช้ในการคำนวณให้เป็นตามดังเมตริกซ์ที่เราได้ออกแบบไว้ข้างต้น ดังแสดงให้ในรูปที่ 4.8เมื่อเรารับคำรหัสจากช่องสัญญาณแล้ว จากนั้นจะนำคำรหัสที่ได้มาเข้ากระบวนการเปลี่ยนตำแหน่ง เมื่อเปลี่ยนตำแหน่งเรียบร้อยแล้วจะนำคำรหัสที่ได้ไปเก็บลงในหน่วยความจำเพื่อทำการประมวลผลต่อไป

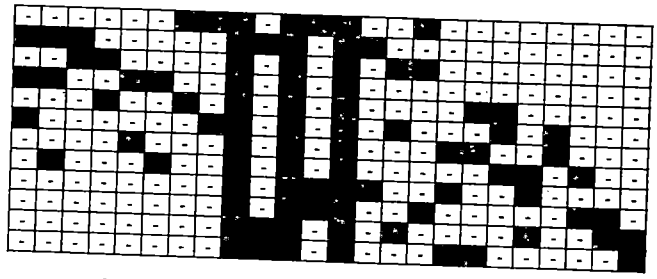
4.4 หน่วยเอาต์พุตบัฟเฟอร์(Output Buffer Unit)



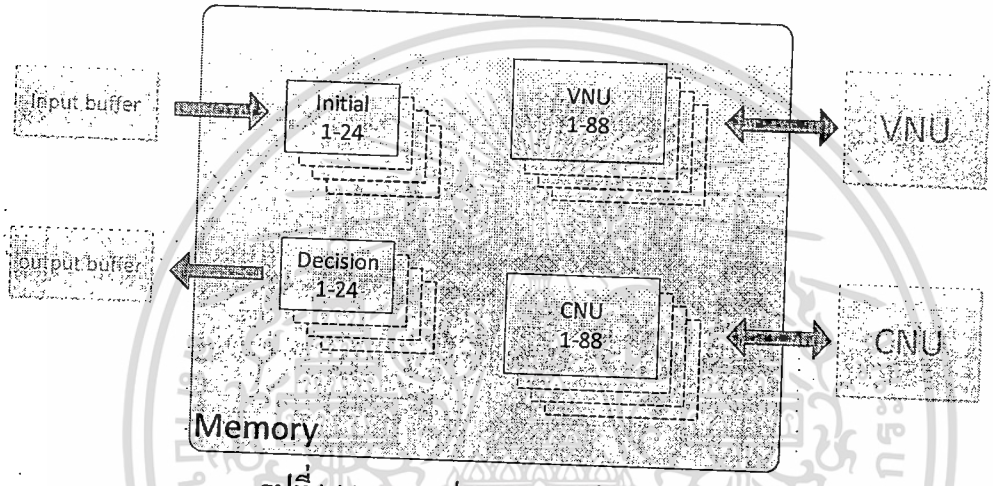
รูปที่4.9โครงสร้างการสลับตำแหน่งคำรหัสของเอาต์พุตบัฟเฟอร์

ในส่วนของโครงสร้างเอาต์พุตบัฟเฟอร์จะทำหน้าสลับตำแหน่งกลับตำแหน่งเดิมจากที่เราได้สลับในส่วนแรก เพื่อให้ได้คำรหัสที่ได้ถูกต้อง ดังแสดงให้ในรูปที่ 4.9เมื่อเราประมวลผลจนเป็นที่เรียบร้อยแล้วคำรหัสที่ได้นั้นยังมีตำแหน่งที่ไม่ถูกต้องเพราะว่าเราได้สลับตำแหน่งไปก่อนหน้านี้ ฉะนั้นเราจะนำคำรหัสจากหน่วยความจำมาสลับตำแหน่งคืนดังเดิมให้ได้ตามเมตริกซ์พื้นฐาน เมื่อเปลี่ยนตำแหน่งเรียบร้อยแล้วเราก็จะได้คำรหัสที่ถูกต้องตามมาตราฐาน

4.5 หน่วยความจำ (Memmory Unit)



รูปที่4.10 เมตริกซ์พาริตี้เช็คที่ใช้ในการถอดรหัส



รูปที่4.11แสดงรูปแบบของหน่วยความจำ

จากรูปที่ 4.11จะแสดงถึงจำนวนของหน่วยความจำเพื่อใช้เก็บข้อมูล โดยโครงสร้างภายในจะเป็น Look up table (LUT) เนื่องจากการประมวลผลที่เราได้ออกแบบไว้เป็นแบบกึ่งขนาน ทำให้การออกแบบโครงสร้างต้องแยกออกเป็นบล็อคตามเมตริกซ์พาริตี้เช็คดังรูปที่ 4.10 เพื่อให้อ่านและเขียนลงหน่วยความจำสามารถทำงานในรูปแบบขนานได้เช่นเดียวกัน โดยแต่หละบล็อกจะมีตำแหน่งเก็บข้อมูลอยู่ 27 ตำแหน่ง และมีความกว้างของบิตที่ 6 บิต ในส่วนของหน่วยความจำหลังจากผ่านอินพุตบัฟเฟอร์คือหน่วยความจำเก็บค่าเริ่มต้น (Lci) จะมีจำนวน 24 บล็อก ถัดมาเป็นหน่วยความจำเพื่อเก็บค่าจากการประมวลผล VNU และ CNU โดยแต่หละหน่วยจะมีจำนวนหน่วยเก็บข้อมูลอยู่ 88 บล็อก เมื่อทำการประมวลผลเสร็จเรียบร้อย หน่วยความจำสุดท้ายคือการเก็บข้อมูลจากการประมวลผลเพื่อตัดสินใจมีจำนวน 24 บล็อก

