ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)

ปริญญา

วิศวกรรมไฟฟ้า | วิศวกรรมไฟฟ้า

สาขา | **ภาควิชา**

เรื่อง ต้นแบบเครื่องถอดรหัสภายในสำหรับตัวถอดรหัสคอนโวลูชันภายนอก

List Viterbi Decoder as an Inner Decoder for Convolutional Vector Symbol Outer Decoder

นามผู้วิจัย นายพงษ์พิสุทธิ์ นรดี

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ผู้ช่วยศาสตราจารย์อุศนา ตัณฑุลเวศม์, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม

(รองศาสตราจารย์มงคล รักษาพัชรวงศ์, Ph.D.)

หัวหน้าภาควิชา

(รองศาสตราจารย์มงคล รักษาพัชรวงศ์, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

ต้นแบบเครื่องถอดรหัสภายในสำหรับตัวถอดรหัสคอนโวลูชันภายนอก

List Viterbi Decoder as an Inner Decoder for Convolutional Vector Symbol Outer Decoder

โดย

นายพงษ์พิสุทธิ์ นรดี

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)
พ.ศ. 2553

พงษ์พิสุทธิ์ นรดี 2553: ต้นแบบเครื่องถอดรหัสภายในสำหรับตัวถอดรหัสคอนโวลูชันภายนอก ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า) สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: ผู้ช่วยศาสตราจารย์อุศนา ตัณฑุลเวศม์, Ph.D. 160 หน้า

งานวิจัยนี้แบ่งออกเป็นสองส่วนหลักๆคือ 1. การออกแบบและสร้างชิ้นงานต้นแบบเครื่องถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์บนบอร์ดอิเล็กทรอนิกส์ FPGA โดยชิ้นงานนี้ถูกออกแบบให้เหมาะสมที่จะเป็นตัวถอดรหัสภายในของรหัสคอนคาทีเนตที่มีการถอดรหัสแบบเวกเตอร์ซิมโบลดีโคดดิงเป็นตัวถอดรหัสภายนอก และ 2. การหาช่องสัญญาณและพารามิเตอร์ที่เหมาะสมของช่องสัญญาณ สำหรับใช้งานกับรหัสคอนคาทีเนตดังกล่าว สำหรับส่วนแรกชิ้นงานได้ถูกสร้างขึ้นบนบอร์ด FPGA และผลการทดสอบแสดงให้เห็นว่า โปรแกรม VHDL ให้ผลลัพธ์ในทุกช่วงการทำงานเหมือนกับโปรแกรม C++ และผลลัพธ์สุดท้ายจากบอร์ด FPGA ก็มีค่าใกล้เคียงกับผลลัพธ์จากโปรแกรม C++ จึงสรุปได้ว่า ชิ้นงานทำงานได้ถูกต้อง สำหรับในส่วนที่สอง รหัสคอนคาทีเนตที่ได้นี้เหมาะที่จะใช้กับช่องสัญญาณที่มีความผิดพลาดติดๆกัน ดังนั้นจึงทำการทดสอบในช่องสัญญาณไร้สายที่มีการจางหายแบบเรย์ลีและแบบไรเชียน โดยมีการทดสอบในช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก เพื่อเปรียบเทียบ สำหรับงานวิจัยนี้ ช่วงความน่าจะเป็นที่ผลลัพธ์แรกผิดที่เราสนใจ จะอยู่ในช่วง 0.01 ถึง 0.1 ซึ่งมีค่าสูงกว่าค่าที่สนใจโดยทั่วไป เนื่องจากสัญลักษณ์ที่เหลืออยู่นั้นจะถูกแก้ไขโดยตัวถอดรหัสภายนอก จากการทดลองพบว่า ตัวอย่างค่า SNR ของสัญญาณที่เหมาะกับตัวถอดรหัสและมาประยุกต์ใช้กับงานแบบนี้จะอยู่ในช่วง 2 ถึง 3.5 dB สำหรับช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ช่วง 6.5 ถึง 8.1 dB สำหรับช่องสัญญาณการจางหายแบบไรเชียนที่ความเร็ว 60 กิโลเมตรต่อชั่วโมงและค่าไรเชียนแฟคเตอร์เท่ากับ 10 และช่วง 8.1 ถึง 9.3 dB สำหรับช่องสัญญาณการจางหายแบบเรย์ลีที่ความเร็ว 120 กิโลเมตรต่อชั่วโมง ซึ่งเป็นช่วงที่ช่องสัญญาณมีคุณภาพต่ำ นอกจากนั้นแล้วค่า SNR ในช่วงดังกล่าว ยังให้ค่าความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิดอยู่ในช่วงที่มีประโยชน์ต่อตัวถอดรหัสภายนอกมาก

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Pongpisut Noradee 2010: List Viterbi Decoder as an Inner Decoder for Convolutional Vector Symbol Outer Decoder. Master of Engineering (Electrical Engineering), Major Field: Electrical Engineering, Department of Electrical Engineering. Thesis Advisor: Assistant Professor Usana Tuntoolavest, Ph.D. 160 pages.

This research is divided into two parts: 1. Designing and implementing a lab prototype of a list-of-2 Viterbi decoder on an FPGA electronic board. The lab prototype was designed to be a suitable inner decoder of a concatenated code, in which the vector symbol decoder is the outer decoder and 2. Finding the suitable channels and their suitable parameter for this concatenated code. For the first part, the lab prototype was implemented on an FPGA board. The test results showed that the outputs from the VHDL program were the same as those from the C++ simulation. It can be concluded from the results that the lab prototype worked properly. For the second part, this concatenated code is suitable for the channel with burst errors. Therefore, it was tested in the wireless channels with Rayleigh and Ricean fading. The additive white Gaussian noise channel was shown for comparison purpose. For this research, the probability that the first output is wrong in consideration is between 0.01 and 0.1, which is higher than the one that is generally used. This is because the remaining error symbols will be corrected again by the outer decoder. The results showed that the suitable SNR of various channels for this decoder was as follows: from 2 to 3.5 dB for the additive white Gaussian noise channel, from 6.5 to 8.1 dB for the SNR of the Ricean fading channel with the speed of 60 km/h and Ricean factor equal to 10, from 8.1 to 9.3 dB for the Rayleigh fading channel with the speed of 120 km/h. These ranges of SNR correspond to low quality channels. Moreover, in these ranges of SNR, the corresponding values of the probability that the second output is wrong given that the first output is wrong are very useful to the outer decoder.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ผู้วิจัยขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.อุศนา ตัณฑุลเวศม์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก รองศาสตราจารย์ ดร.มงคล รักษาพัชรวงศ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ที่ให้คำปรึกษาในการเรียน การค้นคว้าวิจัย ตลอดจนการตรวจแก้ไขวิทยานิพนธ์จนกระทั่งเสร็จสมบูรณ์ และกราบขอบพระคุณ ผู้แทนบัณฑิตวิทยาลัย ที่ได้ให้ความกรุณาตรวจแก้ไขวิทยานิพนธ์ให้สมบูรณ์ยิ่งขึ้น

ขอกราบขอบพระคุณอาจารย์ภาควิชาวิศวกรรมไฟฟ้าทุกท่าน ที่ได้อบรมสั่งสอนและมอบความรู้อันเป็นประโยชน์อย่างยิ่งในการนำไปใช้ประโยชน์ต่อไป และขอขอบคุณ รุ่นพี่จากห้องวิจัย SCORPionที่ให้ความช่วยเหลือและให้คำแนะนำต่างๆ รวมทั้ง คุณรัชนนท์ อินทรสกุล และคุณอรรถภัทร์ สืบเนื่อง ที่ให้ความช่วยเหลือและให้คำแนะนำในการเขียนโปรแกรมต่างๆ เพื่อใช้ในงานวิจัย และขอขอบคุณบัณฑิตวิทยาลัยและสถาบันวิจัยและพัฒนาแห่งมหาวิทยาลัยเกษตรศาสตร์ที่ให้ทุนสนับสนุนการวิจัยครั้งนี้ด้วย

ด้วยความดีหรือประโยชน์อันใดเนื่องจากวิทยานิพนธ์เล่มนี้ ขอมอบแด่คุณพ่อ คุณแม่ ที่ได้อบรมและให้กำลังใจผู้วิจัยตลอดมาในทุกเรื่อง

พงษ์พิสุทธิ์ นรดี

เมษายน 2553

สารบัญ

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

คำอธิบายสัญลักษณ์และคำย่อ

AWGN	=	additive white Gaussian noise
CDF	=	cumulative distribution function
CLBs	=	configurable – specific integrated circuit
DDR2 SDRAM	=	double data rate SDRAM
DMC	=	discrete memoryless channel
EEPROM	=	electrically erasable PROM
FIFO	=	first in first out
FPGA	=	field programmable gate array
FSK	=	frequency shift keying
FSM	=	finite state machine
HDL	=	hardware description language
LCD	=	liquid crystal display
LED	=	light emitting diode
LOS	=	line of sight
LUT	=	lookup table
LVA	=	list Viterbi algorithm
MLSE	=	maximum likelihood sequence estimator
PCB	=	printed circuit board
PC	=	personal computer
PDF	=	probability density function
PLD	=	programmable logic devices
PROM	=	programmable read – only memory
PSK	=	phase shift keying
SDRAM	=	synchronous dynamic random access memory
USB	=	universal serial bus
VHDL	=	very – high – speed integrated circuits HDL
VSD	=	vector symbol decoding

ต้นแบบเครื่องถอดรหัสภายในสำหรับตัวถอดรหัสคอนโวลูชันภายนอก

List Viterbi Decoder as an Inner Decoder for Convolutional Vector Symbol Outer Decoder

คำนำ

เทคโนโลยีระบบสื่อสารโทรคมนาคมเติบโตอย่างรวดเร็วและต่อเนื่อง บนพื้นฐานของรูปแบบการส่งสัญญาณดิจิทัลซึ่งมีข้อดีกว่าระบบการส่งสัญญาณแอนะล็อก เช่น มีความทนทานต่อสัญญาณรบกวน (Noise signal) เป็นผลให้การได้มาซึ่งสัญญาณข้อมูลเดิมที่ถูกต้อง มีความเชื่อถือได้ ทั้งนี้ระบบสื่อสารดิจิทัลนั้น จะให้ความยืดหยุ่นสูงในกระบวนการจัดการสัญญาณ แต่จะมีความยุ่งยากซับซ้อนมากกว่าสัญญาณระบบแอนะล็อก ปัจจุบันเทคโนโลยีการสื่อสารโทรคมนาคมได้มีการพัฒนาไปอย่างรวดเร็ว โดยเฉพาะการสื่อสารไร้สาย (Wireless communications) ได้มีการใช้งานกันมาก เช่น โทรศัพท์เคลื่อนที่ (Mobile phone) การสื่อสารผ่านดาวเทียม (Satellite communication) และวิทยุกระจายเสียง เป็นต้น โดยเฉพาะระบบโทรศัพท์เคลื่อนที่ได้ถูกพัฒนาให้สามารถส่งสัญญาณเสียง สัญญาณภาพ รวมไปถึงการรองรับบริการต่างๆที่จะมีขึ้นในอนาคต ปัญหาสำคัญของการสื่อสารไร้สายคือ สภาพแวดล้อมของตัวกลางที่มีการเปลี่ยนแปลงอยู่ตลอดเวลา ทำให้เกิดสัญญาณรบกวน (Noise) และสัญญาณแทรกสอด (interference) มากมายในช่องสัญญาณ (channel) ซึ่งอาจทำให้การสื่อสารมีความผิดพลาด (error) เกิดขึ้นได้ สำหรับวิธีหลักวิธีหนึ่งที่ทำให้การสื่อสารไร้สายแบบดิจิทัลมีความเชื่อถือมากขึ้น แม้จะมีความบกพร่องต่างๆของช่องสัญญาณ (channel impairment) ก็คือการใช้รหัสช่องสัญญาณ (channel coding) สำหรับรหัสช่องสัญญาณที่นิยมใช้กันมากในช่องสัญญาณแบบนี้ คือ รหัสคอนคาทีเนต

รหัสคอนคาทีเนต (Concatenated code) (Forney, 1966;Lin and Costello, 2004) สามารถแก้ไขความผิดพลาดที่เกิดขึ้นทั้งในรูปแบบที่ผิดหลายบิทติดๆกัน (burst error) จากผลของเฟดดิ้ง (Fading) และความผิดพลาดแบบสุ่ม (random error) ที่เกิดจากสัญญาณรบกวน การเข้ารหัสคอนคาทีเนต มี 2 ส่วนคือ ส่วนแรกจะเป็นส่วนของรหัสภายนอก (Outer code) ซึ่งต้องใช้สัญลักษณ์นอนไบนารี (Nonbinary code) และส่วนของรหัสภายใน (Inner code) ซึ่งมักจะใช้สัญลักษณ์ไบนารี (Binary code)โดยปกติรหัสภายนอกที่ใช้จะเป็นรหัสแบบบล็อก เนื่องจากวิธีการถอดรหัสคอนโวลูชัน ที่ใช้กันโดยทั่วไปไม่เหมาะสมกับการถอดรหัสนอนไบนารี แต่มีวิธีการถอดรหัสหนึ่งที่เรียกว่า การถอดรหัสแบบเวกเตอร์ซิมโบลดีโคดดิง (Vector Symbol Decoding) หรือวีเอสดี

(VSD) ซึ่งถูกออกแบบมา สำหรับถอดรหัสที่ใช้สัญลักษณ์ขนาดใหญ่ (เช่น 32 บิทต่อสัญลักษณ์) โดยเฉพาะ

การถอดรหัสแบบวีเอสดี จะทำงานได้ง่าย เร็ว และมีประสิทธิภาพเพิ่มขึ้น เมื่อสัญลักษณ์ที่ได้รับมีสองตัวเลือก แทนที่จะมีตัวเลือกเดียวที่ใช้ในการตัดสินใจ การได้มาซึ่งตัวเลือกมีหลายวิธี เช่น การใช้ไดเวอร์ซิตี้ (diversity) หรือได้จากวิธีการถอดรหัสภายในที่สามารถให้ผลลัพธ์ได้มากกว่าหนึ่งแบบ เช่น ให้ผลลัพธ์ที่น่าจะเป็นไปได้มากที่สุดสองอันดับแรก เป็นต้น เนื่องจากการวิเคราะห์ประสิทธิภาพของการถอดรหัสวีเอสดี ได้มีการนำการถอดรหัสแบบวิเทอร์บิแบบสองผลลัพธ์ (list Viterbi decoding with list of two) มาใช้ เนื่องจากการถอดรหัสแบบวิเทอร์บิเป็นการถอดรหัสที่เหมาะที่สุด (Optimum receiver) สำหรับการถอดรหัสแบบสัญลักษณ์ไบนารี และเป็นวิธีที่ให้ประสิทธิภาพสูง โดยได้ผลการจำลองการทำงานที่ดี และเนื่องจากขณะนี้ได้มีการพัฒนาชิ้นงานตัวถอดรหัสวีเอสดีอยู่บนบอร์ดอิเล็กทรอนิกส์ FPGA รวมทั้งชิ้นงานตัวเข้ารหัสที่เหมาะสมก็ได้มีการสร้างเสร็จแล้ว ดังนั้น โครงการวิทยานิพนธ์นี้ จะสร้างชิ้นงานถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ บนบอร์ดอิเล็กทรอนิกส์ที่ใช้ชิพ FPGA (Field Programmable Gate Array) รวมทั้งออกแบบระบบสื่อสารในส่วนอื่นๆ เพื่อสร้างต้นแบบของระบบที่ใช้การถอดรหัสวีเอสดีที่จะนำไปใช้ในทางปฏิบัติต่อไป

ชิ้นงานถอดรหัสแบบวิเทอร์บิแบบสองผลลัพธ์ บนบอร์ดอิเล็กทรอนิกส์ FPGA เป็นชิ้นงานใหม่ เนื่องจากวิธีการถอดรหัสแบบวิเทอร์บิแบบหลายผลลัพธ์นั้น (Seshadri and Sungberg, 1994;Roder and Hamzaoui, 2006) ถ้าใช้ด้วยตัวมันเองจะต้องใช้จำนวนผลลัพธ์มาก จึงทำให้ชิ้นงานมีความซับซ้อนมากในทางปฏิบัติ แต่เนื่องจากการใช้งานที่เลือกเป็นการประยุกต์ใช้กับตัวถอดรหัสวีเอสดี จึงเหมาะสมที่จะใช้เพียงสองผลลัพธ์ ดังนั้นความซับซ้อนจะลดน้อยลงมาก เนื่องจากการถอดรหัสวีเอสดีนั้น การมีตัวเลือกสองตัวเลือกเป็นการเพิ่มประสิทธิภาพของตัวถอดรหัสอย่างชัดเจน การเพิ่มตัวเลือกเป็นสามหรือสี่ตัวเลือกจะทำให้ประสิทธิภาพของของวีเอสดีเพิ่มขึ้นด้วย แต่ประสิทธิภาพที่เพิ่มขึ้นนั้น จะเพิ่มจากการใช้สองตัวเลือกเพียงเล็กน้อยเท่านั้น ซึ่งไม่คุ้มค่ากับความซับซ้อนของอุปกรณ์ที่เพิ่มขึ้นด้วย (Tuntoolavest and Seubnaung, 2007)

การสร้างตัวถอดรหัสให้เป็นชิ้นงาน จะใช้เทคโนโลยีการสร้างวงจรรวม (Integrate Circuit) ในการออกแบบสร้างตัวถอดรหัส และการออกแบบวงจรรวมทำได้โดยไม่ยากนัก สามารถจำลองการทำงานและทดสอบการทำงานได้สะดวกบนเครื่องคอมพิวเตอร์ ทำให้สามารถพัฒนาได้อย่าง

รวดเร็วและถูกต้อง ภาษาที่ใช้จะใช้ภาษาบรรยายฮาร์ดแวร์ HDL (Hardware Description Language) เพื่อออกแบบวงจรรวมบนชิพ FPGA ซึ่งมีการใช้งานกันอย่างแพร่หลายในวงการอุตสาหกรรม

ช่องสัญญาณเป็นส่วนสำคัญส่วนหนึ่งที่ทำให้สัญญาณที่ส่งผ่านอากาศเกิดความเสียหายและผิดเพี้ยนไปจากสัญญาณส่ง ดังนั้นการวิเคราะห์ประสิทธิภาพของระบบการสื่อสารไร้สายจำเป็นต้องทำการวิเคราะห์ทั้งระบบสื่อสาร สำหรับระบบสื่อสารที่กล่าวถึงประกอบไปด้วยเครื่องส่ง ช่องสัญญาณและเครื่องรับ ดังนั้นหากต้องการวิเคราะห์ประสิทธิภาพของระบบใดระบบหนึ่งจำเป็นจะต้องสร้างเครื่องส่งและเครื่องรับขึ้นมาเพื่อใช้ในการวิเคราะห์ แต่ในความเป็นจริงแล้วการสร้างอุปกรณ์สื่อสารเพื่อวิเคราะห์ในเบื้องต้นนั้นเป็นการไม่สะดวก เนื่องจากต้นทุนที่ใช้สร้างอุปกรณ์ดังกล่าวมีราคาสูง เสียเวลามาก และปรับแต่งส่วนที่ต้องการวิเคราะห์ได้ยาก ดังนั้นการวิเคราะห์โดยอาศัยแบบจำลองจึงได้รับความนิยมมากว่า แต่การวิเคราะห์โดยใช้แบบจำลองต้องมีการควบคุมส่วนต่างๆในระบบการสื่อสารให้เหมาะสมและเหมือนการสื่อสารจริงมากที่สุด เพื่อให้ผลลัพธ์ที่ได้จากการทดสอบมีความน่าเชื่อถือ

สิ่งสำคัญและมีความจำเป็นอย่างมากในการวิเคราะห์ประสิทธิภาพของระบบการสื่อสาร คือ ช่องสัญญาณ ดังนั้นการกำหนดให้ช่องสัญญาณมีลักษณะคล้ายช่องสัญญาณจริงในการสื่อสารเป็นส่วนสำคัญมากในการวิเคราะห์ประสิทธิภาพของเครื่องรับ โดยธรรมชาติของกระบวนการสื่อสารไร้สาย การแพร่กระจายคลื่นภายในอากาศกรณีที่สภาพแวดล้อมมีสิ่งของวางอยู่โดยรอบหรือมีสิ่งกีดขวางระหว่างการสื่อสาร สัญญาณที่แพร่กระจายไปยังเครื่องรับจะประกอบด้วยหลายเส้นทาง หรืออาจจะเรียกสัญญาณที่เกิดการกระจัดกระจายจากหลายๆทิศทางนี้ว่า การเกิดพหุวิถี (multi-paths) ผลจากการเกิดพหุวิถีนี้ทำให้สัญญาณที่มาถึงเครื่องรับมีผลมาจากสัญญาณมากกว่าหนึ่งทาง ซึ่งในแต่ละทางนั้นจะมีค่าสัมประสิทธิ์การลดทอนที่แตกต่างกันไปทั้งในเชิงแอมพลิจูดและเฟส ทำให้สัญญาณที่ได้รับประกอบไปด้วยผลจากวิถีต่างๆ ถ้าอุปกรณ์ปลายทางกำลังเคลื่อนที่หรือสภาพแวดล้อมรอบๆมีการเปลี่ยนแปลง ผลกระทบจากช่องสัญญาณอาจเปลี่ยนแปลงอย่างสุ่มไปตามเวลา ดังนั้น ณ ขณะหนึ่ง สัญญาณที่รับได้อาจมีการรวมกันแบบหักล้างและในอีกขณะหนึ่งอาจรวมกันแบบเสริม นอกจากการเกิดพหุวิถีขึ้นแล้ว การเกิดปรากฏการณ์ดอปเพลอร์ก็ส่งผลกระทบต่อการสื่อสารของระบบสื่อสารไร้สายด้วย เนื่องจากผลที่ผู้ใช้งานมีการเคลื่อนที่ทำให้คลื่นสัญญาณที่มาถึงมีความถี่ที่เปลี่ยนไป ซึ่งรูปแบบการจางหายที่นิยมใช้กันทั่วไปในการบอกลักษณะของแอมพลิจูดสุ่มที่เป็นผลมาจากช่องสัญญาณพหุวิถีและปรากฏการณ์ดอปเพลอร์ มีอยู่ด้วยกันอยู่ 2 รูปแบบ ได้แก่ การจางหายแบบเรย์ลี (Rayleigh fading) และการจางหายแบบไรเชียน (Ricean fading) ซึ่งในวิทยานิพนธ์ฉบับนี้ จะทำการวิเคราะห์ผลที่มาจากช่องสัญญาณพหุวิถีและ

การเกิดปรากฏการณ์ดอปเพลอร์ โดยใช้ช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก (Additive White Gaussian Noise: AWGN) และช่องสัญญาณการจางหายแบบไรเซียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวกในการวิเคราะห์ประสิทธิภาพของเครื่องถอดรหัสที่สร้างขึ้น

โครงการวิทยานิพนธ์ชิ้นนี้ เป็นงานวิจัยที่นำเอาวิธีการถอดรหัสภายใน สำหรับตัวถอดรหัสคอนโวลูชันภายนอกมาพัฒนา เพื่อสร้างเป็นชิ้นงานต้นแบบ โดยจะทำการจำลองการทำงาน (Simulation) ของตัวถอดรหัสภายในที่ได้สร้างขึ้นมาบนเครื่องคอมพิวเตอร์ว่ามีการทำงานที่ถูกต้องหรือไม่ ถ้าทำงานไม่ถูกต้องก็จะต้องทำการแก้ไขให้ถูกต้องเสียก่อน ก่อนที่จะนำไปทำการโปรแกรมลงชิพบนบอร์ด FPGA เพื่อนำไปสร้างเป็นชิ้นงาน การสร้างชิ้นงานขึ้นมาจริงจะเป็นประโยชน์ในการนำไปพัฒนาตัวถอดรหัสภายใน สำหรับตัวถอดรหัสคอนโวลูชันภายนอกต่อไป

สำหรับประโยชน์ที่คาดว่าจะได้รับจากวิทยานิพนธ์ฉบับนี้ คือ

1. ได้ชิ้นงานต้นแบบของเครื่องถอดรหัสภายในของรหัสคอนคาทีเนตที่สามารถถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ได้

2. ได้องค์ความรู้ใหม่จากการพัฒนาชิ้นงานต้นแบบเครื่องถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์และการออกแบบการเชื่อมต่อระหว่างบอร์ด FPGA กับเครื่องคอมพิวเตอร์

3. ได้ชิ้นงานตัวถอดรหัสภายใน ที่เหมาะสมที่จะนำมาใช้งานร่วมกับตัวถอดรหัสภายนอกแบบเวกเตอร์ซิมโบลดีโคดดิง

4. ตีพิมพ์เผยแพร่ในการประชุมวิชาการระดับนานาชาติที่อยู่ในฐานข้อมูล IEEE Xplore 3 บทความ

วัตถุประสงค์

1. ออกแบบและสร้างชิ้นงานต้นแบบของเครื่องถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์บนบอร์ดอิเล็กทรอนิกส์ FPGA

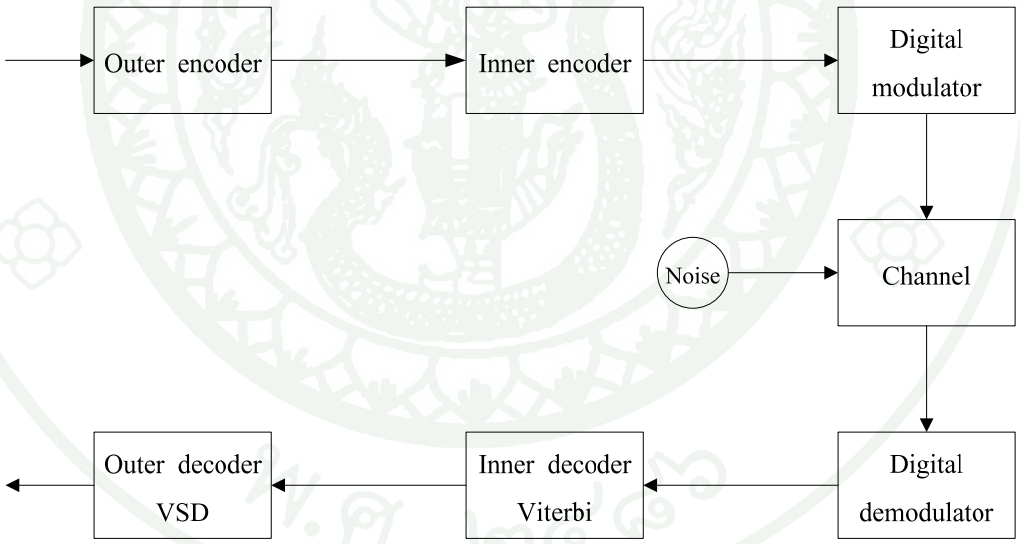
2. หาช่องสัญญาณและพารามิเตอร์ที่เหมาะสมของช่องสัญญาณ

3. เพื่อให้ได้องค์ความรู้ใหม่ จากการพัฒนาชิ้นงานต้นแบบ

การตรวจเอกสาร

บทนำ

ระบบการสื่อสารแบบดิจิทัลเป็นระบบการสื่อสารที่นิยมใช้งานมากในปัจจุบัน เนื่องจากสามารถทนทานต่อสัญญาณรบกวนที่เกิดขึ้นในช่องสัญญาณได้ดี ระบบการสื่อสารแบบดิจิทัลโดยใช้รหัสแบบคอนคาทีเนต (Concatenated code) (Forney, 1966;Lin and Costello, 2004) แสดงดังภาพที่ 1 รหัสคอนคาทีเนตมี 2 ส่วนคือ ส่วนแรกจะเป็นส่วนของรหัสภายนอก (Outer code) เป็นส่วนที่เหมาะกับรหัสแบบนอนไบนารี (Nonbinary code) และส่วนของรหัสภายใน (Inner code) จะเหมาะกับรหัสไบนารี (Binary code) การเข้ารหัส (Encoder) ของรหัสคอนคาทีเนตมี 2 ส่วนคือ ส่วนแรกเป็นการเข้ารหัสภายนอก (Outer encoder) และส่วนการเข้ารหัสภายใน (Inner encoder) สำหรับการถอดรหัสก็มีสองส่วนเช่นกันคือ ส่วนของการถอดรหัสภายใน (Inner decoder) และส่วนของการถอดรหัสภายนอก (Outer decoder) (Proakis, 2001)



ภาพที่ 1 ระบบการสื่อสารโดยใช้รหัสคอนคาทีเนต

โครงงานวิจัยนี้จะให้ความสนใจในการศึกษาและวิจัยในส่วนของการถอดรหัสภายใน (Inner decoder) ซึ่งเป็นส่วนที่นำมาออกแบบและสร้างเป็นชิ้นงานของเครื่องถอดรหัสภายในด้วยบอร์ด FPGA โดยชิ้นงานของเครื่องถอดรหัสภายในของรหัสคอนคาทีเนตที่สร้างขึ้นนี้ จะใช้วิธีการถอดรหัสแบบลิสวิเทอร์บิ (list Viterbi decoding) ซึ่งสามารถถอดรหัสคอนโวลูชันที่ใช้สัญลักษณ์ไบนารีได้ โดยจะให้ผลลัพธ์ที่ควรจะเป็น (most likely decoded sequences) มากที่สุดสอง

ผลลัพธ์ โดยระบุลำดับได้ว่า ผลลัพธ์ที่ได้เป็นผลลัพธ์ที่ควรจะเป็นมากที่สุดและที่ควรจะเป็นรองมา ซึ่งจะแตกต่างจากเครื่องถอดรหัสวิเทอร์บิทั่วไปที่ให้ผลลัพธ์ที่ควรจะเป็นเพียงผลลัพธ์เดียว

การที่เครื่องถอดรหัสภายในสามารถให้ตัวเลือกได้นี้ จะทำให้ตัวถอดรหัสภายนอกแบบเวกเตอร์ซิมโบล (วีเอสดี) มีขั้นตอนการถอดรหัสที่ง่ายลงและแก้ไขความผิดพลาดได้มากขึ้น เนื่องจากตัวถอดรหัสวีเอสดีสามารถนำตัวเลือกที่สองหรือชุดข้อมูลสำรองที่ถูกต้องมาแทนลงในตัวเลือกที่หนึ่งหรือชุดข้อมูลหลักที่พบว่าผิดพลาดได้

ในการสร้างชิ้นงานต้นแบบนี้ จะมีทฤษฎีสำคัญที่เกี่ยวข้องคือ การเข้ารหัสช่องสัญญาณ การเข้ารหัสคอนโวลูชันที่เป็นไบนารี การถอดรหัสคอนโวลูชัน การออกแบบวงจรรวมขนาดใหญ่ การประยุกต์ใช้การถอดรหัสลิสวิเทอร์บิ (List Viterbi Algorithm: LVA) กับวีเอสดีในรหัสคอนคาทีเนต การมอดูเลทสัญญาณ สัญญาณรบกวนและช่องสัญญาณการจางหาย ซึ่งจะกล่าวถึงรายละเอียดของทฤษฎีดังต่อไปนี้

1. การเข้ารหัสช่องสัญญาณ

การเข้ารหัสช่องสัญญาณ (Channel encoding) เป็นส่วนประกอบหนึ่งในระบบสื่อสารดิจิทัล ที่ภาคส่งทำหน้าที่เข้ารหัสข้อมูลให้เป็นคำรหัส ที่ภาครับทำหน้าทำการตรวจสอบความผิดพลาดของข้อมูล และสามารถแก้ไขข้อมูลที่เกิดความผิดพลาดให้เป็นข้อมูลที่ถูกต้องได้ โดยการเข้ารหัสจะจำแนกรหัสออกได้เป็นสองประเภท (Lin and Costello, 2004) คือ

1.1 รหัสบล็อก (Block codes)

การเข้ารหัสบล็อกจะแบ่งบิทข้อมูลที่จะทำการเข้ารหัสออกเป็นกลุ่มหรือที่เรียกว่าบล็อกขนาด k บิท จากนั้นบิทข้อมูลของแต่ละบล็อกก็จะถูกแปลงให้กลายเป็นคำรหัส (Codeword) ที่มีความยาวเท่ากับ n บิท โดยที่ $n > k$ ดังนั้นจึงเรียกการเข้ารหัสนี้ว่า รหัสบล็อก (n, k) รหัสบล็อกเป็นรหัสที่ไม่มีความจำ (Memoryless) กล่าวคือรหัสที่ได้จะไม่มีความสัมพันธ์กันระหว่างข้อมูลชุดปัจจุบันกับชุดก่อนหน้า โดยปกติแล้วชุดรหัสที่ได้จากการเข้ารหัสนั้นจะยังคงประกอบด้วยส่วนของบิทข้อมูลเดิม k บิท และส่วนของบิทพิเศษที่เพิ่มเข้าไปอีก n-k บิท หรือที่เรียกว่า บิตตรวจสอบ (Check bits) เพื่อใช้สำหรับตรวจสอบว่ามีความผิดพลาดในบิทข้อมูลในระหว่างที่ส่งผ่านช่องสัญญาณหรือไม่ ณ ที่ภาครับก็จะมีวงจรที่ทำหน้าที่ถอดรหัส เพื่อดึงบิทข้อมูลเดิมนั้นออกมา

พร้อมกันนั้นก็จะให้ค่าที่เรียกว่า ซินโดรม (Syndrome) ออกมาด้วย โดยค่าซินโดรมนั้นมีไว้สำหรับบ่งบอกว่ามีความผิดพลาดเกิดขึ้นในข้อมูลหรือไม่ หรืออาจใช้ในการบ่งบอกถึงตำแหน่งของบิทที่ผิดด้วย โดยรหัสชนิดนี้ที่รู้จักโดยทั่วไป ได้แก่ รหัสบล็อกแบบเชิงเส้น (Linear Block Code) (Berlekamp, 1984) รหัสแฮมมิ่ง (Hamming, 1950) รหัสวน (Cyclic code) (Prange, 1957) รหัส Reed Solomon (Reed and Solomon, 1960) และรหัส BCH (Hocquenghem, 1959; Bose and Ray-Chaudhuri, 1960) เป็นต้น

1.2 รหัสคอนโวลูชัน (Convolutional codes)

รหัสคอนโวลูชันถูกคิดค้นขึ้นโดย Elias ในปี 1955 (Elias, 1955) เป็นการส่งข้อมูลแบบสัญลักษณ์ที่มีหน่วยความจำ (Memory) โดยจะเรียกได้เป็น (n, k, m) convolutional codes หรือเรียกเป็น k/n convolutional codes โดยที่ n เป็นความยาวของรหัส, k เป็นความยาวของข้อมูล และ m เป็นจำนวนของหน่วยความจำ ตัวอย่างเช่นในงานวิจัยนี้ใช้รหัสเป็น (2, 1, 4) convolutional code ซึ่งหมายถึง ในขั้นตอนการเข้ารหัสในแต่ละคาบนั้นมีข้อมูลต้นกำเนิด 1 สัญลักษณ์ นำมาสนธิกับความจำที่เก็บมาจากชุดข้อมูลก่อนหน้านี้ไม่เกิน 4 คาบ เมื่อเข้ารหัสเสร็จแล้วได้เป็น 2 สัญลักษณ์ของคำรหัส

นอกจากจะจำแนกตามคุณสมบัติมีความจำหรือไม่มีความจำแล้ว เราอาจจำแนกรหัสช่องสัญญาณตามคุณสมบัติอื่นๆได้อีก เช่น

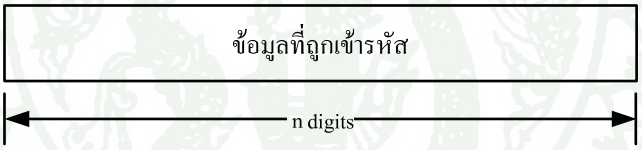
1.2.1 รหัสไบนารี (Binary codes) หมายถึง รหัสที่สัญลักษณ์แต่ละตัวจะเป็นสมาชิกของ GF(2) กล่าวคือ สัญลักษณ์จะมีค่าอยู่ในเซตของ 0 กับ 1 ซึ่งทำให้มีการแทนค่าสัญลักษณ์ของ n, k และ m เป็นจำนวนหนึ่งบิท เช่น (7, 4) binary code หมายถึงรหัสที่มีข้อมูลต้นกำเนิด 4 บิท และคำรหัส 7 บิท เป็นต้น โดยรหัสที่มีใช้เกือบทั้งหมดเป็นรหัสประเภทนี้

1.2.2 รหัสนอนไบนารี (Nonbinary codes) หมายถึง รหัสที่สัญลักษณ์แต่ละตัวจะเป็นสมาชิกของ GF(2^m) โดยที่ $m > 1$ ทำให้มีการแทนค่าสัญลักษณ์ของ n, k และ m เป็นจำนวนมากกว่าหนึ่งบิท ตัวอย่างเช่น Reed-Solomon Code เป็นต้น

1.2.3 รหัสที่เป็นระบบ (Systematic codes) คำรหัสที่เกิดจากรหัสชนิดนี้ จะประกอบไปด้วยสองส่วนคือ ส่วนของข้อมูลต้นกำเนิดจำนวน k ตัว และส่วนของข้อมูลส่วนเกินตรวจสอบ (Redundant checking) จำนวน n-k ตัว (Lin and Castello, 2004)

ภาพที่ 2 รูปแบบของคำรหัสแบบเป็นระบบ

1.2.4 รหัสที่ไม่เป็นระบบ (Nonsystematic codes) คำรหัสของรหัสชนิดนี้จะถูกเข้ารหัสไว้ทั้งหมด ดังนั้นการถอดรหัสจะทำได้ยากกว่ารหัสที่เป็นระบบ เนื่องจากต้องมีการแปลงข้อมูลที่ถูกเข้ารหัสไว้กลับไปเป็นข้อมูลต้นกำเนิดด้วย อย่างไรก็ดี รหัสคอนโวลูชันที่ไม่เป็นระบบมักจะให้ค่าระยะฟรี (d_{free}) มากกว่าแบบเป็นระบบ บางครั้งจึงถูกนำมาใช้งาน



ภาพที่ 3 รูปแบบของคำรหัสแบบไม่เป็นระบบ

2. การเข้ารหัสคอนโวลูชันที่เป็นไบนารี

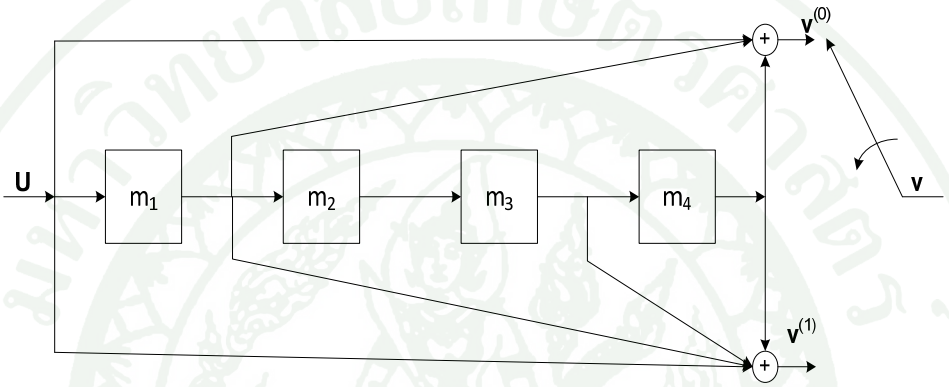
วิธีการเข้ารหัสคอนโวลูชันที่เป็นไบนารี รหัสคอนโวลูชัน (n, k, m) กำหนดให้ n เป็นจำนวนเอาท์พุท, k เป็นจำนวนอินพุท และ m เป็นจำนวนหน่วยความจำต่ออินพุทหรือเป็นจำนวนของชิฟรีจิสเตอร์ (Shift register)

โดยในงานวิจัยนี้ใช้รหัสคอนโวลูชัน (2, 1, 4) ซึ่งมีเมทริกซ์ก่อกำเนิด (Generator matrix) ที่มีฟังก์ชันถ่ายโอน (Tansfer function) เท่ากับ **G**(D) ซึ่งอยู่ในฟิลด์ (Field) ของ GF(2) ดังนี้

$$\mathbf{G}(D) = \left[1 + D + D^4 \quad 1 + D + D^3 + D^4\right] \tag{1}$$

นั่นคือตัวเข้ารหัสจะมีจำนวนเอาท์พุท n = 2, จำนวนอินพุท k = 1 และจำนวนหน่วยความจำ m = 4 และจากสมการที่ (1) เนื่องจาก **G**(D) อยู่ในฟิลด์ของ GF(2) ดังนั้นการดำเนินการทางคณิตศาสตร์ของ **G**(D) กับค่าอื่นๆที่นำมาดำเนินการทางคณิตศาสตร์ จะเป็นแบบมอดูโล-2 (Modulo-2 operation) ทั้งสิ้น

จากสมการ (1) สามารถเขียนเป็นแผนภาพได้ดังนี้



ภาพที่ 4 แผนภาพของตัวเข้ารหัสแบบคอนโวลูชัน (2, 1, 4)

ถ้ากำหนดให้ข้อมูลลำดับเข้าเป็น **u** และเขียนให้อยู่ในรูปสมการ จะได้ดังนี้

$$\mathbf{u} = (u_0 \quad , u_1 \quad , u_2 \quad , \ldots) \tag{2}$$

สำหรับในส่วนของลำดับเอาท์พุทจำนวน 2 เอาท์พุท สามารถเขียนสมการเป็น

$$\begin{aligned} \mathbf{v}^{(0)} &= (v_0^{(0)} \quad , v_1^{(0)} \quad , v_2^{(0)} \quad , \ldots) \\ \mathbf{v}^{(1)} &= (v_0^{(1)} \quad , v_1^{(1)} \quad , v_2^{(1)} \quad , \ldots) \end{aligned} \tag{3}$$

จากลำดับเอาท์พุทจะทำให้ได้คำรหัส **v** ดังสมการต่อไปนี้

$$\mathbf{v} = (v_0^{(0)} v_0^{(1)}, v_1^{(0)} v_1^{(1)}, v_2^{(0)} v_2^{(1)}, \ldots) \tag{4}$$

เนื่องจากรหัสนี้เป็นรหัสเชิงเส้น (Linear code) จึงสามารถหาคำรหัส **v**(D) จากข้อมูลต้นกำเนิด **u**(D) และเมทริกซ์ก่อกำเนิดที่มีฟังก์ชันถ่ายโอน **G**(D) ได้เป็น

$$\mathbf{v}(D) = \mathbf{u}(D)\mathbf{G}(D) \tag{5}$$

แทนค่า **G**(D) ลงในสมการที่ (5) จะได้

$$\mathbf{v}(D) = \left[\mathbf{u}(D)\right]\left[1+D+D^4 \quad 1+D+D^3+D^4\right] \tag{6}$$

จากสมการที่ (6) จะได้

$$\mathbf{v}(D) = \begin{bmatrix} V^{(0)}(D) \\ V^{(1)}(D) \end{bmatrix}^T = \begin{bmatrix} u(D)+u(D)D+u(D)D^4 \\ u(D)+u(D)D+u(D)D^3+u(D)D^4 \end{bmatrix} \tag{7a}$$

หรือ

$$\begin{aligned} v^{(0)}(D) &= u(D)+u(D)D+u(D)D^4 \\ v^{(1)}(D) &= u(D)+u(D)D+u(D)D^3+u(D)D^4 \end{aligned} \tag{7b}$$

กำหนดให้ m_1, m_2, m_3 และ m_4 มีค่าเป็น

$$\begin{aligned} m_1 &= u(D)D \\ m_2 &= u(D)D^2 \\ m_3 &= u(D)D^3 \\ m_4 &= u(D)D^4 \end{aligned} \tag{8}$$

สามารถนำสมการที่ (7b) และ (8) มาเขียนให้สัมพันธ์กับแผนภาพของตัวเข้ารหัสคอนโวลูชัน (2, 1, 4) จะได้ว่า

$$\begin{aligned} v^{(0)} &= u(D)+m_1+m_4 \\ v^{(1)} &= u(D)+m_1+m_3+m_4 \end{aligned} \tag{9}$$

กำหนดให้ลำดับของข้อมูลที่ป้อนเข้าและออกจากวงจรนี้เป็นดังนี้

$$\begin{aligned}\mathbf{u} &= (A_0, A_1, A_2, ..., A_k, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \\ \mathbf{v} &= (v_0^{(0)}v_0^{(1)}, v_1^{(0)}v_1^{(1)}, v_2^{(0)}v_2^{(1)}, ..., v_k^{(0)}v_k^{(1)}, v_{k+1}^{(0)}v_{k+1}^{(1)}, v_{k+2}^{(0)}v_{k+2}^{(1)}, v_{k+3}^{(0)}v_{k+3}^{(1)}, v_{k+4}^{(0)}v_{k+4}^{(1)})\end{aligned} \tag{10}$$

จากสมการที่ (10) ค่า $\mathbf{A}$ และ $\mathbf{v}$ แต่ละตัวเป็นเวกเตอร์ของสัญลักษณ์ขนาด q บิท ซึ่งในงานวิจัยนี้ใช้ q = 32 บิท ส่วนสี่ลำดับสุดท้ายของ $\mathbf{u}$ เป็นการป้อนข้อมูลที่เป็นศูนย์เข้าไปปิดท้ายข้อมูล รวมทั้งเป็นการล้างค่าที่ค้างอยู่ในหน่วยความจำของวงจรเข้ารหัสด้วย โดยจะสามารถแจกแจงลำดับของแต่ละปมของวงจรได้เป็น

$$\begin{aligned}\mathbf{u} &= (A_0, A_1, A_2, ..., A_k, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \\ v^{(0)} &= (v_0^{(0)}, v_1^{(0)}, v_2^{(0)}, ..., v_k^{(0)}, v_{k+1}^{(0)}, v_{k+2}^{(0)}, v_{k+3}^{(0)}, v_{k+4}^{(0)}) \\ v^{(1)} &= (v_0^{(1)}, v_1^{(1)}, v_2^{(1)}, ..., v_k^{(1)}, v_{k+1}^{(1)}, v_{k+2}^{(1)}, v_{k+3}^{(1)}, v_{k+4}^{(1)})\end{aligned} \tag{11}$$

เมื่อป้อนลำดับเหล่านี้ลงในวงจรการเข้ารหัส สามารถเขียนลำดับเหล่านี้ให้อยู่ในรูปสมการที่ (9) โดยที่กำหนดให้ในคาบที่ 1 มีค่าเริ่มต้นของ m_1, m_2, m_3 และ m_4 เป็นศูนย์ จะได้ค่าของคำรหัสเป็นดังนี้

$$\begin{aligned}v_0^{(0)} &= A_0 + m_1 + m_4 = A_0 \\ v_0^{(1)} &= A_0 + m_1 + m_3 + m_4 = A_0\end{aligned} \tag{12}$$

คาบที่ 2 มี $m_1 = A_0$, $m_2 = 0$, $m_3 = 0$ และ $m_4 = 0$ ได้ค่าคำรหัสเป็น

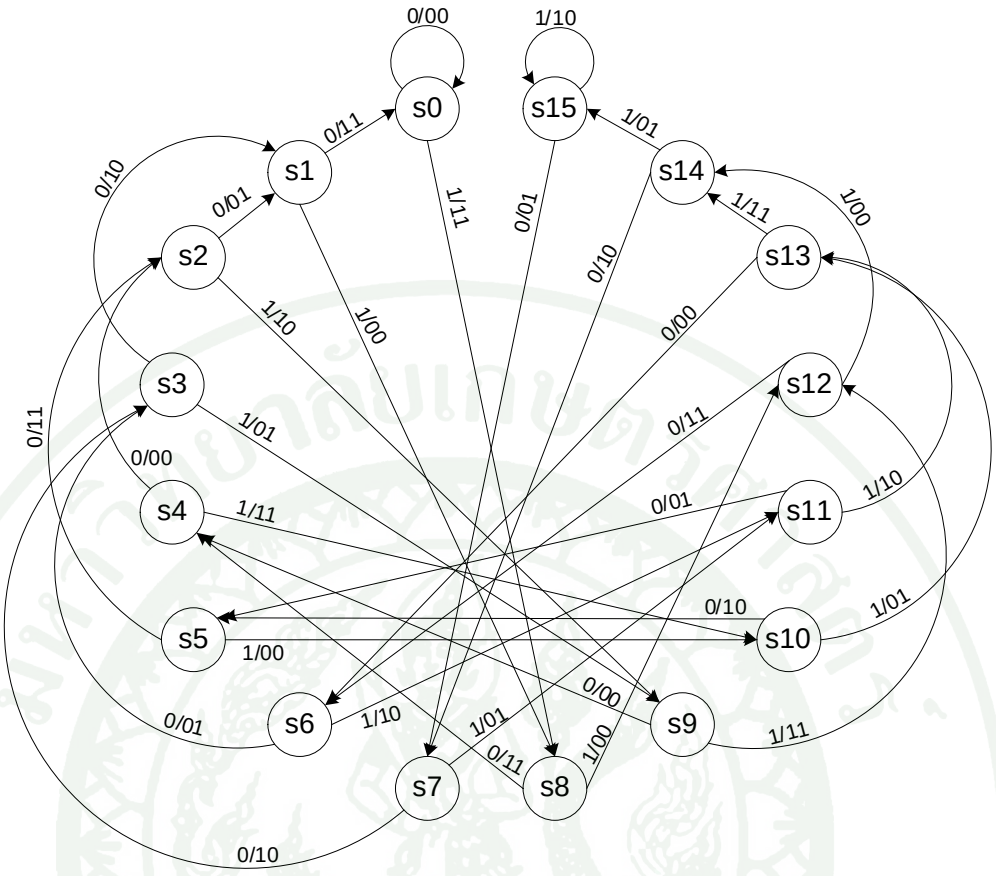
$$\begin{aligned}v_1^{(0)} &= A_1 + A_0 + m_4 = A_1 + A_0 \\ v_1^{(1)} &= A_1 + A_0 + m_3 + m_4 = A_1 + A_0\end{aligned} \tag{13}$$

คาบที่ 3 มี $m_1 = A_1$, $m_2 = A_0$, $m_3 = 0$ และ $m_4 = 0$ ได้ค่าคำรหัสเป็น

$$\begin{aligned}v_2^{(0)} &= A_2 + A_1 + m_4 = A_2 + A_1 \\ v_2^{(1)} &= A_2 + A_1 + m_3 + m_4 = A_2 + A_1\end{aligned} \tag{14}$$

ด้วยวิธีเดียวกันนี้จะได้ค่าคำรหัสอื่นๆ ดังนี้

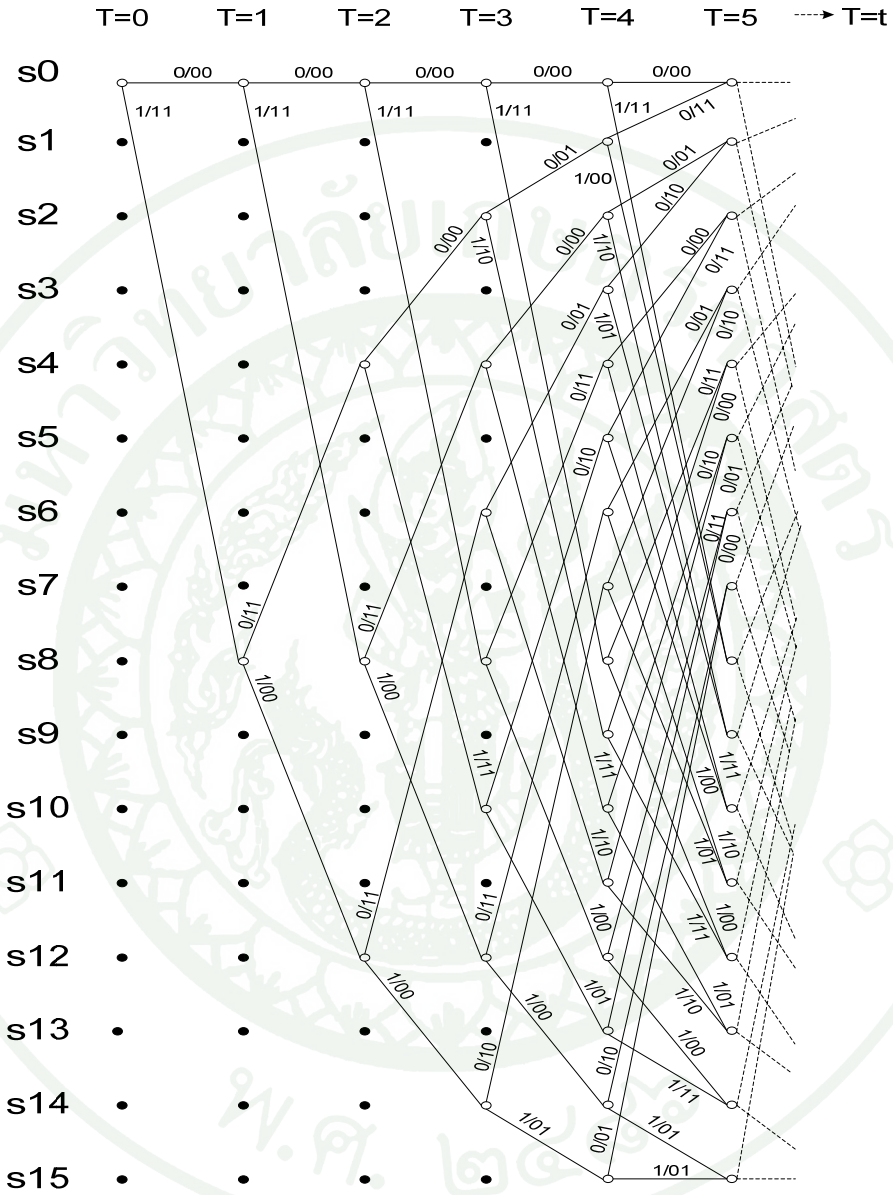
$$
\begin{aligned}
v_3^{(0)} &= A_3 + A_2 + m_4 = A_3 + A_2 \\
v_3^{(1)} &= A_3 + A_2 + A_0 + m_4 = A_3 + A_2 + A_0 \\
v_4^{(0)} &= A_4 + A_3 + A_0 \\
v_4^{(1)} &= A_4 + A_3 + A_1 + A_0 \\
v_5^{(0)} &= A_5 + A_4 + A_1 \\
v_5^{(1)} &= A_5 + A_4 + A_2 + A_1 \\
&\cdots \\
v_k^{(0)} &= A_k + A_{k-1} + A_{k-4} \\
v_k^{(1)} &= A_k + A_{k-1} + A_{k-3} + A_{k-4} \\
v_{k+1}^{(0)} &= \vec{0} + A_k + A_{k-3} = A_k + A_{k-3} \\
v_{k+1}^{(1)} &= \vec{0} + A_k + A_{k-2} + A_{k-3} = A_k + A_{k-2} + A_{k-3} \\
v_{k+2}^{(0)} &= \vec{0} + \vec{0} + A_{k-2} = A_{k-2} \\
v_{k+2}^{(1)} &= \vec{0} + \vec{0} + A_{k-1} + A_{k-2} = A_{k-1} + A_{k-2} \\
v_{k+3}^{(0)} &= \vec{0} + \vec{0} + A_{k-1} = A_{k-1} \\
v_{k+3}^{(1)} &= \vec{0} + \vec{0} + A_k + A_{k-1} = A_k + A_{k-1} \\
v_{k+4}^{(0)} &= \vec{0} + \vec{0} + A_k = A_k \\
v_{k+4}^{(1)} &= \vec{0} + \vec{0} + \vec{0} + A_k = A_k
\end{aligned}
\tag{15}
$$



ภาพที่ 5 แผนภาพสเตทของรหัสคอนโวลูชัน (2, 1, 4)

แผนภาพสเตท (State diagram) คือ แผนภาพแสดงค่าของข้อมูลในชิฟรีจิสเตอร์และเอ้าท์พุทของตัวเข้ารหัส ซึ่งแผนภาพสเตทของตัวเข้ารหัสแบบคอนโวลูชัน (2, 1, 4) แสดงดังภาพที่ 5 ตัวเลขที่อยู่ในวงกลมนั้น หมายถึง สภาวะต่างๆของข้อมูลที่ถูกเก็บไว้ในชิฟรีจิสเตอร์ ในกรณีของตัวเข้ารหัสแบบคอนโวลูชัน (2, 1, 4) ที่ m เท่ากับ 4 จะมีสถานะ (State) เท่ากับ 2^m หรือ 16 สถานะ และสำหรับลูกศรที่ถูกแสดงไว้ในภาพจะแสดงถึงลักษณะการเปลี่ยนแปลงการทำงานจากสถานะหนึ่งไปเป็นอีกสถานะหนึ่ง ซึ่งขึ้นอยู่กับข้อมูลที่ป้อนเข้ามา ณ เวลานั้นๆ นอกจากแสดงการทำงานการเข้ารหัสโดยใช้แผนภาพสเตทในการอธิบายการทำงานแล้วยังมีการแสดงการทำงานของการเข้ารหัสในอีกรูปแบบหนึ่ง โดยการนำข้อมูลที่ถูกแสดงไว้ในแผนภาพสเตทมาแสดงการทำงาน ณ เวลาต่างๆซึ่งเรียกว่า แผนภาพเทรลลิส (Trellis diagram) ซึ่งตัวอย่างของแผนภาพเทรลลิสของการเข้ารหัสจากภาพที่ 5 มีลักษณะดังภาพที่ 6 โดยค่าสถานะอยู่ในแนวตั้งส่วนค่าเวลาอยู่ในแนวนอน ส่วนเส้นทางต่างๆที่อยู่ในแผนภาพเทรลลิสนั้น แสดงถึงลักษณะการเปลี่ยนแปลงสภาวะของวงจร

โดยตัวเลขที่อยู่เหนือทางเดินในแต่ละเส้นทางนั้นแสดงถึงข้อมูลที่ป้อนเข้ามา (k) และข้อมูลที่ถูกส่งออกไป (n) ณ เวลานั้นๆ ซึ่งอยู่ในรูป k/n



ภาพที่ 6 แผนภาพเทรลลิสของการเข้ารหัสคอนโวลูชัน (2, 1, 4)

