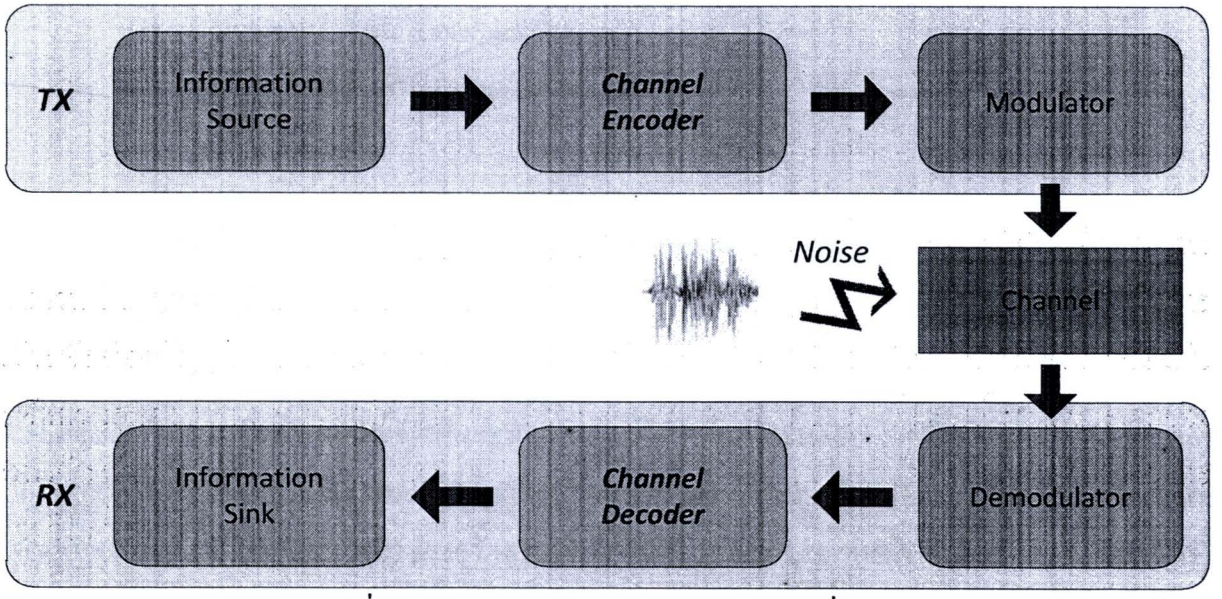
บทที่ 2
ทฤษฎีพื้นฐานและงานวิจัยที่เกี่ยวข้อง

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

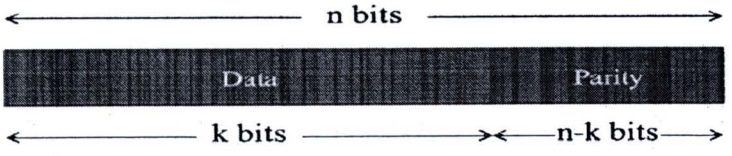
การสื่อสารในระบบต่างๆ สามารถเกิดความผิดพลาดได้ และวิธีลดความผิดพลาดที่เกิดนั้นสามารถทำได้หลายวิธี เช่น วิธีการเพิ่มกำลังส่งของเครื่องส่งในการส่งข้อมูลและวิธีการเข้ารหัส ซึ่งวิธีการเข้ารหัสเพื่อแก้ไขความผิดพลาดของข้อมูล (Error Control Coding) เป็นเทคนิคที่ได้นำไปใช้ สำหรับการรับส่งข้อมูลในระบบการสื่อสารดิจิตอล 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_iหรือไม่ 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

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

อาศัยความสัมพันธ์ตามสมการ จะได้ว่า

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

โดย$\mathbf{I}_k$ 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

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

ซึ่งเป็นเมตริกซ์ขนาดn-kความสัมพันธ์ในสมการข้างบนสามารถทำให้กระชับขึ้นได้เป็น

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

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

รหัสบล็อกเชิงเส้นมีคุณสมบัติที่น่าสนใจคือคุณสมบัติปิด(closure)นั้นคือถ้านำคำรหัสสองคำมา

รวมกันจะได้เป็นคำรหัสใหม่ที่เป็นสมาชิกของรหัสนั้นๆด้วยสามารถพิสูจน์คุณสมบัติข้อนี้ได้โดยง่ายสมมุติให้คำรหัสc_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$m_1 + m_2$จะได้เป็นข้อมูลชุดใหม่ซึ่งเมื่อนำไปคูณกับเมตริกซ์ตัวกำเนิด**G**ก็จะได้คำรหัสที่เป็นสมาชิกหนึ่งของรหัสด้วย

