ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)

ปริญญา

วิศวกรรมไฟฟ้า | วิศวกรรมไฟฟ้า

สาขา | **ภาควิชา**

เรื่อง ตัวต้นแบบระบบรหัสช่องสัญญาณแบบต่อร่วมที่ใช้การถอดรหัสแบบเวกเตอร์ซิมโบลบนซอฟท์คอร์โปรเซสเซอร์

Lab Prototype of Channel Concatenated Coding System with Vector Symbol Outer Decoder on Soft Core Processor

นามผู้วิจัย นายจตุพล ท่นไชย

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(รองศาสตราจารย์อุศนา ตัณฑุลเวศม์, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(ผู้ช่วยศาสตราจารย์เด่นชัย วรเศวต, Ph.D.)

หัวหน้าภาควิชา .....

(รองศาสตราจารย์วิชัย สุระพัฒน์, วศ.ม.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

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

วิทยานิพนธ์

เรื่อง

ตัวต้นแบบระบบรหัสช่องสัญญาณแบบต่อร่วมที่ใช้การถอดรหัสแบบเวกเตอร์ซิมโบลบนซอฟท์คอร์โปรเซสเซอร์

Lab Prototype of Channel Concatenated Coding System with Vector Symbol Outer Decoder on Soft Core Processor

โดย

นายจตุพล ท่นไชย

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)
พ.ศ. 2556

จตุพล ท่นไชย 2556: ตัวต้นแบบระบบบรหัสช่องสัญญาณแบบต่อร่วมที่ใช้การถอดรหัสแบบเวกเตอร์ซิมโบลด้วยซอฟท์คอร์โปรเซสเซอร์ ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า) สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: รองศาสตราจารย์อุศนา ตัณฑุลเวศม์, Ph.D. 88 หน้า

การต่อร่วมเป็นวิธีการในการเพิ่มความน่าเชื่อถือให้กับการสื่อสารข้อมูลด้วยความซับซ้อนที่ลดลง วิทยานิพนธ์นี้เสนอตัวต้นแบบระบบช่องสัญญาณแบบต่อร่วมที่ใช้การถอดรหัสแบบเวกเตอร์ซิมโบลบนซอฟท์คอร์โปรเซสเซอร์และได้มีการทดสอบบนช่องสัญญาณไร้สาย สำหรับซอฟท์คอร์โปรเซสเซอร์บนบอร์ดทดลอง FPGA ทำหน้าที่เป็นส่วนควบคุมเพื่อทำงานร่วมกับพอร์ตอีเทอร์เน็ตและพอร์ตอนุกรม RS-232 ที่ได้เชื่อมต่อกับอุปกรณ์รับส่งสัญญาณไร้สาย ตลอดจนทำหน้าที่ในการเข้ารหัสและถอดรหัส ระบบการเข้ารหัสแบบต่อร่วมประกอบไปด้วยการเข้าหัสภายนอกแบบคอนโวลูชัน (3,2,2) และการเข้ารหัสภายในแบบบีซีเฮช (31,26) หรือการเข้ารหัสภายในแบบรีดโซโลมอน (63,51) สำหรับระบบการถอดรหัสประกอบไปด้วยการถอดรหัสแบบลิสวีเทอร์บีสำหรับรหัสบีซีเฮชหรือการถอดรหัสด้วยวิธีการของ Berlekamp-Massey สำหรับรหัสแบบรีดโซโลมอนและการถอดรหัสภายนอกแบบเวกเตอร์ซิมโบล โดยทำการทดสอบการเข้ารหัสและถอดรหัสแต่ละส่วนแยกกันก่อนในขั้นต้น จากนั้นการทดสอบทั้งระบบจะเลือกใช้รหัสภายในแบบรีดโซโลมอน ซึ่งพบว่าทุกส่วนสามารถทำงานได้ตามการออกแบบ อย่างไรก็ตามได้ใช้เพียงกลุ่มข้อมูลชุดเล็ก ๆ สำหรับการส่งในช่วงการทดสอบและในการทดสอบการส่งสัญญาณไร้สายพบว่าต้องมีการหน่วงเวลา สาเหตุหนึ่งเนื่องจากเป็นการส่งแบบไม่ประสานเวลา ซึ่งในอนาคตควรมีการทดสอบการส่งข้อมูลที่มีความเร็วมากขึ้นกับอุปกรณ์ไร้สายที่มีความเร็วสูง การเลือกการเชื่อมต่อกับอุปกรณ์ที่ดีขึ้นเพื่อลดการหน่วงเวลาและเพิ่มการทดสอบในภาคสนาม

_____ _____ ___ / ___ / __

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Jatupon Thonchai 2013: Lab Prototype of Channel Concatenated Coding System with Vector Symbol Outer Decoder on Soft Core Processor. Master of Engineering (Electrical Engineering), Major Field: Electrical Engineering, Department of Electrical Engineering. Thesis Advisor: Associate Professor Usana Tuntoolvest, Ph.D.
88 pages.

Concatenation is the method to provide reliable data communications with less complexity. This thesis presents a lab prototype of channel concatenated coding system with Vector symbol outer decoding on soft core processor with the test in a wireless channel. The soft core processor on FPGA board was used to control the interface via Ethernet and RS-232 connection with the RF module and perform the encoding and decoding processes. The concatenated encoder system consisted of a (3,2,2) outer convolutional encoder and a (31,26) BCH code or (63,51) Reed Solomon inner encoder. The decoder system consisted of List-of-2 Viterbi algorithm for the BCH inner code and Berlekamp-Massey algorithm for the RS inner code and Vector symbol outer decoder. Each encoder and decoder component was tested separately first. Then, the complete system test was done for the RS inner code system. It was found that all components and the complete system work properly as designed. However, only a small set of symbols was transmitted at a time in the test and the wireless transmission test required delay caused in part by the asynchronous transmission. Future work should include faster data transfer with high speed wireless devices, better interface with peripheral to reduce delay time and field test.

_____ _____ ___ / ___ / __

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ผู้วิจัยขอกราบขอบพระคุณ รศ.ดร.อุศนา ตัณฑุลเวศม์ ประธานกรรมการที่ปรึกษาวิทยานิพนธ์ ผศ.ดร.เด่นชัย วรเศวต กรรมการที่ปรึกษาสาขาวิชารอง ที่ให้คำปรึกษาในการเรียนการค้นคว้าวิจัย ตลอดจนการตรวจแก้ไขวิทยานิพนธ์จนกระทั่งเสร็จสมบูรณ์ และ รศ.ดร.ลัญฉกร วุฒิสิทธิกุลกิจ จากภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัยที่ได้ให้ความกรุณาตรวจแก้ไขวิทยานิพนธ์ให้สมบูรณ์ยิ่งขึ้น

ขอกราบขอบพระคุณคณะอาจารย์จากภาควิชาวิศวกรรมไฟฟ้าที่มอบความรู้ความเข้าใจที่มีประโยชน์อย่างยิ่ง เพื่อนำมาใช้ให้เกิดประโยชน์ในด้านที่ดีเพื่อสังคมต่อไป และขอขอบคุณ คุณวศิน สุขตลอดชีพ และ คุณชโณทัย ไชยวรรณ์ เพื่อนร่วมห้องทดลองผู้ให้คำปรึกษาในงานวิจัยและให้คำแนะนำตลอดมา

ด้วยความดีหรือประโยชน์อันใดเนื่องจากวิทยานิพนธ์เล่มนี้ ขอมอบแด่คุณพ่อ คุณแม่ ที่ได้อบรมและให้กำลังใจผู้วิจัยมาตลอดในทุกเรื่อง

จตุพล ท่นไชย

เมษายน 2556

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

คำอธิบายสัญลักษณ์และคำย่อ

ASIC	=	Application-specific integrated circuit
BCH	=	Bose - Chaudhuri Hocquenghem
CLBs	=	Configurable logic blocks
FPGA	=	Field-programmable gate array
FSK	=	Frequency shift keying
HDL	=	Hardware description language
LUTs	=	Lookup tables
RF	=	Radio Frequency
RS	=	Reed-Solomon
RS-232	=	Recommend Standard 232
SRAM	=	Static random access memory
VHDL	=	Very-high-speed integrated circuits HDL
VSD	=	Vector symbol decoding

ตัวต้นแบบระบบรหัสช่องสัญญาณแบบต่อร่วมที่ใช้การถอดรหัสแบบเวกเตอร์ซิมโบล บนซอฟท์คอร์โปรเซสเซอร์

Lab Prototype of Channel Concatenated Coding System with Vector Symbol Outer Decoder on Soft Core Processor

คำนำ

ปัจจุบันระบบการสื่อสารมีความจำเป็นกับการใช้ชีวิตประจำวันกับผู้คนเป็นอย่างมาก เนื่องจากข้อมูลต่าง ๆ สามารถหาได้บนอินเตอร์เน็ท ซึ่งระบบการสื่อสารในชีวิตประจำวันได้มีการพัฒนาอย่างรวดเร็วโดยเฉพาะระบบการสื่อสารแบบไร้สายที่มีการพัฒนาความเร็วมากกว่าในอดีตเป็นอย่างมาก สำหรับการสื่อสารที่มีองค์ประกอบคือ ผู้ส่งสาร ตัวส่งสาร ช่องสัญญาณ ตัวรับสัญญาณและปลายทางผู้รับสาร หากต้องการความน่าเชื่อถือของข่าวสารจำเป็นต้องมีการทำการแก้ไขความผิดพลาดที่เกิดขึ้น ซึ่งสำหรับช่องสัญญาณที่มีความผิดพลาดเช่น การรบกวนจากสัญญาณรบกวน สัญญาณเกิดการผิดเพี้ยน สัญญาณเกิดการแทรกสอดรบกวน ปัจจัยเหล่านี้ทำให้ประสิทธิภาพการสื่อสารลดลง ดังนั้นในการเพิ่มประสิทธิภาพจึงจำเป็นต้องมีการเข้ารหัสช่องสัญญาณด้วยโดยฝั่งส่งจะทำการเข้ารหัสและฝั่งรับจะทำการถอดรหัส

รหัสช่องสัญญาณนั้นสามารถแบ่งเป็น 2 ประเภทใหญ่คือ รหัสแบบบล็อกและรหัสแบบคอนโวลูชัน โดยที่รหัสแบบบล็อกเป็นการเข้ารหัสที่แบ่งข้อมูลออกเป็นชุด ๆ แล้วจึงนำไปเข้ารหัสซึ่งเป็นการเติมส่วนเติมเต็มเข้าไปผลลัพธ์ที่ได้เรียกว่าคำรหัส คำรหัสเหล่านี้จะผ่านช่องสัญญาณที่ทำให้เกิดความผิดพลาดไปยังฝั่งรับ ทางฝั่งรับสามารถที่จะแก้ไขความผิดพลาดได้เมื่อได้รับคำรหัสที่มีการเข้ารหัสช่องสัญญาณ ส่วนรหัสคอนโวลูชัน เป็นการนำข้อมูลที่ไปเข้ารหัสโดยไม่ได้แบ่งออกมาเป็นชุด ๆ แต่ข้อมูลจะเข้าไปแบบต่อเนื่องซึ่งคำรหัสที่ออกมา จะเป็นอัตราส่วนกับข้อมูลที่เข้าไป

การเข้ารหัสช่องสัญญาณที่มีประสิทธิภาพสูงจะมีความซับซ้อนของวงจรการเข้ารหัสที่มากขึ้น ซึ่งความซับซ้อนนี้ทำให้การสร้างระบบขึ้นมามีค่าใช้จ่ายที่สูงหรือต้องมีวงจรขนาดใหญ่ แต่วิธีการที่จะช่วยลดปัจจัยต่าง ๆ ลงได้คือการเข้ารหัสแบบต่อร่วม (Concatenated Coding) โดยวิธีการเข้ารหัสแบบต่อร่วมเป็นการนำวงจรการเข้ารหัสมาต่อกันซึ่งสามารถต่อแบบขนานหรือแบบ

อนุกรมก็ได้ โดยคำรหัสที่ได้จะเป็นคำรหัสที่มีประสิทธิภาพแต่วงจรที่ใช้สร้างจะเป็นวงจรที่มีความซับซ้อนน้อยกว่าคำรหัสที่มีประสิทธิภาพใกล้เคียงกันแต่ใช้รหัสเดียว โดยการเข้ารหัสแบบต่อร่วมจะมีการแบ่งเป็น 2 ขั้น คือ เป็นการเข้ารหัสแบบภายนอกและการเข้ารหัสแบบภายใน สำหรับวิธีการเข้ารหัสคือทางฝั่งส่งจะทำการเข้ารหัสแบบภายนอกก่อนแล้วจึงนำคำรหัสที่ได้ไปเข้ารหัสแบบภายใน ซึ่งทั้ง 2 ขั้นจะเป็นการเข้ารหัสแบบบล็อกหรือแบบคอนโวลูชันก็ได้ สำหรับวิทยานิพนธ์นี้ทำการเลือกการเข้ารหัสแบบต่อร่วมแบบอนุกรมโดยใช้การเข้ารหัสภายนอกแบบคอนโวลูชันสำหรับนอนไบนารีและการเข้ารหัสภายในแบบบล็อกด้วยวิธีการของรหัสแบบบีซีเฮชและแบบรีดโซโลมอน เมื่อเข้ารหัสแบบต่อร่วมแล้วในการถอดรหัสก็จะต้องทำการถอดรหัสเป็นลำดับขั้น โดยวิธีที่วิทยานิพนธ์เล่มนี้ให้ความสนใจคือ การถอดรหัสแบบลิฟท์ซอฟท์ออฟทูวิเทอร์บีและการถอดรหัสแบบรีดโซโลมอนสำหรับการถอดรหัสภายในและการถอดรหัสแบบเวกเตอร์ซิมโบลสำหรับการถอดรหัสภายนอก