4.6 หน่วยควบคุม (Control Unit)

หน่วยควบคุมจะเป็นหน่วยที่ควบคุมการทำงานของระบบทั้งหมด เมื่อรับคำรหัสผ่านช่องสัญญาณมาแล้ว ข้อมูลที่เก็บในอินพุตบัฟเฟอร์จะถูกเรียกออกมาเพื่อทำการสลับตำแหน่งให้ถูกต้องตามเมตริกซ์พาริตี้เช็คที่ได้ออกแบบไว้ เมื่อทำการสลับตำแหน่งเรียบร้อยแล้วค่าที่ได้จะนำไปเป็นเก็บลงในหน่วยความจำค่าเริ่มต้น (Lci) ซึ่งอยู่ในขั้นตอนของการเซ็ทค่าตั้งต้น (Initial) จากนั้นหน่วยควบคุมจะเริ่มสั่งให้ระบบประมวลผลทำงานโดยการโหลดค่าLciลงในหน่วยความจำ VNUs เพื่อนำค่าเรานี้ไปทำตามสมการที่ 2.53 ซึ่งเป็นการประมวลของ CNUs เมื่อทำการประมวลเช็คโหนดเสร็จแล้วจะนำค่าไปเก็บลงในหน่วยความจำ CNUs จากนั้นหน่วยควบคุมจะสั่งทำการประมวลผลที่บิตโหนดตามสมการ 2.54 (ถ้าเป็นวิธีการผลรวมต่ำสุดดัดแปลงจะทำตามสมการ 2.57)เมื่อประมวลผล VNUs เรียบร้อยจะทำการเก็บข้อมูลลงหน่วยความจำ VNUs เช่นเดียวกัน ซึ่งการกระทำเช่นนี้จะเป็นการประมวลผลเสร็จหนึ่งรอบ เมื่อเริ่มทำการประมวลผลในรอบที่ถัดไป หน่วยควบคุมไม่จำเป็นต้องสั่งให้อ่านข้อมูลจากค่า Lciมาเก็บลงในหน่วยความจำ VNUs อีกเพราะเราได้ข้อมูลจากการประมวลในรอบที่ผ่านมาเรียบร้อยแล้ว จากนั้นหน่วยควบคุมจะสั่งให้ประมวลผลตามขั้นตอนการถอดรหัสจนกระทั่งถึงรอบสูงสุดที่ตั้งไว้ จากนั้นหน่วยควบคุมจะสั่งให้ทำการตัดสินใจแบบระเอียดสมการ 2.55 และการตัดสินใจแบบหยาบสมการ 2.56 และส่งผ่านข้อมูลไปยังหน่วยเอาต์พุตบัฟเฟอร์เพื่อทำการสลับตำแหน่งคืนตำแหน่งเดิม เมื่อทำตามขั้นตอนที่ผ่านมาทั้งหมดถือว่าการถอดรหัสนั้นเสร็จสิ้นเป็นที่เรียบร้อย

จากระบบการถอดรหัสที่เราได้ออกแบบเพื่อนำไปใช้บน FPGAจะเห็นได้ว่าเราได้ละเลยการคำนวณของขั้นตอนพาริตี้เช็ค $H\cdot(\hat{c}_1,\hat{c}_2,\ldots,\hat{c}_n)^T=0$ เนื่องจากการทำขั้นตอนนี้จะมีความซับซ้อนเพิ่มมากยิ่งขึ้นเพราะต้องนำผลลัพท์ทั้งสองค่ามาคูณกันมีผลทำให้ขนาดของฮาร์ดแวร์ใหญ่มากขึ้นจากเดิมเป็นจำนวนมาก และอีกประการหนึ่งคือเมื่อทำการประมวลผลจนครบรอบสูงสุดที่กำหนดไว้การถอดรหัสถือว่าเสร็จสิ้นจึงไม่จำเป็นต้องทำขั้นตอนนี้ เพียงแต่ว่าเมื่อมีการถอดรหัสที่ถูกต้องเสร็จก่อนระบบยังคงทำงานต่อไปทำให้เวลาการถอดรหัสช้าลงแต่ระบบยังคงมีความเร็วเป็นไปตามมาตราที่เรานำมาใช้