3. การถอดรหัสคอนโวลูชัน

การถอดรหัสคอนโวลูชัน (Convolutional decoding) นั้น มีขั้นตอนที่ซับซ้อนกว่าวิธีการเข้ารหัสมาก การถอดรหัสคอนโวลูชันมีหลายวิธีด้วยกันแต่วิธีการถอดรหัสที่นิยมใช้กันอย่าง

แพร่หลายคือ การถอดรหัสแบบวิเทอร์บิ การถอดรหัสแบบวิเทอร์บิเป็นวิธีการถอดรหัสที่ให้ประสิทธิภาพสูง ซึ่งลักษณะการทำงานของอัลกอริทึมวิเทอร์บิ (Viterbi Algorithm) จะเป็นแบบ Maximum Likelihood Decoding โดยผลลัพธ์ที่ได้จากการถอดรหัสจะเป็นเส้นทางเพียงเส้นทางเดียวที่มีความน่าจะเป็นสูงสุดจากเส้นทางทั้งหมดในแผนภาพเทรลลิส ซึ่งมีลักษณะเหมือนกับข้อมูลที่ถูกส่งมากที่สุด

3.1 การถอดรหัสแบบวิเทอร์บิ (Viterbi decoding)

การถอดรหัสแบบวิเทอร์บิ ซึ่งถูกคิดค้นขึ้นโดย Viterbi ในปี 1967 (Viterbi, 1967) การถอดรหัสนี้เป็นการถอดรหัสที่เหมาะที่สุด (Optimum receiver) สำหรับการถอดรหัสแบบสัญลักษณ์ไบนารี การถอดรหัสแบบนี้เป็นวิธีการถอดรหัสที่สะดวกในทางปฏิบัติ โดยใช้หลักการของการถอดรหัสเพื่อให้ได้ผลลัพธ์ที่ควรจะเป็นมากที่สุด (Maximum likelihood decoding) สำหรับการถอดรหัสแบบวิเทอร์บินั้น มีรูปแบบสำหรับการถอดรหัสที่ใช้งานอยู่ 2 ลักษณะด้วยกัน ได้แก่ แบบ Hard Decision และ Soft Decision

1. Hard Decision สำหรับการทำงานของการถอดรหัสที่ใช้กระบวนการตัดสินใจแบบ Hard Decision เป็นการพิจารณาข้อมูลที่รับเข้ามา โดยการพิจารณาว่าข้อมูลที่รับเข้ามาในแต่ละบิท มีค่าของข้อมูลเป็น 0 หรือ 1 เท่านั้น

2. Soft Decision การทำงานของการถอดรหัสที่ใช้กระบวนการตัดสินใจแบบ Soft Decision เป็นการพิจารณาถึงข้อมูลที่รับเข้ามาและทำการตัดสินใจระดับของข้อมูลที่รับเข้ามาโดยแบ่งระดับของสัญญาณที่ใช้ในการคำนวณหาค่าเมทริก (Metric) ที่มากกว่า 2 ระดับ ซึ่งผลลัพธ์ที่ได้นั้น จะได้ข้อมูลรายละเอียดของข้อมูลที่ส่งเข้ามามากกว่ากรณีของ Hard Decision ข้อมูลที่ได้จากการตัดสินใจ (Soft output) จะถูกนำมาใช้ในการคำนวณหาค่าเมทริก เพื่อเปรียบเทียบข้อมูลที่รับเข้ามา ณ เวลานั้นๆกับข้อมูลที่อยู่ในเส้นทางต่างๆ ณ เวลานั้น ซึ่งมีรูปแบบที่ใช้ในการคำนวณที่แตกต่างกันไป

3.2 อัลกอริทึมวิเทอร์บิ (Viterbi Algorithm)

การทำงานต่างๆต้องมีการคำนวณหาความแตกต่างระหว่างข้อมูลที่รับเข้ามาและค่าที่อยู่ในเส้นทางต่างๆ เพื่อใช้ในกระบวนการตัดสินใจ โดยกระบวนการที่ใช้ในการหาเส้นทางที่ดี

ที่สุดนั้น จะใช้วิธีการทำงานที่มีชื่อว่า อัลกอริทึมวิเทอร์บิ (Viterbi and Omura, 1971) เป็นกระบวนการที่ใช้ในการค้นหาเส้นทางที่อยู่ในแผนภาพเทรลลิสที่มีลักษณะที่ใกล้เคียงกับข้อมูลที่รับมากที่สุด เพื่อที่จะนำข้อมูลในเส้นทางนั้นมาคำนวณหาค่าของข้อมูลที่ถูกส่งมา การถอดรหัสจะเริ่มจากการพิจารณาแบ่งข้อมูลที่รับเข้ามาออกเป็นข้อมูลย่อยๆจำนวน m ช่วง แต่ละช่วงมีขนาดของข้อมูลเท่ากับ n_0 บิท จากนั้นทำการสร้างแผนภาพเทรลลิสของรหัสคอนโวลูชันที่มีจำนวนสเตทการทำงานเท่ากับ m สเตท พิจารณาเฉพาะเส้นทางที่มีความเป็นไปได้ว่าจะถูกส่งมาเท่านั้น สำหรับขั้นตอนการทำงานหลักๆของการถอดรหัสวิเทอร์บิมีอยู่ด้วยกัน 4 ขั้นตอน (Seshadri and Sundberg, 1994) ก่อนที่จะเริ่มต้นการทำงานจะต้องกำหนดค่าตัวแปรต่างๆและกำหนดค่าเริ่มต้นของค่าเมทริกของกิ่ง (Branch metric) ในสภาวะเริ่มต้นที่มีข้อมูลเป็นศูนย์ทั้งหมด ให้มีค่าเมทริกเท่ากับศูนย์

ก) กำหนดค่าเริ่มต้น (Initialization)

เริ่มต้นการทำงานที่ช่วงเวลา (Time interval) T = 1 โดยทำการคำนวณหาค่าเมทริกของกิ่งระหว่างข้อมูลที่ได้รับชุดที่ 1 กับข้อมูลในเส้นทางการเปลี่ยนแปลงสถานะ หรือหาได้จากผลรวมของค่าบิทเมทริก (bit metric) ในแผนภาพเทรลลิส โดยค่าบิทเมทริกต้องคำนวณมาจากสัญญาณที่ได้รับ โดยอาจเป็นค่าควรจะเป็น (Likelihood function) หรือค่าระยะยูคลีเดียน (Euclidean distance metrics) หรือระยะแฮมมิ่ง (Hamming distance metrics) ก็ได้

จากนั้นคำนวณหาค่าเมทริกของเส้นทาง (Path metric) หรือค่าของเมทริกสะสม โดยสามารถคำนวณหาค่าได้จากการนำค่าของเมทริกเส้นทางก่อนหน้ามารวมกับค่าเมทริกของกิ่งปัจจุบัน เพื่อใช้ในการตัดสินใจเลือกเส้นทางที่เหมาะสมที่สุดในการเปลี่ยนแปลงข้อมูลไปยังสเตทนั้นๆ

ทำการเปรียบเทียบหรือเลือกเส้นทางที่มีค่าเมทริกของเส้นทางที่ดีที่สุดเอาไว้ โดยเส้นทางที่ถูกเลือกนั้น จะเรียกว่า Survivor ซึ่งเป็นเส้นทางที่ถูกเก็บไว้เพื่อทำการคำนวณช่วงเวลาถัดไป และสำหรับเส้นทางอื่นๆที่ไม่ได้ถูกเลือกนั้น จะถูกลบทิ้งออกไปจากระบบการตัดสินใจ

ข) การทำซ้ำ (Recursion)

เป็นการทำงานซ้ำเหมือนข้อ ก) ตั้งแต่ช่วงเวลา T = 2 ถึงช่วงก่อนช่วงเวลาสุดท้าย ($T = T_{สุดท้าย}-1$)

ค) การสิ้นสุดลง (Termination)

เป็นการทำซ้ำเหมือนข้อ ก) ที่ช่วงเวลาสุดท้าย $T_{สุดท้าย}$ สังเกตได้ว่า $T_{สุดท้าย}$ นั้นมาจากจำนวนของบิทอินพุท ซึ่งเท่ากับ n บิท รวมกับจำนวนของขนาดหน่วยความจำ (m)

ง) การตามรอยกลับ (Trace back)

สำหรับในส่วนการตามรอยกลับของเส้นทางที่ดีที่สุดนั้น เริ่มต้นที่ช่วงเวลาสุดท้าย – 1 ($T = T_{สุดท้าย}-1$) ทำการเลือกเส้นทางที่เป็น Survivor ซึ่งเป็นเส้นทางที่ถูกเลือกเหลืออยู่ย้อนกลับไปจนกระทั่งถึงช่วงเวลาเริ่มต้น ($T = 0$) ของการทำงานที่มีสภาวะในการทำงานเป็นศูนย์ทั้งหมด โดยเส้นทางที่ได้นั้นเป็นเส้นทางที่มีลักษณะที่ใกล้เคียงกับข้อมูลที่รับเข้ามามากที่สุดและจะถูกนำไปใช้ในการคำนวณหาข้อมูลทั้งหมดที่อยู่ในเส้นทางส่งออกไป

ในกรณีที่ค่าบิทเมทริก (bit metric) ที่ใช้เป็นค่าควรจะเป็น (likelihood function) เมื่อพิจารณาที่จุดต่อหนึ่งที่ช่วงเวลาหนึ่ง เมื่อมีเส้นทางสองเส้นหรือมากกว่าเข้ามา ตัวถอดรหัสวิเทอร์บิจะเก็บเฉพาะเส้นทางที่มีค่าเมทริกควรจะเป็นสูงที่สุดไว้ และเส้นทางที่เหลือจะถูกตัดทิ้งไป เนื่องจากเส้นทางที่มีค่าเมทริกควรจะเป็นสูงกว่า จะเป็นเส้นทางที่มีโอกาสที่จะเป็นผลลัพธ์ที่ถูกต้องมากกว่า การตัดเส้นทางอื่นทิ้งไป ทำให้จำนวนการคำนวณ และจำนวนหน่วยความจำที่ต้องใช้น้อยลงกว่าการถอดรหัสแบบให้ผลลัพธ์ที่มีค่าควรจะเป็นสูงสุด (maximum likelihood decoding) ทางทฤษฎีมาก จึงทำให้การถอดรหัสวิเทอร์บิเป็นที่นิยม

ในกรณีที่บิทเมทริกที่ใช้มาจากค่า ระยะยูคลีเดียน (Euclidean distance metrics) หรือระยะแฮมมิ่ง (Hamming distance metrics) เส้นทางที่ควรจะเป็นมากที่สุดที่ตัวถอดรหัสจะเก็บไว้คือ เส้นทางที่มีค่าระยะยูคลีเดียนหรือระยะแฮมมิ่งน้อยที่สุด คือเส้นทางมีค่าเมทริกระยะทาง (distance metric) ต่ำที่สุด เพราะถ้าระยะห่างยิ่งน้อยก็แสดงว่า มีความแตกต่างจากผลลัพธ์ที่ควรจะเป็นน้อย ดังนั้นจึงเป็นเส้นทางที่เป็นไปได้มากที่สุดในการถอดรหัส ตัวถอดรหัสวิเทอร์บิจะเก็บค่าเมทริกของเส้นทางที่มีค่าเมทริกระยะทางต่ำที่สุดส่วนเส้นทางที่เหลือจะตัดทิ้ง รายละเอียดการถอดรหัสแบบวิเทอร์บิสามารถศึกษาเพิ่มเติมได้จาก (Viterbi, 1967;Lin and Costello, 2004)

3.3 อัลกอริทึมลิสวิเทอร์บิ (List Viterbi Algorithm: LVA)

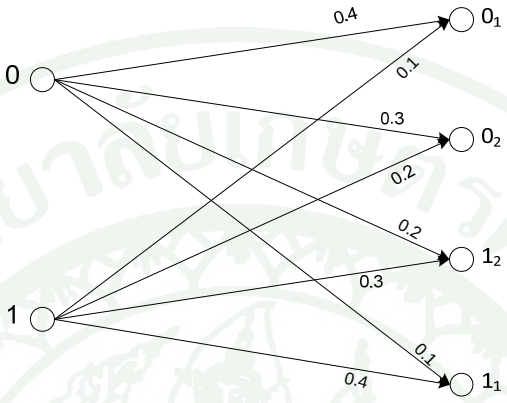
อัลกอริทึมลิสวิเทอร์บิเป็นอัลกอริทึมที่ให้ผลลัพธ์หลายผลลัพธ์ ซึ่ง Seshadri และ Sundbure ได้เสนอแนวคิดของลิสวิเทอร์บิแบบสองผลลัพธ์ในปี 1989 (Seshadri and Sungberg, 1989) และมากกว่าสองผลลัพธ์ในปี 1994 (Seshadri and Sungberg, 1994) ลิสวิเทอร์บิถูกนำไปใช้สำหรับรหัสคอนคาทีเนตในส่วนของตัวถอดรหัสภายใน โดยใช้ในการตรวจสอบความผิดพลาด (Error detection) เท่านั้น

หลักการถอดรหัสแบบลิส (List decoding) โดยทั่วไป จะเป็นการพิจารณาผลลัพธ์ที่เป็นไปได้หลายๆผลลัพธ์ แล้วหาผลลัพธ์ที่ถูกต้อง โดยใช้การตรวจหาความผิดพลาด (error detection) ที่จะระบุเพียงข้อมูลที่ได้รับมีความผิดพลาดหรือไม่ โดยไม่มีการแก้ไข ทำให้ต้องพิจารณาผลลัพธ์ที่เป็นไปได้หลายๆผลลัพธ์ เพื่อให้ผลลัพธ์ที่ถูกต้องไม่ถูกตัดทิ้งไปก่อน

สำหรับ LVA จะมีอยู่ 2 อัลกอริทึมที่ใช้งานกันได้แก่ อัลกอริทึมแรก คือ LVA แบบขนาน (Parallel LVA) ซึ่งเป็นการให้ผลลัพธ์หลายผลลัพธ์ L ทำงานพร้อมกันและให้ผลลัพธ์ที่ควรจะเป็น L พร้อมกันทั้งหมด และอัลกอริทึมที่สอง คือ LVA แบบอนุกรม (Serial LVA) เป็นการให้ผลลัพธ์ที่ควรจะเป็นมากที่สุด k^{th} ทำงานเป็นอันดับแรกจนกระทั่งเสร็จและได้ผลลัพธ์ จากนั้นผลลัพธ์ที่ควรจะเป็นรองลงมา k + 1 และผลลัพธ์อื่นๆก็จะทำงานตามลำดับความน่าจะเป็นไปเรื่อยๆจนกระทั่งเสร็จ

วิธีการถอดรหัสวิเทอร์บิที่ใช้ในโครงการวิจัยนี้ จะใช้วิธีการถอดรหัสแบบลิสวิเทอร์บิ (List Viterbi decoding) ที่ให้ผลลัพธ์ที่ควรจะเป็น (most likely decoded sequences) มากที่สุดสองผลลัพธ์ (Seshadri and Sungberg, 1989) และใช้อัลกอริทึม LVA แบบขนาน เนื่องจากในการนำตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์มาใช้งานร่วมกับตัวถอดรหัสภายนอก เพื่อให้สามารถป้อนข้อมูลให้ตัวถอดรหัสภายนอกได้เร็วขึ้น เพราะการทำงานแบบขนานจะมีความรวดเร็วในการถอดรหัสและเหมาะสมกับการนำไปประยุกต์ใช้งานมากกว่าแบบอนุกรม โดยระบุลำดับได้ว่าผลลัพธ์ที่ได้เป็นผลลัพธ์ที่ควรจะเป็นมากที่สุด และที่ควรจะเป็นรองลงมา หลักการก็คือเก็บค่าเมทริกเส้นทางสองค่าที่ดีสุดที่แต่ละจุดต่อ (Node) และแต่ละช่วงเวลา

<u>ตัวอย่าง</u> เป็นการถอดรหัสแบบลิสวิเทอร์บิแบบสองผลลัพธ์ โดยใช้รหัสคอนโวลูชัน (3,1,2) ซึ่งมีค่า $\mathbf{G}(D) = [(1+D^2) \quad (1+D+D^2)]$ โดยมีบิทข้อมูล $\mathbf{u}$ = (11101) ส่งผ่านช่องสัญญาณในภาพที่ 7 และตารางที่ 1 รายละเอียดเพิ่มเติมศึกษาได้จาก (Lin and Costello, 2004)



ภาพที่ 7 ช่องสัญญาณดีเอ็มซี (Discrete Memoryless Channel; DMC) แบบที่มีอินพุทเป็นไบนารีและสี่เอาท์พุท

ตารางที่ 1 ค่าบิทเมทริกสำหรับช่องสัญญาณของภาพที่ 8 a) ค่าบิทเมทริกที่คำนวณได้ b) ค่าบิทเมทริกที่ใช้ ซึ่งได้ทำการปรับค่าจากข้อ a) ให้เป็นจำนวนเต็มเพื่อให้คำนวณง่าย

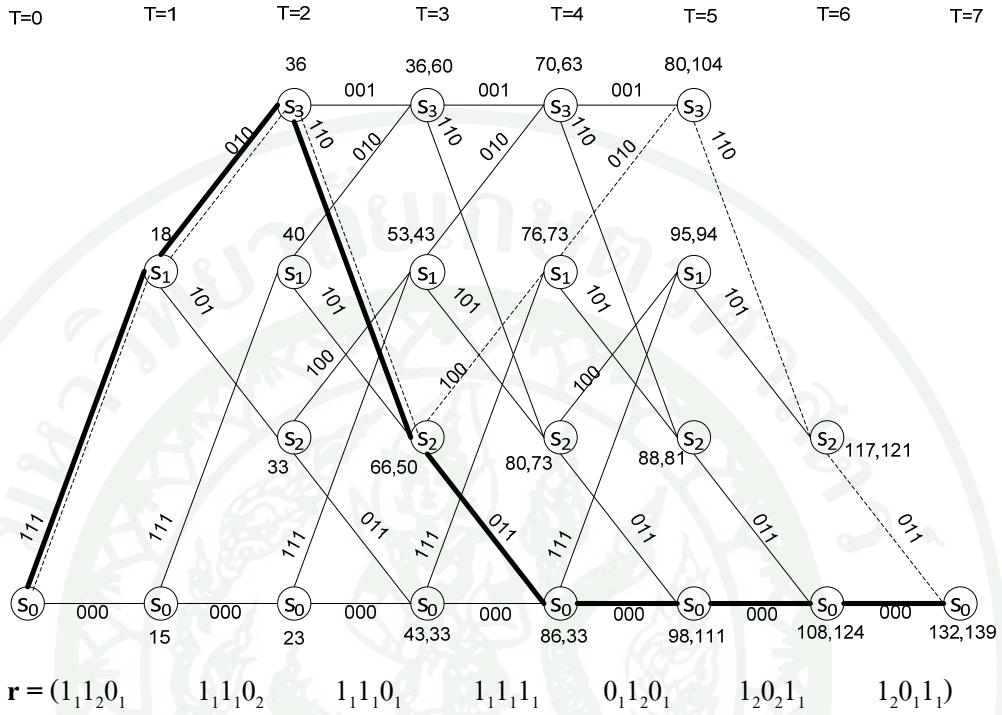
v_1 \ r_1	0_1	0_2	1_1	1_2
0	-0.2	-0.52	-0.7	-1.0
1	-1.0	-0.7	-0.52	-0.2

a)

v_1 \ r_1	0_1	0_2	1_1	1_2
0	10	8	5	0
1	0	5	8	10

b)

ลำดับข้อมูลที่ได้รับ (received sequence) $\mathbf{r} = (1_1 1_2 0_1,\ 1_1 1_1 0_2,\ 1_1 1_1 0_1,\ 1_1 1_1 1_1,\ 0_1 1_2 0_1,\ 1_2 0_2 1_1,\ 1_2 0_1 1_1)$



ภาพที่ 8 ผลลัพธ์ของการถอดรหัสวิเทอร์บิแบบสองผลลัพธ์ สำหรับค่าบิทเมทริกจากตารางที่ 1

เส้นทึบ คือ เส้นทางที่มีความน่าจะเป็นในการถอดรหัสถูกต้องมากที่สุด
เส้นประ คือ เส้นทางที่มีความน่าจะเป็นในการถอดรหัสถูกต้องรองลงมา

ผลลัพธ์ของการถอดรหัสที่มีโอกาสถูกมากที่สุดมีค่าเป็น (111 010 110 011 000 000 000) และรองลงมามีค่าเป็น (111 010 110 100 010 110 011) เมื่อแปลงเป็นข้อมูลหลังจากการถอดรหัสคือ (11000) และ (11011) ตามลำดับ

4. การออกแบบวงจรรวมขนาดใหญ่

จากความก้าวหน้าทางด้านเทคโนโลยีทั้งซอฟต์แวร์และฮาร์ดแวร์ ทำให้การออกแบบวงจรรวมทางด้านดิจิทัลทำได้ง่ายขึ้นมาก และสามารถกระทำได้ในระดับสูงอีกด้วย คือการออกแบบโดยวาดแผนภาพของวงจร (Schematics) โดยอาจเรียกฟังก์ชันสำเร็จรูป (Library) ของวงจรต่างๆมาต่อ

กันได้และที่สำคัญก็คือ การใช้ภาษาอธิบายการทำงานของวงจรแทนการต่อวงจร หรือแทนการทำงานของอุปกรณ์ดิจิทัลที่เรานำมาใช้งาน นอกจากนี้ยังสามารถจำลองการทำงานของวงจรที่ออกแบบจากโปรแกรมที่ช่วยออกแบบได้ทันที และแปลงไฟล์ที่ได้ออกแบบในระดับสูงไปเป็นโครงสร้างการต่อกันของทรานซิสเตอร์หรือเกตย่อยๆ ซึ่งสามารถส่งไปผลิตเป็นไอซีได้ หรือแปลงเป็นข้อมูลที่สามารถนำไปโปรแกรมลงในไอซีประเภท PLD (Programmable Logic Devices) ได้ทันที ขั้นตอนการแปลงจากการออกแบบในขั้นสูงไปเป็นวงจรระดับล่างที่สามารถนำไปใช้งานได้นี้ เรียกว่า การสังเคราะห์ลอจิก (Logic Synthesis)

การออกแบบวงจรดิจิทัลด้วยภาษารวมกับเทคโนโลยีของโปรแกรมสังเคราะห์ลอจิก ทำให้การออกแบบวงจรดิจิทัลที่มีขนาดใหญ่ซับซ้อนเป็นไปได้อย่างมีประสิทธิภาพและรวดเร็วมากขึ้น ถ้าเปรียบเทียบกับวงจรของการพัฒนาซอฟต์แวร์ ก็เหมือนกับการใช้ภาษาระดับสูง (เช่น ภาษาซี หรือปาสคาล) ออกแบบซอฟต์แวร์ แล้วใช้คอมไพล์เลอร์แปลเป็นภาษาแอสเซมบลี แทนที่จะเขียนซอฟต์แวร์เป็นภาษาแอสเซมบลีเอง

Field Programmable Gate Arrays (FPGA) เป็นการออกแบบวงจรรวมประเภท Semi-Custom นั้นคือมีชิปสำเร็จรูปที่มีวงจรบรรจุอยู่ภายในเป็นที่เรียบร้อยแล้ว แต่เป็นวงจรที่ยืดหยุ่น คือสามารถโปรแกรมให้ทำหน้าที่ต่างๆได้ตามต้องการ และสามารถโปรแกรมได้หลายครั้ง การออกแบบวงจรรวมประเภทนี้จะกระทำในระดับสูง คือการเขียนแผนภาพวงจรหรือใช้ภาษาประเภท HDL (Hardware Description Language) ในการระบุฟังก์ชันการทำงานของวงจร ไม่ต้องใช้ Layout เหมือนการออกแบบประเภท Full-Custom ข้อดีของการใช้งาน FPGA คือใช้เวลาในการออกแบบวงจรโดยรวมน้อยลง สามารถแก้ไขเปลี่ยนแปลงวงจรได้ง่าย และไม่จำเป็นต้องทำเป็นจำนวนมากในแต่ละครั้ง ทำให้ต้นทุนต่ำ

ภาษา VHDL (Very high speed integrated circuits Hardware Description Language) เป็นภาษาที่ถูกพัฒนาขึ้นโดยกระทรวงกลาโหมของสหรัฐอเมริกา โดยเป้าหมายก็คือต้องการพัฒนาขีดความสามารถในการออกแบบวงจรรวมให้สูงขึ้น เพื่อที่สามารถรองรับการออกแบบวงจรที่มีความซับซ้อนได้ และต้องการภาษาที่เป็นมาตรฐานหรือเป็นภาษากลางที่เป็นสากล ข้อดีของภาษา VHDL คือ เป็นภาษาที่มีความยืดหยุ่นในการพัฒนา ไม่ยึดติดอยู่กับซอฟต์แวร์ที่ใช้ในการออกแบบ สามารถเปลี่ยนแปลงแก้ไขวงจรได้ง่าย (ชำนาญ และ วัชรากร, 2547)

ขั้นตอนการออกแบบวงจรรวมโดยใช้ FPGA ด้วยโปรแกรมภาษา VHDL มี 7 ขั้นตอน ดังต่อไปนี้

ขั้นตอนแรก ต้องทำการออกแบบ Finite State Machine (FSM) ให้กับวงจรรวมที่จะออกแบบ ซึ่ง FSM เป็นหัวใจสำคัญในการออกแบบหน่วยควบคุม (Control unit) โดยปกติจะบรรยายด้วยแผนภาพไดอะแกรม การออกแบบ FSM จะมี 2 รูปแบบด้วยกันคือ Mealy machine และ Moore machine โดยการออกแบบจะเลือกใช้แบบใดก็ได้หรือจะผสมทั้งสองแบบก็ได้

ขั้นตอนที่สอง ทำการเขียนโปรแกรมด้วยภาษา VHDL พร้อมทั้งกำหนดขาของ FPGA แต่ละขาว่าจะให้ต่อกับอุปกรณ์ใด

ขั้นตอนที่สาม เป็นการนำโปรแกรมที่เขียนเสร็จแล้ว นำมาทำการทดสอบการทำงานโดยการจำลองการทำงาน (Simulation) ซึ่งจะเป็นการใช้โปรแกรมมาจำลองการทำงานบนแผนภาพเวลากับรูปคลื่นสัญญาณ

ขั้นตอนที่สี่ นำโปรแกรมที่ทดสอบการจำลองการทำงานเสร็จแล้ว มาทำการสังเคราะห์วงจร (Synthesize) ซึ่งจะเป็นการตรวจสอบความถูกต้องของภาษาที่เขียน แล้วแปลงเป็นภาษาเครื่องที่เรียกกันว่า Netlist เมื่อเสร็จขั้นตอนนี้จะสามารถดูได้ว่า โปรแกรมที่เขียนจะใช้ LUT (Lookup table) และ Flip-flops จำนวนประมาณเท่าใด

ขั้นตอนที่ห้า เป็นการนำแฟ้ม Netlist ที่ได้จากขั้นตอนที่สี่ มาทำการ Fit หรืออาจเรียกเป็น Implementation ซึ่งจะเป็นการกำหนดการทำงานของ CLBs (Configurable-specific integrated circuit) แต่ละตัว รวมถึงขาสัญญาณภายในและภายนอก FPGA

ขั้นตอนที่หก เป็นขั้นตอนการสร้างแฟ้มไว้สำหรับบันทึกลง FPGA

ขั้นตอนที่เจ็ด นำแฟ้มที่ได้จากขั้นตอนที่หกมาทำการดาวน์โหลดโปรแกรมลง FPGA แล้วทดสอบการทำงานบนบอร์ดทดลอง เป็นอันเสร็จสิ้นกระบวนการออกแบบวงจรรวมโดยใช้ FPGA

5. การประยุกต์ใช้ LVA กับ วีเอสดี ในรหัสคอนคาทีเนต

รหัสคอนคาทีเนต (Concatenated code) (Forney, 1966;Lin and Costello, 2004) เป็นวิธีการสร้างรหัสยาว (Long codes) จากรหัสสั้น (Shorter codes) ทำให้ความซับซ้อนของตัวถอดรหัสลดลง ซึ่งดีกว่าการสร้างรหัสยาวโดยตรง เพราะรหัสยาวจะมีความซับซ้อนของอุปกรณ์มากกว่า รหัสคอนคาทีเนตสามารถถอดรหัสได้เหมือนกับรหัสสั้นและโดยปกติรหัสยาวจะมีประสิทธิภาพที่ดีกว่ารหัสสั้น รหัสคอนคาทีเนตมี 2 ส่วนคือ ส่วนแรกจะเป็นส่วนของรหัสภายนอก (Outer code) เป็นส่วนที่เหมาะกับรหัสแบบนอนไบนารี (Nonbinary code) และส่วนของรหัสภายใน (Inner code) ซึ่งเหมาะกับรหัสไบนารี (Binary code) การเข้ารหัส (Encoder) ของรหัสคอนคาทีเนตมี 2 ส่วนคือ ส่วนแรกเป็นการเข้ารหัสภายนอก (Outer encoder) และส่วนการเข้ารหัสภายใน (Inner encoder) สำหรับการถอดรหัสก็มีสองส่วนเช่นกันคือ ส่วนของการถอดรหัสภายใน(Inner decoder) และส่วนของการถอดรหัสภายนอก (Outer decoder) (Proakis, 2001) แสดงดังภาพที่ 1

Seshadri และ Sundberg (Seshadri and Sundberg, 1994) ศึกษาการใช้ LVA เป็นตัวถอดรหัสภายในของรหัสคอนคาทีเนต โดยมีตัวถอดรหัสภายนอกเป็นรหัสตรวจหาความผิดพลาด (error detecting code) อย่างง่ายๆ อย่างไรก็ตามพบว่า เมื่อลิส (list) มีขนาดใหญ่ขึ้น จะต้องใช้การคำนวณจำนวนมากรวมทั้งขนาดของการจัดเก็บเพิ่มขึ้นด้วย จึงมีความพยายามในการลดความซับซ้อนของ LVA จนกระทั่ง Matzner และ Tuntoolavest (Metzner, 2003; Tuntoolavest and Metzner, 2002) ได้ออกแบบรหัสคอนคาทีเนตขึ้นมาใหม่ แทนการใช้ลิสขนาดใหญ่ที่มีรหัสภายนอกเป็นตัวตรวจหาความผิดพลาด การออกแบบดังกล่าวจะนำมาประยุกต์ใช้กับชิ้นงานต้นแบบ ของการถอดรหัสแบบลิส (List decoding) ที่มีขนาดลิสเล็ก มาเป็นตัวถอดรหัสภายใน โดยมีตัวถอดรหัสภายนอกที่สามารถแก้ไขความผิดพลาด เรียกว่า เวกเตอร์ซิมโบลดีโคดดิง (Vector Symbol Decoding; VSD) (Metzner and Kapturowski, 1990; Haslach and Vinck, 1999; Metzner, 2003) Tuntoolavest ได้แสดงให้เห็นแล้วว่า ตัวเลือกที่เป็นแบบสองผลลัพธ์นั้น เหมาะสมที่สุดสำหรับการนำไปใช้งานในการถอดรหัส เนื่องจากการเพิ่มขนาดของลิสนั้นเกิดผลดีขึ้นเพียงเล็กน้อยเท่านั้นสำหรับการออกแบบรหัสคอนคาทีเนตที่ใช้ LVA กับ วีเอสดี (Tuntoolavest, 2009)

6. การมอดูเลทสัญญาณ (Signal modulation)

เมื่อต้องการส่งสัญญาณเสียงหรือข้อมูลผ่านช่องทางการสื่อสาร จำเป็นอย่างยิ่งที่ต้องอาศัยพลังงานไฟฟ้าช่วยพาสัญญาณเหล่านั้นให้เคลื่อนย้ายจากที่หนึ่งไปยังอีกที่หนึ่ง ขบวนการหรือ

ขั้นตอนในการเพิ่มพลังงานไฟฟ้าดังกล่าวเรียกว่า การมอดูเลท (Modulate) พลังงานไฟฟ้า ซึ่งมีความถี่สูงและคงที่รวมทั้งมีแอมพลิจูด (ขนาด) สูงด้วยนั้น เรียกว่า สัญญาณคลื่นพาห์ (Signal Carrier)

อุปกรณ์สำหรับมอดูเลทสัญญาณ (Modulator) ทำได้โดยการสร้างสัญญาณคลื่นพาห์และรวมเข้ากับสัญญาณข้อมูล เพื่อให้สัญญาณมีความแรงพอที่จะส่งผ่านสื่อกลางไปยังอีกจุดหนึ่งที่อยู่ไกลออกไปได้ และเมื่อถึงปลายทางก็จะมีอุปกรณ์ทำหน้าที่แยกสัญญาณคลื่นพาห์ออกให้เหลือเพียงสัญญาณข้อมูล เรียกวิธีการแยกสัญญาณนี้ว่า การดีมอดูเลท (Demodulation)

การมอดูเลทสัญญาณเป็นเรื่องที่สำคัญมากในการสื่อสารข้อมูล เพราะการเลือกวิธีการมอดูเลทและการดีมอดูเลทที่เหมาะสมจะช่วยให้การส่งข้อมูลข่าวสารมีประสิทธิภาพสูง

การมอดูเลทสัญญาณแบบดิจิทัล (Digital modulation) เป็นการมอดูเลทสัญญาณดิจิทัลเข้ากับคลื่นพาห์ที่เป็นสัญญาณไซน์นั้น มีอยู่หลายรูปแบบ ทั้งนี้ก็เพื่อต้องการให้สัญญาณดิจิทัลเหล่านั้น สามารถส่งผ่านตัวกลางที่ออกแบบมาสำหรับสัญญาณแบบแอนะล็อกได้ เช่น โครงข่ายโทรศัพท์พื้นฐาน ไมโครเวฟ เป็นต้น การมอดูเลทที่ใช้กันทั่วไปได้แก่ การมอดูเลทแบบดิจิทัลทางแอมพลิจูด, การมอดูเลทแบบดิจิทัลทางความถี่, การมอดูเลทแบบดิจิทัลทางเฟสและการมอดูเลทแบบ QAM เป็นต้น สำหรับโครงงานวิจัยนี้ จะใช้การมอดูเลทสัญญาณแบบดิจิทัล 2 แบบ คือ

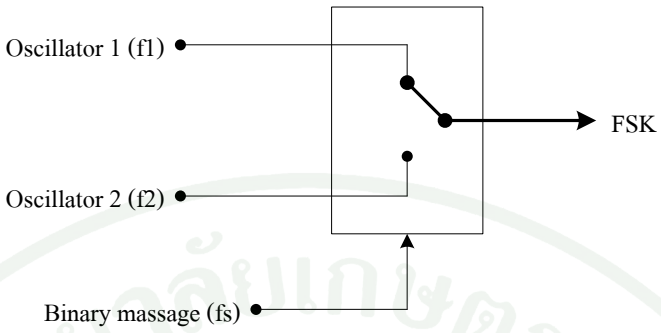
1. การมอดูเลทแบบดิจิทัลทางความถี่ (FSK: Frequency Shift Keying)

การมอดูเลทแบบดิจิทัลทางความถี่เป็นการเปลี่ยนค่าขนาดความถี่ของสัญญาณพาห์คลื่นไซน์ตามบิทข้อมูล

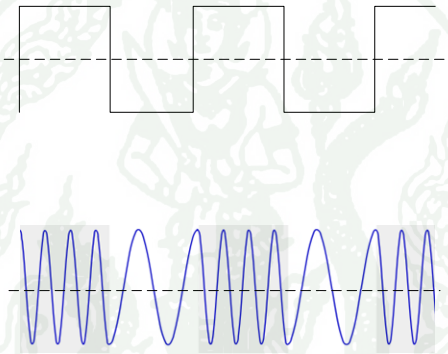
ค่าบิทข้อมูลเป็น ‘0’ ให้ความถี่ของสัญญาณพาห์เท่ากับ f1
ค่าบิทข้อมูลเป็น ‘1’ ให้ความถี่ของสัญญาณพาห์เท่ากับ f2

FSK สามารถทนต่อสัญญาณรบกวนได้ดีกว่า ASK เนื่องจากอุปกรณ์ที่รับข้อมูลไม่ต้องสนใจว่าแอมพลิจูดหรือแรงดันไฟฟ้ามีค่าเปลี่ยนแปลงอย่างไร จะพิจารณาเฉพาะความถี่เท่านั้น แต่ FSK มีข้อเสียที่ สื่อที่ใช้ในการส่งสัญญาณเท่านั้น ซึ่งต้องมีแบนด์วิธที่เพียงพอที่สามารถส่งผ่านความถี่ของสัญญาณที่ต้องการได้

ตัวอย่างของการมอดูเลทแบบ FSK ดังแสดงในภาพที่ 9 และ 10



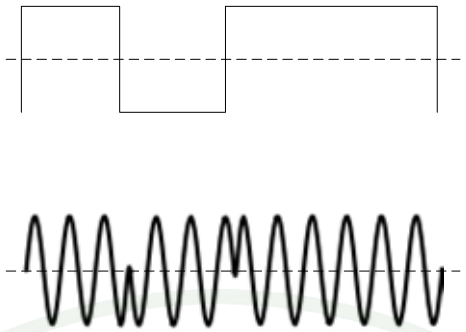
ภาพที่ 9 แสดงสัญญาณข้อมูลดิจิทัลแบบ FSK



ภาพที่ 10 สัญญาณการมอดูเลทแบบดิจิทัลด้วยเทคนิค FSK

2. การมอดูเลทแบบดิจิทัลทางเฟส (PSK: Phase Shift Keying)

หลักการมอดูเลทแบบ PSK คือ ค่าของขนาดและความถี่ของคลื่นพาห์จะไม่มีการเปลี่ยนแปลง แต่ที่จะเปลี่ยนคือเฟสของสัญญาณ กล่าวคือเมื่อมีการเปลี่ยนแปลงสภาวะของบิทจาก ‘1’ ไปเป็น ‘0’ หรือเปลี่ยนจาก ‘0’ ไปเป็น ‘1’ เฟสของคลื่นจะเปลี่ยน (Shift) ไป 180 องศาด้วย ดังภาพที่ 11



ภาพที่ 11 การมอดูเลทแบบดิจิทัลด้วยเทคนิค PSK

การมอดูเลทแบบ PSK สามารถทำได้ทั้งแบบ 4 เฟส (0, 90, 180 และ 270 องศา) และแบบ 8 เฟส (0, 45, 90, 135, 180, 225, 270 และ 315 องศา) เป็นต้น

PSK สามารถทนต่อสัญญาณรบกวนได้ดีกว่า ASK และมีปัญหาในเรื่องของแบนด์วิดท์น้อยกว่า FSK หมายความว่า ถ้ามีสัญญาณรบกวนไม่มากนักจะไม่มีผลถึงกับทำให้อุปกรณ์ที่รับข้อมูลได้รับข้อมูลที่ผิดพลาดไป จึงสามารถเพิ่มประสิทธิภาพของการส่งได้แทนที่จะให้การเปลี่ยนแปลงของเฟสหนึ่งครั้ง หมายถึงข้อมูลเพียงบิทเดียว แต่สามรถเปลี่ยนแปลงเฟสแทนด้วยบิทข้อมูล 2 บิทได้

7. สัญญาณรบกวน (Noise)

สัญญาณรบกวนเป็นสิ่งที่เกิดขึ้นเสมอในระบบสื่อสาร และเกิดขึ้นอย่างหลีกเลี่ยงไม่ได้ ส่วนใหญ่สัญญาณรบกวนเกิดจากชิ้นส่วนอิเล็กทรอนิกส์ที่ใช้อยู่ เช่น ความต้านทาน ทรานซิสเตอร์ และไดโอด เป็นต้น ล้วนเป็นต้นกำเนิดของสัญญาณรบกวนทั้งสิ้น นอกจากนี้สัญญาณรบกวนยังเกิดจากการรบกวนจากภายนอก (Interference) ด้วย

ช่องสัญญาณของระบบสื่อสารที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก เป็นช่องสัญญาณสื่อสารพื้นฐานที่ใช้ในการออกแบบและทดสอบอุปกรณ์สื่อสารโดยทั่วไป เช่น ใช้ทดสอบประสิทธิภาพชุดเข้ารหัสคอนโวลูชันและถอดรหัสวิเทอร์บิ เป็นต้น

สัญญาณรบกวนเกาส์เซียนขาวแบบบวก (Additive White Gaussian Noise: AWGN) เป็นโมเดลของช่องสัญญาณที่ทำให้ประสิทธิภาพของการสื่อสารลดลง ซึ่งเกิดจากการบวกเชิงเส้น

(linear additive) ของสัญญาณรบกวนขาว (white noise) ที่มีความหนาแน่นสเปกทรัล (spectral density) คงที่ กับการแจกแจงแบบเกาส์เซียน (Gaussian distribution) ของแอมพลิจูด โดยสามารถเกิดขึ้นได้ในทุกช่วงความถี่ของช่องสัญญาณของระบบสื่อสาร

สัญญาณข้อมูลกับสัญญาณรบกวนไม่มีสหสัมพันธ์ระหว่างกัน กล่าวคือ ถ้ารูปแบบของช่องสัญญาณเป็น

$$r(t) = s(t) + n(t) \tag{16}$$

โดย $r(t)$ คือ สัญญาณที่รับได้ $s(t)$ คือ สัญญาณที่ถูกส่งไป และ $n(t)$ คือ สัญญาณรบกวนเกาส์เซียนขาวแบบบวก ซึ่งสร้างจากฟังก์ชันความหนาแน่นของความน่าจะเป็น (Probability Density Function: PDF) ที่ค่าเฉลี่ย (Mean) เป็นศูนย์ ดังสมการ (Proakis, 2001)

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \tag{17}$$

โดย σ^2 คือ ค่าความแปรปรวน (Variant), σ คือ ค่าเบี่ยงเบนมาตรฐาน และ x คือ ค่าตัวแปรสุ่ม (Random variable)

8. ช่องสัญญาณการจางหาย (Fading Channel)

8.1 ช่องสัญญาณ (Channel)

ช่องสัญญาณ หมายถึง ตัวกลางที่ให้สัญญาณผ่านจากสายอากาศเครื่องส่งไปยังสายอากาศเครื่องรับ ซึ่งมีความเป็นไปได้หลายรูปแบบ ทั้งที่เป็นอากาศ เป็นสายทองแดง หรือใยแก้วนำแสง โดยที่ช่องสัญญาณจะทำหน้าที่ในการแปลงสัญญาณอินพุทชุดหนึ่ง ให้เป็นสัญญาณเอาท์พุทชุดหนึ่ง และในระบบการสื่อสารไร้สายคลื่นสัญญาณที่ถูกส่งออกมาทางเครื่องส่ง จะไม่ได้เดินทางมาถึงยังเครื่องรับปลายทางเป็นแนวเส้นตรง เพราะต้องพบกับสิ่งกีดขวางในสภาพแวดล้อมที่สัญญาณต้องเคลื่อนที่ผ่าน โดยคลื่นสัญญาณที่มาถึงเครื่องรับ จะเกิดจากการรวมกันของคลื่นหลายวิถีที่มาจากหลายทิศทาง ซึ่งเกิดจากการสะท้อน (Reflection) การเลี้ยวเบน (Diffraction) และการกระจัดกระจาย (Scattering) ผ่านสิ่งกีดขวางต่างๆ เช่น สิ่งก่อสร้าง ต้นไม้ ยานพาหนะ เป็นต้น

โดยเรียกปรากฏการณ์นี้ว่า การเกิดพหุวิถี (Multi path) และผลจากการเกิดพหุวิถีนี้ทำให้สัญญาณที่มาถึงเครื่องรับ มีผลมาจากสัญญาณมากกว่าหนึ่งทาง ซึ่งในแต่ละทางนั้นจะมีค่าสัมประสิทธิ์การลดทอนที่แตกต่างกันไป ทั้งในเชิงแอมพลิจูดและเฟส สัญญาณที่ได้รับประกอบไปด้วยผลจากวิถีต่างๆ โดยสัญญาณในแต่ละทางอาจเขียนให้อยู่ในรูปของเวกเตอร์ของแอมพลิจูดและเฟสได้ ถ้าอุปกรณ์ปลายทางกำลังเคลื่อนที่หรือสภาพแวดล้อมรอบๆมีการเปลี่ยนแปลง ผลกระทบจากช่องสัญญาณอาจเปลี่ยนแปลงอย่างสุ่มไปตามเวลา ดังนั้น ณ ขณะหนึ่ง สัญญาณที่รับได้อาจมีการรวมกันแบบหักล้างและในอีกขณะหนึ่ง อาจรวมกันแบบเสริม ซึ่งรูปแบบของการกระจายตัวที่ใช้กันทั่วไปในการบอกลักษณะของแอมพลิจูดสุ่มที่เป็นผลมาจากช่องสัญญาณพหุวิถี มีอยู่ด้วยกันอยู่ 2 รูปแบบ ได้แก่ การจางหายแบบเรย์ลี (Rayleigh fading) และการจางหายแบบไรเชียน (Ricean fading)

8.2 ดอปเพลอร์ (Doppler)

นอกจากการเกิดพหุวิถีขึ้นแล้ว การเกิดปรากฏการณ์ดอปเพลอร์ (Rappaport, 1995) ก็ส่งผลกระทบต่อการสื่อสารของระบบสื่อสารไร้สายด้วย เนื่องจากผลที่ผู้ใช้งานมีการเคลื่อนที่ทำให้คลื่นสัญญาณที่มาถึงมีความถี่ที่เปลี่ยนไป โดยมุมของสัญญาณที่มาถึง (Angle of arrival: α_n) ถูกนิยามให้เป็นมุมระหว่างคลื่นสัญญาณที่มาถึงวิถีที่ n และทิศทางเคลื่อนที่ของผู้ใช้งาน ดังแสดงในภาพที่ 12 และค่าความถี่ดอปเพลอร์ของคลื่นสัญญาณวิถีที่ n มีค่าดังนี้

$$f_n = f_{\max} \cos \alpha_n \tag{18}$$

เมื่อ $f_{\max}$ คือ ค่าความถี่ดอปเพลอร์สูงสุด ซึ่งขึ้นอยู่กับความเร็วของผู้ใช้งาน (v) และค่าความถี่กลางที่ใช้ในการส่งข้อมูลดังสมการที่ (18)

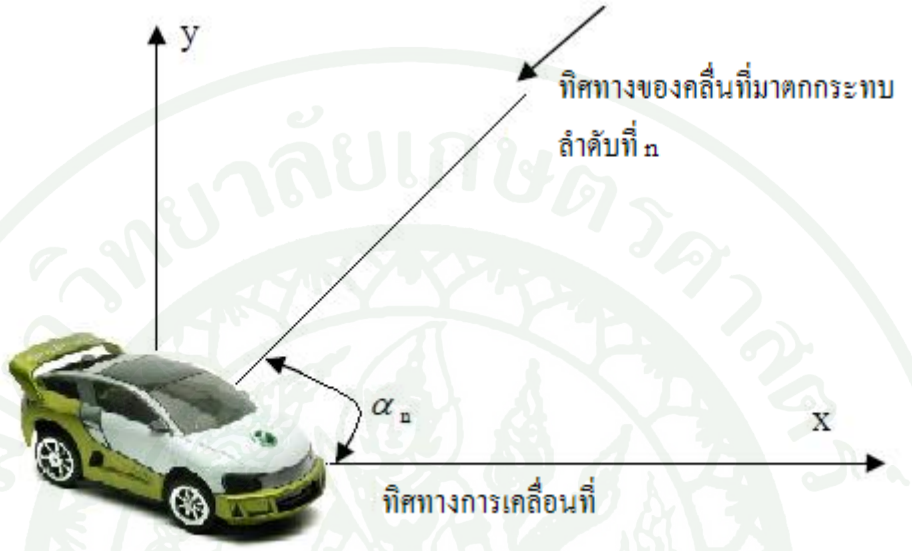
$$f_{\max} = \frac{v}{c_0} f_0 \tag{19}$$

เมื่อ f_0 คือ ความถี่คลื่นสัญญาณพาห์

c_0 คือ ความเร็วแสง มีค่าเท่ากับ 3×10^8 เมตรต่อวินาที

v คือ ความเร็วสัมพัทธ์ระหว่างเครื่องรับส่ง (เมตรต่อวินาที)

เนื่องจากผลของปรากฏการณ์ดอปเพลอร์นี้ ทำให้สเปกตรัมความถี่ของสัญญาณที่ถูกส่งนั้นกระจายออกระหว่างการส่งข้อมูล เมื่อพิจารณาผลเชิงเวลาพบว่า จะทำให้ผลตอบสนองอิมพัลส์ (Impulse response) ของช่องสัญญาณมีการเปลี่ยนแปลงทางเวลา



ภาพที่ 12 แสดงมุม α_n ของคลื่นสัญญาณที่มาถึงของปรากฏการณ์ดอปเพลอร์

8.3 การจางหายของสัญญาณ (Fading)

ในระบบการสื่อสารแบบไร้สาย โดยเฉพาะในระบบโทรศัพท์เคลื่อนที่ ช่องสัญญาณที่เกิดขึ้นมีลักษณะเชิงสุ่ม (Random) และเปลี่ยนแปลงไปตามเวลา เนื่องจากการส่งสัญญาณผ่านช่องสัญญาณไร้สายระหว่างโทรศัพท์เคลื่อนที่กับสถานีฐาน เกิดขึ้นสูงจากพื้นดินไม่มากนัก ดังนั้นสัญญาณที่ส่งอาจเกิดการสะท้อนกับสิ่งกีดขวางที่อยู่ในบริเวณนั้น เช่น อาคาร ต้นไม้ พื้นดิน เป็นต้น ส่งผลให้สัญญาณที่ได้รับปลายทางประกอบไปด้วยสัญญาณสะท้อนจากหลากหลายเส้นทาง ซึ่งมีขนาดและเฟสที่แตกต่างกันออกไป นอกจากนี้การเคลื่อนที่ของสถานีส่ง ขณะที่มีการส่งสัญญาณหรือการที่สภาพแวดล้อมที่อยู่ระหว่างภาคส่งและภาครับมีการเปลี่ยนแปลงไปตามเวลา เช่น การเคลื่อนที่ของรถยนต์ที่อยู่บริเวณรอบๆเครื่องส่งก็มีผลต่อสัญญาณปลายทางที่ได้รับด้วยเช่นกัน ปัจจัยต่างๆที่กล่าวมาข้างต้นนี้ ส่งผลให้สัญญาณที่ได้รับปลายทาง มีการเปลี่ยนแปลงขึ้นลงอย่างรวดเร็ว ทั้งในแง่ของขนาดแอมพลิจูดและเฟสของสัญญาณ ซึ่งเรียกปรากฏการณ์นี้ว่า การจางหายของสัญญาณ (Small scaled fading) หรือเฟดดิง (Fading) (Rappaport, 1995) ในกรณีที่ช่องสัญญาณไร้สายมีสัญญาณสะท้อนจากทิศทางต่างๆจำนวนมาก แต่ไม่มีสัญญาณที่มาจาก

เส้นทางตรง (Line of sight: LOS) ระหว่างภาคส่งกับภาครับ จะเรียกการจางหายที่เกิดขึ้นนี้ว่า เรย์ลีเฟดดิง (Rayleigh fading) เนื่องจากสภาพของเอนเวโลป (Envelop) ของสัญญาณที่ได้รับมีการกระจายตัวทางสถิติเป็นแบบเรย์ลี

8.4 ปัจจัยหลักที่ส่งผลต่อการเกิดการจางหาย

ปัจจัยหลักที่ก่อให้เกิดการจางหาย มีอยู่ด้วยกัน 2 ประการ คือ

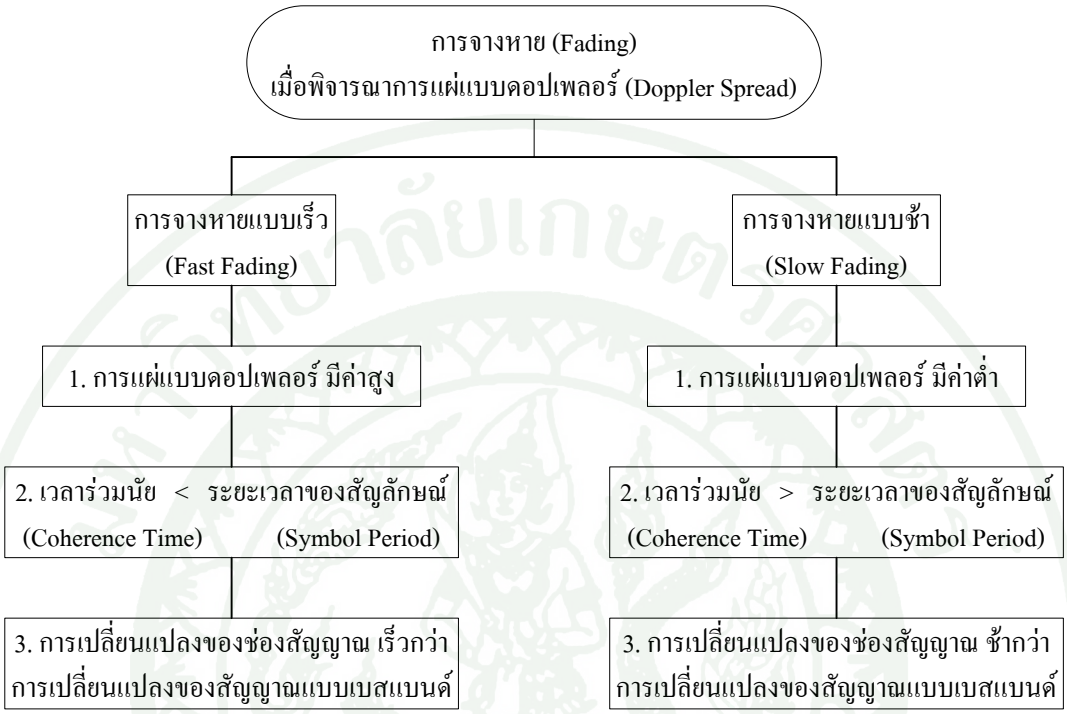
8.4.1 การแผ่แบบประวิงเวลา (Delay spread)

เนื่องจากสัญญาณที่ถูกส่งมาจากต้นทาง เมื่อไปกระทบกับสิ่งกีดขวางที่อยู่ระหว่างภาคส่งกับภาครับ จะเกิดการสะท้อนและหักเห ทำให้สัญญาณที่ได้รับปลายทางประกอบด้วยสัญญาณสะท้อนจากหลายเส้นทาง เมื่อมาถึงยังปลายทางในเวลาที่แตกต่างกันทำให้สัญญาณรวมที่ปลายทางเป็นสัญญาณที่มีการประวิงเวลา หรืออาจเรียกได้ว่า สัญญาณเกิดการแผ่ทางเวลา (Time spread) ผลของการประวิงเวลานั้นทำให้การเดินทางไปยังปลายทางของสัญญาณใช้เวลานานกว่าปกติ ก่อให้เกิดการรบกวนกันของสัญญาณในแต่ละสัญลักษณ์ หรือการแทรกสอดระหว่างสัญลักษณ์ ทั้งนี้การประวิงเวลาของสัญญาณจะเกิดขึ้นมากหรือน้อยขึ้นอยู่กับลักษณะของช่องสัญญาณ เมื่อพิจารณาลักษณะของการจางหายจากการแผ่แบบประวิงเวลา จะได้การจางหาย 2 รูปแบบคือ การจางหายแบบแนวราบ (Flat fading) และการจางหายแบบเลือกความถี่ (Frequency selective fading)

8.4.2 การแผ่แบบดอปเพลอร์ (Doppler spread)

การเคลื่อนที่สัมพัทธ์ระหว่างสถานีเคลื่อนที่กับสถานีฐาน ส่งผลให้สัญญาณที่เดินทางมาในแต่ละเส้นทางเกิดการเลื่อนทางความถี่ เรียกว่า การเลื่อนความถี่แบบดอปเพลอร์ (Doppler shift) โดยความถี่ที่เลื่อนไปมีค่าเป็นบวกหรือลบมากน้อยเพียงใด ขึ้นอยู่กับทิศทางและความเร็วในการเคลื่อนที่ของสถานีโมบายล์ด้วย นอกจากนี้การเคลื่อนที่ของวัตถุที่อยู่บริเวณรอบๆ สถานีโมบายล์ก็ส่งผลให้เกิดการเลื่อนความถี่แบบดอปเพลอร์ที่มีการเปลี่ยนแปลงตามเวลาเหมือนกัน ดังนั้นการแผ่แบบดอปเพลอร์นี้ ทำให้ช่องสัญญาณมีพฤติกรรมเปลี่ยนแปลงไปตามเวลา (Time varying channel) และอัตราการเปลี่ยนแปลงที่เกิดขึ้นจะส่งผลโดยตรงต่อความเร็วของการเกิดการจางหายด้วย เมื่อพิจารณาลักษณะของการแผ่แบบดอปเพลอร์จะส่งผลกระทบต่อการจาง

หายอีก 2 รูปแบบคือ การจางหายแบบเร็ว (Fast fading) และการจางหายแบบช้า (Slow fading) ดังแสดงในภาพที่ 13



ภาพที่ 13 แสดงรูปแบบของการจางหาย เมื่อพิจารณาจากการแผ่แบบดอปเพลอร์

ที่มา: Rappaport (1995)

ก) การจางหายแบบเร็ว (Fast fading)

การแผ่แบบดอปเพลอร์และเวลาร่วมนัย (Coherence time) เป็นพารามิเตอร์ที่ใช้บ่งบอกถึงคุณสมบัติการเปลี่ยนแปลงตามเวลาของช่องสัญญาณ ซึ่งมีผลมาจากการเคลื่อนที่ของโทรศัพท์เคลื่อนที่ เวลาร่วมนัย คือ ช่วงเวลาทางสถิติที่ผลตอบสนองต่อช่องสัญญาณมีค่าไม่เปลี่ยน ทั้งยังเป็นค่าที่บอกให้ทราบถึงความคล้ายคลึงกันของผลตอบสนองช่องสัญญาณในช่วงเวลาหนึ่งอีกด้วย กล่าวคือ สัญญาณที่มาถึงภาครับที่เวลาต่างกันแต่ไม่เกินเวลาร่วมนัย จะได้รับผลกระทบจากช่องสัญญาณใกล้เคียงกัน