สำหรับในงานวิจัยนี้ในการออกแบบจะทำการออกแบบโดยใช้งานซอฟท์คอร์โปรเซสเซอร์บนบอร์ดทดลอง FPGA ให้ทำงานควบคู่กับส่วนเชื่อมต่อทางพอร์ตอีเทอร์เน็ตเพื่อทำการส่งข้อมูลจากคอมพิวเตอร์มายังบอร์ดทดลองโดยมีการปรับค่าให้สามารถใช้งานได้อย่างยืดหยุ่น พอร์ต RS-232 ที่เชื่อมต่ออยู่กับส่วนรับส่งสัญญาณแบบไร้สายที่ใช้ช่วงความถี่ 433 เมกะเฮิรตซ์และใช้หน่วยความจำบนบอร์ดทดลองที่มีการสร้างรหัสแบบต่อร่วมอยู่เพื่อทำการทดสอบประสิทธิภาพเบื้องต้น สำหรับซอฟท์คอร์โปรเซสเซอร์สามารถที่จะลดเวลาในการทำการออกแบบและมีความยืดหยุ่นในการใช้งานและในปัจจุบันมีการใช้งานซอฟท์คอร์โปรเซสเซอร์เพิ่มมากขึ้น ดังนั้นการใช้ซอฟท์คอร์โปรเซสเซอร์จึงเป็นทางเลือกหนึ่งที่น่าสนใจ สำหรับการสร้างรหัสแบบต่อร่วมที่ใช้การเข้ารหัสภายในแบบรีดโซโลมอน (63,51) กับรหัสภายนอกแบบคอนโวลูชัน (3,2,2) และการถอดรหัสภายในด้วยวิธีการของ Berlekamp-Massey กับการถอดรหัสแบบเวกเตอร์ซิมโบลและทดสอบกับทฤษฎีในเบื้องต้นเพื่อเป็นประโยชน์ในการพัฒนาระบบรหัสช่องสัญญาณต่อไปในอนาคต

งานที่ทำซึ่งมีประโยชน์ต่องานวิจัยเกี่ยวกับรหัสช่องสัญญาณ

1. ตัวต้นแบบตัวเข้ารหัสภายในแบบรีดโซโลมอน ตัวเข้ารหัสภายในแบบบีซีเฮช ตัวเข้ารหัสภายนอกแบบคอนโวลูชันบนซอฟท์คอร์โปรเซสเซอร์

2. ตัวต้นแบบตัวถอดรหัสภายในแบบลิสซอฟท์ออฟทูวิเทอร์บี และตัวถอดรหัสภายนอกแบบเวกเตอร์ซิมโบลบนซอฟท์คอร์โปรเซสเซอร์

3. ตัวต้นแบบระบบรหัสแบบต่อร่วมที่ใช้การเข้ารหัสภายนอกคอนโวลูชัน (3,2,2) กับรหัสรีดโซโลมอน (63,51) และตัวถอดรหัสภายในด้วยวิธีการของ Berlekamp-Masseyกับตัวถอดรหัสภายนอกแบบเวกเตอร์ซิมโบลบนซอฟท์คอร์โปรเซสเซอร์

4. ผลการทดสอบระบบรหัสต่อร่วมบนซอฟท์คอร์โปรเซสเซอร์ที่มีการเชื่อมต่อผ่านพอร์ตอีเทอร์เน็ตและพอร์ต RS-232 บนช่องสัญญาณไร้สาย

ผลงานที่ได้รับการตีพิมพ์

1. Tuntoolavest, U. and J. Thonchai. 2011. VHDL Design of a Concatenated Encoding System. **The International Conference on Information and Communication Technology for Embeded Systems (ICICTES)**. 27-29 January 2013, pattaya, Thailand.
2. Thonchai, J., V. Suktalordcheep and U. Tuntoolavest. 2013. Lab prototype of list-of-2 soft viterbi decoder for a BCH inner code in a generalized concatenated coding system. **The International Conference on Information and Communication Technology for Embeded Systems (ICICTES)**. 24-26 January 2013, Samutsongkhram, Thailand.
3. Tuntoolavest, U., V. Suktalordcheep and J. Thonchai. Concatenated Reed-Solomon Inner and Convolutional Outer Codes for Mobile Channels with Soft Core Processor Implementation. ***submitted to*** Kasetsart Journal (Natural Science).

วัตถุประสงค์

1. เพื่อสร้างตัวเข้ารหัสภายในแบบรีดโซโลมอน ตัวเข้ารหัสภายในแบบบีซีเฮช ตัวเข้ารหัสภายนอกแบบคอนโวลูชันลงบนซอฟท์คอร์โปรเซสเซอร์

2. เพื่อสร้างตัวถอดรหัสภายในแบบลิสซอฟท์ออฟทูวิเทอร์บี และตัวถอดรหัสภายนอกแบบเวกเตอร์ซิมโบลลงบนซอฟท์คอร์โปรเซสเซอร์

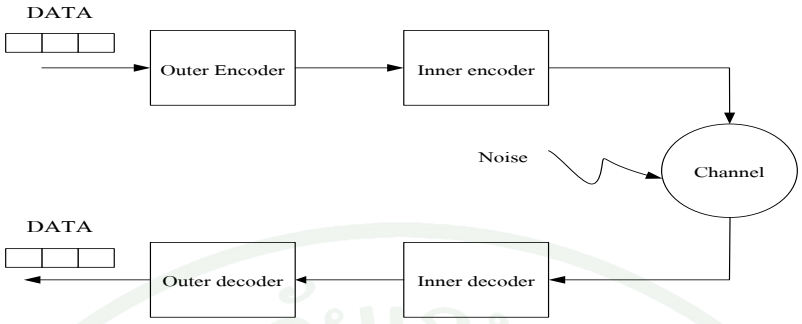
3. ทดสอบการส่งคำรหัสผ่านช่องสัญญาณจริง โดยส่งข้อมูลจากพอร์ตอีเทอร์เน็ตไปยังบอร์ดทดลองและผ่านช่องสัญญาณที่ตัวส่งสัญญาณไร้สายเชื่อมต่อกับพอร์ต RS-232ได้ และรหัสสามารถทำงานได้ถูกต้องตามทฤษฎี

การตรวจเอกสาร

บทนำ

ระบบการสื่อสารที่นิยมใช้กันอย่างแพร่หลายในปัจจุบันคือระบบการสื่อสารแบบดิจิตอล ซึ่งระบบดิจิตอลมีข้อดีคือสามารถสร้างสัญญาณกลับมาใหม่ได้เหมือนเดิมในกรณีที่สัญญาณรบกวนไม่มากทำให้มีความน่าเชื่อถือของข้อมูลสูงและเมื่อมีการพัฒนาระบบดิจิตอลมากขึ้น อุปกรณ์แบบดิจิตอลก็มีราคาถูกลงทำให้เกิดการใช้งานอย่างแพร่หลาย ในการใช้อุปกรณ์ในงานหลากหลายด้าน เช่น ด้านการแพทย์ ด้านการคมนาคม ด้านการก่อสร้าง ด้านการสื่อสาร เป็นต้น ในขณะที่ระบบดิจิตอลถูกพัฒนาไปอย่างรวดเร็วความต้องการความเชื่อถือของข้อมูลจึงเป็นปัจจัยที่ต้องคำนึงถึงเนื่องจากข้อมูลที่มีความสำคัญหากมีความผิดพลาดเกิดขึ้นจะก่อให้เกิดความเสียหายขนาดใหญ่ขึ้นได้ ดังนั้นในการเพิ่มความน่าเชื่อถือของข้อมูลที่ส่งผ่านช่องสัญญาณจำเป็นต้องมีการเข้ารหัสข้อมูลเพื่อให้สามารถตรวจสอบและแก้ไขความผิดพลาดที่เกิดขึ้นระหว่างการส่งข้อมูลผ่านช่องสัญญาณได้

การเข้ารหัสช่องสัญญาณแบ่งได้ 2 ประเภท คือการเข้ารหัสแบบบล็อกและการเข้ารหัสแบบคอนโวลูชัน โดยการเข้ารหัสแบบบล็อกจะประกอบไปด้วย ข้อมูล k บิตและส่วนเติมเต็มเพื่อใช้ตรวจสอบความผิดพลาด $n-k$ บิต และเป็นระบบที่ไม่มีความจำกล่าวคือ คำรหัสปัจจุบันไม่มีความเกี่ยวข้องกับคำรหัสที่ผ่านมา แต่ในการเข้ารหัสแบบคอนโวลูชันเป็นการเข้ารหัสที่มีความจำโดยเมื่อข้อมูลถูกเข้ารหัสแล้วจะเก็บข้อมูลไว้เพื่อใช้ในการสร้างคำรหัสต่อไป วิธีการเข้ารหัสช่องสัญญาณทั้ง 2 วิธีนี้เป็นวิธีการเข้ารหัสที่ใช้กันอย่างแพร่หลาย ต่อมาได้มีการพัฒนาการใช้รหัสที่เรียกว่า รหัสแบบต่อร่วม (Concatenated code) (Fourney, 1966; Lin and Costello, 2004) ดังภาพที่ 1 ซึ่งรหัสแบบต่อร่วมประกอบไปด้วยรหัสภายในและรหัสภายนอก โดยการเข้ารหัสจะทำการเข้ารหัสภายนอกก่อนรหัสภายใน และเมื่อต้องถอดรหัสจะทำการถอดรหัสภายในก่อนรหัสภายนอก ซึ่งวิธีการนี้จะช่วยเพิ่มประสิทธิภาพของรหัสให้สูงขึ้นแต่ความซับซ้อนของระบบจะไม่เพิ่มมากเท่ากับคำรหัสที่มีประสิทธิภาพเท่ากันแต่ไม่ได้ทำการต่อร่วมกัน โดยรหัสที่ใช้สำหรับรหัสแบบต่อร่วมนั้นสามารถใช้ทั้งรหัสแบบบล็อกและรหัสแบบคอนโวลูชัน



ภาพที่ 1 ระบบการเข้ารหัสแบบต่อร่วม

ต่อมาได้มีการเสนอเกี่ยวกับการถอดรหัสแบบเวกเตอร์ซิมโบลดีโคดดิง (Vector Symbol Decoding) (Metzner and Kaprowski, 1990) ซึ่งปรับใช้ได้กับทั้งรหัสแบบบล็อกและรหัสแบบคอนโวลูชันที่ใช้สัญลักษณ์ขนาดใหญ่ ซึ่งข้อดีของการถอดรหัสแบบ VSD คือสามารถแก้ไขความผิดพลาดแบบแถบ (Burst errors) ได้ดีโดยความผิดพลาดเช่นนี้จะเกิดขึ้นมากในช่องสัญญาณแบบไร้สายซึ่งเป็นที่นิยมในปัจจุบัน และได้มีการนำเอาวิธีการนี้ไปพัฒนาอย่างต่อเนื่องเช่น มีการประยุกต์ใช้การถอดรหัสด้วยวิธีเวกเตอร์ซิมโบลกับรหัสคอนโวลูชันที่ไม่เป็นระบบสัญลักษณ์นอนไบนารีพร้อมกับใช้ข้อมูลสำรอง (Tuntoolavest and Metzner, 2002) การเข้ารหัสภายนอกโดยใช้รหัสคอนโวลูชัน (3,2,2) แบบนอนไบนารี เพื่อใช้กับการถอดรหัสแบบเวกเตอร์ซิมโบลดีโคดดิง พร้อมทั้งทดสอบบนบอร์ดทดลอง FPGA (Tuntoolavest and Intharaskul, 2006)

นอกจากนี้ได้มีการออกแบบตัวต้นแบบการถอดรหัสภายในเพื่อใช้สำหรับวิธีการถอดรหัสแบบ VSD โดยทดสอบกับช่องสัญญาณแบบต่าง ๆ (Tuntoolavest and Noradee, 2010) โดยวิธีการถอดรหัสภายในใช้วิธีการที่เรียกว่า ลิฟออฟทูวิเทอร์บี (List Viterbi Algorithm) (Seshadri and Sungburg, 1994; Rodar and Hamzaoui, 2006) การถอดรหัสด้วยวิธีการนี้จะให้ผลลัพธ์ของชุดข้อมูลตัวเลือกรองออกมาและผู้วิจัยได้เลือกใช้ทางเลือกทั้งสองเพื่อนำไปช่วยในการคำนวนการถอดรหัสภายนอกแบบ VSD นอกจากนี้ยังถูกทดลองบนบอร์ดทดลอง FPGA และทำการวัดผลความถูกต้องกับโปรแกรม C++ ด้วย