บทที่ 5

การทดลองระบบถอดรหัสแอลดีพีซี

สำหรับผลการทดลองนั้นจะถูกแบ่งออกเป็น 2 ส่วนหลักด้วยกันคือส่วนของผลการทดลองของขั้นตอนการถอดรหัสแอลดีพีซีกับผลการทดลองจากการสังเคราะห์ระบบถอดรหัสแอลดีพีซีบนเอฟพีจีเอ ซึ่งสามารถอธิบายได้ดังนี้

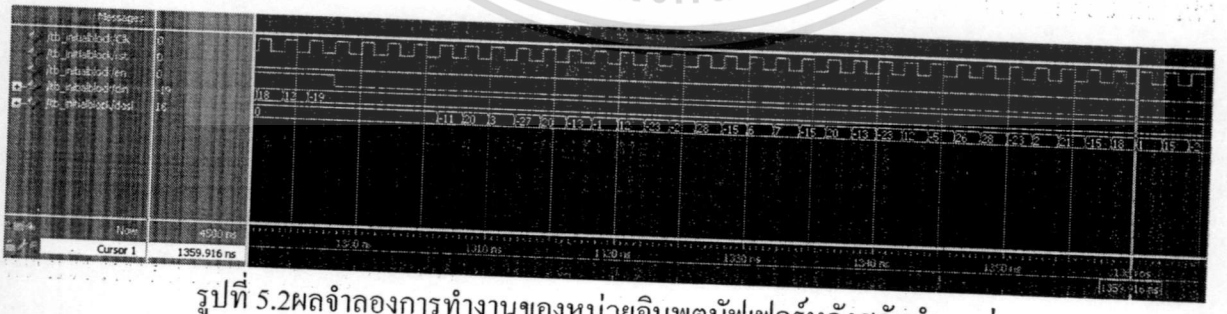
5.1 ผลการทดลองของขั้นตอนการถอดรหัสแอลดีพีซี

สำหรับผลการทดลองของขั้นตอนการถอดรหัสแอลดีพีซีนั้นจะแบ่งผลการทดลองออกเป็นห้าส่วนด้วยกัน อันจะประกอบไปด้วย

5.1.1 สัญญาณการทดลองของขั้นตอนอินพุตบัฟเฟอร์

ในผลการทดลองของอินพุตบัฟเฟอร์โดยการทำงานหลักๆ ของหน่วยนี้คือการสลับลำดับคำรหัสที่รับเข้ามาจากช่องสัญญาณเพื่อให้สามารถประมวลซ้อนทับกันตามที่ออกแบบไว้