สำหรับส่วนต่อไปนี้จะอธิบายถึงการสร้างความสัมพันธ์ที่เป็นประโยชน์กับกระบวนการถอดรหัส นิยามเมตริกส์H ที่มีขนาด (n-k) x n ขึ้นได้ดังนี้

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

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

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

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

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

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

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

นำสัญญาณที่รับได้ในรูปของเวกเตอร์**r** มาถอดรหัสเพื่อหาค่าซินโดรมจะได้ผลดังนี้

$$\mathrm{s} = \mathrm{r}\mathrm{H}^{\mathrm{T}} \tag{1.20}$$

อาศัยสมการที่ 18 และ 20 จะได้

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

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

รูปแบบแตกต่างอยู่ถึง2^k รูปแบบที่จะให้ค่าผลลัพธ์ของการคำนวณซินโดรม 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)คือแทนที่จะต้องคำนวณหาระยะแฮมมิงระหว่างคำรหัสทุกคู่ในชุดรหัสcและเปรียบเทียบค่าระยะแฮมมิงที่คำนวณได้ทั้งหมดเพื่อหาค่าที่ต่ำที่สุด ซึ่งผลลัพธ์ที่ได้คือ ระยะแฮมมิงต่ำสุดของรหัส **c** แต่ถ้าใช้ทฤษฎีบทในการคำนวณสิ่งที่ต้องทำคือคำนวณค่าน้ำหนักของคำรหัสทั้งหมดที่ไม่ใช่ศูนย์ และเปรียบเทียบน้ำหนักแฮมมิงแต่ละค่าที่ได้เพื่อหาค่าที่ต่ำที่สุดโดยผลลัพธ์ที่ได้จะเป็นค่าระยะแฮมมิงต่ำที่สุดของรหัสcด้วย

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: ตำแหน่งของ1กับลูปปิด $(A_{1,1}),(A_{1,4}),(A_{5,4}),(A_{5,1})$

ไซเคิลที่ 2:ตำแหน่งของ1กับลูปปิด $(A_{2,5}),(A_{2,6}),(A_{3,6}),(A_{3,5})$

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$[\rho_2, \rho_3, \dots, \rho_n]$
2) ค่าความหนาแน่นของเลขหนึ่งของแต่ละหลักนิยามโดย$[\lambda_2, \lambda_3, \dots, \lambda_n]$
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 $(n-k) \times n$ 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

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

เขียนเมตริกซ์พาริตี้เช็ค**H** ในรูป

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

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

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

ดังนั้นเมตริกซ์บิตพาริตี้เช็คทั้งหมดหาได้จาก

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

โดยใช้คุณสมบัติการบวกกันแบบมอดูโล2

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

2.3.4 การถอดรหัสแอลดีพีซี

การเข้ารหัสมีผลทำให้ข้อมูลแต่ละบิตมีความสัมพันธ์กันผ่านทางโครงสร้างของเมตริกส์พาริตี้เช็ค**H** ซึ่งการถอดรหัสก็จะอาศัยความสัมพันธ์เหล่านี้มาช่วยในการถอดรหัสอัลกอริทึมสำหรับการถอดรหัสแอลดีพีซีนั้นวิธีการเริ่มแรกมีชื่อเรียกว่าsum-product algorithm (SPA) ซึ่งเป็นรูปแบบการถอดรหัสที่เรียกว่า soft iterative decodingแต่เนื่องจากวิธีการข้างต้นเมื่อนำไปใช้บนฮาร์ดแวร์จะมีความซับซ้อนสูงดังนั้นจึงได้มีการพัฒนารหัสให้มีความซับซ้อนน้อยลงได้มาเป็นอัลกอริทึมที่อยู่ในลอกโดเมนLog-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: $c_1+c_2+c_3+c_4=0$