ในกรณีของการจางหายแบบเร็ว ผลตอบสนองช่องสัญญาณมีการเปลี่ยนแปลงอย่างรวดเร็วภายในช่วงเวลาที่ส่งสัญญาณ ดังนั้นเวลาร่วมนัยของช่องสัญญาณ มีค่าน้อยกว่าช่วงเวลาของสัญลักษณ์และคุณลักษณะของการจางหายจะเปลี่ยนแปลงไปมาหลายครั้งในขณะที่สัญลักษณ์หนึ่งๆถูกส่งไป ส่งผลให้รูปร่างของสัญญาณเบสแบนด์ผิดเพี้ยนไป

ข) การจางหายแบบช้า (Slow fading)

การจางหายแบบช้า เกิดขึ้นเมื่ออัตราการเปลี่ยนแปลงของผลตอบสนองช่องสัญญาณมีค่าน้อยกว่าอัตราการเปลี่ยนแปลงของสัญญาณ หรือเวลาร่วมนัยมีค่ามากกว่าเวลาของสัญลักษณ์ ในกรณีนี้ช่องสัญญาณจะมีผลตอบสนองคงที่ภายในช่วงเวลาหลายสัญลักษณ์ ทำให้ได้รับผลกระทบจากช่องสัญญาณติดกันเป็นช่วงยาว

8.5 การจางหายแบบเรย์ลี หรือเรย์ลีเฟดดิง (Rayleigh Fading)

ถ้าในสัญญาณที่ได้รับไม่มีองค์ประกอบตามเส้นแนวสายตา (Line-of-Sight: LOS) ซึ่งก็คือ เมื่อวิถีตรงถูกบดบัง เช่น การแพร่กระจายสัญญาณระยะไกลในสภาพแวดล้อมกลางแจ้ง (Outdoor) สัญญาณที่ได้รับจะประกอบไปด้วย องค์ประกอบที่กระจัดกระจาย (Scattered) อันเนื่องมาจากการสะท้อนที่ไม่มีวิถีหลัก ซึ่งสามารถแยกเป็นองค์ประกอบร่วมเฟส (In-phase) และองค์ประกอบตั้งฉาก (Quadrature) โดยแต่ละวิถีมีผลต่อทั้งสองส่วนนี้ด้วย จากทฤษฎีขีดจำกัดกลาง (Central Limit Theorem) เมื่อวิถีมีจำนวนมาก ทำให้สามารถอนุมานได้ว่าองค์ประกอบร่วมเฟสและองค์ประกอบตั้งฉากเป็นตัวแปรสุ่มแบบเกาส์ที่มีค่าเฉลี่ยเป็นศูนย์ ดังนั้นแอมพลิจูดทั้งหมดของสัญญาณที่ได้มาจากการบวกเวกเตอร์ขององค์ประกอบทั้งหมด จึงเป็นไปตามนิยามการจางหายแบบเรย์ลี

การจางหายแบบเรย์ลี คือ การลดลงหรือเพิ่มขึ้นของระดับสัญญาณอย่างทันทีทันใด เนื่องมาจากการแทรกสอดระหว่างคลื่นตรงและคลื่นสะท้อนที่มาถึงโทรศัพท์เคลื่อนที่ ลักษณะสัญญาณที่เกิดจากการจางหายแบบเรย์ลีจะขึ้นอยู่กับระยะทาง เวลา และความถี่ของสัญญาณ

การแจกแจงแบบเรย์ลี (Rayleigh distribution) โดยทั่วไปใช้อธิบายลักษณะข้อมูลที่แปรผันตามเวลาของเอนเวลโลปภาครับของสัญญาณการจางหายแบบราบ หรือเอนเวลโลปของส่วนประกอบคลื่นหลายวิถีทางแต่ละตัว ซึ่งเป็นที่ทราบดีว่าเอนเวลโลปของผลรวมของสอง

สัญญาณรบกวนแบบเกาส์เซียนมีการแจกแจงแบบเรย์ลี (Rappaport, 1995) การแจกแจงแบบเรย์ลีมีฟังก์ชันความหนาแน่นของความน่าจะเป็น (Probability density function: pdf) ดังนี้

$$p(r) = \begin{cases} \frac{r}{\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right) & (0 \le r \le \infty) \\ 0 & (r < 0) \end{cases} \tag{20}$$

โดยที่ σ คือ ค่า rms ของสัญญาณแรงดันภาครับก่อนเข้าเอนเวลโลปดีเทคชัน (Envelope detection), σ^2 คือ กำลังเฉลี่ยของสัญญาณภาครับก่อนเข้าเอนเวลโลปดีเทคชัน และ r คือ ค่าตัวแปรสุ่ม (Random variable)ความน่าจะเป็นที่เอนเวลโลปจะมีค่าน้อยกว่าค่าที่ระบุ R ค่า R คือ ค่าที่สอดคล้องกับฟังก์ชันการแจกแจงความน่าจะเป็นสะสม (Cumulative Distribution Function: CDF) ดังสมการ

$$P(R) = \Pr(r \le R) = \int_0^R p(r)dr = 1 - \exp\left(-\frac{R^2}{2\sigma^2}\right) \tag{21}$$

ค่าเฉลี่ย (r_{mean}) ของการแจกแจงแบบเรย์ลี แสดงดังสมการต่อไปนี้

$$r_{mean} = E[r] = \int_0^\infty rp(r)dr = \sigma\sqrt{\frac{\pi}{2}} = 1.2533\sigma \tag{22}$$

และค่าความแปรปรวนของการแจกแจงแบบเรย์ลีกำหนดให้เป็น σ_r^2 ซึ่งแสดงถึง AC power ในสัญญาณเอนเวลโลป

$$\begin{aligned} \sigma_r^2 &= E[r^2] - E^2[r] = \int_0^\infty r^2 p(r)dr - \frac{\sigma^2\pi}{2} \\ &= \sigma^2\left(2 - \frac{\pi}{2}\right) = 0.4292\sigma^2 \end{aligned} \tag{23}$$

ค่า rms ของเอนเวลโลป คือ ค่ารากที่สอง (Square root) ของค่าเฉลี่ยยกกำลังสอง (Mean square) หรือ $\sqrt{2}\sigma$

ค่ามัธยฐาน (Median value) ของ r หาได้จาก

$$\frac{1}{2} = \int_0^{r_{median}} p(r)dr \tag{24}$$

และจะได้

$$r_{median} = 1.177\sigma \tag{25}$$

ดังนั้น ค่าเฉลี่ยและค่ามัธยฐานแตกต่างกัน 0.55 dB เท่านั้นในสัญญาณการจางหายแบบเรย์ลี สังเกตว่า ค่ามัธยฐานเป็นค่าที่มักใช้ในทางปฏิบัติ เพราะว่าการจางหายของข้อมูลเป็นการวัดปกติในทางปฏิบัติและไม่สามารถสมมติรายละเอียดการแจกแจงได้ การใช้ค่ามัธยฐานแทนที่ค่าเฉลี่ยเพื่อความง่ายต่อการเปรียบเทียบความแตกต่างของการแจกแจงของการจางหาย (Fading distribution) ซึ่งอาจมีการเปลี่ยนแปลงเฉลี่ยกว้างขึ้น

8.6 การจางหายแบบไรเชียน หรือไรเชียนเฟดดิง (Ricean Fading)

เมื่อมีองค์ประกอบของสัญญาณชนิดคงที่ (ไม่จางหาย) เกิดขึ้น การแจกแจงความน่าจะเป็นของเอนเวลโลปการจางหายขนาดเล็ก (small-scale fading) ของสัญญาณ จะเรียกว่าการแจกแจงแบบไรเชียน (Rician distribution) (Rappaport, 1995) ซึ่งมีรูปแบบดังสมการ

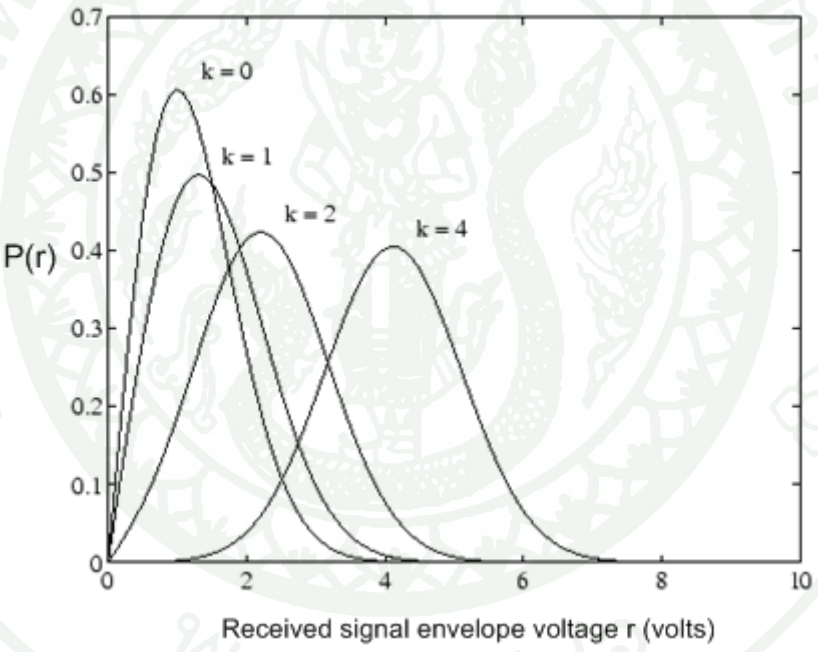
$$p(r) = \begin{cases} \frac{r}{\sigma^2} e^{-\frac{(r^2+A^2)}{2\sigma^2}} I_0\left(\frac{Ar}{\sigma^2}\right) & for(A \geq 0, r \geq 0) \\ 0 & for(r < 0) \end{cases} \tag{26}$$

เมื่อ A คือ แอมพลิจูดที่สูงที่สุด (peak amplitude) ของสัญญาณ, $I_0(.)$ คือ ฟังก์ชันเบสเซลชนิดที่ 1 แบบปรับปรุงที่ออร์เดอร์ 0 (zeroth-order modified Bessel function of the first kind), r คือ ค่าตัวแปรสุ่ม (Random variable) และ σ^2 คือ กำลังเฉลี่ยต่อเวลาของสัญญาณภาครับก่อนเข้าเอนเวลโลปดีเทคชัน การแจงแจงแบบไรเชียนสามารถอธิบายในรูปของพารามิเตอร์ K ได้โดยกำหนดให้ $K = \frac{A^2}{2\sigma^2}$ ซึ่งก็คือค่าอัตราส่วนของกำลังสัญญาณในส่วนของ LOS กับกำลังของ

สัญญาณในส่วนที่มีการแตกกระจาย (Scattered path) นอกจากนี้ค่า K สามารถเปลี่ยนให้อยู่ในเทอมของ dB ดังสมการ

$$K(dB) = 10\log\frac{A^2}{2\sigma^2} \quad dB \tag{27}$$

โดยเรียกค่า K ว่า ค่าไรเชียนแฟคเตอร์ K ซึ่งค่า K จะเป็นตัวบอกลักษณะของการแจกแจงแบบไรเชียน เมื่อ $A \to 0, K \to -\infty \quad dB$ และแอมพลิจูดของสัญญาณลดลง การแจกแจงแบบไรเชียนก็จะเข้าสู่การแจกแจงแบบเรย์ลี (K = 0) ภาพที่ 14 แสดงการเปรียบเทียบค่า pdf ของการจางหายแบบไรเชียนและเรย์ลี



ภาพที่ 14 แสดงค่า pdf ของเรย์ลี (K = 0) และค่า pdf ของไรเชียนที่ค่า K ต่างๆ

ที่มา: Rappaport (1995)

อุปกรณ์และวิธีการ

อุปกรณ์

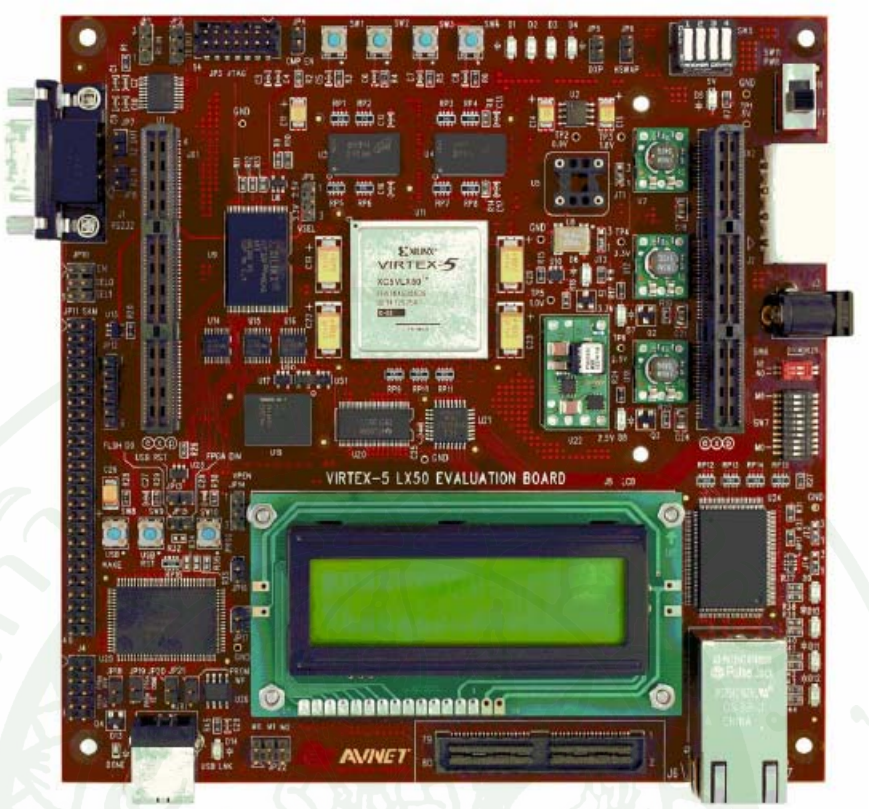
1. เครื่องคอมพิวเตอร์ส่วนบุคคลหรือ PC (Personal Computer)
2. บอร์ดทดลอง FPGA Xilinx ตระกูล Virtex5 ชิปรุ่น XCVLX110 – 1 FF676 C
3. เครื่องดาวน์โหลดโปรแกรม Xilinx Platform Cable USB
4. USB module รุ่น Ezy USB-M01
5. หัวต่อเชื่อม SAMTEC รุ่น QSE-060-01-F-D-A
6. เครื่องมือวัดมัลติมิเตอร์ (Multi meter)
7. สายแพ (Ribbon Cable)

วิธีการ

1. อุปกรณ์ที่ใช้ออกแบบฮาร์ดแวร์ของระบบ

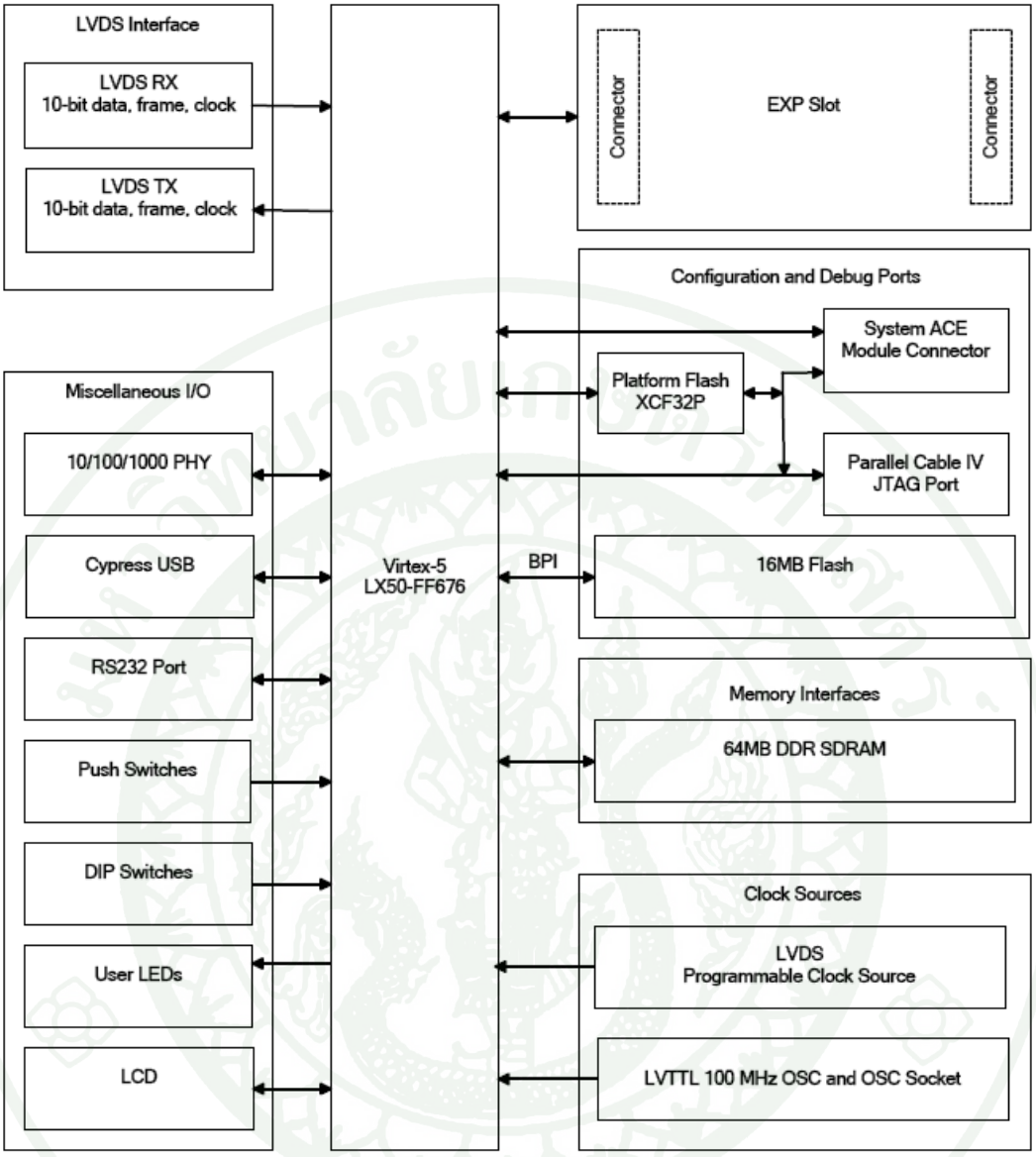
1.1 บอร์ดทดลอง FPGA

บอร์ดทดลอง FPGA ที่ใช้ในโครงการวิทยานิพนธ์นี้ เป็นบอร์ดทดลอง FPGA ของบริษัท Xilinx รุ่น Virtex–5 LX Evaluation Kit ดังแสดงในภาพที่ 15 โดยใช้ชิปตระกูล Virtex-5 เบอร์ XC5VLX110-1 FF676 C ส่วน PROM จะใช้ PROM เบอร์ XCF32P มีความจุ 32 Mbit สำหรับเก็บข้อมูล Configulation ของบอร์ด FPGA ขนาดของ Configurable Logic Blocks (CLBs) เท่ากับ 160 x 54 = 8,640 เซลล์ หนึ่งเซลล์ประกอบด้วย Flip-flop จำนวน 8 ตัว และ 6-input LUT (Look up table) จำนวน 8 ตัว ดังนั้นชิปนี้ประกอบด้วย Flip-flops จำนวน 69,120 ตัว และ 6-input LUT จำนวน 69,120 ชุด หรือคิดเป็น 69,120 x 2^6 = 4,423,680 บิท การดาวน์โหลดโปรแกรมลงบอร์ด FPGA สามารถใช้ Platform Cable USB ที่มีอยู่แล้ว เชื่อมต่อเข้ากับ Connector ที่จัดเตรียมไว้แล้วบนบอร์ด FPGA ซึ่งสามารถใช้งานได้ทันที



ภาพที่ 15 บอร์ดทดลอง FPGA ตระกูล Virtex5 รุ่น XC5VLX110 -1 FF676 C

ในส่วนของอุปกรณ์ต่างๆบนบอร์ด FPGA ดังแสดงในภาพที่ 16 ได้แก่ ช่องเชื่อมต่อสัญญาณ EXP Slot จำนวน 2 ช่อง, ช่องเชื่อมต่อสัญญาณมาตรฐานแบบ RS-232 จำนวน 1 ช่อง, ช่องเชื่อมต่อสัญญาณมาตรฐานแบบ USB 2.0 จำนวน 1 ช่อง, ช่องเชื่อมต่อสัญญาณมาตรฐานแบบ Ethernet 10/100/1000 จำนวน 1 ช่อง, ช่องต่อสำหรับการดาวน์โหลดโปรแกรมจำนวน 1 ช่อง, จอแสดงผล LCD แบบขาวดำ จำนวน 1 จอ, ปุ่มกดและปุ่มโยก จำนวน 8 ปุ่ม, ไฟแสดงผล LED จำนวน 4 ดวง, สัญญาณนาฬิกา จำนวน 10 ตัว, หน่วยความจำ DDR2 SDRAM ขนาด 64 MB ใช้บัสข้อมูลขนาด 32 บิท, หน่วยความจำ FLASH Memory ขนาด 16 MB ใช้บัสข้อมูลขนาด 16 บิท และหน่วยความจำ EEPROM ขนาด 4 MB ใช้สำหรับกรณีที่ต้องการเก็บโปรแกรมที่ดาวน์โหลดลงชิปแล้วไว้ใช้ในกรณีที่ต้องการปิดหรือหยุดการทำงานของบอร์ด



ภาพที่ 16 แสดงแผนภาพส่วนประกอบของบอร์ด FPGA ตระกูล Virtex5 รุ่น XC5VLX110-1FF676

ที่มา: Avnet electronic marketing (2008)

1.2 เครื่องดาวน์โหลดโปรแกรม Platform Cable USB

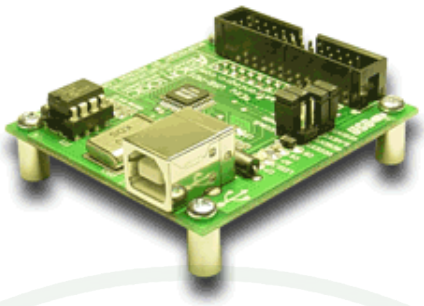
การนำโปรแกรมจากเครื่องคอมพิวเตอร์บรรจุลงบอร์ด FPGA ต้องใช้เครื่องดาวน์โหลดโปรแกรม Platform Cable USB ของบริษัท Xilinx ดังภาพที่ 17 เพื่อช่วยในการดาวน์โหลดโปรแกรมลงชิป FPGA หรือ EEPROM ที่อยู่บนบอร์ด FPGA รุ่น Virtex5 XC5VLX110 โดยมีช่อง

เชื่อมต่อสองช่อง คือ ช่องแรกเป็นช่องต่อ USB ต่อกับเครื่องคอมพิวเตอร์และอีกช่องเป็นช่องต่อ USB ต่อกับบอร์ด FPGA เป็นสายแพแบบ 14 เส้น ส่วนวิธีการเชื่อมต่อประสานทั้งสองช่องเป็นไปตามมาตรฐานของบริษัท Xilinx จึงไม่ขออธิบายรายละเอียดในวิทยานิพนธ์ฉบับนี้

ภาพที่ 17 เครื่องดาวน์โหลดโปรแกรม Platform Cable USB

1.3 โมดูล USB

สำหรับการเชื่อมต่อประสานระหว่างเครื่องคอมพิวเตอร์กับบอร์ด FPGA จะใช้โมดูล USB รุ่น Ezy USB-M01 ของบริษัทแอสทรอนลอจิกรีเสิร์จแอนดิวิลอปเมนต์ จำกัด ดังภาพที่ 18 ทำหน้าที่เป็นทั้งตัวรับส่งข้อมูลจากเครื่องคอมพิวเตอร์ไปยังบอร์ด FPGA และจากบอร์ด FPGA ไปยังเครื่องคอมพิวเตอร์ โดยมีอัตราการส่งถ่ายข้อมูล 1 เมกะไบท์ต่อวินาที หรือ 8 เมกะบิทต่อวินาที มีบัฟเฟอร์ FIFO ขนาด 384 ไบท์สำหรับด้านส่งและขนาด 128 ไบท์สำหรับด้านรับ นอกจากนี้ยังมีโหมดการทำงานบิทแบงก์ (Bit – Bang) ซึ่งอนุญาตให้ใช้บัสข้อมูลเป็นช่องทาง I/O ขนาด 8 บิท เพื่อใช้ในการรับ-ส่งข้อมูลทั่วไปได้

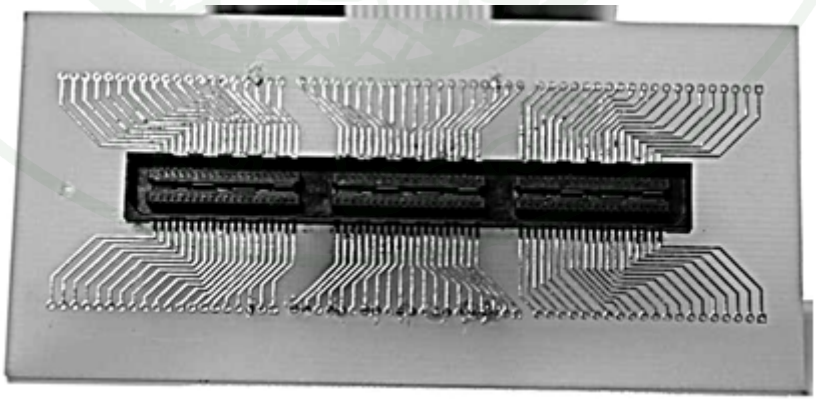


ภาพที่ 18 โมดูล USB รุ่น Ezy USB-M01

ที่มา: Astronlogic Research and Development (n.d.)

1.4 หัวเชื่อมต่อ QSE (รัชนนท์, 2551)

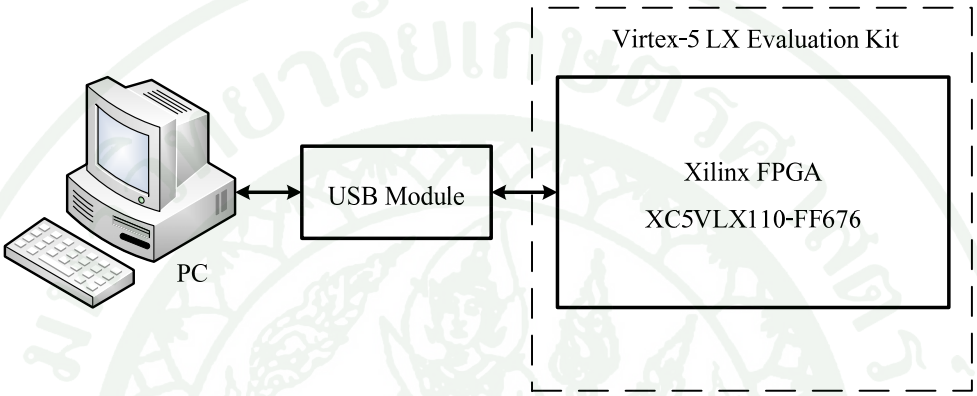
สำหรับอุปกรณ์เชื่อมต่อภายนอกของบอร์ด FPGA จะเชื่อมต่อผ่านช่อง EXP JX1 ที่มีหัวเชื่อมต่อแบบ QTE ของบริษัท SAMTEC จำกัด เป็นหัวเชื่อมต่อตัวผู้ และต้องใช้หัวเชื่อมต่อแบบ QSE ที่เป็นตัวเมียในการเชื่อมต่อ แต่หัวเชื่อมต่อแบบ QSE เป็นอุปกรณ์ชนิด Surface Mount คือ ต้องเชื่อมต่อด้านหนึ่งของหัวเชื่อมต่อ QSE ลงบนแผ่น PCB ก่อน จึงนำไปเชื่อมต่อกับสายสัญญาณ เมื่อเชื่อมต่อสายสัญญาณกับ PCB และเชื่อมต่อ PCB กับหัว QSE แล้ว มีลักษณะดังภาพที่ 19 โดยอีกด้านหนึ่งของขาสัญญาณเชื่อมต่อกับโมดูล USB



ภาพที่ 19 PCB และหัวเชื่อมต่อ QSE

2. การเชื่อมต่อประสาน (Interface)

ในการเชื่อมต่อประสานอุปกรณ์ต่างๆ เข้าเป็นระบบเครื่องถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ จะมีส่วนประกอบ ได้แก่ เครื่อง PC, บอร์ดทดลอง FPGA และโมดูล USB โดยมีลักษณะการเชื่อมต่อประสานดังภาพที่ 20



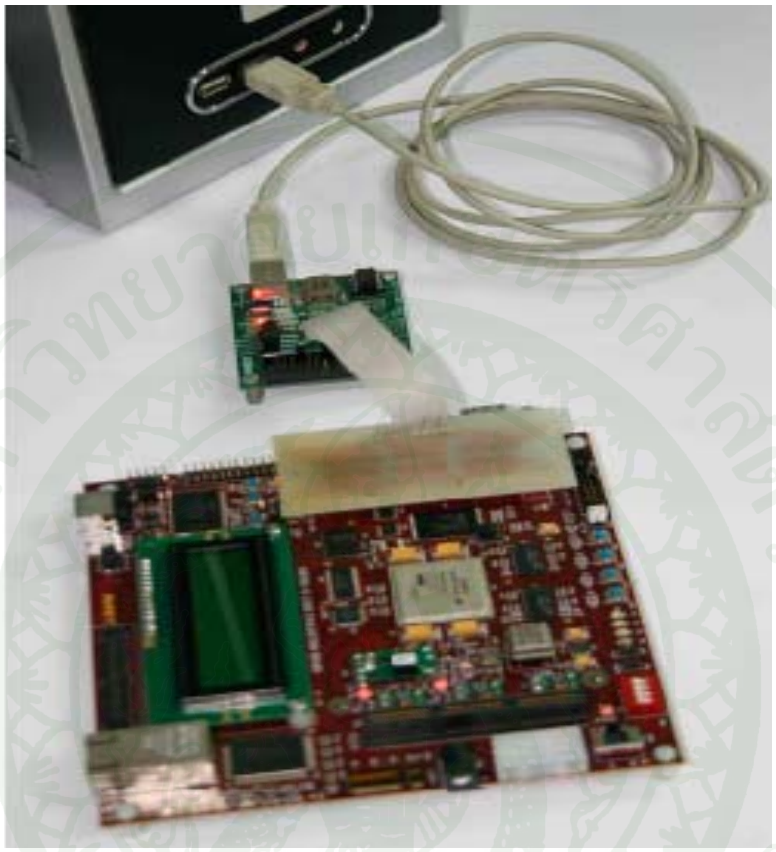
ภาพที่ 20 บล็อกแผนภาพระบบเครื่องถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์

การเชื่อมต่อประสานอุปกรณ์ได้แบ่งออกเป็น 2 กรณี คือ การเชื่อมต่อประสานสำหรับดาวน์โหลดโปรแกรม และการเชื่อมต่อประสานสำหรับรับส่งข้อมูล โดยทั้ง 2 กรณีเป็นการเชื่อมต่อประสานระหว่างเครื่อง PC กับบอร์ด FPGA แตกต่างที่อุปกรณ์ที่ใช้เป็นสื่อกลางในการเชื่อมต่อ

การเชื่อมต่อประสานสำหรับดาวน์โหลดโปรแกรม จะใช้เครื่องดาวน์โหลดโปรแกรม Platform Cable USB คั่นระหว่างบอร์ด FPGA กับเครื่อง PC ดังภาพที่ 21

ภาพที่ 21 การเชื่อมต่อประสานสำหรับดาวน์โหลดโปรแกรม

การเชื่อมต่อประสานสำหรับรับส่งข้อมูล จะใช้โมดูล USB ที่เชื่อมต่อกับ PCB และหัว QSE มาต่อเข้ากับเครื่อง PC และบอร์ด FPGA ดังภาพที่ 22



ภาพที่ 22 การเชื่อมต่อประสานสำหรับรับส่งข้อมูล

ภาพที่ 23 แสดงการเชื่อมต่อประสานอุปกรณ์สำหรับดาวน์โหลดโปรแกรมและอุปกรณ์สำหรับรับส่งข้อมูลเข้ากับเครื่องคอมพิวเตอร์ สำหรับใช้งานจริง

ภาพที่ 23 การเชื่อมต่ออุปกรณ์ดาวน์โหลดโปรแกรมและอุปกรณ์รับส่งข้อมูลเข้ากับเครื่อง PC

ภาพที่ 24 แสดงโปรแกรม Xilinx ISE 10.1.02 ที่ใช้ดาวน์โหลดโปรแกรมในการถอดรหัสลงสู่บอร์ด FPGA

ภาพที่ 24 แสดงโปรแกรม Xilinx ISE 10.1.02 ที่ใช้ดาวน์โหลดโปรแกรมลงบอร์ด FPGA

สำหรับขาสัญญาณที่ใช้ในการเชื่อมต่อ ในกรณีสำหรับดาวน์โหลดโปรแกรมจะไม่กล่าวถึง เนื่องจากเป็นอุปกรณ์มาตรฐานของบริษัท Xilinx จำกัด ส่วนขาสัญญาณที่ใช้ในการเชื่อมต่อรับส่งข้อมูล สามารถดูการเทียบขาสัญญาณได้จากตารางที่ 2

ตารางที่ 2 เทียบตำแหน่งขาสัญญาณบนบอร์ด FPGA กับโมดูล USB

ขาสัญญาณบอร์ด FPGA	ขาสัญญาณ USB-module	ขาของชิป FPGA
User clock input (CLK)	-	E16
Push clock SW 4 (RST)	-	F17
LED 0 (ALM)	-	E11
ขาที่ 86 ของ EXP JX1 (3.3 V)	ขาที่ 8 ของโมดูล USB (3V3_VCCIO)	-

ตารางที่ 2 (ต่อ)

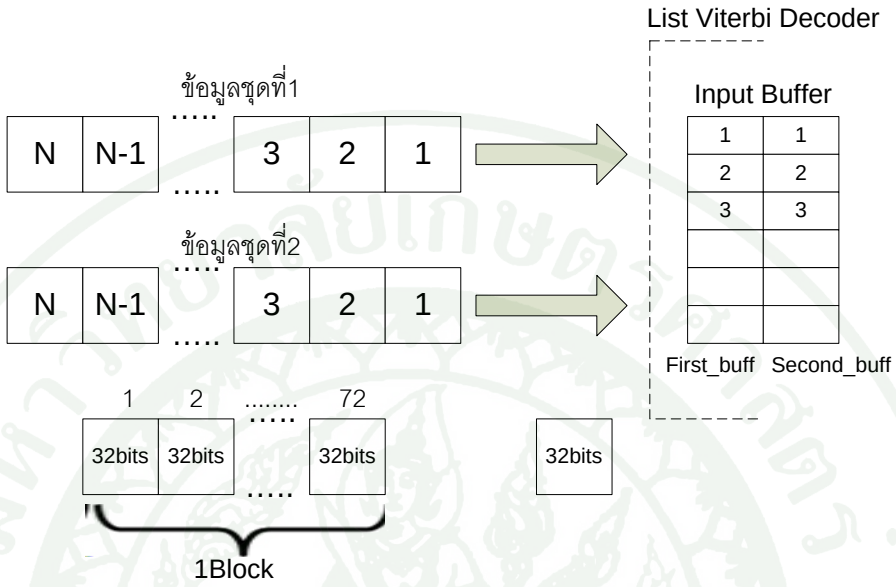
ขาสัญญาณบอร์ด FPGA	ขาสัญญาณ USB-module	ขาของชิป FPGA
ขาที่ 80 ของ EXP JX1	ขาที่ 14 ของโมดูล USB (RXF)	H1
ขาที่ 78 ของ EXP JX1	ขาที่ 15 ของโมดูล USB (TXE)	G1
ขาที่ 74 ของ EXP JX1	ขาที่ 16 ของโมดูล USB (WR)	W1
ขาที่ 72 ของ EXP JX1	ขาที่ 17 ของโมดูล USB (RD)	Y1
ขาที่ 68 ของ EXP JX1	ขาที่ 18 ของโมดูล USB (D7)	E3
ขาที่ 66 ของ EXP JX1	ขาที่ 19 ของโมดูล USB (D6)	F3
ขาที่ 62 ของ EXP JX1	ขาที่ 20 ของโมดูล USB (D5)	F4
ขาที่ 60 ของ EXP JX1	ขาที่ 21 ของโมดูล USB (D4)	F5
ขาที่ 56 ของ EXP JX1	ขาที่ 22 ของโมดูล USB (D3)	E1
ขาที่ 54 ของ EXP JX1	ขาที่ 23 ของโมดูล USB (D2)	E2
ขาที่ 50 ของ EXP JX1	ขาที่ 24 ของโมดูล USB (D1)	R2
ขาที่ 48 ของ EXP JX1	ขาที่ 25 ของโมดูล USB (D0)	R3
ขาที่ 46 ของ EXP JX1	ขาที่ 26 ของโมดูล USB (GND)	-

3. การออกแบบการรับ-ส่งข้อมูลของตัวถอดรหัสลิสวิเทอร์บิ (List Viterbi Decoder)

3.1 ลักษณะข้อมูลสำหรับตัวถอดรหัสลิสวิเทอร์บิ

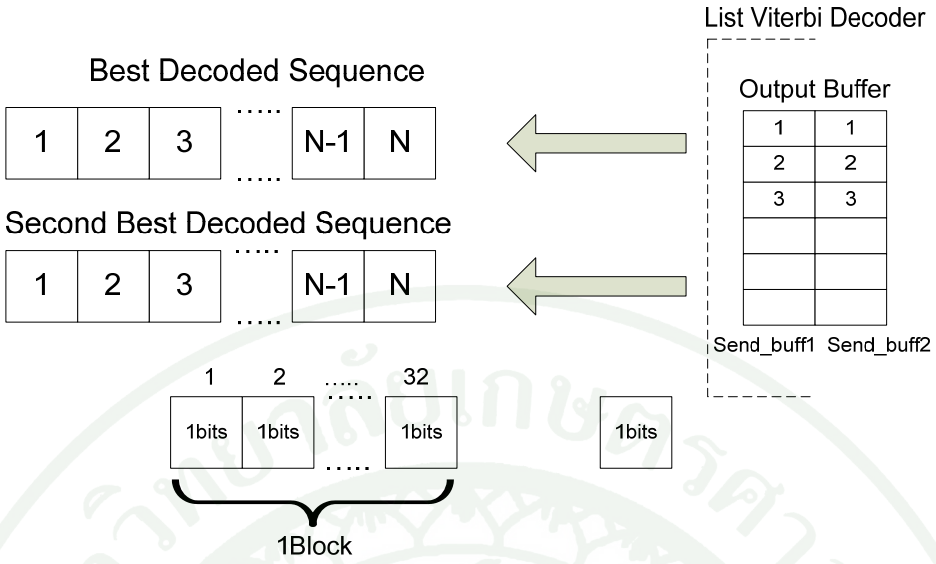
สำหรับข้อมูลที่ป้อนให้กับตัวถอดรหัสลิสวิเทอร์บิ จะถูกส่งจากคอมพิวเตอร์ผ่านโมดูล USB เข้าสู่บอร์ด FPGA เพื่อทำการประมวลผล จำนวนข้อมูลทั้งหมดที่ต้องการสำหรับการทำงานของตัวถอดรหัสในแต่ละครั้งมีจำนวนเท่ากับ 144 ค่า ซึ่งค่าจำนวน 144 ค่านั้น มาจากข้อมูลขนาด 32 บิท ถูกเข้ารหัสคอนโวลูชัน (2, 1, 4) ทำให้ได้คำรหัสขนาด 72 บิท จากนั้นคำรหัสจะถูกมอดูเลท และและส่งผ่านไปยังช่องสัญญาณ ที่ภาครับจะทำการดีมอดูเลท โดยใช้ Matched-filter จำนวน 2 ตัว ทำให้คำรหัสจากขนาด 72 บิท เปลี่ยนเป็นข้อมูลจำนวน 144 ค่า โดยข้อมูลที่ใช้จะถูกแบ่งออกเป็น 2 ชุด ชุดละ72 ค่าและข้อมูลแต่ละค่า มีขนาดเท่ากับ 24 บิท สำหรับการออกแบบการรับข้อมูลนั้น ตัวถอดรหัสลิสวิเทอร์บิจะทำการรับข้อมูลชุดที่ 1 ไปพักไว้ในบัฟเฟอร์ของตัว

ถอดรหัสจนครบ 1 บล็อก (72 ค่า) จากนั้นจึงรับข้อมูลชุดที่ 2 เข้าไปเก็บไว้ในบัฟเฟอร์ของตัวถอดรหัสจนครบขนาด 1 บล็อก (72 ค่า)



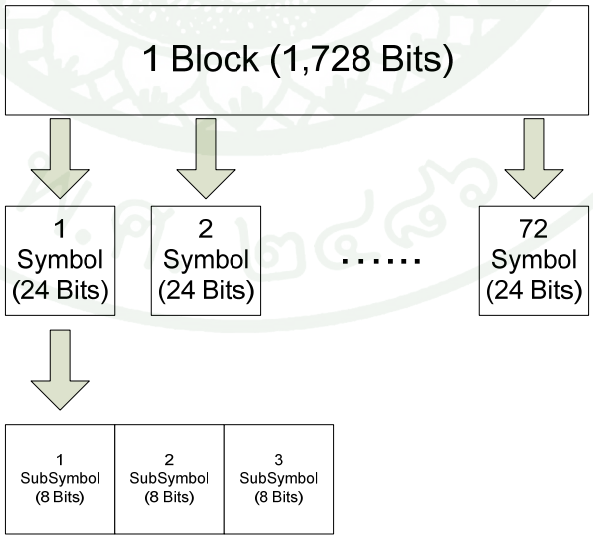
ภาพที่ 25 การนำ Received Symbols เข้าเก็บในอินพุทบัฟเฟอร์ของตัวถอดรหัส

ข้อมูลที่ได้จากการประมวลผลของตัวถอดรหัสจะถูกส่งกลับไปให้คอมพิวเตอร์เพื่อทำการบันทึกลงในแฟ้มข้อมูล (File) โดยจำนวนข้อมูลที่ได้ทำการถอดรหัสแล้ว จะมีจำนวนชุดข้อมูลไม่เท่ากับที่ป้อนเข้าคือ จะได้ข้อมูล 2 ชุด ชุดละ 32 บิท ภาพที่ 25 แสดงจำนวนชุดข้อมูลที่นำเข้าตัวถอดรหัส และภาพที่ 26 แสดงลักษณะของการส่งข้อมูลออกจากตัวถอดรหัส ซึ่งจะเห็นว่าข้อมูลที่ส่งกลับมี 2 ชุด



ภาพที่ 26 การนำ Decoded Symbols ออกจากเอาท์พุทบัฟเฟอร์ของตัวถอดรหัส

จากภาพที่ 25 จะเห็นว่าตัวถอดรหัสต้องรับข้อมูลเข้าไปจำนวน 2 ชุด หรือเท่ากับ 3,456 บิท ถึงจะเริ่มต้นทำการประมวลผล แต่เนื่องจากโมดูล USB มีขนาดบัสข้อมูล 8 บิท ตัวถอดรหัสจึงรับข้อมูลได้ครั้งละ 8 บิทเท่านั้น จึงต้องมีการกำหนดเงื่อนไขการรับข้อมูลในลักษณะวนลูป เพื่อรับข้อมูลให้ได้ครบทั้ง 3,456 บิท โดยในภาพที่ 27 แสดงการแบ่งข้อมูลขนาด 1 บล็อก (1,728 บิท) ออกเป็น 72 ค่า และข้อมูลขนาด 1 ค่า (24 บิท) ถูกแบ่งออกเป็น 3 ค่าย่อย ซึ่งข้อมูลขนาด 1 ค่าย่อย (8 บิท) จะเท่ากับขนาดบัสข้อมูลของโมดูล USB

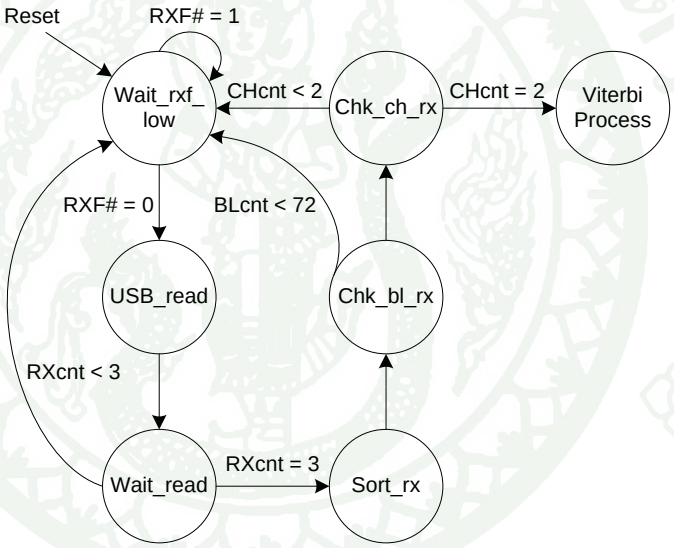


ภาพที่ 27 การแบ่งข้อมูลขนาด 1 บล็อกออกเป็นค่าย่อย

3.2 การรับข้อมูลจากโมดูล USB เข้าสู่ตัวถอดรหัสลิสวิเทอร์บิ

เมื่อข้อมูลจากคอมพิวเตอร์ส่งเข้าสู่โมดูล USB จะถูกเก็บไว้ใน FIFO ของโมดูล USB และที่ขาสัญญาณ RXF# ของโมดูลจะเปลี่ยนสภาวะจากสูงเป็นต่ำ เพื่อบอกให้ทราบว่าขณะนี้มีข้อมูลพร้อมที่จะส่งออกไปให้ตัวถอดรหัสแล้ว โดยตัวถอดรหัสจะใช้เงื่อนไขการเปลี่ยนสภาวะที่ขาสัญญาณ RXF# ในการรับข้อมูลเข้าสู่ตัวถอดรหัส

การเขียนโปรแกรมภาษา VHDL ให้บอร์ด FPGA ทำหน้าที่รับข้อมูลจากโมดูล USB นั้น จะมีขาสัญญาณที่เกี่ยวข้องได้แก่ RXF# , RD# และ D[0…7] โดยสามารถเขียนแผนภาพสเตทได้ดังภาพที่ 28



ภาพที่ 28 แผนภาพสเตทการรับข้อมูลจาก USB เข้าสู่ตัวถอดรหัส

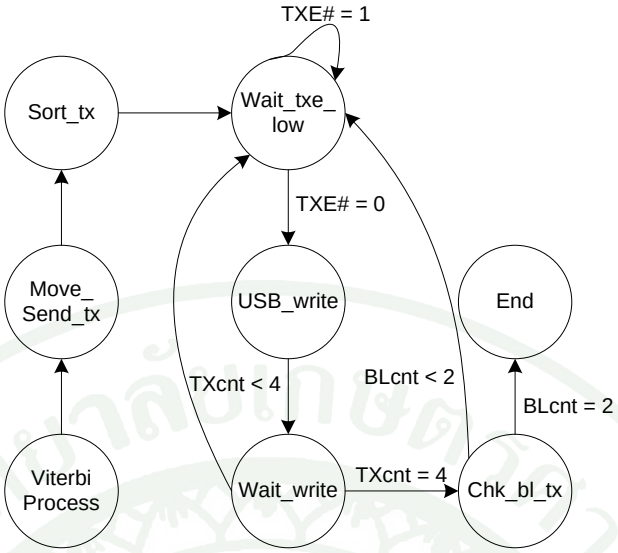
จากภาพที่ 28 สามารถอธิบายการทำงานของแผนภาพสเตทได้ดังนี้ ในขณะเริ่มทำงานหรือมีการรีเซทตัวถอดรหัส ตัวถอดรหัสจะเริ่มทำงานในสเตท Wait_rxf_low ซึ่งจะตรวจสอบสภาวะที่ขาสัญญาณ RXF# ว่าอยู่ในสภาวะใด ถ้าอยู่ในสภาวะสูงหรือเป็นลอจิก 1 ก็ยังรออยู่ที่สเตทนี้ต่อไป แต่หากขาสัญญาณ RXF# เปลี่ยนสภาวะเป็นต่ำหรือเป็นลอจิก 0 ตัวถอดรหัสจะเปลี่ยนไปทำงานในสเตท USB_read เพื่อทำการอ่านข้อมูลจาก FIFO ของโมดูล USB เข้ามาเก็บไว้ในบัฟเฟอร์ค่าย่อยของตัวถอดรหัส โดยใช้การตรวจสอบเงื่อนไขตัวแปรที่ชื่อ RXcnt ที่สเตท Wait_read ว่ามีค่าเท่ากับ 3 หรือไม่ ถ้าค่าของตัวแปรยังน้อยกว่า 3 ก็ให้กลับไปที่สเตท

Wait_rxf_low เพื่อรอรับข้อมูลจนกว่าจะครบ 1 ค่า หรือจำนวน 24 บิท และหากรับเข้ามาครบแล้ว ค่าของตัวแปร RXcnt มีค่าเท่ากับ 3 ตัวถอดรหัสจะเปลี่ยนไปทำงานที่สเตท Sort_rx เพื่อนำข้อมูลในบัฟเฟอร์ค่าย่อยเก็บลงในอินพุทบัฟเฟอร์ของตัวถอดรหัส จากนั้นสเตทก็จะเปลี่ยนเป็น Chk_bl_rx เพื่อตรวจสอบตัวแปรเงื่อนไขที่ชื่อ BLcnt ว่ามีค่าเท่ากับ 72 ค่าหรือไม่ ถ้าค่าของตัวแปรยังน้อยกว่า 72 ก็แสดงว่ายังรับข้อมูลเข้ามายังไม่ครบ 1 บล็อก ก็ให้กลับไปที่สเตท Wait_rxf_low เพื่อรอรับข้อมูลจนกว่าจะครบ 1 บล็อก หรือจำนวน 72 ค่า (1,728 บิท) และหากรับเข้ามาครบแล้วค่าของตัวแปร BLcnt จะมีค่าเท่ากับ 72 ตัวถอดรหัสจะเปลี่ยนสเตทเป็น Chk_ch_rx เพื่อตรวจสอบเงื่อนไขตัวแปรที่ชื่อ CHcnt ว่ามีค่าเท่ากับ 2 หรือไม่ ถ้าค่าของตัวแปรยังน้อยกว่า 2 ก็แสดงว่ายังรับข้อมูลเข้ามาไม่ครบ 2 ชุด ก็ให้กลับไปที่สเตท Wait_rxf_low เพื่อรอรับข้อมูลจนกว่าจะครบ 2 ชุด หรือจำนวน 144 ค่า (3,456 บิท) และหากรับมาครบแล้วค่าของตัวแปร CHcnt จะมีค่าเท่ากับ 2 ก็จะนำข้อมูลที่รับเข้ามาทั้งหมดไปประมวลผลต่อไปในสเตท List-of-2 Viterbi Process ซึ่งมีรายละเอียดการทำงานย่อย และไม่ได้อธิบายในส่วนนี้

3.3 การส่งข้อมูลจากตัวถอดรหัสลิสวิเทอร์บิเข้าสู่โมดูล USB

เมื่อตัวถอดรหัสประมวลผลเสร็จแล้ว ก็จะส่งผลลัพธ์กลับไปให้เครื่องคอมพิวเตอร์ โดยจะเขียนข้อมูลจากเอาท์พุทบัฟเฟอร์ของตัวถอดรหัสลงใน FIFO ของโมดูล USB แต่ถ้าสัญญาณ TXE# มีสภาวะเป็นสูงหรือลอจิก 1 ก็จะยังไม่สามารถเขียนข้อมูลลงใน FIFO ของโมดูล USB ได้ ต้องรอให้สัญญาณ TXE# มีสภาวะเป็นต่ำหรือลอจิก 0 ก่อนจึงจะสามารถเขียนข้อมูลลงใน FIFO ของโมดูล USB ได้ ซึ่งการเขียนข้อมูลนี้จะเขียนได้เพียงครั้งละ 8 บิท เช่นเดียวกับการอ่านข้อมูล

การเขียนโปรแกรมภาษา VHDL ให้บอร์ด FPGA ทำหน้าที่ส่งข้อมูลเข้าสู่โมดูล USB นั้น จะมีขาสัญญาณที่เกี่ยวข้องได้แก่ TXE# , WR และ D[0…7] โดยสามารถเขียนแผนภาพสเตทการทำงานได้ดังภาพที่ 29

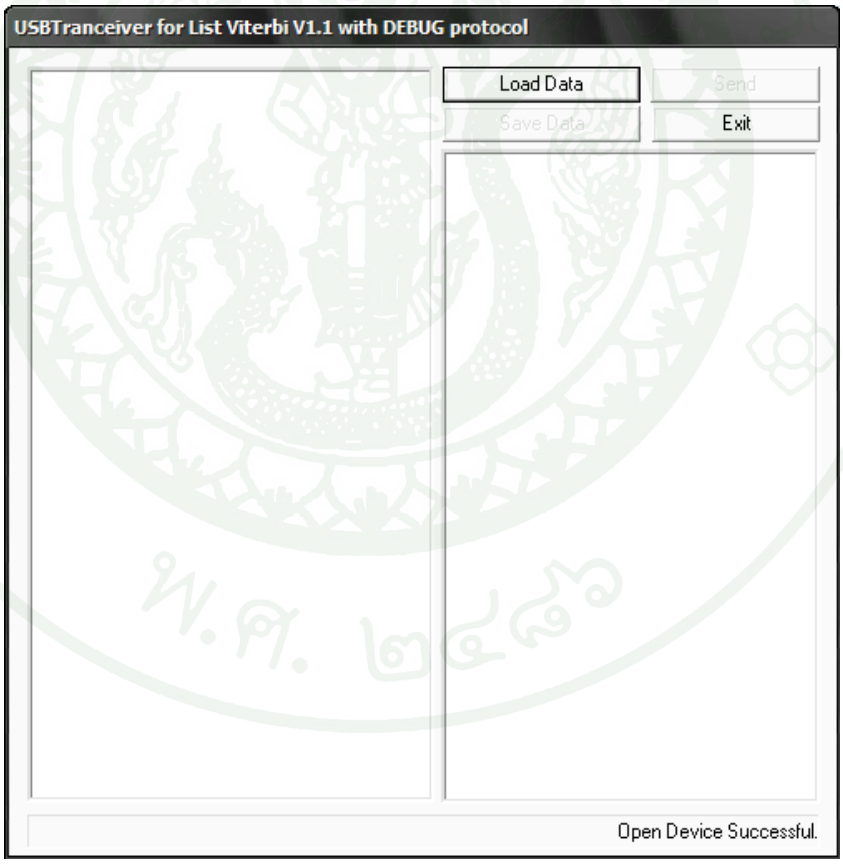


ภาพที่ 29 แผนภาพสเตทการส่งข้อมูลจากตัวถอดรหัสเข้าสู่ USB

จากภาพที่ 29 สามารถอธิบายการทำงานของแผนภาพสเตทได้ดังนี้ เมื่อตัวถอดรหัสประมวลผลเสร็จในสเตท List-of-2 Viterbi Process จะมีการนำค่าของผลลัพธ์ที่ได้รับสองค่ามาเก็บไว้ในเอาท์พุทบัฟเฟอร์ฝั่งส่งในสเตท Move_send_tx จากนั้นนำข้อมูลในเอาท์พุทบัฟเฟอร์ไปจัดเรียงและแยกเป็นส่วนๆขนาดส่วนละ 8 บิท ในสเตท Sort_tx เมื่อจัดเรียงเสร็จจะเปลี่ยนสเตทการทำงานไปที่ Wait_tex_low เพื่อตรวจสอบสัญญาณ TXE# ว่ามีสภาวะเป็นสูงหรือต่ำ ถ้าสัญญาณ TXE# อยู่ในสภาวะสูงหรือลอจิก 1 ก็ยังไม่เปลี่ยนแปลงสเตทการทำงาน เนื่องจากบัสข้อมูลยังไม่ว่าง จึงต้องรอให้สัญญาณ TXE# อยู่ในสภาวะต่ำหรือลอจิก 0 ก่อน ถึงจะเปลี่ยนสเตทการทำงานไปที่ USB_write เพื่อทำการเขียนข้อมูลขนาด 1 สัญลักษณ์ย่อย (8 บิท) จากเอาท์พุทบัฟเฟอร์ลงใน FIFO ของโมดูล USB หลังจากเขียนเสร็จแล้วก็จะเปลี่ยนสเตทไปที่ Wait_write เพื่อตรวจสอบค่าตัวแปรที่ชื่อ TXcnt ว่ามีค่าเท่ากับ 4 หรือไม่ ถ้าค่าของตัวแปรยังน้อยกว่า 4 ก็แสดงว่ายังส่งข้อมูลไม่ครบ 1 สัญลักษณ์หรือ 32 บิท ให้เปลี่ยนสเตทไปที่ Wait_txe_low เพื่อส่งข้อมูลที่เหลือให้ครบ 1 สัญลักษณ์ หากส่งข้อมูลครบแล้วตัวแปร TXcnt จะมีค่าเท่ากับ 4 ก็จะเปลี่ยนสเตทการทำงานไปที่ Chk_txe_tx เพื่อตรวจสอบค่าตัวแปรที่ชื่อ BLcnt ว่ามีค่าเท่ากับ 2 หรือไม่ ถ้าค่าของตัวแปรยังน้อยกว่า 2 แสดงว่ายังส่งข้อมูลไม่ครบ 2 ชุด และให้เปลี่ยนสเตทไปที่ Wait_txe_low เพื่อส่งข้อมูลที่เหลือให้ครบ 2 ชุด หรือ 64 บิท หากส่งข้อมูลจนครบแล้ว ตัวแปร BLcnt จะมีค่าเท่ากับ 2 สเตทการเขียนข้อมูลก็จะสิ้นสุดลง