ในงานวิจัยนี้จะให้ความสนใจในการศึกษาและทดลองในส่วนของระบบรหัสช่องสัญญาณแบบต่อร่วมซึ่งประกอบไปด้วยการเข้ารหัสภายนอกด้วยรหัสคอนโวลูชัน (3,2,2) และการเข้ารหัสภายในบีซีเฮช (31,26) รวมถึงการถอดรหัสภายในแบบลิสซอฟท์ออฟทูวิเทอร์บี แบบรีดโซโลมอน (63,51) และการถอดรหัสภายนอกแบบเวกเตอร์ซิมโบล (Vector Symbol Decoding) โดยทดลอง

บนบอร์ดทดลอง FPGA (Field-programmable gate array) ด้วยวิธีการสร้างซอฟท์คอร์โปรเซสเซอร์เพื่อใช้ในการควบคุมอุปกรณ์เชื่อมต่อคือพอร์ตอีเทอร์เน็ตและพอร์ต RS-232 เพื่อที่จะรับส่งข้อมูลมาทำการเข้ารหัสและถอดรหัส ซึ่งจะทดลองให้เป็นไปตามทฤษฎีโดยมีการส่งผ่านข้อมูลไปยังอุปกรณ์ส่งไร้สายจริงเพื่อทดสอบการตรวจสอบและแก้ไขความผิดพลาดที่เกิดขึ้น

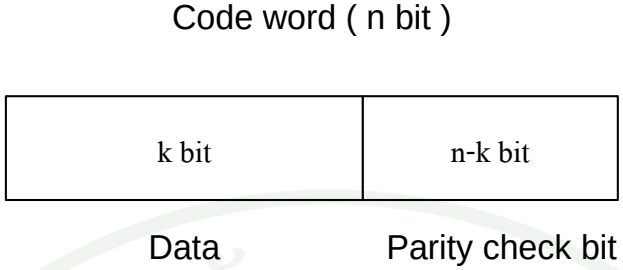
ในการสร้างชิ้นงานนี้ จะมีทฤษฎีสำคัญที่เกี่ยวข้องคือ การเข้ารหัสช่องสัญญาณเช่น การเข้ารหัสแบบคอนโวลูชัน การเข้ารหัสแบบบีซีเฮช การเข้ารหัสแบบรีดโซโลมอน การถอดรหัสด้วยวิธีการลิฟออฟทูวิเทอร์บี การถอดรหัสด้วยวิธีการเวกเตอร์ซิมโบลดีโคดดิง การถอดรหัสด้วยวิธีการของ Berlekamp-Massey การออกแบบระบบภายในบอร์ดทดลองFPGA การรับ-ส่งข้อมูลผ่านอุปกรณ์

1. การเข้ารหัสช่องสัญญาณ

การเข้ารหัสช่องสัญญาณ (Channel encoding) เป็นวิธีการหนึ่งที่จะเพิ่มความน่าเชื่อถือให้กับข้อมูลก่อนที่จะส่งข้อมูลผ่านช่องสัญญาณ ผู้ส่งจะทำการเข้ารหัสเพื่อให้สามารถทนต่อการรบกวนของสัญญาณรบกวนได้ ซึ่งวิธีการเข้ารหัสสามารถแบ่งออกได้เป็น 2 ประเภทได้แก่

1.1 รหัสบล็อก (Block code) (พิสิฐ และคณะ, 2552)

รหัสบล๊อกเป็นรหัสที่จะทำการเข้ารหัสเป็นชุดหรือที่เรียกว่า บล๊อก จากภาพที่ 2โดยจะนำข้อมูลที่มีขนาด k บิตไปทำการเข้ารหัสข้อมูลซึ่งผลลัพธ์ที่ได้ออกมาเป็นคำรหัส (Code word) ที่มีขนาด n บิต โดยส่วนเติมเต็มที่เข้ามามีขนาด $n-k$ บิตจะเป็นบิตที่นำไปใช้ในการตรวจสอบความผิดพลาด รหัสบล็อกเป็นรหัสที่ไม่มีความจำ (Memoryless) กล่าวคือคำรหัสที่ได้จะไม่มีความสัมพันธ์กันระหว่างข้อมูลปัจจุบันและข้อมูลที่ผ่านมา รหัสประเภทนี้มีการใช้งานอย่างแพร่หลายเช่น รหัสไซคริก (Cyclic code) (Prange, 1957) รหัสรีดโซโลมอน (Reed and Solomon, 1960) และรหัสบีซีเฮช (Hocquenghem, 1959; Bose and Ray-Chaudhuri, 1960) เป็นต้น



ภาพที่ 2 โครงสร้างรหัสแบบบล๊อก

1.1.1 รหัสบีซีเฮช

รหัสบีซีเฮช (Hocquenghem, 1959; Bose and Ray-Chaudhuri, 1960) เป็นรหัสประเภทไซคลิกชนิดหนึ่งซึ่งสามารถแก้ไขความผิดพลาดแบบสุ่มหลายบิตได้ดีสำหรับค่าต่าง ๆ ที่เกี่ยวข้องกับรหัสบีซีเฮช มีดังนี้

ความยาวของคำรหัส	$n = 2^m - 1$	(1)
ความยาวของบิตข้อมูล	$k \geq n - mt$	(2)
ความยาวของบิตเช็ก	$n - k \leq mt$	(3)
ระยะแฮมมิงต่ำสุด	$d_{\min} \geq 2t + 1$	(4)

ความยากของคำรหัสหาได้จากสมการที่ (1) ,ความยาวของบิตข้อมูลหาได้จากสมการที่ (2) ,ความยาวของบิตเช็กหาได้จากสมการที่ (3) และ จากสมการ (4) t คือความผิดพลาดที่สามารถแก้ได้

สำหรับการออกแบบรหัสแบบ BCH สามารถทำได้ดังนี้ (พิสิฐ และคณะ, 2552)

1. หาอิลีเมนต์ α ใน $GF(q^m)$ ที่มีอันดับเท่ากับ n โดยที่ m เป็นจำนวนเต็มบวกที่มีค่าน้อยที่สุดที่หารค่า $q^m - 1$ ได้ลงตัว

2. คำนวณหาค่าจาก $\delta = 2t + 1$

3. เลือก $\alpha^b, \alpha^{b+1}, ..., \alpha^{b+\delta-2}$ ให้เป็นรากของพหุนามตัวต้นกำเนิดโดยที่ค่า b เป็นจำนวนเต็มที่มีค่าไม่น้อยกว่า 0

4. จากนั้นทำการคำนวณหาค่าตัวคูณร่วมน้อยของพหุนามจากสมการที่ (5) เพื่อ หา $g(x)$

$$g(x) = LCM\{\Phi_1(x), \Phi_1(x), ..., \Phi_{\delta-1}(x)\} \quad (5)$$

5. คำนวณหาขนาดความยาวสัญลักษณ์ข้อมูล $k = n - r$ โดยที่ r เป็นดีกรีของ $g(x)$

ในงานวิจัยด้านต่าง ๆ ได้มีการนำเอารหัสบีซีเฮชไปใช้งานอย่างหลากหลายเช่น การสร้างตัวต้นแบบสำหรับการใช้งานสื่อสารไร้สายบน FPGA ด้วยการเข้ารหัส บีซีเฮชโดยผลลัพธ์ที่ได้สามารถทำงานได้ที่ความถี่ 249.8 เมกะเฮิรตซ์และใช้ทรัพยากรเพียงเล็กน้อย (Rajesh and Garima, 2011) การถอดรหัสแบบบีซีเฮชสำหรับระบบแบบ DVB-S2 (Digital video broadcasting – satellite 2nd) โดยทำงานได้ที่ความเร็ว 380 MB/s โดยใช้เกตทั้งหมด 4,400 เกตโดยใช้ เทคโนโลยี 0.13 ไมโครเมตร CMOS (Complementary metal-oxide-semiconductor) (Yi-min and *et al*., 2009) เป็นต้น

1.1.2 รหัสรีดโซโลมอน

รหัสรีดโซโลมอน (Reed and Solomon, 1960) จัดว่าเป็นรหัสบล็อกเชิงเส้นที่เป็นกลุ่มย่อยของรหัส บีซีเฮชถูกพัฒนาและคิดค้นขึ้นโดย Irving S. Reed และ Gustav Solomon สำหรับรหัสชนิดนี้ได้ถูกนำมาประยุกต์อย่างหลากหลาย ตัวอย่างเช่น ระบบเชื่อมโยงไมโครเวฟ การสื่อสารผ่านดาวเทียม ระบบดิจิตอลทีวี เป็นต้น

รหัสรีดโซโลมอน สามารถแก้ไขความผิดพลาดได้ดังนี้

$$t = \left(\frac{d_{\min} - 1}{2}\right) = \left(\frac{n-k}{2}\right) \quad (6)$$

จากสมการที่ (6) t คือความผิดพลาดที่สามารถแก้ไขได้ สำหรับรหัสแบบรีดโซโลมอนเป็นรหัสแบบนอนไบนารีกล่าวคือ รหัสรีดโซโลมอนจะมีความยาวบล๊อกทั้งหมด n สัญลักษณ์และความยาวของข้อมูล k สัญลักษณ์เราสามารถคำนวนพหุนามตัวกำเนิดของรหัสรีดโซโลมอนที่สามารถแก้ไขได้ t สัญลักษณ์ได้ดังสมการที่ (7) ดังนี้

$$g(x) = (x - \alpha^b)(x - \alpha^{b+1})....(x^{b+2t-1}) \quad (7)$$

โดยที่ α เป็นพริมิทิฟอีลีเมนต์ของ $GF(q)$ และ $b \geq 0$ เนื่องด้วยตัวกำเนิดสามารถคำนวนได้จากค่าต่ำสุดที่ $2t$ และ รหัสไซคลิกใด ๆ เราคำนวนหาค่า k ได้จาก $k = n - r$ ดังนั้น $k = n - 2t$ เนื่องจาก $g(x)$ ต้องมีดีกรี r เท่ากับ $2t$

1.1.3 การถอดรหัสบีซีเฮชและรหัสรีดโซโลมอน

สำหรับขั้นตอนการถอดรหัสบีซีเฮชและรหัสรีดโซโลมอนสามารถทำได้ดังนี้ (พิสิฐ และคณะ, 2552)

1. การคำนวณหาค่าซินโดรม S = {S_1, S_2, S_3,….., S_{2t}} จาก $r(x)$ ซึ่งหมายถึงสัญญาณที่รับมาได้ในส่วนนี้จะไม่รู้ว่ามีความผิดพลาดรวมอยู่ด้วยหรือไม่

2. คำนวณพหุนามระบุตำแหน่งความผิดพลาด $\sigma(x)$ จากเซตของซินโดรมในข้อที่ 1

3. จากส่วนกลับของรากของ $\sigma(x)$ จะได้ตำแหน่งของความผิดพลาด $\beta_1, \beta_2, ..., \beta_v$

4. คำนวณตำแหน่งของความผิด $j_1, j_2, j_3, ..., j_v$ จาก $\beta_1, \beta_2, ..., \beta_v$ ได้จากข้อที่ 3

5.คำนวณค่าความผิดพลาด $e_1, e_1, e_1, ..., e_v$ ณ ตำแหน่งของความผิดพลาด $j_1, j_2, j_3, ..., j_v$ จากสมการ (8)

$$
\begin{aligned}
S_1 &= e_{j1}\beta_1 + e_{j2}\beta_2 + ... + e_{jv}\beta_v \\
S_2 &= e_{j1}\beta_1^2 + e_{j2}\beta_2^2 + ... + e_{jv}\beta_v^2 \\
S_3 &= e_{j1}\beta_1^3 + e_{j2}\beta_2^3 + ... + e_{jv}\beta_v^3 \\
&\vdots \\
S_{2t} &= e_{j1}\beta_1^{2t} + e_{j2}\beta_2^{2t} + ... + e_{jv}\beta_v^{2t}
\end{aligned}
\qquad (8)
$$

6. แก้ไขคำรหัสให้ถูกต้องด้วยค่าความผิดพลาดจากข้อที่ 5

สำหรับการถอดรหัสโดยวิธีการจากข้อ 1-6 จะเป็นวิธีการคำนวณที่เรียบง่ายแต่ในการคำนวณพหุนามระบุตำแหน่งความผิดพลาดจะเป็นส่วนที่มีความซับซ้อนในการคำนวณและมีผู้เสนอแนวทางในการคำนวณอย่างหลากหลาย หนึ่งในนั้นคือวิธีการของ Berlekamp-Massey ซึ่งอาศัยหลักการของการวนซ้ำเพื่อหาตำแหน่งของความผิดพลาด $\sigma(x)$ โดยสามารถ

