การวางสลับตำแหน่งข้อมูลแบบแถวและหลัก
เพื่อนำไปใช้ในการเข้ารหัสแบบ Turbo Code
Row-Column Interleaver for using in Turbo Code

นายจีระศักดิ์ อินพิการ

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์
สาขาวิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2544

การวางสลับบิตแบบสลับทั้งแถวและหลัก เพื่อนำไปใช้ในการเข้ารหัสแบบ Turbo Code

Row-Column Interleaver for using in Turbo Code

นายจิระศักดิ์ สิทธิกร

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์
สาขาวิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2544

โครงงาน	การวางสลับบิตแบบสลับทั้งแถวและหลัก เพื่อนำไปใช้ในการเข้ารหัสแบบ Turbo Code
โดย	นายจิระศักดิ์ สิทธิกร
อาจารย์ที่ปรึกษา	อาจารย์พีระพงษ์ อุฑารสกุล
สาขาวิชา	วิศวกรรมโทรคมนาคม
ภาคการศึกษาที่	1/2544

บทคัดย่อ

เนื่องจากการเข้ารหัสและถอดรหัสแบบ Turbo Code เป็นการเข้ารหัสและถอดรหัสที่มีประสิทธิภาพมาก และเป็นเรื่องที่มีผู้สนใจศึกษามากที่สุดเรื่องหนึ่ง โครงงานนี้จึงทำการศึกษาการเข้ารหัสและถอดรหัสแบบ Turbo Code เพื่อจะปรับปรุงการทำงานในส่วน Deterministically Generate Interleaver โดยสร้าง Row-Column Interleaver และทำการทดสอบโปรแกรมการเข้ารหัสและถอดรหัสแบบ Turbo Code เพื่อเปรียบเทียบประสิทธิภาพ ของการเข้ารหัสและถอดรหัสแบบ Turbo Code ที่ใช้ Interleaver หลายแบบที่ความยาวเฟรมแตกต่างกัน และยังได้สร้างเครื่องมือสร้างแบบจำลองการเข้ารหัสและถอดรหัสแบบ Turbo Code (Turbo Code Toolbox) ด้วยโปรแกรม Matlab ที่สามารถเลือกใช้เครื่องมือสร้าง Interleaver และแบบจำลองช่องสัญญาณต่างๆกันได้

จากผลการทดสอบที่ได้ Row-Column Interleaver มีแนวโน้มที่ดีกว่าเมื่อเทียบกับ Circular Shifting Interleaver เล็กน้อย และดีกว่า Pseudo-Random Interleaver ที่ความยาวเฟรมน้อย แต่ที่ความยาวเฟรมมากๆ Pseudo-Random Interleaver จะให้ผลที่ดีที่สุด

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงด้วยดี ผู้จัดทำขอกราบขอบพระคุณเป็นอย่างสูง สำหรับ บุคคล และกลุ่มบุคคลต่างๆ ที่ได้กรุณาให้คำปรึกษา แนะนำ ช่วยเหลือ อย่างดียิ่ง ทั้งในด้านวิชาการ และด้านการดำเนินงาน ดังนี้

- อ.พีระพงษ์ อุฑารสกุล ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานชิ้นนี้ ที่ให้คำแนะนำ คำปรึกษาในการศึกษา และ สนับสนุนการดำเนินงานมาโดยตลอด
- อ.ดร. รังสรรค์ ทองทา และ อ.ประโยชน์ คำสวัสดิ์ ซึ่งกรุณาสละเวลาอันมีค่ามาเป็นคณะกรรมการสอบในโครงงานนี้
- อ.สมศักดิ์ วาณิชอนันต์ชัย ซึ่งสนับสนุน และ ให้คำแนะนำอันมีประโยชน์ในการดำเนินงาน
- บุคลากรสาขาวิชาวิศวกรรมโทรคมนาคม ซึ่งสนับสนุนการดำเนินงาน
- เหล่าคณาจารย์ ซึ่งได้ประสิทธิ์ประสาทวิชาความรู้

ท้ายนี้ขอกราบขอบพระคุณ บิดา มารดา ที่ให้การเลี้ยงดูอบรมและส่งเสริมการศึกษาเป็นอย่างดีตลอดมาในอดีต จนทำให้ผู้จัดทำประสบความสำเร็จในชีวิตตลอดมา

จิระศักดิ์ สิทธิกร

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมา

การสื่อสารข้อมูลได้รับการพัฒนาอย่างต่อเนื่องมาตลอด ทั้งนี้เป็นเพราะต้องการให้ข้อมูลถูกต้องและมีประสิทธิภาพ การเข้ารหัสและการถอดรหัสสัญญาณจึงเป็นส่วนที่มีความสำคัญมาก ในปัจจุบัน

Turbo Code ถูกคิดค้นเมื่อปี ค.ศ. 1993 โดย Berrou, Glavieux และ Thitimajshima ซึ่งชื่อ Turbo Code มาจากส่วนที่เป็นการถอดรหัสที่ทำงานคล้ายกับเครื่องยนต์ Turbocharger โดยการถอดรหัสจะอาศัยการนำข้อมูลที่เป็นผลลัพธ์จากการถอดรหัสชุดอื่นมาช่วยในการถอดรหัส และทำการวนถอดรหัสหลายรอบ Turbo Code เป็นการเข้ารหัสและถอดรหัสแบบ Parallel Concatenated Convolutioncode (PCCC) ซึ่งมีความสามารถในการแก้ไขข้อผิดพลาดของข้อมูลที่มีประสิทธิภาพมากกว่าการเข้ารหัสและถอดรหัสแบบอื่น โดยทำให้ข้อมูลมีอัตราความผิดพลาดต่อบิต (BER, Bit Error Rate) น้อยมากขณะที่ส่งข้อมูลด้วยอัตราของพลังงานต่อสัญญาณรบกวน (SNR, Signal to Noise Ratio) มีค่าต่ำ [1]

ดังนั้นโครงงานนี้จึงทำการศึกษาการเข้ารหัสและถอดรหัสแบบ Turbo Code เพื่อสร้างโปรแกรมแบบจำลองการเข้ารหัสและถอดรหัสแบบ Turbo Code และสร้างเครื่องมือสร้างแบบจำลองการเข้ารหัสและถอดรหัสแบบ Turbo Code (Turbo Code Toolbox) เพื่อความสะดวกในการใช้งาน และทำการปรับปรุงการเข้ารหัสและถอดรหัสแบบ Turbo Code ในส่วนที่เป็น Deterministically Generate Interleaver เพื่อศึกษาผลการเข้ารหัสและถอดรหัสแบบ Turbo Code ที่ใช้ Interleaver ต่างกัน เพราะในอนาคตการเข้ารหัสและถอดรหัสแบบ Turbo Code อาจสามารถรองรับเทคโนโลยีใหม่แทนที่การเข้ารหัสและถอดรหัสแบบที่ใช้ในปัจจุบัน

1.2 วัตถุประสงค์ของโครงงาน

- ศึกษาการทำงานของการเข้ารหัสและถอดรหัสแบบ Turbo Code และแบบจำลองช่องสัญญาณ (Channel Model) ซึ่งประกอบด้วย AWGN (Additive White Gaussian Noise), Ricean Fading และ Rayleigh Fading
- สร้างเครื่องมือสร้างแบบจำลอง Turbo Code สำหรับ โปรแกรม Matlab
- ปรับปรุงประสิทธิภาพ Turbo Code ในส่วนที่เป็น Deterministically Generate Interleaver
- วิเคราะห์ประสิทธิภาพของ Turbo Code เมื่อใช้ Interleaver ต่างชนิดกันที่ความยาวเฟรมต่างกัน

1.3 ขอบเขตของโครงงาน

โครงงานนี้ทำการศึกษากระบวนการทำงานของการเข้ารหัสและถอดรหัสแบบ Turbo Code เพื่อปรับปรุงประสิทธิภาพในส่วนที่เป็น Interleaver และเครื่องมือสร้างแบบจำลองการเข้ารหัสและถอดรหัสแบบ Turbo Code แล้วเปรียบเทียบวิเคราะห์ประสิทธิภาพของ Turbo Code เมื่อใช้ Interleaver ต่างชนิดกันที่ความยาวเฟรมต่างกัน

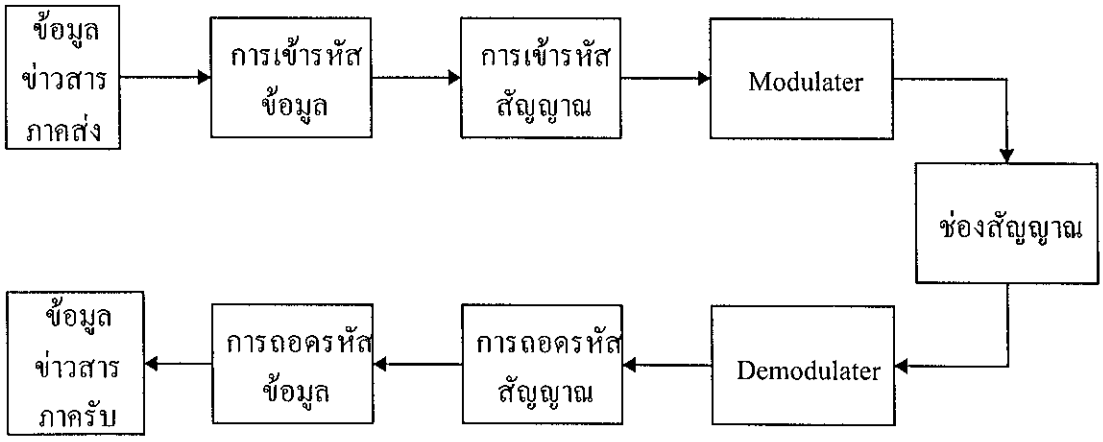
1.4 ขั้นตอนการดำเนินงาน

- ศึกษาทฤษฎี Turbo Code ทั้งส่วนที่เป็น Encoder และ Decoder
- ศึกษาการเขียนโปรแกรมการทำงานการเข้ารหัสและถอดรหัสแบบ Turbo Code
- ศึกษาเกี่ยวกับ Simulink ของโปรแกรม Matlab
- สร้างเครื่องมือสร้างแบบจำลอง Turbo Code สำหรับ โปรแกรม Matlab
- ศึกษาแบบจำลองช่องสัญญาณ ซึ่งประกอบด้วย AWGN , Ricean Fading และ Rayleigh Fading
- ปรับปรุงประสิทธิภาพ Turbo Code ในส่วนที่เป็น Interleaver
- สร้างโปรแกรมของแบบจำลองช่องสัญญาณ และ Interleaver ต่างๆ ซึ่งทั้งส่วนที่เป็น M-file function และเครื่องมือสร้างแบบจำลอง สำหรับ โปรแกรม Matlab
- วิเคราะห์ผล Turbo Code เมื่อใช้ Interleaver ต่างชนิดกันที่ความยาวเฟรมต่างกัน
- สรุปผลการทำงาน

บทที่ 2
ทฤษฎีของการเข้ารหัสและถอดรหัส

การสื่อสารข้อมูลในระบบโทรคมนาคม มีปัญหาเกี่ยวกับสัญญาณรบกวนทั้งในระบบ และสัญญาณรบกวนในขณะที่ส่งผ่านช่องสัญญาณ ทำให้เกิดความผิดพลาดของข้อมูล จึงจำเป็นที่ระบบต้องมีการตรวจสอบความผิดพลาดข้อมูล (Error Detection) และมีความสามารถในการแก้ไขข้อผิดพลาดของข้อมูล (Error Correction) หรือป้องกันข้อผิดพลาดได้ (Error Preventing) [2]

การเข้ารหัสเพื่อตรวจสอบความผิดพลาดข้อมูล เป็นการเข้ารหัสโดยการเพิ่ม Parity Check Bit เพื่อจะตรวจสอบว่าข้อมูลมีความผิดพลาดหรือไม่ ซึ่งตรวจสอบได้ในระดับหนึ่ง ถ้าพบว่ามีความผิดพลาดภาครับจะส่งคำสั่งให้ภาคส่งส่งข้อมูลมาใหม่ หรือถ้าภาคส่งไม่ได้รับข้อมูลจนเกินช่วงเวลาหนึ่ง (Timeout) ภาคส่งจะส่งข้อมูลเดิมอีกครั้ง เรียกว่า ARQ (Automatic Repeat Request) ในรายงานนี้จะกล่าวเฉพาะส่วนที่เป็นการเข้ารหัสเพื่อแก้ไขข้อผิดพลาดของข้อมูล ได้แก่การเข้ารหัสที่มีการถอดรหัสแบบ FEC (Forward Error Correction) [2] โดยการเพิ่ม Redundancy Check Bit ในการเข้ารหัสเพื่อสามารถแก้ไขข้อมูลที่ผิดพลาดได้ในการถอดรหัส ซึ่งขึ้นอยู่กับวิธีการเข้ารหัสและถอดรหัสแบบต่างๆ



ภาพที่ 2-1 ภาพแสดงระบบสื่อสารข้อมูล

ภาพที่ 2-1 เป็นภาพระบบสื่อสารข้อมูลโดยเริ่มจากการเข้ารหัสข้อมูลข่าวสาร แล้วนำสัญญาณข่าวสารที่ได้มาเข้ารหัสเพิ่ม Redundancy Check Bit เป็นผลลัพธ์การเข้ารหัสข้อมูล (Code word) เพื่อทำการมอดูเลต (Modulated) ส่งออก เมื่อสัญญาณจากภาคส่งผ่านช่องสัญญาณทำให้สัญญาณที่ส่งออกเปลี่ยนแปลง เนื่องจากสัญญาณรบกวนจากสิ่งแวดล้อมต่างๆ เมื่อนำสัญญาณที่รับมาทำการดีมอดูเลต (Demodulated) ได้เป็นรหัสข้อมูล จึงนำไปถอดรหัสเพื่อให้เป็นสัญญาณข่าวสาร ในขั้นตอนนี้จะช่วยตรวจสอบความผิดพลาด แก้ไขข้อผิดพลาดได้ เนื่องจากการเข้ารหัสและถอดรหัสแบบต่างๆ แล้วนำสัญญาณข่าวสารมาถอดรหัสข้อมูลเป็นข่าวสารที่ต้องการ

การเข้ารหัสและถอดรหัสเพื่อแก้ไขข้อผิดพลาดของข้อมูล หรือป้องกันข้อผิดพลาด มีการเข้ารหัสและถอดรหัสด้วยกันหลายแบบ ซึ่งรายงานฉบับนี้จะกล่าวถึง 3 แบบ ได้แก่ Linear Block Code, Convolutional Code และ Turbo Code เนื่องจาก Linear Block Code เป็นการเข้ารหัสและถอดรหัสแบบเบื้องต้นที่เป็นพื้นฐานการเข้ารหัสและถอดรหัสแบบอื่น และ Convolutional Code เป็นการเข้ารหัสและถอดรหัสที่มีส่วนเกี่ยวข้องกับการเข้ารหัสและถอดรหัสแบบ Turbo Code [3]

2.1 Linear Block Code

Linear Block Code เป็นการเข้ารหัสแบบ FEC โดยอาศัยการเข้ารหัสสร้าง Syndrome สำหรับสร้างรหัสข้อมูล ในการเข้ารหัส และตรวจสอบความผิดพลาดของข้อมูลพร้อมแก้ไขข้อมูลที่ผิดพลาด ในการถอดรหัส

2.1.1 การเข้ารหัส Linear Block Code

Linear Block Code แสดงด้วย Code (n, k) ทำให้ได้ผลลัพธ์การเข้ารหัสข้อมูล ซึ่งประกอบด้วยข้อมูล จำนวน k บิต และ Redundancy Check Bit จำนวน $n-k$ บิต และได้ผลลัพธ์รหัสข้อมูลที่ถูกต้อง (Subspace S) จำนวน 2^k ชุด จากผลลัพธ์รหัสข้อมูล (Space V) ทั้งหมดจำนวน 2^n ชุด

การเข้ารหัสเพื่อให้ได้ผลลัพธ์การเข้ารหัสข้อมูล (U) จากข้อมูลที่รหัสโดยนำเมตริกของข้อมูล (m , Message) จำนวน k บิต คูณกับ เมตริกของรหัส (G , Generator Matrix) ดังสมการที่ 2-1 ซึ่งเมตริกของรหัส (G) ประกอบด้วย Arbitrary Matrix (P) ขนาด $(k, n-k)$ และ เมตริกคุณสมบัติ (I , Identity Matrix) ขนาด k, k ดังสมการที่ 2-2

$$U = mG \qquad \text{สมการที่ 2-1}$$

$$G = \left[P_{k,n-k} \middle| I_k\right] \qquad \text{สมการที่ 2-2}$$

ตัวอย่าง การเข้ารหัส Linear Block Code

กำหนดให้ $G = \left[\begin{array}{ccc|ccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$

จะได้ผลการเข้ารหัสดังตารางที่ 2-1

ข้อมูล (m)	ผลการเข้ารหัสข้อมูล (U)
0 0 0	0 0 0 0 0 0
0 0 1	1 1 0 0 0 1
0 1 0	1 0 1 0 1 0
0 1 1	0 1 1 0 1 1
1 0 0	0 1 1 1 0 0
1 0 1	1 0 1 1 0 1
1 1 0	1 1 0 1 1 0
1 1 1	0 0 0 1 1 1

ตารางที่ 2-1 ตารางแสดงตัวอย่างการเข้ารหัส Linear Block Code

2.1.2 การถอดรหัส Linear Block Code

เนื่องจากการเข้ารหัสเพื่อให้ได้ผลลัพธ์การเข้ารหัสข้อมูล (U) ซึ่งจะประกอบด้วยบิตข้อมูล และ Redundancy Check Bit เมื่อมีความผิดพลาดของข้อมูลการถอดรหัสจะตรวจสอบว่าข้อมูลบิตใดผิด แล้วจึงแก้ไข โดยเริ่มจากสร้าง Parity Check Matrix (H^T) จาก เมตริกของรหัส (G) ดังสมการที่ 2-3 ถึงสมการที่ 2-6

$$GH^T = [0] \qquad \text{สมการที่ 2-3}$$

$$H = \left[I_{n-k} \middle| P^T \right] \qquad \text{สมการที่ 2-4}$$

$$H^T = \begin{bmatrix} I_{n-k} \\ P_{k,n-k} \end{bmatrix} \qquad \text{สมการที่ 2-5}$$

$$GH^T = \left[P_{k,n-k} \middle| I_k \right] \times \begin{bmatrix} I_{n-k} \\ P_{k,n-k} \end{bmatrix} = [0] \qquad \text{สมการที่ 2-6}$$

จากสมการที่ 2-1 และสมการที่ 2-6 จะได้ผลดังสมการที่ 2-7

$$UH^T = mGH^T = m[0] = [0] \qquad \text{สมการที่ 2-7}$$

กำหนดให้ R เป็นบิตที่รับได้ และ e เป็นข้อมูลที่แสดงบิตที่ผิดพลาด ดังนั้นจะได้ดังสมการที่ 2-8 สามารถหา กลุ่มบิตที่แสดงความผิดพลาด (S , Error Syndrome) ได้ดังสมการที่ 2-9

$$R = U + e \qquad \text{สมการที่ 2-8}$$

$$S = RH^T = (U + e)H^T = eH^T \qquad \text{สมการที่ 2-9}$$

จากสมการที่ 2-9 ถ้า $R = U + e = U$ จะทำให้ $S = [0]$ ซึ่งแสดงว่าไม่เกิดการผิดพลาดของข้อมูล สามารถถอดรหัสข้อมูลได้จากตารางที่ 2-1

2.1.2.1 การตรวจสอบความผิดพลาดข้อมูล

Linear Block Code ที่แสดงด้วย Code (n,k) จะได้ผลลัพธ์รหัสข้อมูลทั้งหมดจำนวน 2^n ชุด และได้ผลลัพธ์รหัสข้อมูลที่ถูกต้องจำนวน 2^k ชุด ซึ่งหากเกิดความผิดพลาดของข้อมูล จนเปลี่ยนเป็นผลลัพธ์รหัสข้อมูลอีกตัวก็ไม่สามารถตรวจสอบว่าข้อมูลผิดพลาดได้ ดังนั้นจำนวนชุดข้อมูลที่ผิดพลาดแล้วสามารถตรวจสอบได้เป็นจำนวน $2^n - 2^k$ ชุด

2.1.2.2 การแก้ไขข้อผิดพลาดของข้อมูล

การแก้ไขข้อผิดพลาดของข้อมูลสามารถปฏิบัติ โดยเริ่มจากการสร้างตารางแสดงกลุ่มบิตที่แสดงความผิดพลาด (S)ซึ่งสามารสร้างได้จากการนำข้อมูลที่แสดงบิตที่ผิดพลาด (e) คูณกับ Parity Check Matrix (H^T) ดังสมการที่ 2-9

ตัวอย่าง การแก้ไขข้อผิดพลาด Linear Block Code

จาก $$G = \left[\begin{array}{ccc|ccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

สร้างตารางแสดงกลุ่มบิตที่แสดงความผิดพลาด (S) เพื่อตรวจสอบข้อมูลที่แสดงบิตที่ผิดพลาด (e) ได้ดังสมการที่ 2-9 ดังผลในตารางที่ 2-2

นำข้อมูลที่แสดงบิตที่ผิดพลาดe	กลุ่มบิตที่แสดงความผิดพลาด S
0 0 0 0 0 0	0 0 0
0 0 0 0 0 1	1 1 0
0 0 0 0 1 0	1 0 1
0 0 0 1 0 0	0 1 1
0 0 1 0 0 0	0 0 1
0 1 0 0 0 0	0 1 0
1 0 0 0 0 0	1 0 0
0 0 1 0 0 1	1 1 1

ตารางที่ 2-2 ตารางแสดงตัวอย่างกลุ่มบิตที่แสดงความผิดพลาด S

2.1.2.3 ประสิทธิภาพของ Linear Block Code

เนื่องจาก Linear Block Code ไม่สามารถตรวจสอบความผิดพลาดข้อมูล และแก้ไขข้อผิดพลาดของข้อมูลได้ทุกกรณี จึงสามารถหาประสิทธิภาพในการตรวจสอบความผิดพลาดข้อมูล และการแก้ไขข้อผิดพลาดของข้อมูล ได้จาก Hamming Weight (d), Hamming Distance (W) และ Minimum Distance ($d_{\min}$) ของผลลัพธ์การเข้ารหัสข้อมูล 2 ชุด [2] โดย Hamming Weight แสดงถึงจำนวนบิต "1" ในผลลัพธ์การเข้ารหัสข้อมูลแต่ละชุด Hamming Distance แสดงถึงจำนวนบิตที่แตกต่างกันในผลลัพธ์การเข้ารหัสข้อมูล 2 ชุด และ Minimum Distance แสดงถึงจำนวนบิตที่แตกต่างกันน้อยที่สุดในผลลัพธ์การเข้ารหัสข้อมูล

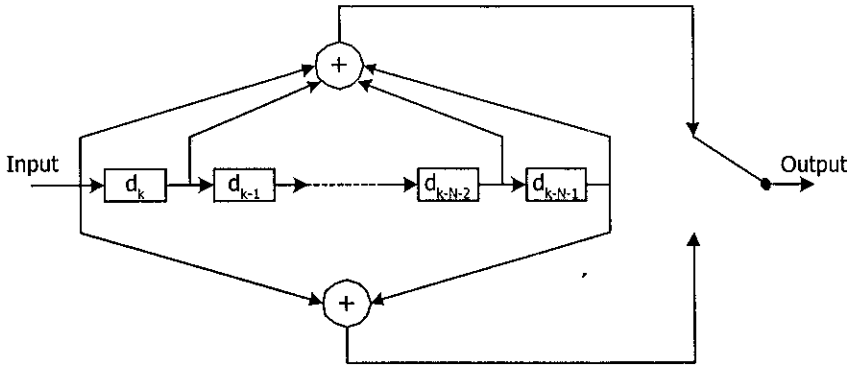
จำนวนรหัสที่สามารถตรวจสอบความผิดพลาดข้อมูลเป็น $(d_{\min}-1)$ จำนวน และจำนวนรหัสที่สามารถแก้ไขข้อผิดพลาดของข้อมูลให้ถูกต้องเป็น $\frac{1}{2}(d_{\min}-1)$ จำนวน