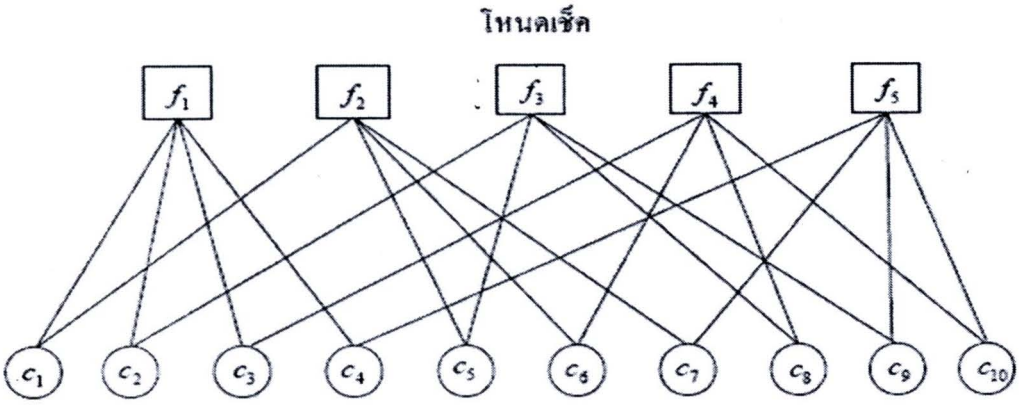
โหนดเช็ค ที่ 2: $c_1+c_5+c_6+c_7=0$

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.

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

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

ขั้นตอนที่ 3 การคำนวณหาค่าบิตโหนดในรอบที่สอง

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

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

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

ขั้นตอนที่ 4 การตัดสินอย่างละเอียด

หาความน่าจะเป็นของค่าบิตโหนดqทุกตัว

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

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

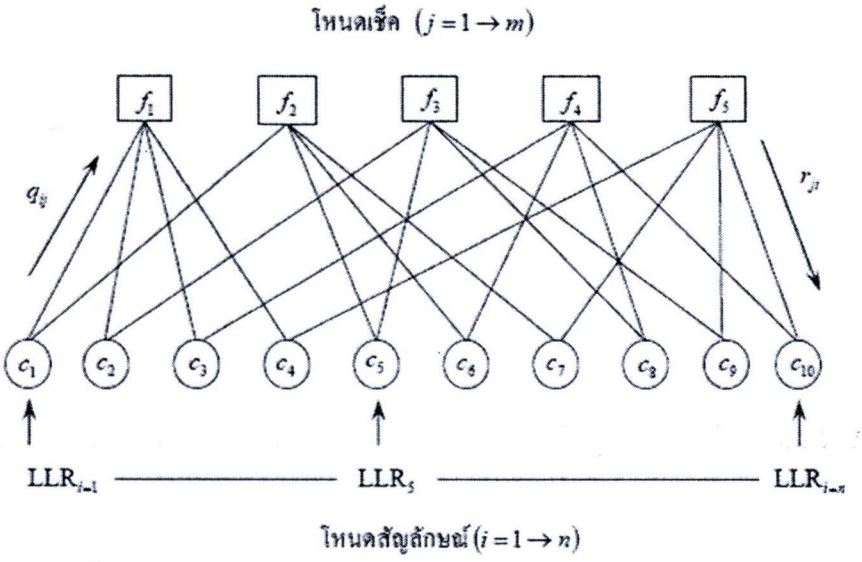
โดยเลือกค่า K_{ij}เพื่อให้ $Q_i(0) + Q_i(1) = 1$

ขั้นตอนที่ 5 การตัดสินใจอย่างหยาบ

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

ถ้า $\mathbf{cH}^{\mathrm{T}} = \mathbf{0}$ 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) ของตัวแปรสุ่มc_iตามสมการ โดยในแต่ละบิตโหนดสัญลักษณ์iจะส่งค่าความน่าจะเป็นของข่าวสารแบบซอฟต์ไปที่โหนดเช็คjผ่านตามเส้นความสัมพันธ์ที่เชื่อมถึงกันและจากนั้นที่โหนดเช็คj ก็จะทำการคำนวณค่าความน่าจะเป็นของข่าวสารที่ส่งมาจากบิตโหนดI แล้วจึงส่งผลที่ได้ของข่าวสารแบบซอฟต์ไปให้บิตโหนดi อีกครั้งเพื่อนำข้อมูลที่ได้ไปใช้ในการตัดสินใจว่าควรจะเป็น 0 หรือ 1