เปรียบเทียบวิธีการนี้เป็นวิธีการหาค่าสัมประสิทธิ์ของการป้อนกลับของชิฟรีจิสเตอร์ที่มีขนาดเล็กที่สุดสำหรับขั้นตอนของอัลกอริทึมของวิธีการ Berlekamp-Massey โดยอาศัยหลักการวนซ้ำเพื่อหาพหุนามตำแหน่งความผิดพลาดได้จากสมการพหุนามระบุตำแหน่งความผิดพลาดดังนี้ (พิสิฐ และคณะ, 2552)

$$\sigma^{(n)}(x) = 1 + \sigma_1^{(n)}x + \sigma_2^{(n)}x^2 + ... + \sigma_{v_n}^{(n)}x^{v_n}, v_n \le n \tag{9}$$

โดยที่จากสมการที่ (9) $\sigma^{(n)}(x)$ คือค่าพนุหนามดีกรีต่ำที่สุด $V_{(n)}$ สำหรับผลต่างระหว่างพจน์ซ้ายและพจน์เรียกว่าผลต่างดิสครีแพนซี (discrepancy) ของรอบที่ n หรือสมการที่ (10) (พิสิฐ และคณะ , 2552)

$$d_n = s_{n+1} + \sigma_1^{(n)}s_n + \sigma_2^{(n)}s_{n-1} + ... + \sigma_{v_n}^{(n)}s_{n-v_{n+1}} \tag{10}$$

สำหรับสมการรอบที่ $n+1$หาได้จากสมการที่ (11) (พิสิฐ และคณะ, 2552)

$$\sigma^{(n+1)}(x) = \sigma^{(n)}(x) - d_n d_m^{-1}x^{n-m}\sigma^{(nm)}(x) \tag{11}$$

โดยที่ m เป็นรอบก่อนหน้ารอบที่ ที่ให้ค่า $d_m \neq 0$ และให้ค่า $m - v_m$ มากที่สุด
และจากสมการ(10) สามารถหาความสัมพันธ์ได้ดังสมการที่ (12) (พิสิฐ และคณะ, 2552)

$$V_{n+1} = \max[v_n, v_m + n - m] \tag{12}$$

จากสมการ(11) การจะทำให้ V_{n+1} มีค่าน้อยที่สุดต้องให้ $v_m + n - m$ มีค่าน้อยที่สุดเช่นกัน
สำหรับขั้นตอนของอัลกอลิทึม Berlekamp-Massey มีดังนี้ (พิสิฐ และคณะ, 2552)

1. กำหนดค่าเริ่มต้น $\sigma^{(-1)}(x) = 1, v_{-1} = 0, d_{-1} = 1$และ $\sigma^{(1)}(x) = 1, v_0 = 0, d_0 = s_1, n = 0$
2. ถ้า $d_n = 0$ ให้ $\sigma^{(n+1)}(x) = \sigma^{(n)}(x)$ และ $V_{n+1} = v_n$
3. ถ้า $d_m \neq 0$ ให้หารอบ m ก่อนหน้ารอบที่ n เมื่อ $1 \le m \le n$ ที่ทำให้ค่า $d_m \neq 0$ และให้ค่า $m - v_m$ มากที่สุดและคำนวณ $\sigma^{(n+1)}(x)$ จากสมการ (11) และ ค่า V_{n+1} จากสมการที่ (12)
4. ถ้า $n = 2t - 1$หรือ $n \ge v_n + t - 1$ ให้ $\sigma(x) = \sigma^{(n+1)}(x)$ แล้วหยุดทำงาน
5. เพิ่มค่า n ขึ้น 1 ค่า

6. คำนวณค่า d_n จากสมการ(10)
7. กลับไปทำขั้นตอนที่ 2

เมื่อได้ $\sigma(x)$ ซึ่งเป็นพหุนามระบุตำแหน่งความผิดพลาดแล้วก็จะนำไปคำนวณตำแหน่งที่มีความผิดพลาดต่อไปก่อนที่จะหาค่าความผิดพลาดและนำค่าที่ได้ไปแก้ไข

1.2 รหัสคอนโวลูชัน (พิสิฐ และคณะ, 2552)

รหัสแบบคอนโวลูชันถูกพัฒนาขึ้นในปี 1955 (P.Elias, 1955) ซึ่งเป็นรหัสที่มีความจำ กล่าวคือผลลัพธ์ที่ได้ขึ้นกับข้อมูลทั้งในอดีตและปัจจุบัน โดยการเข้ารหัสแบบคอนโวลูชันจะไม่แบ่งข้อมูลออกเป็นบล็อก แต่จะนำเข้าข้อมูลที่ไม่จำกัดความยาวไปเข้ารหัสอย่างต่อเนื่อง โดยมีอัตรารหัสบอกได้ถึงบิตอินพุตและบิตเอาท์พุต

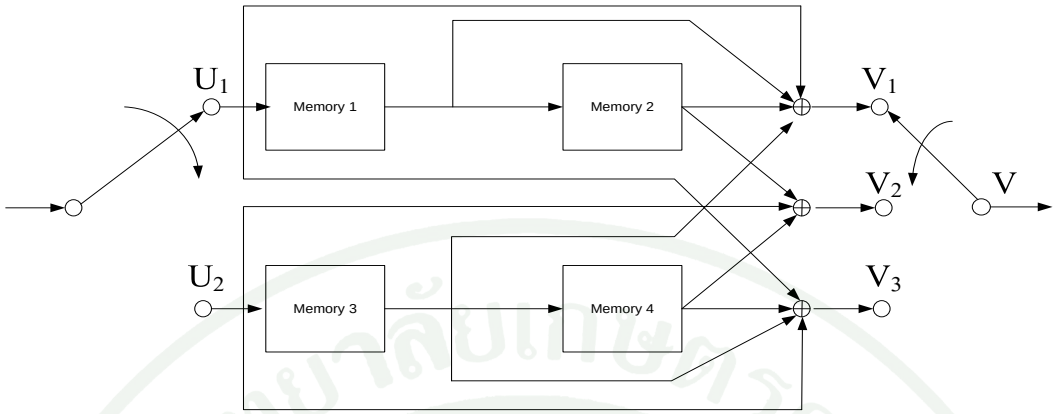
$$\text{Code rate} = \frac{k}{n} \tag{13}$$

ซึ่งจาก (13) ค่าอัตรารหัสนั้นต้องมีค่าไม่เกิน 1 เนื่องจาก k เป็นความยาวของข้อมูลหลังจากเข้ารหัสแล้วต้องมีการเติมส่วนช่วยแก้ไขเข้าไปในคำรหัสดังนั้น คำรหัสที่มีความยาว n บิตจะต้องมากกว่าค่า k

การเข้ารหัสแบบคอนโวลูชันสามารถแบ่งออกเป็น 2 ประเภทคือ การเข้ารหัสคอนโวลูชันแบบไบนารี ซึ่งข้อมูลนั้นแต่ละตัวเป็นสมาชิกของ $GF(2)$ โดยจะมีค่าอยู่ในเซตของ 0 กับ 1 ซึ่งต่างจากรหัสคอนโวลูชันอีกประเภทคือ การเข้ารหัสคอนโวลูชันแบบนอนไบนารี โดยคำรหัสของการเข้ารหัสจะเป็นสมาชิกของ $GF(2^m)$ โดยที่ $m>1$ กล่าวคือในแต่ละข้อมูลที่ทำการเข้ารหัสต่อ 1 ครั้งจะมีขนาด m บิต ตัวอย่างการเข้ารหัสแบบคอนโวลูชัน (3,2,2) ซึ่งมีเมตริกฟังก์ชันถ่ายโอน G(D) ดังสมการที่ (14)

$$G(D) = \begin{bmatrix} 1+D+D^2 & D^2 & 1 \\ D & 1+D^2 & 1+D+D^2 \end{bmatrix} \tag{14}$$

ตัวเข้ารหัสจะมี 3 อินพุต และ 2 เอาท์พุต และมีจำนวนหน่วยความจำต่ออินพุต ซึ่งสามารถเขียนเป็นลักษณะของวงจรการเข้ารหัสได้ดังภาพที่ 3



ภาพที่ 3 วงจรการเข้ารหัสคอนโวลูชัน (3,2,2)

ถ้ากำหนดให้อินพุตเป็นดังสมการที่ (15)

$$\begin{aligned} U^{(1)} &= \left(u_0^{(1)}, u_1^{(1)}, u_2^{(1)}, u_3^{(1)}, \ldots\right) \\ U^{(2)} &= \left(u_0^{(2)}, u_1^{(2)}, u_2^{(2)}, u_3^{(2)}, \ldots\right) \end{aligned} \tag{15}$$

และให้ลำดับของเอาต์พุตเป็นดังสมการที่ (16)

$$\begin{aligned} V^{(1)} &= \left(v_0^{(1)}, v_1^{(1)}, v_2^{(1)}, v_3^{(1)}, \ldots\right) \\ V^{(2)} &= \left(v_0^{(2)}, v_1^{(2)}, v_2^{(2)}, v_3^{(2)}, \ldots\right) \\ V^{(3)} &= \left(v_0^{(3)}, v_1^{(3)}, v_2^{(3)}, v_3^{(3)}, \ldots\right) \end{aligned} \tag{16}$$

ซึ่งจะได้คำรหัสเป็นดังสมการที่ (17)

$$V^{(1)} = \left(v_0^{(1)}v_0^{(2)}v_0^{(3)}, v_1^{(1)}v_1^{(2)}v_1^{(3)}, v_2^{(1)}v_2^{(2)}v_2^{(3)}, v_3^{(1)}v_3^{(2)}v_3^{(3)}, \ldots\right) \tag{17}$$

โดยที่การคำนวนคำรหัสง่ายขึ้นจากสมการที่ (18)

$$V(D) = U(D) \cdot G(D) \tag{18}$$

จะได้ว่า

$$V(D) = \begin{bmatrix} U^{(1)}(D) & U^{(2)}(D) \end{bmatrix} \begin{bmatrix} 1+D+D^2 & D^2 & 1 \\ D & 1+D^2 & 1+D+D^2 \end{bmatrix} \tag{19}$$

จากสมการ (19) จะได้คำรหัสคือ

$$\begin{aligned} V^{(1)}(D) &= U^{(1)}(D) + U^{(1)}(D) \bullet D + U^{(1)}(D) \bullet D^2 + U^{(2)}(D) \bullet D \\ V^{(2)}(D) &= U^{(1)}(D) \bullet D^2 + U^{(2)}(D) + U^{(2)}(D) \bullet D^2 \\ V^{(3)}(D) &= U^{(1)}(D) + U^{(2)}(D) + U^{(2)}(D) \bullet D + U^{(2)}(D) \bullet D^2 \end{aligned} \tag{20}$$

จากสมการที่ (20) ซึ่งสามารถเปรียบเทียบกับภาพที่ 3 ได้โดยมีหน่วยความจำที่มีความสัมพันธ์กับสมการคือ *Memory 1* จาก $U^{(1)}(D) \bullet D$ *Memory 2* จาก $U^{(1)}(D) \bullet D^2$ *Memory 3* จาก $U^{(2)}(D) \bullet D$ *Memory 3* จาก $U^{(2)}(D) \bullet D^2$

2. การถอดรหัสด้วยวิธีการลิสออฟทูวิเทอร์บี (List-of-2 Viterbi)

การถอดรหัสแบบวิเทอร์บี (Viterbi) (Vitebi, 1967) เป็นการถอดรหัสที่ให้ประสิทธิภาพสูงสุดสำหรับการถอดรหัส (Optimum solution) โดยการถอดรหัสใช้วิธีการที่เรียกว่า Maximum likelihood ซึ่งด้วยวิธีการดังนี้ทำให้ผลลัพธ์ที่ออกมาจากการถอดรหัสแบบวิเทอร์บีเหมือนกับข้อมูลต้นฉบับที่ส่งมามากที่สุด สำหรับวิธีการถอดรหัสด้วย Viterbi สามารถแบ่งออกได้เป็น2ลักษณะคือ

1. การถอดรหัสแบบการตัดสินใจแบบฮาร์ด (Hard decision) วิธีการนี้จะใช้การตัดสินใจข้อมูลที่รับมาทีละบิตโดยค่าของความจริงมีค่าเป็น 0 หรือ 1 เท่านั้น

2. การถอดรหัสแบบซอฟท์ (Soft decision) วิธีการนี้จะใช้พิจรณาข้อมูลที่รับมาโดยแบ่งระดับของการตัดสินใจไว้มากกว่า 2 ระดับ ซึ่งจะทำให้ได้ความละเอียดในการตัดสินใจมากกว่าวิธีการแบบฮาร์ด

สำหรับขั้นตอนการทำงานของวิธีการถอดรหัสแบบ วิเทอร์บีนั้นจะมีขั้นตอนดังนี้

1. การทำการคำนวนค่า Branch matric ที่รับเข้ามากับค่าข้อมูลในเส้นทางการเปลี่ยนแปลงสถานะต้องคำนวนทุก ๆ เส้นทาง

2. คำนวนหาค่า Path matric ทำได้โดยเป็นผลรวมจากค่า Branch matric รวมกับค่าเส้นทางก่อนหน้าสเตทปัจจุบัน หลังจากนั้นจะทำการหา Survivor path ซึ่งเป็นเส้นทางที่ดีที่สุดถูกเก็บไว้เพื่อทำการคำนวนต่อไป