2.2 Convolutional Code

Convolutional Code เป็นวิธีการเข้ารหัสและถอดรหัสแบบ FEC ที่มีความสามารถในการแก้ไขข้อผิดพลาดของข้อมูลที่มีประสิทธิภาพมากกว่า Linear Block Code เนื่องจากการเข้ารหัสแบบ Convolutional Code อาศัยลำดับของสถานะที่ต่อเนื่องกันจากข้อมูลที่จะทำการเข้ารหัส ดังนั้นการถอดรหัสจึงไม่ได้ถอดรหัสเป็นชุดย่อยๆเหมือนกับ Linear Block Code แต่อาศัยข้อมูลที่รับมาได้ทั้งหมดในการถอดรหัส และ Convolutional Code ยังใช้ในระบบโทรศัพท์เคลื่อนที่ในปัจจุบัน [3]

2.2.1 Convolutional Encoder

Convolutional Encoder สามารถเข้ารหัสได้ โดยใช้ Tapped Ship Register และ Modulo – 2 โดยผลของการเข้ารหัส 1 บิต จะทำให้เกิดผลการเข้ารหัสหลายบิต และการเข้ารหัสของแต่ละบิตจะขึ้นอยู่กับบิตที่ทำการเข้ารหัส และบิตที่ผ่านมา [4]



ภาพที่ 2-2 ภาพแสดง Convolutional Encoder

ภาพที่ 2-2 เป็นภาพแสดงวงจร Convolutional Encoder ที่ใช้ Code Generator 2 ชุด ทำให้ได้ผลการเข้ารหัสที่มี rate = ½ และได้จำนวนสถานะ state = 2^{N-1} สถานะ ซึ่งเป็นผลจากจำนวน Register ที่ใช้

ตัวอย่าง Convolutional Encoder G = (5,2), rate = ½

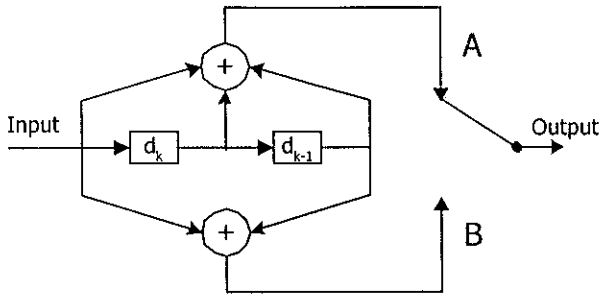
G = (5,2) เป็นการแสดงถึงรหัสในเลขฐาน 8 เพื่อความสะดวก ซึ่งเมื่อเข้ารหัสจะใช้เลขฐาน 2 ดังนั้น G = (1 1 1,1 0 1) Code Generator แถวแรก (1 1 1) แสดงถึงผลลัพธ์ของการเข้ารหัสตัวแรก และ Code Generator แถวที่สอง (1 0 1) แสดงถึงผลลัพธ์ของการเข้ารหัสตัวที่สอง ซึ่งเกิดจากความสัมพันธ์จากบิตที่ผ่านการเข้ารหัส 2 ตัวที่ได้เข้ารหัสก่อนหน้านั้นคือ d_{k-1} และ d_{k-2} ดังภาพที่ 2-2 ผลลัพธ์ A ได้จากการ Modulo – 2 ระหว่าง d_k , d_{k-1} และ d_{k-2} และ ผลลัพธ์ B ได้จากการ Modulo – 2 ระหว่าง d_k และ d_{k-2} ทำให้ผลลัพธ์จากการเข้ารหัสของบิต d_k เป็นดังสมการที่

$$A_k = d_k \oplus d_{k-1} \oplus d_{k-2} \qquad \text{สมการที่ 2-10}$$

หรือ $$A_k = (d_k + d_{k-1} + d_{k-2})(\text{mod} - 2) \qquad \text{สมการที่ 2-11}$$

$$B_k = d_k \oplus d_{k-2} \qquad \text{สมการที่ 2-12}$$

หรือ $$B_k = (d_k + d_{k-2})(\text{mod} - 2) \qquad \text{สมการที่ 2-13}$$

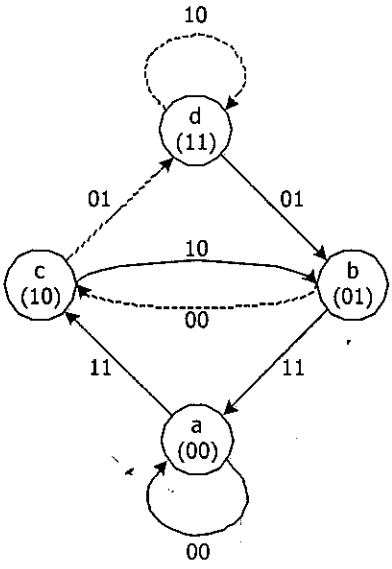


ภาพที่ 2-3 ภาพแสดง Convolutional Encoder, G = (5,2)

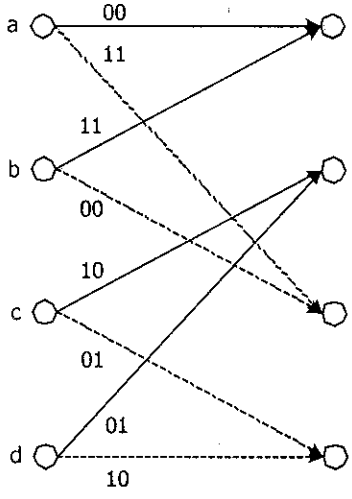
จากภาพที่ 2-3 สามารถแสดงให้เห็นถึงการเข้ารหัสแบบ Convolutional ได้ในระดับหนึ่งเพื่อให้สะดวกขึ้นจึงแสดงผลการเข้ารหัสแบบ Convolutional ดังในตารางที่ 2-3

สถานะเริ่มต้น	บิตข้อมูล	ข้อมูล	บิตผลลัพธ์	สถานะ
a : (0 , 0)	0	000	00	a : (0 , 0)
	1	100	11	c : (1 , 0)
b : (0 , 1)	0	001	11	a : (0 , 0)
	1	101	00	c : (1 , 0)
c : (1 , 0)	0	010	10	b : (0 , 1)
	1	110	01	d : (1 , 1)
d : (1 , 1)	0	011	01	b : (0 , 1)
	1	111	10	d : (1 , 1)

ตารางที่ 2-3 ตารางแสดงผล Convolutional Encoder, G = (5,2)



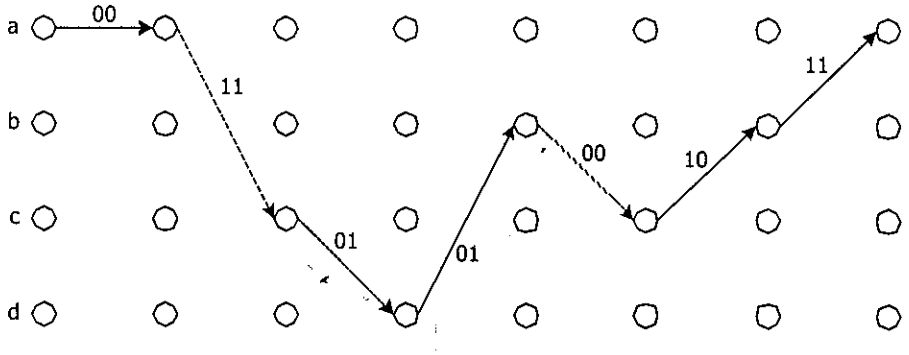
ภาพที่ 2-4 ภาพแสดงแผนผังของสถานะ Convolutional Encoder, G = (5,2)



ภาพที่ 2-4 ภาพแสดง Trellis Diagrams ของ Convolutional Encoder, G = (5,2)

Convolutional Code สามารถแสดงการเข้ารหัสด้วย แผนผันของสถานะ (State Diagrams) ดังภาพที่ 2-4 และสามารถแสดงด้วย Trellis Diagrams ดังแสดงในภาพที่ 2-5 ซึ่งภายในวงกลมแสดงถึงสถานะ เส้นทึบแสดงถึงบิตที่ต้องการเข้ารหัสเป็น 0 เส้นประแสดงถึงบิตที่ต้องการเข้ารหัสเป็น 1 และ ตัวเลขที่อยู่บนเส้นแสดงถึงผลลัพธ์การเข้ารหัส และยังแสดงการเข้ารหัสด้วย

ตัวอย่าง การเข้ารหัส Convolutional Code, G = (1 1 1,1 0 1) ด้วย Trellis Diagrams
กำหนดให้ ข้อมูลบิตที่ทำการเข้ารหัสเป็น 0 1 1 0 1 0 0



ภาพที่ 2-4 ภาพแสดงการเข้ารหัส Convolutional Code, G = (5,2) ด้วย Trellis Diagrams

จากภาพที่ 2-4 จะได้ผลลัพธ์การของเข้ารหัสแบบ Convolutional Code เป็น

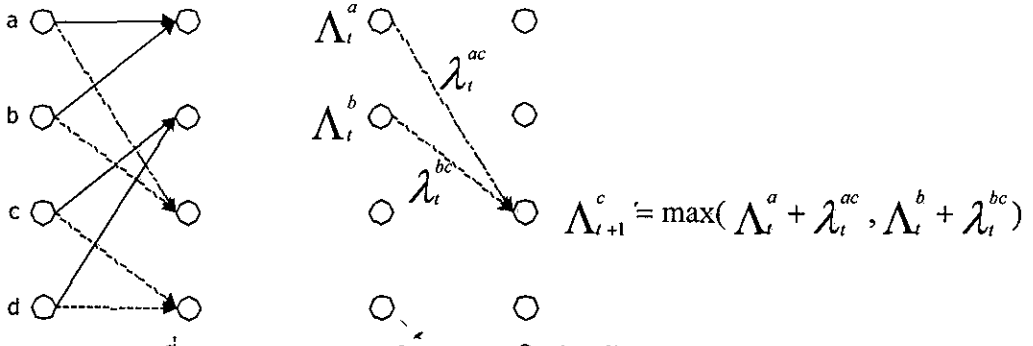
0 0 1 1 0 1 0 1 0 0 1 0 1 1

2.2.2 Convolutional Decoder (Viterbi Decoding)

เนื่องจากการเข้ารหัสแบบ Convolution Code การเข้ารหัสของแต่ละบิตจะขึ้นอยู่กับบิตที่ทำการเข้ารหัส และบิตที่ผ่านมา หรือ สถานะเป็นอยู่ขณะนั้น การถอดรหัสจึงต้องคำนึงถึงบิตที่รับมาอยู่สถานะใด และจะถอดรหัสออกมาเป็นบิตอะไร และยังสามารถแก้บิตข้อมูลที่ผิดให้ถูกต้องได้ การถอดรหัส Convolution Code จำเป็นต้องใช้ Trellis Diagrams [4] ซึ่งรายงานจะกล่าวถึงการถอดรหัสโดยอาศัย Viterbi Decoding

Viterbi Decoding อาศัยความน่าจะเป็นระหว่างสถานะในการถอดรหัสว่าข้อมูลที่เข้ามาสถานะที่คำนวณ (s) ควรเป็นข้อมูลจากสถานะใดที่ผ่านมา (s') โดยประมวลผลจากบิตข้อมูลที่รับ จากข้อมูลที่มี 2^{N-1} สถานะ สามารถหาค่าเมตริก Λ_{t+1}^{s} ซึ่งเป็นสิ่งแสดงความน่าจะเป็นว่าที่สถานะ s ณ เวลา t+1 ควรจะมาเปลี่ยนมากจากสถานะ s' ใด ณ เวลา t ได้ดังสมการที่ 2-14 โดยที่ค่า $\lambda_t^{s's}$ เป็นค่าแสดงถึงความแตกต่างระหว่างค่าที่รับมาได้กับค่าผลลัพธ์จากการเข้ารหัสของสถานะ s' ไปยังสถานะ s [4]

$$\Lambda_{t+1}^{s} = \max_{s'}(\Lambda_t^{s'} + \lambda_t^{s's}) \qquad \text{สมการที่ 2-14}$$



ภาพที่ 2-5 ภาพแสดงการคำนวณเมตริกสำหรับ Viterbi Trellis Decoding

ตัวอย่าง การถอดรหัส Convolutional Code โดย Viterbi Trellis Decoding กำหนดให้ ข้อมูลบิตที่ทำรับได้เป็น 1 0 1 1 0 1 0 0 0 0 1 0 1 1

ขั้นตอนในการถอดรหัส Convolutional Code โดย Viterbi Trellis Decoding

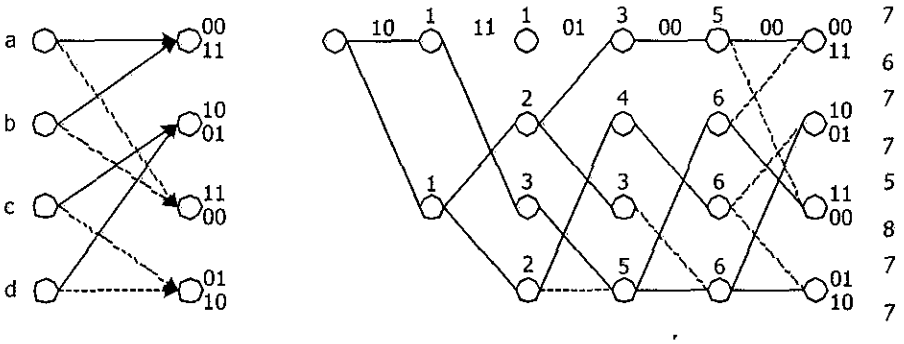
ขั้นที่ 1 เริ่มจากข้อมูลเริ่มต้นจะเริ่มที่เวลา t = 0 สถานะ a (0,0) เพราะใน Ship Register ยังไม่มีข้อมูล และจะได้เมตริก $\Lambda_0^a = 0$ ดังนั้น จากสถานะ a(0,0) จะไปยังสถานะ b(0,1) หรือ สถานะ c(1,0) ด้วยข้อมูลที่ได้เป็น 0 0 หรือ 1 1 แต่ข้อมูลที่รับมาเป็น 1 0 คำนวณค่าเมตริก $\Lambda_1^a = 1$ และ $\Lambda_1^c = 1$ ดังภาพที่ 2-6-a

ขั้นที่ 2 จากข้อมูลที่รับมาเป็น 1 1 คำนวณหาค่าเมตริก $\Lambda_2^a = 1$, $\Lambda_2^b = 2$, $\Lambda_2^c = 3$ และ $\Lambda_2^d = 2$ ดังภาพที่ 2-6-b

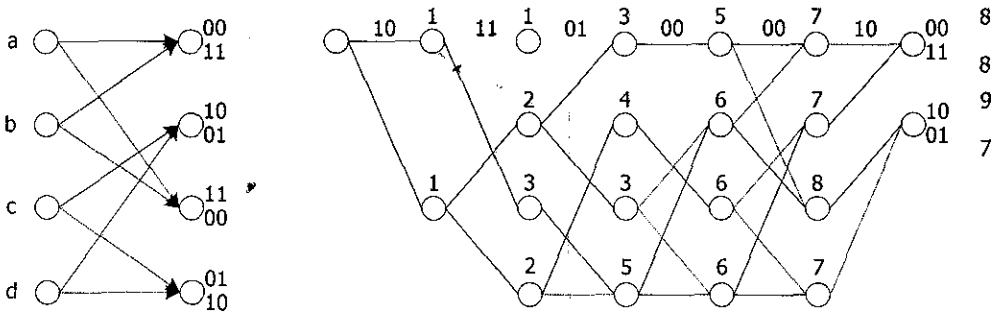
ขั้นที่ 3 และ ขั้นที่ 4 คำนวณค่าเมตริกจากค่าที่รับมาเพื่อตรวจว่ามาจากสถานะใด

ขั้นที่ 5 คำนวณค่าเมตริกเฉพาะที่เปลี่ยนสถานะไปเป็นสถานะ a (0,0) หรือ b (0,1) เพื่อทำให้สถานะสุดท้ายเป็นสถานะ a (0,0) ซึ่งกำหนดไว้จากการเข้ารหัส

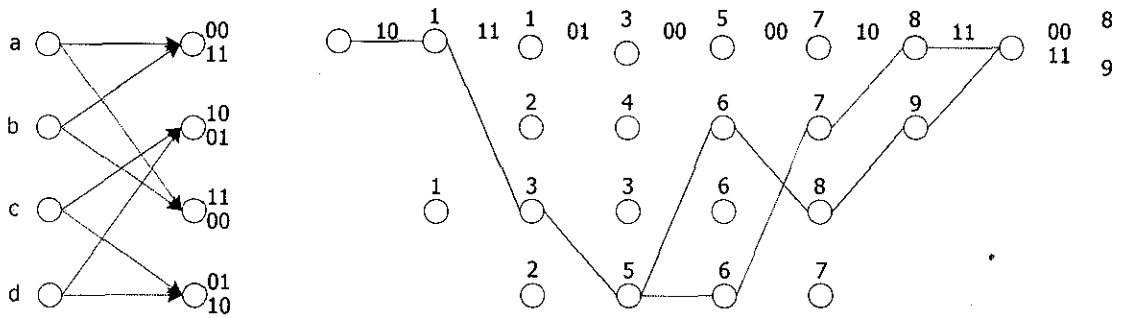
ขั้นที่ 6 คำนวณค่าเมตริกเฉพาะสถานะ a (0,0) ซึ่งเป็นสถานะสุดท้าย แล้วหาเส้นทางการเข้ารหัสจากสถานะ a (0,0) ที่เวลา t = 6 ถึงสถานะ a (0,0) ที่เวลา t = 0 แล้วคำนวณการเข้ารหัสจากสถานะต่างๆ เป็นข้อมูลข่าวสาร



(e)



(f)



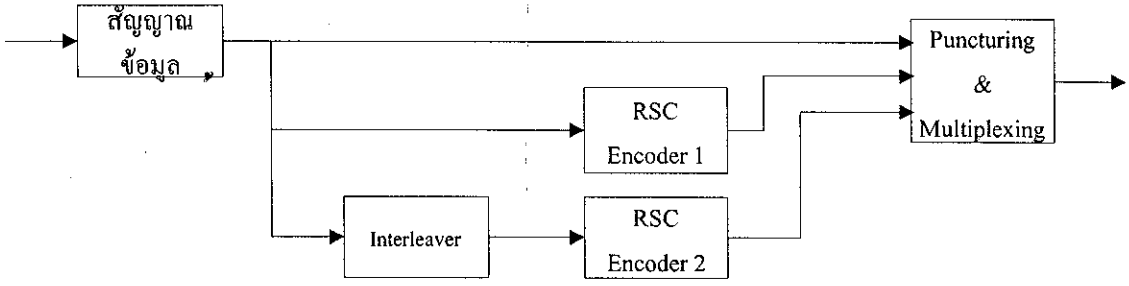
(g)

ภาพที่ 2-6 ภาพแสดงการถอดรหัส Viterbi Trellis Decoding

2.3 Turbo Code

2.3.1 Turbo Code Encoder

Turbo Code Encoder เป็นการเข้ารหัสที่ประกอบด้วย Recursive Systematic Convolutional (RSC) Encoder 2 ตัว [1] และส่วนประกอบที่มีความสำคัญด้วยกันในการเข้ารหัส 3 ส่วนด้วยกัน คือ Interleveaver, RSC Encoder และ Puncturing and Multiplexing ดังแสดงในภาพที่ 2-7



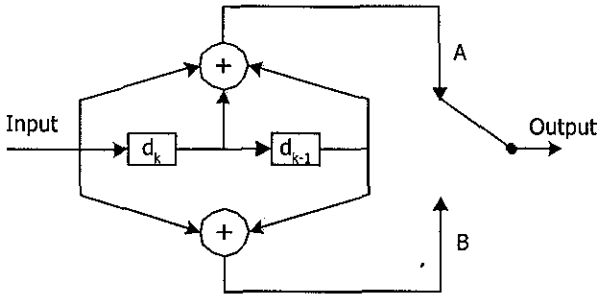
ภาพที่ 2-7 ภาพแสดง Turbo Encoder

2.3.1.1 Interleaver

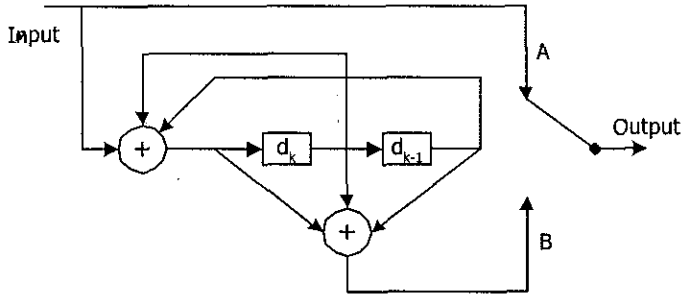
Interleaver เป็นส่วนสำคัญของการเข้ารหัสแบบ Turbo Code ทำหน้าที่สลับลำดับข้อมูลก่อนที่จะทำการเข้ารหัสแบบ RSC เพื่อสร้าง Redundancy Check Bit ของข้อมูลชุดเดียวกันให้มีความแตกต่างกัน ซึ่งปัจจุบัน Interleaver สำหรับ Turbo Code มี 2 ชนิดได้แก่ Pseudo-Random Interleaver และ Deterministically Generated Interleaver [5] ซึ่งจะกล่าวในหัวข้อ 2.4

2.3.1.2 Recursive Systematic Convolutional Encoder

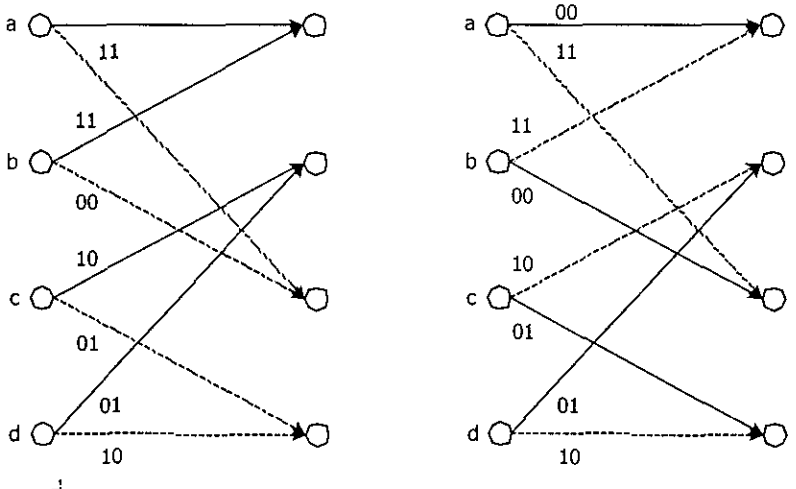
RSC Encoder เป็นตัวเข้ารหัสที่เหมาะสมกับ Turbo Code [1] ซึ่งความคล้ายกับ Convolutional Code แต่ Convolutional Code เป็น Nonsystematic Convolutional (NSC) Code และส่วนที่แตกต่างจาก Convolutional Code ได้แก่ วงจรเข้ารหัส และลำดับสถานะการเข้ารหัส