3.4 โปรแกรมคอมพิวเตอร์ที่ใช้ติดต่อกับโมดูล USB

งานวิจัยนี้ ได้นำบอร์ด FPGA มาเชื่อมต่อกับเครื่องคอมพิวเตอร์ โดยใช้การเชื่อมต่อประสานอุปกรณ์ (Interfacing) เป็นโมดูล USB ที่ใช้ไอซีรุ่น FT245BM ของบริษัท FTDI จำกัด ซึ่งการนำมาใช้งานจะต้องโปรแกรมที่บอร์ด FPGA และคอมพิวเตอร์ให้ทำงานสัมพันธ์กับโมดูล USB โดยที่คอมพิวเตอร์ได้นำโปรแกรม D2XX Direct Drivers (Future Technology Devices International Ltd. [FTDI], 2002) ของ FTDI มาใช้งาน ซึ่งมีความสามารถสร้างการสื่อสารระหว่างโมดูล USB กับระบบปฏิบัติการคอมพิวเตอร์ได้ โดยการสร้างโปรแกรมที่ใช้รับส่งข้อมูลบนเครื่องคอมพิวเตอร์ สามารถสร้างมาจากการนำเครื่องมือของ D2XX Direct Drivers ไปผนวกลงใน Microsoft Visual C++ ซึ่งโปรแกรมที่ใช้สื่อสารกับโมดูล USB ในงานวิจัยนี้มีชื่อว่า “USB Transceiver for List Viterbi V1.1 with DEBUG protocal” มีลักษณะดังภาพที่ 30



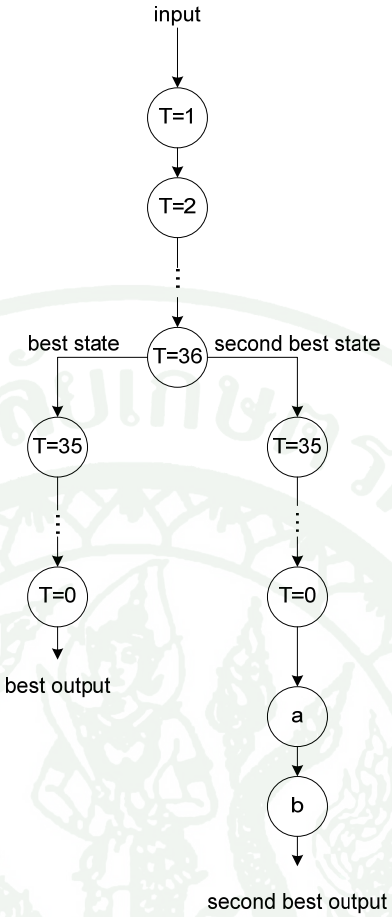
ภาพที่ 30 หน้าต่างของโปรแกรม USB Transceiver for List Viterbi V1.1 with DEBUG protocal

USB Transceiver for List Viterbi V1.1 with DEBUG protocal มีความสามารถในการสร้างการติดต่อกับโมดูล USB และรับส่งข้อมูลที่เป็นไฟล์ขนาดไม่เกิน 1 ล้านไบต์ผ่านทางโมดูล USB โปรแกรมนี้จะเริ่มต้นทำงานด้วยการสร้างการติดต่อระหว่างคอมพิวเตอร์กับโมดูล USB หลังจากที่สร้างการติดต่อสำเร็จแล้ว โปรแกรมจะให้เลือกไฟล์ที่ต้องการส่ง แล้วจึงเริ่มแบ่งส่งไฟล์ข้อมูลไปยังโมดูล USB และรับข้อมูลที่มาจากโมดูล USB มาเก็บลงไฟล์ที่ระบุ สุดท้ายโปรแกรมจะยุติการติดต่อกับโมดูล USB โดยโปรแกรมจะเรียกใช้ฟังก์ชันที่มากับ D2XX Direct Drivers (FTDI, 2002) ในทุกขั้นตอนที่มีการเรียกใช้งานโมดูล USB

4. การออกแบบแผนภาพสเตทการทำงานของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์

4.1 การออกแบบแผนภาพสเตทสำหรับเขียนโปรแกรมภาษา VHDL

ชิ้นงานต้นแบบของเครื่องถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ที่ใช้อัลกอริทึม LVA แบบขนาน โดยใช้โปรแกรมภาษา VHDL นั้น แผนภาพสเตทสำหรับแต่ละฟังก์ชันเป็นการออกแบบโดยใช้ FSM (Finite-State Machine) เป็นตัวควบคุมการทำงานของตัวถอดรหัส จะเห็นได้ว่าสเตทการทำงานของโปรแกรมภาษา VHDL นั้น ไม่เหมือนกับสเตทการทำงานของรหัสคอนโวลูชัน ซึ่งเป็นสเตท s0 ถึง s15 ดังแสดงในภาพที่ 33 และภาพที่ 31 เป็นการแสดงแผนภาพสเตทการทำงานโดยรวมของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์



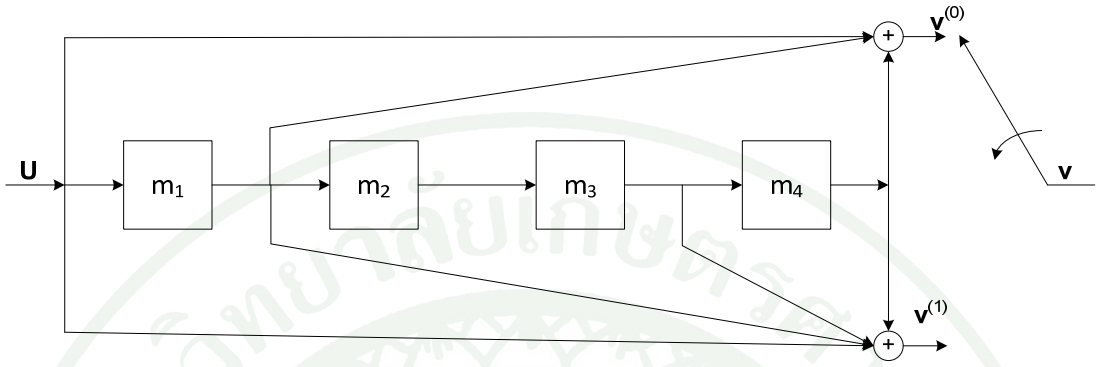
ภาพที่ 31 แผนภาพสเตทของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์

ที่มา: Tuntoolavest and Noradee (2009)

ก่อนที่จะเข้าสู่กระบวนการทำงานของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ โดยใช้โปรแกรมภาษา VHDL จะขออธิบายตัวอย่างวิธีการถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ เพื่อความเข้าใจดังตัวอย่าง เป็นการแสดงแนวคิดของการถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ โดยใช้บิทข้อมูลขนาด 5 บิท ซึ่งเป็นข้อมูลที่มีความยาวบิทสั้นมาก โดยขนาดของข้อมูลสามารถทำให้เพิ่มขึ้นได้และขนาดที่ต้องการในโครงการวิจัยนี้คือ 32 บิท

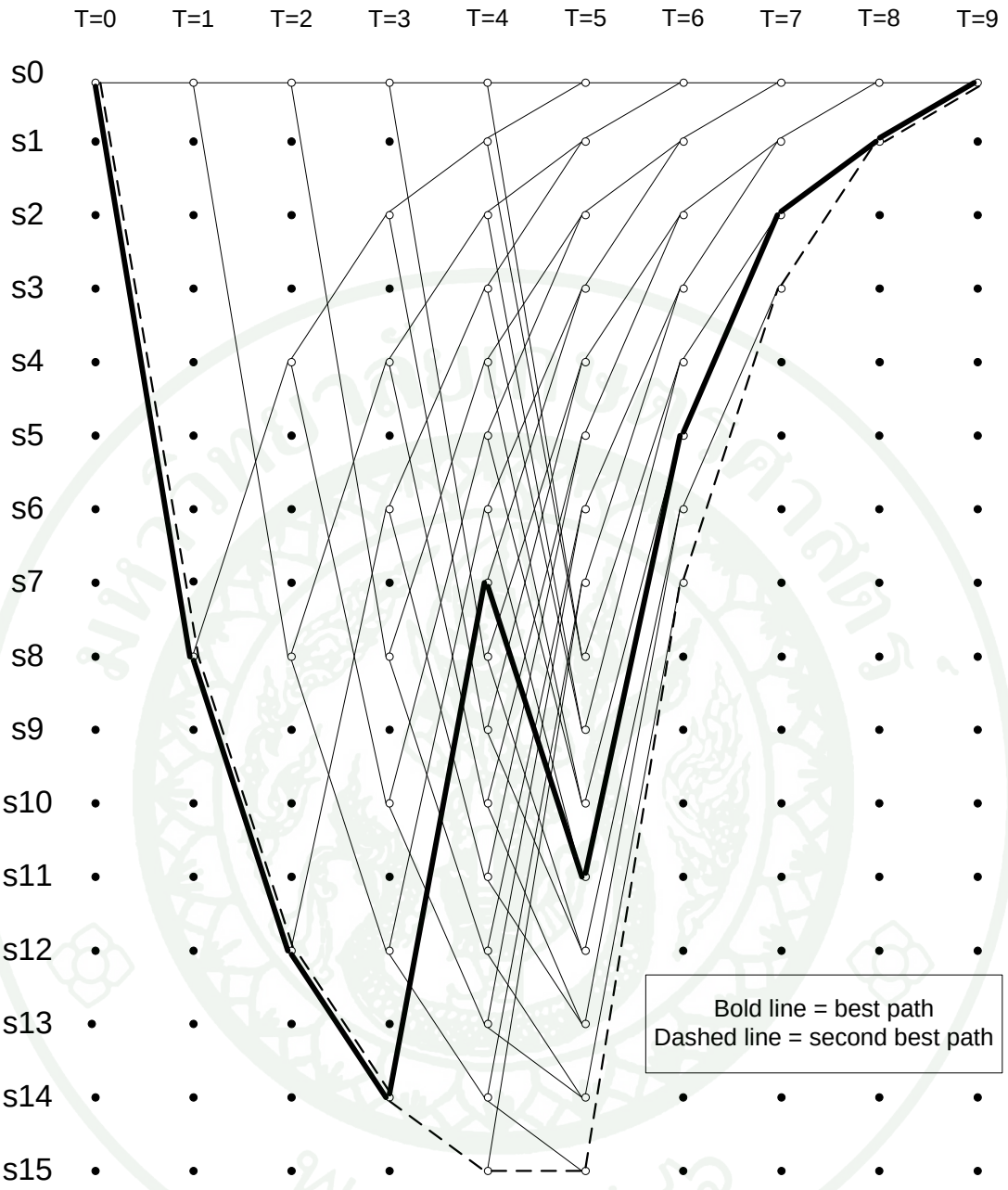
<u>ตัวอย่างที่1</u> พิจารณาการถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ โดยใช้รหัสคอนโวลูชัน (2,1,4) มีค่าฟังก์ชันถ่ายโอน (Transfer function) ดังนี้ $\mathbf{G}(D) = [(1+D+D^4) \quad (1+D+D^3+D^4)]$ โดยเหตุที่ขนาดหน่วยความจำมีค่าเท่ากับ 4 ทำให้จำนวนสเตทเท่ากับ $2^4 = 16$ สเตทในแผนภาพเทรลลิสและต้องป้อนบิทข้อมูลที่เป็นศูนย์ขนาด 4 บิทเข้าไปปิดท้ายข้อมูลหลังจากสิ้นสุดการเข้ารหัส เพื่อล้างค่าที่ค้างอยู่ในหน่วยความจำของวงจรเข้ารหัส ดังภาพที่ 32 แสดงแผนภาพของการเข้ารหัส

คอนโวลูชัน ภาพที่ 33 เป็นการแสดงแผนภาพเทรลลิสสำหรับลำดับของข้อมูล (Input sequence) เท่ากับ **u** = (11101) และในภาพที่ 34 แสดงเมทริกของเส้นทางที่สอดคล้องกับแผนภาพเทรลลิส



ภาพที่ 32 แผนภาพของตัวเข้ารหัสแบบคอนโวลูชัน (2,1,4)

ในการหาเส้นทางที่ดีที่สุดและเส้นทางที่ดีที่สุดรองลงมานั้น อินพุทเมทริกของบิทนำเข้ามาจากโปรแกรม Matlab โดยใช้ฟังก์ชัน raylieghchan() และ awgn() ในการสร้าง Rayliegh fading ที่มีช่องสัญญาณเป็น Additive white Gaussian noise (AWGN) สำหรับ Doppler = 50 และ SNR = 10 ส่วนการมอดูเลท (Modulate) และดีมอดูเลท (Demodulate) จะใช้การมอดูเลทเชิงความถี่แบบไบนารี (Binary Frequency Shift Keying, BFSK) เป็นมอดูเลเตอร์ (Modulator) ในภาคส่งและดีมอดูเลเตอร์ (Demodulator) ในภาครับ สำหรับการเขียนโปรแกรมใน Matlab มีขั้นตอนและรายละเอียดการทำงานตามหัวข้อที่ 5



ภาพที่ 33 แสดงแผนภาพเทรลลิสของการถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์

ที่มา: Tuntoolavest and Noradee (2009)

State	T=0	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8	T=9
0	0	4.15(0)	133.05(0)	194.29(0)	206.43(0)	251.04(1)	310.32(1)	426.65(1)	436.12(0),396.37(1)	532.86(1),457.45(1)
1					250.29(2)	309.55(3)	407.49(3)	378.61(3)	450.41(2), 375(3)	
2				248.96(4)	27.43(4)	224.56(5)	344.09(5)	423.25(5)		
3					309.06(6)	403.3(7)	373.98(7)	374.94(7)		
4			187.72(8)	15.3(8)	158.33(8)	284.29(9)	313.75(9)			
5					223.81(10)	343.32(11)	404.08(11)			
6				307.73(12)	99.56(12)	190.55(13)	347.25(13)			
7					402.8(14)	369.8(15)	370.31(15)			
8		177.7(0)	14.17(0)	134.18(0)	218.44(0)	250.66(1)				
9					283.92(2)	309.69(3)				
10				188.85(4)	39.45(4)	224.18(5)				
11					342.69(6)	403.43(7)				
12			306.6(8)	75.41(8)	146.32(8)	284.67(9)				
13					190.17(10)	343.19(11)				
14				367.85(12)	87.54(12)	190.93(13)				
15					369.17(14)	369.66(15)				

Bold line = best path

Dashed line = second best path

ภาพที่ 34 แสดงค่าเมทริกของเส้นทางของแผนภาพเทรลลิส

ที่มา: Tuntoolavest and Noradee (2009)

เส้นทึบ คือ เส้นทางที่มีความน่าจะเป็นในการถอดรหัสถูกต้องมากที่สุด

เส้นประ คือ เส้นทางที่มีความน่าจะเป็นในการถอดรหัสถูกต้องรองลงมา

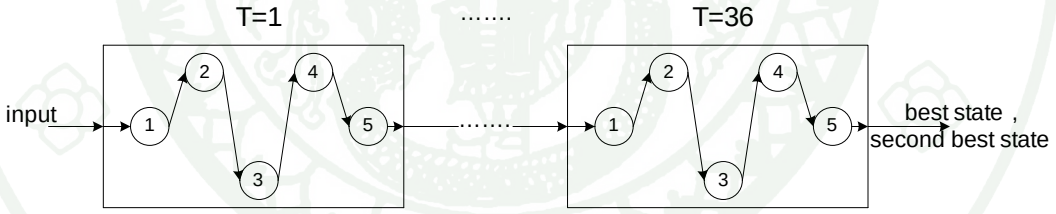
ในภาพที่ 33 จุดทึบและจุดกลวงเป็นตัวกำหนดเส้นทาง โดยที่จุดกลวงแสดงสเตทที่ใช้งานส่วนจุดทึบแสดงสเตทที่ไม่ได้ใช้งาน จำนวนที่อยู่ในแต่ละวงเล็บในภาพที่ 34 คือ สเตทก่อนหน้าที่นำไปสู่เมทริกของเส้นทาง และเพื่อให้ภาพดูง่ายขึ้นจะแสดงเมทริกของเส้นทางที่ดีที่สุดรองลงมาเฉพาะช่วงที่สเตทเหมือนกับเมทริกของเส้นทางที่ดีที่สุด ซึ่งช่วยให้เห็นความแตกต่างของค่าเมทริกเส้นทางที่แตกต่างกัน จนกระทั่งทั้งสองเส้นทางมาบรรจบกันจนถึงจุดสิ้นสุด

เมทริกของกิ่งมีบทบาทสำคัญในเรื่องที่ว่าอัลกอริทึมจะเลือกสเตทที่ดีที่สุดและสเตทที่ดีที่สุดรองลงมาอย่างไรในแต่ละช่วงเวลา เพราะว่าเมทริกของเส้นทางสะสม (Accumulated path metric) มาจากผลรวมของเมทริกของเส้นทางของสเตทก่อนหน้ากับเมทริกของกิ่ง เพื่อความเข้าใจ

ยิ่งขึ้น พิจารณาช่วงเวลา T = 8 และ T = 9 ที่ T = 8 เมทริกของเส้นทางที่สูงที่สุดของสเตท 0 (s0) มีค่าเท่ากับ 436.12 ซึ่งสูงกว่าเมทริกของเส้นทางรองลงมาของสเตท 1 (s1) มีค่าเท่ากับ 375 แต่เมทริกของเส้นทางรองลงมาที่ T = 9 มาจากสเตท 1 (s1) เพราะว่าเมทริกของกิ่งจากสเตท 1 (s1) ถึงสเตท 0 (s0) จากช่วงเวลา T = 8 ถึง T = 9 มีค่าเท่ากับ 82.4487 ซึ่งสูงกว่าเมทริกของกิ่งจากสเตท 0 (s0) ถึงสเตท 0 (s0) จากช่วงเวลา T = 8 ถึงช่วงเวลา T = 9 ที่มีค่าเท่ากับ 0.513184 มาก ดังนั้น เมทริกของเส้นทางสะสมของ 375+82.4487 = 457.4487 ซึ่งสูงกว่า 436.12+0.513184 = 436.633184 จึงเลือกสเตท 1 เป็นทั้งสเตทที่ดีที่สุดและสเตทที่ดีที่สุดรองลงมา

ผลลัพธ์ของการถอดรหัสที่มีโอกาสถูกมากที่สุดเป็น (11 00 00 10 01 01 11 01 11) และรองลงมามีค่าเป็น (11 00 00 01 10 01 10 10 11) ซึ่งแปลงเป็นข้อมูลหลังจากการถอดรหัสคือ (11101) และ (11111) ตามลำดับ

ส่วนของการกำหนดค่าเริ่มต้น (Initialization), การเรียกซ้ำ (Recursion) และการสิ้นสุดลง (Termination) เป็นการรวมการทำงานเข้าด้วยกันและทำงานภายใน 5 สเตทของโปรแกรมภาษา VHDL สำหรับแต่ละช่วงเวลา (Time interval) ดังแสดงในภาพที่ 35



ภาพที่ 35 แผนภาพสเตทส่วนการกำหนดค่าเริ่มต้น (Initialization), การเรียกซ้ำ (Recursion) และการสิ้นสุดลง (Termination)

ที่มา: Tuntoolavest and Noradee (2009)

<u>สเตท1</u> ภาครับทำการรับค่าอินพุทเมทริกของบิทปัจจุบันและกำหนดค่าสเตทของรหัสคอนโวลูชัน (s0 ถึง s15) และเมทริกของกิ่งที่ฟังก์ชันที่ต้องการในการทำงานในแต่ละช่วงเวลา โดยเริ่มต้นการทำงานที่ช่วง T = 1 ซึ่งมีการทำงานเพียง 2 สเตทฟังก์ชันเท่านั้น คือ สเตท s0 และ s8 ดังแสดงในภาพที่ 33

สเตท2 คำนวณหาค่าเมทริกของกิ่ง และบันทึกสเตทของรหัสคอนโวลูชันก่อนหน้าที่สอดคล้องกัน

สเตท3 คำนวณหาค่าเมทริกของเส้นทาง โดยสามารถคำนวณหาค่าได้จาก

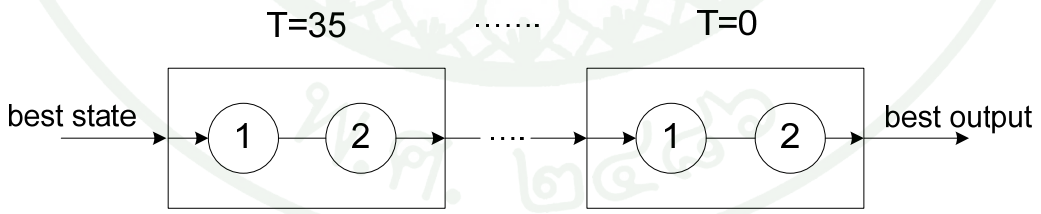
เมทริกของเส้นทาง = เมทริกของเส้นทางก่อนหน้า + เมทริกของกิ่งปัจจุบัน

สเตท4 เปรียบเทียบค่าเมทริกของเส้นทางของแต่ละเส้น และเก็บบันทึกสองค่าที่มากที่สุดเอาไว้

สเตท5 เก็บบันทึกค่าเมทริกของเส้นทางที่ดีที่สุด และเมทริกของเส้นทางที่ดีที่สุดรองลงมา รวมถึงบันทึกค่าสเตทของรหัสคอนโวลูชันก่อนหน้าเมทริกของเส้นทางที่บันทึกไว้เพื่อนำค่าเมทริกเหล่านี้ไปใช้ในส่วนของการตามรอยกลับ (Trace back)

ทำซ้ำ สเตท1 ถึง สเตท5 สำหรับช่วงเวลา T = 2 ถึง T = 36 สังเกตได้ว่า T = 36 นั้น มาจากจำนวนของบิทอินพุท ซึ่งเท่ากับ 32 บิท รวมกับจำนวนของขนาดหน่วยความจำ (m = 4)

สำหรับในส่วนการตามรอยกลับ (Trace back) ของเส้นทางที่ดีที่สุด (Best path) นั้นมีการทำงานอยู่ 2 สเตทของโปรแกรมภาษา VHDL สำหรับแต่ละช่วงเวลา ดังแสดงตามภาพที่ 36



ภาพที่ 36 แผนภาพสเตทสำหรับการตามรอยกลับของเส้นทางที่ดีที่สุด

ที่มา: Tuntoolavest and Noradee (2009)

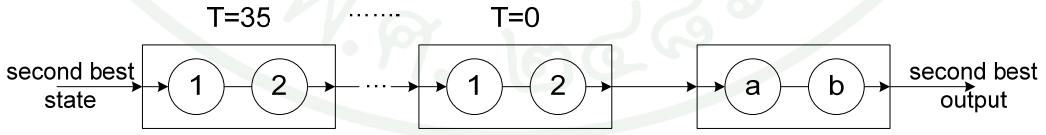
สเตท1 เริ่มต้นการทำงานจาก T = 35 (หรือสำหรับกรณีทั่วไป จะเริ่มที่ช่วงเวลาสุดท้าย – 1) โดยใช้ข้อมูลจากสเตท5 ของภาพที่ 35 มาใช้ในการตามรอยกลับและบันทึกสเตทของ

รหัสคอนโวลูชันของเมทริกของเส้นทางที่ดีที่สุด ณ ช่วงเวลาปัจจุบันเอาไว้ สำหรับตัวอย่างที่ T = 35 อัลกอริทึมทำการเก็บบันทึกค่าสเตทที่ดีที่สุดที่ T = 35 เพื่อนำไปสู่เส้นทางที่ดีที่สุดที่จุดสิ้นสุดของเทรลลิส (T = 36) และที่ T = 34 ทำการเก็บบันทึกค่าสเตทที่ดีที่สุดที่ T = 35 เพื่อนำไปสู่เส้นทางที่ดีที่สุดที่ T = 35 เป็นต้น

สเตท2 ถอดรหัสหาค่าบิทของการถอดรหัสที่ดีที่สุด (Best decoded(data) bit) จากที่พบใน สเตทที่ดีที่สุดที่แต่ละช่วงเวลา

ทำซ้ำ สเตท1 และ สเตท2 สำหรับช่วงเวลา T = 34 ถึง T = 0 เมื่อทำเสร็จสิ้นแล้วจะได้ผลลัพธ์ของสเตทที่ดีที่สุด (Best state sequence) และลำดับของการถอดรหัสที่ดีที่สุด (Best decoded (data) sequence)

สำหรับในส่วนการตามรอยกลับของเส้นทางที่ดีที่สุดรองลงมา (Second best path) อัลกอริทึมจะกระทำเหมือนกับสเตท1และสเตท2 ของเส้นทางที่ดีที่สุด โดยจะแตกต่างกันที่อัลกอริทึมของเส้นทางที่ดีที่สุดรองลงมาจะใช้สเตทที่ดีที่สุดรองลงมา (Second best state) แทนที่ในแต่ละช่วงเวลา ไปสู่การหาสเตทและบิทของการถอดรหัส จะเห็นได้ชัดว่า ผลลัพธ์ที่ได้จะไม่ใช่ผลลัพธ์ของเส้นทางที่ดีที่สุดรองลงมาทั้งหมด เพราะว่าอัลกอริทึมจะทำการเลือกค่าเมทริกที่มีค่าเมทริกต่ำกว่าเสมอที่แต่ละสเตท การทำให้ผลลัพธ์ของเส้นทางที่ดีที่สุดรองลงมาถูกต้อง(Correct second best path) นั้น โดยที่เส้นทางที่ดีที่สุดรองลงมาต้องแยกออก (Diverge) จากเส้นทางที่ดีที่สุดเพียงครั้งเดียวเท่านั้น ทำได้โดยการเพิ่มสเตทการทำงานของโปรแกรมภาษา VHDL เข้าไปที่ตอนท้ายอีก 2 สเตท ดังแสดงในภาพที่ 37



ภาพที่ 37 แผนภาพสเตทสำหรับการตามรอยกลับของเส้นทางที่ดีที่สุดรองลงมา

ที่มา: Tuntoolavest and Noradee (2009)

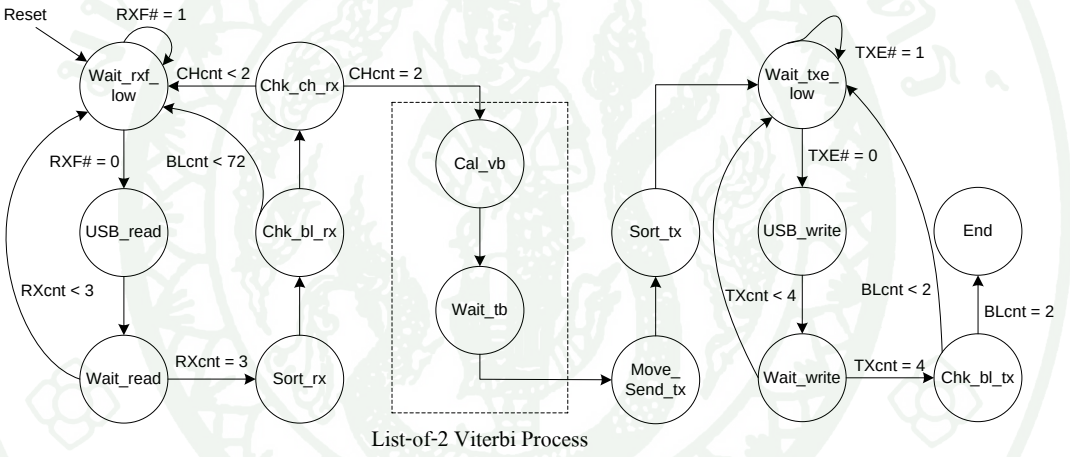
สเตทa เปรียบเทียบสเตทของรหัสคอนโวลูชันที่ได้จากการตามรอยกลับของเส้นทางที่ดีที่สุด กับเส้นทางตามรอยกลับที่ใช้สเตทที่ดีที่สุดรองลงมา โดยเริ่มต้นจาก T = 35, 34, … ไปเรื่อยๆจนกระทั่งถึงช่วงเวลาแรกที่สเตทจากทั้งสองการตามรอยกลับมีความแตกต่างกัน และหยุดที่ช่วงเวลานั้น โดยจะเรียกว่า ช่วงเวลา T_a วิธีการนี้ทำให้ได้ช่วงเวลาที่สเตทของเส้นทางที่ดีที่สุดและเส้นทางที่ดีที่สุดรองลงมา วิ่งเข้าหากันและรวมกันจนกระทั่งสิ้นสุดของแผนภาพเทรลลิส พิจารณาตัวอย่างภาพที่ 33 จากอัลกอริทึมพบว่าสเตทจากสองการตามรอยกลับมีความแตกต่างกันที่ T = 7 ซึ่งวิถีทางดังกล่าวทำให้ทั้งสองเส้นทางหาทางรวมเข้าด้วยกันจาก T = 8 ถึงจุดสิ้นสุดของแผนภาพเทรลลิส

สเตทb ส่วนสำคัญที่ทำให้เส้นทางที่ดีที่สุดรองลงมาถูกต้องคือ เส้นทางนั้นจะมีเมทริกของเส้นทางสูงสุดตลอด จนกระทั่งถึงช่วงเวลาที่มาบรรจบกับเส้นทางที่ดีที่สุด อย่างไรก็ตามยังคงมีเส้นทางอื่นๆที่มีค่าเมทริกของเส้นทางสูงกว่า เพราะฉะนั้น เมื่อสเตทaหยุดที่ช่วง T_a สเตทb จะเริ่มการดำเนินการตามรอยกลับตามภาพที่ 34 แต่เริ่มที่ $T = T_a - 1$ แทนที่จะเริ่มจาก T = 35 (หรือช่วงเวลาสุดท้าย – 1) ซึ่งวิธีการนี้จะใช้เมทริกของเส้นทางที่ดีที่สุดที่แต่ละสเตท โดยพฤติการณ์นี้จะเป็นการหาเส้นทางที่ดีที่สุดโดยใช้สเตทที่ดีที่สุดแทนสเตทที่ดีที่สุดรองลงมา จนกระทั่งสิ้นสุดที่ T = 0 ทำให้ได้ผลลัพธ์ของเส้นทางที่ดีที่สุดรองลงมาถูกต้อง (Correct second best path) และได้ลำดับของการถอดรหัสที่ดีที่สุดรองลงมา (Second best decoded (data) sequence)

4.2 การทำงานของตัวถอดรหัสบนบอร์ด FPGA

จากภาพที่ 38 เมื่อเริ่มต้นทำงานหรือมีการกดปุ่มรีเซต สเตทเริ่มต้นจะเป็น Wait_rxf_low ในสเตทนี้มีการตรวจสอบสัญญาณ RXF จากโมดูล USB ถ้า RXF ยังมีสภาวะเป็นลอจิกสูง สเตทก็จะยังไม่เปลี่ยนแปลงจนกว่า RXF จะมีสภาวะเป็นลอจิกต่ำ สเตทก็จะเปลี่ยนไปเป็น USB_read โดยในสเตทนี้บอร์ด FPGA จะมีการส่งสัญญาณ RD ออกไปให้โมดูล USB เพื่อทำให้โมดูล USB ส่งข้อมูลเข้ามาในบอร์ด FPGA ผ่านบัสข้อมูล D[0..7] โดยข้อมูลที่เข้ามานี้จะถูกนำไปเก็บไว้ในบัฟเฟอร์ฝั่งรับ แล้วจะเปลี่ยนสเตทไปเป็น Wait_read โดยในสเตทนี้จะรอให้ค่าต่างๆที่ทำในสเตท USB_read มีการเปลี่ยนแปลง และมีการตรวจสอบค่าตัวแปร RX_cnt ซึ่งใช้สำหรับนับจำนวนครั้งในการรับข้อมูล ในบทความนี้ได้ใช้การประมวลผลขนาด 24 บิท แต่การทำงานของโมดูล USB สามารถรับหรือส่งข้อมูลได้ครั้งละ 8 บิท ดังนั้นต้องมีการรับข้อมูลเข้ามาถึง 3 ครั้งจึงจะทำให้ได้ขนาดของข้อมูลตามที่ต้องการ การรับข้อมูลเข้ามาเพิ่มจะมีการเปลี่ยนสเตทกลับไปเป็น Wait_rxf_low แล้วเริ่มทำงานเหมือนเดิมจนกว่า RX_cnt มีค่าเท่ากับ 3 หลังจากนั้นก็จะ

เปลี่ยนสเตทไปเป็น Sort_rx โดยในสเตทนี้จะทำการย้ายข้อมูลที่อยู่ในบัฟเฟอร์ฝั่งรับเข้าไปเก็บไว้ในบัฟเฟอร์ที่เตรียมไว้สำหรับเก็บข้อมูลที่มีขนาดเท่ากับ 24 บิท เพื่อใช้ในการประมวลผลต่อไป ในสเตทถัดไปจะเป็น Chk_bl_rx โดยจะทำการตรวจสอบค่าตัวแปร BL_cnt ตัวแปรนี้ใช้สำหรับนับจำนวนค่าที่รับเข้ามา ซึ่งจำนวนสัญลักษณ์ที่ต้องการสำหรับการทำงานของตัวถอดรหัสนี้เท่ากับ 72 ค่า ถ้าหารับค่ายังไม่ครบก็จะกลับไปรับข้อมูลใหม่ในสเตท Wait_rxf_low จนกว่าจะได้ข้อมูลครบทั้ง 72 ค่า เมื่อรับจำนวนค่าเข้ามาครบแล้ว ตัวถอดรหัสจะเปลี่ยนสเตทเป็น Chk_ch_rx เพื่อตรวจสอบเงื่อนไขตัวแปรที่ชื่อ CHcnt ว่ามีค่าเท่ากับ 2 หรือไม่ ถ้าค่าของตัวแปรยังน้อยกว่า 2 ก็แสดงว่ายังรับข้อมูลเข้ามาไม่ครบ 2 ชุด ก็ให้กลับไปที่สเตท Wait_rxf_low เพื่อรอรับข้อมูลจนกว่าจะครบ 2 ชุด หรือจำนวน 144 ค่า และหากรับมาครบแล้วค่าของตัวแปร CHcnt จะมีค่าเท่ากับ 2 ก็จะนำข้อมูลที่รับเข้ามาทั้งหมดไปประมวลผลต่อไปในสเตท List-of-2 Viterbi Process



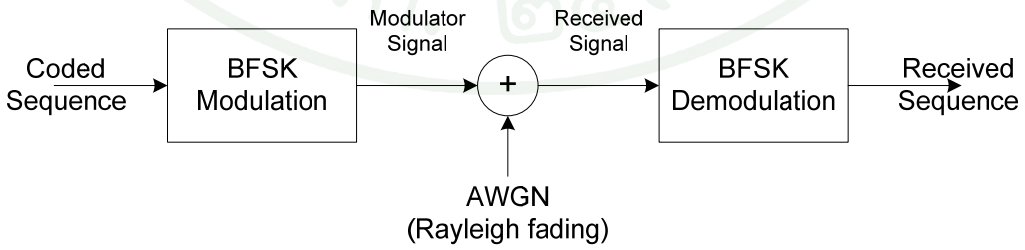
ภาพที่ 38 แผนภาพสเตทการทำงานของตัวถอดรหัสบนบอร์ด FPGA

List-of-2 Viterbi Process มีการทำงาน 2 สเตทย่อย คือ สเตท Cal_vb และสเตท Wait_tb การทำงานในสเตท Cal_vb นี้ จะเป็นส่วนของการกำหนดค่าเริ่มต้น (Initialization), การเรียกซ้ำ (Recursion) และการสิ้นสุดลง (Termination) รายละเอียดสามารถดูจากหัวข้อ 4.1 สเตทต่อไปคือ Wait_tb เป็นส่วนการตามรอยกลับ (Trace back) ของเส้นทางที่ดีที่สุด (Best path) และเส้นทางที่ดีที่สุดรองลงมา (Second best path) รายละเอียดสามารถดูจากหัวข้อ 4.1 เสร็จแล้วจะได้ลำดับของการถอดรหัสที่ดีที่สุด (Best decoded (data) sequence) และได้ลำดับของการถอดรหัสที่ดีที่สุดรองลงมา (Second best decoded (data) sequence)

เมื่อตัวถอดรหัสประมวลผลเสร็จในสเตท List-of-2 Viterbi Process จะมีการนำค่าของผลลัพธ์ที่ได้รับสองค่ามาเก็บไว้ในเอาท์พุทบัฟเฟอร์ฝั่งส่งในสเตท Move_send_tx จากนั้นนำข้อมูลในเอาท์พุทบัฟเฟอร์ไปจัดเรียงและแยกเป็นส่วนๆขนาดส่วนละ 8 บิท ในสเตท Sort_tx เมื่อจัดเรียงเสร็จจะเปลี่ยนสเตทการทำงานไปที่ Wait_tex_low เพื่อตรวจสอบสัญญาณ TXE# ว่ามีสภาวะเป็นสูงหรือต่ำ ถ้าสัญญาณ TXE# อยู่ในสภาวะสูงหรือลอจิก 1 ก็ยังไม่เปลี่ยนแปลงสเตทการทำงาน เนื่องจากบัสข้อมูลยังไม่ว่าง จึงต้องรอให้สัญญาณ TXE# อยู่ในสภาวะต่ำหรือลอจิก 0 ก่อนถึงจะเปลี่ยนสเตทการทำงานไปที่ USB_write เพื่อทำการเขียนข้อมูลขนาด 1 สัญลักษณ์ย่อย (8 บิท) จากเอาท์พุทบัฟเฟอร์ลงใน FIFO ของโมดูล USB หลังจากเขียนเสร็จแล้วก็จะเปลี่ยนสเตทไปที่ Wait_write เพื่อตรวจสอบค่าตัวแปรที่ชื่อ TXcnt ว่ามีค่าเท่ากับ 4 หรือไม่ ถ้าค่าของตัวแปรยังน้อยกว่า 4 ก็แสดงว่ายังส่งข้อมูลไม่ครบ 1 สัญลักษณ์ หรือ 32 บิท ให้เปลี่ยนสเตทไปที่ Wait_txe_low เพื่อส่งข้อมูลที่เหลือให้ครบ 1 สัญลักษณ์ หากส่งข้อมูลครบแล้วตัวแปร TXcnt จะมีค่าเท่ากับ 4 ก็จะเปลี่ยนสเตทการทำงานไปที่ Chk_txe_tx เพื่อตรวจสอบค่าตัวแปรที่ชื่อ BLcnt ว่ามีค่าเท่ากับ 2 หรือไม่ ถ้าค่าของตัวแปรยังน้อยกว่า 2 แสดงว่ายังส่งข้อมูลไม่ครบ 2 ชุด และให้เปลี่ยนสเตทไปที่ Wait_txe_low เพื่อส่งข้อมูลที่เหลือให้ครบ 2 ชุด หรือ 64 บิท หากส่งข้อมูลจนครบแล้ว ตัวแปร BLcnt จะมีค่าเท่ากับ 2 สเตทการทำงานก็จะสิ้นสุดลง

5. การสร้างอินพุทเมทริกของบิทโดยใช้โปรแกรม Matlab

ในการพัฒนาตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ อินพุทที่ใช้มีลักษณะเป็นบิทเมทริก Matlab เป็นโปรแกรมที่เหมาะสมกับการนำไปใช้ในการศึกษาถึงผลกระทบของสัญญาณรบกวนที่มีต่ออัตราการเกิดความผิดพลาด เพราะว่าโปรแกรม Matlab มีฟังก์ชันและชุดคำสั่งการใช้งานที่ง่ายและสะดวกต่อการนำไปใช้ในการหาค่าบิทเมทริก ซึ่งมีลักษณะการทำงานดังภาพที่ 39



ภาพที่ 39 แผนภาพการทำงานของโปรแกรม Matlab ในการหาค่าบิทเมทริก

การทำงานของโปรแกรม Matlab ที่ใช้ในการสร้างอินพุทเมทริกของบิทของการถอดรหัสแบบลิสวิเทอร์บิแบบสองผลลัพธ์ มีขั้นตอนการทำงานดังนี้

ขั้นตอนแรก กำหนดค่าอินพุทของข้อมูลขนาด 32 บิท โดยใช้ลำดับของข้อมูลเป็นศูนย์ทั้งหมด เพื่อให้ง่ายต่อการตรวจสอบความผิดพลาด นำข้อมูลมาเข้ารหัสคอนโวลูชัน (2, 1, 4) จะได้เอาท์พุทจากการเข้ารหัสเป็น 72 บิท ซึ่งจะเป็นอินพุทที่ใช้ในการมอดูเลทต่อไป

ขั้นตอนที่สอง การมอดูเลทแบบ BFSK โดยเริ่มจากการกำหนดค่าเริ่มต้น คือ bit period (T_b) เท่ากับ 1, ความถี่ sampling (f_s) เท่ากับ 8.404 GHz, ความถี่พาห์ (f_c) เท่ากับ 2.099 GHz และ 2.101 GHz ถ้าบิทของข้อมูลเป็น 0 จะใช้ความถี่เป็น 2.099 GHz แต่ถ้าบิทของข้อมูลเป็น 1 จะใช้ความถี่เป็น 2.101 GHz จากนั้นทำการมอดูเลทสัญญาณอินพุทกับความถี่พาห์เข้าด้วยกัน โดยสมการ

$$transmitted = \cos(2\pi f_c t) \qquad \text{โดยที่ } t = 0, \frac{T_b}{f_s}, \frac{2T_b}{f_s}, ..., T_b \tag{28}$$

ขั้นตอนที่สาม จำลองช่องสัญญาณ

3.1 การจำลองช่องสัญญาณเกาส์เซียนขาวแบบบวก

การจำลองช่องสัญญาณเกาส์เซียนขาวแบบบวก สามารถสร้างได้จากสมการ

$$gauss\_noise = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) \tag{29}$$

โดย μ คือ ค่าเฉลี่ยมีค่าเท่ากับศูนย์, σ^2 คือ ค่าความแปรปรวน สามารถหาค่า σ^2 ได้จากสมการ $SNR(dB) = 10\log\frac{E_b}{N_0}$ ซึ่ง SNR (signal-to-noise power ratio) เป็นค่าอัตราระหว่างกำลังของสัญญาณเทียบกับสัญญาณรบกวนมีหน่วยเป็น เดซิเบล (dB) ดังนั้น σ^2 หรือ $\frac{N_0}{2}$ จึงมีค่าเท่ากับ $\frac{E_b}{2\cdot 10^{(SNR(dB)/10)}}$ ซึ่ง E_b คือ กำลังของสัญญาณ และ N_0 คือ ค่ากำลังของสัญญาณรบกวน

เมื่อได้ค่าของสัญญาณรบกวนเกาส์เซียนขาวแบบบวก จากการจำลองช่องสัญญาณ จากนั้นก็นำค่าสัญญาณรบกวนที่ได้ไปรวมกับสัญญาณที่ได้จากการมอดูเลท

3.2 การจำลองช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก (Rayleigh fading with AWGN channel)

การจำลองช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ขั้นตอนแรกเป็นการสร้างสัญญาณการจางหายแบบเรย์ลี โดยสร้างจากสมการการจางหายของแอมพลิจูด (r_i) (Kostov, 2003) แสดงดังสมการ

$$r_i = \sqrt{x_i^2 + y_i^2} \tag{30}$$

เมื่อ x_i และ y_i เป็นตัวแปรแบบเกาส์เซียนชนิดคงที่ที่มีค่าเฉลี่ยเป็นศูนย์

จากนั้นเป็นการสร้างค่าการจางหาย Rayleigh-distributed ที่ไม่มีความสัมพันธ์กัน (uncorrelated) ในโปรแกรม MATLAB และเป็นที่ทราบกันดีว่า ค่าเฉลี่ยกำลังสอง (mean-squared) ของการแจกแจงแบบเรย์ลีมีค่าเท่ากับ $(2\cdot\sigma^2)$ เมื่อ σ^2 เป็นค่าความแปรปรวนของสัญญาณรบกวนเกาส์เซียน เมื่อต้องการให้การแจกแจงแบบเรย์ลีมีค่าเฉลี่ยกำลังสองเท่ากับ 1 ($E(r^2) = 1$) เพื่อที่จะให้กำลังของสัญญาณ (signal power) และ SNR มีค่าเท่ากัน ดังนั้นเพื่อต้องการให้ $E(r^2) = 1$ สามารถเขียนสมการ (30) ให้อยู่ในรูปสมการใหม่ได้ดังนี้

$$r = \sqrt{\frac{\left(x_i + y_i\right)^2}{2*\sigma^2}} \tag{31}$$

เราสามารถนำสมการ (31) มาเขียนเป็นโปรแกรม MATLAB ในรูปของ m-file ได้ดังนี้

```
1. %Rayleigh fading
2. fd = 233.44; %Doppler shift(Hz)
3. M = 10;   %Number of paths.
4. c = sqrt(2/M);
5. w = 2*pi*fd;
6. for n = 1:M
7.    alpha = (2*pi*n-pi+(2*pi*rand-pi))/(4*M);
8.    ph1 = 2*pi*rand-pi;
```

```
9.    ph2 = 2*pi*rand-pi;
10.   x = c*sin(w*t*cos(alpha)+ph1);
11.    y = c*sin(w*t*sin(alpha)+ph2);
12. end
13. R = sqrt(x.^2+y.^2)/sqrt(2*sigma);
```

จากโปรแกรมดังกล่าว (Kostov, 2003) ได้กำหนดให้ $\sigma^2 = 1$ แต่สำหรับในงานวิจัยนี้ไม่ได้กำหนดค่า σ^2 เป็นค่าคงที่ แต่จะกำหนดค่ากำลังของสัญญาณเป็นค่าคงที่ โดยค่า σ^2 จะขึ้นอยู่กับค่า SNR ที่ใช้จากโปรแกรม MATLAB สำหรับการจางหายแบบเรย์ลีที่มีผลของดอปเพลอร์ จะใช้ส่วนประกอบ quardrature ของกระบวนการการจางหายแบบเรย์ลี โดย x_i เป็นค่าตัวอย่างจากการะบวนการสุ่มของ x(t) และ y_i เป็นค่าตัวอย่างจากการะบวนการสุ่มของ y(t) ซึ่งมีความสัมพันธ์กับสมการที่ (30) และ (31) แสดงดังสมการ

$$x(t) = \sqrt{\frac{2}{M}} \sum_{n=1}^{M} \cos\left(\omega_d t \cos\alpha_n + \phi_n\right) \tag{32}$$

$$y(t) = \sqrt{\frac{2}{M}} \sum_{n=1}^{M} \cos\left(\omega_d t \sin\alpha_n + \varphi_n\right) \tag{33}$$

ซึ่ง $\alpha_n = \dfrac{2\pi n - \pi + \theta_n}{4M}, \quad n = 1, 2, 3, ..., M$

เมื่อ ω_d เป็นความถี่ดอปเพลอร์ที่มีมุมสูงสุด (maximum angular Doppler frequency), ϕ_n, φ_n และ θ_n เป็นตัวแปรที่อิสระต่อกันและมีการแจกแจงแบบยูนิฟอร์ม (Uniform) ตั้งแต่ช่วง $[-\pi, \pi)$ สำหรับทุก n

เมื่อได้ค่าการจางหายแบบเรย์ลีแล้ว จากนั้นก็นำค่าสัญญาณการจางหายที่ได้ไปคูณเข้ากับสัญญาณที่ได้จากการมอดูเลท แล้วก็นำไปรวมกับค่าของสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ที่ได้จากข้อ 3.1

3.3 การจำลองช่องสัญญาณการจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก (Ricean fading with AWGN channel)

การจำลองช่องสัญญาณการจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ขั้นตอนแรกเป็นการสร้างสัญญาณการจางหายแบบไรเชียน โดยสร้างจากสมการการจางหายของแอมพลิจูด (r_i) (Kostov, 2003) แสดงดังสมการ

$$r_i = \sqrt{(x_i + \beta)^2 + y_i^2} \tag{34}$$

เมื่อ x_i และ y_i เป็นตัวแปรแบบเกาส์เซียนชนิดคงที่ที่มีค่าเฉลี่ยเป็นศูนย์ และ β^2 เป็นกำลังของเส้นทาง dominant

จากนั้นเป็นการสร้างค่าการจางหาย Ricean-distributed ที่ไม่มีความสัมพันธ์กัน (uncorrelated) ในโปรแกรม MATLAB และเป็นที่ทราบกันดีว่า ค่าเฉลี่ยกำลังสอง (mean-squared) ของการแจกแจงแบบไรเชียนมีค่าเท่ากับ $2 \cdot \sigma^2 (K+1)$ เมื่อ σ^2 เป็นค่าความแปรปรวนของสัญญาณรบกวนเกาส์เซียน เมื่อต้องการให้การแจกแจงแบบไรเชียนมีค่าเฉลี่ยกำลังสองเท่ากับ 1 หรือ $E(r^2) = 1$ เพื่อที่จะให้กำลังของสัญญาณ (signal power) และ SNR มีค่าเท่ากัน ดังนั้นเพื่อต้องการให้ $E(r^2) = 1$ สามารถเขียนสมการ (34) ให้อยู่ในรูปสมการใหม่ได้ดังนี้

$$r = \sqrt{\frac{\left((x_i + \sqrt{2K}) + y_i\right)^2}{2 * \sigma^2 (K+1)}} \tag{35}$$

เมื่อ K เป็นค่าไรเชียนแฟคเตอร์ มีค่าเท่ากับ $K = \frac{\beta^2}{2\sigma^2}$ โดย β คือ แอมพลิจูดที่สูงที่สุด (peak amplitude) ของสัญญาณ

เราสามารถนำสมการ (35) มาเขียนเป็นโปรแกรม MATLAB ในรูปของ m-file ได้ดังนี้

```
1. %Rayleigh fading
2. fd = 233.44; %Doppler shift(Hz)
3. M = 10;   %Number of paths.
4. Kdb = 10;       %K factor(dB)
```

```
5. K = 10^(Kdb/10); Mi = 1;
6. c = sqrt(2/M);
7. w = 2*pi*fd;
8. for n = 1:M
9.    alpha = (2*pi*n-pi+(2*pi*rand-pi))/(4*M);
10.   ph1 = 2*pi*rand-pi;
11.   ph2 = 2*pi*rand-pi;
12.   x = c*sin(w*t*cos(alpha)+ph1);
13.   y = c*sin(w*t*sin(alpha)+ph2);
14.end
15.R=sqrt((x+sqrt(2*K*sigma)).^2+y.^2)/sqrt(1/(2*sigma*(K+1)));
```

จากโปรแกรมดังกล่าว (Kostov, 2003) ได้กำหนดให้ $\sigma^2 = 1$ แต่สำหรับในงานวิจัยนี้ไม่ได้กำหนดค่า σ^2 เป็นค่าคงที่ แต่จะกำหนดค่ากำลังของสัญญาณเป็นค่าคงที่ โดยค่า σ^2 จะขึ้นอยู่กับค่า SNR ที่ใช้จากโปรแกรม MATLAB สำหรับการจางหายแบบไรเชียนที่มีผลของดอปเพลอร์ จะใช้ส่วนประกอบ quardrature ของกระบวนการการจางหายแบบไรเชียน โดย x_i เป็นค่าตัวอย่างจากการะบวนการสุ่มของ x(t) และ y_i เป็นค่าตัวอย่างจากการะบวนการสุ่มของ y(t) ซึ่งมีความสัมพันธ์กับสมการที่ (34) และ (35) แสดงดังสมการ

$$x(t) = \sqrt{\frac{2}{M}} \sum_{n=1}^{M} \cos\left(\omega_d t \cos\alpha_n + \phi_n\right) \tag{36}$$

$$y(t) = \sqrt{\frac{2}{M}} \sum_{n=1}^{M} \cos\left(\omega_d t \sin\alpha_n + \varphi_n\right) \tag{37}$$

ซึ่ง $\alpha_n = \dfrac{2\pi n - \pi + \theta_n}{4M}, \quad n = 1, 2, 3, ..., M$

เมื่อ ω_d เป็นความถี่ดอปเพลอร์ที่มีมุมสูงสุด (maximum angular Doppler frequency) , ϕ_n, φ_n และ θ_n เป็นตัวแปรที่อิสระต่อกันและมีการแจกแจงแบบยูนิฟอร์ม (Uniform) ตั้งแต่ช่วง $[-\pi, \pi)$ สำหรับทุก n

เมื่อได้ค่าการจางหายแบบไรเชียนแล้ว จากนั้นก็นำค่าสัญญาณการจางหายที่ได้ไปคูณเข้ากับสัญญาณที่ได้จากการมอดูเลท แล้วก็นำไปรวมกับค่าของสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ที่ได้จากข้อ 3.1

<u>ขั้นตอนที่สี่</u> ดิจิทัลดีมอดูเลทเป็นขั้นตอนของการแปลข้อมูลดิจิทัลออกมาจากสัญญาณที่ได้รับ โดยใช้วิธีการของ Matched – filter ในการแปลสัญญาณ การแปลสัญญาณที่ได้รับว่าเป็นสัญญาณดิจิทัลบิท 0 หรือ 1 นั้น จะใช้ Matched – filter จำนวน 2 ตัว โดย filter อันแรกเป็นตัวแปลสัญญาณดิจิทัลของอินพุทที่เป็นบิท 0 และอันที่สองเป็นตัวแปลสัญญาณดิจิทัลของอินพุทที่เป็นบิท 1 ซึ่งผลลัพธ์ทั้งสองที่ได้ ก็คือ ค่าอินพุทเมทริกของบิท โดยค่าอินพุทเมทริกของบิทที่ได้จะนำไปใช้เป็นค่าอินพุทของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ต่อไป

ในโครงงานวิทยานิพนธ์นี้ นอกจากจะใช้การมอดูเลทแบบ BFSK แล้ว ยังใช้การมอดูเลทแบบ BPSK ด้วย การมอดูเลทแบบ BPSK มีขั้นตอนการทำงานที่คล้ายกับการมอดูเลทแบบ BFSK โดยการมอดูเลทแบบ BPSK นั้น จะใช้ค่า bit period (T_b) เท่ากับ 1, ความถี่ sampling (f_s) เท่ากับ 8.404 GHz , ความถี่พาห์ (f_c) เท่ากับ 2.101 GHz สำหรับการเปลี่ยนแปลงสภาวะของบิทจาก ‘1’ ไปเป็น ‘0’ หรือเปลี่ยนจาก ‘0’ ไปเป็น ‘1’ เฟสของคลื่นจะเปลี่ยน (Shift) ไป 180 องศา ดังสมการ

$$BPSK\_tx = \begin{cases} \sin(2\pi f_c t), & input = '1' \\ -\sin(2\pi f_c t), & input = '0' \end{cases} \tag{38}$$

โดยที่ $t = 0, \frac{T_b}{f_s}, \frac{2T_b}{f_s}, ..., T_b$

สำหรับการดีมอดูเลทของ BPSK นั้น ก็จะคล้ายกับของ BFSK โดยใช้ Matched - filter จำนวน 2 ตัว และวิธีการทำงานก็เหมือนกับของ BFSK และในส่วนของช่องสัญญาณก็จะใช้ช่องสัญญาณเหมือนกับขั้นตอนที่สามที่ได้กล่าวมาแล้วข้างต้น

6. การคำนวณหาค่า Doppler shift

การหาค่า maximum Doppler shift สำหรับระบบ 3G โดยสมมติความถี่พาห์ (Carrier frequency) ที่ใช้สื่อสารเป็น 2.099 GHz และ 2.101 GHz สำหรับการมอดูเลทแบบ BFSK และความเร็วที่ใช้สำหรับอุปกรณ์สื่อสารเคลื่อนที่ คือ 60, 90 และ 120 กิโลเมตรต่อชั่วโมง โดยสามารถคำนวณหาได้จากสมการต่อไปนี้

$$f_d = \frac{v \times f_c}{c} \tag{39}$$

เมื่อ f_d คือ Doppler shift (Hz), v คือ ความเร็วของการเคลื่อนที่ (เมตรต่อวินาที), f_c คือ ความถี่พาห์ (Hz) และ c คือ ความเร็วแสงมีค่าเท่ากับ $3x10^8$ เมตรต่อวินาที

ความเร็ว 60 Km/h

$$f_d = \frac{60\times10^3\times2.101\times10^9}{60\times60\times3\times10^8} = 116.72Hz$$

ความเร็ว 90 Km/h

$$f_d = \frac{90\times10^3\times2.101\times10^9}{60\times60\times3\times10^8} = 175.08Hz$$

ความเร็ว 120 Km/h

$$f_d = \frac{120\times10^3\times2.101\times10^9}{60\times60\times3\times10^8} = 233.44Hz$$

ในโครงงานวิทยานิพนธ์นี้ นอกจากจะใช้การมอดูเลทแบบ BFSK แล้ว ยังใช้การมอดูเลทแบบ BPSK ด้วย โดยความถี่พาห์ (Carrier frequency) ที่ใช้สื่อสารเป็น 2.101 GHz ความเร็วที่ใช้สำหรับอุปกรณ์สื่อสารเคลื่อนที่ คือ 60, 90 และ 120 กิโลเมตรต่อชั่วโมง เช่นกัน ดังนั้นค่า Doppler shift ที่ได้จึงมีค่าเท่ากัน

7. การเปรียบเทียบความถี่

การเปรียบเทียบค่าความถี่ที่ใช้ในการมอดูเลทในโปรแกรม Matlab สามารถเปรียบเทียบได้จากสมการ

$$transmitted = \cos(2\pi f_c t) \qquad \text{โดยที่ } t = 0, \frac{T_b}{f_s}, \frac{2T_b}{f_s}, ..., T_b \tag{40}$$

โดยค่าความถี่ sampling (f_s) เท่ากับ 8.404 GHz, ความถี่พาห์ (f_c) เท่ากับ 2.099 GHz และ 2.101 GHz, Bit period (T_b) เท่ากับ 1

ที่อินพุทบิทเท่ากับ ‘0’ และ $t = \frac{T_b}{f_s}$ จะได้

$$transmitted = \cos(2\pi f_c t) = \cos(2\pi(2099\times10^6)(\frac{1}{8404\times10^6})) = \cos(2\pi\times\frac{2099}{8404})$$

ที่อินพุทบิทเท่ากับ ‘1’ และ $t = \frac{T_b}{f_s}$ จะได้

$$transmitted = \cos(2\pi f_c t) = \cos(2\pi(2101\times10^6)(\frac{1}{8404\times10^6})) = \cos(2\pi\times\frac{2101}{8404})$$

ที่อินพุทบิทเท่ากับ ‘0’ และ $t = \frac{2T_b}{f_s}$ จะได้

$$transmitted = \cos(2\pi f_c t) = \cos(2\pi(2099\times10^6)(\frac{2}{8404\times10^6})) = \cos(2\pi\times\frac{2099\times2}{8404})$$