3. หลังจากคำวนทุกช่วงเวลาของข้อมูลที่รับมาแล้วจะทำการเลือกเส้นทางที่เป็น survivor ของช่วงเวลาสุดท้ายไปจนถึงช่วงเวลาเริ่มต้นหลังจากทำการตามรอยกลับไปแล้วผลลัพธ์ที่ได้จะเป็นข้อมูลที่ดีที่สุด

ต่อมาในปี 1989 Seshadri และ Sundbure ได้เสนอแนวคิดในการใช้ลิฟออฟทูวิเทอร์บี (Seshadri and Sungberg, 1989) ซึ่งวิธีการนี้เป็นวิธีการที่ให้ผลลัพธ์ที่ดีที่สุดมา 2 ผลลัพธ์ และได้มีการแสดงผลลัพธ์ในการสร้างและออกแบบลิสออฟทูวิเทอร์บีด้วยภาษา VHDL สำหรับบอร์ดทดลอง FPGA (Tuntoolavest and Noradee, 2009)

3. การถอดรหัสแบบเวกเตอร์ซิมโบล

การถอดรหัสแบบเวกเตอร์ซิมโบล (Mezner and Kapturowski, 1990) ถูกนำเสนอโดย Metzner และ Kupturowski เป็นวิธีการที่เหมาะสมกับรหัสแบบนอนไบนารีและมีประสิทธิภาพในการถอดรหัสข้อมูลที่มีความผิดพลาดแบบแถบ (Burst error) ได้ดี ต่อมาได้มีการพัฒนาการนำข้อมูล 2 ชุดมาช่วยในการถอดรหัส (Tuntoolavest and Metzner, 2001;Tuntoolavest, 2004) และได้ทำการทดสอบการถอดรหัสแบบ VSD ลงบนบอร์ดทดลอง FPGA (Tuntoolavest and Seubnaung, 2007) ซึ่งให้ผลลัพธ์ออกมาสามารถทำการถอดรหัสได้เป็นอย่างดี

ขั้นตอนการการถอดรหัสด้วยวิธีการแบบเวกเตอร์ซิมโบลดีโคดดิงสำหรับรหัสคอนโวลูชัน (Tuntoolavest, 2009)

1. เริ่มจากการรับข้อมูลเข้ามาเป็นลำดับ และทำการคำนวนหาซินโดรม โดยอ้างอิงจาก n สัญลักษณ์ของรหัส สำหรับรหัสคอนโวลูชัน (3,2,2)ก็จะรับมา 3 สัญลักษณ์

2. ถ้าหากซิมโดรมที่คำนวนได้มีค่าเป็น 0 ตัวถอดรหัสจะพิจารณาให้ไม่มีความผิดพลาดเกินขึ้นใน n สัญลักษณ์แรก ซึ่งจะถือว่าสัญลักษณ์นี้ถูกต้องและนำไปทำการดึงข้อมูลกลับจากโครงสร้างของรหัสต่อไป

3. ตัวถอดรหัสทำการเก็บข้อมูลที่ถูกถอดรหัสแล้วไปใช้สำหรับการถอดรหัสครั้งต่อไป

4. จากนั้นจะทำการทำซ้ำข้อ 1 จนกว่าจะเจอซินโดรมที่ไม่เป็น 0 ทำจนกว่าจะสิ้นสุดข้อมูลที่รับมา

5. เมื่อตัวถอดรหัสเจอซินโดรมที่ไม่เป็น 0 จะทำการแก้ไขความถูกต้องด้วยซินโดรมเดียว ถ้าสามารถแก้ไขได้จะทำการเก็บค่าไว้ในหน่วยความจำเพื่อช่วยในการถอดรหัสต่อไป จากนั้นตัวถอดรหัสจะทำการคำนวนหาซินโดรมต่อไป จนกว่าจะเจอซินโดรมที่ไม่เป็น 0 หรือจนกว่าจะสิ้นสุดข้อมูลที่รับมา

6. ถ้าตัวถอดรหัสไม่สามารถทำการแก้ไขได้ซินโดรมเดียว จะทำการแก้ไขด้วยตัวเลือกรอง ถ้าสามารถถอดรหัสได้ จะเก็บข้อมูลไว้ในหน่วยความจำจากนั้นจะทำการคำนวนหาซินโดรมต่อไป จนกว่าจะเจอซินโดรมที่ไม่เป็นศูนย์หรือสิ้นสุดข้อมูลที่รับมา

7. ตัวถอดรหัสจะทำการแก้ไขข้อมูลด้วยผลรวมศูนย์ ถ้าไม่สามารถทำการแก้ไขด้วยตัวเลือกรองหรือหากยังเหลือข้อมูลของสัญลักษณ์ที่มีความผิดพลาดเหลือ สามารถพบได้จากการคำนวนซินโดรมต่อไป

8. การแก้ไขข้อมูลด้วยผลรวมศูนย์ ตัวถอดรหัสจะทำการหาผลรวมศูนย์และสามารถระบุตำแหน่งความผิดพลาดได้ ถ้าหากการแก้ไขข้อมูลด้วยผลรวมศูนย์สำเร็จ จะไปยังข้อ 10

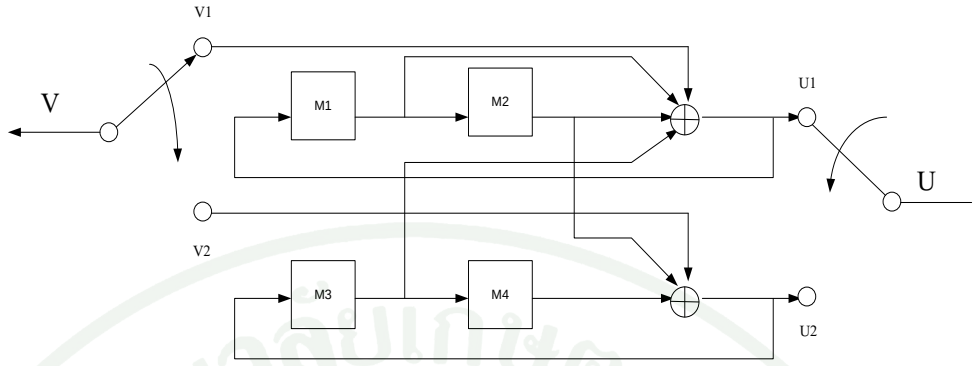
9. ถ้าการแก้ไขด้วยผลรวมศูนย์ไม่สำเร็จ จะทำการนำข้อมูลมาช่วยจำนวน n สัญลักษณ์ ซึ่งสัญลักษณ์ที่เข้ามาจะนำมาช่วยในการคำนวนซินโดรม ในข้อ 6

10. ตัวถอดรหัสจะทำการหาค่าที่แท้จริงของสัญลักษณ์ที่ผิดพลาดและทำการแก้ไข

11. หลังจากทำการถอดรหัสเรียบร้อยแล้ว สัญลักษณ์ทั้งหมดจะถูกนำมาพิจารณาและนำมาเก็บไว้ในหน่วยความจำ หลังจากนั้นจะทำการคำนวนซินโดรมต่อไปจนกว่าจะพบซินโดรมที่ไม่เป็น 0 หรือหมดชุดข้อมูลที่รับมา

4. การกู้ข้อมูลกลับต้นฉบับ (อรรถภัทร์ สืบเนื่อง, 2552)

หลังจากทำการแก้ไขความผิดพลาดด้วยวิธีการของเวกเตอร์ซิมโบลดีโคดดิงเสร็จสิ้น จะต้องทำการดึงข้อมูลคืนกลับจากคำรหัสที่ได้มีการทำการแก้ไขเพื่อให้ได้ข้อมูลต้นฉบับกลับคืนมา สำหรับการเข้ารหัสแบบไม่เป็นระบบ (3,2,2) คอนโวลูชันโคดที่ใช้ในงานวิจัยนี้ สามารถดึงข้อมูลโดยผ่านคำรหัสเข้าไปในวงจรดังภาพที่ 4



ภาพที่ 4 วงจงการกู้ข้อมูลกลับต้นฉบับ

สำหรับการทำงานของวงจรการดึงคำรหัสกลับต้นฉบับจะทำงานด้วยการแบ่งคำรหัสที่ได้ตามขนาดของสัญลักษณ์ออกเป็น 3 สัญลักษณ์ป้อนข้อมูลที่เป็นคำรหัสที่ 1 และ คำรหัสที่ 2 เข้าไปในวงจร โดยวงจรจะทำการนำคำรหัสที่ 1 Xor กับความจำที่ 1 ความจำที่2 และความจำที่ 3 สำหรับคำรหัสที่2 จะผ่านไปทำการ Xor กับความจำที่4และความจำที่ 2 หลังจากได้ข้อมูลต้นฉบับคืนมาจึงป้อน ข้อมูลต้นฉบับที่ 1 ไปเก็บค่าในหน่วยความจำที่ 1 และ นำข้อมูลด้นฉบับที่ 2 ไปเก็บค่าในหน่วยความจำที่ 3

5. การออกแบบวงจรภายใน FPGA (Jonh, 2000)

Field- programmable gate array (FPGA) เป็นวงจรรวมดิจิตอลที่ถูกออกแบบให้ผู้ใช้สามารถโปรแกรมการทำงานลงไปในตัวชิพได้โดยการโปรแกรมการทำงานของ FPGA จะทำได้จะต้องใช้ Hardware description language (HDL) ซึ่งเป็นภาษาที่ใช้อธิบายการทำงานของวงจรภายใน FPGA โดยผู้ใช้สามารถทำการตั้งค่าได้หลังจากผ่านการผลิตจากโรงงานแล้วอีกทั้งยังมีราคาถูกกว่าวงจรประเภท ASIC (Application specific integrated circuit) ซึ่งมีประสิทธิภาพมากกว่าแต่ถูกสร้างมาเพื่อให้ใช้งานอย่างจำกัดเท่านั้น

ส่วนประกอบของ FPGA ปะกอบไปด้วย Configurable logic blocks (CLBS) อยู่ภายในชิพซึ่งทำหน้าที่สร้างการทำงานลอจิกแบบต่างๆตามที่ผู้ใช้งานต้องการ โดยแต่ละ CLBs จะประกอบไปด้วย Lookup Table (LUT) ต่ออยู่กับ Flip-flops โดยความซับซ้อนของระบบนั้นขึ้นอยู่กับว่าทางผู้ผลิตจะออกแบบมาให้ซับซ้อนเพียงใด และอีกส่วนหนึ่งภายใน FPGA คือ I/O pins (Input output

pins) เป็นส่วนที่ใช้ในการเชื่อมต่อกับส่วนด้านนอกพินสามารถต่อเข้ากับ I/O pins ได้หลากหลายมาตรฐาน

ขั้นตอนในการทำการออกแบบการทำงานของ FPGA ทำได้โดย

1. การออกแบบวงจร โดยการออกแบบวงจรให้เป็นไปตามการทำงานด้วยการเขียน HDL (Hardware description Language) โดยขั้นตอนนี้จะทำการกำหนดสิ่งที่ผู้ใช้ต้องการให้วงจรทำงาน

2. การตรวจสอบความถูกต้อง โดยการจำลองการทำงานของค่าทางเวลาเพื่อตรวจสอบพฤติกรรมการทำงานของอุปกรณ์ที่ต้องการ

3. การสังเคราะห์วงจร ขั้นตอนนี้จะทำการสังเคราะห์วงจรผ่านซอฟท์แวร์ที่มีความเข้ากันได้กับ FPGA โดยการสังเคราะห์จะให้เน็ทลิสต์ในระดับเกตออกมา หมายความว่า หลังจากการสังเคราะห์วงจรแล้วผู้ใช้จะได้ค่าความเชื่อมโยงต่าง ๆ ในชิพ FPGA ออกมา

4. การแปลงดีไซน์ที่สังเคราะห์แล้วเพื่อไปใช้กับอุปกรณ์ FPGA ขั้นตอนนี้จะนำเน็ทลิสต์ไปแปลงเป็นวงจรแล้วจึงไปทำการวางข้อมูลเพื่อวงจรต่างๆ บนอุปกรณ์จริง สุดท้ายจะเป็นการวิเคราะห์ทางด้านเวลาถึงความเป็นไปได้ของวงจรที่สร้าง

การนำไฟล์แบบบิตไปใช้กับอุปกรณ์ โดยการโปรแกรมผ่านสายข้อมูลเพื่อนำการออกแบบที่ทำไปใช้กับอุปกรณ์จริง

ในปัจจุบันได้มีการพัฒนาวิธีการใช้งาน FPGA ขึ้นอีกรูปแบบหนึ่งคือ การใช้งานซอฟท์คอร์โปรเซสเซอร์ (Soft core processor) บน FPGA โดยวิธีการนี้เป็นที่นิยมเพิ่มขึ้นอย่างมาก เนื่องมาจากสามารถลดขนาดอุปกรณ์และราคาลงได้ สำหรับทางด้านการใช้งานซอฟท์คอร์โปรเซสเซอร์จะสามารถเขียนโปรแกรมภาษา C เพื่อควบคุมลงไปบนชิพ FPGA ได้และยังใช้พลังงานน้อยในการทำงานและข้อได้เปรียบสำคัญคือ ผู้ใช้งานสามารถสร้างหลายชุดในชิพ FPGA 1 ตัวขึ้นอยู่กับความจุของลอจิกเกตภายใน FPGA ด้วย โดยในตารางที่ 1 เป็นตารางสำหรับงานวิจัยด้านต่าง ๆ ที่น่าสนใจซึ่งมีการพัฒนาเกี่ยวกับรหัสและทำการทดลองบนบอร์ดทดลอง FPGA