ภาพที่ 2-8 ภาพแสดง Nonsystematic Convolutional (NSC) Code



ภาพที่ 2-9 ภาพแสดง Recursive Systematic Convolutional (NSC) Code



ภาพที่ 2-10 ภาพแสดง Trellis Decoding ของ NSC Code และ RSC Code

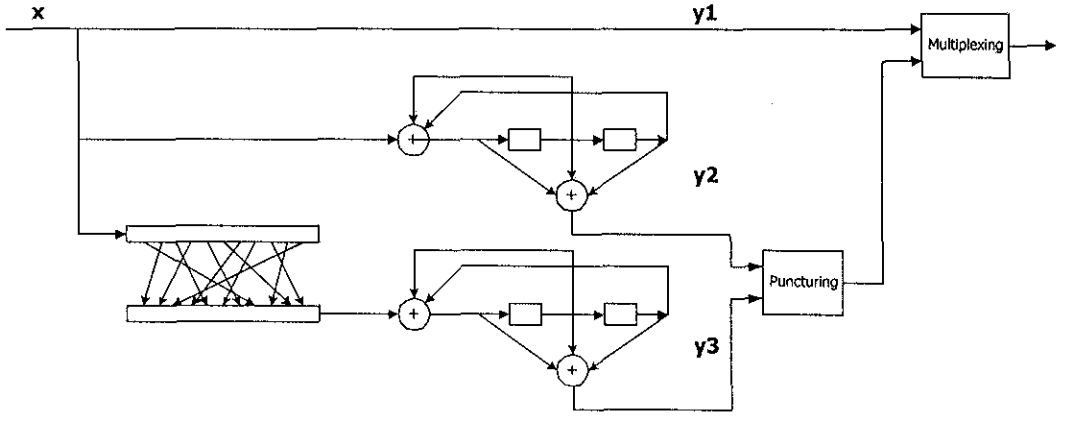
ภาพที่ 2-8 กับภาพที่ 2-9 แสดงให้เห็นข้อแตกต่างว่า NSC Code สถานะของการเข้ารหัสจะขึ้นอยู่กับเฉพาะ บิตในปัจจุบัน แต่ RSC Code จะนำบิตที่ผ่านไปแล้ววนกลับมาใช้ในการเป็นสถานะอีกครั้ง และผลลัพธ์ของ RSC Code ที่ได้จะประกอบด้วย บิตที่ทำการเข้ารหัส และ Redundancy Check Bit แต่ RSC Code จะให้ผลลัพธ์ที่เป็น Redundancy Check Bit และจากภาพที่ 2-10 พบว่าในทุกสถานะ RSC Code สามารถเกิดได้จากบิตที่เป็น 1 และ 0 แต่ NSC Code จะเกิดได้จากบิตที่เข้ารหัสเป็นเฉพาะ 1 หรือ 0 เท่านั้น

2.3.1.3 Puncturing and Multiplexing

Puncturing เป็นส่วนที่กำหนดจำนวน Redundancy Check Bit ของ Turbo Code Encoder เพื่อส่งไป Multiplexing ทำให้มี rate ต่างกัน rate ที่มากที่สุด คือ 1/(n+1) เมื่อ n คือ จำนวน RSC Code ที่ใช้ในการเข้ารหัส

Multiplexing ทำหน้าที่รวมข้อมูลที่ส่งขนานกันมาให้เป็นข้อมูลที่อนุกรมกัน เพื่อที่จะทำการ Modulated ส่งออกเป็นสัญญาณ Pass Band

ตัวอย่าง Turbo Code Encoder ที่ประกอบด้วย Recursive Systematic Convolutional (RSC) Encoder เพียง 2 ตัว โดยใช้ Generator RSC Code, G = [7,5] ดังแสดงในภาพที่ 2-11



ภาพที่ 2-11 ภาพแสดง Turbo Code Encoder (G = [7,5])

2.3.2 Turbo Code Decoder (Iterative Decoding)

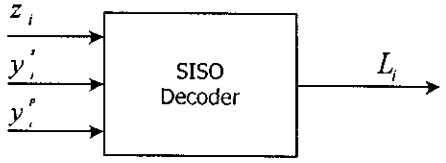
Turbo Code เป็นการเข้ารหัสที่มีการถอดรหัสที่อาศัยการนำค่าที่ถอดรหัสแล้วกลับมาคิดใหม่ด้วย เหมือนเครื่องยนต์ Turbocharger ซึ่งกระบวนการถอดรหัสต้องมีประสิทธิภาพที่จะช่วยแก้ข้อผิดพลาดที่เกิดขึ้น [1]

สำหรับปัญหาในการกำหนดสถานะของ Markov process เมื่อสัญญาณถูกรบกวนทำให้เกิดผลกระทบต่อ Trellis Base Solution มีวิธีที่นิยมใช้แก้ปัญหา 2 วิธี ได้แก่ Maximum A Posteriori (MAP) Algorithm และ Viterbi Algorithm ซึ่ง MAP Algorithm จะทำให้เกิดค่า Symbol Error Rate น้อย ส่วน Viterbi Algorithm จะทำให้เกิดค่า FER น้อย [6]

ปัญหาในการถอดรหัส Turbo Code คือความยุ่งยากในการประมาณการต่อเนื่องของสถานะต่างๆ สำหรับการใช้ Hard-bit Decisions จะให้ผลเนื่องจากข้อมูล 2 ชุดที่มีความสัมพันธ์กันน้อยมาก จึงการมีอาศัย Soft-bit Decisions ซึ่งใช้การกระจายตัวแบบ A Posteriori Probability (APP) เป็นการกระจายตัวที่อาศัยความน่าจะเป็นในการเรียงลำดับโดยมีเงื่อนไขจากการรับค่าสัญญาณเข้ามา โดยจะสามารถหาค่า Log-Likelihood Ratio (L_i , LLR's) ดังสมาการที่ 2-15

$$L_i = \ln \frac{P[m_i = 1 | y]}{P[m_i = 0 | y]} \qquad \text{สมการที่ 2-15}$$

จากภาพที่ 2-12 SISO Decoder จะได้ค่า Log-Likelihood Ratio (L_i , LLR's) ดังสมการที่ 2- เมื่อ y_i^s คือ ค่าข้อมูลจริง (Systematic), y_i^p คือ ค่า Parity และ Z_i คือผลการการถอดรหัสตัวอื่น

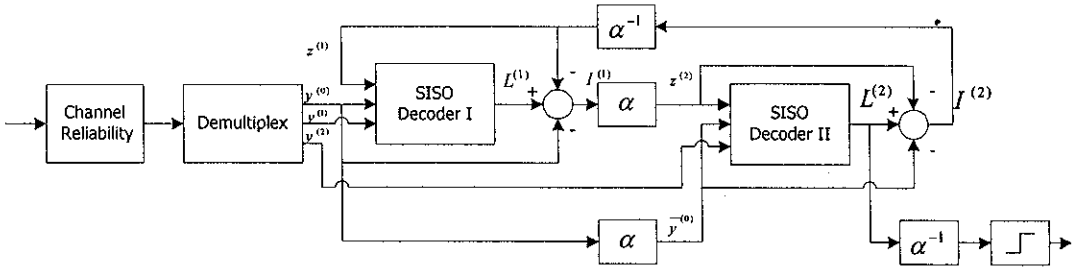


ภาพที่ 2-12 ภาพแสดง Soft -Input Soft-Output Decoder RSC Code

สำหรับค่า LLR's จะใช้ Channel Reliability (L_c) เป็นความน่าเชื่อถือของระดับในการรับสัญญาณจากช่องสัญญาณ ซึ่งคำนวณหาค่าได้ดังสมการที่ 2-16

$$L_c = \frac{4arE_b}{N_0} = 4ar10^{0.1E_b/N_0(dB)} \qquad \text{สมการที่ 2-16}$$

กระบวนการถอดรหัสเริ่มจากทำการแยกข้อมูลที่ได้รับเป็น 3 ชุด ได้แก่ ข้อมูลข่าวสาร ($y^{(0)}$), ข้อมูลจากการเข้ารหัส RSC code ($y^{(1)}$) และข้อมูลจาก Interleaver และการเข้ารหัส RSC code ($y^{(2)}$) จากนั้นนำ $y^{(0)}$, $y^{(1)}$ และ $z^{(1)}$ ซึ่งเป็นผลจาก SISO Decoder II แต่ถ้าทำการถอดรหัสรอบแรก $z^{(1)}$ จะเป็นศูนย์ เมื่อข้อมูล 3 ชุด ผ่าน SISO Decoder I จะได้ผลลัพธ์เป็น $L^{(1)}$ นำ $L^{(1)}$ ลบกับค่า $z^{(1)}$ และ $y^{(0)}$ ได้ $I^{(1)}$ นำผลลัพธ์ $I^{(1)}$ ผ่าน Interleaver เพื่อให้ได้ข้อมูลที่มีความสัมพันธ์กับ $y^{(2)}$ ได้ผลลัพธ์เป็น $z^{(2)}$ นำ $y^{(0)}$ ที่ผ่าน Interleaver จะได้ $\bar{y}^{(0)}$, $y^{(2)}$ และ $z^{(2)}$ ผ่าน SISO Decoder II จะได้ผลลัพธ์เป็น $L^{(2)}$ นำ $L^{(2)}$ ลบกับค่า $z^{(2)}$ และ $y^{(0)}$ ที่ผ่าน Interleaver ได้ $I^{(2)}$ นำผลลัพธ์ $I^{(2)}$ ผ่าน Invert Interleaver เพื่อให้ได้ข้อมูลที่มีความสัมพันธ์กับ $y^{(1)}$ จนครบจำนวนรอบการ Iterative แล้วจึงนำค่า $I^{(2)}$ ที่ได้ท้ายสุดผ่าน Invert Interleaver แล้วนำผลที่ได้ผ่านกระบวนการ hard-limiting ดังภาพที่ 2-13



ภาพที่ 2-13 ภาพแสดง Turbo Code Decoder

สำหรับ SISO Decoder ซึ่งมีวิธีที่นิยมใช้ 2 วิธี ได้แก่ MAP Algorithm และ Viterbi Algorithm พบว่า แต่ในรายงานนี้จะจึงกล่าวเฉพาะ MAP Algorithm

Symbol-by-Symbol MAP Algorithm เป็นกระบวนการถอดรหัส Trellis-Basesd Soft-Output ซึ่งแตกต่างกับ Viterbi Algorithm โดยที่ MAP ให้ค่าที่มากที่สุดของกระบวนการถอดรหัส Likelihood Trellis และค่าความน่าจะเป็นน้อยที่สุดที่ให้ผิดพลาด (Word Error Probability) ซึ่ง MAP จะให้ความน่าจะเป็นบิตที่ผิดพลาดที่ต่ำ

2.3.2.1 Maximum A Posteriori (MAP) Algorithm

MAP Algorithm เริ่มจากการคำนวณความน่าจะเป็นที่เป็นไปได้ของบิตถัดไป โดยคำนวณหาความน่าจะเป็นของสถานะต่อไป $P[s_i \rightarrow s_{i+1}|y]$ ดังสมการที่ 2-17 และสามารถใช้ Markov Process ดังสมการที่ 2-18 โดยค่าตัวแปรแสดงดังสมการที่ 2-19 ถึงสมการที่ 2-20

$$P[s_i \rightarrow s_{i+1}|y] = \frac{P[s_i \rightarrow s_{i+1}, y]}{P[y]} \qquad \text{สมการที่ 2-17}$$

$$P[s_i \rightarrow s_{i+1}|y] = \alpha(s_i)\gamma(s_i \rightarrow s_{i+1})\beta(s_{i+1}) \qquad \text{สมการที่ 2-18}$$

$$\alpha(s_i) = P[s_i, (y_0, \ldots, y_{i-1})] \qquad \text{สมการที่ 2-19}$$

$$\gamma(s_i \rightarrow s_{i+1}) = P[s_{i+1}, y_i | s_i] \qquad \text{สมการที่ 2-20}$$

$$\beta(s_{i+1}) = P[(y_{i+1}, \ldots, y_{L-1})|s_{i+1}] \qquad \text{สมการที่ 2-21}$$

จากสมการที่ 2-20 สามารถเปลี่ยนรูปได้ดังสมการที่ 2-22 โดย m_i แสดงถึงข้อมูล และ x_i แสดงถึงผลการเข้ารหัสที่เปลี่ยนสถานะจาก $s_i \rightarrow s_{i+1}$ และ $P[m_i]$ หาได้จากการรับค่า z_i ดังสมการที่ 2-23 $P[y_i|x_i]$ ดังสมการที่ 2-24

$$\begin{aligned}\gamma(s_i \rightarrow s_{i+1}) &= P[s_{i+1}|s_i]P[y_i|s_i \rightarrow s_{i+1}] \\ &= P[m_i]P[y_i|x_i]\end{aligned} \qquad \text{สมการที่ 2-22}$$

$$P[m_i] = \begin{cases} \dfrac{e^{z_i}}{1+e^{z_i}} & : m_i = 1 \\ \dfrac{1}{1+e^{z_i}} & : m_i = 0 \end{cases} \qquad \text{สมการที่ 2-22}$$

$$P[y_i|x_i] = \frac{1}{\sqrt{\pi N_0/E_s}} e^{\frac{-E_s}{N_0}\sum_{q=0}^{n-1}[y_i^q - a_i^{(q)}(2x_i^{(q)}-1)]^2} \quad \text{สมการที่ 2-24}$$

ความน่าจะเป็นของ $\alpha(s_i)$ สามารถหาค่า Forward Recusion และ ความน่าจะเป็นของ $\beta(s_i)$ สามารถหาค่า Backward Recusion ดังสมการที่ 2-25 และสมการที่ 2-26

$$\alpha(s_i) = \sum_{s_{i-1}} \alpha(s_{i-1})\gamma(s_{i-1} \rightarrow s_i) \quad \text{สมการที่ 2-25}$$

$$\beta(s_i) = \sum_{s_{i+1}} \beta(s_{i+11})\gamma(s_i \rightarrow s_{i+1}) \quad \text{สมการที่ 2-26}$$

การหาความน่าจะเป็นของการเปลี่ยนสถานะ $P[s_i \rightarrow s_{i+1}|y]$ สามารถคำนวณได้ดังสมการที่ 2-27 และ สมการที่ 2-28

$$P[m_i = 1|y] = \sum_{S_1} P[s_i \rightarrow s_{i+1}|y] \quad \text{สมการที่ 2-27}$$

$$P[m_i = 0|y] = \sum_{S_0} P[s_i \rightarrow s_{i+1}|y] \quad \text{สมการที่ 2-28}$$

เมื่อ $S_1 = \{s_i \rightarrow s_{i+1} : m = 1\}$ คือการแสดงการเปลี่ยนสถานะจากสถานะหนึ่งไปยังสถานะถัดไปเมื่อข้อมูลเป็นบิต 1 และ $S_1 = \{s_i \rightarrow s_{i+1} : m = 0\}$ คือการแสดงการเปลี่ยนสถานะจากสถานะหนึ่งไปยังสถานะถัดไปเมื่อข้อมูลเป็นบิต 0 จะได้ค่า LLR's ดังสมการที่ 2-29

$$L_i = \ln \frac{\sum_{S_1} \alpha(s_i)\gamma(s_i \rightarrow (s_{i+1})\beta(s_{i+1})}{\sum_{S_0} \alpha(s_i)\gamma(s_i \rightarrow (s_{i+1})\beta(s_{i+1})} \quad \text{สมการที่ 2-29}$$

2.3.2.2 max-log-MAP Algorithm และ log-MAP Algorithm

MAP Algorithm สามารถใช้คำนวณความน่าจะเป็นที่เป็นไปได้ของบิตถัดไป แต่ก็ยังมีปัญญาหาในการนำไปใช้งาน ซึ่งสามารถใช้ log-domain ในการแก้ปัญญาหาได้โดยอาศัย Jacobian Logarithm ดังสมการที่ 2-30 และ $f_c(|y-x|)$ เป็นฟังก์ชันที่แสดงความแตกต่างระหว่างค่าสองค่า ซึ่ง x และ y มีความสัมพันธ์กันจึงสามารถประมาณให้เป็นศูนย์ได้ ดังนั้นจะได้ผลดังสมการที่ 2-31

$$\begin{aligned}\ln(e^{x}+e^{y}) &= \max(x,y)+\ln(1+e^{-|y-x|}) \\ &= \max(x,y)+f_c(|y-x|)\end{aligned} \qquad \text{สมการที่ 2-30}$$

$$\ln(e^{x}+e^{y}) \approx \max(x,y) \qquad \text{สมการที่ 2-31}$$

กำหนดให้ $\bar{\gamma}(s_i \to s_{i+1})$ แสดงถึงค่า natural logarithm ของ $\gamma(s_i \to s_{i+1})$ ดังสมการที่ 2-32 และสามารถแสดงดังสมาการที่ 2-33

$$\begin{aligned}\bar{\gamma}(s_i \to s_{i+1}) &= \ln\gamma(s_i \to s_{i+1}) \\ &= \ln P[m_i] + \ln P[y_i|x_i]\end{aligned} \qquad \text{สมการที่ 2-32}$$

$$\begin{aligned}\bar{\gamma}(s_i \to s_{i+1}) &= \ln P[m_i] - \frac{1}{2}\ln\left(\pi\frac{N_0}{E_s}\right) - \frac{E_s}{N_0}\sum_{q=0}^{n-1}\left[y_i^{(q)} - a_i^{(q)}(2x_i^{(q)}-1)\right] \\ &= \lambda(s_i \to s_{i+1})\end{aligned}$$

สมการที่ 2-33

$\hat{\alpha}(s_i)$ แสดงถึงค่า natural logarithm ของ $\alpha(s_i)$ แสดงดังสมการที่ 2-34 และ $\hat{\beta}(s_i)$ แสดงถึงค่า natural logarithm ของ $\beta(s_i)$ แสดงดังสมการที่ 2-35

$$\begin{aligned}\hat{\alpha}(s_i) &= \ln\alpha(s_i) \\ &= \ln\left\{\sum_{S_{i-1}} e^{\hat{\alpha}(s_{i-1})+\hat{\gamma}(s_{i-1}\to s_i)}\right\} \\ &= \max_{S_{i-1}}{}^{*}\left[\alpha(s_{i-1})+\hat{\gamma}(s_{i-1}\to s_i)\right]\end{aligned} \qquad \text{สมการที่ 2-34}$$

$$\hat{\beta}(s_i) = \ln \beta(s_i)$$
$$= \ln\left\{\sum_{S_{i+1}} e^{\hat{\beta}(s_{i+1})+\hat{\gamma}(s_i \to s_{i+1})}\right\} \quad \text{สมการที่ 2-35}$$
$$= \max_{S_{i+1}}{}^{*} [\alpha(s_{i+1}) + \hat{\gamma}(s_i \to s_{i+1})]$$

จาก $\hat{\alpha}(s_i)$ และ $\hat{\beta}(s_i)$ สามารถหาค่า LLR's ได้ดังสมการที่ 2-36

$$L_i = \ln\left\{\sum_{S_1} \hat{\alpha}(s_i) + \hat{\gamma}(s_i \to (s_{i+1}) + \hat{\beta}(s_{i+1})\right\} - \ln\left\{\sum_{S_0} \hat{\alpha}(s_i) + \hat{\gamma}(s_i \to (s_{i+1}) + \hat{\beta}(s_{i+1})\right\}$$
$$= \max_{S_1}{}^{*}\left[\hat{\alpha}(s_i) + \hat{\gamma}(s_i \to (s_{i+1}) + \hat{\beta}(s_{i+1})\right] - \max_{S_0}{}^{*}\left[\hat{\alpha}(s_i) + \hat{\gamma}(s_i \to (s_{i+1}) + \hat{\beta}(s_{i+1})\right]$$

สมการที่ 2-36

ลำดับกระบวนการ max-log-MAP Algorithm และ log-MAP Algorithm

1. Forward Recusion

- กำหนดค่าเริ่มต้นของ $\hat{\alpha}(s_0)$ ดังสมการที่ 2-37

$$\hat{\alpha}(s_0) = \begin{cases} 1 & : s_0 = 0 \\ 0 & : s_0 \neq 0 \end{cases} \quad \text{สมการที่ 2-37}$$

- กำหนดให้ $i = 1$
- คำนวณ $\hat{\alpha}(s_i)$ ดังสมการที่ 2-34
- เพิ่มค่า i แล้วคำนวณหาค่า $\hat{\alpha}(s_i)$ ดังสมการที่ 2-34 จนครบ L ค่า

2. Backward Recusion ดังสมการ

- กำหนดค่าสุดท้ายของ $\hat{\beta}(s_i)$ ดังสมการที่ 2-38

$$\hat{\beta}(s_L) = \begin{cases} 1 & : s_L = 0 \\ 0 & : s_L \neq 0 \end{cases} \quad \text{สมการที่ 2-38}$$

- ถ้าจาก tellis ไม่สามารถหาค่าสุดท้ายได้ ให้กำหนด $\beta(s_i)$ ดังสมการที่ 2-39

$$\beta(s_L) = 0 \quad \forall s_L \qquad \text{สมการที่ 2-39}$$

- กำหนดให้ $i = L - 1$
- คำนวณ $\beta(s_i)$ ดังสมการที่ 2-35
- ลดค่า i แล้วคำนวณหาค่า $\beta(s_i)$ ดังสมการที่ 2-34 จนครบ L ค่า

3. คำนวณค่า LLR's ดังสมการที่ 2-36 สำหรับค่า $i = (0,..., L-1)$

2.4 Interleaver

2.4.1 Pseudo-Random Interleaver

Pseudo-Random Interleaver เป็นส่วนที่ช่วยในการสลับบิตข้อมูลโดยจำลองการสุ่มลำดับในการสลับบิต คือสุ่มลำดับการสลับบิตเก็บค่าไว้แล้วจึงใช้ค่าเหล่านั้นเป็น Interleaver ซึ่งพัฒนาเป็น S-Random Interleaver ซึ่งจะทำการหาชุดการลำดับบิตที่ดีจาก Pseudo-Random Interleaver แต่ก็ยังมีปัญหาสำหรับเฟรมที่มีขนาดความยาวบิตมากเพราะทำให้เปลืองหน่วยความจำสำหรับเก็บค่า Interleaver [5]

2.4.2 Deterministically Generate Interleaver

Deterministically Generate Interleaver เป็นส่วนที่ช่วยสร้างลำดับการสลับบิตข้อมูลซึ่งเป็น Interleaver ที่สร้างขึ้นอย่างคงตัว เพื่อให้ง่ายต่อการนำไปใช้งาน ซึ่งได้แก่ Block Interleaver B_N และพัฒนาเป็น Circular Shifting Interleaver L_N [7]

2.4.2.1 Block Interleaver

Block Interleaver [7] สร้างจากการกำหนดเมตริกขนาด $N = m \times n$ โดย N คือจำนวนความยาวของบิตทั้งหมด, m คือจำนวนแถวของเมตริก และ n คือจำนวนหลักของ

เมตริก สามารถหาสมการฟังก์ชัน เพื่อหาลำดับของ Block Interleaver ดังสมการที่ 2-40 โดย $\lfloor \cdot \rfloor$ แสดงถึงฟังก์ชัน floor คือการปัดทศนิยมลงเป็นจำนวนเต็ม

$$D_N(i) = ni + \lfloor i/m \rfloor \pmod{N}, \quad 0 \le i < N \qquad \text{สมการที่ 2-40}$$

2.4.2.2 Circular Shifting Interleaver

Circular Shifting Interleaver [7], [8] เป็นการกระจายที่ เรียกว่า Nonrandom Permutation Based on Circular Shifting สามารถหาสมการฟังก์ชัน เพื่อหาลำดับของ Circular Shifting Interleaver ดังสมการที่ 2-41 โดยมีเงื่อนไข $k = \sqrt{2N} - 1$ [8]

$$L_N(i) \equiv ki + v \pmod{N}, \quad 0 \le i < N \qquad \text{สมการที่ 2-41}$$

2.4.3 Row-Column Interleaver

เนื่องจากการสลับบิตข้อมูลเพื่อให้ข้อมูลที่อยู่ติดกันห่างกันออกให้มากที่สุดนั้น เป็นการกระทำเพื่อลดผลกระทบเนื่องจากสัญญาณรบกวน ดังนั้นจึงมีแนวคิดในการสร้าง Interleaver โดยการสร้างเมตริกแล้วทำการสลับลำดับทั้งในส่วนที่เป็นแถวและหลัก แล้วนำเมตริกที่ได้มาเรียงกัน ดังนี้