ที่อินพุทบิทเท่ากับ ‘1’ และ $t = \frac{2T_b}{f_s}$ จะได้

$$transmitted = \cos(2\pi f_c t) = \cos(2\pi(2101\times10^6)(\frac{2}{8404\times10^6})) = \cos(2\pi\times\frac{2101\times2}{8404})$$

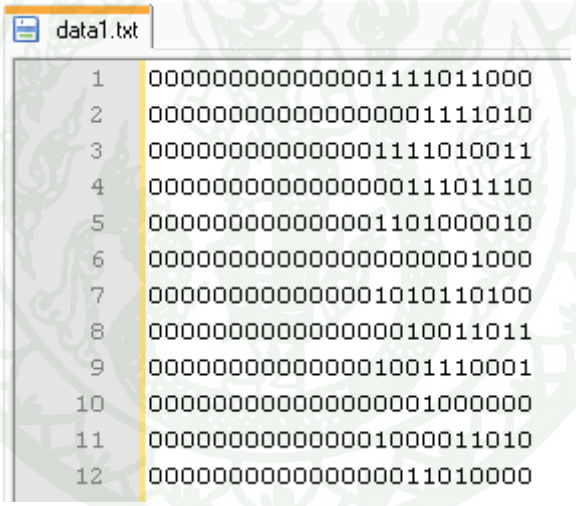
ดังนั้น ค่าที่ได้จากการเปรียบเทียบความถี่ข้างต้น เป็นค่าความถี่ที่นำไปใช้ในการมอดูเลทแบบ BFSK ในโปรแกรม Matlab โดยเปลี่ยนมาใช้ความถี่ใหม่ที่ได้เป็น ความถี่ sampling (f_s) เท่ากับ 8,404 Hz, ความถี่พาห์ (f_c) เท่ากับ 2,099 Hz และ 2,101 Hz ตามลำดับ และการมอดูเลทแบบ BPSK ในโปรแกรม Matlab ก็เปลี่ยนมาใช้ความถี่ใหม่ที่ได้เช่นกัน โดยค่าความถี่ sampling (f_s) เท่ากับ 8,404 Hz, ความถี่พาห์ (f_c) เท่ากับ 2,101 Hz

8. การทดสอบการทำงานของตัวถอดรหัสบนบอร์ด FPGA

หลังจากเตรียมอุปกรณ์และโปรแกรมพร้อมแล้ว ก็เป็นการทดสอบการทำงานของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์บนบอร์ดอิเล็กทรอนิกส์ FPGA ซึ่งมี 3 ขั้นตอนดังต่อไปนี้

8.1 เตรียมไฟล์ที่นำไปถอดรหัส

ข้อมูลที่นำไปถอดรหัสนั้น ได้แบ่งออกเป็น 2 ไฟล์ ซึ่งแต่ละไฟล์มีขนาด 72 ค่า แต่ละค่ามีขนาด 24 บิท เพื่อความสะดวกในการทดสอบ เพราะสามารถนำข้อมูลที่ได้มาใช้ทั้งการจำลองการทำงานและบนบอร์ด FPGA โดยอินพุทข้อมูลที่ใช้นำมาจากโปรแกรม Matlab ทำให้ได้ข้อมูลดังภาพที่ 40



ภาพที่ 40 ตัวอย่างไฟล์ข้อมูลที่ใช้ในการถอดรหัส

8.2 ดาวน์โหลดโปรแกรมลงชิป FPGA

การดาวน์โหลดโปรแกรมลงชิป FPGA ทำได้โดยใช้เครื่องดาวน์โหลดโปรแกรม Xilinx Platform Cable USB นำมาต่อเข้ากับบอร์ด FPGA และเครื่องคอมพิวเตอร์ โดยใช้โปรแกรม iMPACT ที่มากับชุดของโปรแกรม Xilinx ISE ทำการดาวน์โหลดโปรแกรมลงชิป FPGA ซึ่งโปรแกรมที่ดาวน์โหลดนั้นได้จากขั้นตอนการ Implementation ของโปรแกรม Xilinx ISE

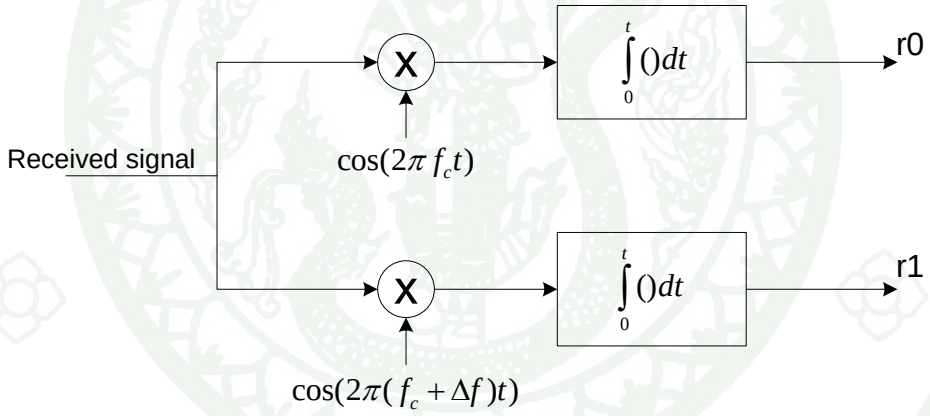
8.3 ส่งไฟล์ที่นำไปถอดรหัสและบันทึกผล

การส่งไฟล์จากคอมพิวเตอร์เข้าไปถอดรหัสยังบอร์ด FPGA ทำได้โดยใช้เครื่องดาวน์โหลดโปรแกรม USB Transceiver เป็นตัวควบคุมการรับส่งข้อมูล เมื่อส่งข้อมูลที่ต้องการทดสอบลงชิป FPGA เสร็จแล้ว ก็ต้องทำการบันทึกผลที่ได้จากการทดสอบการทำงานบนบอร์ด FPGA เพื่อนำข้อมูลที่ได้มาตรวจสอบว่าทำงานถูกต้องหรือไม่ต่อไป

ผลและวิจารณ์

ผล

ผลทดสอบการทำงานของตัวถอดรหัสบนบอร์ด FPGA โดยข้อมูลอินพุทที่ใช้ทดสอบการถอดรหัสเป็นข้อมูลที่มีความยาว 32 บิท ซึ่งเมื่อนำไปเข้ารหัสคอนโวลูชัน (2,1,4) ดังแสดงตามภาพที่ 2 แล้วจะได้คำรหัสที่มีขนาด 72 บิท จากนั้นคำรหัสจะถูกมอดูเลทแบบดิจิทัลด้วยเทคนิค BFSK และ BPSK สัญญาณที่ถูกมอดูเลทแล้วจะถูกส่งผ่านไปยังช่องสัญญาณที่จำลองขึ้นมา จากนั้นเครื่องรับจะทำการดีมอดูเลท โดยใช้ Matched-filter จำนวน 2 ตัว โดย Matched-filter ตัวแรกถูกออกแบบให้เข้าคู่กับสัญญาณของบิท “0” และ Matched-filter ตัวที่สองถูกออกแบบให้เข้าคู่กับสัญญาณของบิท “1” ดังแสดงในภาพที่ 41



ภาพที่ 41 การดีมอดูเลทสัญญาณโดยใช้ Matched-filter

จากภาพที่ 41 ค่า r0 เป็นผลลัพธ์ของการเข้าคู่ (Matched) สัญญาณที่ได้รับกับ Matched-filter ตัวแรกและค่า r1 เป็นผลลัพธ์ของการเข้าคู่ (Matched) สัญญาณที่ได้รับกับ Matched-filter ตัวที่สอง ซึ่งผลลัพธ์ทั้งสองที่ได้นี้ จะถูกนำไปใช้เป็นค่าอินพุทเมทริกของแต่ละบิท (bit metric) สำหรับการถอดรหัสด้วยวิธีลิสวิเทอร์บิ ค่าบิทเมทริก r0 และ r1 นี้เป็นค่าจำนวนจริง ซึ่งสัญญาณแต่ละบิทที่เครื่องรับได้รับจะให้ผลลัพธ์ของการดีมอดูเลทเป็น r0 จำนวน 1 ค่าและ r1 จำนวน 1 ค่า ดังนั้นสำหรับคำรหัสขนาด 72 บิทที่ใช้จะทำให้เกิดค่า r0 จำนวน 72 ค่า และ r1 จำนวน 72 ค่า เมื่อส่งไปยังบอร์ด FPGA ข้อมูลที่บอร์ดได้รับจะเท่ากับ 144 ค่า แต่เนื่องจากบอร์ด FPGA ต้องรับข้อมูลในรูปแบบไบนารี ดังนั้นค่าบิทเมทริกที่ได้จะถูกแปลงเป็นลำดับเลขฐาน 2 โดยใช้ 24 บิทต่อ

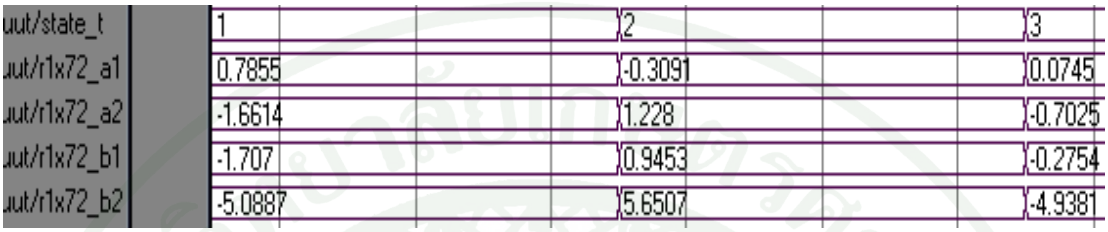
1 ค่า จากนั้นต้องแปลงค่าลำดับเลขฐาน 2 เป็นลำดับเลขฐาน 16 จากขนาด 24 บิท เป็นขนาด 3 ไบท์ (1 ไบท์ เท่ากับ 8 บิท) เพราะโปรแกรม USB Transceiver for List Viterbi V1.1 with DEBUG protocol เป็นโปรแกรมที่ใช้ในการรับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์กับบอร์ด FPGA ไม่สามารถรับส่งข้อมูลที่เป็นเลขไบนารีได้โดยตรง ต้องแปลงเป็นลำดับเลขฐาน 16 ก่อนจึงสามารถรับส่งข้อมูลได้ และผลที่ได้จากการทดสอบตัวถอดรหัสมีดังนี้

1. ผลการจำลองการทำงานของตัวถอดรหัส

ผลลัพธ์ที่ได้จากการจำลองการทำงานของโปรแกรมภาษา VHDL เปรียบเทียบกับโปรแกรมภาษา C++ เพื่อต้องการที่จะทราบว่าโปรแกรมภาษา VHDLนั้น ทำงานได้ถูกต้องหรือไม่ โดยเปรียบเทียบการทำงานของทุกช่วงของโปรแกรม แต่ในหัวข้อนี้จะนำผลลัพธ์ที่สำคัญมาพิจารณาเท่านั้น ส่วนผลลัพธ์ของโปรแกรม C++ ที่แสดงในรูปภาพและตารางมาจากโปรแกรมที่ใช้ในการแสดงประสิทธิภาพของการถอดรหัสคอนโวลูชันเวกเตอร์ซิมโบล (Tuntoolavest, 2003) โดยโปรแกรม C++ มีการเปลี่ยนแปลงการรับค่าอินพุท ซึ่งรับค่าอินพุทบิทเมทริกจากโปรแกรม Matlab เข้ามาแทนที่การจำลองการทำงานของช่องสัญญาณด้วยตัวมันเอง ทำให้ทั้งสองโปรแกรมสามารถเปรียบเทียบผลลัพธ์ที่ได้โดยตรง

สำหรับผลการจำลองการทำงานจะใช้อินพุทบิทเมทริกของช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเป็นเกาส์เซียนขาวแบบบวก (Additive white Gaussian noise: AWGN) สำหรับพารามิเตอร์ต่างๆสร้างโดยใช้โปรแกรม Matlab ผลลัพธ์ที่แสดงเป็นลำดับของข้อมูลที่เป็นศูนย์ทั้งหมด 32 บิท โดยมีพารามิเตอร์ของช่องสัญญาณคือ Doppler = 50 Hz และ SNR = 10 สำหรับวิธีการมอดูเลท (Modulation) ที่ใช้เป็น BFSK (Binary frequency shift keying) และที่ภาครับเป็นคู่ของ Mached-filter ที่จับคู่กับ 2 ความถี่พาห์ (Carrier frequency) การมอดูเลทและการดีมอดูเลท (Demodulation) เป็นการสร้างจากโปรแกรม Matlab โดยผลลัพธ์ที่ได้จากโปรแกรมภาษา VHDL จะแสดงด้วยรูปคลื่น (Waveform) ของสัญญาณ โดยรูปคลื่นของแต่ละภาพจะแสดงถึงช่วงเวลาและค่าที่ได้รับในแต่ละช่วงเวลา ส่วนผลลัพธ์ที่ได้จากโปรแกรม C++ เป็นการอ่านจากแฟ้มข้อมูล (File) ของเอาท์พุท เพราะผลลัพธ์ที่ได้จากการจำลองการทำงานจะถูกเก็บไว้ในรูปของแฟ้มข้อมูล จากนั้นจะย่อผลลัพธ์ไว้ในตารางเพื่อความสะดวกในการนำมาเปรียบเทียบ โดยเป็นการแสดงค่าบางช่วงเท่านั้น

ภาพที่ 42 แสดงอินพุทบิทเมทริก (เอาท์พุทของดีมอดูเลเตอร์) ซึ่งสร้างมาจากโปรแกรม Matlab โดยใช้เป็นอินพุทของการถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ ในโปรแกรมภาษา VHDL จากภาพ state_t ทำหน้าที่ในการกำหนดช่วงเวลาการทำงานของโปรแกรม ส่วน r1x72_a1, r1x72_a2, r1x72_b1 และ r1x72_b2 เป็นค่าอินพุทบิทเมทริกที่ใช้ในการถอดรหัส



ภาพที่ 42 อินพุทบิทเมทริกสร้างโดยโปรแกรม Matlab

อธิบายรายละเอียดที่แสดงอยู่ในภาพที่ 42 ได้ดังนี้

state_t เป็นค่าที่ใช้ในการกำหนดช่วงเวลาการทำงานจาก T = 1 ถึง T = 36

r1x72_a1 ค่าอินพุทที่มาจากบิทเมทริกของ r0 จาก Mached filter

r1x72_b1 ค่าอินพุทที่มาจากบิทเมทริกของ r1 จาก Mached filter

r1x72_a2 ค่าอินพุทที่มาจากบิทเมทริกของ r0 จาก Mached filter และเป็นค่าบิทเมทริกที่ถัดจากค่า r1x72_a1

r1x72_b2 ค่าอินพุทที่มาจากบิทเมทริกของ r1 จาก Mached filter และเป็นค่าบิทเมทริกที่ถัดจากค่า r1x72_b1

ภาพที่ 43 และตารางที่ 3 เป็นการแสดงเมทริกของเส้นทางที่ดีที่สุดจำนวน 3 ช่วง (T = 5 ถึง T = 7) ระหว่างสถานะคงที่จากโปรแกรมภาษา VHDL และโปรแกรม C++ จากภาพที่ 43 pmetric_f0 ถึง pmetric_f15 แสดงค่าเมทริกของเส้นทางที่ดีที่สุดตั้งแต่สเตท 0 ถึงสเตท 15 ของแต่ละช่วงเวลา

/testbench1/uut/pmetric_f0	81.6292	85.3168	117.768
/testbench1/uut/pmetric_f1	78.6893	109.141	122.177
/testbench1/uut/pmetric_f2	109.075	98.6675	132.912
/testbench1/uut/pmetric_f3	67.0617	111.968	121.138
/testbench1/uut/pmetric_f4	89.7548	93.733	138.67
/testbench1/uut/pmetric_f5	92.04	124.285	145.145
/testbench1/uut/pmetric_f6	111.902	95.0296	134.478
/testbench1/uut/pmetric_f7	54.0738	116.159	130.001
/testbench1/uut/pmetric_f8	87.1055	88.2566	129.001
/testbench1/uut/pmetric_f9	78.8707	118.809	135.477
/testbench1/uut/pmetric_f10	114.552	96.3823	144.146
/testbench1/uut/pmetric_f11	67.2431	121.636	139.669
/testbench1/uut/pmetric_f12	84.2785	90.2765	127.436
/testbench1/uut/pmetric_f13	91.8586	114.617	126.614
/testbench1/uut/pmetric_f14	106.426	98.4861	123.244
/testbench1/uut/pmetric_f15	53.8924	106.491	121.995

ภาพที่ 43 ตัวอย่างเมทริกของเส้นทางที่ดีที่สุดสำหรับช่วง T = 5 ถึง T = 7 จากโปรแกรมภาษา VHDL

อธิบายรายละเอียดที่แสดงอยู่ในภาพที่ 43 ได้ดังนี้

pmetric_fi เป็นเมทริกของเส้นทางที่ดีที่สุดของสเตท si โดยที่ i = 0, 1, 2, 3,…,15

ตารางที่ 3 ตัวอย่างเมทริกของเส้นทางที่ดีที่สุดสำหรับช่วง T = 5 ถึง T = 7 จากโปรแกรม C++

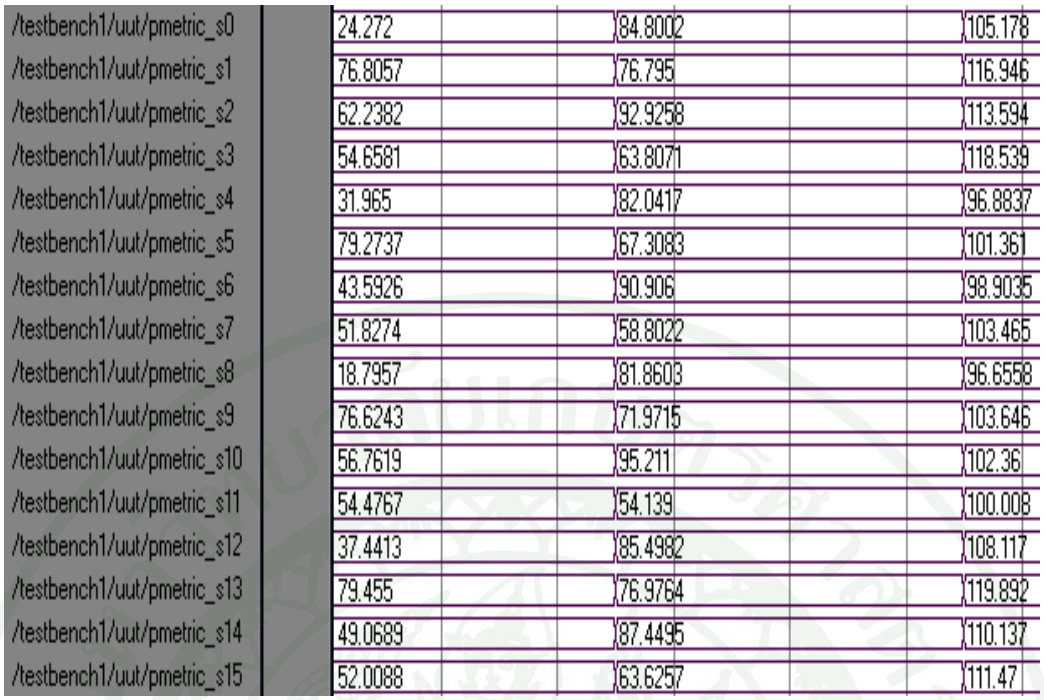
State	T = 5	T = 6	T = 7
0	81.2692	85.3168	117.7680
1	78.8693	109.1410	122.1770
2	109.0750	98.6675	132.9120
3	67.0617	111.9680	121.1380
4	89.7548	93.7330	138.6700
5	92.0400	124.2850	145.1450
6	111.9020	95.0296	134.4780

ตารางที่ 3 (ต่อ)

State	T = 5	T = 6	T = 7
7	54.0738	116.1590	130.0010
8	87.1055	88.2566	129.0010
9	78.8707	118.8090	135.4770
10	114.5520	96.3823	144.1460
11	67.2431	121.6360	139.6690
12	84.2785	90.2765	127.4360
13	91.8586	114.6170	126.6140
14	106.4260	98.4861	123.2440
15	53.8924	106.4910	121.9950

จากภาพที่ 43 และตารางที่ 3 พบว่าค่าเมทริกของเส้นทางที่ดีที่สุดที่ได้จากโปรแกรมภาษา VHDL และโปรแกรม C++ นั้น มีค่าเหมือนกันหมด เช่น ที่ T = 5 ค่า pmetric_f8 และค่าที่สเตท 8 มีค่าเท่ากัน คือ 87.1055 เป็นต้น

ภาพที่ 44 และตารางที่ 4 เป็นการแสดงเมทริกของเส้นทางที่ดีที่สุดรองลงมาจำนวน 3 ช่วง (T = 5 ถึง T = 7) ระหว่างสถานะคงที่จากโปรแกรมภาษา VHDL และโปรแกรม C++ จากภาพที่ 44 pmetric_s0 ถึง pmetric_s15 แสดงค่าเมทริกของเส้นทางที่ดีที่สุดรองลงมาตั้งแต่สเตท 0 ถึงสเตท 15 ของแต่ละช่วงเวลา



/testbench1/uut/pmetric_s0	24.272	84.8002	105.178
/testbench1/uut/pmetric_s1	76.8057	76.795	116.946
/testbench1/uut/pmetric_s2	62.2382	92.9258	113.594
/testbench1/uut/pmetric_s3	54.6581	63.8071	118.539
/testbench1/uut/pmetric_s4	31.965	82.0417	96.8837
/testbench1/uut/pmetric_s5	79.2737	67.3083	101.361
/testbench1/uut/pmetric_s6	43.5926	90.906	98.9035
/testbench1/uut/pmetric_s7	51.8274	58.8022	103.465
/testbench1/uut/pmetric_s8	18.7957	81.8603	96.6558
/testbench1/uut/pmetric_s9	76.6243	71.9715	103.646
/testbench1/uut/pmetric_s10	56.7619	95.211	102.36
/testbench1/uut/pmetric_s11	54.4767	54.139	100.008
/testbench1/uut/pmetric_s12	37.4413	85.4982	108.117
/testbench1/uut/pmetric_s13	79.455	76.9764	119.892
/testbench1/uut/pmetric_s14	49.0689	87.4495	110.137
/testbench1/uut/pmetric_s15	52.0088	63.6257	111.47

ภาพที่ 44 ตัวอย่างเมทริกของเส้นทางที่ดีที่สุดรองลงมาสำหรับช่วง T = 5 ถึง T = 7 จากโปรแกรมภาษา VHDL

อธิบายรายละเอียดที่แสดงอยู่ในภาพที่ 44 ได้ดังนี้

pmetric_si เป็นเมทริกของเส้นทางที่ดีที่สุดรองลงมาของสเตท si โดยที่ i = 0, 1, 2, 3,…,15

ตารางที่ 4 ตัวอย่างเมทริกของเส้นทางที่ดีที่สุดรองลงมาสำหรับช่วง T = 5 ถึง T = 7 จากโปรแกรม C++

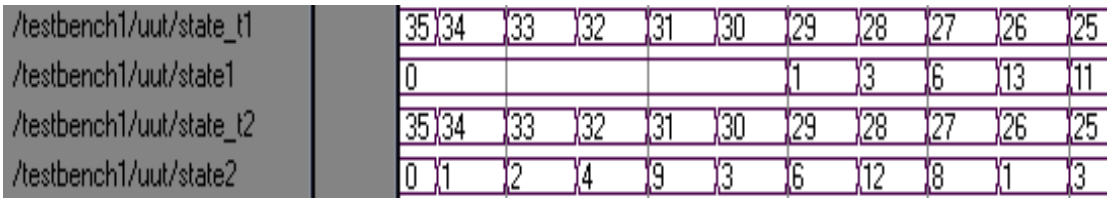
State	T = 5	T = 6	T = 7
0	24.2720	84.8002	105.1780
1	76.8057	76.7950	116.9460
2	62.2382	92.9258	113.5940
3	54.6581	63.8071	118.5390
4	31.9650	82.0417	96.8837

ตารางที่ 4 (ต่อ)

State	T = 5	T = 6	T = 7
5	79.2737	67.3083	101.3610
6	43.5926	90.9060	98.9035
7	51.8274	58.8022	103.4650
8	18.7957	81.8603	96.6558
9	76.6243	71.9715	103.6460
10	56.7619	95.2110	102.3600
11	54.4767	54.1390	100.0080
12	37.4413	85.4982	108.1170
13	79.4550	76.9764	119.8920
14	49.0689	87.4495	110.1370
15	52.0088	63.6257	111.4700

จากภาพที่ 44 และตารางที่ 4 พบว่าค่าเมทริกของเส้นทางที่ดีที่สุดรองลงมาที่ได้จากโปรแกรมภาษา VHDL และโปรแกรม C++ นั้น มีค่าเหมือนกันหมด เช่น ที่ T = 5 ค่า pmetric_s8 และค่าที่สเตท 8 มีค่าเท่ากัน คือ 18.7957 เป็นต้น

ภาพที่ 45 ตารางที่ 5 และ 6 แสดงลำดับของสเตทที่ดีที่สุดและสเตทที่ดีที่สุดรองลงมาจากโปรแกรมภาษา VHDL และโปรแกรม C++ จากภาพ state_t1 และ state_t2 ทำหน้าที่ในการกำหนดช่วงเวลาการตามรอยกลับของโปรแกรม state1 และ state2 เป็นค่าสเตทที่ได้จากการตามรอยกลับ จากตาราง 5 และ 6 T ทำหน้าที่ในการกำหนดช่วงเวลาการตามรอยกลับ best_state_seq เป็นค่าสเตทที่ดีที่สุดที่ได้จากการตามรอยกลับและ sec_state_seq เป็นค่าสเตทที่ดีที่สุดรองลงมาที่ได้จากการตามรอยกลับ



ภาพที่ 45 ลำดับของสเตทที่ดีที่สุดและสเตทที่ดีที่สุดรองลงมาจากการตามรอยกลับ(T=35 ถึง T=25)

อธิบายรายละเอียดที่แสดงอยู่ในภาพที่ 45 ได้ดังนี้

State_t1 เป็นค่าที่ใช้ในการกำหนดช่วงการทำงานของการตามรอยกลับ ตั้งแต่ T = 35 ถึง T = 25 ของผลลัพธ์ที่ควรจะเป็นมากที่สุด

State1 เป็นค่าสเตทที่ได้จากการตามรอยกลับของผลลัพธ์ที่ควรจะเป็นมากที่สุด

State_t2 เป็นค่าที่ใช้ในการกำหนดช่วงการทำงานของการตามรอยกลับ ตั้งแต่ T = 35 ถึง T = 25 ของผลลัพธ์ที่ควรจะเป็นรองลงมา

State2 เป็นค่าสเตทที่ได้จากการตามรอยกลับของผลลัพธ์ที่ควรจะเป็นรองลงมา

ตารางที่ 5 ลำดับของสเตทที่ดีที่สุดจากการตามรอยกลับ(T=35 ถึง T=25) จาก C++

T	35	34	33	32	31	30	29	28	27	26	25
best_state_seq[t] is	0	0	0	0	0	0	1	3	6	13	11

ตารางที่ 6 ลำดับของสเตทที่ดีที่สุดรองลงมาจากการตามรอยกลับ(T=35 ถึง T=25) จาก C++

T	35	34	33	32	31	30	29	28	27	26	25
sec_state_seq[t] is	0	1	2	4	9	3	6	12	8	1	3

ภาพที่ 45 ตารางที่ 5 และ 6 พบว่าลำดับของสเตทที่ดีที่สุดและดีที่สุดรองลงมาที่ได้จากโปรแกรมภาษา VHDL และโปรแกรม C++ นั้น มีค่าเหมือนกัน เช่น ที่ State_t1 หรือ T = 30

ค่า state1 และค่า best_state_seq มีค่าเหมือนกันคือ 0 และที่ State_t2 หรือT = 30 ค่า state2 และค่า sec_state_seq มีค่าเหมือนกันคือ 3 เป็นต้น

ภาพที่ 46 แสดงลำดับข้อมูลของการถอดรหัสที่ดีที่สุดและการถอดรหัสที่ดีที่สุดรองลงมาจากโปรแกรมภาษา VHDL โดย rec_out1 เป็นลำดับของการถอดรหัสที่ดีที่สุด และ rec_out2 เป็นลำดับของการถอดรหัสที่ดีที่สุดรองลงมา จากกรณีนี้สามารถเห็นได้ว่าลำดับของการถอดรหัสที่ดีที่สุดเป็นการถอดรหัสที่ถูกต้องเพียงผลลัพธ์เดียว เพราะว่าอินพุทเป็นลำดับของศูนย์ทั้งหมด ซึ่งเหมือนกับผลลัพธ์ของการถอดรหัสที่ดีที่สุด

ภาพที่ 46 ลำดับข้อมูลของการถอดรหัสที่ดีที่สุดและการถอดรหัสที่ดีที่สุดรองลงมา จาก VHDL

อธิบายรายละเอียดที่แสดงอยู่ในภาพที่ 46 ได้ดังนี้

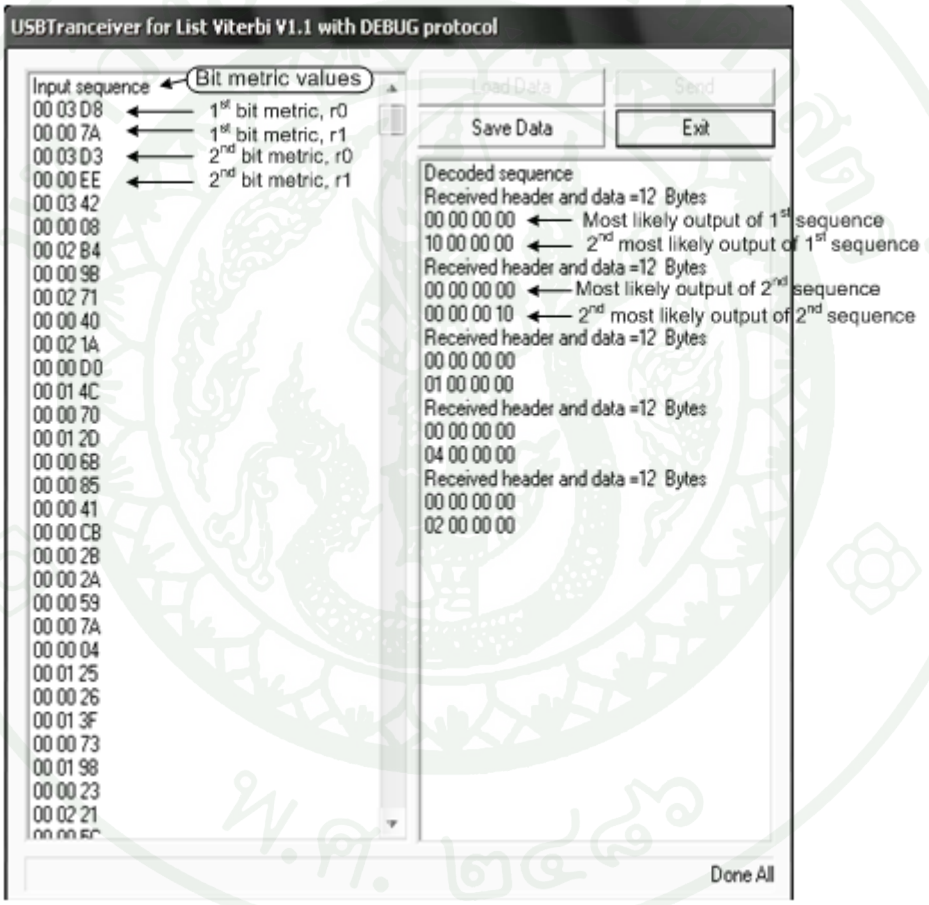
rec_out1 ผลลัพธ์ที่ควรจะเป็นมากที่สุด
rec_out2 ผลลัพธ์ที่ควรจะเป็นรองลงมา

จากผลการทดลองทั้งหมดที่ได้พบว่า ผลลัพธ์ของการถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ที่ได้จากโปรแกรมภาษา VHDL เปรียบเทียบกับผลลัพธ์จากโปรแกรม C++ โดยเปรียบเทียบการทำงานในช่วงเวลาต่างๆของโปรแกรม ซึ่งผลลัพธ์ที่ได้มีค่าเหมือนกัน

2. ผลการทำงานของตัวถอดรหัสบนบอร์ด FPGA

ผลการทดสอบการทำงานของตัวถอดรหัสบนบอร์ด FPGA ข้อมูลที่ใช้ทดสอบเป็นลำดับข้อมูลเป็นศูนย์ทั้งหมด (all-zero sequence) โดยทดสอบจำนวน 5 ครั้ง และใช้ช่องสัญญาณแบบเกาส์เซียนขาวแบบบวก กำหนดค่า SNR = 10 ดังแสดงในภาพที่ 47 จากรูปเป็นโปรแกรม “USB Transceiver for List Viterbi V1.1 with DEBUG protocal” ที่ใช้ในการรับส่งข้อมูลกับบอร์ด FPGA พร้อมทั้งมีการแสดงผลค่าที่ถูกส่งเข้าบอร์ด FPGA และค่าที่รับมาจากบอร์ด FPGA ซึ่งประกอบไป

ด้วยหน้าต่างย่อย 2 ช่อง ได้แก่ หน้าต่าง “Input sequence” แสดงค่าอินพุทเมทริกของแต่ละบิทที่ส่งเข้าบอร์ด FPGA และหน้าต่าง “Decoded sequence” แสดงผลลัพธ์ที่ถอดรหัสเสร็จแล้วส่งมาจากบอร์ด FPGA โดยแต่ละหน้าต่างอ่านค่าจากซ้ายไปขวา แล้วอ่านบนลงล่าง โดยค่าที่อยู่ในหน้าต่าง Input sequence มีขนาด 3 ไบท์ หรือ 24 บิทในหนึ่งแถว และในหน้าต่าง Decoded sequence มีขนาด 4 ไบท์ หรือ 32 บิท ในหนึ่งแถว ส่วนผลลัพธ์ที่ได้จากการถอดรหัสในแต่ละครั้งจะได้ผลลัพธ์ 2 เอาท์พุท โดยผลลัพธ์แรกแสดงเอาท์พุทของการถอดรหัสที่ดีที่สุด และผลลัพธ์ที่สองแสดงเอาท์พุทของการถอดรหัสที่ดีที่สุดรองลงมา



ภาพที่ 47 ผลการถอดรหัสบนบอร์ด FPGA กรณีข้อมูลมีค่าเป็นศูนย์ทั้งหมดและค่า SNR มีค่าสูง

การอ่านค่าของภาพที่ 47 สามารถอ่านค่าและเปรียบเทียบได้จากตารางที่ 7 ซึ่งเป็นตารางเปรียบเทียบเลขฐาน 2 ขนาด 4 บิท กับเลขฐาน 16 ตัวอย่างเช่น หน้าต่าง Input sequence แสดงค่าบิทเมทริกของอินพุทที่ส่งเข้าบอร์ด FPGA โดยค่าที่แสดงเป็นเลขฐาน 16 เช่น ค่าบิทเมทริกแรกคือ 00 03 D8 เป็นเลขฐาน 16 เมื่อแปลงเป็นเลขฐาน 2 จะได้ 00000000 00000011 11011000 เป็นต้น

และหน้าต่าง Decoded sequence แสดงเอาท์พุทที่ถอดรหัสเสร็จแล้วส่งมาจากบอร์ด FPGA ซึ่งมีข้อความ Received header and data = 12 Bytes เป็นตัวเว้นวรรคการถอดรหัสในแต่ละครั้ง แต่ละครั้งของการถอดรหัสจะได้ผลลัพธ์ 2 เอาท์พุท โดยเอาท์พุทที่แสดงเป็นเลขฐาน 16 เช่น ผลลัพธ์แรกของการถอดรหัส เป็นเอาท์พุทของการถอดรหัสที่ดีที่สุดคือ 00 00 00 00 เมื่อแปลงเป็นเลขฐาน 2 จะได้ 00000000 00000000 00000000 00000000 และผลลัพธ์ที่สองเป็นเอาท์พุทของการถอดรหัสที่ดีที่สุดรองลงมาคือ 10 00 00 00 เมื่อแปลงเป็นเลขฐาน 2 จะได้ 00010000 00000000 00000000 00000000 เป็นต้น หากต้องการทราบค่าเอาท์พุทที่ได้จากการถอดรหัสว่าถูกต้องหรือไม่ จะต้องนำค่าเอาท์พุทที่ได้ไปเปรียบเทียบกับข้อมูลที่ถูกส่งเข้าเครื่องถอดรหัสคอนโวลูชัน (2,1,4) จากภาพที่ 47 ผลลัพธ์ที่ได้จากการถอดรหัสที่ดีที่สุดสามารถถอดรหัสได้ถูกต้องทั้งหมด เพราะค่า SNR ที่ใช้มีค่ามากพอที่ทำให้ถอดรหัสได้ถูกต้องทั้งหมด

ตารางที่ 7 ตารางเปรียบเทียบเลขไบนารีกับเลขฐาน 16

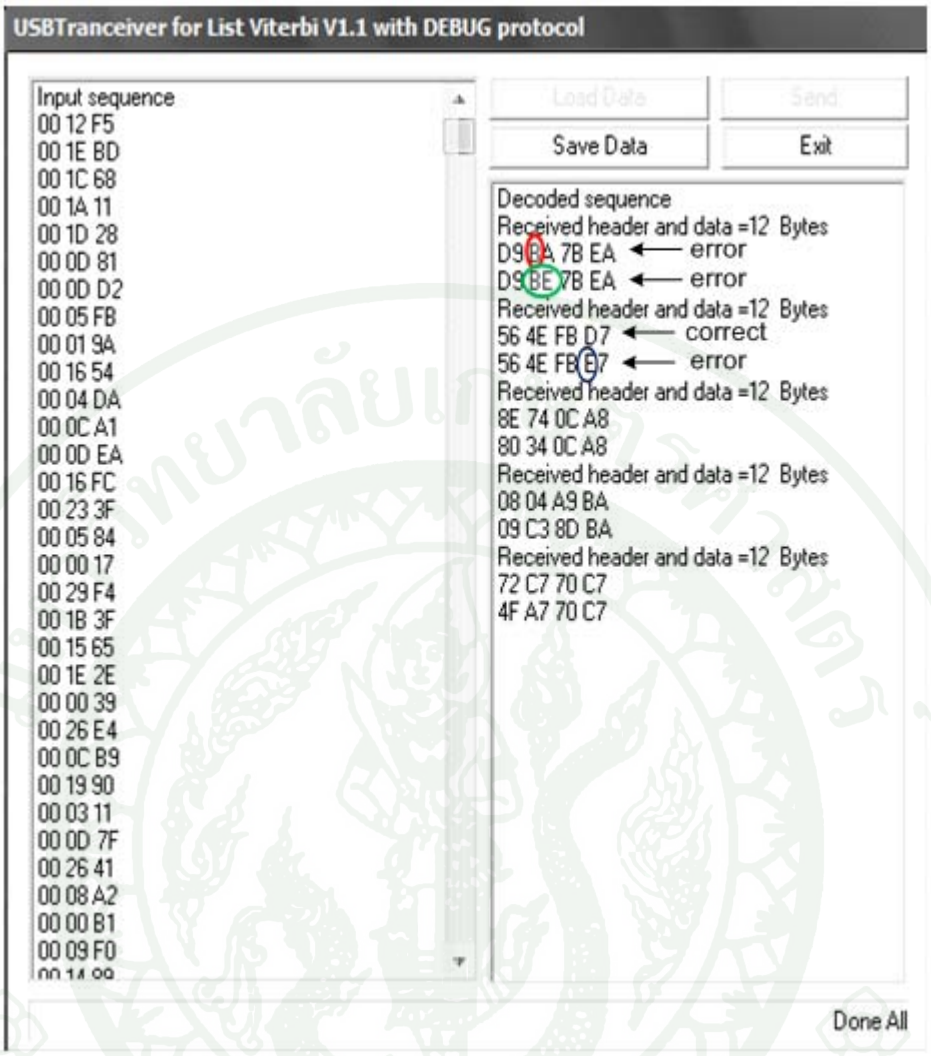
Hexadecimal	Binary 4 bits
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101

ตารางที่ 7 (ต่อ)

Hexadecimal	Binary 4 bits
E	1110
F	1111

ภาพที่ 48 แสดงผลการทำงานของตัวถอดรหัส โดยข้อมูลที่ใช้ทดสอบการถอดรหัสเป็นข้อมูลอินพุทแบบทั่วไป (general input) ซึ่งก็คือ ข้อมูลค่าใดๆที่ไม่เป็นศูนย์ทั้งหมดก่อนเข้ารหัสที่เครื่องส่งขนาด 32 บิท โดยทดสอบจำนวน 5 ครั้ง และกำหนดค่า SNR เท่ากับ 7 ซึ่งมีค่าอินพุทที่ใช้ในการทดสอบ ดังนี้

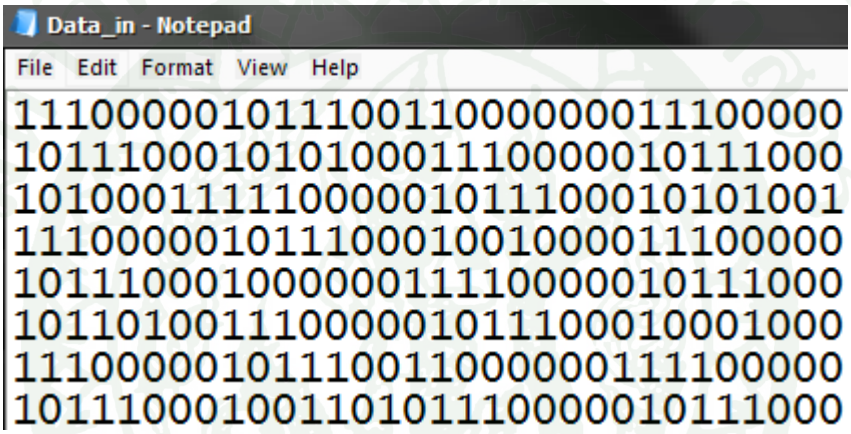
1. 11011001 11011010 01111011 11101010 แปลงเป็นเลขฐาน 16 จะได้ D9 DA 7B EA
2. 01010110 01001110 11111011 11010111 แปลงเป็นเลขฐาน 16 จะได้ 56 4E FB D7
3. 10001110 01110100 00001100 10101000 แปลงเป็นเลขฐาน 16 จะได้ 8E 74 0C A8
4. 00001000 00000100 10101001 10111010 แปลงเป็นเลขฐาน 16 จะได้ 08 04 A9 BA
5. 01110010 11000111 01110000 11000111 แปลงเป็นเลขฐาน 16 จะได้ 72 C7 70 C7



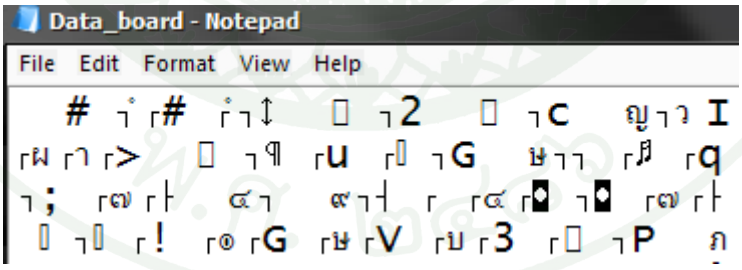
ภาพที่ 48 ผลจากการถอดรหัสบนบอร์ด FPGA ที่เป็นข้อมูลอินพุทแบบทั่วไป จำนวน 5 อินพุท

ผลลัพธ์ที่ได้จากการถอดรหัสถูกแสดงในหน้าต่าง “Decoded sequence” ดังแสดงตามภาพที่ 48 โดยผลของการถอดรหัสผลลัพธ์แรก เอาท์พุทของการถอดรหัสที่ดีที่สุดถอดรหัสผิดพลาด ค่าที่ถอดรหัสผิดพลาดคือ ค่า B (1011) ซึ่งเป็นค่าที่ถูกวงกลมไว้ ส่วนค่าที่ถูกต้องคือ D (1101) และเอาท์พุทของการถอดรหัสที่ดีที่สุดรองลงมาก็ถอดรหัสผิดพลาดเช่นกัน ค่าที่ผิดพลาดคือ BE (1011 1011) ส่วนค่าที่ถูกต้องคือ DA (1101 1010) ผลของการถอดรหัสผลลัพธ์ที่ 2, 3, 4 และ 5 นั้น เอาท์พุทของการถอดรหัสที่ดีที่สุดสามารถถอดรหัสได้ถูกต้องทั้งหมด ซึ่งสามารถเปรียบเทียบผลได้จากค่าในหน้าต่าง “Decoded sequence” กับค่าอินพุทที่ใช้ในการทดสอบ

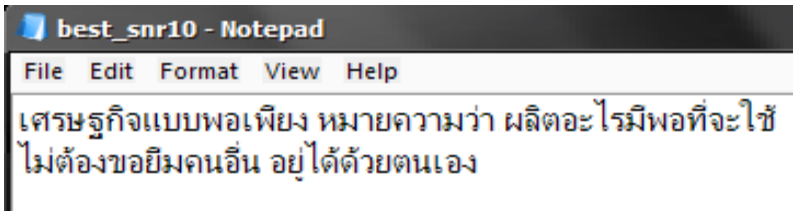
ผลการทำงานของตัวถอดรหัสบนบอร์ด FPGA โดยข้อมูลที่ใช้ทดสอบการถอดรหัสเป็นข้อความ ซึ่งก็คือ “เศรษฐกิจแบบพอเพียง หมายความว่า ผลิตอะไรมีพอที่จะใช้ ไม่ต้องขอยืมคนอื่นอยู่ได้ด้วยตนเอง” จากนั้นนำไปแปลงเป็นรหัส ASCII (American Standard Code for Information Interchange) ข้อมูลที่ได้จะเป็นเลขไบนารีแสดงดังภาพที่ 49 จากนั้นนำไปเข้ารหัสและทดสอบโดยใช้ช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ใช้การมอดูเลทแบบ BPSK และกำหนดค่า SNR เท่ากับ 10 ก่อนที่จะนำข้อมูลที่ได้ไปทดสอบการทำงานบนบอร์ด FPGA ต้องแปลงข้อมูลจากไบนารีเป็นเลขฐาน 16 ก่อน โดยใช้โปรแกรม Bin2Hex ช่วยในการแปลงข้อมูลและข้อมูลที่ได้แสดงดังภาพที่ 50



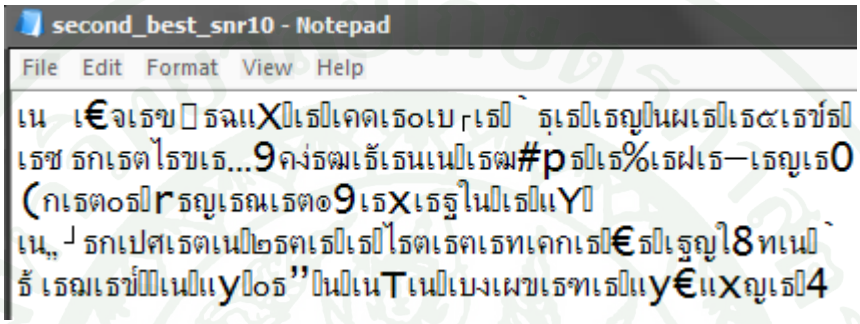
ภาพที่ 49 ตัวอย่างข้อมูลที่แปลงจากข้อความเป็นไบนารี โดยใช้รหัส ASCII



ภาพที่ 50 ตัวอย่างข้อมูลที่แปลงจากไบนารีเป็นเลขฐาน 16 โดยใช้โปรแกรม Bin2Hex



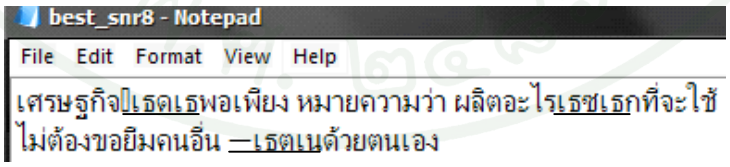
a)



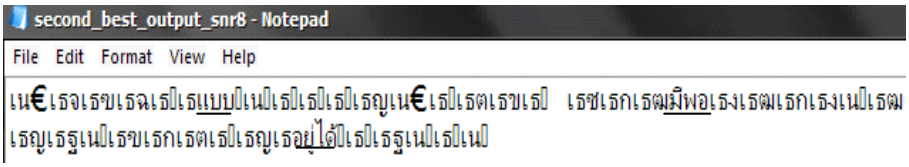
b)

ภาพที่ 51 ผลการถอดรหัสบนบอร์ด FPGA ที่ SNR = 10 โดย a) เป็นผลลัพธ์ของการถอดรหัสผลลัพธ์แรก และ b) เป็นผลลัพธ์ของการถอดรหัสผลลัพธ์ที่สอง

ผลลัพธ์ที่ได้จากการถอดรหัสแสดงดังภาพที่ 51 พบว่า ผลลัพธ์ที่ได้จากการถอดรหัสของผลลัพธ์แรก สามารถถอดรหัสได้ถูกต้องทั้งหมด เพราะค่า SNR ที่ใช้มีค่ามากพอทำให้สามารถถอดรหัสได้ถูกต้องทั้งหมด



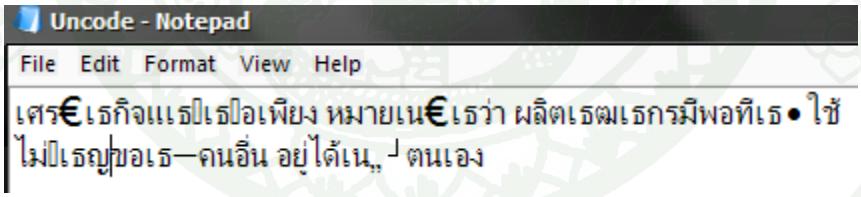
a)



b)

ภาพที่ 52 ผลการถอดรหัสบนบอร์ด FPGA ที่ SNR = 8 โดย a) เป็นผลลัพธ์ของการถอดรหัสผลลัพธ์แรก และ b) เป็นผลลัพธ์ของการถอดรหัสผลลัพธ์ที่สอง

ภาพที่ 52 แสดงผลการทำงานของตัวถอดรหัส โดยใช้ช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ใช้การมอดูเลทแบบ BPSK และกำหนดค่า SNR เท่ากับ 8 ภาพที่ 52 a) แสดงผลลัพธ์ที่ได้จากการถอดรหัสผลลัพธ์แรก โดยผลลัพธ์ที่ขีดเส้นใต้เป็นผลลัพธ์ที่ผลลัพธ์แรกถอดรหัสผิดพลาด ส่วนภาพที่ 52 b) แสดงผลลัพธ์ที่ได้จากการถอดรหัสผลลัพธ์ที่สอง โดยผลลัพธ์ที่ขีดเส้นใต้เป็นผลลัพธ์ที่ผลลัพธ์ที่สองถอดรหัสถูกต้อง ซึ่งผลลัพธ์ที่สองที่ถอดรหัสถูกต้องนั้น จะถูกนำไปแทนที่ผลลัพธ์แรกที่ถอดรหัสผิดตามภาพที่ 52 a) ในตัวถอดรหัสภายนอกแบบวิเอสดีต่อไป



ภาพที่ 53 ผลจากการถอดรหัสของข้อมูลที่ไม่ได้เข้ารหัส (uncoded)

ภาพที่ 53 แสดงผลของข้อมูลที่ไม่ได้เข้ารหัส (uncoded) โดยใช้ช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ใช้การมอดูเลทแบบ BPSK และกำหนดค่า SNR เท่ากับ 10 ซึ่งผลลัพธ์ที่ได้มีประสิทธิภาพที่เลวร้ายกว่าผลลัพธ์ที่ได้จากการเข้ารหัสมาก

3. ผลการสังเคราะห์วงจร (Synthesis) สำหรับตัวถอดรหัส

ผลการสังเคราะห์วงจร สำหรับตัวถอดรหัสบนชิป FPGA Virtex5 รุ่น XC5VLX110 มีขนาดของ CLBs เท่ากับ 8640 เซลล์ ประกอบไปด้วย Flip-flops 69120 ตัว และ 6-input LUT 69120 ชุดหรือคิดเป็น 69120 x 2^6 = 4423680 โลจิกเกท จากผลการสังเคราะห์วงจร ทำให้ทราบค่าประมาณการใช้ทรัพยากรบนชิป FPGA ดังภาพที่ 54 โดย จำนวน Flip-flop ใช้ไป 33341 ตัว จากทั้งหมด 69120 ตัว หรือคิดเป็น 48% และจำนวน LUT ใช้ไป 20265 ตัว จากทั้งหมด 69120 ตัว หรือคิดเป็น 29% เนื่องจากแต่ละเซลล์ของ CLBs นั้นประกอบไปด้วย Flip-flop และ LUT ดังนั้น เมื่อรวมทั้งสองเข้าด้วยกันเป็น CLBs แล้วจะมีการใช้งานรวมเป็น 38531 ชุด จากทั้งหมด 69120 ชุด คิดเป็น 55.75% หรือ 2465984 โลจิกเกท ประกอบด้วยการใช้งานทั้ง Flip-flop และ LUT 15075 ชุด หรือ 39% ใช้งานเฉพาะ LUT 5190 ชุด หรือ 13% ใช้งานเฉพาะ Flip-flop 18266 ชุด หรือ 47% มีการใช้งานขาสัญญาณรับส่งข้อมูลของบอร์ดทดลอง FPGA ใช้ไป 14 ขา จากทั้งหมด 440 ขาสัญญาณ คิดเป็น 3% ประกอบด้วยขาสัญญาณควบคุม 6 ขา และขาสัญญาณข้อมูล 8 ขา

Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	33,341	69,120	48%
Number used as Flip Flops	33,341		
Number of Slice LUTs	20,265	69,120	29%
Number used as logic	20,220	69,120	29%
Slice Logic Distribution			
Number of occupied Slices	10,294	17,280	59%
Number of LUT Flip Flop pairs used	38,531		
Number with an unused Flip Flop	5,190	38,531	13%
Number with an unused LUT	18,266	38,531	47%
Number of fully used LUT-FF pairs	15,075	38,531	39%
Number of unique control sets	409		
Number of slice register sites lost to control set restrictions	631	69,120	1%
IO Utilization			
Number of bonded IOBs			
Number of bonded	14	440	3%
Specific Feature Utilization			
Number of BUFG/BUFGCTRLs	2	32	6%
Number used as BUFGs	2		

ภาพที่ 54 ผลการสังเคราะห์วงจร ด้วยโปรแกรม Xilinx ISE 10.1.02 โดยนำมาแสดงเป็นบางส่วน

4. ผลลัพธ์จากการถอดรหัสจากบอร์ด FPGA เทียบกับโปรแกรม C++

สำหรับหัวข้อนี้เป็นการเปรียบเทียบผลลัพธ์ที่ได้จากการจำลองการทำงานของตัวถอดรหัสด้วยโปรแกรมภาษา C++ กับการทำงานของตัวถอดรหัสบนบอร์ด FPGA โดยข้อมูลที่ใช้ทดสอบการถอดรหัสเป็นข้อมูลอินพุทแบบทั่วไป (general input) ก่อนเข้ารหัสที่เครื่องส่งขนาด 32 บิท การทำงานของตัวถอดรหัสจะใช้อินพุทบิทเมทริกที่ได้จากช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ซึ่งใช้การมอดูเลทแบบ BFSK โดยใช้โปรแกรม MATLAB ในการจำลองช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก และจำลองดีมอดูเลเตอร์สำหรับการมอดูเลทแบบ BFSK

ตารางที่ 8 และ 9 แสดงผลลัพธ์ที่ได้จากการจำลองการทำงานของตัวถอดรหัสด้วยโปรแกรมภาษา C++ กับการทำงานของตัวถอดรหัสบนบอร์ด FPGA โดยผลลัพธ์ที่ได้จากการจำลองการทำงานของตัวถอดรหัสด้วยโปรแกรมภาษา C++ ใช้จำนวนในการทำงาน 10,000 ครั้ง และผลที่ได้จากการทำงานของตัวถอดรหัสบนบอร์ด FPGA ใช้จำนวนในการทำงาน 800 ครั้ง การที่ใช้จำนวนครั้งในการทำงานน้อยกว่าการจำลองการทำงานด้วยโปรแกรมภาษา C++ นั้น เพราะว่าบิทเมทริกที่ใช้ต้องแปลงค่าจากจำนวนจริงเป็นเลขไบนารี และแปลงจากเลขไบนารีเป็นเลขฐาน 16 ก่อนส่งค่าอินพุทจากเครื่องคอมพิวเตอร์ไปยังบอร์ด FPGA และต้องส่งค่าเอาท์พุทจากบอร์ด FPGA มายังเครื่องคอมพิวเตอร์ เมื่อได้ค่าเอาท์พุทแล้วต้องแปลงค่าเอาท์พุทที่ได้จากเลขฐาน 16 เป็นเลขไบนารีอีก ทำให้มีความล่าช้าและใช้เวลาในการทำงานมาก จึงเลือกใช้จำนวนข้อมูลที่น้อยกว่าเพื่อลดระยะเวลาในการทำงานให้น้อยลง

ตารางที่ 8 ผลจำลองการทำงานของตัวถอดรหัสด้วยโปรแกรม C++ และใช้มอดูเลทแบบ BFSK ในการทำงาน 10,000 ครั้ง

SNR	First output is wrong	Second output is wrong	P1	P2
0	9582	9259	0.9582	0.9663
1	8927	8194	0.8927	0.9178
2	7389	6387	0.7389	0.8643
3	5296	3475	0.5296	0.6561
4	3120	1191	0.3120	0.3817
5	1244	244	0.1244	0.1961

ตารางที่ 8 (ต่อ)

SNR	First output is wrong	Second output is wrong	P1	P2
6	358	45	0.0358	0.1257
7	119	11	0.0119	0.0924
8	30	2	0.0030	0.0667
9	4	0	0.0004	$< 10^{-4}$
10	0	0	$< 10^{-4}$	N/A

หมายเหตุ N/A หมายความว่า ข้อมูลไม่มีความหมายที่จะนำมาประยุกต์ใช้ได้ (not applicable)