ตารางที่ 1 ตารางการวิจัยเกี่ยวกับการใช้งาน FPGA ในด้านต่าง ๆ

ชื่อเรื่อง	ผู้แต่ง	ปี	รายละเอียด
FPGA Implementation and Verification of Reed Solomon (63,47,8) Code in SDR System	Yi hua chen,Chang Lueng Chu,Chan Chun Yehk,Kun Fenng Lin	2012	การสร้างรหัสรีดโซโลมอน (63,47,8) บนบอร์ด FPGA
An application of MB-OFDM UWB technology for wireless speaker systems	Do-Hoon kim,Kyu-Min keng,Chanyoung Lee	2012	การปรับรุงคุณภาพของการส่งสัญญาณเสียงผ่านระบบไร้สายโดยการใช้การเข้ารหัสแบบ RS (255,171) และทดลองบนบอร์ดทดลอง FPGA
Design and Implementation of Reed-Solomon Decoder for 802.16 Neworking using FPGA	Bhawa Tiwari, Rajesh Mesha	2012	การออกแบบและสร้างตัวถอดรหัสแบบ Reed-Solomon สำหรับ Wimax network โดยใช้ FPGA
Choice and Implementation of a reed Solomon Code for low power low data rate communication systems	Lionel Biard and Dominique Noguet	2007	การทดลองการสร้าง Reed-solomon ในระบบที่ต้องการการประหยัดพลังงานและวิเคราะห์ถึงประสิทธิภาพของรหัสกับความซับซ้อนของวงจร FPGA
Experimental Demonstration of Concatenated LDPC and RS codes by FPGA emulator	Takeshi Mizuochi,Yoshiaki konishi,Yoshikuni Miyata and *et al*.	2009	ทดลองการสร้างรหัสแบบต่อร่วมชนิด LDPC และ RS แบบต่าง ๆ บนบอร์ดจำลอง FPGA และทดสอบกับการส่งข้อมูลความเร็วสูง

ตารางที่ 1 (ต่อ)

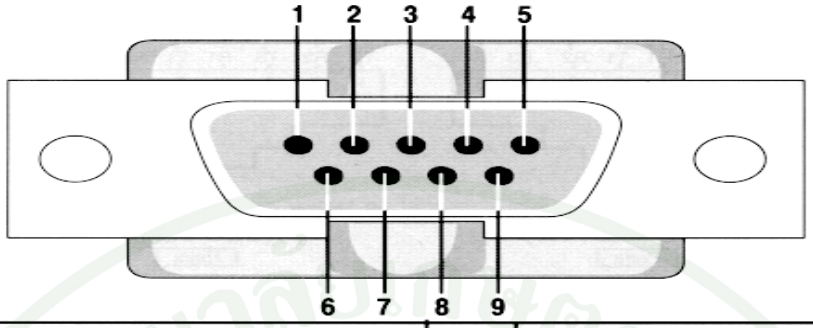
ชื่อเรื่อง	ผู้แต่ง	ปี	รายละเอียด
Soft core robot with joint wheel motion controller	Carvalhosa, A., P. Machado., A. Sousa and J.C. Alves	2009	การออกแบบและใช้งาน Soft core เพื่อช่วยในการควบคุมข้อต่อระหว่างล้อสำหรับหุ่นยุนต์
Design & development of soft-core processor based remote terminal units for nuclear reactors	Aditya, G., A. S. Raj and et.al.	2011	การออกแบบรีโมตเพื่อควบคุมโดยใช้ soft-core บนบอร์ดทดลองแบบ FPGA เพื่อควบคุมเตาปฏิกรนิวเคลียร์
Design of SINS/GPS integrated navigation system based on dual NiosII soft-core	Zhi, c., L. Miao and J. Shen.	2012	การออกแบบระบบนำทางโดยเลือกใช้ NiosII soft-core

6. การรับ-ส่งข้อมูลผ่านอุปกรณ์

ในการรับส่งข้อมูลของอุปกรณ์แต่ละชนิดจะมีมาตรฐานแตกต่างกันสำหรับงานวิจัยชิ้นนี้สนใจในการส่งข้อมูลผ่านพอร์ตอีเทอร์เน็ตและ RS-232 เนื่องจากการใช้งานที่หลากหลายของพอร์ตทั้ง 2 ที่อุปกรณ์ในท้องตลาดปัจจุบันรองรับดังนั้นในการทดลองส่งจริงจึงสามารถหาอุปกรณ์ที่นำมาร่วมทดสอบได้อย่างลงตัวและราคาไม่แพงมากนัก

6.1 RS-232 (Recommended Standard 232)

เป็นมาตรฐานการเชื่อมต่อข้อมูลแบบอนุกรมใช้เพื่อเพิ่มระยะทางในการส่งข้อมูลแบบอนุกรมให้สามารถส่งได้ระยะทางที่มากขึ้น โดยมีการเปลี่ยนระดับแรงดันของลอจิกจากเดิมอยู่ที่ในช่วง 0-5 v หรือ 0-3.3 โวลต์เป็นช่วง -15 ถึง15โวลต์สำหรับการจัดขาของคอนเน็กเตอร์อนุกรมแบบ DB9 แสดงในภาพที่ 5



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

ภาพที่ 5 คอนเน็กเตอร์อนุกรมแบบ RS-232

ที่มา : Se-Education,Co.,ltd (2006)

จากภาพที่ 5 มีการทำงานของขาต่าง ๆ ดังนี้

ขาที่ 1 DCD (Data Carrier Detect) ใช้สำหรับเมื่อมีการส่งสัญญาณพาร์จากอุปกรณ์สื่อสารข้อมูล
ขาที่ 2 RXD (Receive Data) ใช้สำหรับรับข้อมูล
ขาที่ 3 TXD (Transmitted Data) ใช้สำหรับส่งข้อมูล
ขาที่ 4 DTR (Data Terminal Ready) ใช้สำหรับบอกอุปกรณ์ปลายทางให้รับรู้ว่าต้องการติดต่อด้วย
ขาที่ 5 GND (Signal Ground) กราน์ของระบบ
ขาที่ 6 DSR (Data Set Ready) ใช้คู่กับขาที่ 4 เพื่อตรวจสอบการเชื่อมต่อระหว่างอุปกรณ์
ขาที่ 7 RTS (Request To Send) เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมา
ขาที่ 8 CTS (Clear To Send) ใช้คู่กับขาที่ 7 เพื่อตรวจสอบว่าอุปกรณ์ที่เชื่อมต่อพร้อมรับข้อมูลหรือไม่
ขาที่ 9 RI (Ring Indicator) ใช้แสดงสถานะสัญญาณเรียกจากโทรศัพท์ปกติจะไม่ได้ใช้งาน

สำหรับอัตราการส่งข้อมูลสามารถใช้งานได้หลายระดับด้วยกันเช่น 300 1200 2400 4800 9600 19200 38400 56000 เป็นต้น

6.2 ETHERNET (เกรียงศักดิ์, 2544)

อีเทอร์เน็ตเป็นเทคโนโลยีแบบแลนด์ที่ใช้กันอย่างแพร่หลายในปัจจุบัน ซึ่งมีอุปกรณ์ที่รองรับมากเนื่องจากมีความเร็วในการรับส่งข้อมูลที่สูง ระบบอีเทอร์เน็ตเหมาะกับงานที่มีอัตราส่งข้อมูลเป็นช่วงๆ ในการรับส่งข้อมูลของอีเทอร์เน็ตแต่ละครั้งนั้นจะทำการรับส่งกันอย่างไม่เป็นระเบียบ โดยจะส่งข้อมูลออกมาหากมีการชนกันของข้อมูลจะทำการหยุดส่งและรอเวลา แล้วจึงส่งข้อมูลใหม่อีกครั้ง

อีเทอร์เน็ตในช่วงแรกจะทำการส่งข้อมูลแบบบัสต่อมามีการพัฒนาไปสู่การส่งแบบสตาร์ที่รวมสายข้อมูลเข้าศูนย์กลาง สำหรับมาตรฐานของอีแทอร์เน็ตคือมาตรฐาน IEEE 802.3 โดยการจำแนกข้อมูลของอีเทอร์เน็ตจะสามารถแยกได้ดังนี้

1. ความเร็ว มีตั้งแต่ 10,100,1000 เมกะบิตต่อวินาที ซึ่งเป็นอัตราการส่งข้อมูลสูงที่สุดที่สามารถส่งได้ในทางทฤษฎี แต่ในทางปฏิบัติจะส่งข้อมูลได้ช้ากว่าความเร็วนี้อย่างมาก
2. การส่งสัญญาณทางไฟฟ้า ซึ่งประกอบด้วย 2 ลักษณะคือ Baseband เป็นการส่งสัญญาณแบบดิจิตอล 0 และ 1 แรงดันไฟฟ้า 0 และ 5 โวลต์ โดยไม่มีการรวมกับความถี่สูงอื่น ข้อเสียของวิธีการนี้คือ มีการรบกวนได้ง่าย ระยะทางใกล้ อีกลักษณะหนึ่งคือ Broadband เป็นการส่งสัญญาณรวมกับความถี่สูง เพื่อให้ส่งได้ไกลและมีสัญญาณรบกวนที่น้อยกว่าทำให้มีความเร็วที่มากขึ้น แต่ข้อเสียคือการทำงานของอุปกรณ์ที่ซับซ้อนที่มากกว่าและมีราคาที่สูงกว่า

3. สายที่ใช้ โดยมีการพัฒนาสายขึ้นเรื่อยมาทั้ง Fiber optic และสายแบบ UTP (Unshielded twisted pair) โดยแต่ละสายมีรหัสดังนี้

รหัส 5 ใช้สายแบบ Thick Coaxial ขนาดใหญ่โยงถึงกัน ความยาวสูงสุดไม่เกิน 500 เมตร

รหัส 2 ใช้สายแบบ Thin Coaxial ขนาดเล็กโยงถึงกัน ความยาวสูงสุดไม่เกิน 200 เมตร

รหัส T ใช้สำหรับสายแบบ UTP แบบที่เรียกว่าสาย Category 5 หรือ Cat 5 โดยต่อจากทุกเครื่องเข้าอุปกรณ์รวมสาย ระยะความยาวสูงสุด 100 เมตรโดยประมาณ

รหัส F ใช้สำหรับระบบที่ใช้สาย Fiber optic ซึ่งความยาวหลายร้อยเมตรขึ้นไป

อุปกรณ์และวิธีการ

อุปกรณ์

1. เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer)
2. บอร์ดทดลองรุ่น Altium Nanoboard 3000 ชิป Xilinx Spartan-3AN รุ่น XCS1400AN-4FGG676C
3. สายอีเทอร์เน็ต (Ethernet Cable)
4. สายจัม (Jumper Wire)
5. เครื่องรับ-ส่ง สัญญาณวิทยุแบบ FSK (FSK RF transmitter-receiver)
6. โปรแกรม MATLAB
7. โปรแกรม Altium Designer Release 09