ขั้นตอนที่ 1 การคำนวณค่าเริ่มต้น

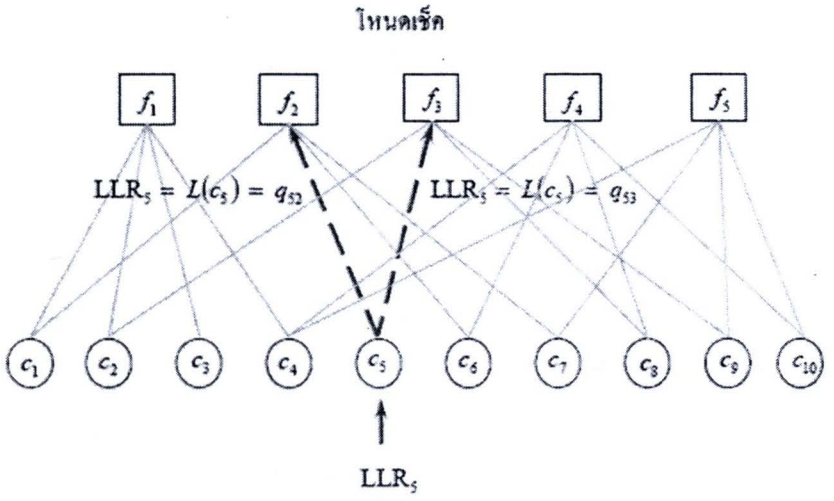
$$L\left(q_{ij}\right) = L\left(c_i\right) = 2y_i / \sigma^2 \tag{1.44}$$

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

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

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

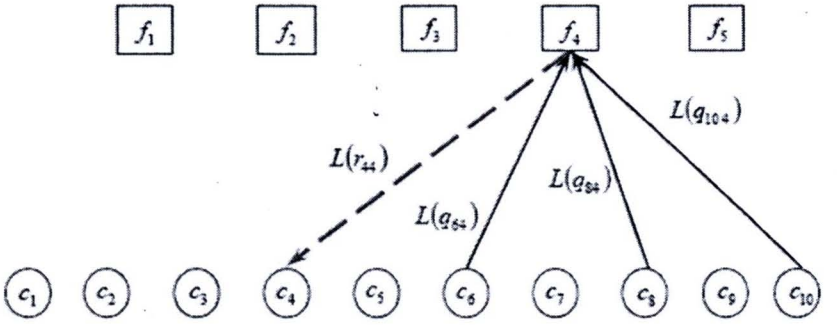
เมื่อ

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

นิยามให้

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

$Vj\setminus i$ โดยแทนการพิจารณาข่าวสารจากทุกบิตโหนดที่เชื่อมต่อกับเช็คโหนดj ยกเว้นบิตโหนดที่กำลังพิจารณารูปที่2.6แสดงแผนภาพการคำนวณค่า$L(r_{44})$

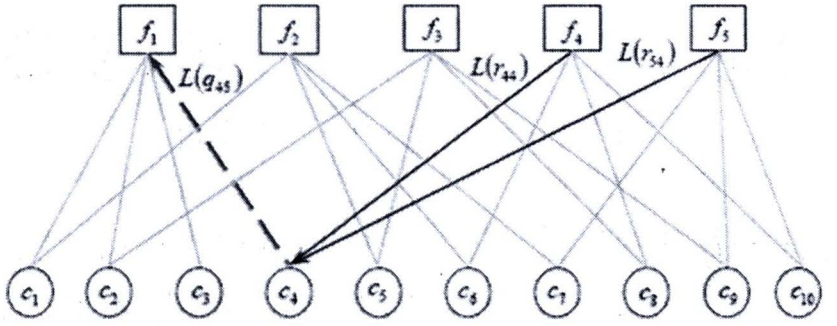


รูปที่2.6แสดงแผนภาพการคำนวณค่า$L(r_{44})$

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

$$L\left(q_{ij}\right)=L\left(c_i\right)+\sum_{j'\in C_{i\backslash j}} L\left(r_{j'i}\right) \tag{1.48}$$