ขั้นที่ 1 สร้างเมตริกเริ่มต้น

โดยเริ่มจากเมื่อ N คือจำนวนความยาวของบิตทั้งหมด หาค่า $F = \lfloor \sqrt{N} \rfloor$ คำนวณค่า B จาก $B = \lceil N/F \rceil$ สร้างเมตริก ที่มีขนาด F แถว B หลัก โดยที่ $\lceil \cdot \rceil$ แสดงถึงฟังก์ชัน ceil คือการปัดทศนิยมขึ้นเป็นจำนวนเต็ม

ขั้นที่ 2 สร้างการสลับบิตในแนวหลัก

โดยเริ่มจากจำนวนหลักขนาด B หลัก หาค่า $f_B = \lfloor \sqrt{B} \rfloor$ คำนวณค่า b_B จาก $b_B = \lceil B/f_B \rceil$ สร้างเมตริก Interleaver $(f_B \times b_B)$ เก็บค่าที่ได้เป็น b' แล้วทำการสลับบิตในหลักตามค่า b'

ขั้นที่ 3 สร้างการสลับบิตในแนวแถว

โดยเริ่มจากจำนวนแถวขนาด F แถว หาค่า $f_F = \lfloor \sqrt{F} \rfloor$ คำนวณค่า b_F จาก $b_F = \lceil B / f_F \rceil$ สร้างเมตริก Interleaver $(f_F \times b_F)$ เก็บค่าที่ได้เป็น f' แล้วทำการสลับบิตในแถวตามค่า f'

ตัวอย่าง จำนวนความยาวของบิตทั้งหมด 40 บิต

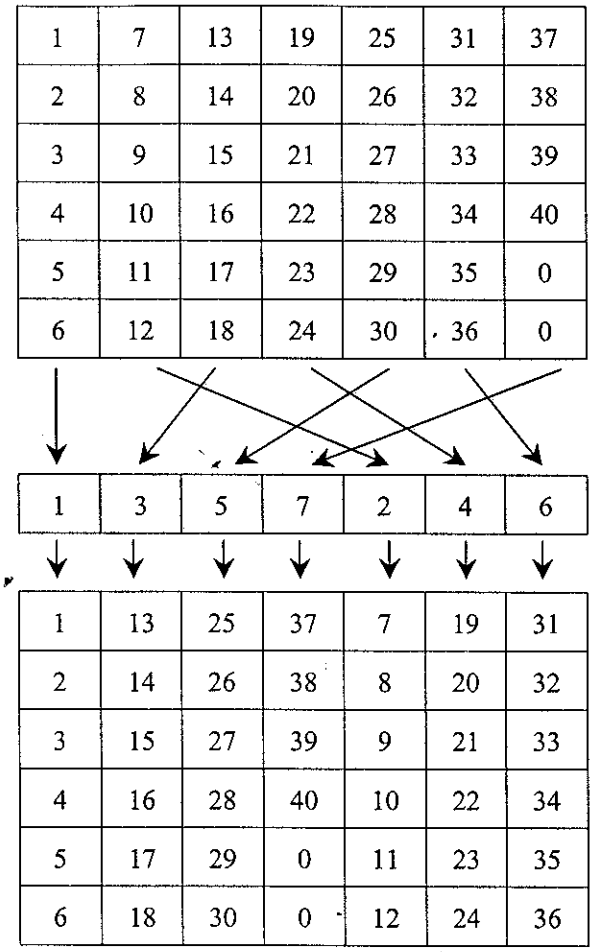
ขั้นที่ 1 สร้างเมตริกเริ่มต้น

เมื่อ $N = 40$ คำนวณค่า $F = \lfloor \sqrt{40} \rfloor = 6$ และ $B = \lceil 40 / 6 \rceil = 7$ สร้างเมตริก ที่มีขนาด 6 แถว 7 หลัก

1	7	13	19	25	31	37
2	8	14	20	26	32	38
3	9	15	21	27	33	39
4	10	16	22	28	34	40
5	11	17	23	29	35	0
6	12	18	24	30	36	0

ขั้นที่ 2 สร้างการสลับบิตในแนวหลัก

จำนวนแถวขนาด 7 หลัก คำนวณค่า $f_B = \lfloor \sqrt{7} \rfloor = 2$ และ $b_b = \lceil 7 / 2 \rceil = 4$ สร้างเมตริก Interleaver (2×4) เก็บค่าที่ได้เป็น $b' = [1 \quad 3 \quad 5 \quad 7 \quad 2 \quad 4 \quad 6 \quad 8]$ แต่เนื่องจากหลักมีขนาด 7 หลักดังนั้น $b' = [1 \quad 3 \quad 5 \quad 7 \quad 2 \quad 4 \quad 6]$ แล้วทำการสลับบิตในหลักตาม b'



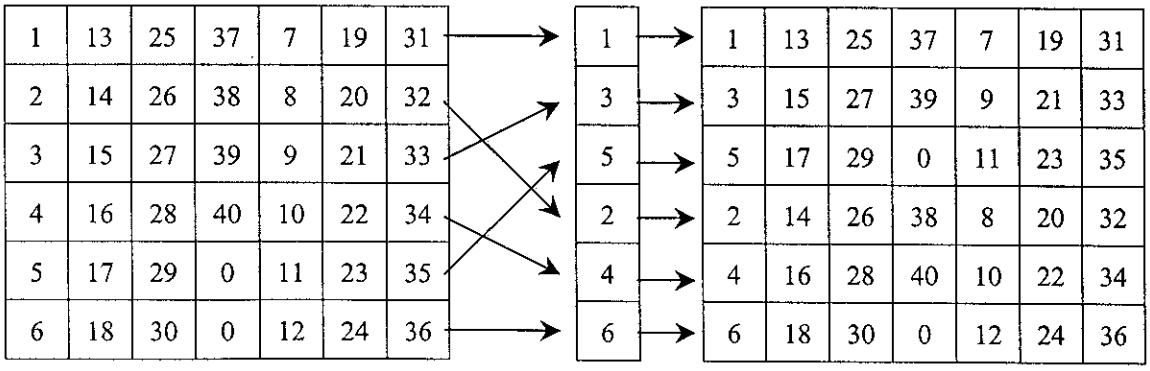
1	7	13	19	25	31	37
2	8	14	20	26	32	38
3	9	15	21	27	33	39
4	10	16	22	28	34	40
5	11	17	23	29	35	0
6	12	18	24	30	36	0

1	3	5	7	2	4	6

1	13	25	37	7	19	31
2	14	26	38	8	20	32
3	15	27	39	9	21	33
4	16	28	40	10	22	34
5	17	29	0	11	23	35
6	18	30	0	12	24	36

ขั้นที่ 3 สร้างการสลับบิตในแนวแถว

จำนวนแถวขนาด 6 แถว คำนวณค่า $f_F = \lfloor \sqrt{6} \rfloor = 2$ และ $b_F = \lceil 6/2 \rceil = 3$ สร้างเมตริก Interleaver (2×3) เก็บค่าที่ได้เป็น $f' = [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 6]$ แล้วทำการสลับบิตในแถวตาม f'



1	13	25	37	7	19	31
2	14	26	38	8	20	32
3	15	27	39	9	21	33
4	16	28	40	10	22	34
5	17	29	0	11	23	35
6	18	30	0	12	24	36

1
3
5
2
4
6

1	13	25	37	7	19	31
3	15	27	39	9	21	33
5	17	29	0	11	23	35
2	14	26	38	8	20	32
4	16	28	40	10	22	34
6	18	30	0	12	24	36

ดังนั้นจะได้ Interleaver ที่มีการสลับบิตดังนี้ [1 13 25 37 7 19 31 3 15 27 39
9 21 33 5 17 29 11 23 35 2 14 26 38 8 20 32 4 16 28 40
10 22 34 6 18 30 12 24 36]

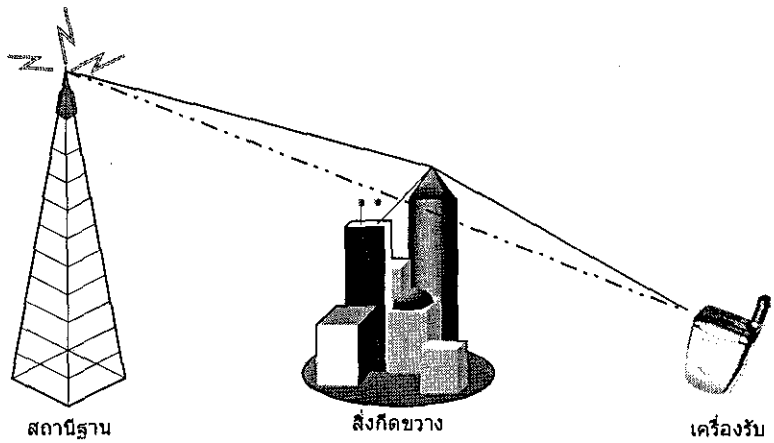
บทที่ 3
ทฤษฎีของแบบจำลองช่องสัญญาณ

ในโครงงานนี้สนใจศึกษาการเข้ารหัสแบบ Turbo Code เพื่อนำไปใช้ในระบบโทรศัพท์เคลื่อนที่ ซึ่งต่างจากการสื่อสารระบบมีสายที่สัญญาณไม่สามารถส่งตรงจากต้นทางถึงปลายทางเสมอได้ไป และยังมีผลกระทบจากสิ่งแวดล้อม ดั้งนั้นโครงงานนี้จึงทำการศึกษาและสร้างแบบจำลองช่องสัญญาณ เพื่อจำลองผลกระทบที่เกิดขึ้นกับสัญญาณให้คล้ายกับสถานะการจริงมากที่สุด

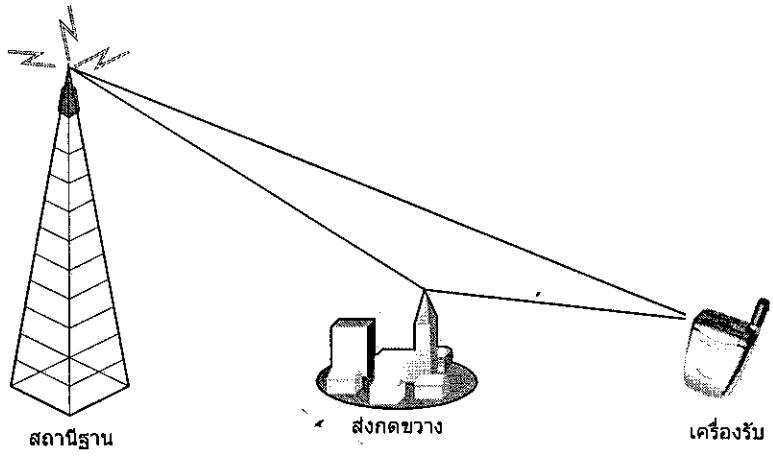
3.1 ทฤษฎีโทรศัพท์เคลื่อนที่เบื้องต้น

3.1.1 เส้นทางการแพร่กระจาย (Propagation path)

การแพร่กระจายสัญญาณของระบบโทรศัพท์เคลื่อนที่ หรือระบบสื่อสารไร้สายระหว่างสถานีฐานและเครื่องรับ ประกอบด้วยช่วงที่สัญญาณระหว่างสถานีฐานและเครื่องรับสื่อสัญญาณกันได้โดยตรง เรียกว่า Line of Sight ดังภาพที่ 3-1 และช่วงที่สัญญาณระหว่างสถานีฐานและเครื่องรับสื่อสัญญาณกันโดยการแพร่กระจาย เรียกว่า Out of Sight ดังภาพที่ 3-2



ภาพที่ 3-1 ภาพแสดงการแพร่กระจายสัญญาณของระบบโทรศัพท์เคลื่อน แบบ Out of Sight



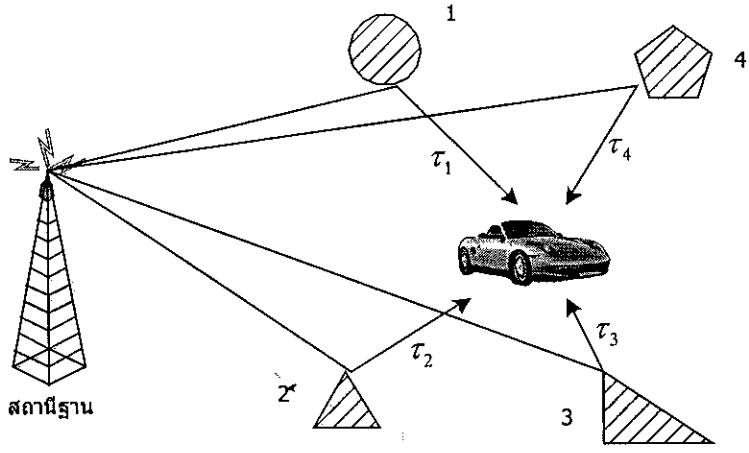
ภาพที่ 3-2 ภาพแสดงการแพร่กระจายสัญญาณของระบบโทรศัพท์เคลื่อน แบบ Line of Sight

3.1.2 Multipath Fading Due to Scattering Factors

Multipath Fading คือสัญญาณที่เป็นข้อมูลชุดเดียวกันที่เครื่องรับสัญญาณรับได้ เกิดเนื่องจากในระบบสื่อสารไร้สายเครื่องรับสัญญาณไม่ได้รับเพียงเฉพาะ Line of Sight หรือเพียงแค่สัญญาณเดียวจากสถานฐาน เพราะสัญญาณเกิดการสะท้อนกับสิ่งแวดล้อม ประกอบกับเครื่องรับสัญญาณย่อมต้องมีการเคลื่อนที่ Multipath Fading เป็นส่วนที่มีความสำคัญที่จำเป็นต้องคำนึงถึงสำหรับการออกแบบระบบโทรศัพท์เคลื่อนที่ หรือระบบสื่อสารไร้สายอื่นๆ การเกิด Multipath Fading สามารถแบ่งได้เป็น 3 ประเภท [9] ได้แก่

3.1.2.1 สาเหตุจากสิ่งแวดล้อม และเครื่องรับสัญญาณไม่เคลื่อนที่

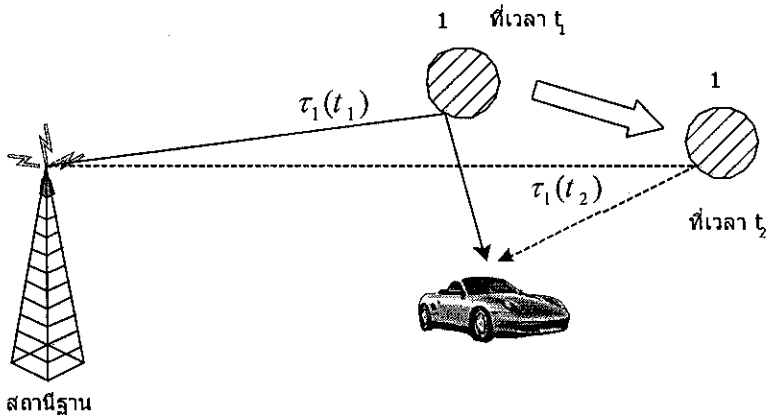
เมื่อทั้งสิ่งแวดล้อม และเครื่องรับสัญญาณไม่เคลื่อนที่ ดังนั้นสัญญาณที่เครื่องรับรับได้จะเป็นเฉพาะสัญญาณที่เกิดจากสิ่งแวดล้อม N จุดเท่านั้น ดังภาพที่ 3-3



ภาพที่ 3-3 ภาพแสดงการเกิด Multipath Fading
เมื่อสิ่งแวดล้อม และเครื่องรับสัญญาณไม่เคลื่อนที่

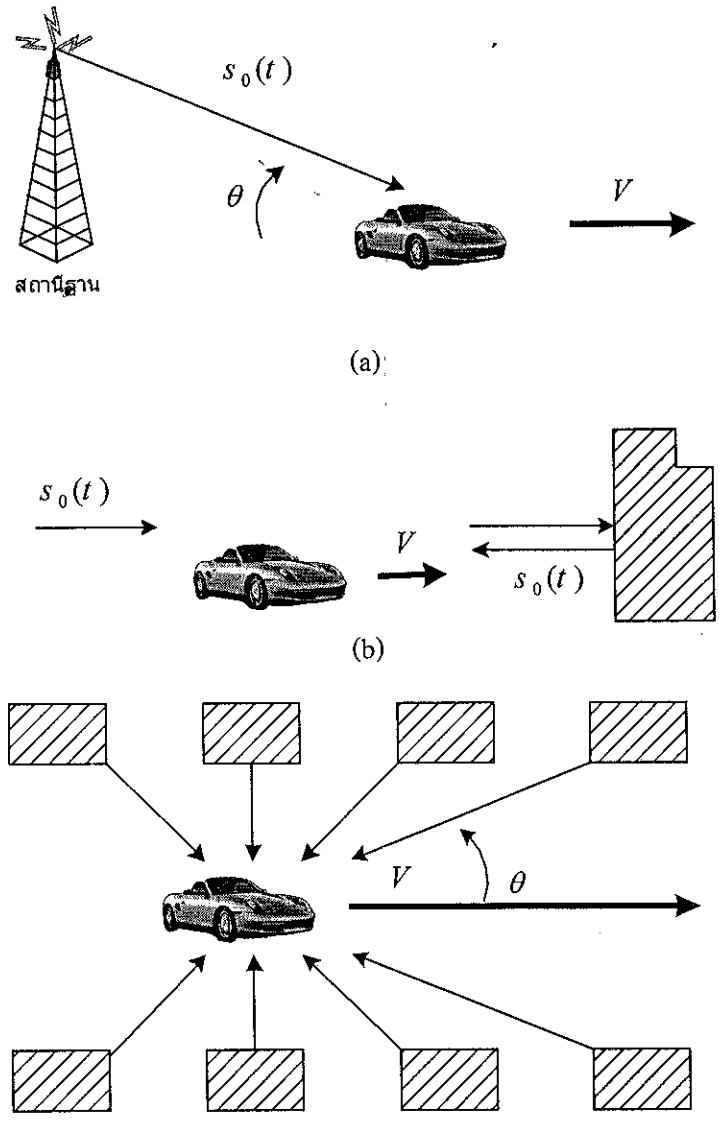
3.1.2.2 สาเหตุจากสิ่งแวดล้อม หรือเครื่องรับสัญญาณเคลื่อนที่

เมื่อเครื่องรับสัญญาณไม่เคลื่อนที่ แต่สิ่งแวดล้อมเกิดการเคลื่อนที่ ดังนั้น สัญญาณที่เครื่องรับรับได้จะถูกลดทอนเนื่องจากความแตกต่างของช่วงเวลา [9] ดังภาพที่ 3-4



ภาพที่ 3-4 ภาพแสดงการเกิด Multipath Fading
เมื่อเครื่องรับสัญญาณไม่เคลื่อนที่ แต่สิ่งแวดล้อมเคลื่อนที่

เมื่อสิ่งแวดล้อมไม่เคลื่อนที่ แต่เครื่องรับสัญญาณเคลื่อนที่ซึ่งสามารถพิจารณาได้เป็น 3 ประเภท คือ พิจารณาเมื่อไม่มีสิ่งแวดล้อมอยู่, พิจารณาเมื่อมีสิ่งแวดล้อมอยู่หน่วยเดียว และ พิจารณาเมื่อมีสิ่งแวดล้อมมากกว่า1หน่วย [9] ดังภาพที่ 3-5 (a), (b) และ (c)



ภาพที่ 3-5 ภาพแสดงการเกิด Multipath Fading
เมื่อสิ่งแวดล้อมไม่เคลื่อนที่ แต่เครื่องรับสัญญาณเคลื่อนที่

3.1.2.3 สาเหตุจากสิ่งแวดล้อม และเครื่องรับสัญญาณเคลื่อนที่

จะพิจารณาทั้งส่วนที่สิ่งแวดล้อมเคลื่อนที่ และส่วนที่เครื่องรับสัญญาณเคลื่อนที่ โดยสัญญาณที่รับได้จะเป็น ผลจากสิ่งแวดล้อมเคลื่อนที่, ผลจากการถูกลดทอนเนื่องจากความแตกต่างของช่วงเวลา และ ผลจาการไม่ได้รับสัญญาณโดยตรง [9]

3.1.3 มาตรฐาน IS-95

IS – 95 เป็นมาตรฐานของโทรศัพท์มือถือ และสถานีฐาน สำหรับ Dual-Mode Wide Band Spread Spectrum มาตรฐานนี้นำมาใช้งานในประเทศอเมริกากับ 800 MHz cellular bands ต่อมาดัดแปลงมาใช้กับ Personal Communications Service (PCS) 1900 MHz มาตรฐานนี้เสนอโดยบริษัท Qualcomm ซึ่งได้รับความร่วมมือจาก AT&T, Motorola และหน่วยงานอื่นที่เกี่ยวข้อง IS-95 เป็น direct-sequence CDMA ที่ผู้ใช้แต่ละคนจะมี Pseudorandom Code ที่ไม่เหมือนกัน ปี 1988, The Cellular Telecommunications Industry Association (CTIA) กำหนด User's Performance Requirements (UPR) สำหรับ Cellular Carrier's ในระบบ Cellular ยุคต่อไป ซึ่ง IS-95 สามารถตอบสนองต่อข้อกำหนดต่างๆได้ ซึ่งข้อกำหนดต่างๆเหล่านี้ได้แก่

- สะดวกต่อการเปลี่ยนแปลง และสามารถใช้ได้ในระบบแอนะลอก (Analog System)
- รวดเร็วสำหรับการใช้งานและเหมาะสมกับราคาสำหรับ Dual-Mode Radios และ Cells
- มีความจุช่องสัญญาณเพิ่มขึ้นมากกว่าความจุ ในระบบแอนะลอกอย่างมาก
- ความเป็นส่วนตัว
- มีความเพียงพอสามารถรองรับต่อการเติบโตของ Second-Generation Technology
- สามารถปรับปรุงคุณภาพต่างๆ เช่น คุณภาพเสียง, คุณภาพในการบริการในส่วนที่เป็น Dropped Calls, ความคงที่ของระดับคุณภาพเสียง และอื่นๆ
- ง่ายต่อการแสดงความสามารถที่โดดเด่น

ในการใช้งานข้อกำหนดที่สำคัญมากและทำให้ CDMA เป็นที่ยอมรับเพราะมีความยืดหยุ่นในการใช้ งานสูงคือ Dual-Mode Operation ซึ่งสามารถทำได้โดยใช้ CDMA Channel ในช่วงความถี่ของ AMPS เมื่อผู้ใช้เปิดเครื่อง เครื่องลูกข่ายจะหา CDMA Control Channel ถ้าเจอก็จะเริ่มสื่อสารโดยใช้ CDMA Protocol ถ้าไม่เจอก็จะหา AMPS Control Channel และสื่อสารโดยใช้ระบบ AMPS แทน [10]

3.2 แบบจำลองช่องสัญญาณ AWGN

AWGN (Additive White Gaussian Noise) เป็นแบบจำลองช่องสัญญาณที่มีความต่อเนื่องและไม่เป็นฟังก์ชันของเวลา (Continuous Time Invariant) มีความสำคัญมากเพราะใช้เป็นแบบจำลองช่องสัญญาณเพื่อใช้เปรียบเทียบประสิทธิภาพของการเข้ารหัสแบบต่างๆ และเป็นที่ยอมรับอย่างกว้างขวาง