วิธีการ

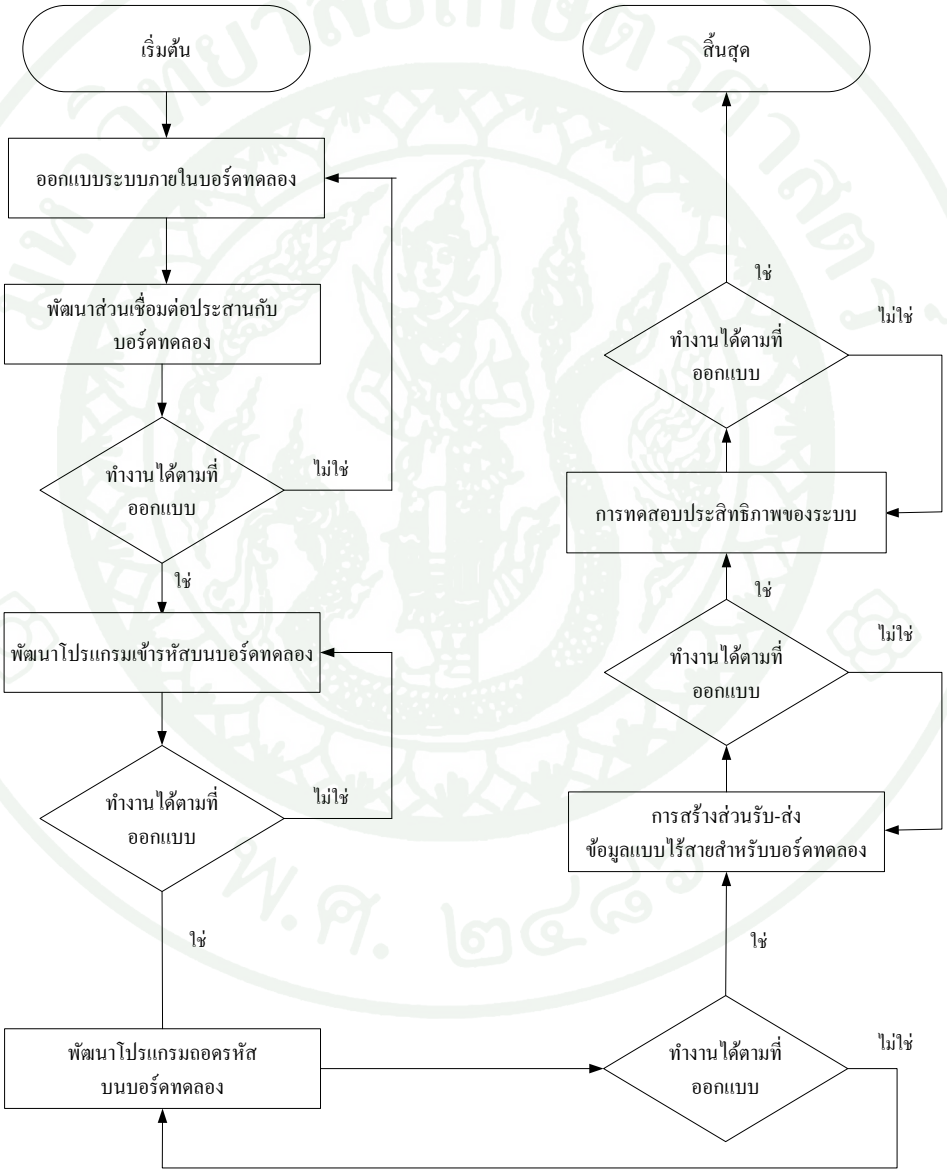
ในงานวิจัยชิ้นนี้สนใจในการสร้างตัวต้นแบบของระบบการเข้ารหัสและถอดรหัสแบบต่อร่วมลงบนบอร์ดทดลอง FPGA โดยระบบนั้นประกอบไปด้วย

1. ส่วนเชื่อมต่อระหว่างคอมพิวเตอร์ส่วนบุคคลไปยังบอร์ดทดลอง
2. ส่วนการเข้ารหัสแบบต่อร่วม
3. ส่วนการรับ-ส่งสัญญาณแบบไร้สายผ่านช่องสัญญาณ
4. ส่วนการถอดรหัสแบบต่อร่วม

ในการสร้างตัวต้นแบบของระบบการเข้ารหัสและถอดรหัสแบบต่อร่วมจะต้องทำความเข้าใจในส่วนของการเข้ารหัสและถอดรหัสก่อนเพื่อให้สามารถออกแบบได้อย่างมีประสิทธิภาพ เมื่อมีความเข้าใจการเข้ารหัสและถอดรหัสช่องสัญญาณก็จะเริ่มขั้นตอนการออกแบบตาม ภาพที่ 6 ซึ่งประกอบไปด้วย

1. การออกแบบระบบภายในบอร์ดทดลองเพื่อความเหมาะสมสำหรับการเข้ารหัสและถอดรหัส

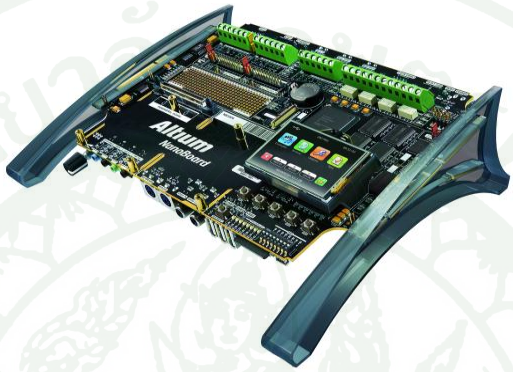
2. การสร้างส่วนเชื่อมต่อประสานสำหรับบอร์ดทดลอง
3. พัฒนาโปรแกรมการเข้ารหัสลงบนบอร์ทดลอง
4. พัฒนาโปรแกรมการถอดรหัสลงบนบอร์ดทดลอง
5. การสร้างส่วนรับ-ส่งข้อมูลแบบไร้สายสำหรับบอร์ดทดลอง
6. การทดสอบประสิทธิภาพของระบบ



ภาพที่ 6 แผนผังการการออกแบบระบบการเข้าและถอดรหัสแบบต่อร่วม

1. การออกแบบระบบภายในบอร์ดทดลอง

1. บอร์ดทดลองรุ่น Altium Nanoboard 3000



ภาพที่ 7 Altium Nanoboard 3000

ที่มา: Altium Limited co., Ltd. (2009)

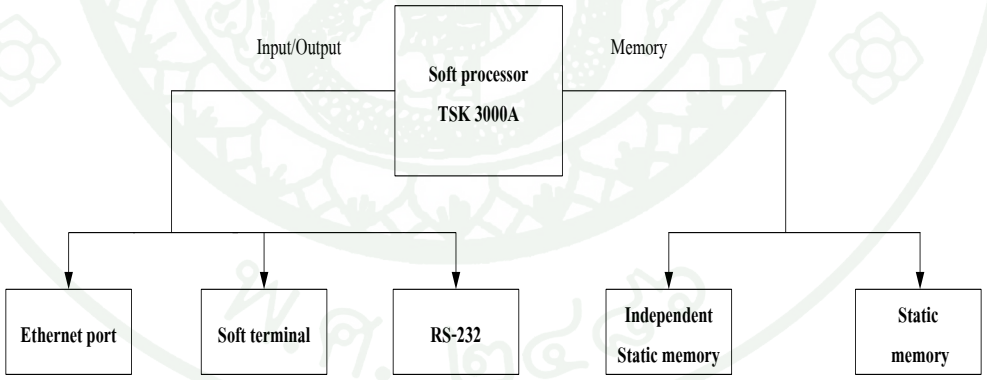
ในการออกแบบอุปกรณ์สำหรับการสร้างระบบการเข้ารหัสและถอดรหัสแบบต่อร่วมที่ใช้สำหรับงานวิจัยนี้คือ Altium Nanoboard 3000 จากภาพที่ 7 ซึ่งเป็นบอร์ดทดลองที่มีชิพ FPGA ของบริษัท Xilinx รุ่น Spartan-3AN XCS1400AN-4FGG676C ที่มีความจุวงจรเท่ากับ 1,400,000 เกต สามารถใช้ร่วมกับโปรแกรม Altium Designer ได้เป็นอย่างดีเพื่อการออกแบบให้ได้วงจรในแบบที่ต้องการ ในส่วนของอุปกรณ์ที่อยู่บนบอร์ดทดลองประกอบไปด้วย

1. จอ TFT LCD (Thin film transistor liquid crystal display) ขนาด 240x320 พิกเซล
2. USB hubs ที่รองรับการเชื่อมต่อแบบ USB ได้ถึง 3 ช่อง
3. พอร์ต SVGA (Super video ıgraphics array) ขนาด 24บิต ความถี่ 80 MHz,
4. รองรับการเชื่อมต่อมาตรฐานอย่างหลากหลาย เช่น RS-232 RS-485 Ps/2 10/100 fast Ethernet USB2.0 S/PDIF MIDI
5. มีความถี่นาฬิกาที่สามารถโปรแกรมได้ตั้งแต่ 6 MHz ถึง 200 MHz และความถี่นาฬิกาสำหรับบอร์ดทดลองที่มีความเร็ว 20 MHz

6. หน่วยความจำบนบอร์ดทดลองที่สามารถใช้ได้ประกอบไปด้วย SRAM (Static random-access memory) ขนาด 1 MB SDRAM (Synchronous dynamic random-access memory) ขนาด 64 MB Flash Memory ขนาด 16 MB เอสแรมที่สามารถใช้งานได้อย่างอิสระมีขนาด 512 KB จำนวน 2 ชิ้น

1.2 โปรแกรม Altium Designer และการออกแบบด้วยระบบ Open bus

โปรแกรม Altium Designer เป็นโปรแกรมที่ใช้สำหรับการออกแบบ FPGA บนบอร์ดทดลองของบริษัท Altium ซึ่งสำหรับการออกแบบจะแบ่งออกเป็น 2 ส่วนใหญ่ ๆ คือ การออกแบบฮาร์ดแวร์ของระบบและการออกแบบซอฟต์แวร์ของระบบ โดยต้องเริ่มจากการออกแบบระบบ Open bus ซึ่งเป็นระบบเฉพาะสำหรับโปรแกรม Altium Designer ทำหน้าที่ในการเลือกการเชื่อมต่อระหว่างอุปกรณ์บนบอร์ดทดลองและชิพ FPGA ซึ่งแสดงในภาพที่ 8 องค์ประกอบสำหรับระบบที่ต้องการสร้างขึ้นประกอบไปด้วย ซอฟท์โปรเซสเซอร์ TSK3000A ทำหน้าที่ประมวลผลการทำงานของระบบซึ่งมีการเชื่อมต่ออินพุตเอาต์พุตทางซ้ายมือของรูป และเชื่อมต่อกับหน่วยความจำทางขวามือของภาพ



ภาพที่ 8 ระบบ Open bus

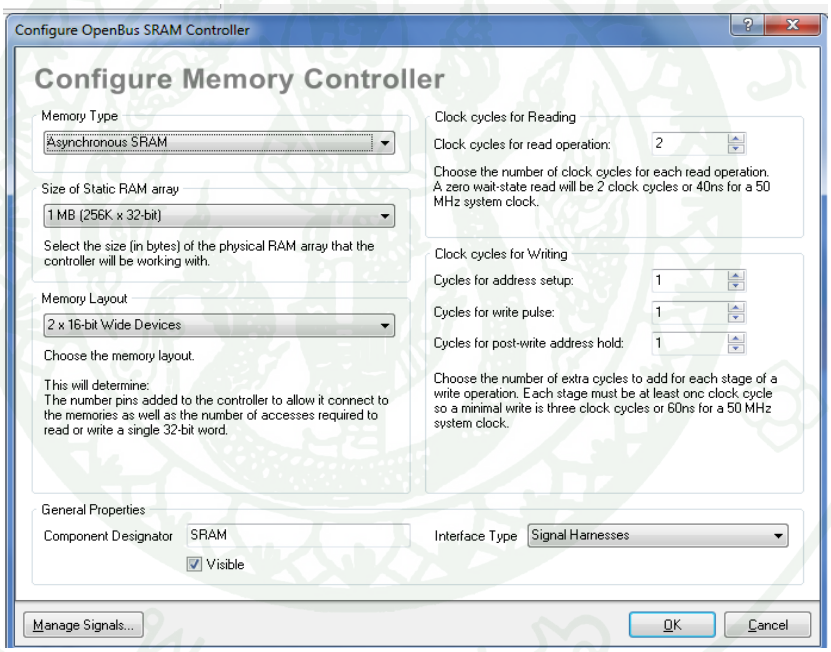
องค์ประกอบส่วนของอินพุตและเอาต์พุตประกอบไปด้วย 3 อุปกรณ์คือ

1. Serial Communication Port เพื่อใช้ในการสื่อสารกับพอร์ตอนุกรม RS-232 โดยจะนำไปเชื่อมต่อกับอุปกรณ์รับ-ส่งสัญญาณแบบไร้สายต่อไป

2. Soft terminal เป็นส่วนที่ถูกจำลองขึ้นเพื่อเป็นส่วนแสดงผลลัพธ์ที่ได้จากการประมวลผลภายในบอร์ดทดลอง

3. Ethernet port ส่วนนี้จะใช้สำหรับการเชื่อมต่อกับพอร์ตอีเทอร์เน็ตบนบอร์ดทดลองซึ่งจะใช้เชื่อมต่อประสานเข้ากับคอมพิวเตอร์ส่วนบุคคล