รูปที่ 5.1ผลจำลองการทำงานของหน่วยอินพุตบัฟเฟอร์ก่อนสลับตำแหน่ง

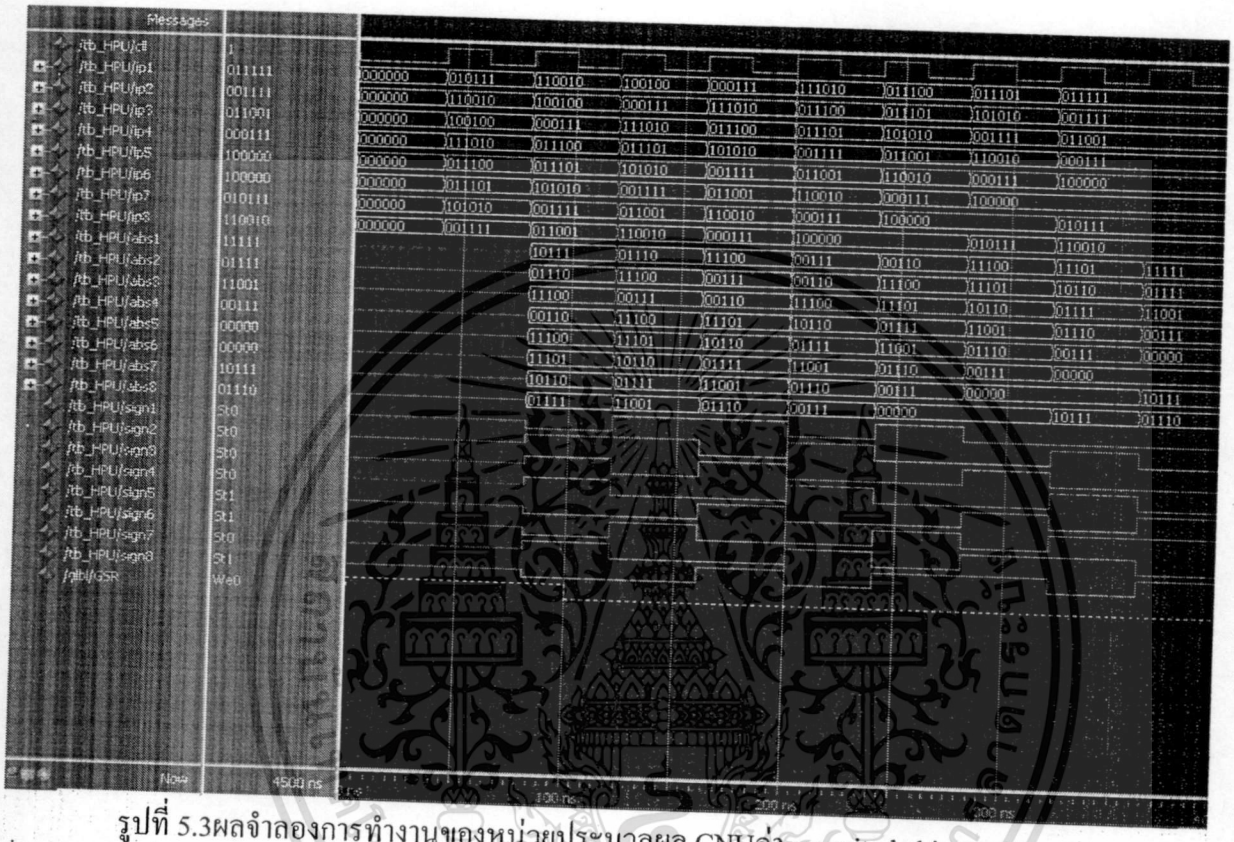


รูปที่ 5.2ผลจำลองการทำงานของหน่วยอินพุตบัฟเฟอร์หลังสลับตำแหน่ง

จากรูปที่ 5.1 แสดงให้เห็นถึงคำรหัสที่รับเข้ามาจากช่องสัญญาณ โดยตามเมตริกซ์พาริตี้เช็คที่ออกแบบไว้ตามรูปที่ 3.3 แสดงให้เห็นว่าคำรหัสของบล็อกลำดับที่ 7 จะต้องย้ายมาอยู่ตำแหน่งบล็อกที่ 1 ซึ่งตำแหน่งของบล็อกที่ 7 ในการจำลองการทำงานจะเริ่มอยู่ในคาบเวลาที่ 326ns จนถึง 380ns ซึ่งจะมีจำนวน

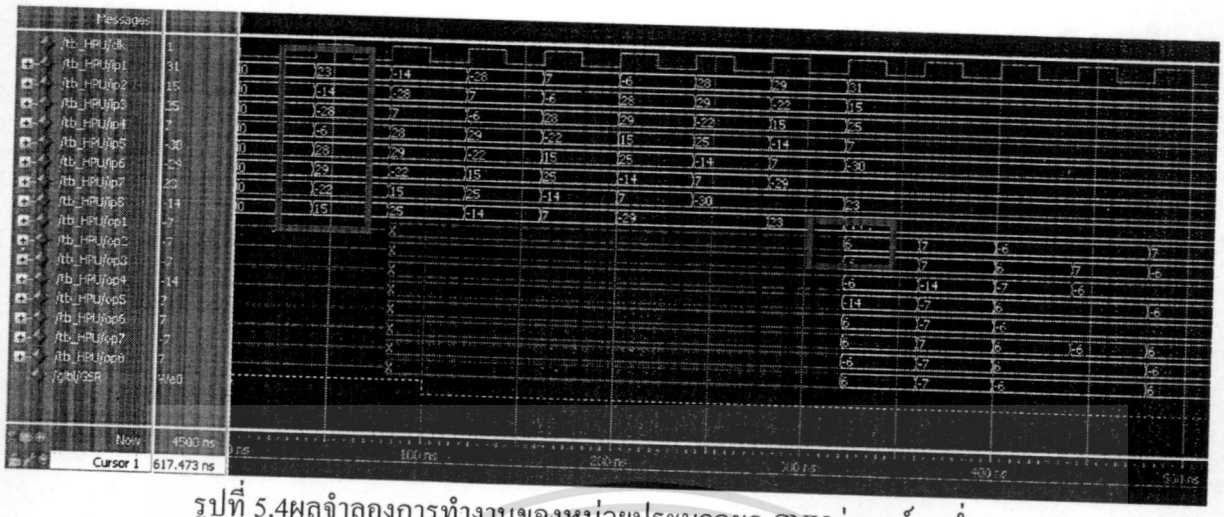
27 ค่าตามขนาดของบล็อก เมื่อทำการสลับตำแหน่งแล้วเอาต์พุตที่ออกมาจากหน่วยความจำอันดับแรกคือบล็อคที่ 7 จะแสดงให้เห็นดังรูปที่ 5.2