ตารางที่ 9 ผลการทำงานของตัวถอดรหัสบนบอร์ด FPGA และใช้มอดูเลทแบบ BFSK ในการทำงาน 800 ครั้ง

SNR	First output is wrong	Second output is wrong	P1	P2
0	758	732	0.9475	0.9657
1	716	650	0.8950	0.9078
2	596	518	0.7450	0.8456
3	426	270	0.5325	0.6338
4	242	91	0.3025	0.3760
5	102	19	0.1275	0.1863
6	30	4	0.0375	0.1333
7	9	0	0.0113	$< 10^{-3}$
8	0	0	$< 10^{-3}$	N/A
9	0	0	$< 10^{-3}$	N/A
10	0	0	$< 10^{-3}$	N/A

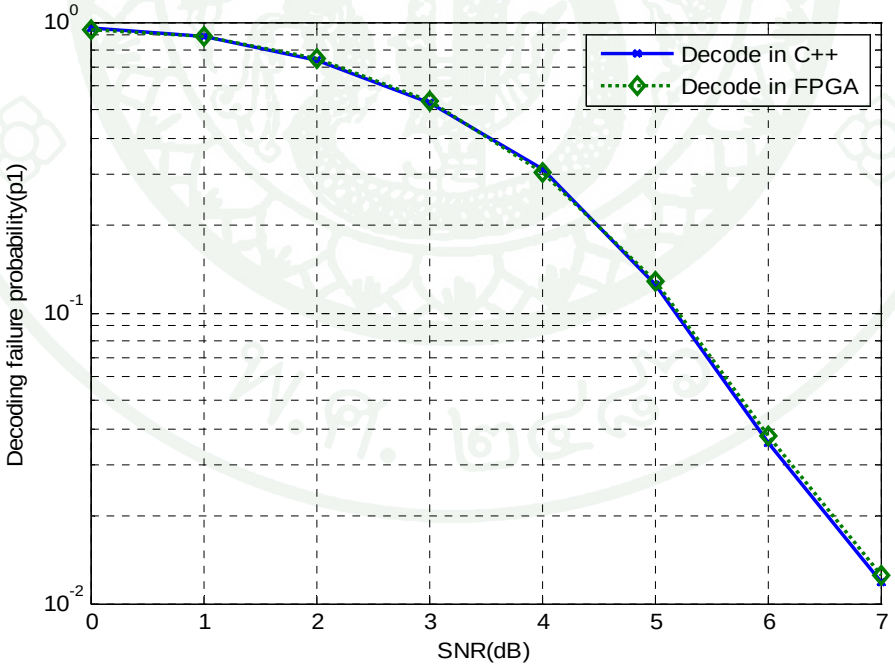
หมายเหตุ N/A หมายความว่า ข้อมูลไม่มีความหมายที่จะนำมาประยุกต์ใช้ได้ (not applicable)

โดย P1 คือ ความน่าจะเป็นที่ผลลัพธ์แรกผิด (Probability of the first output failure)

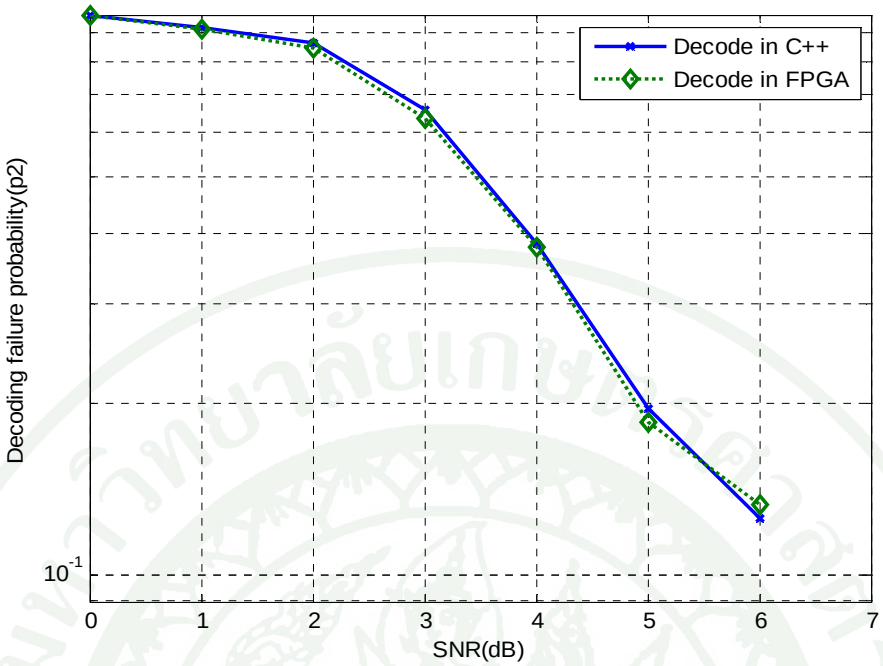
P2 คือ ความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิด (Probability of error of the second output given that the first output is wrong)

จากตารางที่ 8 และ 9 จะพิจารณาช่วงที่สนใจของ P1 ที่ช่วงระหว่าง 0.1-0.01 และพิจารณา P2 ที่มีค่าน้อยกว่า 0.5 ตารางที่ 8 เมื่อ SNR มีค่ามากกว่าหรือเท่ากับ 9 ตัวถอดรหัสสามารถถอดรหัสได้ถูกต้องทั้งหมด และตารางที่ 9 ที่ P2 เมื่อ SNR มีค่ามากกว่าหรือเท่ากับ 7 ตัวถอดรหัสสามารถถอดรหัสได้ถูกต้องทั้งหมดเช่นกัน สำหรับ SNR มีค่าระหว่าง 5 ถึง 8 ในตารางที่ 8 และ SNR มีค่าระหว่าง 5 ถึง 9 ในตารางที่ 9 ตัวถอดรหัสวีเอสดีสามารถถอดรหัสได้ถูกต้องโดยง่าย เพราะ P2 มีค่าต่ำ (0.16 ถึง 0)

จากตารางที่ 8 เมื่อ SNR มากกว่าหรือเท่ากับ 10 dB และตารางที่ 9 เมื่อ SNR มากกว่าหรือเท่ากับ 8 dB จะพบว่า P1 จะมีค่าที่ต่ำพียงพอ ซึ่งทำให้ไม่จำเป็นต้องใช้ตัวเลือกที่สองอีกต่อไป และจะสังเกตได้ว่า P2 จะไม่มีความหมาย ถ้า P1 เท่ากับ 0 เพื่อให้เข้าใจได้ง่ายขึ้นจะกำหนดให้ค่า P2 เท่ากับ N/A ซึ่งหมายความว่า ข้อมูลไม่มีความหมายที่จะนำมาประยุกต์ใช้ได้ (not applicable)



ภาพที่ 55 เปรียบเทียบผลความน่าจะเป็นที่ผลลัพธ์แรกผิดของตัวถอดรหัสโปรแกรมภาษา C++ และตัวถอดรหัสบนบอร์ด FPGA



ภาพที่ 56 เปรียบเทียบผลความน่าจะเป็นความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิดของตัวถอดรหัสโปรแกรมภาษา C++ และตัวถอดรหัสบนบอร์ด FPGA

จากภาพที่ 55 และ 56 เป็นการนำค่าที่ได้จากตารางที่ 7 และ 8 มาแสดงผลเป็นกราฟ เพื่อทำการเปรียบเทียบผลลัพธ์ที่ได้ จากภาพพบว่าค่าที่ได้จากการจำลองการทำงานของตัวถอดรหัสด้วยโปรแกรมภาษา C++ และจากการทำงานของตัวถอดรหัสบนบอร์ด FPGA มีผลลัพธ์ที่ได้ใกล้เคียงกันมาก แตกต่างกันเพียงเล็กน้อย เนื่องจากจำนวนครั้งในการทำงานไม่เท่ากันทำให้ค่าที่ได้มีความแตกต่างกัน

5. วิเคราะห์ประสิทธิภาพของตัวถอดรหัสลิสวิเทอร์บิจากช่องสัญญาณ

สำหรับการทดสอบประสิทธิภาพการทำงานของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ ข้อมูลที่ใช้ทดสอบการถอดรหัสเป็นข้อมูลอินพุทแบบทั่วไป (general input) ก่อนเข้ารหัสที่เครื่องส่งขนาด 32 บิท การทำงานของตัวถอดรหัสจะใช้อินพุทบิทเมทริกที่ได้จากช่องสัญญาณ โดยเปลี่ยนการมอดูเลทแบบ BFSK เป็น BPSK เพราะ BPSK สามารถทนต่อสัญญาณรบกวนได้ดีกว่า BFSK และใช้โปรแกรม MATLAB สร้างการจำลองช่องสัญญาณ จำลองการมอดูเลทและจำลองตัวดีมอดูเลเตอร์ สำหรับช่องสัญญาณจะใช้ช่องสัญญาณสามแบบคือ ช่องสัญญาณรบกวน

เกาส์เซียนขาวแบบบวก ช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก (Rayleigh fading with AWGN channel) และช่องสัญญาณการจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก (Ricean fading with AWGN channel) การวิเคราะห์ประสิทธิภาพของตัวถอดรหัสจะใช้ค่า SNR กับค่าความน่าจะเป็นในการถอดรหัสผิดพลาดในการวิเคราะห์ประสิทธิภาพจากทั้งสามช่องสัญญาณ

ในส่วนของช่องสัญญาณการจางหายแบบเรย์ลีและไรเชียน จากการทดลองวัดเอนเวลโลปของสัญญาณที่วัดได้หลายๆครั้งในหลายๆพื้นที่ พบว่าในพื้นที่ตัวเมืองและชานเมือง การเปลี่ยนแปลงของเอนเวลโลปของสัญญาณที่รับได้ที่เวลาใดๆจะมีการเปลี่ยนแปลงใกล้เคียงกับการกระจายแบบเรย์ลี ส่วนในพื้นที่ชนบท พบว่าการเปลี่ยนแปลงของเอนเวลโลปของสัญญาณที่รับได้ที่เวลาใดๆจะมีการเปลี่ยนแปลงใกล้เคียงกับการกระจายแบบไรเชียน (Jakes, 1974) โดยในโครงงานวิทยานิพนธ์นี้ จะใช้การสื่อสารเคลื่อนที่ในการวิเคราะห์ประสิทธิภาพของช่องสัญญาณการจางหายแบบเรย์ลีและไรเชียน ซึ่งใช้ความเร็วในการวิเคราะห์ที่แตกต่างกันคือ ที่ความเร็วที่ 60, 90 และ 120 กิโลเมตรต่อชั่วโมง ตามลำดับ สำหรับความเร็วที่ใช้จะต้องนำไปคำนวณเป็นค่า Doppler shift ก่อนนำไปใช้ในช่องสัญญาณ ซึ่งค่า Doppler shift ที่ใช้มีค่า 116.72 dB (60 กิโลเมตรต่อชั่วโมง) 175.08 dB (90 กิโลเมตรต่อชั่วโมง) และ 233.44 dB (120 กิโลเมตรต่อชั่วโมง) ตามลำดับ

5.1 ผลลัพธ์ของช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก

ผลลัพธ์ที่ได้จากตารางที่ 10 เป็นผลลัพธ์ที่ได้จากการจำลองการทำงานของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ โดยใช้ช่องสัญญาณเกาส์เซียนขาวแบบบวก และใช้ค่าพลังงานต่อบิท (E_b) เท่ากับ 1

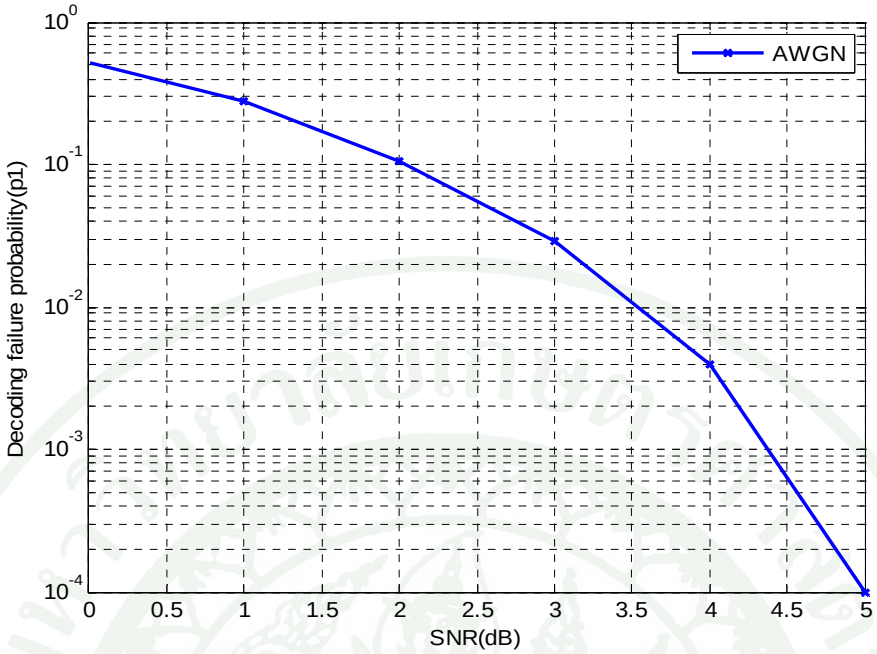
ตารางที่ 10 ผลลัพธ์ที่ได้จากช่องสัญญาณแบบ AWGN และใช้มอดูเลทแบบ BPSK ในการทำงาน 10,000 ครั้ง

SNR	First output is wrong	Second output is wrong	P1	P2
0	5234	4136	0.5234	0.7902
1	2758	1795	0.2758	0.6508
2	1052	549	0.1052	0.5218
3	293	102	0.0293	0.3481
4	40	9	0.0040	0.2250
5	10	1	0.0010	0.1000
6	0	0	$< 10^{-4}$	N/A
7	0	0	$< 10^{-4}$	N/A
8	0	0	$< 10^{-4}$	N/A
9	0	0	$< 10^{-4}$	N/A
10	0	0	$< 10^{-4}$	N/A

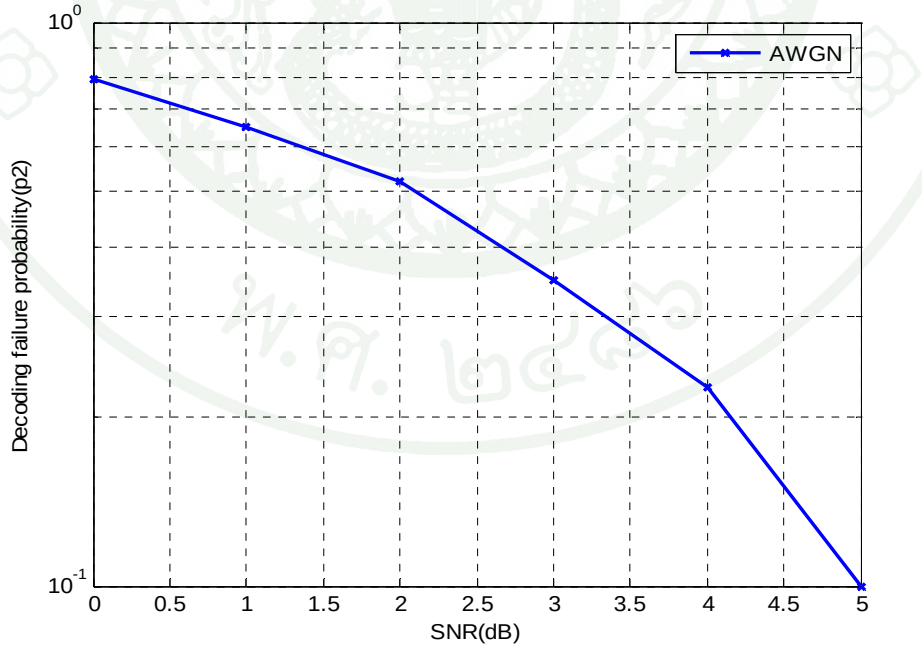
หมายเหตุ N/A หมายความว่า ข้อมูลไม่มีความหมายที่จะนำมาประยุกต์ใช้ได้ (not applicable)

โดย P1 คือ ความน่าจะเป็นที่ผลลัพธ์แรกผิด (Probability of the first output failure)

P2 คือ ความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิด (Probability of error of the second output given that the first output is wrong)



ภาพที่ 57 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์แรกผิดของช่องสัญญาณแบบ AWGN ในการทำงาน 10,000 ครั้ง



ภาพที่ 58 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิดของช่องสัญญาณแบบ AWGN ในการทำงาน 10,000 ครั้ง

จากภาพที่ 57 จะพบว่า เมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นที่ผลลัพธ์แรกผิด (P1) จะมีค่าลดลง และเมื่อเพิ่มจนกระทั่ง SNR มีค่าเท่ากับ 6 P1 จะมีค่าเท่ากับศูนย์ ส่วนภาพที่ 58 ก็เช่นเดียวกัน เมื่อ SNR มีค่าเพิ่มขึ้นความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิด (P2) จะมีค่าลดลง และเมื่อเพิ่มจนกระทั่ง SNR มีค่าเท่ากับ 6 P2 จะมีค่าเท่ากับ N/A เพราะ P2 มาจากผลลัพธ์ที่สองถอดรหัสผิดพลาดหารด้วยผลลัพธ์ที่หนึ่งถอดรหัสผิดพลาด ดังนั้น ถ้าผลลัพธ์ที่หนึ่งถอดรหัสได้ถูกต้องทั้งหมด ก็แสดงว่า ผลลัพธ์ที่หนึ่งถอดรหัสผิดพลาดเท่ากับศูนย์ P2 จึงไม่สามารถหาค่าได้

5.2 ผลของช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก

ผลลัพธ์ที่ได้จากตารางที่ 11, 12 และ 13 เป็นผลลัพธ์ที่ได้จากการจำลองการทำงานของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ โดยใช้ช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ที่มีความเร็วแตกต่างกันคือ ที่ความเร็วที่ 60, 90 และ 120 กิโลเมตรต่อชั่วโมง ตามลำดับ และให้ค่าพลังงานต่อบิท (E_b) เป็นค่าคงที่เท่ากับ 5.6 การเพิ่มค่า E_b จะทำให้ประสิทธิภาพของตัวถอดรหัสสูงขึ้นและทำให้การวิเคราะห์ประสิทธิภาพง่ายขึ้นด้วย สำหรับที่ SNR = 10 จะใช้จำนวนในการทำงาน 50,000 ครั้ง แทนการทำงาน 10,000 ครั้ง เนื่องจากการทำงาน 10,000 ครั้ง เป็นการทำงานที่น้อยเกินไปสำหรับ SNR ในช่วงดังกล่าว ทำให้ผลลัพธ์ที่ได้ยังไม่เพียงพอที่จะนำไปวิเคราะห์ประสิทธิภาพได้

ตารางที่ 11 ผลของช่องสัญญาณการจางหายแบบเรย์ลี ที่ 60 กิโลเมตรต่อชั่วโมง ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	SNR	First output is wrong	Second output is wrong	P1	P2
60	116.72	0	9972	9937	0.9972	0.9964
60	116.72	2	9661	9489	0.9661	0.9821
60	116.72	4	8016	7332	0.8016	0.9146
60	116.72	6	3717	2643	0.3717	0.7110
60	116.72	8	545	217	0.0545	0.3981
60	116.72	10	91*	14*	0.0018	0.1538

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

ตารางที่ 12 ผลของช่องสัญญาณการจางหายแบบเรย์ลี ที่ 90 กิโลเมตรต่อชั่วโมง ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	SNR	First output is wrong	Second output is wrong	P1	P2
90	175.08	0	9982	9962	0.9982	0.9979
90	175.08	2	9761	9648	0.9761	0.9884
90	175.08	4	8464	7906	0.8464	0.9337
90	175.08	6	4533	3399	0.4533	0.7498
90	175.08	8	808	343	0.0808	0.4245
90	175.08	10	172*	29*	0.0034	0.1686

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

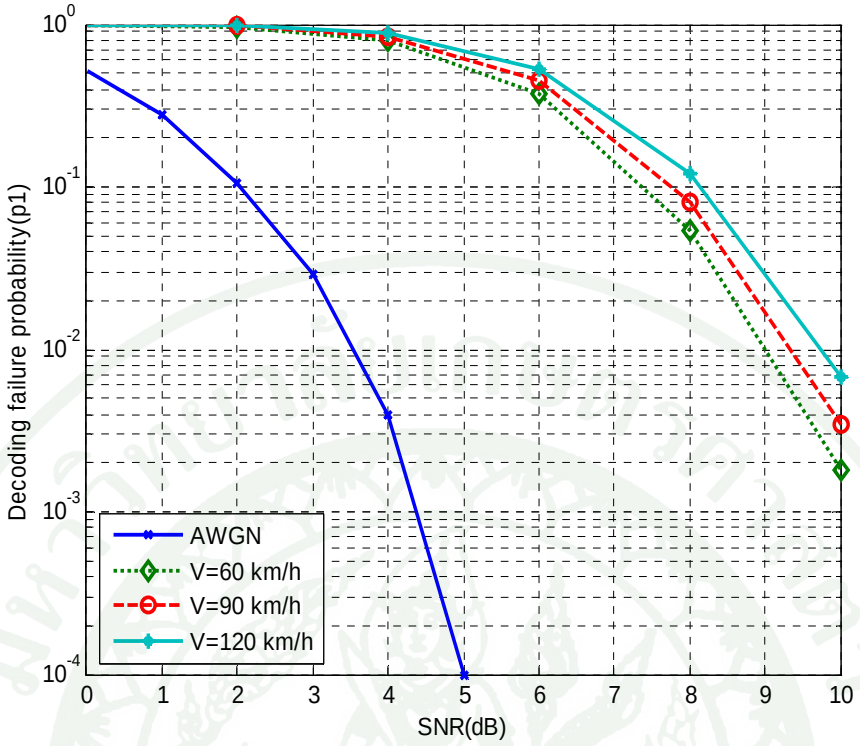
ตารางที่ 13 ผลของช่องสัญญาณการจางหายแบบเรย์ลี ที่ 120 กิโลเมตรต่อชั่วโมง ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	SNR	First output is wrong	Second output is wrong	P1	P2
120	233.44	0	9983	9976	0.9983	0.9992
120	233.44	2	9837	9755	0.9837	0.9916
120	233.44	4	8840	8396	0.8840	0.9497
120	233.44	6	5282	4223	0.5282	0.7995
120	233.44	8	1219	588	0.1215	0.4824
120	233.44	10	334*	69*	0.0067	0.2065

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

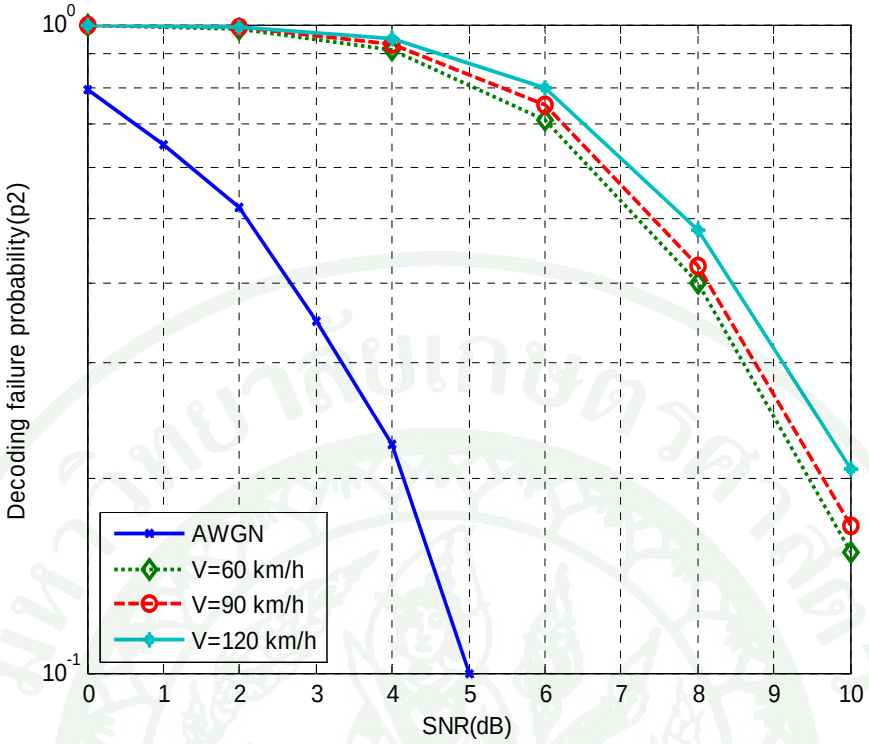
โดย P1 คือ ความน่าจะเป็นที่ผลลัพธ์แรกผิด (Probability of the first output failure)

P2 คือ ความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิด (Probability of error of the second output given that the first output is wrong)



ภาพที่ 59 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบเรย์ลี ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง ในการทำงาน 10,000 ครั้ง

จากภาพที่ 59 จะพบว่า เมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นที่ผลลัพธ์แรกผิด (P1) จะมีค่าลดลง และเมื่อเพิ่มความเร็วประสิทธิภาพในการถอดรหัสจะลดลงตามความเร็วที่เพิ่มขึ้น ดังนั้น ที่ความเร็ว 60 กิโลเมตรต่อชั่วโมง จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าความเร็ว 90 และ 120 กิโลเมตรต่อชั่วโมง ที่ความเร็ว 90 กิโลเมตรต่อชั่วโมง จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าความเร็ว 120 กิโลเมตรต่อชั่วโมง และช่องสัญญาณเกาส์เซียนขาวแบบบวก จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก



ภาพที่ 60 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบเรย์ลี ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง ในการทำงาน 10,000 ครั้ง

ส่วนภาพที่ 60 ก็เช่นเดียวกัน เมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นของการถอดรหัสที่ดีที่สุดรองลงผิดพลาด (P2) จะมีค่าลดลง และเมื่อเพิ่มความเร็วประสิทธิภาพในการถอดรหัสจะลดลงตามความเร็วที่เพิ่มขึ้น ดังนั้น ที่ความเร็ว 60 กิโลเมตรต่อชั่วโมง จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าความเร็ว 90 และ 120 กิโลเมตรต่อชั่วโมง ที่ความเร็ว 90 กิโลเมตรต่อชั่วโมง จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าความเร็ว 120 กิโลเมตรต่อชั่วโมง และช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวกจะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก

5.3 ผลของช่องสัญญาณการจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก

ผลลัพธ์ที่ได้จากตารางที่ 14, 15, 16, 17, 18, 19, 20, 21 และ 22 เป็นผลลัพธ์ที่ได้จากการจำลองการทำงานของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ โดยใช้ช่องสัญญาณการจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ที่ความเร็วแตกต่างกันคือ ความเร็วที่ 60, 90 และ 120 กิโลเมตรต่อชั่วโมง ตามลำดับ และค่าไรเชียนแฟคเตอร์ K ที่ใช้ในการวิเคราะห์ประสิทธิภาพในที่นี้จะใช้ 3 ค่าคือ K = 1, K = 3 และ K = 10 และให้ค่าพลังงานต่อบิท (E_b) เป็นค่าคงที่เท่ากับ 5.6 การเพิ่มค่า E_b จะทำให้ประสิทธิภาพของตัวถอดรหัสสูงขึ้นและทำให้การวิเคราะห์ประสิทธิภาพง่ายขึ้นด้วย ส่วนที่ SNR = 10 จะใช้จำนวนในการทำงาน 50,000 ครั้ง แทนการทำงาน 10,000 ครั้ง เนื่องจากการทำงาน 10,000 ครั้ง เป็นการทำงานที่น้อยเกินไปสำหรับ SNR ในช่วงดังกล่าว ทำให้ผลลัพธ์ที่ได้ยังไม่เพียงพอที่จะนำไปวิเคราะห์ประสิทธิภาพได้

สำหรับค่าไรเชียนแฟคเตอร์ K นั้น จะเป็นตัวชี้วัดสถานะของช่องสัญญาณ ถ้าหากค่าไรเชียนแฟคเตอร์ K มีค่ามากขึ้นค่าพลังงานใน LOS จะมีค่าเพิ่มขึ้นด้วย และเมื่อค่าไรเชียนแฟคเตอร์ K มีค่าสูงมากๆ ส่วนประกอบ LOS จะมีค่าการจางหายน้อยมากๆทำให้ประสิทธิภาพในการส่งสัญญาณมีค่าสูงขึ้น และช่องสัญญาณก็จะกลายเป็นสัญญาณรบกวนเกาส์เซียนขาวแบบบวก

ตารางที่ 14 ผลของช่องสัญญาณการจางหายแบบไรเชียน ที่ 60 กิโลเมตรต่อชั่วโมง และค่า K = 1 ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	K	SNR	First output is wrong	Second output is wrong	P1	P2
60	116.72	1	0	9968	9937	0.9968	0.9968
60	116.72	1	2	9629	9437	0.9629	0.9800
60	116.72	1	4	7912	7164	0.7912	0.9054
60	116.72	1	6	3528	2490	0.3528	0.7057
60	116.72	1	8	481	191	0.0481	0.3971
60	116.72	1	10	65*	10*	0.0013	0.1538

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

ตารางที่ 15 ผลของช่องสัญญาณการจางหายแบบไรเชียน ที่ 60 กิโลเมตรต่อชั่วโมง และค่า K = 3 ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	K	SNR	First output is wrong	Second output is wrong	P1	P2
60	116.72	3	0	9960	9915	0.9960	0.9954
60	116.72	3	2	9525	9318	0.9525	0.9782
60	116.72	3	4	7557	6775	0.7557	0.8965
60	116.72	3	6	3026	2064	0.3026	0.6821
60	116.72	3	8	355	136	0.0355	0.3831
60	116.72	3	10	40*	4*	0.0008	0.1000

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

ตารางที่ 16 ผลของช่องสัญญาณการจางหายแบบไรเชียน ที่ 60 กิโลเมตรต่อชั่วโมง และค่า K = 10 ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	K	SNR	First output is wrong	Second output is wrong	P1	P2
60	116.72	10	0	9902	9825	0.9902	0.9922
60	116.72	10	2	9159	8816	0.9159	0.9625
60	116.72	10	4	6289	5289	0.6289	0.8409
60	116.72	10	6	1796	1046	0.1796	0.5824
60	116.72	10	8	137	37	0.0137	0.2700
60	116.72	10	10	8*	0*	0.0002	$< 10^{-4}$

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

ตารางที่ 17 ผลของช่องสัญญาณการจางหายแบบไรเชียน ที่ 90 กิโลเมตรต่อชั่วโมง และค่า K = 1 ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	K	SNR	First output is wrong	Second output is wrong	P1	P2
90	175.08	1	0	9980	9958	0.9980	0.9978
90	175.08	1	2	9734	9621	0.9734	0.9884
90	175.08	1	4	8345	7681	0.8345	0.9204
90	175.08	1	6	4277	3210	0.4277	0.7505
90	175.08	1	8	750	357	0.0750	0.4760
90	175.08	1	10	136*	21*	0.0027	0.1544

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

ตารางที่ 18 ผลของช่องสัญญาณการจางหายแบบไรเชียน ที่ 90 กิโลเมตรต่อชั่วโมง และค่า K = 3 ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	K	SNR	First output is wrong	Second output is wrong	P1	P2
90	175.08	3	0	9963	9951	0.9963	0.9988
90	175.08	3	2	9694	9545	0.9694	0.9846
90	175.08	3	4	8115	7493	0.8115	0.9233
90	175.08	3	6	3921	2854	0.3921	0.7279
90	175.08	3	8	605	256	0.0605	0.4231
90	175.08	3	10	97*	12*	0.0019	0.1237

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

ตารางที่ 19 ผลของช่องสัญญาณการจางหายแบบไรเชียน ที่ 90 กิโลเมตรต่อชั่วโมง และค่า K = 10 ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	K	SNR	First output is wrong	Second output is wrong	P1	P2
90	175.08	10	0	9930	9891	0.9930	0.9961
90	175.08	10	2	9441	9215	0.9441	0.9761
90	175.08	10	4	7160	6246	0.7160	0.8723
90	175.08	10	6	2665	1684	0.2665	0.6319
90	175.08	10	8	272	92	0.0272	0.3382
90	175.08	10	10	19*	1*	0.0004	0.0526

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

ตารางที่ 20 ผลของช่องสัญญาณการจางหายแบบไรเชียน ที่ 120 กิโลเมตรต่อชั่วโมง และค่า K = 1 ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	K	SNR	First output is wrong	Second output is wrong	P1	P2
120	233.44	1	0	9979	9972	0.9979	0.9993
120	233.44	1	2	9821	9725	0.9821	0.9902
120	233.44	1	4	8783	8303	0.8783	0.9453
120	233.44	1	6	5093	3983	0.5093	0.7824
120	233.44	1	8	1107	567	0.1107	0.5122
120	233.44	1	10	209*	53*	0.0058	0.1828

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

ตารางที่ 21 ผลของช่องสัญญาณการจางหายแบบไรเชียน ที่ 120 กิโลเมตรต่อชั่วโมง และค่า K = 3 ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	K	SNR	First output is wrong	Second output is wrong	P1	P2
120	233.44	3	0	9983	9968	0.9983	0.9985
120	233.44	3	2	9804	9686	0.9804	0.9879
120	233.44	3	4	8555	7977	0.8555	0.9324
120	233.44	3	6	4839	3712	0.4839	0.7671
120	233.44	3	8	939	420	0.0939	0.4473
120	233.44	3	10	216*	37*	0.0043	0.1713

หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

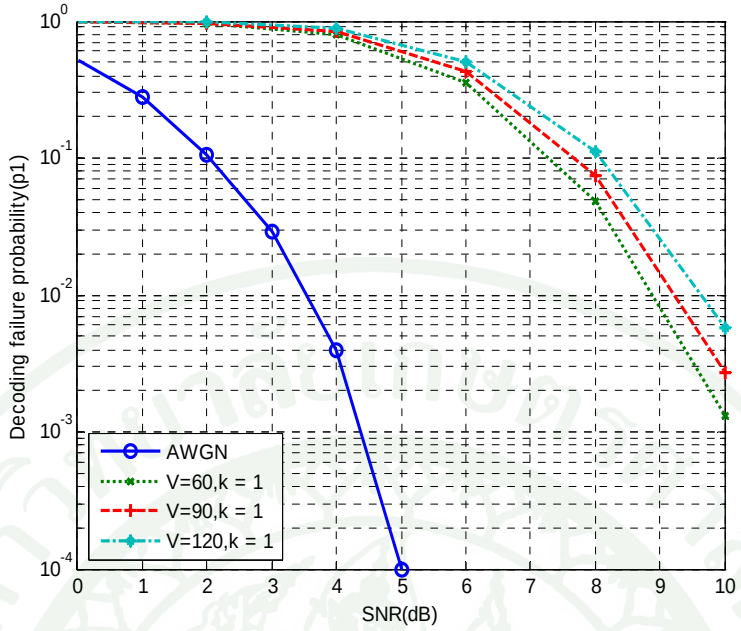
ตารางที่ 22 ผลของช่องสัญญาณการจางหายแบบไรเชียน ที่ 120 กิโลเมตรต่อชั่วโมง และค่า K = 10 ในการทำงาน 10,000 ครั้ง

V(km/h)	Doppler	K	SNR	First output is wrong	Second output is wrong	P1	P2
120	233.44	10	0	9961	9929	0.9961	0.9968
120	233.44	10	2	9586	9385	0.9586	0.9790
120	233.44	10	4	7691	6929	0.7697	0.9002
120	233.44	10	6	3239	2241	0.3239	0.6918
120	233.44	10	8	412	141	0.0412	0.3422
120	233.44	10	10	58*	10*	0.0012	0.1724

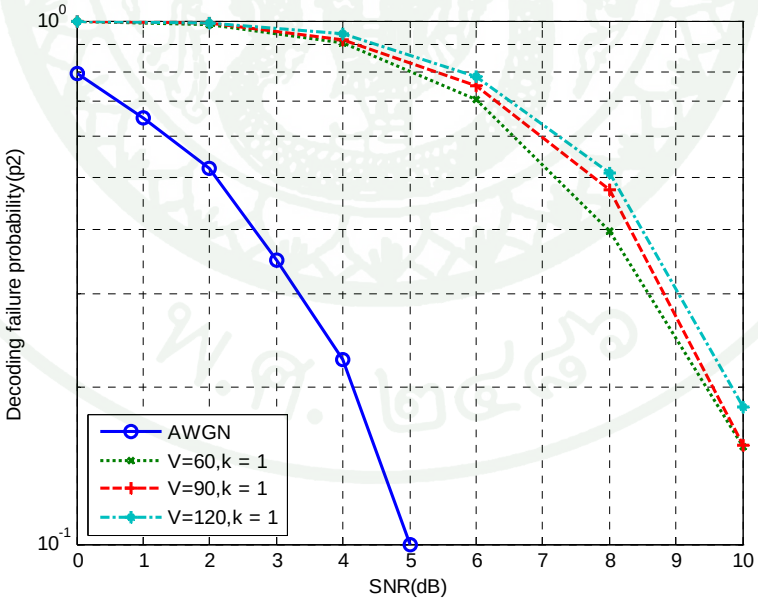
หมายเหตุ * ใช้จำนวนในการทำงาน 50,000 ครั้ง

โดย P1 คือ ความน่าจะเป็นที่ผลลัพธ์แรกผิด (Probability of the first output failure)

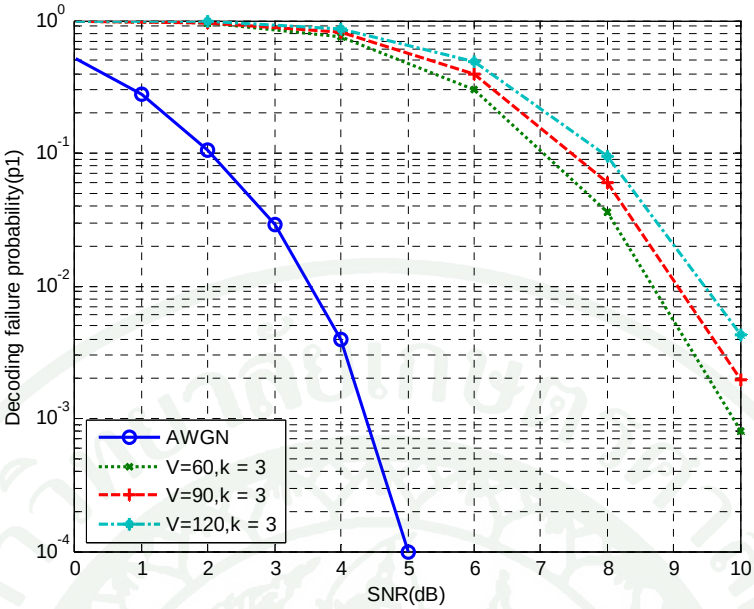
P2 คือ ความน่าจะเป็นของความผิดพลาดของตัวเลือกที่สอง เมื่อตัวเลือกที่หนึ่งผิด (Probability of error of the second output given that the first output is wrong)



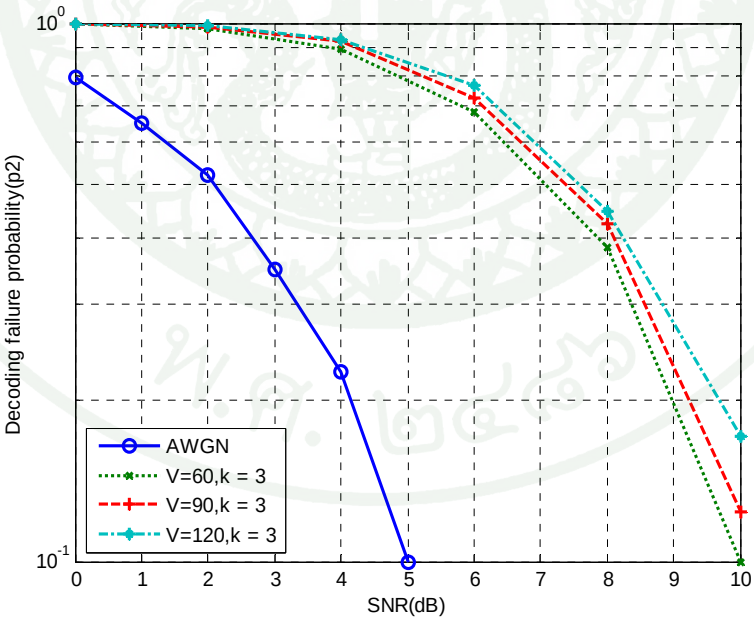
ภาพที่ 61 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบไรเชียน ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง และ K = 1 ในการทำงาน 10,000 ครั้ง



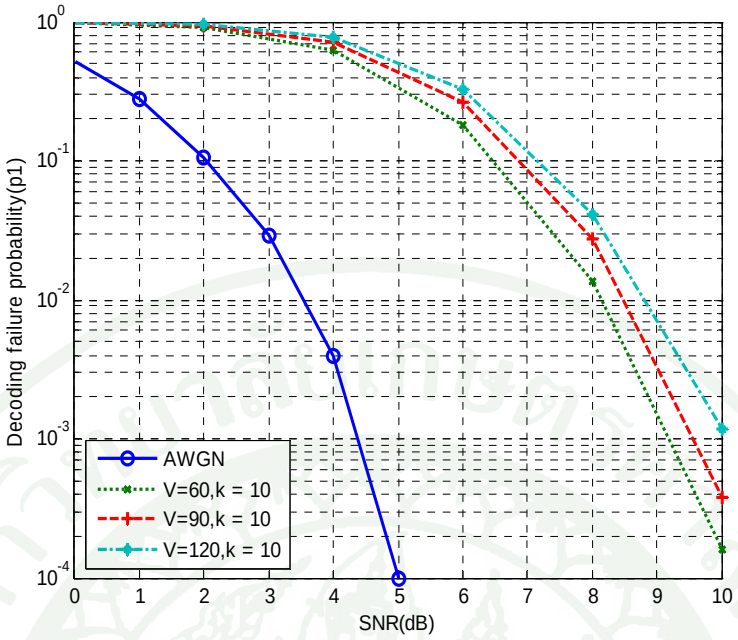
ภาพที่ 62 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบไรเชียน ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง และ K = 1 ในการทำงาน 10,000 ครั้ง



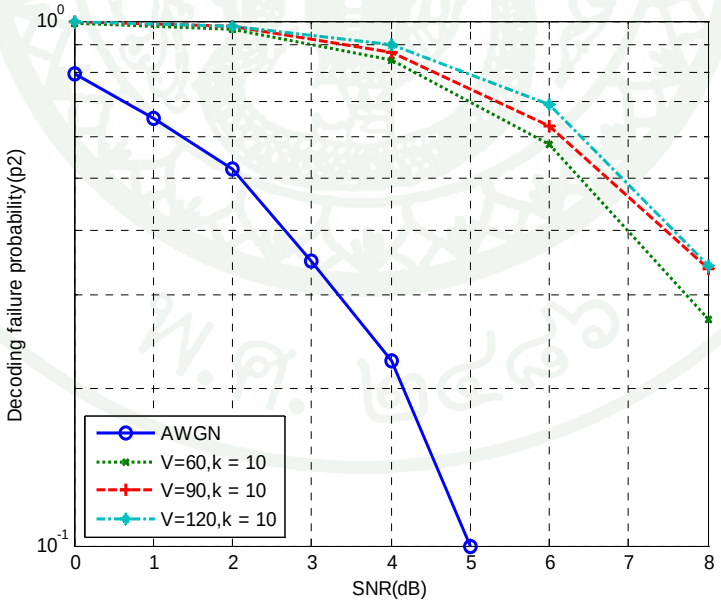
ภาพที่ 63 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบไรเชียน ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง และ K = 3 ในการทำงาน 10,000 ครั้ง



ภาพที่ 64 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบไรเชียน ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง และ K = 3 ในการทำงาน 10,000 ครั้ง



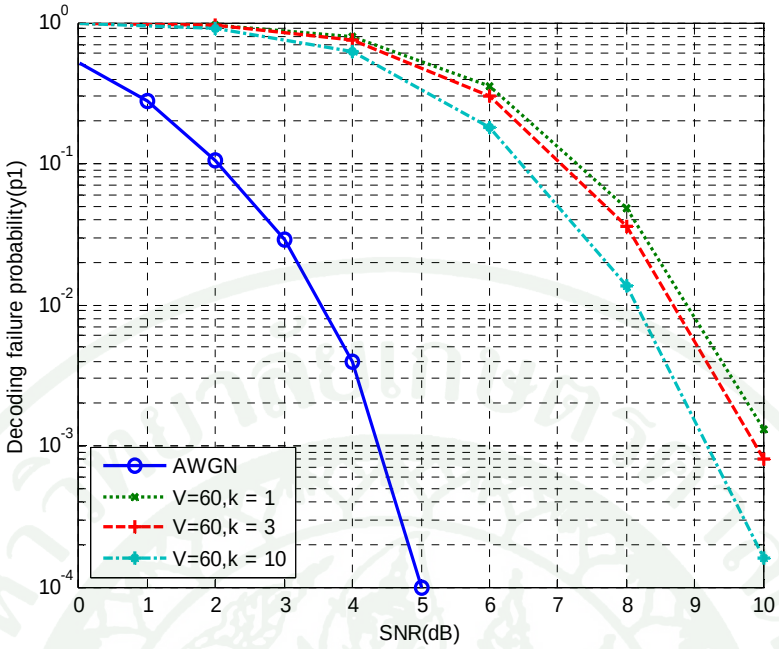
ภาพที่ 65 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบไรเชียน ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง และ K = 10 ในการทำงาน 10,000 ครั้ง



ภาพที่ 66 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบไรเชียน ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง และ K = 10 ในการทำงาน 10,000 ครั้ง

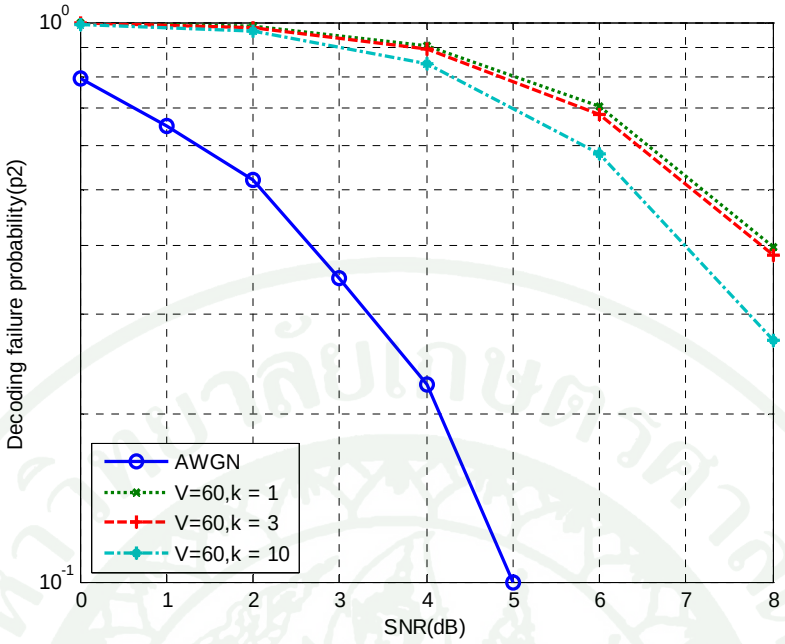
จากภาพที่ 61 เป็นการเปรียบเทียบประสิทธิภาพของการถอดรหัสที่ความเร็วแตกต่างกันและค่าไรเชียนแฟคเตอร์ K ที่ใช้มีค่าเท่ากันคือ K = 1 จากภาพจะพบว่า เมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นที่ผลลัพธ์แรกผิด (P1) จะมีค่าลดลง และเมื่อเพิ่มความเร็วประสิทธิภาพในการถอดรหัสจะลดลงตามความเร็วที่เพิ่มขึ้น ดังนั้น ที่ความเร็ว 60 กิโลเมตรต่อชั่วโมง จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าความเร็ว 90 และ 120 กิโลเมตรต่อชั่วโมง ที่ความเร็ว 90 กิโลเมตรต่อชั่วโมง จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าความเร็ว 120 กิโลเมตรต่อชั่วโมง ช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณการจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ส่วนภาพที่ 63 และ 65 ผลลัพธ์ที่ได้จะเหมือนกับภาพที่ 61 แตกต่างกันที่ค่าไรเชียนแฟคเตอร์ K ที่ใช้ โดยภาพที่ 63 ใช้ค่า K = 3 และภาพที่ 65 ใช้ค่า K = 10

ส่วนภาพที่ 62 ก็เช่นเดียวกัน เป็นการเปรียบเทียบประสิทธิภาพของการถอดรหัสที่ความเร็วแตกต่างกันและค่าไรเชียนแฟคเตอร์ K ที่ใช้มีค่าเท่ากันคือ K = 1 จากภาพจะพบว่า เมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิด (P2) จะมีค่าลดลง และเมื่อเพิ่มความเร็วประสิทธิภาพในการถอดรหัสจะลดลงตามความเร็วที่เพิ่มขึ้น ดังนั้น ที่ความเร็ว 60 กิโลเมตรต่อชั่วโมง จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าความเร็ว 90 และ 120 กิโลเมตรต่อชั่วโมง และที่ความเร็ว 90 กิโลเมตรต่อชั่วโมง จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าความเร็ว 120 กิโลเมตรต่อชั่วโมง ช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ส่วนภาพที่ 64 และ 66 ผลลัพธ์ที่ได้จะเหมือนกับภาพที่ 58 แตกต่างกันที่ค่าไรเชียนแฟคเตอร์ K ที่ใช้ โดยภาพที่ 64 ใช้ค่า K = 3 และภาพที่ 66 ใช้ค่า K = 10



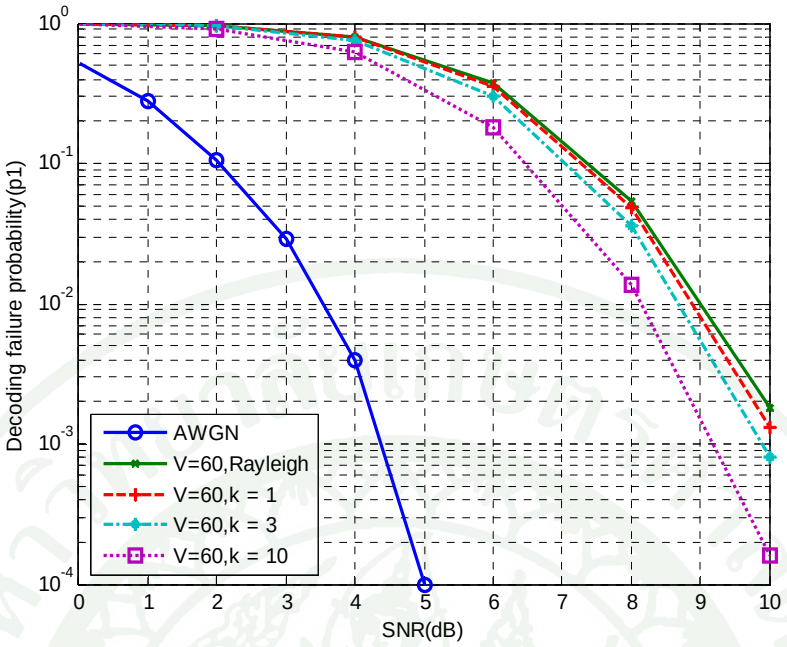
ภาพที่ 67 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบไรเชียน ที่ความเร็ว 60 กิโลเมตรต่อชั่วโมง และ K = 1, 3 และ 10 ในการทำงาน 10,000 ครั้ง

ภาพที่ 67 เป็นการเปรียบเทียบประสิทธิภาพของการถอดรหัสที่ความเร็ว 60 กิโลเมตรต่อชั่วโมงและค่าไรเชียนแฟคเตอร์ K ที่ใช้มีค่าเท่ากับ 1, 3 และ 10 ตามลำดับ จากภาพจะพบว่าเมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นที่ผลลัพธ์แรกผิด (P1) จะมีค่าลดลง และเมื่อเพิ่มค่า K ประสิทธิภาพในการถอดรหัสจะเพิ่มขึ้นตามค่า K ที่เพิ่มขึ้น ดังนั้น ที่ K = 10 จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่า K = 1 และ 3 และที่ K = 3 จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่า K = 1 และช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก โดยในที่นี้จะขอเปรียบเทียบประสิทธิภาพที่ความเร็ว 60 กิโลเมตรต่อชั่วโมงเท่านั้น เนื่องจากที่ความเร็ว 90 และ 120 กิโลเมตรต่อชั่วโมง ผลลัพธ์ที่ได้มีลักษณะเหมือนกัน



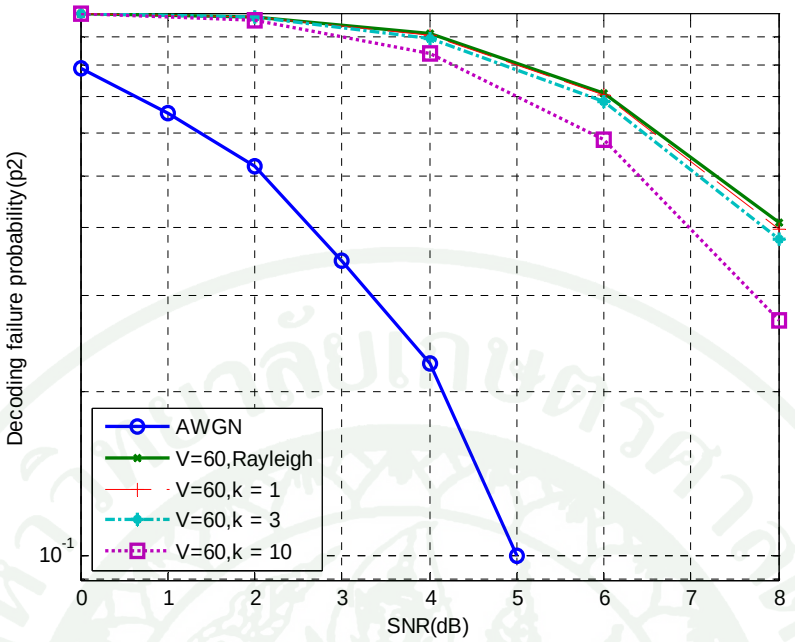
ภาพที่ 68 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบไรเชียน ที่ความเร็ว 60 กิโลเมตรต่อชั่วโมง และ K = 1, 3 และ 10 ในการทำงาน 10,000 ครั้ง

ภาพที่ 68 เป็นการเปรียบเทียบประสิทธิภาพของการถอดรหัสที่ความเร็ว 60 กิโลเมตรต่อชั่วโมงและค่าไรเชียนแฟคเตอร์ K ที่ใช้มีค่าเท่ากับ 1, 3 และ 10 ตามลำดับ จากภาพจะพบว่าเมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิด (P2) จะมีค่าลดลง และเมื่อเพิ่มค่า K ประสิทธิภาพในการถอดรหัสจะเพิ่มขึ้นตามค่า K ที่เพิ่มขึ้น ดังนั้น ที่ K = 10 จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่า K = 1 และ 3 และที่ K = 3 จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่า K = 1 และช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก โดยในที่นี้จะขอเปรียบเทียบประสิทธิภาพที่ความเร็ว 60 กิโลเมตรต่อชั่วโมงเท่านั้น เนื่องจากที่ความเร็ว 90 และ 120 กิโลเมตรต่อชั่วโมง ผลลัพธ์ที่ได้มีลักษณะเหมือนกัน



ภาพที่ 69 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบเรย์ลีเทียบกับไรเชียน ที่ความเร็ว 60 กิโลเมตรต่อชั่วโมง และ K = 1, 3 และ 10 ในการทำงาน 10,000 ครั้ง

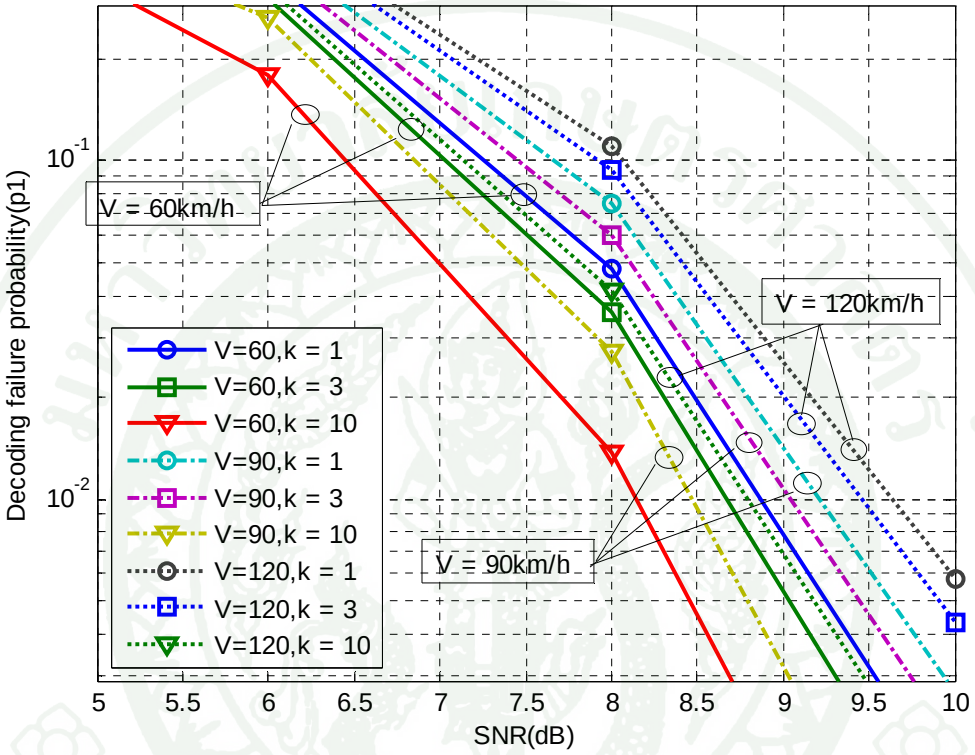
ภาพที่ 69 เป็นการเปรียบเทียบประสิทธิภาพของการถอดรหัสของช่องสัญญาณการจางหายแบบเรย์ลีเทียบกับไรเชียนที่ความเร็ว 60 กิโลเมตรต่อชั่วโมงและค่าไรเชียนแฟคเตอร์ K ที่ใช้มีค่าเท่ากับ 1, 3 และ 10 ตามลำดับ จากภาพจะพบว่า เมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นที่ผลลัพธ์แรกผิด (P1) จะมีค่าลดลง และเมื่อเพิ่มค่า K ประสิทธิภาพในการถอดรหัสจะเพิ่มขึ้นตามค่า K ที่เพิ่มขึ้น ดังนั้น ที่ K = 10 จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่า K = 1 และ 3 และที่ K = 3 จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่า K = 1 ช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณจางหายแบบเรย์ลีและไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก และช่องสัญญาณจางหายแบบไรเชียนมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณจางหายแบบเรย์ลี โดยในที่นี้จะขอเปรียบเทียบประสิทธิภาพที่ความเร็ว 60 กิโลเมตรต่อชั่วโมงเท่านั้น เนื่องจากที่ความเร็ว 90 และ 120 กิโลเมตรต่อชั่วโมง ผลลัพธ์ที่ได้มีลักษณะเหมือนกัน