การสร้างแบบจำลอง AWGN จะทำการสุ่มค่าสัญญาณรบกวนเป็น Normally Distributed (Gaussian) ที่มีค่าเฉลี่ยเป็นศูนย์ และมีความแปรปรวน (Variance) $\sigma^2 = \frac{N_0}{2}$ โดยที่ N_0 คือค่าความหนาแน่นพลังงานของสัญญาณรบกวนที่เป็น Single-Sided Band ทำให้ได้ค่าเป็นเวกเตอร์ n เมื่อสัญญาณที่ส่งออกมาเป็นเวกเตอร์ x ดังนั้นสัญญาณที่รับได้ดังสมการที่ 3-1

$$y(t) = x(t) + n(t) \qquad \text{สมการที่ 3-1}$$

เนื่องจากสัญญาณที่ส่งออกมาย่อมต้องมีพลังงาน โดยกำหนดให้ E_s เป็นพลังงานเฉลี่ยเทียบกับที่ชุดข้อมูล (Symbol) ดังนั้นค่าเบี่ยงเบนของสัญญาณรบกวนที่เป็น AWGN เป็นดังสมการที่ 3-2

$$\sigma^2 = \frac{N_0}{2E_s} \qquad \text{สมการที่ 3-2}$$

3.3 แบบจำลองช่องสัญญาณ Rayleigh Fading

Rayleigh Fading เป็นแบบจำลองช่องสัญญาณแสดงคุณลักษณะของสัญญาณวิทยุแถบแคบ (Narrowband Radio Signals) เมื่อสัญญาณประกอบด้วยสัญญาณ Multiple Propagation Environment และความน่าจะเป็นของการกระจายของสัญญาณจะกำหนดใน Gaussain ที่มีค่าเฉลี่ยเป็นศูนย์ คือมีขนาดเป็นผลคูณของ Rayleigh Distribution [11]

การสร้างแบบจำลองช่องสัญญาณ Rayleigh Fading เริ่มจากการคำนวณหาซึ่งลักษณะของช่องสัญญาณ Rayleigh Fading สร้างได้ดังสมการที่ 3-4 ซึ่ง $a(t)$ เป็นสัญญาณรบกวนที่เกิดจากช่องสัญญาณ Rayleigh Fading ω_c เป็นความถี่คลื่นพาห์ในเชิงมุม $T_c(t)$ และ $T_s(t)$ เป็นค่าขนาดของ Fading โดยสุ่มตัวแปรจาก Gaussain ที่มีค่าเฉลี่ยเป็นศูนย์ และพลังงานเฉลี่ยเป็น σ^2 และ [4] โดยที่ค่า $r(t)$ และ $\varphi(t)$ มีค่าดังสมการที่ 3-5 และสมการที่ 3-6

$$a(t) = T_c(t)\cos(\omega_c t) + T_s(t)\sin(\omega_c t) \qquad \text{สมการที่ 3-3}$$

$$a(t) = r(t)\cos(\omega_c t + \varphi(t)) \qquad \text{สมการที่ 3-4}$$

$$r(t) = \sqrt{T_c{}^2(t) + T_s{}^2(t)} \qquad \text{สมการที่ 3-5}$$

$$\varphi(t) = \tan^{-1}\left(\frac{T_s(t)}{T_c(t)}\right) \qquad \text{สมการที่ 3-6}$$

$r(t)$ มีฟังก์ชันการกระจายตัวตามความน่าจะเป็นของ Rayleigh (Rayleigh Probability Density Function) ดังสมการที่ 3-7

$$P(r) = \begin{cases} \frac{r}{\sigma^2} e^{\left(-\frac{r}{2\sigma^2}\right)} & 0 \le r \le \infty \\ 0 \end{cases} \qquad \text{สมการที่ 3-7}$$

3.4 แบบจำลองช่องสัญญาณ Ricean Fading

Ricean Fading เป็นแบบจำลองช่องสัญญาณซึ่งแสดงคุณลักษณะของสัญญาณวิทยุ (Radio Signals) ที่ประกอบด้วยสัญญาณ Line of Sight และสัญญาณ Multiple Non-Direct และความน่าจะเป็นของการกระจายของสัญญาณจะกำหนดใน Gaussain ที่มีค่าเฉลี่ยเป็นศูนย์ โดยมีขนาดเป็นผลคูณของ Rician Ddistribution [11] ซึ่งสามารถสร้างแบบจำลองช่องสัญญาณ Ricean Fading

การสร้างแบบจำลองช่องสัญญาณ Ricean Fading คล้ายกับลักษณะของช่องสัญญาณ Rayleigh Fading เพียงเพิ่มส่วนที่ประกอบด้วยสัญญาณ Line of Sight ดังสมการที่ 3-8 ซึ่ง α เป็นขนาดของ Specular (Line of Sight) component

$$a'(t) = (\alpha + T_c(t))\cos(\omega_c t) + T_s(t)\sin(\omega_c t) \qquad \text{สมการที่ 3-8}$$

3.5 การสร้างแบบจำลองช่องสัญญาณ Rayleigh Fading และ Ricean Fading

1. กำหนดค่าตัวแปรความเร็วโทรศัพท์เคลื่อน (v) ความถี่คลื่นพาห์ (f_c) และจำนวนความถี่หลัก (N, Frequency Domain Point) คำนวณหาค่า Doppler frequency (f_d) ดังสมการที่ 3-9

$$f_d = \frac{v \cdot f_c}{C} \qquad \text{สมการที่ 3-9}$$

2. คำนวณหาค่า ช่วงห่างความถี่ Δf ได้ดังสมการที่ 3-10 เพื่อสร้างเส้นความถี่ (Spectral Line) จำนวน $N-1$ จุด

$$\Delta f = \frac{2f_d}{N-1} \qquad \text{สมการที่ 3-10}$$

3. สุ่มตัวแปรจำนวนเชิงซ้อนที่มีการกระจายตัวเป็น Gaussian จำนวน $N/2$ จุด เป็นแหล่งสัญญาณรบกวนด้านบวก (Positive Frequency Components of the Noise Source) แล้วทำการสังยุคแหล่งสัญญาณรบกวนด้านบวก เป็นแหล่งสัญญาณรบกวนด้านลบ (Negative Frequency Components of the Noise Source)

4. ทำการสุ่มเหมือนขั้นตอนที่ 3 อีกครั้ง

5. นำแหล่งสัญญาณรบกวนจากขั้นตอนที่ 3 คูณกับ Fading Spectrum $\sqrt{S(f)}$ เป็น In-phase Noise Source และนำแหล่งสัญญาณรบกวนจากขั้นตอนที่ 4 คูณกับ Fading Spectrum เป็น Quadrature Noise Source ซึ่งค่า Fading Spectrum $\sqrt{S(f_c)}$ หาได้จากสมการที่ 3-11

$$S(f) = \frac{g}{\pi \cdot f_d \sqrt{1 - \left(\frac{f - f_c}{f_d}\right)^2}} \qquad \text{สมการที่ 3-11}$$

$$R(t) = J_0(2\pi f_d t) \qquad \text{สมการที่ 3-12}$$

6. ทำการ Inverse Fast Fourier Transfrom ค่า In-phase Noise Source และ Quadrature Noise Source ได้ค่า T_c และ T_s

7. คำนวณสัญญาณที่รับได้เนื่องจาก Fading

A. Rayleigh Fading

สัญญาณที่รับได้จะเป็น $r(t)$ ดังสมการที่ 3-13 โดย $a(t)$ คำนวณจากสมการที่ 3-14 และ $n(t)$ เป็น สัญญาณรบกวนที่เป็น AWGN

$$r(t) = a(t) \cdot x(t) + n(t) \qquad \text{สมการที่ 3-13}$$

$$a(t) = \left| T_c + jT_s \right| \qquad \text{สมการที่ 3-14}$$

B. Ricean Fading

สัญญาณที่รับได้จะเป็น $r'(t)$ ดังสมการที่ 3-15 โดย $a'(t)$ คำนวณจากสมการที่ 3-16 และ $n(t)$ เป็น สัญญาณรบกวนที่เป็น AWGN

$$r'(t) = a'(t) \cdot x(t) + n(t)$$ สมการที่ 3-15

$$a'(t) = \left|\alpha + T_c + jT_s\right|$$ สมการที่ 3-16

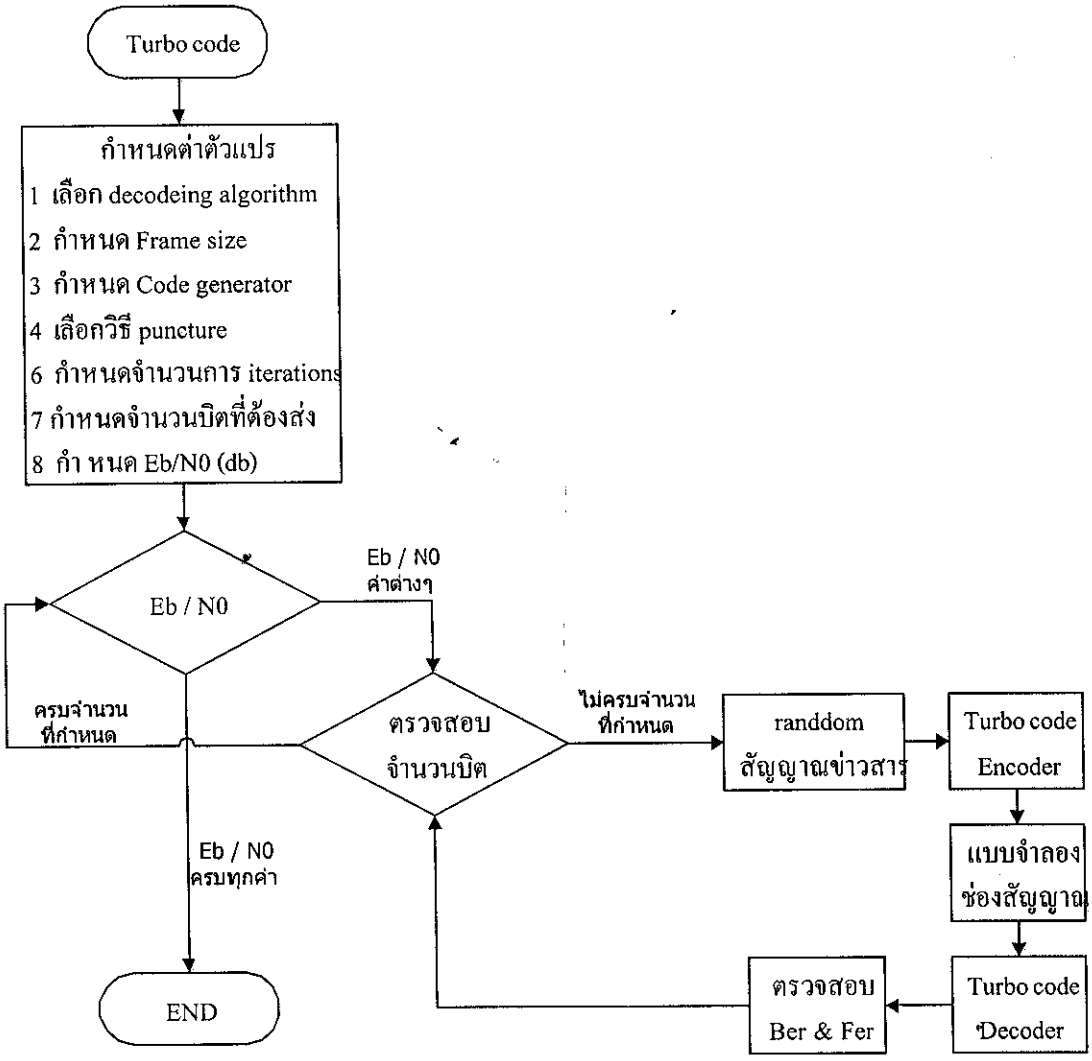
บทที่ 4
การทำงานของโปรแกรม
การ สร้าง และใช้งานเครื่องมือสร้างแบบจำลอง Turbo code

4.1 การทำงานของโปรแกรม Turbo code

เนื่องจากโปรแกรมที่ใช้ทดสอบ Turbo code ซึ่งเป็นโปรแกรมชนิด M-file Function ของ Matlab มีความซับซ้อนมากจึงแสดงลำดับการทำงานในเบื้องต้น เพื่อให้สะดวกต่อการศึกษาการทำงานของโปรแกรม โดยแยกอธิบายเป็นส่วน ไม่ได้อธิบายถึงการทำงานของแต่ละ M-file Function

4.1.1 การทำงานโดยรวมของโปรแกรม

โปรแกรมจะเริ่มทำการเก็บค่าตัวแปรต่างๆ ที่เกี่ยวกับข้อมูลการเข้ารหัส การเลือกกำหนด Interleaver และแบบจำลองช่องสัญญาณ แล้วเริ่มทำงานด้วยค่า Eb / N0 (dB) ทีละค่า (ภายในโปรแกรมเก็บค่าเป็นตัวแปรเมตริก) โดยจากตรวจสอบว่าจำนวนบิตครบจำนวนที่ต้องการส่งหรือไม่ แล้วจึงสุ่มบิตของข่าวสารเป็นบิต 1 และ 0 โดยมีขนาดน้อยกว่าความยาวเฟรม (ซึ่งขึ้นอยู่กับความยาวของ Code Generator) นำไปเข้ารหัสด้วย Turbo Code Encoder ได้เป็นสัญญาณที่มีขนาด 1 และ -1 (เนื่องจากการส่งสัญญาณจริงคิดเป็น BPSK) จากนั้นนำสัญญาณที่เข้ารหัสผ่านแบบจำลองช่องสัญญาณต่างๆซึ่งขึ้นอยู่กับโปรแกรมว่าใช้ช่องสัญญาณแบบใด นำสัญญาณที่ผ่านช่องสัญญาณมาถอดรหัสด้วย Turbo Code Decoder ได้ผลเป็นบิตข่าวสารที่รับได้เป็นบิต 1 และ 0 ทำการเปรียบเทียบระหว่างบิตที่ส่งออกมา และบิตที่รับได้เก็บเป็นค่าตัวแปร BER และ FER แล้วจึงเริ่มตรวจสอบว่าจำนวนบิตครบจำนวนที่ต้องการส่งอีกครั้ง เมื่อจำนวนบิตครบจำนวนที่ต้องการส่งแล้ว จึงเริ่มการทำงานที่ Eb / N0 (dB) ค่าใหม่จนครบทุกค่า ดังภาพที่ 4-1

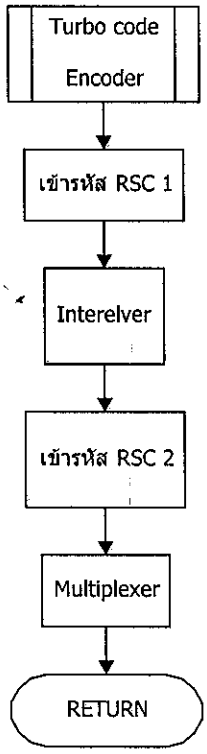


ภาพที่ 4-1 ภาพแสดงการทำงานของโปรแกรมโดยรวม

4.1.2 การทำงาน Turbo Code Encoder

โปแกรมส่วนที่เป็นการเข้ารหัสจะทำการเข้ารหัส โดยใช้ Code Generator ที่กำหนดไว้ แล้วทำการเพิ่มบิตเพื่อให้ สถานะของการเข้ารหัสสุดท้ายเหมือนกับสถานะเริ่มต้น ซึ่งจะได้ความยาวเป็นจำนวนเท่ากับความยาวเฟรม โดยจะได้ผลลัพธ์เป็นข้อมูลที่เพิ่มจำนวนบิต กับข้อมูลจากการเข้ารหัสแบบ RSC ซึ่งเป็น Parity Check Bit ชุดแรก จากนั้นนำข้อมูลที่เพิ่มจำนวนบิตผ่านการสลับบิตจาก Interleaver ที่กำหนดไว้ แล้วนำไปเข้ารหัสแบบ RSC อีกครั้ง ได้ผลลัพธ์เป็น Parity

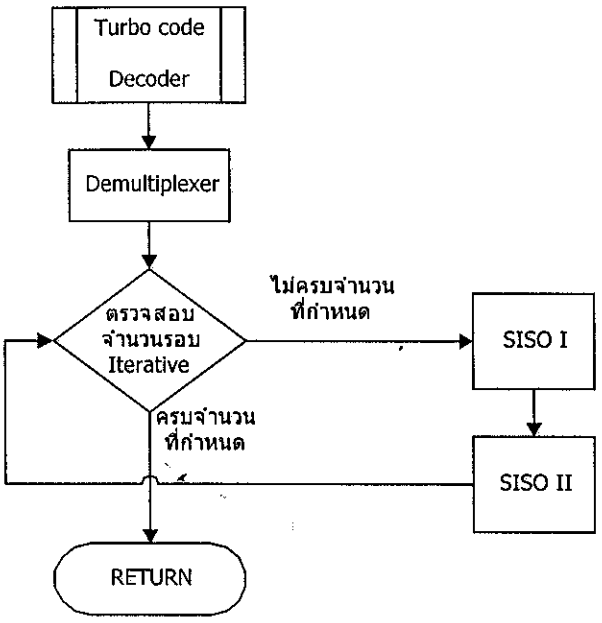
Check Bit ชุดที่สอง แล้วทำการรวมข้อมูลทั้งสามชุดเป็นข้อมูลอนุกรมชุดเดียวกันด้วย Multiplexing โดยมี Puncturing เป็นตัวกำหนดจำนวน Parity Check Bit ที่จะส่งออก ดังภาพที่ 4-2



ภาพที่ 4-2 ภาพแสดงการทำงานของโปรแกรม Turbo Code Encoder

4.1.3 การทำงาน Turbo Code Decoder

โปแกรมส่วนที่เป็นการเข้ารหัสจะทำการนำข้อมูลที่รับมาได้แยกเป็น ข้อมูลข่าวสาร และ Parity Check Bit 2 ชุด ด้วย Demultiplexer นำข้อมูลที่เป็น ข้อมูลข่าวสาร Parity Check Bit ชุดแรก และ ผลที่การถอดรหัสจาก SISO II Decoder (การ Iterative ครั้งแรกยังไม่ได้ค่าดังกล่าว จึงกำหนดเป็นค่าศูนย์) ผ่าน SISO I Decoder ได้ผลลัพธ์ที่ได้ ข้อมูลข่าวสารที่ผ่านการสลับบิต และ Parity Check Bit ชุดที่สอง ผ่าน SISO II Decoder จากนั้นปฏิบัติเหมือนเดิมในส่วนที่เป็น SISO I Decoder และ SISO II Decoder จนครบจำนวน Iterative จึงได้ข้อมูลที่ทำการถอดรหัส ดังภาพที่ 4-2



ภาพที่ 4-3 ภาพแสดงการทำงานของโปรแกรม Turbo Code Decoder

4.2 การสร้างเครื่องมือสร้างแบบจำลอง Turbo code

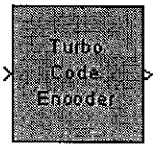
4.2.1 การสร้างเครื่องมือสร้างข้อมูลสำหรับเข้ารหัส

Generated Source เป็นเครื่องมือสำหรับสร้างข้อมูลที่มีบิตเป็น 0 และ 1 เพื่อใช้ทำการเข้ารหัส โดยภายในประกอบด้วย S-Function ที่ทำการเรียกฟังก์ชัน J_gen.m ที่ทำการสุ่มค่าข้อมูลจากฟังก์ชันใน Matlab ดังภาพที่ 4-4

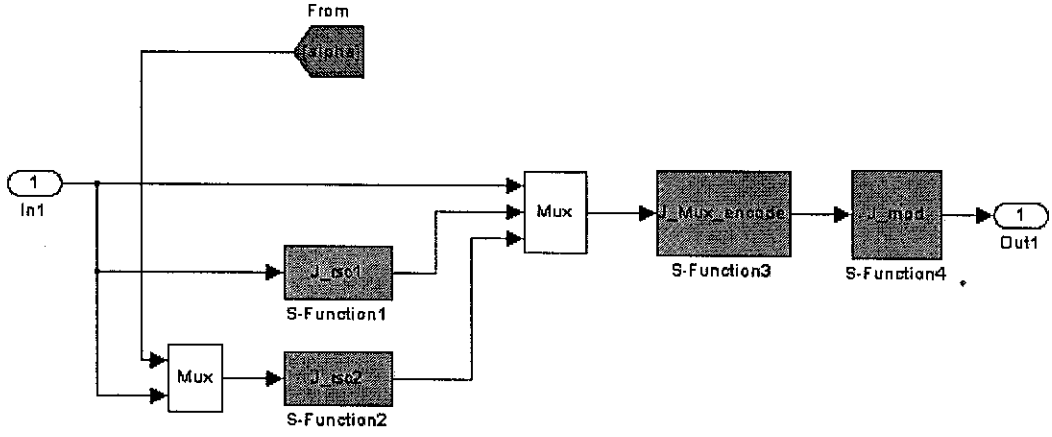
ภาพที่ 4-4 ภาพแสดง Generated Source และส่วนประกอบภายใน

4.2.2 การสร้างเครื่องมือ Turbo Code Encoder

Turbo Code Encoder (ภาพที่ 4-5) เป็นเครื่องมือสำหรับเข้ารหัส Turbo Code โดยภายในประกอบด้วย S-Function ที่ทำการเรียกฟังก์ชัน J_res1.m, J_res2.m เพื่อเข้ารหัสข้อมูลแบบ RSC แล้วนำผลลัพธ์การเข้ารหัสที่ได้ มารวมกันจากข้อมูลขนานกันมาให้เป็นข้อมูลที่อนุกรมกันด้วย Puncturing โดยส่วนที่เป็น J_mux_encode.m แล้วทำการมอดูเลตเป็นสัญญาณขนาด +1 / -1 ด้วย J_mod ดังภาพที่ 4-6



ภาพที่ 4-5 ภาพแสดงเครื่องมือเข้ารหัส Turbo Code

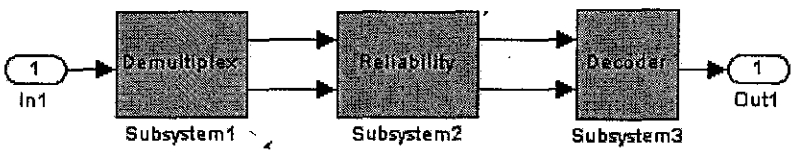


ภาพที่ 4-6 ภาพแสดง Turbo Code

4.2.3 การสร้างเครื่องมือ Turbo Code Decoder

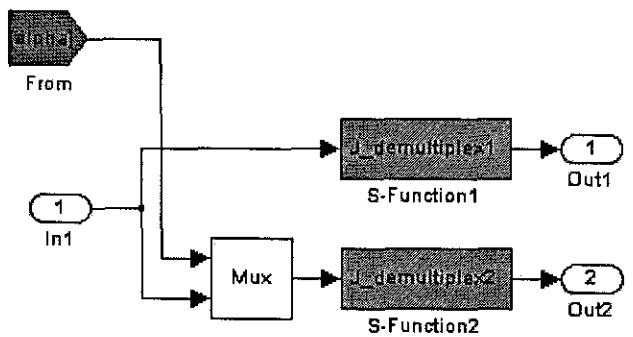
Turbo Code Decoder (ภาพที่ 4-7) เป็นเครื่องมือสำหรับเข้าถอดรหัส Turbo Code โดยเมื่อรับข้อมูลเข้ามาได้แล้วจะผ่านเครื่องมือที่เป็น Demultiplex, Reliability และ Decoder ดังภาพที่ 4-8

ภาพที่ 4-7 ภาพแสดงเครื่องมือถอดรหัส Turbo Code



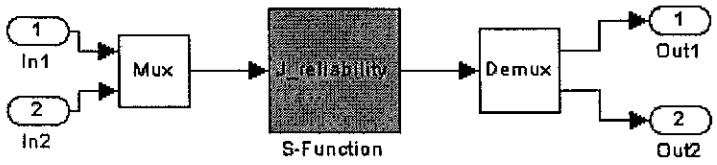
ภาพที่ 4-8 ภาพแสดงส่วนประกอบภายในของเครื่องมือถอดรหัส Turbo Code