5.1.2 สัญญาณการทดลองของขั้นตอนการประมวลผล CNUs



รูปที่ 5.3ผลจำลองการทำงานของหน่วยประมวลผล CNUค่า magnitude bitและ Sign bit

เมื่อคำรหัสถูกสลับตำแหน่งเรียบร้อยแล้วซึ่งอยู่ในขั้นตอนของตั้งค่าเริ่มต้น หลังจากนั้นจะเข้าสู่ขั้นตอนการประมวลผลของหน่วย CNUs ดังรูปที่ 5.3 เนื่องจากการประมวลผลจะแบบ 2'complement ดังนั้นเราจึงต้องจำลองการทำงานแบบเลขฐานสอง ตามรูปจะเห็นว่าในแต่หละแถวจะมีอินพุตสูงสุดอยู่ที่ 8 อินพุต (ip1 - ip8) โดยการประมวลผลจะทำงานตามสัญญาณนาฬิกา (clk) สำหรับตัวอย่างการคำนวณจะแสดงให้เห็นได้ในกรอบที่เราได้ทำไว้ เมื่อเราคำนวณที่ ip1 ซึ่งมีค่า 010111 (23)โดยเราจะไม่นำค่าที่ตำแหน่งนั้นมาคำนวณ จากนั้นจะนำค่าจากอินพุตที่เหลือทั้งหมดมาแยกบิตสัญลักษณ์กับขนาดบิต โดยขนาดบิตที่ตำแหน่ง abs1 มีค่าเท่ากับ 10111 และบิตสัญลักษณ์เท่ากับ 0 เมื่อทำการแยกค่าทั้งสองแยกออกจากกันแล้วจึงนำไปหาค่าต่ำสุดจากทั้งหมด 8 อินพุตโดยจะแสดงให้เห็นดังภาพที่ 5.4 เมื่อเราไม่คิดที่ตำแหน่งที่ต้องการค่าต่ำสุดจากทั้งหมดที่เหลือคือ 6 และเมื่อได้ค่าต่ำสุดแล้วจึงมาหาค่าบิตสัญลักษณ์ จะเห็นได้ว่าบิตสัญลักษณ์ที่เหลือมีจำนวนเป็นคู่จึงทำให้เอาต์พุตที่ 1 มีค่าเป็นบวก ดังนั้นผลลัพธ์ของการคำนวณอินพุตที่ 1 (op1) จะมีค่าเท่ากับ 6



รูปที่ 5.4ผลจำลองการทำงานของหน่วยประมวลผล CNUค่าเอาต์พุตต่ำสุด

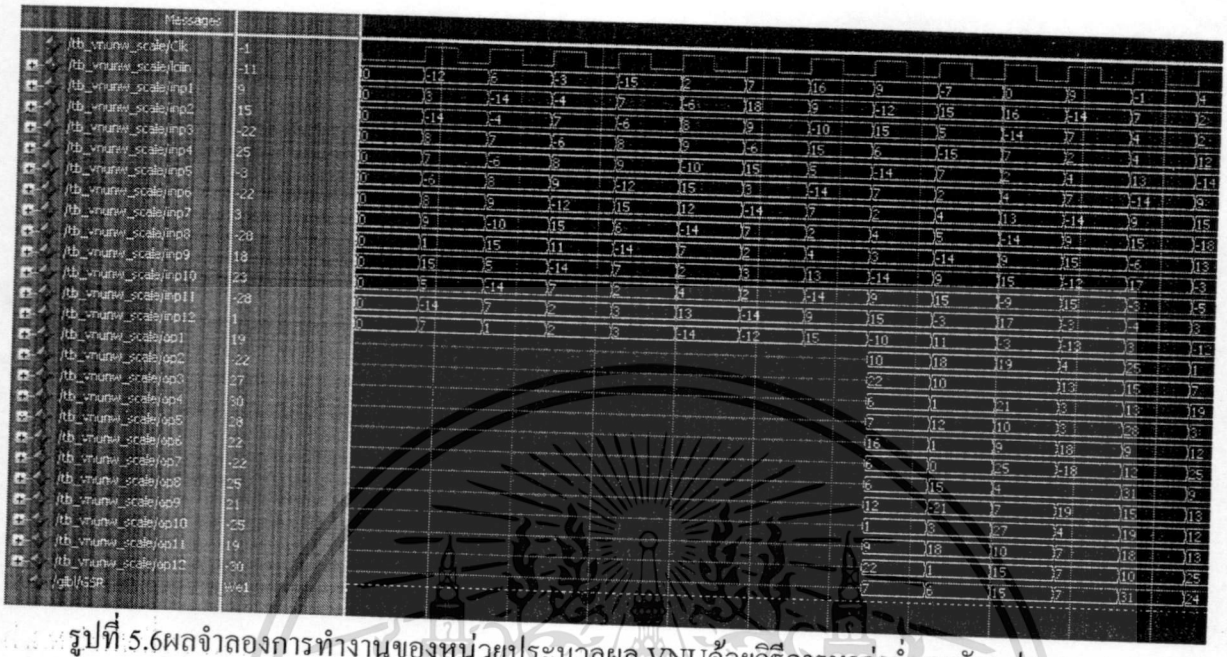
5.1.3 สัญญาณการทดลองของขั้นตอนการประมวลผล VNU

การประมวลของ VNU จะกระทำต่อหลังจากการประมวลผล CNU โดยจะแสดงให้เห็นทั้งวิธีการหาค่าต่ำสุดและวิธีการหาค่าต่ำสุดดัดแปลงดังนี้

รูปที่ 5.5ผลจำลองการทำงานของหน่วยประมวลผล VNUด้วยวิธีการหาค่าต่ำสุด (MS)