ภาพที่ 70 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบเรย์ลีเทียบกับไรเชียน ที่ความเร็ว 60 กิโลเมตรต่อชั่วโมง และ K = 1, 3 และ 10 ในการทำงาน 10,000 ครั้ง

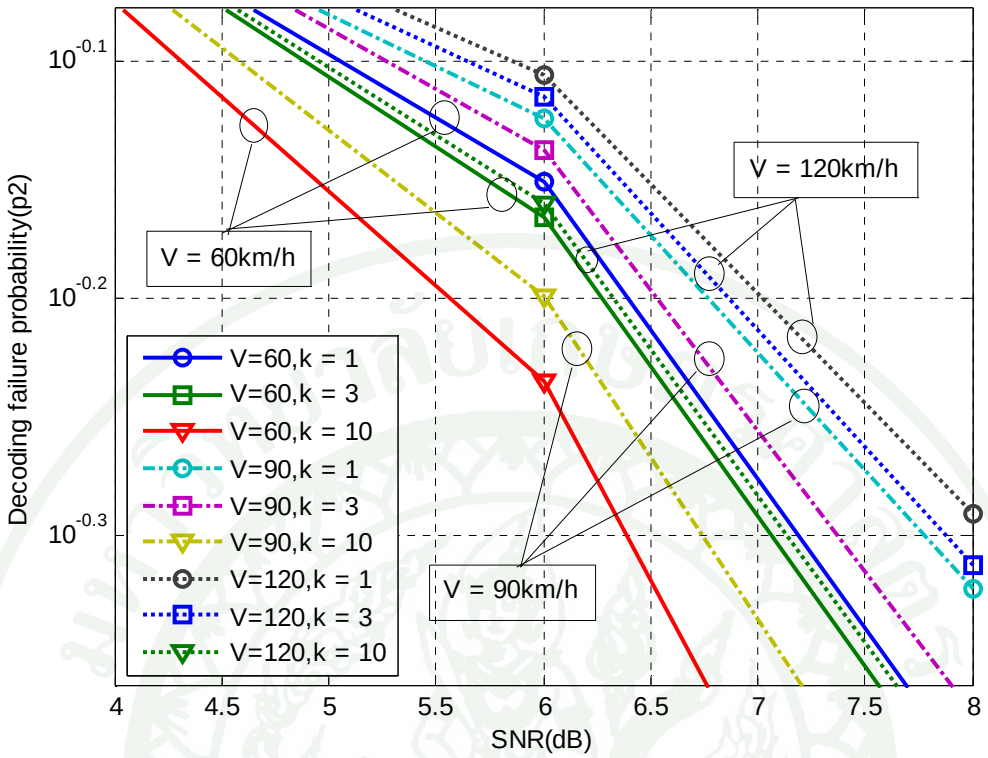
ภาพที่ 70 เป็นการเปรียบเทียบประสิทธิภาพของการถอดรหัสของช่องสัญญาณการจางหายแบบเรย์ลีเทียบกับไรเชียนที่ความเร็ว 60 กิโลเมตรต่อชั่วโมงและค่าไรเชียนแฟคเตอร์ K ที่ใช้มีค่าเท่ากับ 1, 3 และ 10 ตามลำดับ จากภาพจะพบว่า เมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นที่ผลลัพธ์ที่สองผิด เมื่อผลลัพธ์แรกผิด (P2) จะมีค่าลดลง และเมื่อเพิ่มค่า K ประสิทธิภาพในการถอดรหัสจะเพิ่มขึ้นตามค่า K ที่เพิ่มขึ้น ดังนั้น ที่ K = 10 จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่า K = 1 และ 3 และที่ K = 3 จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่า K = 1 ช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก จะมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณจางหายแบบเรย์ลีและไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก และช่องสัญญาณจางหายแบบไรเชียนมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณจางหายแบบเรย์ลี โดยในที่นี้จะขอเปรียบเทียบประสิทธิภาพที่ความเร็ว 60 กิโลเมตรต่อชั่วโมงเท่านั้น เนื่องจากที่ความเร็ว 90 และ 120 กิโลเมตรต่อชั่วโมง ผลลัพธ์ที่ได้มีลักษณะเหมือนกัน

ภาพที่ 71 และ 72 แสดงประสิทธิภาพการถอดรหัสของช่องสัญญาณการจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง และค่าไรเชียนแฟคเตอร์ K ที่ใช้มีค่าเท่ากับ 1, 3 และ 10 ตามลำดับ



ภาพที่ 71 ผลลัพธ์ความน่าจะเป็นที่ผลลัพธ์แรกผิดของช่องสัญญาณการจางหายแบบไรเชียน ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง และ K = 1, 3 และ 10 ในการทำงาน 10,000 ครั้ง

จากภาพที่ 71 พบว่า ที่ความเร็ว 60 กม./ชม. และ K = 10 จะมีประสิทธิภาพของการถอดรหัสดีที่สุด ส่วน ที่ความเร็ว 90 กม./ชม. และ K = 10, ที่ความเร็ว 60 กม./ชม. และ K = 3, ที่ความเร็ว 120 กม./ชม. และ K = 10, ที่ความเร็ว 60 กม./ชม. และ K = 1, ที่ความเร็ว 90 กม./ชม. และ K = 3, ที่ความเร็ว 90 กม./ชม. และ K = 1, ที่ความเร็ว 120 กม./ชม. และ K = 3 และที่ความเร็ว 120 กม./ชม. และ K = 1 จะมีประสิทธิภาพรองลงมาตามลำดับ



ภาพที่ 72 ผลลัพธ์ความน่าจะเป็นของการถอดรหัสที่ดีที่สุดรองลงผิดพลาดของช่องสัญญาณการจางหายแบบไรเชียน ที่ความเร็ว 60, 90 และ 120 กิโลเมตรต่อชั่วโมง และ K = 1, 3 และ 10 ในการทำงาน 10,000 ครั้ง

ส่วนภาพที่ 72 พบว่า ที่ความเร็ว 60 กม./ชม. และ K = 10 จะมีประสิทธิภาพของการถอดรหัสดีที่สุด ส่วน ที่ความเร็ว 90 กม./ชม. และ K = 10, ที่ความเร็ว 60 กม./ชม. และ K = 3, ที่ความเร็ว 120 กม./ชม. และ K = 10, ที่ความเร็ว 60 กม./ชม. และ K = 1, ที่ความเร็ว 90 กม./ชม. และ K = 3, ที่ความเร็ว 90 กม./ชม. และ K = 1, ที่ความเร็ว 120 กม./ชม. และ K = 3 และ ที่ความเร็ว 120 กม./ชม. และ K = 1 จะมีประสิทธิภาพรองลงมาตามลำดับ ซึ่งผลลัพธ์ที่ได้จาก P2 นี้จะมีลักษณะเหมือนกันกับ P1

จากผลการทดลองของช่องสัญญาณการเคลื่อนที่ (Mobile channel) ที่ได้ พบว่า ค่า P1 ที่ช่วง 0.01 ถึง 0.1 ของการมอดูเลทแบบ BPSK ในช่องสัญญาณต่างๆเป็นช่วงที่มีค่าสูงและน่าสนใจ ดังนั้นจะพบ SNR ที่สนใจในช่วง P1 ซึ่งแสดงดังตารางที่ 23

ตารางที่ 23 แสดงค่า SNR ที่พบ ในช่วง P1 เท่ากับ 0.01 ถึง 0.1 ในช่องสัญญาณการเคลื่อนที่

Channel	Velocity (km/h)	Rician factor (K)	SNR (dB)	1-P2
AWGN	-	-	2.0 - 3.5	0.48 - 0.72
Rayleigh fading	60	-	7.2 – 9.0	0.47 - 0.72
Rayleigh fading	90	-	7.8 - 9.4	0.54 - 0.75
Rayleigh fading	120	-	8.1 - 9.8	0.52 - 0.77
Ricean fading	60	1	7.2 - 8.1	0.35 - 0.61
Ricean fading	60	3	7.0 - 8.7	0.46 - 0.70
Ricean fading	60	10	6.4 - 8.1	0.66 - 0.74
Ricean fading	90	1	7.6 - 9.2	0.47 - 0.72
Ricean fading	90	3	7.5 – 9.0	0.49 - 0.72
Ricean fading	90	10	6.9 - 8.5	0.50 - 0.75
Ricean fading	120	1	8.0 - 9.6	0.49 - 0.76
Ricean fading	120	3	7.9 - 9.4	0.58 - 0.77
Ricean fading	120	10	7.1 - 8.8	0.52 - 0.74

ผลลัพธ์ของ SNR ที่ได้ดังตารางนี้ เป็น SNR ที่มีค่าต่ำมากสำหรับการใช้งานโดยทั่วไป และตัวถอดรหัสลิสวิเทอร์บิ เมื่อใช้งานร่วมกับตัวถอดรหัสวีเอสดีในรหัสคอนคาทีเนตเหมาะที่จะใช้ในการถอดรหัสสำหรับช่องสัญญาณที่มีคุณภาพต่ำมากๆ ส่วนค่า P2 ที่ได้มีค่าสูงมากพอที่จะทำให้ตัวถอดรหัสภายนอกแบบวีเอสดีมีประสิทธิภาพในการถอดรหัสสูงขึ้น เมื่อผลลัพธ์แรกผิด ตัวถอดรหัสวีเอสดีสามารถนำผลลัพธ์ที่สองที่ถอดรหัสถูกต้องมาแทนลงในผลลัพธ์แรกที่ผิดได้ ซึ่งค่าความน่าจะเป็นที่ผลลัพธ์ที่สองสามารถช่วยในการถอดรหัสถูกต้องเท่ากับ 1- P2

สำหรับกรณีที่ค่า P1 มีค่าน้อยกว่า 0.001 หรือ (10^{-3}) ในที่นี้จะไม่พิจารณา เพราะถือว่าเป็นค่ามาตรฐานในการใช้งานทั่วไป ทำให้ไม่จำเป็นต้องใช้ตัวถอดรหัสวีเอสดีที่เหมาะสมกับช่องสัญญาณคุณภาพต่ำ

วิจารณ์

1. ผลการจำลองการทำงานของตัวถอดรหัส

ผลลัพธ์การถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ที่ได้จากโปรแกรมภาษา VHDL เปรียบเทียบกับผลลัพธ์จากโปรแกรมภาษา C++ โดยเปรียบเทียบการทำงานระหว่างช่วงกลางถึงช่วงสุดท้ายของโปรแกรม ผลลัพธ์ที่ได้เหมือนกัน คือ เส้นทางที่ดีที่สุด เส้นทางที่ดีที่สุดรองลงมา เมทริกที่ดีที่สุด เมทริกที่ดีที่สุดรองลงมา ลำดับข้อมูลของการถอดรหัสที่ดีที่สุด และลำดับข้อมูลของการถอดรหัสที่ดีที่สุดรองลงมา มีค่าเหมือนกัน ผลลัพธ์ของโปรแกรม C++ ที่แสดงในรูปภาพและตารางมาจากโปรแกรมที่ใช้ในการแสดงประสิทธิภาพของการถอดรหัสคอนโวลูชันเวกเตอร์ซิมโบล โดยโปรแกรมภาษา C++ มีการเปลี่ยนแปลงการรับค่าอินพุท ซึ่งรับค่าอินพุทบิทเมทริกจากโปรแกรม Matlab เข้ามาแทนที่การจำลองการทำงานของช่องสัญญาณด้วยตัวมันเอง ทำให้ทั้งสองโปรแกรมสามารถเปรียบเทียบกันได้โดยตรง จะสังเกตได้ว่าโครงสร้างของโปรแกรมภาษา C++ จะไม่เหมือนกันทั้งหมดกับโปรแกรมภาษา VHDL เพราะว่าทั้งสองโปรแกรมมีความแตกต่างในการออกแบบแผนภาพสเตทการทำงาน จากผลลัพธ์ที่เหมือนกันของทั้งสองโปรแกรมนี้ ทำให้มีเหตุผลรับรองการทำงานของโปรแกรมภาษา VHDL ก่อนนำเอาโปรแกรมภาษา VHDL ที่ได้บรรจุลงบอร์ดอิเล็กทรอนิกส์ FPGA

2. ผลการทำงานของตัวถอดรหัสบนบอร์ด FPGA

ผลทดสอบการทำงานของตัวถอดรหัสบนบอร์ด FPGA โดยการทดสอบนี้ใช้ช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก และการมอดูเลทแบบ BFSK สำหรับข้อมูลที่ใช้ทดสอบเป็นลำดับข้อมูลที่เป็นศูนย์ทั้งหมดและข้อมูลอินพุทแบบทั่วไป ซึ่งทดสอบการทำงานจำนวน 5 ครั้ง ค่า SNR ที่ใช้เท่ากับ 7 และ 10 dB จากการทดสอบพบว่า เมื่อผลการถอดรหัสผลลัพธ์แรกถอดรหัสถูกต้อง ผลการถอดรหัสผลลัพธ์ที่สองจะต้องผิดพลาด และเมื่อผลการถอดรหัสผลลัพธ์แรกผิดพลาด ผลการถอดรหัสผลลัพธ์ที่สองอาจผิดพลาดหรือถูกต้องก็ได้ ซึ่งถูกต้องตรงตามทฤษฎี

3. ผลการสังเคราะห์วงจร (Synthesis) สำหรับตัวถอดรหัส

ผลจากการสังเคราะห์วงจรของตัวถอดรหัส ทำให้ทราบค่าประมาณของจำนวนโลจิกเกทที่ใช้คือ 2465984 โลจิกเกท แต่จำนวนโลจิกเกทที่ชิป FPGA Virtex5 รุ่น XC5VLX110 มีคือ 4423680 โลจิกเกท โดยตัวถอดรหัสจะใช้โลจิกเกทไปเพียง 55.75% ของที่ชิป FPGA จะเห็นได้ว่าใช้ทรัพยากรที่มีไม่คุ้มค่า เพราะมีการใช้ทรัพยากรเกินความจำเป็น ในงานวิจัยต่อไปจะต้องทำการแก้ไขให้ตัวถอดรหัสมีขนาดของจำนวนโลจิกลดลง เนื่องจากตัวถอดรหัสลิสวิเทอร์บิถูกออกแบบให้ใช้งานร่วมกับตัวถอดรหัสวีเอสดีซึ่งใช้รหัสคอนโวลูชัน (3, 2, 2) ในรหัสคอนคาทีเนตและรหัสวีเอสดีต้องการอินพุทในการถอดรหัส 6 สัญลักษณ์ ประกอบด้วยตัวเลือกที่หนึ่ง 3 สัญลักษณ์และตัวเลือกที่สอง 3 สัญลักษณ์ ทำให้ต้องใช้ตัวถอดรหัสลิสวิเทอร์บิ 3 ตัว ในการทำงานพร้อมกันแบบขนาน เพื่อให้ได้เอาท์พุทที่ต้องการ 6 สัญลักษณ์

4. ผลลัพธ์จากการถอดรหัสจากบอร์ด FPGA เทียบกับโปรแกรม C++

ประสิทธิภาพในเทอมของความน่าจะเป็นในการถอดรหัสผิดพลาดเปรียบเทียบระหว่างผลลัพธ์ของตัวถอดรหัสจากบอร์ด FPGA และผลลัพธ์จากโปรแกรมภาษา C++ โดยผลลัพธ์ที่ได้เหมือนกันสำหรับการใช้อินพุทบิทเมทริกที่ได้จากช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวกแบบเดียวกัน แตกต่างกันบ้างเพียงเล็กน้อยเพราะจำนวนการทดลองไม่เท่ากัน โดยตัวถอดรหัสจากบอร์ด FPGA จะใช้จำนวนการทดลองที่น้อยกว่าโปรแกรมภาษา C++ เพราะว่าช่วงของความสนใจของ P1 มีค่าสูง (0.1 – 0.01) ดังนั้น การใช้จำนวนการทดลองเพียง 800 ครั้ง ก็ทราบลักษณะของค่าความน่าจะเป็น P1 โดยประมาณได้ จึงไม่มีความจำเป็นที่จะต้องใช้จำนวนการทดลองที่สูงขึ้น และช่วงของความน่าจะเป็นนี้จะสูงกว่าช่วงปกติที่สนใจ ($< 10^{-3}$) เพราะว่าตัวถอดรหัสลิสวิเทอร์บิเป็นแค่ตัวถอดรหัสภายในของระบบรหัสคอนคาทีเนตเท่านั้น และความผิดพลาดที่แก้ไขโดยตัวถอดรหัสภายในไม่ได้ก็จะถูกแก้ไขโดยตัวถอดรหัสภายนอกต่อไป

5. วิเคราะห์ประสิทธิภาพของตัวถอดรหัสลิสวิเทอร์บิจากช่องสัญญาณ

การทดสอบประสิทธิภาพการทำงานของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ ข้อมูลที่ใช้ทดสอบการถอดรหัสเป็นข้อมูลอินพุทแบบทั่วไป การทำงานของตัวถอดรหัสจะใช้อินพุทบิทเมทริกที่ได้จากช่องสัญญาณ โดยใช้การมอดูเลทแบบ BPSK สำหรับช่องสัญญาณจะใช้ช่องสัญญาณสามแบบคือ ช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ช่องสัญญาณการจางหาย

แบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก และช่องสัญญาณการจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก การวิเคราะห์ประสิทธิภาพของตัวถอดรหัสจะใช้ค่า SNR กับค่าความน่าจะเป็นในการถอดรหัสผิดพลาดในการวิเคราะห์ประสิทธิภาพจากทั้งสามช่องสัญญาณ

จากผลลัพธ์ที่ได้พบว่า ช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก เมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นของการถอดรหัสผิดพลาดจะมีค่าลดลง ช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก เมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นของการถอดรหัสผิดพลาดจะมีค่าลดลง แต่เมื่อเพิ่มความเร็วประสิทธิภาพในการถอดรหัสจะลดลงตามความเร็วที่เพิ่มขึ้น และช่องสัญญาณการจางหายแบบไรเชียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก เมื่อ SNR มีค่าเพิ่มสูงขึ้นความน่าจะเป็นของการถอดรหัสผิดพลาดจะมีค่าลดลง แต่เมื่อเพิ่มความเร็วประสิทธิภาพในการถอดรหัสจะลดลงตามความเร็วที่เพิ่มขึ้น และเมื่อเพิ่มค่าไรเชียนแฟคเตอร์ K ความน่าจะเป็นของการถอดรหัสผิดพลาดจะมีค่าลดลง ซึ่งผลลัพธ์ที่ได้ดังกล่าวถูกต้องตรงตามทฤษฎี

ผลลัพธ์ที่ได้จากตารางที่ 23 เป็นผลลัพธ์จากการวิเคราะห์ค่า P1 ที่มีค่าอยู่ระหว่างช่วง 0.01 ถึง 0.1 ซึ่งเป็นช่วงที่มีค่าค่อนข้างสูง การเลือกพิจารณาค่า P1 ในช่วงดังกล่าว เพราะว่าเป็นค่าความน่าจะเป็นของการถอดรหัสผิดพลาดของตัวถอดรหัสภายในของระบบรหัสคอนคาทีเนตโดยผลลัพธ์ที่ผิดที่ตัวถอดรหัสภายในไม่สามารถถอดรหัสได้ จะถูกส่งไปให้ตัวถอดรหัสภายนอกแบบวีเอสดีทำการถอดรหัสต่อไป ซึ่งตัวถอดรหัสแบบวีเอสดีสามารถถอดรหัสให้ถูกต้องได้เกือบทั้งหมดหรือถูกต้องทั้งหมดได้

สำหรับค่า P1 ที่ช่วงสูงกว่า 0.1 (>0.1) ช่องสัญญาณจะมีคุณภาพที่ต่ำมากๆและจำนวนของผลลัพธ์ที่ผิดก็จะสูงขึ้นตามไปด้วย จึงไม่นิยมนำมาใช้งานในทางปฏิบัติ ส่วนค่า P1 ที่ช่วงต่ำกว่า 0.01 (10^{-3} ถึง 10^{-2}) จำนวนของผลลัพธ์ที่ผิดจะน้อยกว่า 1% ทำให้การใช้ตัวถอดรหัสภายในที่ให้ผลลัพธ์ได้หลายผลลัพธ์มีประโยชน์เพียงเล็กน้อยเท่านั้นสำหรับตัวถอดรหัสภายนอก แต่ในการใช้งานทั่วไปมักต้องการความถูกต้องมากกว่านี้ จึงยังน่าจะต้องใช้งานตัวถอดรหัสภายนอกอยู่ ส่วนค่า P1 ที่ช่วงต่ำกว่า 0.001 ($<10^{-3}$) ช่องสัญญาณมีค่าความน่าจะเป็นของการถอดรหัสถูกต้องที่ดีพอทำให้สามารถใช้งานตัวถอดรหัสวิเทอร์บิเพียงลำพังได้ โดยไม่ต้องใช้งานร่วมกับตัวถอดรหัสภายนอกอีก

ต่อมาเป็นการพิจารณาประโยชน์ของการใช้งานตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ โดยผลลัพธ์ที่สองจะมีประโยชน์ต่อตัวถอดรหัสภายนอกแบบวีเอสดีก็ต่อเมื่อผลลัพธ์ที่สองถอดรหัสได้ถูกต้องและผลลัพธ์แรกถอดรหัสผิด ซึ่งผลลัพธ์ที่สองที่ถอดรหัสถูกจะช่วยเพิ่มประสิทธิภาพและช่วยให้ตัวถอดรหัสแบบวีเอสดีถอดรหัสได้ง่ายขึ้นด้วย

สรุปและข้อเสนอแนะ

สรุป

ชิ้นงานต้นแบบของเครื่องถอดรหัสภายในที่สามารถถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ที่สร้างขึ้น จากการจำลองการทำงานของตัวถอดรหัส โดยใช้อินพุทเมทริกของช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก โดยใช้โปรแกรม Matlab ในการจำลองช่องสัญญาณรวมถึงการมอดูเลทและดีมอดูเลท พบว่า ผลลัพธ์ที่ได้จากโปรแกรมภาษา VHDL เปรียบเทียบกับผลลัพธ์จากโปรแกรม C++ มีค่าเหมือนกัน

ผลการทำงานของตัวถอดรหัสบนบอร์ด FPGA โดยใช้อินพุทเมทริกของช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก และมอดูเลทแบบ BFSK สำหรับการถอดรหัสด้วยวิธีลิสวิเทอร์บิ ค่าบิทเมทริกที่ได้จะถูกแปลงเป็นข้อมูลแบบไบนารี สำหรับใช้กับบอร์ด FPGA ผลการถอดรหัสบนบอร์ด FPGA จะได้ผลลัพธ์ 2 เอาท์พุท คือ เอาท์พุทของการถอดรหัสที่ดีที่สุด และเอาท์พุทของการถอดรหัสที่ดีที่สุดรองลงมา ทำการทดสอบข้อมูลที่มีลำดับข้อมูลเป็นศูนย์ทั้งหมดและข้อมูลอินพุทแบบทั่วไป พบว่า ผลการถอดรหัสถูกต้อง และมีผลการถอดรหัสบางข้อมูลที่ผิดพลาด

ผลลัพธ์จากการถอดรหัสจากบอร์ด FPGA เปรียบเทียบกับโปรแกรม C++ โดยใช้อินพุทเมทริกที่ได้จากช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก และการมอดูเลทแบบ BFSK โดยทดสอบการทำงานของตัวถอดรหัสด้วยโปรแกรม C++ จำนวน 10,000 ครั้ง และการทำงานของตัวถอดรหัสด้วยโปรแกรม FPGA จำนวน 800 ครั้ง พบว่า ผลลัพธ์ที่ได้ใกล้เคียงกัน

สำหรับการวิเคราะห์ประสิทธิภาพของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ โดยใช้อินพุทเมทริกที่ได้จากช่องสัญญาณ 3 แบบคือ ช่องสัญญาณรบกวนเกาส์เซียนขาวแบบบวก ช่องสัญญาณการจางหายแบบเรย์ลีที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก และช่องสัญญาณการจางหายแบบไรเซียนที่มีสัญญาณรบกวนเกาส์เซียนขาวแบบบวก พบว่า ช่องสัญญาณทั้ง 3 แบบ เมื่อ SNR มีค่าสูงขึ้น ค่า P1 และ P2 จะมีค่าลดลง และเมื่อความเร็วเพิ่มขึ้น ค่า P1 และ P2 จะมีค่าเพิ่มขึ้นตามความเร็วที่เพิ่มขึ้นด้วย ในกรณีของช่องสัญญาณการจางหายแบบเรย์ลีเทียบกับไรเซียน พบว่า ช่องสัญญาณการจางหายแบบไรเซียนมีประสิทธิภาพในการถอดรหัสที่ดีกว่าช่องสัญญาณการจางหายแบบเรย์ลี และในกรณีของช่องสัญญาณการจางหายแบบไรเซียนที่มีสัญญาณรบกวน

เกาส์เซียนขาวแบบบวก พบว่า เมื่อให้ความเร็วคงที่ ถ้า K เพิ่มขึ้น ค่า P1 และ P2 จะลดลงด้วย และเมื่อให้ K คงที่ ถ้าเพิ่มความเร็ว P1 และ P2 จะมีค่าเพิ่มขึ้นตามความเร็วที่เพิ่มขึ้นด้วย

ผลการทดลองของช่องสัญญาณที่ได้ พบว่า ค่า P1 ที่ช่วง 0.01 ถึง 0.1 ในช่องสัญญาณต่างๆ เป็นช่วงที่ SNR มีค่าต่ำมากสำหรับการใช้งานโดยทั่วไป และตัวถอดรหัสลิสวิเทอร์บิ เมื่อใช้งานร่วมกับตัวถอดรหัสวีเอสดีในรหัสคอนคาทีเนตเหมาะสำหรับการถอดรหัสช่องสัญญาณที่มีคุณภาพต่ำมากๆ ส่วนค่า P2 ที่ได้มีค่าต่ำพอที่จะช่วยทำให้ตัวถอดรหัสภายนอกแบบวีเอสดีมีประสิทธิภาพในการถอดรหัสสูงขึ้น

ผลลัพธ์ที่ได้เป็นการยืนยันว่า ชิ้นงานต้นแบบของเครื่องถอดรหัสภายในบนบอร์ด FPGA ทำงานได้ถูกต้องตามที่ออกแบบไว้ และสามารถใช้เป็นชิ้นส่วนของฮาร์ดแวร์ของรหัสคอนคาทีเนตได้

ข้อเสนอแนะ

1. ก่อนที่จะดำเนินการออกแบบซอฟต์แวร์ ควรทำการศึกษาวิธีการทำงานของฮาร์ดแวร์ก่อน เนื่องจากฮาร์ดแวร์มีความยืดหยุ่นน้อย ต่างจากซอฟต์แวร์ที่มีความยืดหยุ่นมาก

2. โปรแกรมแปลงภาษา (Interpreter) ของส่วนที่ใช้จำลองการทำงานและส่วนที่ใช้สร้างระบบ (Implementation) เป็นคนละส่วนกัน ทำให้เมื่อใช้คำสั่งในโปรแกรมภาษา VHDL ด้วยคำสั่งเดียวกัน แต่เมื่อแปลงเป็นภาษาเครื่อง (Machine language) แล้ว อาจทำให้ได้ผลลัพธ์ที่แตกต่างกัน นำไปใช้งานจริงบนบอร์ด FPGA จะให้ผลลัพธ์ถูกต้องเหมือนการจำลองการทำงาน

งานในอนาคต

ทำการแก้ไขโปรแกรมภาษา VHDL ของตัวถอดรหัสลิสวิเทอร์บิแบบสองผลลัพธ์ให้มีขนาดของจำนวนโลจิกเกทลดลง โดยการแก้ไขวิธีการเขียนโปรแกรมใหม่ เช่น ถ้าใน 1 process มีตัวแปรอยู่หลายตัว ต้องทำการแก้ไขใหม่ให้ใน 1 process มีตัวแปรอยู่เพียงตัวเดียวเท่านั้น เพื่อลดการทำงานวนซ้ำของวงจรลง และทำให้จำนวนโลจิกเกทไม่สิ้นเปลืองด้วย เป็นต้น

นำชิ้นงานตัวถอดรหัสภายในแบบลิสวิเทอร์บิสองผลลัพธ์ มาทำการต่อแบบขนาน เพื่อป้อนข้อมูลให้ตัวถอดรหัสภายนอกแบบเวกเตอร์ซิมโบลดีโคดดิงได้รวดเร็วขึ้น

นำชิ้นงานตัวถอดรหัสภายในแบบลิสวิเทอร์บิสองผลลัพธ์ที่ได้ มาใช้งานร่วมกับตัวถอดรหัสภายนอกแบบเวกเตอร์ซิมโบลดีโคดดิง ในรหัสคอนคาทีเนตที่ได้ออกแบบไว้

เอกสารและสิ่งอ้างอิง

ชำนาญ ปัญญาใส และ วัชรากร หนูทอง. 2547. **ภาษา VHDL สำหรับการออกแบบวงจรดิจิตอล.** ซีเอ็ดยูเคชัน, กรุงเทพฯ.

รัชนนท์ อินทราสกุล. 2551. **การออกแบบและสร้างระบบเพื่อใช้สำหรับการเข้ารหัสถอดรหัสสัญลักษณ์นอนไบนารีด้วยบอร์ด FPGA.** วิทยานิพนธ์ปริญญาโท, มหาวิทยาลัยเกษตรศาสตร์.

Astronlogic Research and Development. n.d. **User Manual: Ezy USB-M01.** Bangkok.

Avnet electronics marketing. 2008. **Xilinx Virtex-5 LX Evaluation Kit User guide.** User manual. Available Source: http://www.avnet.em.com.

Berlekamp, E. R. 1974. Key Papers in the Development of Coding Theory. **IEEE Press.** New York.

Bose, R. C. and D. K. Ray-Chaudhuri. 1960. On a Class of Error Correcting Binary Group Codes. **Inform. Control.** 3: 68-79.

Ellias, P. 1955. Coding for Noisy Channels. **IRE Convention Record.** 3: 37-46.

Forney, G.D. Jr. 1966. **Concatenated Codes.** MIT Press. Cambridge. MA.

Future Technology Devices International Ltd. 2002. **D2XX Driver.** User manual. Available Source: http://www.ftdichip.com.

Hamming, R. W. 1950. Error Detecting and Error Correcting Codes. **Bell Syst. Tech. J.** 29: 147-160.

Haslach, C. and A. J. Han Vinck. 1999. A decoding algorithm with restrictions for array codes. **IEEE Trans. Inform. Theory.** 45: 2339-2344.

Hocquenghem, A. 1959. Codes Correcteurs d'Erreurs. **Chiffres.** 2: 147-156.

Jakes, W.C. 1974. **Mobile Microwave communication.** John Wiley & Suns.

Kostov, N. 2003. Mobile Radio Channel Modeling in MATLAB. **Radioengineering.** 12: 12-16.

Lin, S. and Daniel J. Costello. 2004. **Error control coding: fundamentals and applications.** Information theory. Pearson-Prentice Hall.

Metzner, J. J. and E. J. Kapturowski. 1990. A general decoding technique applicable to replicated file disagreement location and concatenated code decoding. **IEEE Trans. Inform. Theory.** 36: 911-917.

Metzner, J. J. 2003. Vector symbol decoding with list inner symbol decisions. **IEEE Trans. Comm.** 51(3).

Proakis, John G. 2001. **Digital communications.** McGraw-Hill. Singapore.

Prange, E. 1957. Cyclic Error-Correcting Codes in Two Symbols. **Air Force Cambridge Research Center.** 57: 103.

Rappaport, Theodore S. 1995. **Wireless Communications Principles and Practice.** Prentice Hall PRT. New Jersey.

Reed, I. S. and G. Solomon. 1960. Polynomial Codes Over Certain Finite Fields. **SIAM J.** 8: 300-304.

Roder, M. and R. Hamzaoui. 2006. Fast tree-trellis list Viterbi decoding. **IEEE Transactions on Communications.** 54(3): 453-461.

Seshadri, N. and C. E. W. Sungberg. 1989. General Viterbi algorithms for error detection with convolutional codes. **IEEE GLOBECOM 1989.** 3: 1534-1538.

Seshadri, N. and C. E. W. Sungberg. 1994. List Viterbi decoding algorithm with applications. **IEEE Transactions on Communications.** 42(1): 313-323.

Seo, Y. S. 1991. **A new decoding technique for convolutional codes.** Ph.D. Thesis. Pennsylvania State University.

Tuntoolavest, U. and J. J. Matzner. 2002. Vector symbol decoding with symbol decisions and outer convolutional codes for reliable communications. **Integrated Computer-Aided Engineering Journal.** 9: 101-116.

Tuntoolavest, U. 2003. Performance comparison between a maximum and a non-maximum distance outer code decoding. **The 3rd International Symposium on Communications and Information Technologies Proceeding (ISCIT 2003)**. Songkhla, Thailand.

Tuntoolavest, U. and A. Seubnaung. 2007. Performance Investigation of Convolutional Vector Symbol Decoding with Larger then Two Choices and with Incomplete Second Choices. **Proceedings of the 45th Kasetsart University Annual conference 2007.** Kasetsart University, Thailand.

Tuntoolavest, U. 2009. **Vector Symbol Decoding for Wireless Fading Channel.** VDM publishing. ISBN-13: 978-3639132861 (www.amazon.com).

Tuntoolavest, U. and P. Noradee. 2009. Design and Implementation of List-of-2 Viterbi Decoder with VHDL and its Application. **ECTI-CON 2009.** Pattaya, Chonbury, Thailand.

Viterbi, A. J. 1967. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. **IEEE Trans. Inform. Theory.** IT13: 260-269.

Viterbi, A.J. and J.K. Omura. 1979. **Principle of Digital Communication and Coding**. McGraw-Hill. New York.

ภาคผนวก

ภาคผนวก ก

ผลงานที่ได้รับการตีพิมพ์

Design and Implementation of List-of-2 Viterbi Decoder with VHDL and its Application

Usana Tuntoolavest* and Pongpisut Noradee
Department of Electrical Engineering
Kasetsart University, Bangkok, Thailand
Tel: +66-2-9428555 ext 1565, Fax: +66-2-942-8555 ext 1550
E-mail: fengunt@ku.ac.th and g4965128@ku.ac.th

***Abstract*— To use List viterbi decoding (LVA) as inner decoder with error detection outer codes, the list size must be large enough to almost always include the correct seqeunce. However, if error correction is used, a very short list of two can be enough. Convolutional Vector Symbol Decoding (conv. VSD) has been shown that it could select the correct input sequence from a list of multiple sequences. Previous work showed that list of 2 choice was the most effective size in terms of performance and complexity from the implementation viewpoint of conv. VSD. In addition, list-of-2 Viterbi has been simluated with C++ program to demonstrate that it is suitable as the inner decoder for conv. VSD outer codes for some channels. In this paper, we propose the design and implementation of LVA with list of 2 in a hardware language called VHDL. This VHDL program can readily be implemented on an FPGA board and used as the inner decoder for the conv. VSD. The results show that VHDL program worked exactly as designed by comparing thier outputs with the output of previous C++ program.**

I. INTRODUCTION

List Viterbi Decoding Algorithm (LVA) was proposed by Seshadri and Sundberg [1] in 1994. It extended the concept of the original Viterbi decoding algorithm [2] by providing many possible decoded sequences instead of only one most likely sequence. By using a long enough list, error detection code can be used to check and select the correct decoded sequence from the list. In 2000 [3] and 2001 [4], Chen and Sundberg showed the performance of LVA for wireless channel and for continuous transmission. In 2006, Roder and Hamzaoui proposed the faster LVA using tree-trellis [5]. Recently in 2008, another group of researcher presented joint erasure marking and LVA [6]. However, in all these works, the list must be long enough.

This paper proposes the design and implementation of LVA with very short list of only 2 possible sequences. The short list greatly reduces the computation. To implement LVA in hardware, the concept of LVA was designed with state diagram and written in a hardware language called VHDL (Very high speed integrated circuit Hardware Description Language). The VHDL language was selected due to the ease of simulation and because it can readily be implemented on an FPGA board. To implement with VHDL, the algorithm of list-of-2 Viterbi was designed using state diagram to indicate the steps of the algorithm. This state diagram design refers to the state in VHDL and not the state of the convolutional code.

Supported by the Graduate School of Kasetsart University and the Kasetsart University Research and Development Institute

Vector Symbol Decoding (VSD) is a nonbinary decoding algorithm that can be applied to both block and convolutional codes with nonbinary symbols. It was first proposed by Metzner in [7]. It was shown that it can select the correct symbol from a list of many possible values of that symbol. Convolutional Vector Symbol Decoding (Conv. VSD) proposed in [8] was based on the original VSD principle with attractive feature that it can start decoding with a partial and usually very short received sequence instead of having to wait for the whole received sequence as in the block VSD case. In the implementation viewpoint, the optimum list size for conv. VSD was shown to be a list of two possible values for each symbol [9]. Larger list gave little added performance with much higher complexity.

List-of-2 Viterbi has been shown to be suitable as the inner decoder for the conv. VSD outer code for concatenated codes. The good performances of these concatenated codes were shwn in [10] for a simplified two state fading channel, an AWGN channel and a Rayliegh fading with AGWN channel. In addition, the phase II of convolutional Vector symbol decoder [11], which can be used as the outer decoder of the list-of-2 Viterbi inner decoder, have also been implemented on an FPGA board. This makes it simpler to later combine the inner and outer decoder into a single hardware.

II. BACKGROUND

The LVA algorithm can be divided into 4 paths as the original Viterbi: 1. Initialization, 2. Recursion, 3.Termination, and 4. Trace back. The related equations can be found in [1]. This paper focuses on the design and implementation with VHDL and does not repeat the widely known Viterbi algorithm to save space. Only key points that differentiate the List-of-2 Viterbi and the original Viterbi are stated. List-of-2 Viterbi used in this paper was the parallel LVA [1], which meant that the algorithm kept two highest accumulated metrics instead of only one highest metric at each state and each time interval. The metrics were correlation metrics. The best path and the second best path were the paths that had the highest and the second highest path metric values respectively.

At each state, two accumulated metrics were calculated from the accumulated metric of the previous state plus the current branch metric. During the steady state, there were four paths that compete with each other at the incoming of every

978-1-4244-3388-9/09/$25.00 ©2009 IEEE

state. The best two paths with the two highest metric were recorded. The best two paths might come from the same previous state or from different previous states. It is important to note that the most likely path and the second most likely path must diverge only once and then merge back [1].

An example with very short length of 5 data bits is used to illustrate the list-of-2 Viterbi idea. The data size can be increased and the intended size was set to 32 data bits for the application as the inner code of VSD outer code.

Example 1: Consider a (2,1,4) convolutional code with $G(D) = [(1+D+D^4)\ (1+D+D^3+D^4)]$ that terminates after five bit inputs. Since the memory size is 4, there are $2^4 = 16$ states in the trellis diagram and 4 more "0" must be input to the encoder after the termination to clear the memory. The trellis diagram for the input sequence (11101) is shown in Fig. 1 and the corresponding path metrics are shown in Table 1.

To find the best and second best paths, the input bit metrics were imported from Matlab using the raylieghchan() and awgn() functions to generate a Rayleigh fading channel with additive white Gaussian noise for Doppler = 50 and SNR = 10. The modulation scheme was binary frequency shift keying.

Notice that the solid circles and the dashed circles match the paths in Fig. 1. The number in each parenthesis in Table 1 is the previous state that leads to the path metric. To simplify the table, the second best path metrics are shown only when they help differentiate the different path metric values when the two paths join toward the end.

Note that the branch metrics are not shown here due to limited space. They also play an important role in how the algorithm selects the best and second best states at each time interval. This is because the accumulated path metric results from the addition of the path metric of the previous state and the branch metric. To make this clear, consider the time interval T=8 and T=9. At t=8, the highest path metric of state 0 (=436.12) was higher that the second path metric of state 1 (=375), but the second best path at T=9 came from state 1

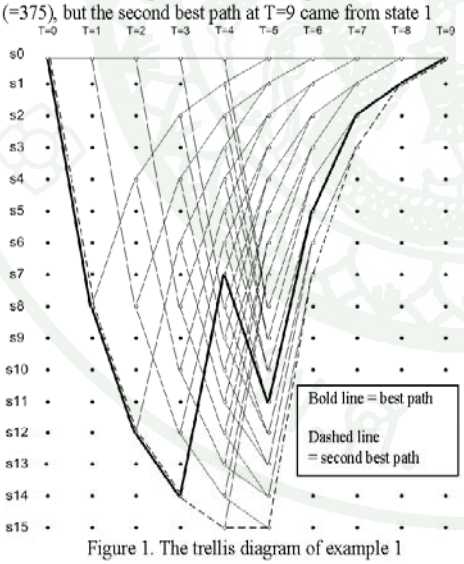


Figure 1. The trellis diagram of example 1

Table 1. The path metric of the trellis diagram in Fig.1

State	T=0	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8	T=9
0	0	4.15(0)	133.05(0)	194.29(0)	206.43(0)	251.04(1)	310.32(1)	426.65(1)	436.12(0),396.37(1)	532.86(1),457.45(1)
1					250.29(2)	309.55(3)	407.49(3)	378.61(3)	450.41(2), 375(1)	
2				240.96(4)	27.40(4)	224.56(5)	344.09(5)	423.25(5)		
3					309.06(6)	403.3(7)	373.98(7)	374.94(7)		
4			187.75(8)	15.3(8)	158.33(8)	284.59(9)	313.75(9)			
5					223.81(10)	343.32(11)	404.08(11)			
6				207.73(12)	99.56(12)	190.55(13)	347.25(13)			
7					402.8(14)	369.8(15)	470.31(15)			
8		177.7(0)	14.17(0)	134.10(0)	210.44(0)	250.66(1)				
9					283.92(2)	309.69(3)				
10				188.85(4)	29.45(4)	224.10(5)				
11					342.69(6)	403.43(7)				
12			306.6(8)	75.41(8)	146.32(8)	284.67(9)				
13					190.17(10)	343.19(11)				
14				367.89(12)	87.54(12)	190.93(13)				
15					269.17(14)	269.66(15)				

Bold line = best path

Dashed line
= second best path

instead of state 0. This is because the branch metric from state 1 to state 0 from T=8 to T=9 (=82.4487) was much higher than the branch metric from state 0 to state 0 from T=8 to T=9 (=0.513184). Therefore, the accumulated path metric of 375+82.4487 was higher than 436.12+0.513184 and state 1 was selected as both the best and second best states.

III. Method

The list-of-2 Viterbi was implemented for the code in example 1 with 32 bits data sequence. The design in VHDL used state diagram design. Note that these states are for VHDL and they were not the same as the states of the convolutional code, which were s0 to s15 as shown in Fig.1.

The initialization, recursive and termination parts were combined together and done in 5 VHDL states for each time interval as shown in Fig. 2:

State ① Receive current bit metrics as inputs and set the Conv. code states (s0 to s15) and the branches that need to function at each time interval. Start with interval T = 1, which has only two functioning state s0 and s8 as shown in Fig.1

State ② Calculate the necessary branch metrics and record the corresponding previous conv. code state.

State ③ Calculate the path metric:

Path metric = previous path metric + current branch metric

State ④ Compare the competing path metrics and keep the two highest values.

State ⑤ Keep record of the best and second best path metrics and the corresponding previous conv. code states that lead to those metrics to be used in the trace back part. Repeat state ① to⑤ for time interval T= 2 to T=36. Note that T = 36 came from the number of input bits (32 bits) plus the number of memory size (m = 4).

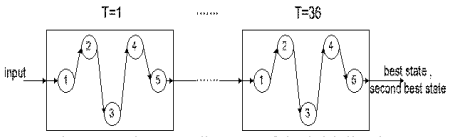


Figure 2. The state diagram of the initialization, recursion and termination parts

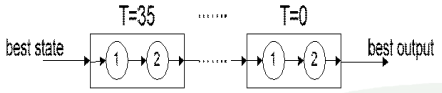


Figure 3. The state diagram for the trace back part of the best path

For the trace back part of the best path, there were 2 VHDL states for each time intervals as shown in Fig. 3:

State ① Start from T = 35 (or last time interval - 1 for more general cases). Use the information from state ⑤ of Fig. 2. to trace and record the conv. code state of the best path metric at current time interval. For example, at T= 35, the algorithm kept the record of the best state at T = 35 that led to the best path at the end of the trellis (T = 36). At T =34, it kept the record of the best state at T = 34 that led to the best state at T = 35.

State ② Decode the best decoded (data) bit from the best state found at each interval. Repeat state ① to ② of Fig. 3 for T = 34 to T = 0. This will give the best state sequence and the best decoded (data) sequence.

For the trace back part of the second best path, the algorithm would perform the same state ① to ② of Fig. 3 as the best path. The difference was that the it would use the second best state at each time interval to find the state and the decoded bit. Obviously, the result would not be the overall second best path because the algorithm always picked the lower of the two metrics at each state. To make the result the correct second best path that diverged from the best path only once as mentioned before, two additional VHDL states were added at the end as shown in Fig. 4:

State ⓐ Compare the conv. code state obtained from the trace back of best path and the trace back of the path that uses second best states starting from T = 35, 34,... and so on until the first interval that the states from the two trace backs are different. Stop at that interval. Call it T_a. This gives the interval when the states of the best and the second best paths are going to merge back and join together until the end. Consider example 1 in section 2, the algorithm will find that the two trace backs give different states at T =7, which means that the two paths will join together from T = 8 to the end of the trellis.

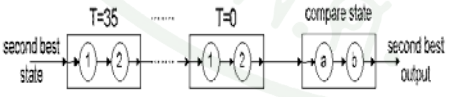


Figure 4. The state diagram for the trace back part of the second best path

State ⓑ It should be emphasized that the correct second best path is the path that has the highest path metric up until the time interval when it join the best path. Otherwise, there must be other paths with higher path metric. Therefore, when state ⓐ stops at interval T_a, state ⓑ will start the same trace back process in Fig. 3, but begin at T = T_a-1 instead of T = 35 (or the last time interval -1). This will be done using the best path metric at each state. This procedure will find the best path that ends at the second best state at T = T_a, which will result in the correct second best path and the second best decoded (data) seqeunce.

IV. Results

The simulations were done for the (2,1,4) code from example 1 using the input bit metrics of the Rayliegh fading with AWGN channel for various parameters (not shown). The results shown are for a 32-bit all-zero data sequence and channel parameters: Doppler=50 and SNR=10. The modulation technique was BFSK with a matched filter receiver. The channel, modulation and demodulation were simulated in Matlab and input to VHDL and C++. The results from VHDL were in waveforms. The results from C++ were summarized in tables for the ease of comparison. Only a few intervals are shown due to limited space.

Fig. 5 shows the input bit metrics (output of demodulator) generated from Matlab. Fig. 6, 7 and Table 2 show the best and second best path metrics for three intervals (T= 5 to T= 7) during the steady state from VHDL and C++.

Fig. 8, 9 and 10 show that the best and second best state sequences were the same from VHDL and C++ programs. Fig. 11 shows the best and second best decoded data sequence. It can be seen that in this case the best decoded sequence was the correct one because the input was an all zero sequence.

/testbench1/uut/state_t	1	2	3
/testbench1/uut/r1x72_a1	0.7855	-0.3091	0.0745
/testbench1/uut/r1x72_a2	-1.6614	1.228	-0.7025
/testbench1/uut/r1x72_b1	-1.707	0.9453	-0.2754
/testbench1/uut/r1x72_b2	5.0887	-5.6507	4.9381

Figure 5. The bit metrics generated by Matlab were used as inputs of Lits-of-2 Viterbi in VHDL.

/testbench1/uut/pmetric_t0	81.6232	185.3788	117.788
/testbench1/uut/pmetric_t1	78.6803	109.141	122.177
/testbench1/uut/pmetric_t2	108.075	188.6675	132.912
/testbench1/uut/pmetric_t3	67.0617	111.988	121.138
/testbench1/uut/pmetric_t4	88.7548	193.733	138.67
/testbench1/uut/pmetric_t5	92.04	124.285	145.145
/testbench1/uut/pmetric_t6	111.912	195.0296	134.476
/testbench1/uut/pmetric_t7	54.0738	116.155	130.001
/testbench1/uut/pmetric_t8	87.1055	188.2566	129.001
/testbench1/uut/pmetric_t9	78.8707	118.809	135.477
/testbench1/uut/pmetric_t10	114.552	196.3823	144.145
/testbench1/uut/pmetric_t11	67.2431	121.636	139.889
/testbench1/uut/pmetric_t12	84.2795	190.2765	127.435
/testbench1/uut/pmetric_t13	91.8586	114.617	126.614
/testbench1/uut/pmetric_t14	106.426	198.4867	123.244
/testbench1/uut/pmetric_t15	53.8824	106.491	121.995

Figure 6. Best path metrics of the 16 conv. code states for the T = 5 to T = 7 interval from VHDL

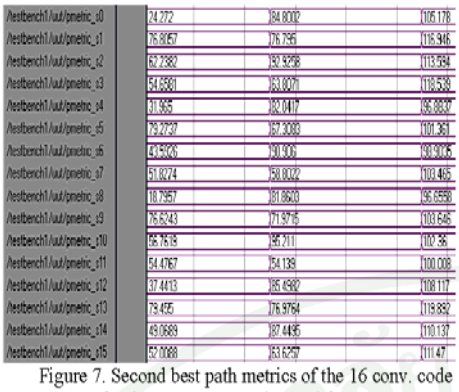


Figure 7. Second best path metrics of the 16 conv. code states for the T = 5 to T = 7 interval from VHDL

Table 2. Path metrics of the 16 conv. code states for the T = 5 to T =7 intervals from C++

a) Best path metrics

state	T=5	T=6	T=7
0	81.6292	85.3168	117.768
1	78.6893	109.141	122.177
2	109.075	98.6675	132.912
3	67.0617	111.998	121.139
4	89.7548	93.733	138.67
5	92.04	124.285	145.145
6	111.902	96.0296	134.478
7	54.0738	116.159	130.001
8	87.1055	88.2566	129.001
9	78.8707	118.809	135.477
10	114.552	96.3623	144.146
11	67.2431	121.636	139.669
12	84.2785	90.2765	127.436
13	91.8586	114.617	126.614
14	106.426	98.4861	123.244
15	53.8924	106.491	121.995

b) Second best path metrics

state	T=5	T=6	T=7
0	24.272	84.8002	105.178
1	76.8057	76.795	116.946
2	62.2382	92.9258	113.594
3	54.6581	63.8071	118.639
4	31.965	82.0417	96.8837
5	79.2737	67.3083	101.361
6	43.5926	90.906	98.9035
7	51.8274	58.8022	103.465
8	18.7957	81.8603	96.6558
9	76.6243	71.9715	103.646
10	56.7618	95.211	102.36
11	54.4767	54.139	100.008
12	37.4413	85.4982	108.117
13	79.455	76.9764	119.892
14	49.0688	87.4495	110.137
15	52.0088	63.6257	111.47

Figure 8. The best ("state1" signal) and second best ("state2" signal) state sequences from the trace back for T = 35 to T = 25

T	35	34	33	32	31	30	29	28	27	26	25
best_state_seq[T] is	0	0	0	0	0	0	1	3	6	13	11

Figure 9. The best state sequence from C++ for T=35toT= 25

T	35	34	33	32	31	30	29	28	27	26	25
second_state_seq[T] is	0	1	2	4	9	3	6	12	8	1	3

Figure 10. The second best state sequence from C++ for T = 35 to T = 25

Figure 11. Best ("rec_out1" signal) and second best ("rec_out2" signal) decoded data sequences

V. Discussions and Conclusions

The VHDL results have been compared with the results of C++ program for list-of-2 Viterbi. They are exactly the same for all intermediate and final steps and various channel parameters. Specifically, the best and second paths, the best and second best metrics and the best and second best decoded data sequences are the same. The results of C++ shown in tables came from the same program that was used to show the performance of conv. VSD with list-of-2 Viterbi in [10]. Some modifications were made to the C++ program to receive the input bit metrics from Matlab instead of simulating the channel itself. This was done so as to be able to compare the results from the two programs directly. Note that the structure of the C++ program was not exactly the same as the VHDL one because the C++ was not designed as a state diagram. These same results from the two programs with two structures confirm the validity of the VHDL program. The next step will be inputting this VHDL program onto an FPGA board and test the actual hardware. Then the output of the list-of-2 Viterbi decoder can be used as the input of the conv. VSD decoder to obtain a concatenated conv. code decoder .

References

[1] N. Seshadri and C. W. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Trans. on Communications*, vol.42, 1994, pp.313-323

[2] S. Lin and D. J. Constello, Jr., *Error Control Coding*, 2nd edition, 2004, Pearson Prentice Hall.

[3] B. Chen and C. W. Sundberg, "List Viterbi algorithms for wireless systems," *VTC 2000 spring*,Tokyo,2000, pp.1016-1020

[4] B. Chen and C. W. Sundberg, "List Viterbi algorithms for continuous transmission," *IEEE Trans. on Communications*, vol. 49, issue 5, May 2001, pp.784 - 792

[5] M. Roder and R. Hamzaoui, "Fast tree-trellis list Viterbi decoding," *IEEE Trans. on Communications*, vol. 54, issue 3, March 2006, pp. 453 - 461

[6] T. Li, W. H. Mow and M. Siu, "Joint erasure marking and list Viterbi algorithm for decoding in unknown non-Gaussian noise, *IEEE Trans. on Wireless Communications*, vol. 7, issue 3, March 2008, pp. 787 – 792

[7] J.J. Metzner and E.J. Kapturowski, "A general decoding technique applicable to replicated file disagreement location and concatenated code decoding," *IEEE Trans. Inf. Theory*, vol.36, pp.911-917, July 1990.

[8] U. Tuntoolavest and J.J. Metzner, "Vector symbol decoding with list symbol decisions and outer convolutional codes for reliable communications," *Integrated Computer-Aided Engineering Journal*, vol. 9, no. 2, 2002, p. 101-116, IOS Press.

[9] U. Tuntoolavest and A. Seubnaung, "Performance Investigation of Convolutional Vector Symbol Decoding with Larger than Two Choices and with Incomplete Second Choices," *Kasetsart Journal (Natural Science)*, v 41, n 5, January 2008, p 364-370.

[10] U. Tuntoolavest, "Performance comparison between a maximum and a non-maximum distance outer code decoding," *International Symposium on Communications and Information Technologies Proceeding (ISCIT 2003)*, Sept. 2003, Thailand.

[11] U. Tuntoolavest, A. Seubnaung, and R. Intharasakul, "Convolutional Vector Symbol Decoder Phase II on FPGA: Correct with Second Choice," *ISCIT 2007*, Oct. 2007, Sydney, Australia, pp. 140-145

Lab Prototype of a List-of-2 Viterbi Decoder: A Diversity Inner Decoder for the Outer Vector Symbol Decoder

Usana Tuntoolavest* and Pongpisut Noradee
Department of Electrical Engineering
Kasetsart University, Bangkok, Thailand
Tel: +66-2-9428555 ext 1565, Fax: +66-2-942-8555 ext 1550
E-mail: fengunt@ku.ac.th and g4965128@ku.ac.th

***Abstract*- This paper presents the implementation result of List-of-2 soft-decision Viterbi decoder on FPGA platform. It was to be used as the inner decoder of a special designed concatenated code, of which the outer decoder is Vector Symbol Decoder (VSD). Since list-of-2 Viterbi was not the final decoding stage, it was not required to provide low decoding failure probability. Therefore, only a small list of 2 was enough. This use of list inner decoder is useful to VSD only if the second inner decoded sequence is correct when the first one was wrong. The result shows that this probability of interest is high for low-SNR AWGN channel and BFSK scheme. Therefore, List-of-2 Viterbi would be attractive for VSD for this channel. The operation of this lab prototype was tested by comparing its decoding failure probabilities with the results from C++. The results showed that they were almost the same, which proved that the lab prototype worked properly.**

I. Introduction

List-of-2 Viterbi decoder is based on the List Viterbi Decoding (LVA) Algorithm that was first presented by Seshadri and Sundberg [1] in 1994. LVA combines the concept of list decoding [2], [3] with Viterbi algorithm (VA) [4]. List decoding provides a list of more than one possible decoded sequence. That is the decoder helps reduce the ambiguity of the transmitted message, but does not make the final decision as to which message is assumed to have been transmitted [5]. By combining list decoding with VA, Seshadri and Sundberg [1] proposed the use of LVA as an inner decoder of a concatenated code. Their proposed outer decoder was a simple error detecting code, which could select the correct codeword from the list of many possible choices given that the list contained enough choices that the correct choice was almost always in the list. However, using long list size significantly increased the number of computations and storage size from the original VA. There were attempts to reduce the complexity of LVA such as by using List Single-Wrong-Turn (LSWT) Viterbi decoding [6]. This is not necessary for a very small list size of 2 presented in this paper.

Instead of long list with the error detecting outer code, Metzner and Tuntoolavest [7, 8] implicitly used another design of concatenated code. This designed is explicitly explained in this paper as an application of the presented lab prototype. This design employs list decoding with much shorter list size as the inner decoder and relied on a specific outer error correcting decoder called "Vector Symbol Decoding" (VSD) to either select the right codeword if it is on the list or to correct the errors if the right codeword is not on the list. Tuntoolavest [9] showed further that list of 2 choices was optimum for the implementation viewpoint since longer lists led to little gain for this design of LVA-VSD concatenated code. However, the performance was based on an assumption of input error probabilities and not on an actual channel condition.