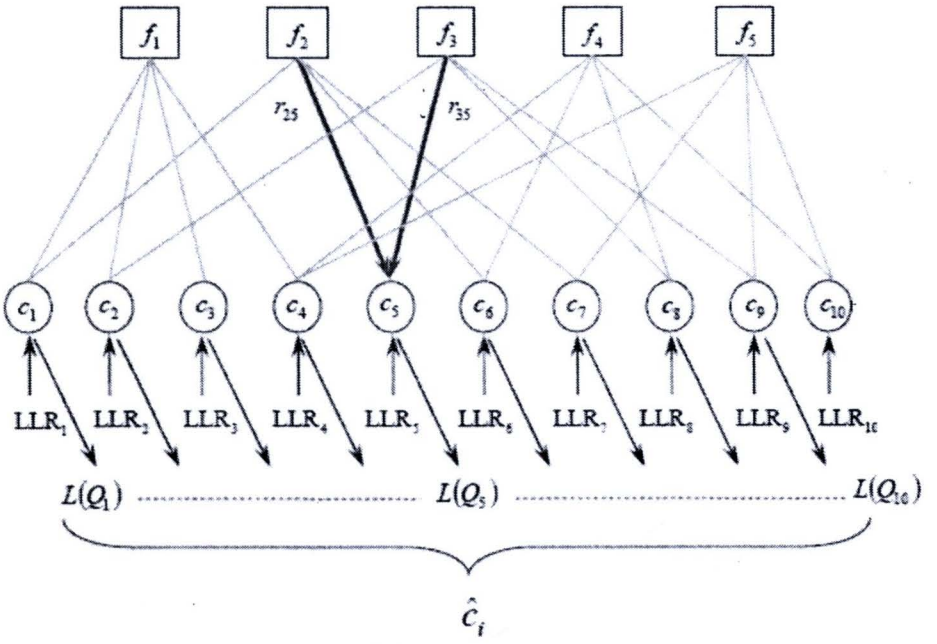
$Ci\backslash j$ แทนการพิจารณาผลรวมของข่าวสาร$L(r_{ji})$ จากทุกเช็คโหนดj ที่เชื่อมต่อกับบิตโหนดยกเว้นข่าวสารที่ใช้เส้นทางเดียวกับ$L(q_{ij})$ รูปที่7 แสดงแผนภาพการปรับปรุงข่าวสารของ$L(q_{ij})$



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

ขั้นตอนที่ 4เป็นการคำนวณหาค่าซอฟต์เอาต์พุตของการถอดรหัสของแต่ละบิตi ตั้งแต่บิตที่ 1ถึงบิตที่n ผ่านสมการที่ 2.49

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



รูปที่2.8แสดงแผนภาพการหาค่าซอฟต์เอาต์พุตของการถอดรหัส

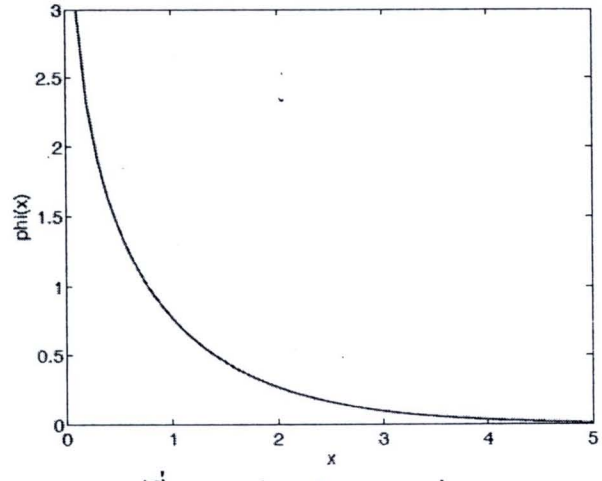
ขั้นตอนที่ 5เป็นการนำค่าซอฟต์เอาต์พุตที่ได้จากขั้นตอนที่ 4 ของแต่ละบิตมาทำการตัดสินใจแบบหยาบผ่านสมการที่ 2.50

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

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

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

ปรับปรุงฟังก์ชั่น $\phi(x)$

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

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

เมื่อ

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

ปรับปรุงได้เป็น

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

เมื่อสรุปขั้นตอนทั้งหมดของวิธีการหาค่าต่ำสุดจะแสดงให้เห็นได้ดังนี้

ขั้นตอนที่ 1 การคำนวณค่าเริ่มต้น เนื่องจากเป็นขั้นตอนที่มาจากการประมาณค่า ค่า AWGN จึงไม่มีผลต่อค่าเริ่มต้นจึงปรับปรุงสมการได้ดังนี้

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

ขั้นตอนที่ 2คำนวณหาค่าที่เช็คโหนด (CNU)

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

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 γ = 0.6-0.8 สำหรับค่าที่เหมาะสมที่จะนำไปใช้ในการออกแบบฮาร์ดแวร์จะใช้ค่า γ = 0.75 ซึ่งเราใช้ในการวิจัยครั้งนี้

2.4 งานวิจัยที่เกี่ยวข้อง

ในช่วงเวลาหลายปีที่ผ่านมา ได้มีการค้นคว้าเกี่ยวกับการวิเคราะห์สัญญาณไฟฟ้ากล้ามเนื้อแขน ซึ่งจะมีทั้งการวิเคราะห์ความผิดปรกติของกล้ามเนื้อ และการจำแนกลักษณะเฉพาะของกล้ามเนื้อ ดังนั้นผู้จัดทำจึงนำเอางานวิจัยเกี่ยวข้องกับการจำแนกลักษณะเด่นของกล้ามเนื้อแขนในปีที่ผ่านมาสามารถกล่าวโดยสรุปได้ดังนี้