จากรูปที่ 5.5 แสดงถึงการจำลองการประมวลผลของหน่วย VNU ด้วยวิธีการหาค่าต่ำสุด (MS) การประมวลผลในขั้นตอนนี้จะเป็นการหาผลรวมของหลักแต่หละหลักโดยเมื่อคำนวณที่ตำแหน่งใดจะไม่เอาค่าที่ตำแหน่งนั้นมาคำนวณ สามารถแสดงการคำนวณตัวอย่างได้โดยเมื่อเราคิดที่ตำแหน่งของอินพุตที่ 1 (ip 1) ซึ่งมาค่าเท่ากับ 3 จะประมวลผลโดยนำค่าทั้งหมดมารวมกันพร้อมทั้งค่า Lciด้วย มีค่าการรวมดังนี้

5+3-14+7-6+9+7+1+2-14+9-5+12 = 16 จากนั้นนำมาลบตำแหน่งที่หาคือ 16-3 = 13 จึงได้เป็นลัพธ์ในตำแหน่งที่หนึ่งนั้นเอง

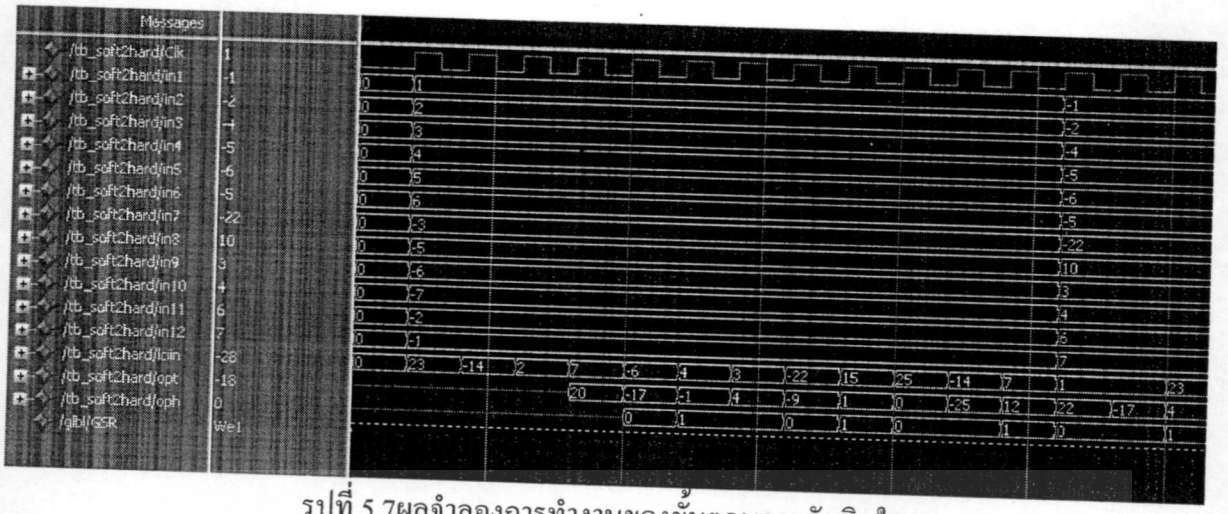


รูปที่ 5.6ผลจำลองการทำงานของหน่วยประมวลผล VNUด้วยวิธีการหาค่าต่ำสุดดัดแปลง(MMS)

จากรูปที่ 5.6 จะเป็นผลจำลองการทำงานของหน่วยประมวลผล VNUด้วยวิธีการหาค่าต่ำสุดดัดแปลง(MMS)จะเห็นได้ว่ามีอินพุตสูงสุดของแต่หละหลักอยู่ที่ 12 อินพุตและมีอินพุต Lciอีกหนึ่งอินพุต โดยหลักการของหน่วยประมวลนี้จะเป็นการหาค่าของผลรวมทั้งหมดและนำมาคูณกับสเกลลิ่งเฟคเตอร์ และสำหรับตัวอย่างการคำนวณจะแสดงให้เห็นได้ดังกรอบที่เราได้ล้อมไว้ ซึ่งจะเป็นการคำนวณที่อินพุตที่ 1 (inp1) และจะนำผลรวมทั้งหมดมารวมกัน โดยไม่รวมค่าที่ตำแหน่งที่กำลังคำนวณ มีค่าดังนี้ 17-3 = 14 และจากนั้นจะนำค่าที่ได้ไปคูณกับสเกลลิ่งเฟคเตอร์ = 0.75 เมื่อเราคำนวณตามปกติแล้วจะได้ค่าเท่ากับ 10.5 แต่เนื่องจากในฮาร์ดแวร์เราออกแบบนั้นสามารถเก็บได้เฉพาะจำนวนเต็มเท่านั้น ดังนั้นผลลัพธ์สุดท้ายจึงมีค่าเท่ากับ 10 จะเห็นได้ว่าจากค่าที่เราคำนวณได้จากฮาร์ดแวร์นั้นมีความคลาดเคลื่อนจึงอาจส่งผลต่อประสิทธิในการถอดรหัส

5.1.4 สัญญาณการทดลองของขั้นตอนการตัดสินใจ (Decision)

ในขั้นตอนของขบวนการสินใจจะกระทำเมื่อการประมวลผลถึงรอบสูงสุดที่กำหนดไว้หน่วยประมวลผลเพื่อตัดสินใจจะอ่านข้อมูลจากหน่วยความจำ VNUs มาประมวลผลโดยคำนวณแบบละเอียดก่อนและนำมาคำนวณแบบหยาบ