Demultiplex จะทำการแยกข้อมูลที่รับได้เป็น 2 ชุด ซึ่งประกอบด้วย ข้อมูลข่าวสารกับข้อมูลที่เป็น Parity Check Bit ชุดแรก และข้อมูลข่าวสารที่ผ่านการสลับบิตกับข้อมูลที่เป็น Parity Check Bit ชุดที่สอง โดยอาศัย S-Function ที่ทำการเรียกฟังก์ชัน J_demultiplex1.m และ J_demultiplex2.m ดังภาพที่ 4-9



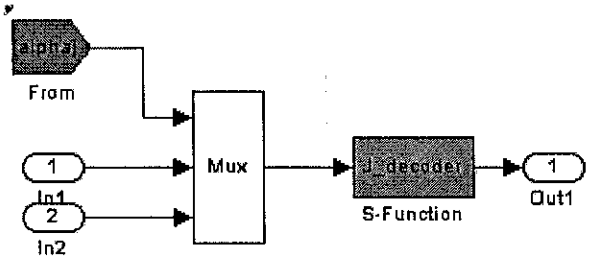
ภาพที่ 4-9 ภาพแสดงส่วนประกอบภายในของ Demultiplex

Reliability เป็นเครื่องมือที่เป็น Channel Reliability ซึ่งคำนวณจาก Eb/No (dB) เพื่อเป็นค่าระดับในการรับสัญญาณสำหรับ SISO Decoder ในการคำนวณหา Log-Likelihood Ratio โดยจะนำค่าทั้งหมด คูณกับระดับที่คำนวณได้ โดยอาศัย S-Function ที่ทำการเรียกฟังก์ชัน J_reliability.m ดังภาพที่ 4-10



ภาพที่ 4-10 ภาพแสดงส่วนประกอบภายในของ Reliability

Decoder เป็นเครื่องมือที่เป็นส่วนที่รับข้อมูลที่ผ่าน Demultiplex และ Reliability มาทำการถอดรหัสข้อมูลแบบ Turbo Code โดยอาศัย S-Function ที่ทำการเรียกฟังก์ชัน J_decoder.m ดังภาพที่ 4-11



ภาพที่ 4-11 ภาพแสดงส่วนประกอบภายในของ Decoder

4.2.4 การสร้างเครื่องมือสำหรับ Interleaver

เนื่องจากโครงงานทำการศึกษาเกี่ยวกับ Interleaver และได้มีการคิด Interleaver ด้วยวิธีการวางสลับบิตแบบทั้งสลับทั้งแถวและหลัก ขึ้นจึงต้องสร้างเครื่องมือสำหรับ Interleaver เพื่อใช้ในการเข้ารหัสแบบ Turbo Code

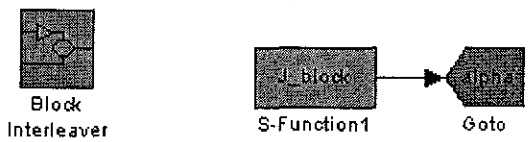
4.3.4.1 Pseudo-Random Interleaver

Pseudo-Random Interleaver เป็นเครื่องมือสร้าง Interleaver ที่มีการสุ่มแบบแบบ Pseudo โดยภายในประกอบด้วย S-Function ที่ทำการเรียกฟังก์ชัน J_alpha.m แล้วส่งค่าลำดับการ Interleaver ไปยังตัวแปร alpha ซึ่งสามารถเรียกใช้ได้ (เป็นตัวแปรชนิด global) ดังภาพที่ 4-12

ภาพที่ 4-12 ภาพแสดง Pseudo-Random Interleaver และส่วนประกอบภายใน

4.2.4.2 Block Interleaver

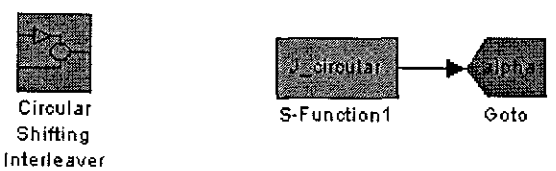
Block Interleaver เป็นเครื่องมือสร้าง Interleaver ที่มีสลับลำดับการเรียงบิตแบบ Block โดยภายในประกอบด้วย S-Function ที่ทำการเรียกฟังก์ชัน J_block.m แล้วส่งค่าลำดับการ Interleaver ไปยังตัวแปร alpha ซึ่งสามารถเรียกใช้ได้ ดังภาพที่ 4-13



ภาพที่ 4-13 ภาพแสดง Block Interleaver และส่วนประกอบภายใน

4.2.4.3 Circular Shifting Interleaver

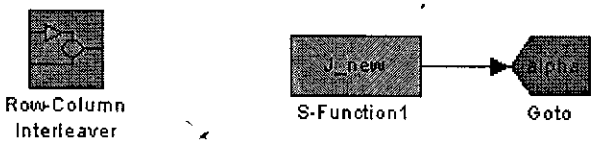
Circular Shifting Interleaver เป็นเครื่องมือสร้าง Interleaver ที่มีสลับลำดับการเรียงบิตแบบ Circular Shifting โดยภายในประกอบด้วย S-Function ที่ทำการเรียกฟังก์ชัน J_circular.m แล้วส่งค่าลำดับการ Interleaver ไปยังตัวแปร alpha ซึ่งสามารถเรียกใช้ได้ ดังภาพที่ 4-14



ภาพที่ 4-14 ภาพแสดง Circular Shifting Interleaver และส่วนประกอบภายใน

4.2.4.4 Row-Column Interleaver

Row-Column Interleaver เป็นเครื่องมือสร้าง Interleaver ที่มีสลับลำดับการเรียงบิตแบบ โดยภายในประกอบด้วย S-Function ที่ทำการเรียกฟังก์ชัน J_new.m แล้วส่งค่าลำดับการ Interleaver ไปยังตัวแปร alpha ซึ่งสามารถเรียกใช้ได้ ดังภาพที่ 4-15

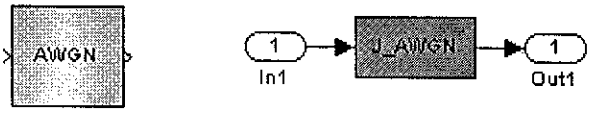


ภาพที่ 4-15 ภาพแสดง และส่วนประกอบภายใน

4.2.5 การสร้างเครื่องมือสำหรับแบบจำลองช่องสัญญาณ

4.2.5.1 AWGN Channel Model

AWGN Channel Model เป็นเครื่องมือสร้างแบบจำลองแบบ AWGN โดยภายในประกอบด้วย S-Function ที่ทำการเรียกฟังก์ชัน J_AWGN.m ซึ่งทำหน้าที่นำสัญญาณที่รับเข้ามาบวกกับสัญญาณรบกวนที่มีการกระจายตัวแบบ Gaussian ที่มีค่าเฉลี่ยเป็นศูนย์ และมีค่าเบี่ยงเบน $\sigma^2 = \frac{N_0}{2E_s}$ ดังภาพที่ 4-16

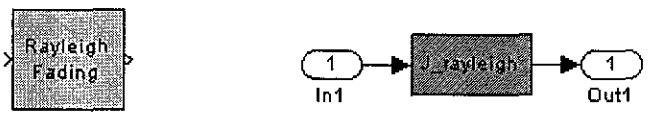


ภาพที่ 4-16 ภาพแสดง AWGN Channel Model และส่วนประกอบภายใน

4.2.5.2 Rayleigh Fading Channel Model

Rayleigh Fading Channel Model เป็นเครื่องมือสร้างแบบจำลองแบบ Rayleigh Fading โดยภายในประกอบด้วย S-Function ที่ทำการเรียกฟังก์ชัน J_Rayleigh.m ซึ่งทำหน้าที่นำ

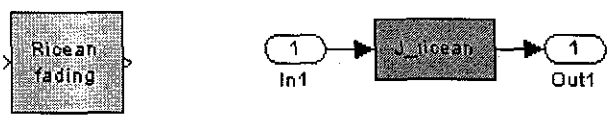
สัญญาณที่รับเข้ามาคูณกับสัญญาณที่เป็น Rayleigh Fading แล้วบวกกับสัญญาณรบกวนที่มีการกระจายตัวแบบ Gaussian ที่มีค่าเฉลี่ยเป็นศูนย์ และมีค่าเบี่ยงเบน $\sigma^2 = \frac{N_0}{2E_s}$ ดังภาพที่ 4-17



ภาพที่ 4-17 ภาพแสดง Rayleigh Fading Channel Model และส่วนประกอบภายใน

4.2.5.3 Ricean Fading Channel Model

Ricean Fading Channel Model เป็นเครื่องมือสร้างแบบจำลองแบบ Ricean Fading โดยภายในประกอบด้วย S-Function ที่ทำการเรียกฟังก์ชัน J_ Ricean.m ซึ่งทำหน้าที่นำสัญญาณที่รับเข้ามาคูณกับสัญญาณที่เป็น Ricean Fading แล้วบวกกับสัญญาณรบกวนที่มีการกระจายตัวแบบ Gaussian ที่มีค่าเฉลี่ยเป็นศูนย์ และมีค่าเบี่ยงเบน $\sigma^2 = \frac{N_0}{2E_s}$ ดังภาพที่ 4-18

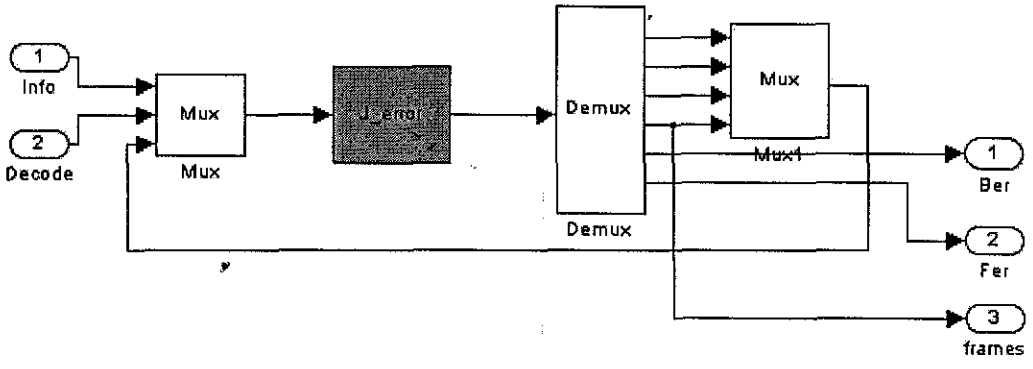


ภาพที่ 4-18 ภาพแสดง Ricean Fading Channel Model และส่วนประกอบภายใน

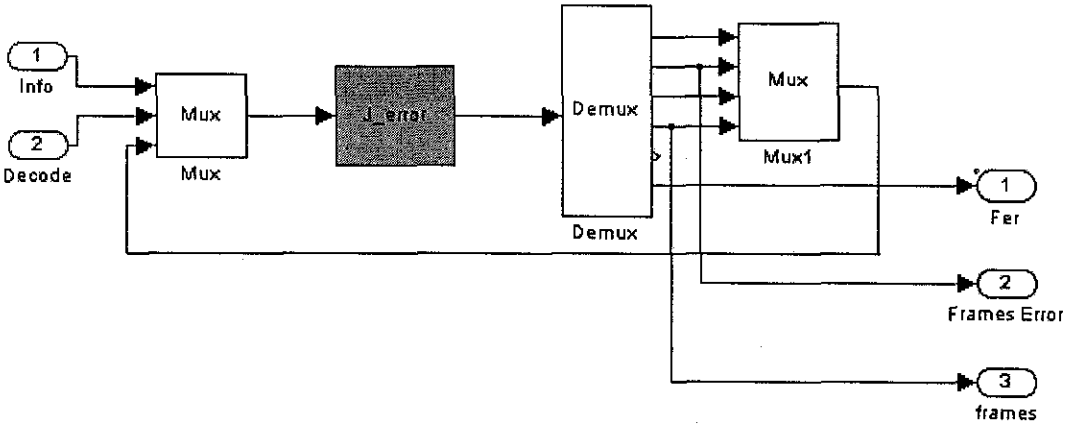
4.2.6 การสร้างเครื่องมือสำหรับเครื่องวัดความผิดพลาด

เครื่องวัดความผิดพลาดจะทำการรับข้อมูลก่อนทำการเข้ารหัส และข้อมูลหลังจากเข้ารหัสผ่านแบบจำลองช่องสัญญาณแล้วถอดรหัสกลับ มาเปรียบเทียบกันเพื่อหาค่า BER และ FER โดยภายในประกอบด้วย S-Function ที่ทำการเรียกฟังก์ชัน J_error.m แล้วส่งค่าลำดับการ Interleaver ไปยังตัวแปร alpha ซึ่งสามารถเรียกใช้ได้ ดังภาพที่ 4-19 ซึ่งเครื่องมือแต่ละชนิดจะทำการเลือกค่าต่างกันออกมา ดังภาพที่ 4-20 ถึง ภาพที่ 4-22

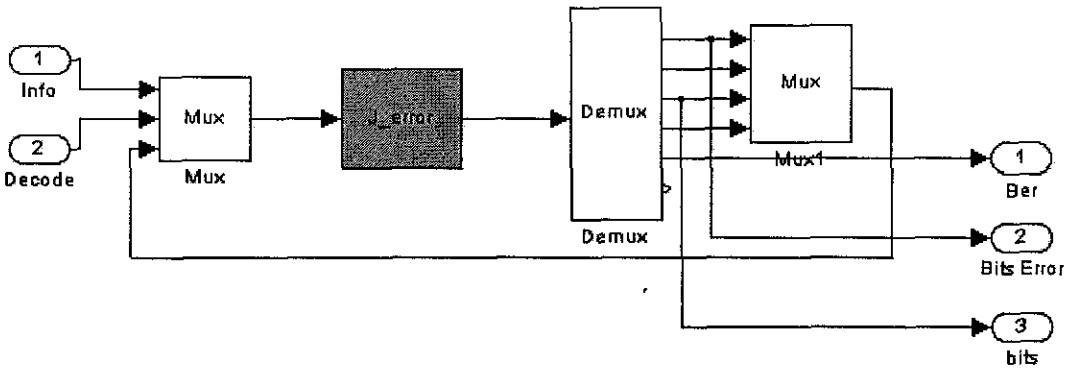
ภาพที่ 4-19 ภาพแสดงเครื่องมือสำหรับเครื่องวัดความผิดพลาด



ภาพที่ 4-20 ภาพแสดงส่วนประกอบภายในของเครื่องมือสำหรับเครื่องวัดความผิดพลาด (1)



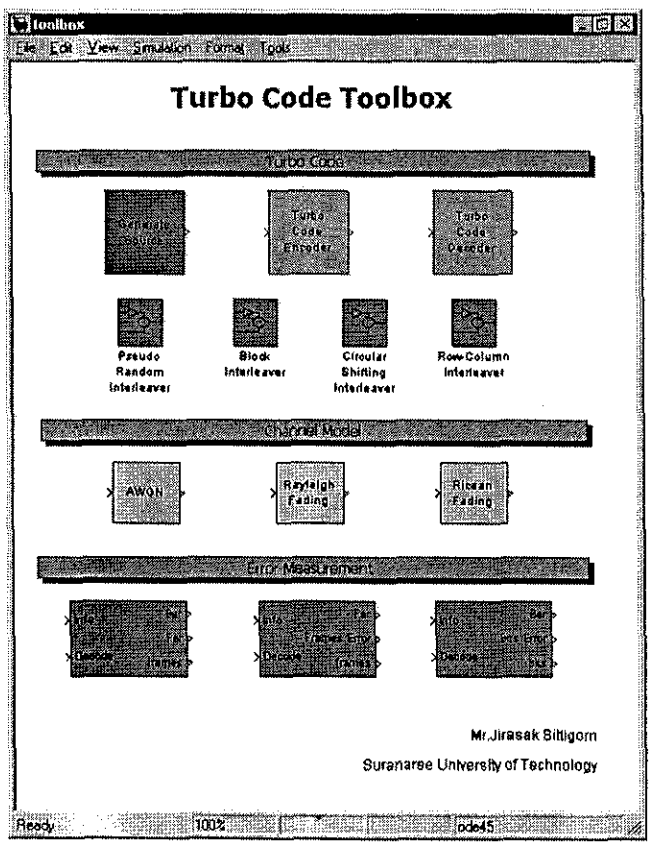
ภาพที่ 4-21 ภาพแสดงส่วนประกอบภายในของเครื่องมือสำหรับเครื่องวัดความผิดพลาด (2)



ภาพที่ 4-22 ภาพแสดงส่วนประกอบภายในของเครื่องมือสำหรับเครื่องวัดความผิดพลาด (3)

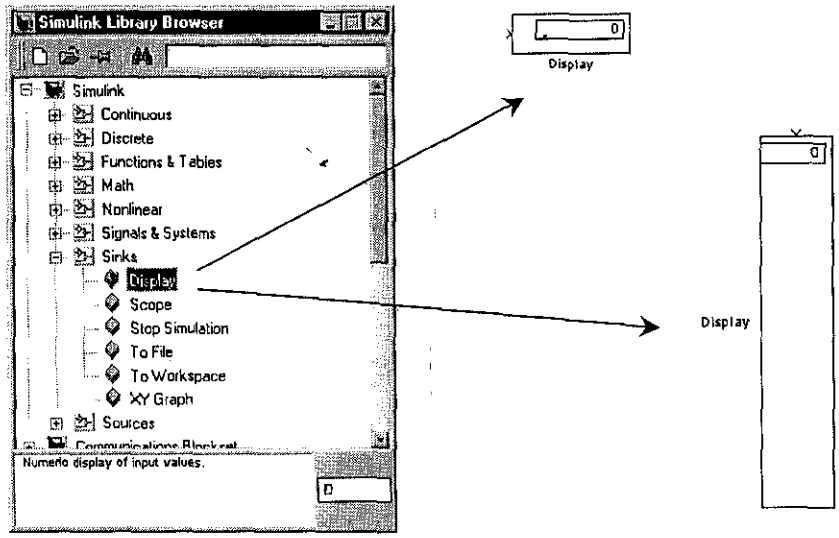
4.4 การใช้งานเครื่องมือสร้างแบบจำลอง Turbo Code

การใช้งานเครื่องมือสร้างแบบจำลอง Turbo Code จำเป็นต้องเลือก เครื่องมือที่ใช้งานให้ครบต่อเครื่องมือแต่ละชนิดให้ถูกต้อง และกำหนดค่าต่างๆในเครื่องมือให้ถูกต้อง



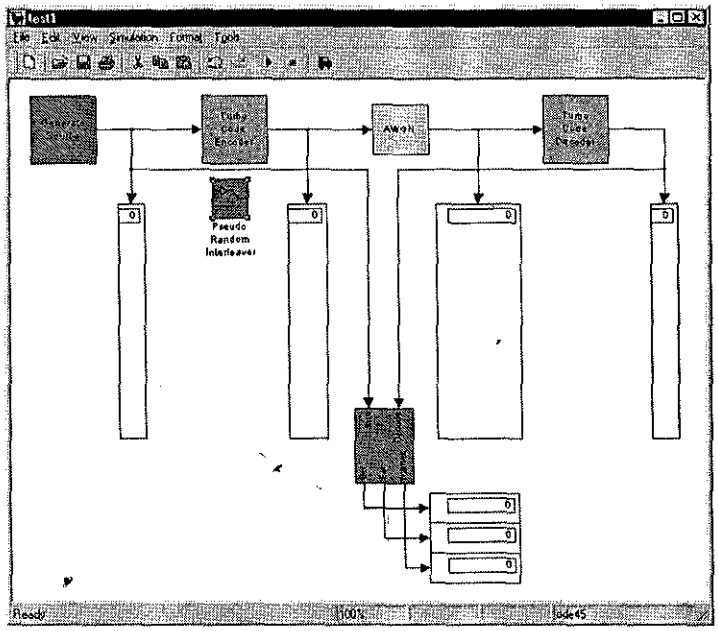
ภาพที่ 4-23 ภาพแสดงส่วนประกอบเครื่องมือสร้างแบบจำลอง Trubo Code

ขั้นตอนที่ 1 การเลือกเครื่องมือสร้างแบบจำลอง Turbo Code จำเป็นต้องมี Generate Source, Turbo Code Encoder, Turbo Code Decoder, Interleaver ชนิดใดชนิดหนึ่ง และแบบจำลองช่องสัญญาณ และควรจะมี Error Measurement และ Display (จาก Simulink Library => Simulink => Sinks => Display หรือ ภาพที่ 4-24) เพื่อแสดงค่าต่างๆ



ภาพที่ 4-23 ภาพแสดงการใช้ Display

ขั้นตอนที่ 2 เริ่มสร้างแบบจำลองใหม่ (โดย File => New => Model หรือ ใช้แป้น Ctrl + N) แล้วนำส่วนประกอบต่างๆ ในขั้นที่ 1 มาต่อกันโดยเชื่อมจุดต่างๆ ให้ครบ ดังตัวอย่างภาพที่ 4-33



ภาพที่ 4-23 ภาพแสดงตัวอย่างเครื่องมือสร้างแบบจำลอง Turbo Code

ขั้นตอนที่ 3 กำหนดค่าของเครื่องมือที่ใช้งานให้ถูกต้องสัมพันธ์กัน

Generate Source

- length of data คือ จำนวนข้อมูลที่ทำการสุ่มเพื่อทำการเข้ารหัส มีจำนวน = ความยาวเฟรม - (K-1) บิต โดยที่ K เท่ากับจำนวนความยาวของ Code Generator ใน Turbo Encoder

Turbo Code Encoder

- ความยาวเฟรม เป็นความยาวข้อมูลข่าวสาร (จำนวนบิตต่อเฟรม ในการส่งข้อมูล)
- Code Generator เป็นการกำหนด RSC Code
- unpunctured เป็นตัวกำหนดจำนวนข้อมูลที่ส่งออก

  unpunctured จะให้ข้อมูล rate 1/3 คือมีความยาวเป็น 3 เท่าของ ความยาวเฟรม

  punctured (ไม่เลือก unpunctured) จะให้ข้อมูล rate 1/2 คือมีความยาวเป็น 2 เท่าของ

Turbo Code Decoder

- ความยาวเฟรม กำหนดให้เหมือนใน Turbo Code Encoder
- Code Generator เป็นการกำหนด RSC code เพื่อทำการถอดรหัส กำหนดให้เหมือนใน Turbo Code Encoder

- Number of Iterations เป็นการกำหนดจำนวน Iterative
- Fading amplitude กำหนดให้เป็น 1
- Eb/N0 (in dB) สำหรับใช้ใน reliability
- Decoding algorithm เลือก algorithm ในการถอดรหัส
- unpunctured กำหนดให้เหมือนใน Turbo Code Encoder

Pseudo Random Interleaver

- ความยาวเฟรม กำหนดให้เหมือนใน Turbo Code Encoder

Block Interleaver

- ความยาวเฟรม กำหนดให้เหมือนใน Turbo Code Encoder
- m และ n เป็นค่าในการกำหนด Block Interleaver โดยมีความสัมพันธ์ จำนวน ความยาวเฟรม = m * n

Circular Shifting Interleaver

- ความยาวเฟรม กำหนดให้เหมือนใน Turbo Code Encoder
- k และ v เป็นค่าในการกำหนด Circular Shifting Interleaver

Row-Column Interleaver

- ความยาวเฟรม กำหนดให้เหมือนใน Turbo Code Encoder

AWGN

- Eb/N0 (in dB) แสดงถึงกำลังในการส่งข่อมูล กำหนดให้เหมือนใน Turbo Code Decoder
- unpunctured กำหนดให้เหมือนใน Turbo Code Encoder