รหัสแอลดีพีซีได้รับการคิดค้นขึ้นมาโดย Robert Gallagerแห่งสถาบันMITในปี1963โดยรหัสนี้มีประสิทธิภาพเข้าใกล้แชนนอนลิมิตแต่ในขณะนั้นแทบจะไม่ได้รับความสนใจแต่อย่างใด เนื่องจากการนำไปใช้งานยังมีความซับซ้อนสูง และ 20 ปีถัดจากนั้นMichaelTanner [2] ได้นำเสนอกราฟของรหัสแอลดีพีซีเพื่อให้เข้าใจง่ายขึ้นโดยเรียกว่าbi-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].

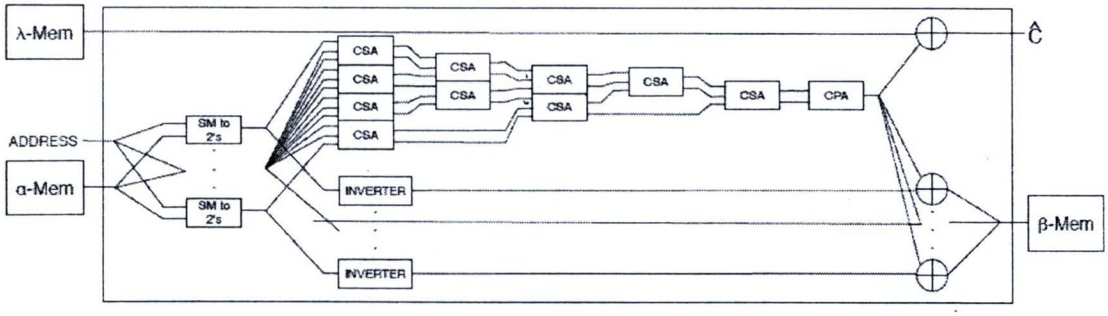
สังเกตุได้ว่าช่องว่าง (-) ของเมตริกซ์ หรือบล็อคที่ไม่มีการคำนวณมีจำนวนมากพอที่จะทำการสับเปลี่ยนตำแหน่ง แต่การสับเปลี่ยนตำแหน่งจะต้องไม่ส่งผลกระทบต่อประสิทธิภาพการถอดรหัสดังนั้นกฎการสลับตำแหน่งจะแสดงให้เห็นได้รูปแบบด้านล่าง

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

เมตริกซ์มาตราฐาน เมตริกซ์ที่ถูกสลับตำแหน่งแล้ว

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

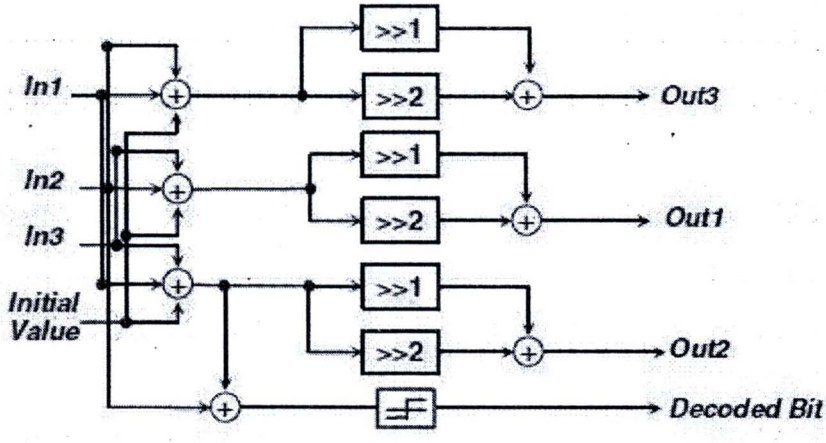
ในส่วนของหน่วยประมวลผลเราได้นำโครงสร้างของ Hiroyuki SHIMAJIRIและคณะ [6]ที่ออกแบบสำหรับการสื่อสารไร้สายตามมาตราฐานIEEE 802.11n โดยเรานำไปใช้ออกแบบฮาร์ดแวร์ในการถอดรหัสด้วยวิธีการหาค่าต่ำสุด (Min-Sum Algorithm)



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