รูปที่ 5.7ผลจำลองการทำงานของขั้นตอนการตัดสินใจ

จากรูปที่ 5.7 สามารถแสดงการคำนวณตามตัวอย่างได้ดังนี้ โดยการคำนวณแบบละเอียดจะนำค่าทั้งหมดในหลักมารวมกัน รวมทั้งตำแหน่งที่คำนวณด้วยซึ่งแตกต่างจากการคำนวณในขั้นตอนของ VNU ซึ่งไม่นำตำแหน่งที่คำนวณมาคิด โดยมีค่าทั้งหมดดังนี้1+2+3+4+5+6-3-5-6-7-2-1+23 = 20 เมื่อได้ค่าการติดสินใจแบบละเอียดแล้วจึงนำมาตัดสินใจแบบหยาบ จะเห็นได้ว่าตามสมการที่ 2.56เมื่อใดที L(Qi) มีค่ามากกว่า 0 จะได้ผลลัพธ์เท่ากับ 0 ดังแสดงดังรูปด้านบนที่ ophเอาต์พุตที่ออกเริ่มแรกมีค่าเท่ากับ 0

5.1.5 สัญญาณการทดลองของขั้นตอนเอาต์พุตบัฟเฟอร์

รูปที่ 5.8ผลจำลองการทำงานของหน่วยเอาต์พุตบัฟเฟอร์

เมื่อการประมวลผลทั้งหมดเสร็จสิ้นและผ่านขั้นตอนของการติดสินใจเรียบร้อยแล้ว ข้อมูลจะถูกส่งผ่านไปยังเอาต์พุตบัฟเฟอร์เพื่อทำการสลับตำแหน่งคืนดังเดิมดังเช่นในรูปที่ 5.8 จะเห็นได้ว่าตำแหน่งที่เริ่มเก็บลงหน่วยความจำจะเริ่มที่ตำแหน่ง 324ซึ่งเป็นตำแหน่งของบล็อกที่ 1 ที่ถูกย้ายไปก่อนหน้านี้จะอยู่ที่ตำแหน่งของบล็อกที่ 13 โดยแต่หละบล็อคมีขนาดเท่ากับ 27 แต่จุดเริ่มเก็บข้อมูลจะต้องถูกลบออกอีกหนึ่งบล็อคเท่ากับ 12*27 = 324 เมื่อทำการสลับตำแหน่งคืนดังเดิมทั้งหมดจึงสามารถนำข้อมูลไปใช้ได้อย่างถูกต้อง

5.2 ผลการทดลองจากการสังเคราะห์ของเอฟพีจีเอ

จากโครงสร้างพื้นฐานที่เราออกแบบไว้ในส่วนที่ผ่านมา หน่วยประมวลผลทั้งหมดของระบบถอดรหัสแอลดีพีซีจะถูกสร้างโดยใช้ภาษา Verilog และจำลองการทำงานด้วย ModelSim XE การออกแบบทั้งหมดจะถูกสังเคราะห์ด้วยโปรแกรม Xilinx 10.1 โดยใช้อุปกรณ์ Xilinx XC2VP30-7

Xilinx Utilization Summary		
	MS	MMS
Number of Slices	10261/13696	11131/13696
Number of Slice Flip Flops	15982/27392	17216/27392
Number of 4 input LUTs	19376/27392	20080/27392
Maximum Frequency (MHz)	186.02	186.02

ตารางที่ 5.1เปรียบเทียบการใช้พื้นที่และความถี่ของฮาร์ดแวร์

จากตารางที่ 5.1 แสดงถึงจำนวนฮาร์ดแวร์ของระบบการถอดรหัสแอลดีพีซีที่ถูกสังเคราะห์ในโปรแกรม Xilinx ซึ่งใช้เมตริกซ์พาริตี้เช็คความรูปที่ 3.3 มีขนาดคำรหัส 649 บิตและอัตรารหัสที่ 0.5 เหตุผลที่เราไม่นำมาเมตริกซ์พาริตี้เช็คตามรูปที่ 3.4มาสังเคราะห์จะชี้แจงในบทสุดท้ายโดยวิธีการหาค่าต่ำสุดจะมีการใช้พื้นที่น้อยกว่าวิธีการหาค่าต่ำสุดดัดแปลงเนื่องจากการประมวลผลในขั้นตอนของ VNU ไม่ต้องทำการสเกลลิ่งเฟคเตอร์ และในส่วนความเร็วสูงสุดที่ทำงานได้จะมีความเร็วเท่ากันคือ 186.02 MHz