List-of-2 Viterbi decoding can be viewed as providing diversity to VSD, the outer decoder. This diversity is very useful for VSD, since it improves the performance of VSD and often simplifies the decoding steps. VSD is a general decoding technique that can decode any linear block or convolutional codes with large nonbinary symbols (typically 32-bit or larger symbols). The details of this VSD algorithm were shown for three channels using math models in [9].

The next step was to implement the decoder on the FPGA platform as Zhang and Ryan did in 2009 for the decoder of LDPC (Low Density Parity Check Code) since the results that based on FPGA decoder were "reliable and repeatable" [10]. In addition to the use of FPGA, the test system was designed in such a way that each component can be replaced by a hardware unit. To achieve this, the C++ program was modified to receive input from Matlab to make the simulation more realistic [11]. In the same paper, the List-of-2 Viterbi was also designed and written with VHDL. The preliminary result was shown by comparing the intermediate results from VHDL with the C++ only for an input sequence and an example of error pattern. However, the actual hardware was not implemented or tested and the input was limited to one received sequence at a time. More work was necessary to reach the goal of a LVA-VSD prototype system.

In this paper, we present the actual hardware of List-of-2 Viterbi decoder on a FPGA board. In addition, the input is allowed to include many received sequences at a time. The number of received sequences is only limited by the program that is used to transfer the file between the computer and the FPGA board and not by the decoder itself. The encoder, the modulator, the channel and the demodulator were simulated by Matlab in such a way that they could readily be realized by hardware components. The List-of-2 Viterbi decoder then

Supported by the Kasetsart University Research and Development Institute

978-1-4244-5607-9/10/$25.00 ©2010 ECTI

received the bit metrics from the demodulator as its input sequences and returned the decoded sequences back. By allowing a large number of input sequences, the performance in terms of decoding failure probability can be found directly from the operation of this prototype.

II. Background

A. Concept of List-f-2 Viterbi

The concept of List-f-2 Viterbi was extended from the original well-known VA. The general algorithm for LVA with any list size was described in [1]. The differences between the List-of-2 Viterbi and the original Viterbi, including an example was shown in [10]. Basically, the List-of-2 Viterbi decoder has to keep two highest accumulated (correlation) metrics instead of one (the most highest value) at each node and at each stage of the trellis. During the steady states, there will be four completing paths coming into each node. These paths are ranked and the two paths with the highest accumulated metrics are recorded. It should be noted that these two survival paths can come from the same or different previous states. Based on the nature of the algorithm, both outputs are always different. Moreover, both paths must diverge from each other only once. This is one of the properties that can be used to check whether the decoding performed correctly.

B. LVA-VSD concatenated coding system

Although this concept was mentioned in [8,10], this section will explicitly explain the concepts and an example of the LVA-VSD concatenated coding system. The system uses List-of-2 Viterbi as the inner decoder and uses VSD as the outer decoder as shown in Fig.1. The outer encoder can be a block or convolutional code with nonbinary symbols. The inner encoder is a binary convolutional code.

As an example, let the outer encoder be a (3,2,2) convolutional code with 32-bit symbols and the transfer function

$$G(D)=\begin{bmatrix} 1+D+D^2 & D^2 & 1 \\ D & 1+D^2 & 1+D+D^2 \end{bmatrix}$$

Each 32-bit symbol is then transmitted to the inner encoder. Suppose the inner decoder is a binary (2,1,4) convolutional code with G(D) = [(1+D+D^4) (1+D+D^3+D^4)]. The output codeword for each 32-bit input symbol then contains 72 bits (=(32 data bits + 4 tail bits)*2) as shown in fig. 2.

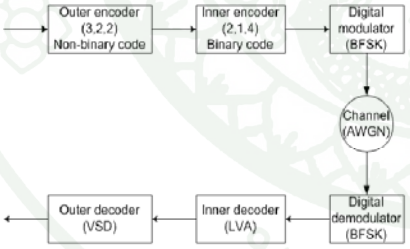


Figure 1. block diagram of a communication system with LVA-VSD

Figure 2. The input-output for each component in the transmitter

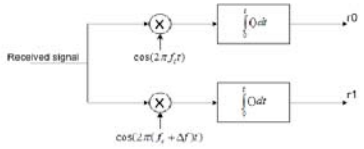


Figure 3. Matched filter receiver for BFSK signal

Figure 4. The input-output for each component in the receiver

Suppose the modulation is a binary frequency shift keying (BFSK) and the channel is an AWGN channel. The demodulator is then a set of two matched filters, each matches to each carrier frequency as shown in fig. 3. The output *r0* and *r1* are the bit metrics to be input to the inner decoder. The bit metric values are real values for soft-decision decoding with the list-of-2 Viterbi inner decoder. For each 144 input bit metric values (72 for *r0* + 72 for *r1*), the inner decoder outputs two ordered (most likely and second most likely) choices of 32-bit symbols to the outer VSD decoder as shown in fig. 4.

The performance of the overall system depends on the decoding failure probability of both LVA and VSD. For this paper, we only concern with LVA. Since List-of-2 Viterbi outputs two choices of 32-bit symbols to VSD, the two probabilities in consideration are *p1*, the probability that the first choice is wrong and *p2*, the probability that the second choice is wrong given that the first choice was wrong. This *p2* is defined as stated because the second choice is only useful for VSD if it is correct when the first choice was wrong. This is due to the fact that VSD can easily select the correct symbol from its input list.

It should be noted that since List-of-2 Viterbi is not the final decoding stage, it is not required to provide low decoding failure probability. By analyzing the results from [9], *p1* in the range of 0.1-0.01 and *p2* < 0.5 are good enough to make the overall decoding failure probability of the LVA-VSD system be in the usual range of interest (10^{-5} to 10^{-3}). Therefore, the previously specified ranges of *p1* and *p2* are of interest.

III. Hardware Implementation

The lab prototype was implemented by downloading the VHDL program of the List-of-2 Viterbi to the FPGA chip in the selected FPGA board, which was the Virtex-5 LX Evaluation Kit from Avnet. This board used the FPGA chip number XC5VLX110-FF676 C in the Virtex-5 family from Xilinx. This board has 160x54 = 8640 CLBs (Configurable

Logic Blocks). Each CLB consisted of eight flip-flops and eight 6-input LUT (Look Up Table), which resulted in 4,423,680 gates. This high number of gates was planned for future work. The clock signal frequency was about 550 Mhz.

The program downloading was done by using the Platform Cable USB as shown in fig. 5a). To transfer data between the computer and the board, a USB module was used as the interface as shown in fig. 5b). This USB module was Ezy USB-M01 from Astron Logic Research and Development. It could transfer data at the rate 1 MB/sec. It also employed IC FT245BM from FTDI (Future technology Devices International Ltd.) company. Fig 6. shows the complete system during test.

IV. Test system

To test this lab prototype, Matlab was used to simulate the outer encoder, inner encoder, modulator, channel and demodulator. These components employed the codes, AWGN channel and BFSK modulation scheme as described in the example of section II part B. The data sequence was also generated randomly with Matlab and input to the outer encoder. The outputs of the demodulator were then transferred to the FPGA board via an interface program called "USB Transceiver for List Viterbi V1.1 with DEBUG protocol". Next, the List-of-2 Viterbi decoder on the board performed the decoding process and output the two decoded sequences for each received sequence back to the computer via the same program. The "USB Transceiver for List Viterbi V1.1 with DEBUG protocol" was written by Seubnaung with C++ and function tools from the D2XX Direct Drivers of FTDI company [12]. The USB transceiver program allowed 1 MB of data file to be transmitted at a time.

a)Download VHDL with Platform cable b) Transfer data via USB module

Figure 5. The interface between the FPGA board and the computer

Figure 6. System during the test

The lab prototype was implemented by downloading the VHDL program of the List-of-2 Viterbi to the FPGA chip in the selected FPGA board. To run performance test, two different points need to be considered. First, to check whether the decoder decoded each received sequence successfully, the data sequence was compared to the decoded sequences. If it was the same as the most likely decoded sequence, this particular decoding was successful by itself. If it was the same as the second most likely decoded sequence, the decoding was not successful by itself. However, since the correct sequence was in the list, the outer decoder would almost always be able to correct this erroneous sequence.

Second, to check whether the prototype works properly as designed, the decoding result was recorded for a test sequence of 400 32-bit data sequences. The probabilities *p1* and *p2* were analyzed and compared with the probabilities resulted from List-of-2 Viterbi in C++ program.

V. Results

Examples of the display of The USB transceiver program on the computer screen are shown in fig. 7 and 8. The results in fig. 7 were for the case of all-zero data sequences and SNR = 8. In this case, the codewords were all-zero codewords. These codewords were then passed through the BFSK modulator, the

Figure 7 Example of test results for an all-zero data sequence, AWGN channel with SNR = 8 and BFSK modulation

AWGN channel with SNR = 8 and the demodulator. Each of the demodulator outputs (bit metrics *r0* and *r1*) was converted from real values to a 24-bit word and shown as a line of hexadecimal value in the left-hand side of display window. The right-hand side shows that all of the most likely decoded sequences were decoded successfully. That is they are all 00 00 00 00_{16} (= 32 zero bits). In addition, all of the second most likely decoded sequences were wrong since they were not all zeroes. This is as expected since List-of-2 Viterbi was designed such that the two outputs would always be different.

Fig. 8 shows the results from another case of randomly generated data sequences and a lower SNR (SNR = 7). The codewords were encoded by the (2,1,4) inner encoder and passed through the same system. It is not as easy to see whether the decoded sequences were correct on the right-hand side of fig 8 as in fig.7. However, by comparing the decoded sequences with the data sequences, both decoded sequences for the first received sequence were discovered to be wrong. For the remaining 4 received sequences, the most likely decoded sequences were decoded successfully.

Since List-of-2 Viterbi is to be used as the inner decoder of VSD, the relationship between the two decoding failure probabilities (*p1* and *p2*) and the input probability of VSD are of interest. This can be described as follows. First, *p1* is the same as the (first choice) input symbol error probability of VSD, the outer encoder. Second, consider that the second choice will be helpful to VSD only if it is corrected when the first choice was wrong. This probability that the second choice in the list would be helpful is equal to *1-p2*.

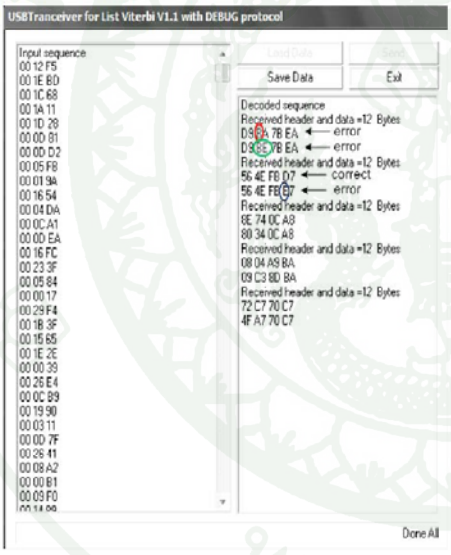


Figure 8. Example of test results for a general data sequence, AWGN channel with lower SNR (SNR=7) and BFSK modulation

Table I and II show the results from the C++ simulation and from the hardware test. Since the range of interest is *p1* between 0.1-0.01 and *p2* < 0.5 as discussed in section II, low SNR values were considered. From these two tables, it is seen that with $E_b/N_0 \geq 7$ dB, the correct choice will always be in the list of the two decoded sequences from List-of-2 Viterbi decoder. For E_b/N_0 between 5 to 7 dB, VSD can easily correct the error symbols since *p2* is low (0.16 to 0). For example, with $E_b/N_0 = 5$ dB, the input symbol error probability of VSD is 0.125 or an average of 125 error symbols in 1,000 (first choice) received symbols. Of this 125 error symbols in the first choice, approximately 84 of them (1-0.16 = 0.84) will have correct second choices. Therefore, only about 41 error symbols will remain after VSD finished with picking the correct symbols from the input list. As a result, using list-of-2 Viterbi instead of the original Viterbi as the inner decoder for VSD will help reduce the symbol error probability from 0.125 to 0.041 after the first simple VSD decoding stage.

When $E_b/N_0 \geq 8$ dB, *p1* was low enough (= 0 for these tests) that the second choice was no longer necessary. Note that *p2* had no meaning when *p1* = 0 and its values were shown as 0 in Table I and II for convenience in these cases.

Due to the interest in the high value range of decoding failure probability, only relatively small number of trials was needed. For C++ simulations, 1000 trials were used. This gave almost the same results as the Hardware test of 400 trials. Fig. 9 and 10 clearly demonstrate the closeness of the results from C++ and hardware for *p1* and *p2* respectively.

TABLE I
SIMULATION RESULT FROM C++ PROGRAM USING 1,000 TRIALS

E_b/N_0 (dB)	#of trials that first choice is wrong	#of trials that second choice is wrong	P1	P2
0	950	918	0.95	0.9663
1	889	816	0.889	0.9179
2	730	631	0.73	0.8644
3	523	338	0.523	0.6463
4	308	57	0.308	0.1851
5	125	20	0.125	0.16
6	30	4	0.03	0.1333
7	11	0	0.011	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0

TABLE II
HARDWARE TEST RESULT WITH 400 TRIALS

SNR	#of trials that first choice is wrong	#of trials that second choice is wrong	P1	P2
0	379	366	0.9475	0.9657
1	358	325	0.895	0.9078
2	298	252	0.745	0.8456
3	213	135	0.5325	0.6338
4	121	24	0.3025	0.1983
5	51	8	0.1275	0.1569
6	15	2	0.0375	0.1333
7	5	0	0.0125	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0

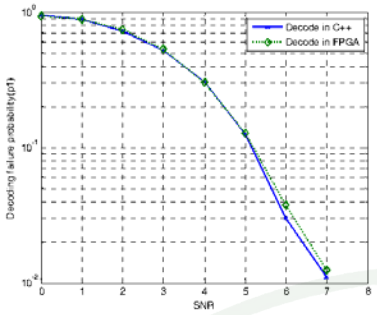


Figure 9. The decoding failure probability of the first choice (p1) from FPGA board and from C++

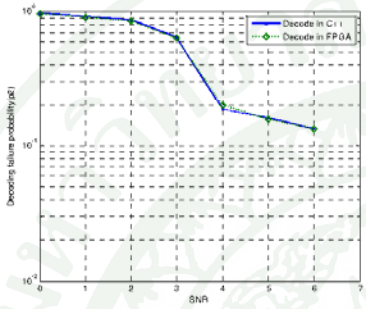


Figure 10. The decoding failure probability of the second choice given that the first choice is wrong (p2) from FPGA board and from C++

VI. Discussions

The performance in terms of decoding failure probabilities have been compared between the results from the lab prototype on FPGA board and the results from C++ program. They are virtually the same for the AWGN channel. The slight difference was from the fact that the number of trials for the lab prototype was smaller than that from the C++ program.

The numbers of trials were quite low because the range of interest for the decoding failure probability, *p1*, is quite high (0.1-0.01). Thus, there is no need to run more trials. This range of probability is higher than the usual range of interest ($<10^{-3}$) because LVA is only the inner decoder in the LVA-VSD concatenated coding system. Therefore, LVA is not required to correct many errors. The remaining errors will be corrected by the outer decoder. The overall decoding failure probability will then be in the usual range of interest.

For the list-of-2 decoding to be helpful to the outer decoder, the probability that the second choice is correct when the first choice was wrong (*1-p2*) should be high. It was seen from table I and II that *1- p2* > 0.8 with $E_b/N_0 \geq 4$ dB.

For $E_b/N_0 < 4$ dB, *p1* is approximately higher than 0.5. This means more than half of the input (first choice) symbols to VSD would be wrong. For this extremely high input error probability, it is no longer practical to try to correct errors.

VII. Conclusions

The LVA-VSD concatenated coding system was explicitly explained in this paper as an application of the list-of-2 Viterbi decoder on FPGA platform. Due to the low value of *p2* in the range of interest for *p1*, using list-of-2 Viterbi decoder will be very helpful for VSD in AWGN channel.

The same performance results between the hardware test and the simulation in C++ lead to two more useful conclusions. First, this confirmed that the LVA prototype worked properly as designed. Therefore, this LVA prototype on FPGA board can be a hardware component of the LVA-VSD concatenated coding system. Second, the modified C++ program was also confirmed to work correctly. This program can then be used to gain more insights for other channel conditions that the LVA-VSD concatenated coding system is preferred over other coding system structure.

Acknowledgment

The authors thank Mr. Auttaphud Seubnaung for writing the "USB Transceiver for List Viterbi V1.1 with DEBUG protocol" program.

References

[1] N. Seshadri and C. W. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Trans. on Communications*, vol.42, pp.313-323, Feb/Mar/Apr 1994.

[2] P. Elias, List decoding for noisy channels MIT Res. Lab. Electron., Cambridge, MA, Sep. 20, 1957.

[3] J. M. Wozencraft, List decoding MIT Res. Lab. Electron., Cambridge, MA, Jan. 15, 1958.

[4] S. Lin and D. J. Constello, Jr., *Error Control Coding*, 2nd edition, 2004, Pearson Prentice Hall.

[5] C. Bai, B. Mielczarek, W A. Krzymien and LJ. fair, "Improved analysis of list decoding and its application to convolutional codes and Turbo codes," *IEEE Trans. On Information Theory*, vol. 53, no. 2, pp. 615-627, Feb 2007.

[6] L. Lijofi, D. Cooper and B. Canpolat, "A reduced complexity list single-wrong-turn (SWT) Viterbi decoding algorithm," Proceedings of the 15th IEEE Inter. Sym. on Personal, Indoor and Mobile Radio Communications, 2004 (PIMRC 2004), vol. 1, pp. 274 – 279, 5-8 Sept. 2004.

[7] J.J. Metzner, "Vector Symbol Decoding with list inner symbol decision," *IEEE Trans. on Communications*, vol.51, no. 3, pp.371-380, Mar 2003.

[8] U. Tuntoolavest and J.J. Metzner, "Vector symbol decoding with list symbol decisions and outer convolutional codes for reliable communications," *Integrated Computer-Aided Engineering Journal*, vol. 9, no. 2, IOS Press, 2002, p. 101-116.

[9] U. Tuntoolavest, *Vector Symbol Decoding for Wireless Fading Channels*, VDM publishing, 184 pages, 2009, ISBN-13: 978-3639132861 (www.amazon.com)

[10] Y. Zhang and W.E. Ryan, "Toward low LDPC-code floors: a case study." *IEEE Trans. on Communications*, vol.57, no. 6, pp.1566-1573, June 2009

[11] U. Tuntoolavest and P. Noradee, "Design and Implementation of List-of-2 Viterbi Decoder with VHDL and its Application," *ECTI-CON 2009*, vol. 2, pp. 946-949, May 6-9, 2009, Pattaya, Chonburi, Thailand

[12] http://www.ftdichip.com/Drivers/D2XX.htm

Suitable Mobile Channel Conditions for a Concatenated Coding System with List-of-2 Viterbi Inner Decoder

Usana Tuntoolavest* and Pongpisut Noradee
Department of Electrical Engineering
Kasetsart University, Bangkok, Thailand
E-mail: fengunt@ku.ac.th Tel: +66-2-9428555 ext 1565

***Abstract*— List-of-2 Viterbi decoder provides two decoded outputs for each input sequence. The two outputs are the one with the highest likelihood value and the one with the second highest likelihood value. Previous work proposed this decoder as the inner decoder of a concatenated coding system with Vector Symbol Decoder (VSD) as the outer decoder. Since both outputs of the inner decoder are to be input to VSD, the error statistics of both outputs are of interest. This paper investigates the error statistics in mobile channels using the models of Rayleigh fading and Ricean fading channels with Doppler shifts. Results show that this inner decoder provides useful output error statistics for VSD for some particular mobile channel conditions. By selecting appropriate criteria for these statistics, the suitable SNR ranges can be concluded for this concatenated coding system for various channel models. For example, for Rayleigh fading channel with the speed of 120 km/h, the SNR range of interest is 8.1-9.8 dB. For Ricean fading channel with the speed of 60 km/h and Ricean factor of 10, this range is 6.4-8.1 dB. This information is useful for the application of this coding system to mobile communications.**

I. INTRODUCTION

Error control coding is a widely used method to ensure reliable data communications. Concatenated codes are often chosen for wireless and mobile applications due to their ease of implementation. These codes were first proposed by Forney [1] in 1966. A concatenated code usually consists of an inner code and an outer code. In this paper, a concatenated code with a convolutional inner code and a convolutional outer code is considered. The inner decoder uses the soft decision list-of-2 Viterbi algorithm. The outer decoder uses the Vector Symbol Decoding (VSD) technique.

The list-of-2 Viterbi algorithm (VA) is a special case of List Viterbi Algorithm (LVA) proposed by Seshadri and Sundberg [2]. LVA is the Viterbi algorithm that provides more than one output in the order of their likelihood values. Therefore, list-of-2 VA provides two outputs. In this paper, the one with highest likelihood value will be called "the most likely decoded sequence" or "the first choice" and the one with the second highest likelihood value will be called "the second most likely decoded sequence" or "the second choice".

Supported by the Kasetsart University Research and Development Institute

VSD is a decoding technique for any linear codes with large nonbinary symbols. It was first proposed by Metzner and Kapturowski in [3]. Haslach and Han Vinck also presented a similar idea in [4] calling it "a decoding algorithm for array codes". This decoding technique uses verification-based method. Instead of finding error symbols, it verifies the correct symbols. The symbols that cannot be verified to be correct by the algorithm are then labeled as error symbols. The error values can then be found if certain conditions are met. At first, VSD was mostly applied for nonbinary block codes. The modification of VSD for nonbinary convolutional codes was shown in [5]. Since VSD is a nonbinary decoding technique, it can be applied as an outer decoder of a concatenated code. Moreover, it is suitable to use with a list inner decoder such as LVA. This is because for many error cases, VSD can easily pick the correct choice from the list of more than one alternative choice provided by the list inner decoder [6]. The detail of VSD can be found in [7].

The design and implementation of list-of-2 Viterbi decoder in a Field Programmable Gate Array (FPGA) board was presented in [8] and [9]. The structure of the LVA-VSD concatenated coding system was explained in detail in [9]. The simulation result with one set of Doppler shift and SNR value for Rayleigh fading channel was shown in [8] to verify the function of the FPGA board. The performance of list-of-2 VA in AWGN channel with Binary Frequency Shift Keying (BFSK) modulation was shown in [9].

In this paper, the performance of list-of-2 VA was investigated more thoroughly and more systematically for various mobile channel models. This is so that the resulting error statistics can lead to suitable SNR range for each channel condition. The channel models are Rayleigh fading and Ricean fading channels with three different Doppler shifts correspond to the velocities of 60, 90 and 120 km/h. For Ricean fading channels, three different Ricean factors are also considered for each velocity. The channel models were written with MATLAB based on the m-files shown in [10]. Using the m-files provided us more understanding to the models than using the built-in functions of MATLAB. For Rayleigh fading channel with Doppler effect, [10] implements the modified sum-of-sinusoids method proposed by [11]. This

model has been shown in [11] to be a good approximation to Jakes [12] and Clarke's models [13].

The output error statistics of list-of-2 VA was analyzed in terms of the probability of decoding failure of the most likely and the second most likely decoded sequences. By selecting appropriate criteria for these error statistics, the suitable SNR ranges for the LVA-VSD system can be concluded for various mobile channel conditions. These results are useful for the application of LVA-VSD system in mobile communications.

This paper is organized as the followings. Section II covers the background including the system overview and mobile channels. Section III explains the method and how we simulated the channel models as well as the clarification of suitable channel conditions. Section IV shows the simulation results and the analyzed results. Section V and VI discusses and concludes the results.

II. BACKGROUND

A. System Overview

The concatenated code shown in fig.1 consists of a (3,2,2) nonbinary convoltuional outer code and a (2,1,4) binary convolutional inner code. The modulation technique is Binary Phase Shift Keying (BPSK). The channels in consideration are the Rayleigh fading channel and Ricean fading channel with Doppler shifts. The inner decoder is the soft-decision list-of-2 VA. The outer decoder is VSD with 32-bit symbols. Note that the simulations in this paper are done for the block diagram in the dashed box in fig. 1 only. The outer code and the outer decoder are shown to provide suitable design parameters for the inner encoder and the inner decoder.

Specifically, the (2,1,4) inner code has $G(D) = [(1+D+D^4)\ (1+D+D^3+D^4)]$. The input data sequence is terminated every 32 bit since the outer encoder use 32-bit symbols. Each inner encoded sequence contains 72 bits $((32 + 4)\cdot 2 = 72)$. The output of the matched filter provides two correlation metrics for each received bit for a total of 144 metrics for each input sequence. The list-of-2 VA used these input metrics to find the best and second best paths and output these two paths in order of likelihood to the outer decoder. Therefore, the outer decoder received two 32-bit decoded sequences from the inner decoder for each encoded inner sequence. Note that the algorithm of lists-of-2 VA can be found in [2]. An example

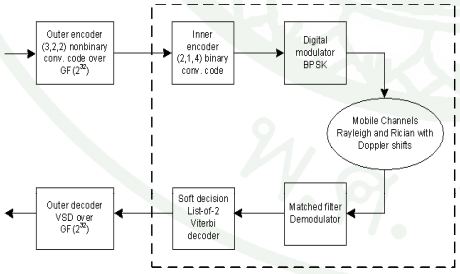


Fig. 1 Concatenated coding system in consideration

has also been shown in [8]. Basically, the list-of-2 VA used the same decoding concept as the conventional VA except that it will keep two highest path metrics instead of only one highest path metric at each state and each time interval. This is so that it can keep track of both the best and second best paths.

B. Mobile Channels

The mobile radio channels are often modeled as a Rayleigh fading channel or a Ricean fading channel. The Rayleigh fading is useful since it is the worst case scenario. For this scenario, there are only a large number of reflective paths, but there is no dominant signal path, i.e., no line-of-sight (LOS) component. The received envelope of the flat fading in this case follows Rayleigh distribution. The Rayleigh probability density function (pdf) can be expressed as [14]

$$p_{ray}(r) = \frac{r}{\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right) \quad ,(0 \le r \le \infty) \tag{1}$$

And $p_{ray}(r) = 0$ when $r < 0$ since the envelope must have a positive value.

If there is a dominant LOS in addition to the large number of reflective paths, the received envelope in this case is modeled based on Ricean distribution. The Ricean pdf can be expressed as [14]

$$p_{rice}(r) = \frac{r}{\sigma^2}\exp\left(-\frac{r^2+A^2}{2\sigma^2}\right)\cdot I_0\left(\frac{rA}{\sigma^2}\right) \quad ,(0 \le r \le \infty) \tag{2}$$

Where $I_0(x)$ is the modified Bessel function of the first kind and zero order.

The Ricean factor K is the ratio of the LOS power to the diffuse component power, which can be shown as [14]

$$K = A^2/2\sigma_0^2 \tag{3}$$

The Ricean factor K is important because by varying K, we can obtain different fading effects. For the case that $K = 0$, the Ricean fading channel is the same as a Rayleigh fading channel since $K = 0$ means no LOS. For the case that $K = \infty$, the Ricean fading channel is the same as an AWGN channel since the LOS is very strong compared to other paths.

In addition to the fading envelope, Doppler effect is also included in the mobile channel models in consideration. The Doppler effect is the phenomenon that the received frequency is shifted from the transmitted frequency due to the relative movement between the transmitter and the receiver. Assuming that the direction of the receiver movement aligned with the direction of wave propagation, the maximum Doppler shift is [14]

$$\upsilon_{\max} = f_c \cdot \frac{v}{c_0} \tag{4}$$

Where f_c = carrier frequency in Hz

v = mobile station velocity (assuming the transmitter is stationary)

$c_0 = 3 \cdot 10^8$ m/s (the speed of light)

This maximum Doppler frequency shift is the one used in the simulations.

III. METHOD

The system is the same as described in section II. From the block diagram in fig. 1, MATLAB was used to simulate the (2,1,4) convolutional inner encoder, the BPSK modulator, the channels and the matched filter demodulator. The soft decision list-of-2 VA in C++ program written for previous work [8] was used for the simulation of inner decoder. Since this paper did not use the built-in function of MATLAB for channel models, this part is explained in more details in subsection A. In addition, the meaning of suitable channel conditions is clarified in subsection B.

A. Channel Model Simulation

The simulations of channel models were done with MATLAB based on the m-files shown in [10]. For Rayleigh fading channel, the fading envelope can be generated by the sum of two quadrature Gaussian random processes [16]. That is

$$r_{i,ray} = \sqrt{x_i^2 + y_i^2} \tag{5}$$

Where x_i and y_i are samples of the zero-mean stationary Gaussian random processes, each with unit variance.

For Ricean fading channel, the fading envelope $r_{i,rician}$ can be generated from [10]

$$r_{i,ricean} = \sqrt{(x_i + A)^2 + y_i^2} \tag{6}$$

Where x_i and y_i are samples of the zero-mean stationary Gaussian random processes, each with variance σ_0^2 and A^2 is the power of the dominant path.

For the Rayleigh fading with Doppler effect, the quadrature components of Rayleigh fading process are [10]

$$x(t) = \sqrt{\frac{2}{M}} \sum_{n=1}^{M} \cos\left(\omega_d t \cos\alpha_n + \phi_n\right) \tag{7}$$

$$y(t) = \sqrt{\frac{2}{M}} \sum_{n=1}^{M} \cos\left(\omega_d t \sin\alpha_n + \varphi_n\right) \tag{8}$$

with

$$\alpha_n = \frac{2\pi n - \pi + \theta_n}{4M}, \quad n = 1, 2, 3, \ldots, M \tag{9}$$

Where ω_d is the maximum angular Doppler frequency, ϕ_n , φ_n and θ_n are statistically independent and uniformly distributed random variable over $[-\pi, \pi)$ for all n.

Using eq. (5) with the value of x_i and y_i from the samples of eq. (7), (8), we can obtain the fading amplitude of Rayleigh channel with Doppler shift. Note that the selected carrier frequency was 2.101 GHz. The sampling frequency was 8.404 GHz. The Doppler shift was computed from eq. (4) for the velocities of 60, 90 and 120 km/h.

For the Ricean fading channel with Doppler shift, the m-file was modified to include the "A" term as shown in eq. (6) using the same x_i and y_i as in the Rayleigh fading channel.

B. Suitable Channel Conditions

Channel conditions here refer to the type of mobile channel model (Rayleigh or Ricean), the corresponding velocity, additional factor (Ricean factor) and the SNR values. The suitable conditions are the ones that list-of-2 VA as the inner decoder provides suitable output error statistics to VSD, the outer decoder. Since other conditions are quite standard, this paper focuses on whether there exists any suitable SNR range for these standard conditions.

There are two criteria for a suitable range of SNR. The first one is that it allows enough error symbols to the outer code such that the use of the outer code is justified. The second one is that there must be enough cases that the second output of LVA is correct when the first output is wrong. This first criterion is due to the fact that a good quality channel with high SNR needs only a simple one level code. There is no need to use the concatenated code with two code levels. The second criterion is due to if the second LVA output is almost always wrong when the first LVA output is wrong, there is not enough benefit to justify using the more complex LVA instead of the conventional VA.

IV. RESULTS

The performance of list-of-2 VA was investigated in several channel conditions. The first set is the Rayleigh fading channel that is corrupted by AWGN. This channel was simulated for three velocities: 60, 90 and 120 km/h. The second set is the Ricean fading channel that is also corrupted by AWGN. For Ricean fading channel, each of the three velocities was also simulated for three values of Ricean factor K = 1, 3 and 10. The last channel is the AWGN channel with no fading. This channel is shown for comparison purpose.

As previously discussed, the list-of-2 Viterbi decoder provides two outputs. Both outputs are to be the inputs of VSD, the outer decoder. From the viewpoint of VSD, the LVA output with the highest likelihood is "the first choice" since it is more likely to be correct and should be selected first. The LVA output with the second highest likelihood is "the second choice" since it is less likely to be correct. VSD will consider the use of second choice only when the syndrome computed from the first choice symbols is wrong.

The error statistics p_1 and p_2 of these two LVA outputs are of interest. p_1 is the probability that the first choice is wrong. p_2 is the probability that the second choice is wrong given that the first choice is wrong. Consequently, the probability that the second choice is correct given that the first choice is wrong ($p_{2correct}$) equals to 1-p_2.

Table I shows the simulation results of the output error statistics of list-of-2 Viterbi decoder in an AWGN channel. Table II shows the simulation results of the output error statistics of list-of-2 Viterbi decoder in the Rayleigh fading channel for three velocities. To visualize the comparison, p_1 for this case is also shown as a graph in fig. 2 as well as in Table II. However, p_2 is only shown in tables to save space and because it is not necessary to compare p_2 of different cases. The value of p_2 will only be analyzed to gain insight on the usefulness of the second choice to the outer decoder.

TABLE I
OUTPUT ERROR STATISTICS OF LVA IN AWGN CHANNEL

AWGN		
SNR(dB)	p_1	p_2
0	0.5234	0.7902
1	0.2758	0.6508
2	0.1052	0.5218
3	0.0293	0.3481
4	0.0040	0.2250
5	0.0010	0.1000
6	$< 10^{-4}$	N/A

TABLE II
OUTPUT ERROR STATISTICS OF LVA IN RAYLEIGH FADING CHANNEL

Rayleigh fading channel						
	V = 60 km/h		V = 90 km/h		V = 120 km/h	
SNR(dB)	p_1	p_2	p_1	p_2	p_1	p_2
0	0.9972	0.9965	0.9982	0.9980	0.9983	0.9993
2	0.9661	0.9822	0.9761	0.9884	0.9837	0.9917
4	0.8016	0.9147	0.8464	0.9337	0.8840	0.9498
6	0.3717	0.7111	0.4533	0.7498	0.5282	0.7995
8	0.0545	0.3982	0.0808	0.4245	0.1215	0.4824
10	0.0018	0.1538	0.0034	0.1686	0.0067	0.2066

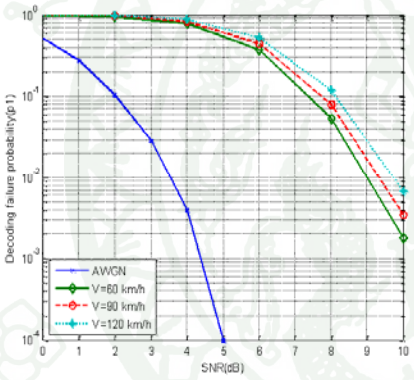


Fig. 2 Probability that the first choice is wrong (p_1) in AWGN channel and in Rayleigh fading channel at 60, 90 and 120 km/h

Table III shows the simulation results of the output error statistics of list-of-2 VA in Ricean fading with AWGN channel for three values of Ricean factors K = 1, 3 and 10. Fig. 3 shows the comparison of p_1 for Rayleigh and Ricean fading channels at 60 km/h for three values of K. Note that the performance of the AWGN channel and the Rayliegh fading with the velocity of 60 km/h are shown in both fig. 2 and 3 for the ease of comparison. Although AWGN is not a mobile channel model, its performance is also shown as the baseline performance.

TABLE III
OUTPUT ERROR STATISTICS OF LVA IN RICEAN FADING CHANNEL

A) WITH RICEAN FACTOR, K = 1

Ricean fading channel						
	V = 60 km/h		V = 90 km/h		V = 120 km/h	
SNR(dB)	p_1	p_2	p_1	p_2	p_1	p_2
0	0.9968	0.9969	0.9980	0.9978	0.9979	0.9993
2	0.9629	0.9801	0.9734	0.9884	0.9821	0.9902
4	0.7912	0.9055	0.8345	0.9204	0.8783	0.9453
6	0.3528	0.7058	0.4277	0.7505	0.5093	0.7824
8	0.0481	0.3971	0.0750	0.4760	0.1107	0.5122
10	0.0013	0.1538	0.0027	0.1544	0.0058	0.1828

B) WITH RICEAN FACTOR, K = 3

Ricean fading channel						
	V = 60 km/h		V = 90 km/h		V = 120 km/h	
SNR(dB)	p_1	p_2	p_1	p_2	p_1	p_2
0	0.996	0.9955	0.9963	0.9988	0.9983	0.9985
2	0.9525	0.9783	0.9694	0.9846	0.9804	0.9880
4	0.7557	0.8965	0.8115	0.9234	0.8555	0.9324
6	0.3026	0.6821	0.3921	0.7279	0.4839	0.7671
8	0.0355	0.3831	0.0605	0.4231	0.0939	0.4473
10	0.0008	0.1000	0.0019	0.1237	0.0043	0.1713

C) WITH RICEAN FACTOR, K = 10

Ricean fading with AWGN channel						
	V = 60 km/h		V = 90 km/h		V = 120 km/h	
SNR(dB)	p_1	p_2	p_1	p_2	p_1	p_2
0	0.9902	0.9922	0.9930	0.9961	0.9961	0.9968
2	0.9159	0.9626	0.9441	0.9761	0.9586	0.9790
4	0.6289	0.8410	0.7160	0.8723	0.7697	0.9002
6	0.1796	0.5824	0.2665	0.6319	0.3239	0.6919
8	0.0137	0.2700	0.0272	0.3382	0.0412	0.3422
10	0.0002	$<10^{-4}$	0.0004	0.0526	0.0012	0.1724

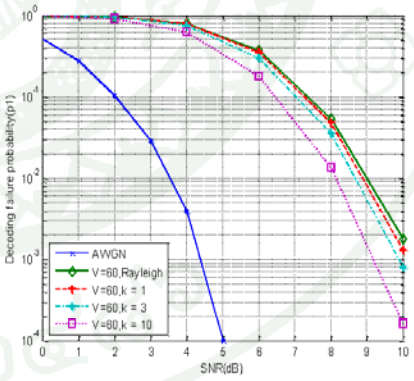


Fig. 3 Probability that the first choice is wrong (p_1) in AWGN channel and at 60 km/h in Rayleigh channel and Ricean fading channel with K = 1, 3 and10

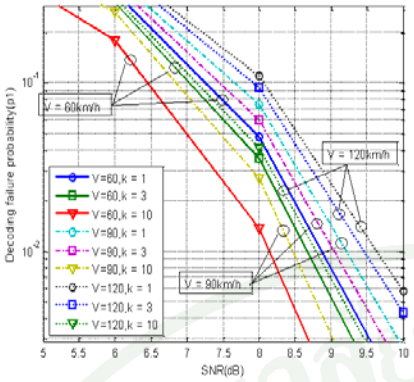


Fig. 4 Probability that the first choice is wrong (p_1) in Ricean fading channel with K = 1, 3 and 10 and at 60, 90 and 120 km/h

TABLE IV
SNR RANGE OF INTEREST FOR ALL CHANNEL CONDITIONS

Channel	Velocity (km/h)	Ricean factor (K)	SNR (dB)	$P_{2correct}$
AWGN	-	-	2 - 3.5	0.48 - 0.72
Rayleigh fading	60	-	7.2 - 9	0.47 - 0.72
Rayleigh fading	90	-	7.8 - 9.4	0.54 - 0.75
Rayleigh fading	120	-	8.1 - 9.8	0.52 - 0.77
Ricean fading	60	1	7.2 - 8.1	0.35 - 0.61
Ricean fading	60	3	7 - 8.7	0.46 - 0.70
Ricean fading	60	10	6.4 - 8.1	0.66 - 0.74
Ricean fading	90	1	7.6 - 9.2	0.47 - 0.72
Ricean fading	90	3	7.5 - 9	0.49 - 0.72
Ricean fading	90	10	6.9 - 8.5	0.50 - 0.75
Ricean fading	120	1	8 - 9.6	0.49 - 0.76
Ricean fading	120	3	7.9 - 9.4	0.58 - 0.77
Ricean fading	120	10	7.1 - 8.8	0.52 - 0.74

Fig. 4 shows p_1 for all conditions of Ricean fading channels. The graph is zoomed in so that all nine cases can be distinguished. Table IV shows the SNR ranges of interest for all selected channel conditions. These SNR ranges are analyzed and concluded from the results of table I, II, III and fig. 2, 3 and 4.

V. DISCUSSION

The results in table IV are analyzed from the design criteria that the probability that the most likely decoded sequence is wrong (p_1) is approximately between 0.01-0.1. This rather high range of p_1 is selected as the designed criteria for the concatenated coding system in consideration because it is the decoding error probability of the inner decoder. The remaining 1% to 10% error symbols will be input to VSD, the outer decoder, which will correct all or almost all of them.

For higher range of p_1 (>0.1), the channels are of very low quality and the number of error symbols are too high for practical applications. For the lower range of p_1 (such as p_1 around 10^{-3} to 10^{-2}), the number of error symbols are less than 1% and the benefit of using list inner decoder is less obvious although the outer decoder is still required. For $p_1 < 10^{-3}$, the channel is probably good enough for many applications that the conventional Viterbi decoder can be used by itself. In other words, no concatenated code is needed. For mobile channels, however, the channel is of quite low quality and usually is not good enough for this last case.

Next, consider the benefit of using list-of-2 VA to obtain the second choice. For the second choice to be useful to VSD, it has to be correct when the first choice is wrong. This correct second choice will help simplify and improve the performance of VSD, the outer decoder. As previously explained, the probability that the second choice is correct given that the first choice is wrong $p_{2correct} = 1-p_2$. Table IV shows that $p_{2correct}$ is from at least 35% to 77% and the average $p_{2correct}$ is more than 50%. This means that when the first choice is wrong, the second choice will be correct for more than 50% most of the times. This strengthens the criteria used in selecting the suitable SNR ranges.

These selected SNR ranges result in significant benefit to VSD, the outer decoder. Note that the LVA-VSD system can be employed for lower and higher SNR ranges but the list-of-2 VA will provide less benefit over conventional VA. Since the list-of-2 VA is more complex than the conventional VA, the complexity will need to be taken into account with the performance improvement for other ranges of SNR.

VI. CONCLUSIONS

The paper presents suitable SNR ranges of various mobile channel conditions that the list-of-2 Viterbi decoder provides good probability of correct second choice to the outer decoder. These results will help with the design of the LVA-VSD concatenated coding system for mobile applications.

ACKNOWLEDGMENT

This work is supported by the Kasetsart University Research and Development Institute. The Authors would also like to thank NTC Telecommunication Research Laboratory at Department of Electrical Engineering, Kasetsart University for its facility.

REFERENCES

[1] G.D. Forney, Jr., *Concatenated Codes*, MIT Press, Cambridge Mass., 1966.
[2] N. Seshadri and C-E. W. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Transaction on Communications*, vol. 42, pp. 313-323, Feb./Mar./Apr. 1994
[3] J.J. Metzner and E.J. Kapturowski, "A general decoding technique applicable to replicated file disagreement location and concatenated code decoding," *IEEE Transaction on Information Theory*, vol.36, pp.911-917, July 1990
[4] C. Haslach and A.J. Han Vinck, "A decoding algorithm with restrictions for array codes," *IEEE Transaction on Information Theory*, vol. 45, no. 7, pp. 2339-2344, Nov. 1999.

[5] U. Tuntoolavest and J.J. Metzner, "Vector symbol decoding with list symbol decisions and outer convolutional codes for reliable communications," *Integrated Computer-Aided Engineering Journal*, vol. 9, no. 2, 2002, p. 101-116, IOS Press, ISBN 1069-2509.

[6] J.J. Metzner, "Vector Symbol Decoding with list inner symbol decision," *IEEE Transactions on Communications*, vol.51, no. 3, pp.371-380, Mar 2003.

[7] U. Tuntoolavest, *Vector Symbol Decoding for Wireless Fading Channels*, VDM publishing, 2009, ISBN-13: 978-3639132861 (www.amazon.com)

[8] U. Tuntoolavest and P. Noradee, "Design and Implementation of List-of-2 Viterbi Decoder with VHDL and its Application," *Proceedings of the 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology (ECTI-CON 2009)*, vol. 2, pp. 946-949, May 6-9, 2009, Pattaya, Chonburi, Thailand.

[9] U. Tuntoolavest and P. Noradee, "Lab Prototype of a List-of-2 Viterbi Decoder: A Diversity Inner Decoder for the Outer Vector Symbol Decoder," *Proceedings of ECTI-CON 2010*, in press.

[10] N. Kostov, "Mobile Radio Channel Modeling in MATLAB," *Radioengineering*, vol. 12, no. 4, pp. 12-16, Dec 2003. (www.google.com)

[11] Y.R. Zheng and C. Xiao, "Improved Models for theGeneration of Multiple Uncorrelated Rayleigh Fading Waveforms," *IEEE Communications Letters*, vol. 6, no. 6, June, 2002.

[12] W.C. Jakes, *Microwave Mobile Communications*, Wiley, 1974, reprinted by IEEE Press in 1994.

[13] R.H. Clarke, "A statistical theory of mobile-radio reception," *Bell Systems Technical Journal*, vol. 47, pp. 957-1000, July-Aug. 1968

[14] A.F. Molisch, *Wireless Communications*, John Wiley & Sons, England, 2005.

[15] T.S. Rappaport, *Wireless Communications: Principles and Practice, second edition*, Prentice Hall, New Jersey, 2002.

ภาคผนวก ข

ข้อมูลขาที่ใช้บนบอร์ด FPGA

ตารางผนวกที่ ข1 รายละเอียดขา On-board clock sources ที่ต่ออยู่กับขา FPGA

Signal name	FPGA Pin
CLK_100MHZ	E18
USB_IFCLK	E10
CLK_SYNTH0_P/N	AB10, AB9
CLK_SYNTH1_P/N	AC23, AC22
GMII_RX_CLK	AC8
GMII_TX_CLK	AC17
GBE_MCLK	AD8

ตารางผนวกที่ ข2 รายละเอียดขา User clock ที่ต่ออยู่กับขา FPGA

Signal name	FPGA Pin
CLK_SOCKET	E16

ตารางผนวกที่ ข3 รายละเอียดขา Push button ที่ต่ออยู่กับขา FPGA

Signal name	FPGA Pin
SWITCH_PB1	B1
SWITCH_PB2	B2
SWITCH_PB3	E8
SWITCH_PB4	F17

ตารางผนวกที่ ข4 รายละเอียดขา DIP switch ที่ต่ออยู่กับขา FPGA

Signal name	FPGA Pin
SWITCH0	B26
SWITCH1	C26
SWITCH2	D26

ตารางผนวกที่ ข4 (ต่อ)

Signal name	FPGA Pin
SWITCH3	D25

ตารางผนวกที่ ข5 รายละเอียดขา EXP JX1 ที่ต่ออยู่กับขา FPGA

EXP Pin	Signal name	FPGA Pin
1	EXP1_SE_IO_1	D14
2	EXP1_SE_IO_0	G4
3	EXP1_SE_IO_3	G5
4	EXP1_SE_IO_2	H7
5	2.5V	-
6	2.5V	-
7	EXP1_SE_IO_5	H4
8	EXP1_SE_IO_4	H6
9	EXP1_SE_IO_7	J5
10	EXP1_SE_IO_6	J4
11	2.5V	-
12	2.5V	-
13	EXP1_SE_IO_9	J6
14	EXP1_SE_IO_8	K7
15	EXP1_SE_IO_11	K6
16	EXP1_SE_IO_10	L4
17	2.5V	-
18	2.5V	-
19	EXP1_SE_IO_13	L3
20	EXP1_SE_IO_12	L7
21	EXP1_SE_IO_15	M6
22	EXP1_SE_IO_14	M5

ตารางผนวกที่ ข5 (ต่อ)

EXP Pin	Signal name	FPGA Pin
23	2.5V	-
24	2.5V	-
25	EXP1_SE_IO_17	M1
26	EXP1_SE_IO_16	M2
27	EXP1_SE_IO_19	M4
28	EXP1_SE_IO_18	M7
29	2.5V	-
30	2.5V	-
31	EXP1_SE_IO_21	N7
32	EXP1_SE_IO_20	N4
33	EXP1_SE_IO_23	N1
34	EXP1_SE_IO_22	N6
35	2.5V	-
36	2.5V	-
37	EXP1_SE_IO_25	N2
38	EXP1_SE_IO_24	P5
39	EXP1_SE_IO_27	N3
40	EXP1_SE_IO_26	P1
41	EXP1_SE_IO_28	P3
42	EXP1_DIFF_CLK_IN_p	E12
43	EXP1_SE_CLK_IN	D13
44	EXP1_DIFF_CLK_IN_n	F12
45	GND	-
46	GND	-
47	EXP1_SE_IO_29	R1
48	EXP1_SE_IO_30	R3

ตารางผนวกที่ ข5 (ต่อ)

EXP Pin	Signal name	FPGA Pin
49	EXP1_SE_CLK_OUT	P4
50	EXP1_SE_IO_31	R2
51	GND	-
52	GND	-
53	EXP1_DIFF_p21	E6
54	EXP1_DIFF_p20	E2
55	EXP1_DIFF_n21	E5
56	EXP1_DIFF_n20	E1
57	GND	-
58	GND	-
59	EXP1_SE_IO_32	T2
60	EXP1_DIFF_p18	F5
61	EXP1_SE_IO_33	T3
62	EXP1_DIFF_n18	F4
63	GND	-
64	GND	-
65	EXP1_DIFF_p19	E7
66	EXP1_DIFF_p16	F3
67	EXP1_DIFF_n19	F7
68	EXP1_DIFF_n16	E3
69	GND	-
70	GND	-
71	EXP1_DIFF_p17	F2
72	EXP1_DIFF_CLK_OUT_p	Y1
73	EXP1_DIFF_n17	G2
74	EXP1_DIFF_CLK_OUT_n	W1
75	GND	-

ตารางผนวกที่ ข5 (ต่อ)

EXP Pin	Signal name	FPGA Pin
76	GND	-
77	EXP1_DIFF_p15	K3
78	EXP1_DIFF_p14	G1
79	EXP1_DIFF_n15	K2
80	EXP1_DIFF_n14	H1
81	EXP1_DIFF_p13	J1
82	EXP1_DIFF_p12	H3
83	EXP1_DIFF_n13	H2
84	EXP1_DIFF_n12	J3
85	3.3V	-
86	3.3V	-
87	EXP1_DIFF_p11	L2
88	EXP1_RCLK_DIFF_p10	K5
89	EXP1_DIFF_n11	K1
90	EXP1_RCLK_DIFF_n10	L5
91	3.3V	-
92	3.3V	-
93	EXP1_DIFF_p9	G6
94	EXP1_DIFF_p8	P6
95	EXP1_DIFF_n9	G7
96	EXP1_DIFF_n8	R7
97	3.3V	-
98	3.3V	-
99	EXP1_DIFF_p7	R6
100	EXP1_DIFF_p6	U2
101	EXP1_DIFF_n7	R5
102	EXP1_DIFF_n6	U1

ตารางผนวกที่ ข5 (ต่อ)

EXP Pin	Signal name	FPGA Pin
103	3.3V	-
104	3.3V	-
105	EXP1_DIFF_p5	AA2
106	EXP1_DIFF_p4	V2
107	EXP1_DIFF_n5	Y2
108	EXP1_DIFF_n4	V1
109	3.3V	-
110	3.3V	-
111	EXP1_DIFF_p3	AC2
112	EXP1_DIFF_p2	AB2
113	EXP1_DIFF_n3	AC1
114	EXP1_DIFF_n2	AB1
115	3.3V	-
116	3.3V	-
117	EXP1_DIFF_p1	AE1
118	EXP1_DIFF_p0	AF2
119	EXP1_DIFF_n1	AD1
120	EXP1_DIFF_n0	AE2
121	GND	-
122	GND	-
123	GND	-
124	GND	-
125	GND	-
126	GND	-
127	GND	-
128	GND	-
129	GND	-

ตารางผนวกที่ ข5 (ต่อ)

EXP Pin	Signal name	FPGA Pin
130	GND	-
131	GND	-
132	GND	-

ตารางผนวกที่ ข6 รายละเอียดขา EXP JX2 ที่ต่ออยู่กับขา FPGA

EXP Pin	Signal name	FPGA Pin
1	EXP2_SE_IO_1	E15
2	EXP2_SE_IO_0	H19
3	EXP2_SE_IO_3	G21
4	EXP2_SE_IO_2	H21
5	2.5V	-
6	2.5V	-
7	EXP2_SE_IO_5	G22
8	EXP2_SE_IO_4	H22
9	EXP2_SE_IO_7	H23
10	EXP2_SE_IO_6	J21
11	2.5V	-
12	2.5V	-
13	EXP2_SE_IO_9	J20
14	EXP2_SE_IO_8	J19
15	EXP2_SE_IO_11	J23
16	EXP2_SE_IO_10	K21
17	2.5V	-
18	2.5V	-
19	EXP2_SE_IO_13	K20

ตารางผนวกที่ ข6 (ต่อ)

EXP Pin	Signal name	FPGA Pin
20	EXP2_SE_IO_12	L20
21	EXP2_SE_IO_15	L19
22	EXP2_SE_IO_14	L23
23	2.5V	-
24	2.5V	-
25	EXP2_SE_IO_17	L22
26	EXP2_SE_IO_16	M21
27	EXP2_SE_IO_19	M22
28	EXP2_SE_IO_18	M25
29	2.5V	-
30	2.5V	-
31	EXP2_SE_IO_21	M24
32	EXP2_SE_IO_20	M20
33	EXP2_SE_IO_23	M26
34	EXP2_SE_IO_22	N22
35	2.5V	-
36	2.5V	-
37	EXP2_SE_IO_25	N23
38	EXP2_SE_IO_24	N21
39	EXP2_SE_IO_27	N24
40	EXP2_SE_IO_26	N19
41	EXP2_SE_IO_28	N26
42	EXP2_DIFF_CLK_IN_p	F14
43	EXP2_SE_CLK_IN	D15
44	EXP2_DIFF_CLK_IN_n	E13
45	GND	-

ตารางผนวกที่ ข6 (ต่อ)

EXP Pin	Signal name	FPGA Pin
46	GND	-
47	EXP2_SE_IO_29	P24
48	EXP2_SE_IO_30	P19
49	EXP2_SE_CLK_OUT	M19
50	EXP2_SE_IO_31	P23
51	GND	-
52	GND	-
53	EXP2_DIFF_p21	E25
54	EXP2_DIFF_p20	E22
55	EXP2_DIFF_n21	E26
56	EXP2_DIFF_n20	E23
57	GND	-
58	GND	-
59	EXP2_SE_IO_32	P25
60	EXP2_DIFF_p18	E21
61	EXP2_SE_IO_33	P26
62	EXP2_DIFF_n18	E20
63	GND	-
64	GND	-
65	EXP2_DIFF_p19	F24
66	EXP2_DIFF_p16	G24
67	EXP2_DIFF_n19	F25
68	EXP2_DIFF_n16	G25
69	GND	-
70	GND	-
71	EXP2_DIFF_p17	F22

ตารางผนวกที่ ข6 (ต่อ)

EXP Pin	Signal name	FPGA Pin
72	EXP2_DIFF_CLK_OUT_p	V26
73	EXP2_DIFF_n17	F23
74	EXP2_DIFF_CLK_OUT_n	U26
75	GND	-
76	GND	-
77	EXP2_DIFF_p15	H24
78	EXP2_DIFF_p14	G26
79	EXP2_DIFF_n15	J24
80	EXP2_DIFF_n14	H26
81	EXP2_DIFF_p13	J25
82	EXP2_DIFF_p12	G20
83	EXP2_DIFF_n13	J26
84	EXP2_DIFF_n12	F20
85	3.3V	-
86	3.3V	-
87	EXP2_DIFF_p11	K25
88	EXP2_RCLK_DIFF_p10	K23
89	EXP2_DIFF_n11	K26
90	EXP2_RCLK_DIFF_n10	K22
91	3.3V	-
92	3.3V	-
93	EXP2_DIFF_p9	L24
94	EXP2_DIFF_p8	P21
95	EXP2_DIFF_n9	L25
96	EXP2_DIFF_n8	P20
97	3.3V	-

ตารางผนวกที่ ข6 (ต่อ)

EXP Pin	Signal name	FPGA Pin
98	3.3V	-
99	EXP2_DIFF_p7	R22
100	EXP2_DIFF_p6	R25
101	EXP2_DIFF_n7	R23
102	EXP2_DIFF_n6	R26
103	3.3V	-
104	3.3V	-
105	EXP2_DIFF_p5	U24
106	EXP2_DIFF_p4	T24
107	EXP2_DIFF_n5	U25
108	EXP2_DIFF_n4	T25
109	3.3V	-
110	3.3V	-
111	EXP2_DIFF_p3	Y25
112	EXP2_DIFF_p2	W25
113	EXP2_DIFF_n3	Y26
114	EXP2_DIFF_n2	W26
115	3.3V	-
116	3.3V	-
117	EXP2_DIFF_p1	AC26
118	EXP2_DIFF_p0	AB25
119	EXP2_DIFF_n1	AB26
120	EXP2_DIFF_n0	AA25
121	GND	-
122	GND	-
123	GND	-

ตารางผนวกที่ ข6 (ต่อ)

EXP Pin	Signal name	FPGA Pin
124	GND	-
125	GND	-
126	GND	-
127	GND	-
128	GND	-
129	GND	-
130	GND	-
131	GND	-
132	GND	-

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นายพงษ์พิสุทธิ์ นรดี
วัน เดือน ปี ที่เกิด	25 มกราคม 2526
สถานที่เกิด	จังหวัดศรีสะเกษ
ประวัติการศึกษา	วศ.บ. (วิศวกรรมไฟฟ้า) เกียรตินิยมอันดับ 1 มหาวิทยาลัยรังสิต
ตำแหน่งหน้าที่การงานปัจจุบัน	นักศึกษาปริญญาโท มหาวิทยาลัยเกษตรศาสตร์ บางเขน
สถานที่ทำงานปัจจุบัน	อาคารวิศวกรรมศาสตร์ 60 ปี ชั้น 5 ห้อง 505/5
ผลงานดีเด่นและรางวัลทางวิชาการ	
ทุนการศึกษาที่ได้รับ	ทุนสนับสนุนคุณภาพงานวิจัยระดับบัณฑิตศึกษา มหาวิทยาลัยเกษตรศาสตร์ (2550)