Rayleigh Fading

- mobile speeds (km/hr) กำหนดความเร็วของเครื่องรับสัญญาณ
- carry frequency กำหนดความถี่พาห์
- Eb/N0 (in dB) กำหนดให้เหมือนใน Turbo Code Encoder

- unpunctured กำหนดให้เหมือนใน Turbo Code Encoder

Ricean Fading

- Specular (Line of Sight) component กำหนดส่วนประกอบของสัญญาณ Line of Sight (น้อยกว่าหรือเท่ากับ 1)
- mobile speeds (km/hr) กำหนดความเร็วของเครื่องรับสัญญาณ
- carry frequency กำหนดความถี่พาห์
- Eb/N0 (in dB) กำหนดให้เหมือนใน Turbo Code Encoder
- unpunctured กำหนดให้เหมือนใน Turbo Code Encoder

Error Measurement

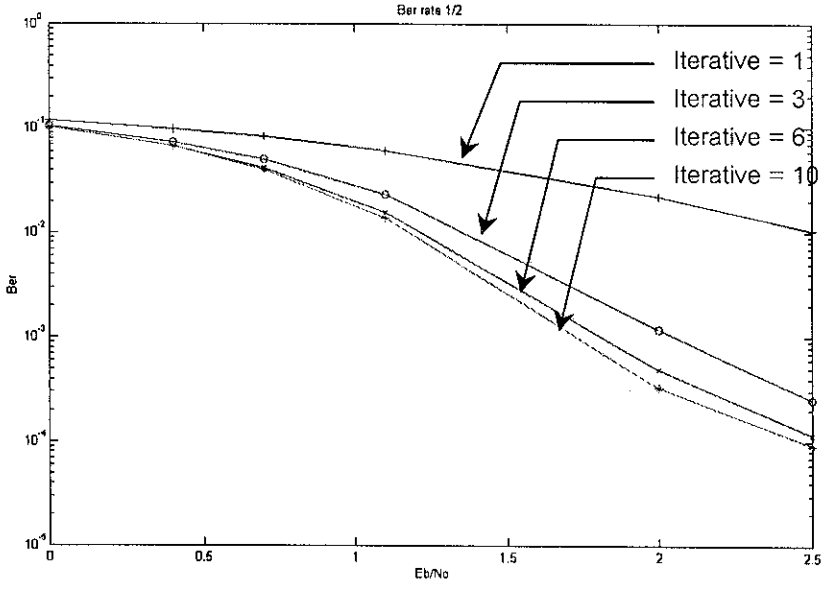
- Length of Information กำหนดให้เหมือน length of data ใน Generate Source เพื่อใช้คำนวณ BER และ FER

บทที่ 5
วิเคราะห์ผล

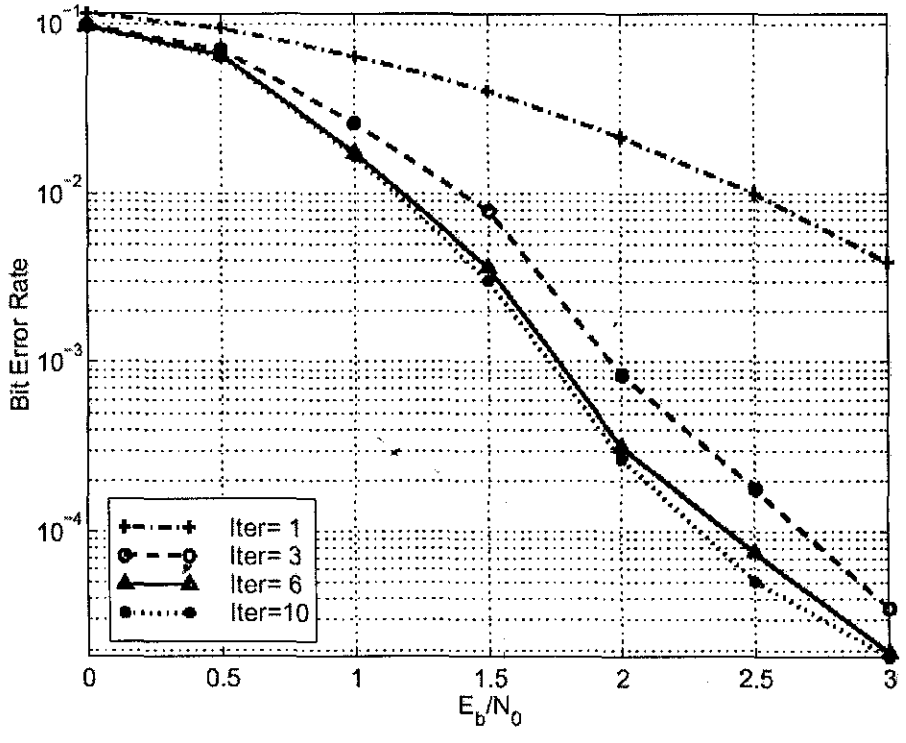
5.1 การเปรียบเทียบผลที่ได้จากโปรแกรม

5.1.1 การเปรียบเทียบผลของโปรแกรม Turbo Code บนแบบจำลองช่องสัญญาณ AWGN

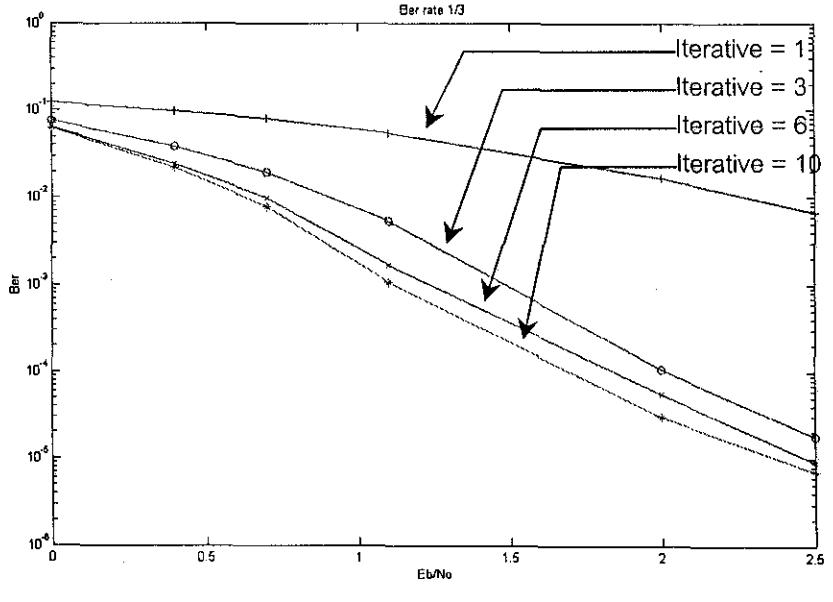
เพื่อเป็นการทดสอบโปรแกรมจึงนำผลจากโปรแกรมมาเปรียบเทียบกับงานวิจัยอื่นๆ สำหรับ ผลการทดสอบโปรแกรม Turbo Code จะเปรียบเทียบ BER ที่ Iterative 1, 3, 6 และ 10 โดยใช้ G = (1 1 1 , 1 0 1), ความยาวเฟรม 400 บิต, ที่ rate = 1/2 และ rate = 1/3 บนแบบจำลองช่องสัญญาณ AWGN เปรียบเทียบกับผลการทดสอบของ Yufei Wu [6] ดังภาพที่ 5-1 ถึง ภาพที่ 5-4



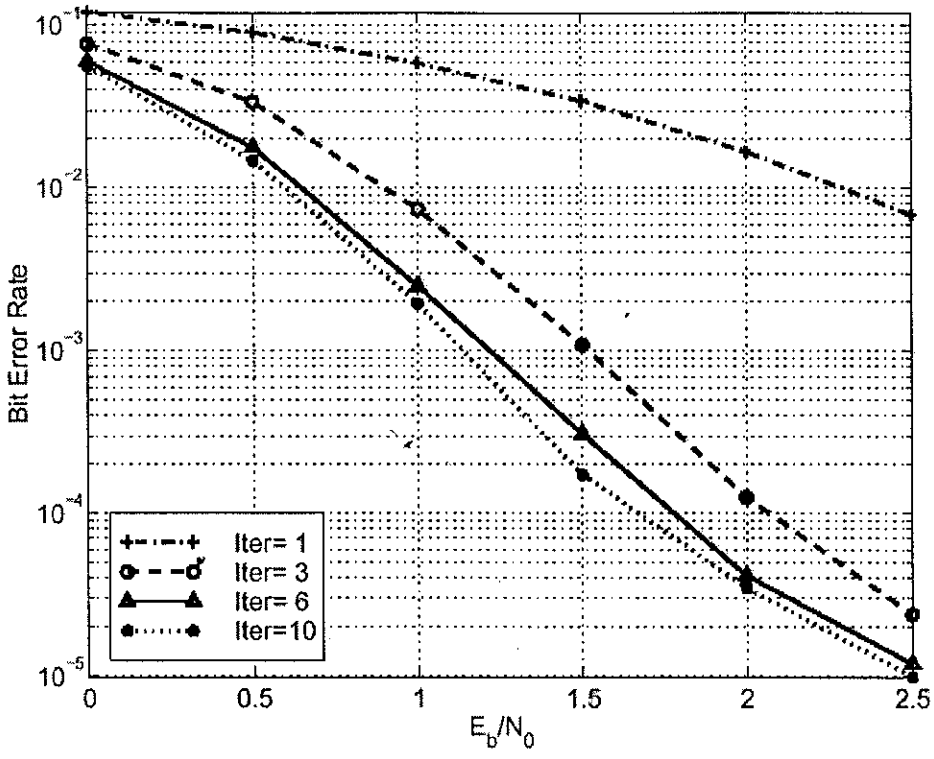
ภาพที่ 5-1 ภาพแสดงผลการทดสอบโปรแกรม Turbo Code เทียบระหว่าง BER กับ Eb/N0 โดยใช้ความยาวเฟรม 400 บิต, G = (1 1 1 , 1 0 1), ที่ rate = 1/2



ภาพที่ 5-2 ภาพแสดงผลการทดสอบ Turbo Code ของ Yufei Wu โดยเทียบ
ระหว่าง BER กับ Eb/N0 โดยใช้ความยาวเฟรม 400 บิต, G = (1 1 1 , 1 0 1), ที่ rate = 1/2



ภาพที่ 5-3 ภาพแสดงผลการทดสอบโปรแกรม Turbo Code เทียบระหว่าง BER กับ Eb/N0
โดยใช้ความยาวเฟรม 400 บิต, G = (1 1 1 , 1 0 1), ที่ rate = 1/3

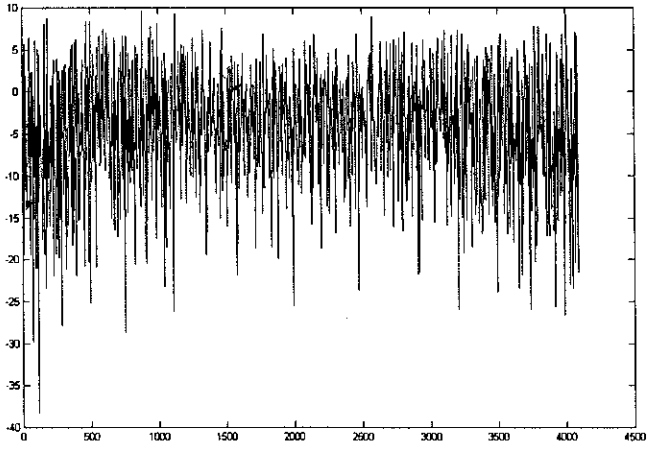


ภาพที่ 5-4 ภาพแสดงผลการทดสอบ Turbo Code ของ Yufei Wu โดยเทียบ
ระหว่าง BER กับ Eb/N0 โดยใช้ ความยาวเฟรม = 400 บิต, G = (1 1 1 , 1 0 1), ที่ rate = 1/3

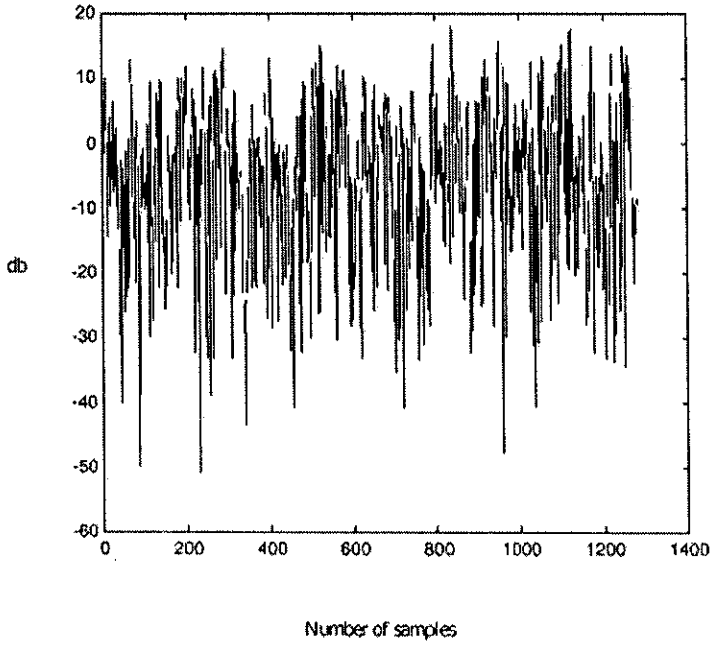
จากผลการทดสอบโปรแกรมในภาพที่ 5-1 ถึง ภาพที่ 5-4 พบว่าผลการเปรียบเทียบ BER ที่ Iterative 1, 3, 6 และ 10 โดยใช้ G = (1 1 1 , 1 0 1), ความยาวเฟรม = 400 บิต, ที่ rate = 1/2 และ rate = 1/3 บนแบบจำลองช่องสัญญาณ AWGN ที่ได้ใกล้เคียงกัน จึงแสดงว่าโปรแกรมที่ใช้มีความถูกต้อง และสามารถนำไปใช้ในการทดลองอื่นๆได้

5.1.2 การเปรียบเทียบผลของโปรแกรมสร้างแบบจำลองช่องสัญญาณ

เพื่อเป็นการทดสอบโปรแกรมสร้างแบบจำลองช่องสัญญาณจึงนำผลจากโปรแกรมมาเปรียบเทียบกับการวิจัยอื่นๆ โดยเปรียบเทียบผลโปรแกรมสร้างแบบจำลองช่องสัญญาณ Rayleigh Fading ที่ความเร็วเครื่องรับ 100 km/hr ความถี่พาห์ 900 MHz กับผลการทดองของ Moataz Mohamed Salah [8] ดังภาพที่ 5-5 และ ภาพที่ 5-6



ภาพที่ 5-5 ภาพแสดงผลการทดสอบโปรแกรมสร้างแบบจำลองช่องสัญญาณ Rayleigh Fading ที่ความเร็วเครื่องรับ 100 km/hr ความถี่พาห์ 900 MHz



ภาพที่ 5-6 ภาพแสดงผลการทดสอบ Rayleigh Fading ของ Moataz Mohamed Salah

จากผลการทดสอบโปรแกรมสร้างแบบจำลองช่องสัญญาณ Rayleigh Fading ที่ความเร็วเครื่องรับ 100 km/hr ความถี่พาห์ 900 MHz กับผลจาก ดังภาพที่ 5-7 และ ภาพที่ 5-8 พบว่าผลที่ได้ใกล้เคียงกัน จึงแสดงว่าโปรแกรมสร้างแบบจำลองช่องสัญญาณ Rayleigh Fading ที่ใช้มีความถูกต้อง

5.2 ความสัมพันธ์ระหว่าง Average Weight กับ Bit Distance ของ Interleaver ที่ความยาวเฟรมต่างกัน

เนื่องจาก Interleaver ที่ต่างกันจะส่งผลให้ประสิทธิภาพของ การเข้ารหัสถอดรหัสของ Turbo Code ต่างกัน และสำหรับ Deterministically Generate Interleaver ที่มีรูปแบบการสลับบิตอย่างแน่นนอน จึงสร้างความสัมพันธ์ระหว่าง Average Weight ($\overline{w}$) กับ Bit Distance (d_b) ของ Interleaver เพื่อในไปเปรียบเทียบกับผลการเข้ารหัสถอดรหัสของ Turbo Code ที่ใช้ Interleaver ต่างกัน

โดยนิยามว่า Bit Distance คือระยะที่บิตห่างจากกันใน Interleaver และ Average Weight คือการเฉลี่ยค่าความแตกต่างของลำดับการ Interleaver ที่ Bit Distance ใดๆ ดังสมการที่ 5-1 โดย N คือความยาวเฟรม และ i คือลำดับบิตที่อยู่ใน Interleaver

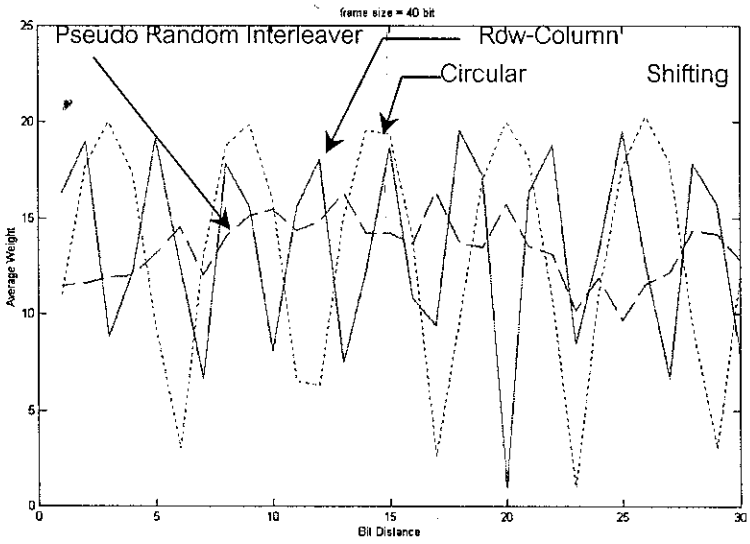
$$\overline{w}(d_b) = \frac{1}{N - d_b} \sum_{i=1}^{N-d_b} dist\,(i, i + d_b) \qquad \text{สมการที่ 5-1}$$

จากนั้นได้นำ Pseudo-Random Interleaver, Circular Shifting Interleaver และ Row-Column Interleaver ที่ความยาวเฟรมต่างกันมาหาความสัมพันธ์ ดังภาพที่ 5-7 ถึงภาพที่ 5-9

ภาพที่ 5-7 แสดงความสัมพันธ์ระหว่าง Average Weight กับ Bit Distance ของ Pseudo Random Interleaver, Circular Shifting Interleaver (k = 7, a = 0) และ Row-Column Interleaver ที่ความยาวเฟรม 40 บิต

ภาพที่ 5-8 ภาพแสดงความสัมพันธ์ระหว่าง Average Weight กับ Bit Distance ของ Pseudo Random Interleaver, Circular Shifting Interleaver (k = 31, a = 0) และ Row-Column Interleaver ที่ความยาวเฟรม 1024 บิต

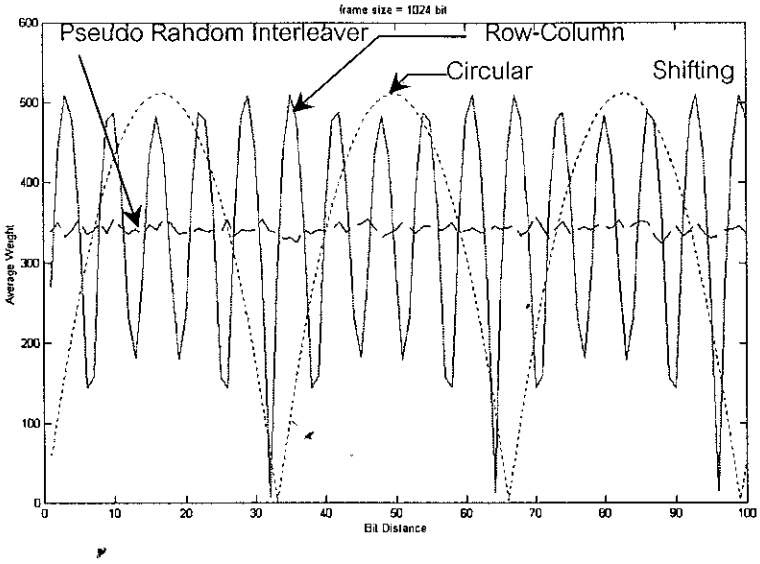
ภาพที่ 5-9 ภาพแสดงความสัมพันธ์ระหว่าง Average Weight กับ Bit Distance ของ Pseudo Random Interleaver, Circular Shifting Interleaver (k = 127, a = 0) และ Row-Column Interleaver ที่ความยาวเฟรม 16384 บิต



ภาพที่ 5-7 ภาพแสดงความสัมพันธ์ระหว่าง Average Weight กับ Bit Distance (ความยาวเฟรม 40 บิต)

จากภาพที่ 5-7 แสดงให้เห็นว่าที่ ความยาวเฟรม 40 บิต มีข้อสังเกตดังนี้

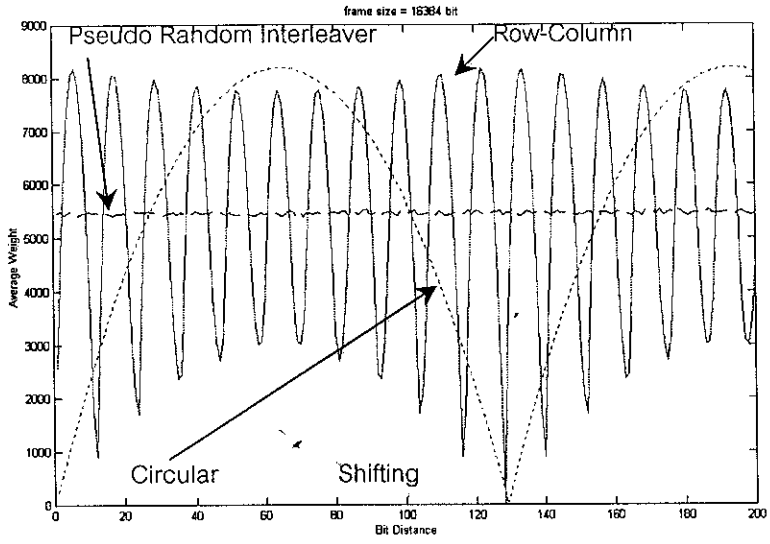
1. ที่ Bit Distance มีค่าน้อย (บิตที่อยู่ติดกัน) Pseudo Random Interleaver จะมีค่า Average Weight น้อยที่สุด และ Row-Column Interleaver จะมีค่า Average Weight มากที่สุด แสดงว่าบิตที่อยู่ติดกันของ Row-Column Interleaver จะมาจากบิตที่อยู่ห่างกันมากที่สุด
2. สำหรับ Average Weight โดยรวมของ Interleaver ทุกแบบมีต่างกันมากนัก



ภาพที่ 5-8 ภาพแสดงความสัมพันธ์ระหว่าง Average Weight กับ Bit Distance (ความยาวเฟรม 1024 บิต)

จากภาพที่ 5-8 แสดงให้เห็นว่าที่ ความยาวเฟรม 1024 บิต มีข้อสังเกตดังนี้

1. ที่ Bit Distance มีค่าน้อย (บิตที่อยู่ติดกัน) Circular Shifting Interleaver จะมีค่า Average Weight น้อยที่สุด และ Row-Column Interleaver จะมีค่า Average Weight มากที่สุด แสดงว่าบิตที่อยู่ติดกันของ Row-Column Interleaver จะมาจากบิตที่อยู่ห่างกันมากที่สุด
2. สำหรับ Average Weight โดยรวม
   2.1 Pseudo-Random Interleaver จะมีค่า Average Weight ใกล้เคียงกันทุก Bit Distance
   2.2 Circular Shifting Interleaver และ Row-Column Interleaver มีค่า Average Weight ที่น้อยที่สุดใกล้เคียงกัน
   2.3 จำนวน Average Weight ที่มากที่สุดของ Row-Column Interleaver จะมากว่า ของ Circular Shifting Interleaver