บทที่ 6

สรุปผลการวิจัย และข้อเสนอแนะ

6.1 สรุปผลการวิจัย

ในรายงานฉบับนี้เราได้นำเสนอถึงวิธีการสลับลำดับเมตริกซ์พาริตี้เช็ค เพื่อใช้บนมาตรฐานของ IEEE 802.11n โดยเหมาะสมที่จะนำไปใช้กับการถอดรหัสแอลดีพีซี ด้วยวิธีการหาค่าต่ำสุดMS หรือวิธีการหาค่าต่ำสุดดัดแปลงMMS ซึ่งเราได้แสดงให้เห็นว่าหน่วยประมวลของเราที่ได้ออกแบบไว้นั้นใช้พื้นที่น้อยกว่าของ [4] แต่อย่างไรก็ตามในการประมวลผลเมื่อไม่ได้ทำงานในรูปแบบที่ซ้อนทับกันจะใช้ไซเคิลมากกว่า 10% และการนำเสนอของเรานั้นยังได้แสดงให้เห็นถึงการทำงานของหน่วยประมวล VNU ว่าทำงานอยู่ตลอดเวลาในขณะที่หน่วยประมวลผล CNU มีช่วงว่างอยู่ 6 ไซเคิลของ VNU ดังนั้นเราจึงสามารถนำมาพัฒนาต่อได้เพื่อให้ทำงานอยู่ในสภาวะที่เหมาะสมดังแสดงให้ในรายงานนี้

6.2 ปัญหาที่เกิดขึ้นและข้อเสนอแนะ

จากการออกแบบเมตริกซ์พาริตี้เช็คที่ผ่านมาเมื่อเรานำไปใช้สังเคราะห์บนเอฟพีจีเอจะไม่สามารถทำตามรูปแบบที่ได้ออกแบบไว้ทั้งหมด เนื่องจากการประมวลผลเป็นแบบกึ่งขนานทำให้การประมวลผลจะต้องทำขนานกันในบางส่วน ดังนั้นการอ่านข้อมูลจากหน่วยความจำ จำเป็นที่จะต้องอ่านและเขียนขนานกันด้วยในบางเวลา ซึ่งผลกระทบจากส่วนนี้ทำให้ไม่สามารถรวมหน่วยความจำเป็นบล็อกเดียวกันได้ จึงต้องแยกบล็อกหน่วยความออกเป็นส่วนๆ เพื่อทำให้อ่านและเขียนข้อมูลขนานกันได้ เมื่อเป็นเช่นนี้โปรแกรมที่ใช้ในการสังเคราะห์ระบบการถอดรหัสจึงสังเคราะห์ออกมาเป็น look up table แทนที่จะสังเคราะห์ออกมาเป็น RAM เพราะขนาดของหน่วยความจำแต่หละบล็อกมีขนาดเล็ก และทำให้ขนาดของฮาร์ดแวร์มีขนาดใหญ่ขึ้นจากเดิมอีกด้วย และอีกประการหนึ่งคือโครงสร้างการถอดรหัสแอลดีพีซีตามเมตริกซ์พาริตี้เช็คที่ได้ออกแบบไว้เพื่อลดขนาดของจำนวนประมวลผล เมื่อนำมาใช้งานจริงจะไม่สามารถทำให้เล็กลงตามที่ออกแบบไว้ดังเช่นเมตริกซ์ในรูปแบบที่ 4 ที่มีขนาดเล็กที่สุด เพราะเนื่องมาจากหน่วยความจำนั้นเอง โดยบล็อกของหน่วยความจำที่เราสามารถนำใช้ได้มีขนาดเล็กที่สุดคือขนาดเท่า27 ตำแหน่งการแก้ไขปัญหาอาจจะต้องพัฒนาการควบคุมหน่วยจำต่อไป เนื่องจากปัญหาหลักมาจากหน่วยจำ

อย่างไรก็ตามการทำให้ระบบการประมวลผลสามารถซ้อนทับกันในช่วงขณะหนึ่ง จะส่งผลให้การถอดรหัสเสร็จเร็วยิ่งขึ้นและประหยัดพลังงานตามลงไปด้วย รวมประสิทธิภาพการถอดรหัสยังอยู่ในมาตราฐานที่เรานำมาใช้

เอกสารอ้างอิง

1. Xilinx Inc., *"Virtex-II proDeverlopment System: Data Sheet"*. 2009.
2. J D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity check codes," *Electron. Lett.*.vol. 32, p. 1645, 1996.
3. Draft STANDARD for Information Technology – Telecommu-nications and information exchange between systems - Local and metropolitan area networks - Specific requirements-, IEEE P802.11n./D3.00, Sept. 2007.
4. Xin-Yu Shih, Cheng-Zhou Zhan, Cheng-Hung Lin, and An-Yeu (Andy) Wu, "An 8.29 mm^2 52 mW Multi-Mode LDPC Decoder Design for Mobile WiMAX System in 0.13 μm CMOS Process," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 3, March 2008.
5. M. Karkooti, P. Radosavljevic, and J. R. Cavallaro, "Configurable LDPC Decoder Architectures for Regular and Irregular Codes," *J. Signal Processing Systems,* vol. 53, pp. 73-88, October 2008.
6. Hiroyuki SHIMAJIRI, Akifumi HAYASHI and Takeo YOSHIDA, "RTL design of LDPC decoder for IEEE802.11n WLAN," *ISCIT 2009. 9th International Symposium on Communications and Information Technology*, 2009, Sep. 2009, pp. 302-307.

ภาคผนวก