ภาพที่ 5-9 ภาพแสดงความสัมพันธ์ระหว่าง Average Weight กับ Bit Distance (ความยาวเฟรม 16384 บิต)

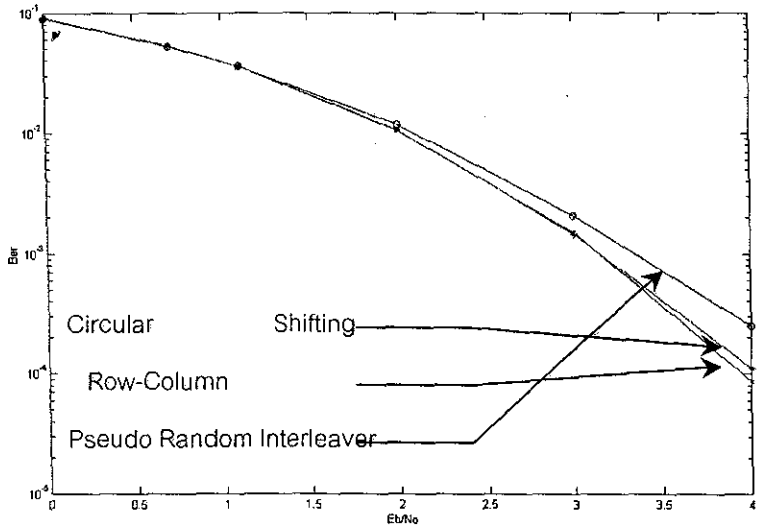
จากภาพที่ 5-9 แสดงให้เห็นว่าที่ ความยาวเฟรม 16384 บิต มีข้อสังเกตดังนี้

1. ที่ Bit Distance มีค่าน้อย (บิตที่อยู่ติดกัน) Circular Shifting Interleaver จะมีค่า Average Weight น้อยที่สุด และ Row-Column Interleaver จะมีค่า Average Weight มากที่สุด แสดงว่าบิตที่อยู่ติดกันของ Row-Column Interleaver จะมาจากบิตที่อยู่ห่างกันมากที่สุด
2. สำหรับ Average Weight โดยรวม
   2.1 Pseudo-Random Interleaver จะมีค่า Average Weight ใกล้เคียงกันทุก Bit Distance
   2.2 Circular Shifting Interleaver และ Row-Column Interleaver มีค่า Average Weight ที่น้อยที่สุดใกล้เคียงกัน
   2.3 จำนวน Average Weight ที่มากที่สุดของ Row-Column Interleaver จะมากว่า ของ Circular Shifting Interleaver

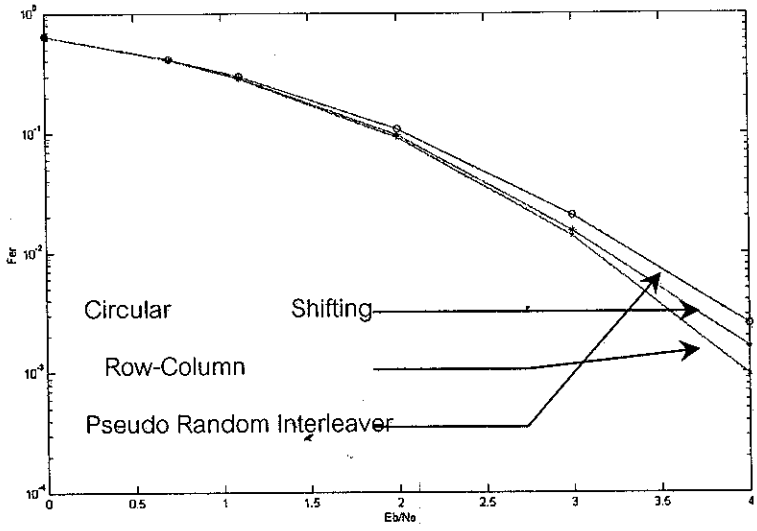
5.3 ผลการทดสอบ Turbo Code เนื่องจากการใช้ Interleaver ที่ต่างกันบนแบบจำลองช่องสัญญาณ AWGN ที่ความยาวเฟรม ต่างกัน

5.3.1 การทดสอบ Turbo Code ที่ ความยาวเฟรม 40 บิต

โดยใช้ Interleaver ที่ต่างกัน ซึ่งได้แก่ Pseudo-Random Interleaver, Circular Shifting Interleaver และ Row-Column Interleaver ที่ ความยาวเฟรม 40 บิต, G = (1 1 1 , 1 0 1), rate = 1/2 ได้ผลการทดสอบดังภาพที่ 5-10 และ ภาพที่ 5-11



ภาพที่ 5-10 ภาพแสดงผลการทดสอบโปรแกรม Turbo Code เทียบระหว่าง BER กับ Eb/N0 โดยใช้ ความยาวเฟรม 40 บิต, G = (1 1 1 , 1 0 1), ที่ rate = 1/2

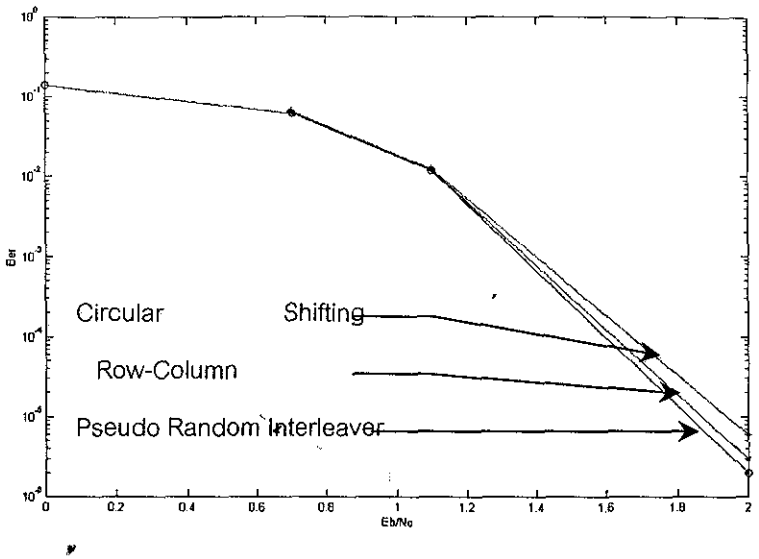


ภาพที่ 5-11 ภาพแสดงผลการทดสอบโปรแกรม Turbo Code เทียบระหว่าง FER กับ Eb/N0 โดยใช้ ความยาวเฟรม 40 บิต, G = (1 1 1 , 1 0 1), ที่ rate = 1/2

จากผลการทดสอบ Turbo Code ภาพที่ 5-10 และภาพที่ 5-11 แสดงการเปรียบเทียบระหว่าง BER, FER กับ Eb/N0 ที่ความยาวเฟรม 40 บิต ระหว่าง Pseudo-Random Interleaver, Circular Shifting Interleaver และ Row-Column Interleaver พบว่าผลการการทดสอบ Turbo Code ที่ใช้ Interleaver ทั้ง 3 แบบ มีค่าใกล้เคียงกันโดยผลของ Turbo Code ที่ใช้ Row-Column Interleaver และ Circular Shifting Interleaver ให้ผลที่ใกล้เคียงกัน และดีกว่าผลของ Turbo Code ที่ใช้ Pseudo-Random Interleaver เล็กน้อย

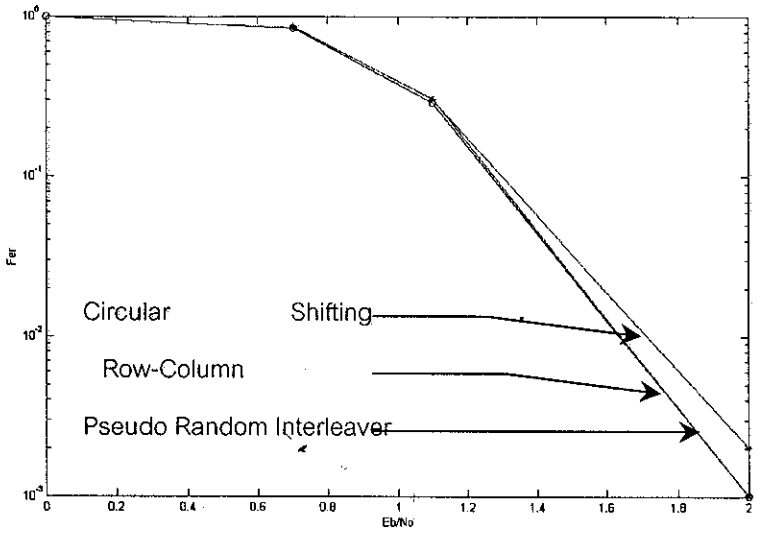
5.3.2 การทดสอบ Turbo Code ที่ ความยาวเฟรม 1024 บิต

โดยใช้ Interleaver ที่ต่างกัน ซึ่งได้แก่ Pseudo-Random Interleaver, Circular Shifting Interleaver และ Row-Column Interleaver ที่ ความยาวเฟรม 1024 บิต, G = (1 0 0 1 1 , 1 1 1 0 1), rate = 1/2 ได้ผลการทดสอบดังภาพที่ 5-12 และ ภาพที่ 5-13



ภาพที่ 5-12 ภาพแสดงผลการทดสอบโปรแกรม Turbo Code เทียบระหว่าง BER กับ Eb/N0 โดยใช้ ความยาวเฟรม 1024 บิต, G = (1 0 0 1 1 , 1 1 1 0 1), ที่ rate = 1/2

จากผลการทดสอบ Turbo Code ภาพที่ 5-12 แสดงการเปรียบเทียบระหว่าง BER กับ Eb/N0 ที่ความยาวเฟรม 1024 บิต ระหว่าง Pseudo-Random Interleaver, Circular Shifting Interleaver และ Row-Column Interleaver พบว่าผลการทดสอบของ Turbo Code ที่ใช้ Pseudo-Random Interleaver มีค่า BER น้อยกว่าที่ใช้ Interleaver แบบอื่นเล็กน้อย และผลของ Turbo Code ที่ใช้ Row-Column Interleaver และ Circular Shifting Interleaver ให้ผลที่ใกล้เคียงกัน โดยเมื่อใช้ Row-Column Interleaver จะได้ค่า BER น้อยกว่าเมื่อใช้ Circular Shifting Interleaver เล็กน้อย

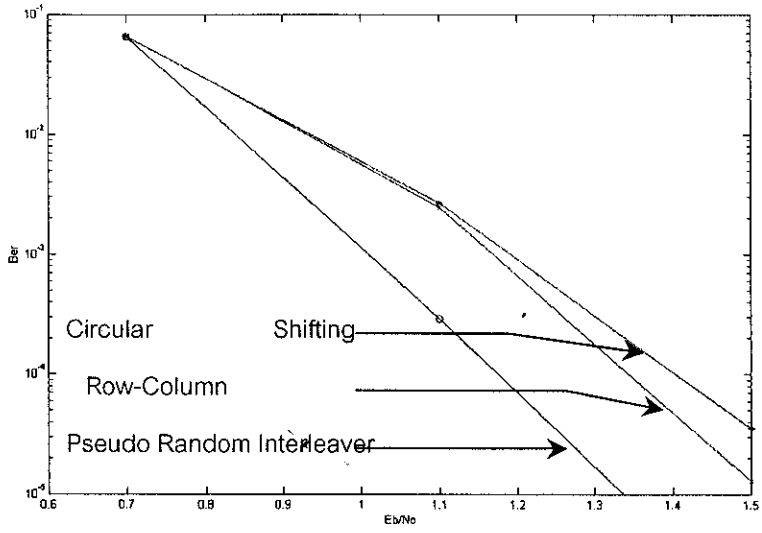


ภาพที่ 5-13 ภาพแสดงผลการทดสอบโปรแกรม Turbo Code เทียบระหว่าง FER กับ Eb/N0 โดยใช้ ความยาวเฟรม 1024 บิต, G = (1 0 0 1 1 , 1 1 1 0 1), ที่ rate = 1/2

จากผลการทดสอบ Turbo Code ภาพที่ 5-13 แสดงการเปรียบเทียบระหว่าง FER กับ Eb/N0 ที่ความยาวเฟรม 1024 บิต ระหว่าง Pseudo-Random Interleaver, Circular Shifting Interleaver และ Row-Column Interleaver พบว่าผลการทดสอบของ Turbo Code ที่ใช้ Pseudo-Random Interleaver ให้ค่า FER ที่ใกล้เคียงกับ ผลการทดสอบของ Turbo Code ที่ใช้ Row-Column Interleaver และ ดีกว่าผลการทดสอบ Turbo Code ที่ใช้ Circular Shifting Interleaver เล็กน้อย

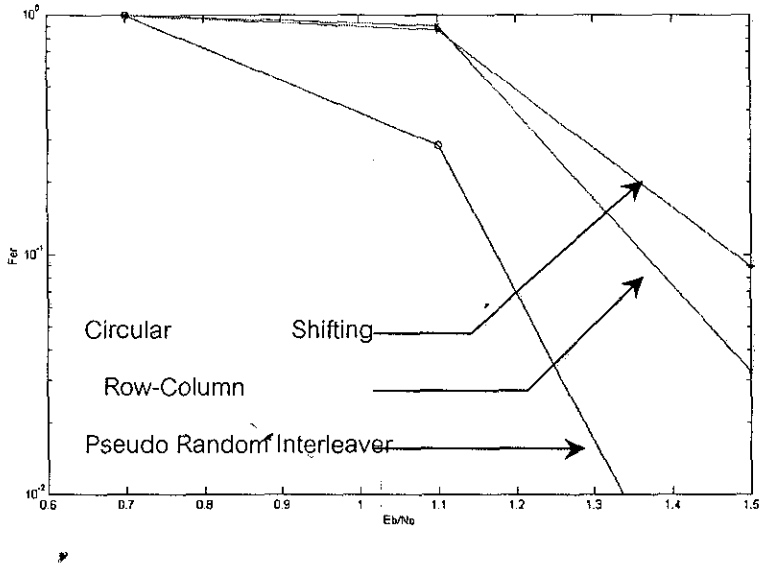
5.3.3 การทดสอบ Turbo Code ที่ ความยาวเฟรม 16384 บิต

โดยใช้ Interleaver ที่ต่างกัน ซึ่งได้แก่ Pseudo-Random Interleaver, Circular Shifting Interleaver และ Row-Column Interleaver ที่ ความยาวเฟรม 16384 บิต, G = (1 0 0 1 1 , 1 1 1 0 1), rate = 1/2 ได้ผลการทดสอบดังภาพที่ 5-14 และ ภาพที่ 5-15



ภาพที่ 5-14 ภาพแสดงผลการทดสอบโปรแกรม Turbo Code เทียบระหว่าง BER กับ Eb/N0 โดยใช้ ความยาวเฟรม 16384 บิต, G = (1 0 0 1 1 , 1 1 1 0 1), ที่ rate = 1/2

จากผลการทดสอบ Turbo Code ภาพที่ 5-14 แสดงการเปรียบเทียบระหว่าง BER กับ Eb/N0 ที่ความยาวเฟรม 16384 บิต ระหว่าง Pseudo-Random Interleaver, Circular Shifting Interleaver และ Row-Column Interleaver พบว่าผลการทดสอบของ Turbo Code ที่ใช้ Pseudo-Random Interleaver มีค่า BER น้อยกว่าที่ใช้ Interleaver แบบอื่นอย่างเห็นได้ชัด และผลของ Turbo Code ที่ใช้ Row-Column Interleaver และ Circular Shifting Interleaver ให้ผลที่ใกล้เคียงกัน โดยเมื่อใช้ Row-Column Interleaver จะได้ค่า BER น้อยกว่าเมื่อใช้ Circular Shifting Interleaver เล็กน้อย



ภาพที่ 5-15 ภาพแสดงผลการทดสอบโปรแกรม Turbo Code เทียบระหว่าง FER กับ Eb/N0 โดยใช้ ความยาวเฟรม 16384 บิต, G = (1 0 0 1 1 , 1 1 1 0 1), ที่ rate = 1/2

จากผลการทดสอบ Turbo Code ภาพที่ 5-13 แสดงการเปรียบเทียบระหว่าง FER กับ Eb/N0 ที่ความยาวเฟรม 16384 บิต ระหว่าง Pseudo-Random Interleaver, Circular Shifting Interleaver และ Row-Column Interleaver พบว่าผลการทดสอบของ Turbo Code ที่ใช้ Pseudo-Random Interleaver มีค่า FER น้อยกว่าที่ใช้ Interleaver แบบอื่นอย่างเห็นได้ชัด และผลของ Turbo Code ที่ใช้ Row-Column Interleaver และ Circular Shifting Interleaver ให้ผลที่ใกล้เคียงกัน โดยเมื่อใช้ Row-Column Interleaver จะได้ค่า FER น้อยกว่าเมื่อใช้ Circular Shifting Interleaver เล็กน้อย

5.4 สรุปผลของการทดสอบโปรแกรมที่ความยาวเฟรมต่างกันเปรียบเทียบกับความสัมพันธ์ระหว่าง Average Weight กับ Bit Distance ของ Interleaver ที่ความยาวเฟรมต่างกัน

5.4.1 สรุปผลของการทดสอบโปรแกรมมีความยาวเฟรมน้อย

เมื่อทดสอบโปรแกรมที่ความยาวเฟรมน้อยพบกว่า BER และ FER ของ Turbo Code ที่ใช้ Row-Column Interleaver และ Circular Shifting Interleaver ค่าใกล้เคียงกัน และมีค่าน้อยกว่า Turbo Code ที่ใช้ Pseudo-Random Interleaver เนื่องจาก Average Weight โดยรวมของ Interleaver ที่ความยาวเฟรมน้อยไม่แตกต่างกันมากนัก และ Row-Column Interleaver มีแนวโน้มของผลที่ดีกว่า Circular Shifting Interleaver เนื่องจากค่า Average Weight มีค่าสูงกว่า

5.4.2 สรุปผลของการทดสอบโปรแกรมมีความยาวเฟรมมาก

เมื่อทดสอบโปรแกมที่ความยาวเฟรมมากพบกว่า BER และ FER ของ Turbo Code ที่ใช้ Pseudo-Random Interleaver และมีค่าน้อยกว่ามากเมื่อเทียบกับ Turbo Code ที่ใช้ Row-Column Interleaver และ Circular Shifting Interleaver ค่าใกล้เคียงกัน เนื่องจาก Average Weight โดยรวมของ Pseudo-Random Interleaver ที่ความยาวเฟรมมาก ให้ค่าที่สูงและคงที่ทุก Bit Distance สำหรับ Row-Column Interleaver และ Circular Shifting Interleaver ให้ค่า BER และ FER ที่ใกล้เคียงกัน เมื่อความยาวเฟรมมาก และ Row-Column Interleaver มีแนวโน้มของผลที่ดีกว่า Circular Shifting Interleaver เนื่องจากค่า Average Weight มีค่าสูงกว่า

บทที่ 6
บทสรุป และข้อเสนอแนะ

6.1 บทสรุป

ผลจากการทำโครงงานสามารถสรุปได้ดังนี้

- สามารถสร้าง Row-Column Interleaver ที่เป็น Deterministically Generate Interleaver ซึ่งเป็นวิธีใหม่ที่มีแนวโน้มน่าพอใจเมื่อเทียบกับ Circular Shifting Interleaver
- สามารถสร้างแบบจำลองช่องสัญญาณ ซึ่งประกอบด้วย AWGN, Ricean Fading และ Rayleigh Fading ได้
- สามารถใช้เครื่องมือสร้างแบบจำลอง Turbo Code สำหรับ โปรแกรม Matlab ในการเข้ารหัสถอดรหัสแบบ Turbo Code ได้ โดยเลือกใช้ Interleaver และ แบบจำลองช่องสัญญาณได้หลายชนิด
- สามารถนำเครื่องมือสร้างแบบจำลอง Turbo Code มาวิเคราะห์ประสิทธิภาพของ Turbo Code เมื่อใช้ Interleaver ต่างชนิดกันที่ความยาวเฟรมต่างกันได้

6.2 ปัญหาและอุปสรรคในการทำงาน

- ไม่มีความรู้ในการสร้าง Simulink ของโปรแกรม Matlab จึงใช้เวลามากในการศึกษา
- สำหรับการทดสอบผลโปรแกรมใช้เวลามากในการดำเนินการ
- สำหรับการเตรียมข้อมูลทำโครงงาน ไม่ได้มีการเตรียมข้อมูลล่วงหน้าทำให้ใช้เวลามากในการเตรียมข้อมูล การศึกษาทำความเข้าใจ

6.3 ขีดจำกัดของโครงงาน

- สำหรับเครื่องมือสร้างแบบจำลองช่องสัญญาณ Ricean Fading และ Rayleigh Fading ไม่ควรใช้ข้อมูลที่มีความยาวเฟรมมากกว่า 4000 เพราะโปรแกรมกำหนดไว้เพื่อไม่ให้การสร้างแบบจำลองช่องสัญญาณนานเกินไป และตัวโปรแกรมจะใช้หน่วยความจำมากในการเก็บตัวแปร
- โปรแกรมสำหรับทดสอบผลต้องใช้เวลามากเนื่องจากต้องทำการสุ่มเก็บค่าผลการทดสอบให้มากพอ

6.4 ข้อเสนอแนะเพิ่มเติม

- โปรแกรมสำหรับทดสอบถ้าเปลี่ยนจากการใช้ Matlab เป็นภาษา C น่าจะใช้เวลาน้อยลง
- สำหรับส่วนที่เป็นเครื่องมือสร้างแบบจำลอง Turbo Code สามารถสร้างส่วนที่เป็น Interleaver หรือแบบจำลองช่องสัญญาณอื่นเพิ่มเติมได้

รายการอ้างอิง

[1] Claude Berrou, Alain Glavieux และ Punya Thitimajshima, 1993, "Near Shannon Limit Error-Correcting Coding and Decoding : Turbo-Codes (1)", IEEE

[2] สมศักดิ์ วาณิชอนันต์ชัย, 2000, "เอกสารประกอบการเรียนการสอนวิชา 409422 DIGITAL COMMUNICATION", มหาวิทยาลัยเทคโนโลยีสุรนารี

[3] Joseph C. Liberti, JR. และ Theodore S. Rappaport, "Smart Antennas for Wireless Communications: IS-95 and Third Genration CDMA Applications", Prentice Hall Communications Engineering and Emerging Technologies Series

[4] Martin Bossert, 1999, "Channel Coding for Telecommunications", WILEY

[5] Akira SHIBUTANI, Hirohito SUDA และ Yasushi YAMAO, 2000, "Performance of W-CDMA Mobile Radio with Turbo Codes Using Prime Interleaver", IEEE

[6] Yufei Wu, May 1999, "Design and Implementation of Parallel and Serial Concatenated Convolutional Codes", Blacksburg, Virginia

[7] Oscar Y. Takeshita และ Daniel J. Costello, Jr., 2000, "New Deterministic Interleaver Designs for Turbo Codes", IEEE

[8] Moataz Mohamed Salah, June 2000, "Turbo Codes for Wireless Mobile Communication Systems Applications", APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

[9] William C.Y. Lee, 1998, "Mobile Communicatioins Engineering", McGraw-Hill Telecommunications

[10] เรืองฤทธิ์ เพชรวกุล, 2000, " website วิชา Mobile Communications คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย", http://www.geocities.com/learncdma/

[11] www.e-technik.uni-kl.de, 2001, " CDMA Glossary A-F", [www document] URL http://www.e-technik.uni-kl.de/baier/Staff/Clos/further_info/cdma/DefAtoF.html