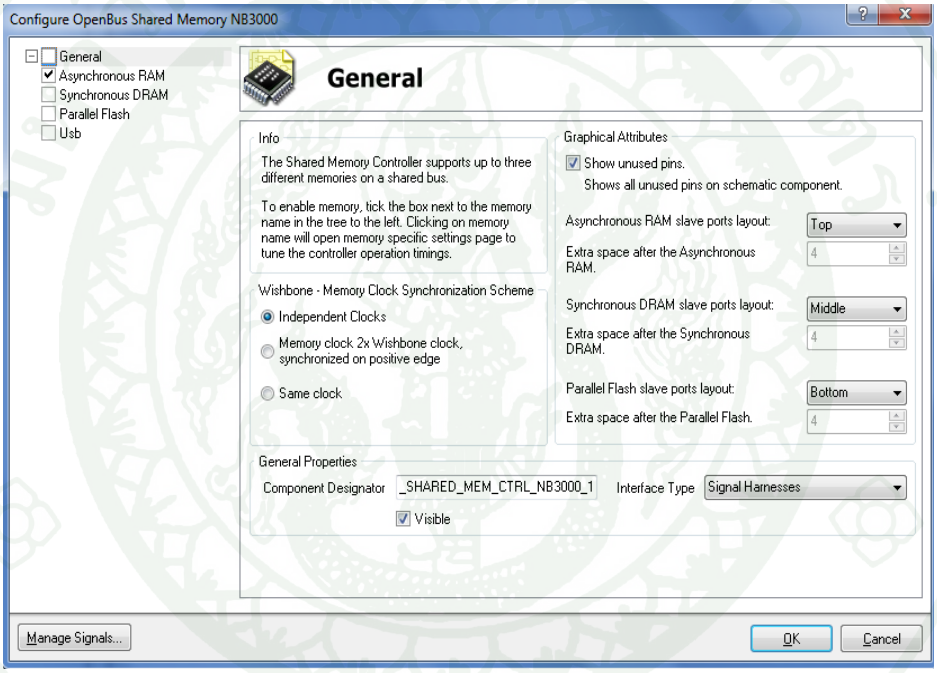
องค์ประกอบของส่วนหน่วยความจำจะประกอบด้วย 2 อุปกรณ์คือ เอสแรม(SRAM) และหน่วยความจำร่วม (Shared Memory) โดยหลังจากทำการเชื่อมต่อระหว่างอุปกรณ์แล้วจำเป็นที่จะต้องกำหนดค่าในหน่วยความจำนั้นไว้ตามต้องการ



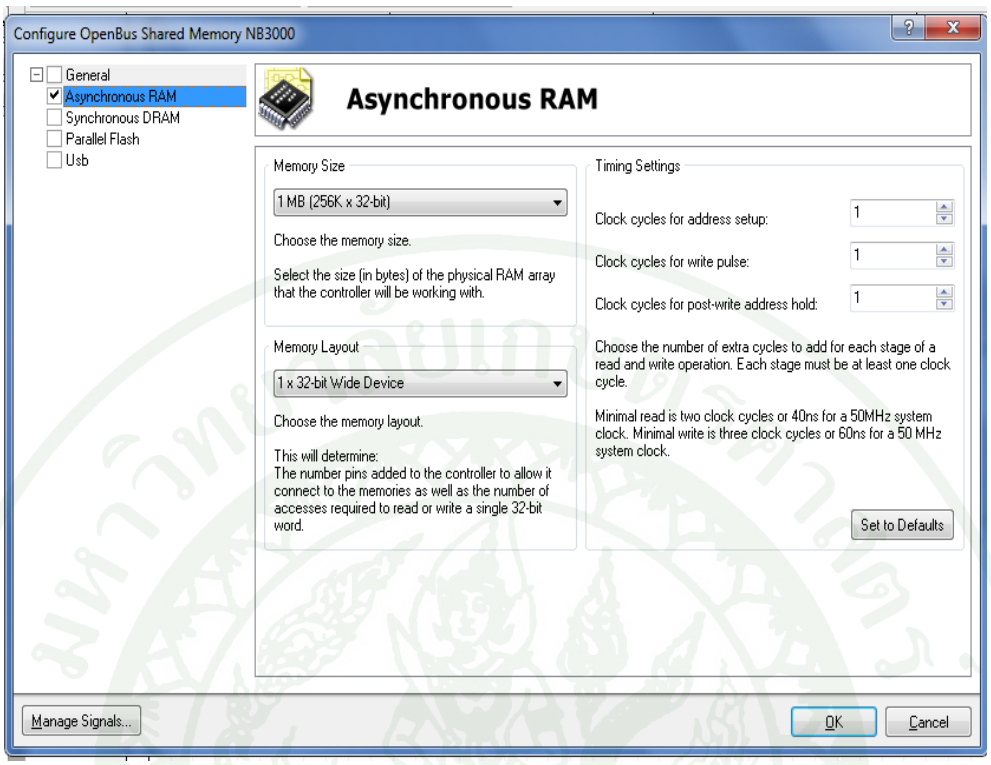
ภาพที่ 9 ภาพการตั้งค่าแอสแรม (SRAM)

สำหรับในการตั้งค่า เอสแรม (SRAM) นั้น จากภาพที่ 9 สามารถที่จะตั้งค่าขนาดของความจุได้หลายค่า เช่น 128 KB 256 KB 512 KB 1 MB เป็นต้น แต่การตั้งค่าที่มากเกินกว่าที่ทรัพยากรในบอร์ดทดลองมีนั้นจะทำให้หลังจากที่ทำการประมวลผลระบบ Open bus ก่อนไปทำขั้นตอนต่อไปเกิดค่าความผิดพลาดขึ้นดังนั้นในการออกแบบของงานวิจัยชิ้นนี้จะใช้ขนาดความจุ 1 MB สำหรับเอสแรม SRAM ซึ่งเป็นความจำเต็มขนาดของหน่วยความจำประเภทนี้ที่อยู่บนบอร์ดทดลอง

การตั้งค่า หน่วยความจำร่วม (Shared Memory) ของ Nanoboard 3000 นั้นสามารถเลือกได้ทั้งหมด 4 ค่า เนื่องจากหน่วยความจำทั้ง 4 ใช้ตัวควบคุมตัวเดียวกันในการควบคุมจึงสามารถที่จะเลือกหน่วยความจำที่ต้องการใช้งานได้ ในงานวิจัยชิ้นนี้จะเลือกใช้ เอสแรม (SRAM) ของหน่วยความจำร่วม (Shared memory) เพิ่ม เพื่อรองรับข้อมูลที่ต้องการนำไปใช้ในการประมวลผล การทำการตั้งค่าจะเป็นไปตามภาพที่ 10 โดยจะเลือกความถี่ของนาฬิกาที่เป็นอิสระจากบอร์ดทดลองและในส่วนของ Asynchronous Ram จะเลือกขนาดความจุของหน่วยความจำเป็น 1 MB ขนาดของข้อมูล 32 บิต ดังภาพที่ 11



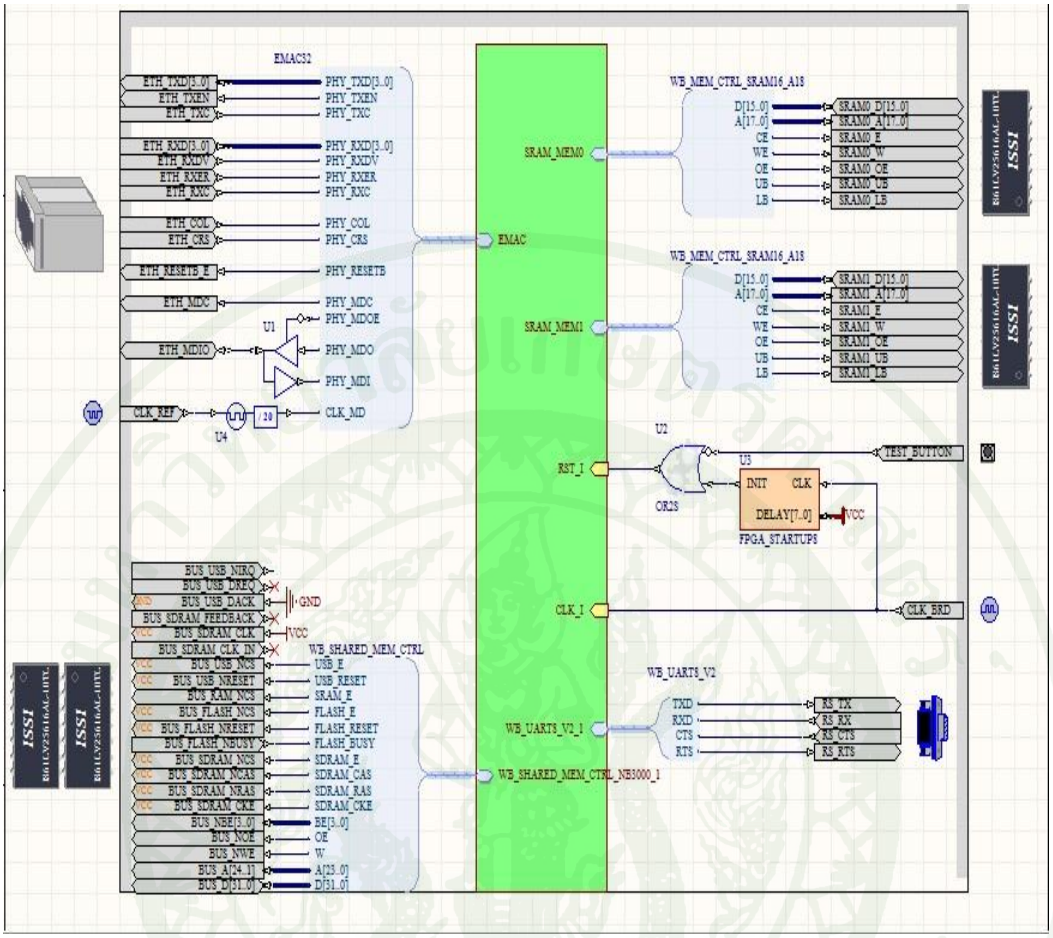
ภาพที่ 10 การตั้งค่าหน่วยความจำร่วม (Shared Memory)



ภาพที่ 11 การตั้งค่า Asynchronous RAM

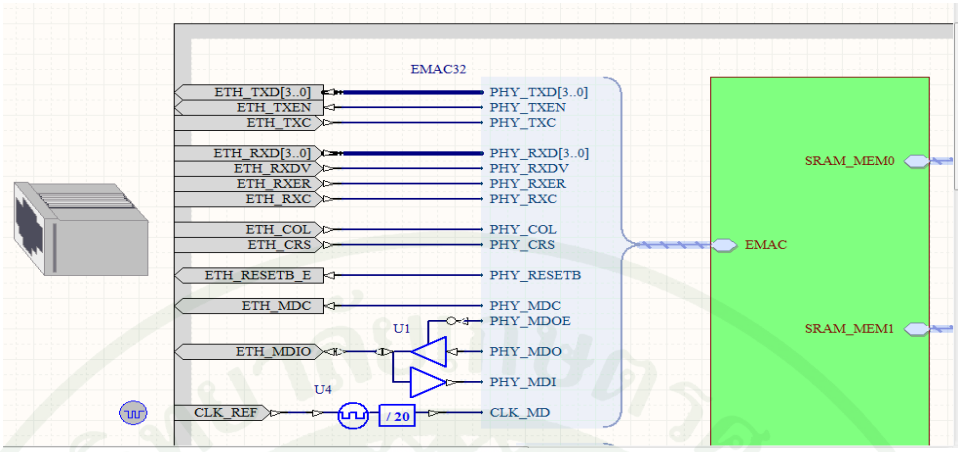
1.1 การออกแบบแผนภาพเค้าร่าง(Schematic)และการเชื่อมโยงระหว่างอุปกรณ์

หลังจากออกแบบด้วยระบบ Open bus เสร็จแล้ว ขั้นตอนต่อไปคือการเชื่อมโยงค่าที่ได้จากการประมวลผลโดยระบบ Open bus กับอุปกรณ์ที่มีบนบอร์ดทดลองเพื่อให้สามารถใช้งานได้ตามการออกแบบโดยเริ่มจากการนำค่าที่ได้จากระบบ Open bus มาสร้างส่วนที่เป็นส่วนหลักสำหรับการเชื่อมต่ออุปกรณ์ต่าง ๆ ดังภาพที่ 12



ภาพที่ 12 รูปการเชื่อมต่อระหว่างหน่วยประมวลผลและอุปกรณ์ต่าง ๆ บนบอร์ดทดลอง

ในงานวิจัยนี้จะเลือกใช้ซอฟท์คอร์โปรเซสเซอร์เป็นส่วนควบคุมการทำงานของอุปกรณ์ต่าง ๆ บนบอร์ดทดลองอุปกรณ์ทั้งหมดที่เลือกใช้จะอยู่ในภาพที่ 12 โดยมีซอฟท์คอร์โปรเซสเซอร์ เป็นศูนย์กลางเชื่อมต่อกับหน่วยความจำและอินพุตเอาต์พุตของระบบ สำหรับการเชื่อมต่อระหว่างอุปกรณ์จะมี 2 แบบคือแบบ Wire ใช้สำหรับการเชื่อมต่อสัญญาณทั่วไป และแบบ Bus ใช้สำหรับการเชื่อมต่อข้อมูลที่มีขนาดของสัญญาณมากกว่า 1 เส้น



ภาพที่ 13 การเชื่อมต่อระหว่างส่วนประมวลผลและอุปกรณ์Ethenet

ในการเชื่อมต่อระหว่างพอร์ต อีเทอร์เน็ตกับส่วนประมวลผลนั้นจะสามารถทำการเชื่อมต่อได้ดังภาพที่ 13 โดย ขา CLK_REF นั้นจะต้องเชื่อมต่อกับตัวหารค่าความถี่ลง 20 เท่าเพื่อให้สามารถทำงานได้ตามที่ต้องการ ในตารางที่ 2 จะแสดงการเชื่อมต่อระหว่างอุปกรณ์และประเภทของการเชื่อมต่อ สำหรับชิพที่ทำการเชื่อมต่อบนอุปกรณ์ที่มากับบอร์ดทดลองคือ RTL8201CL ทำงานที่ 3.3 V

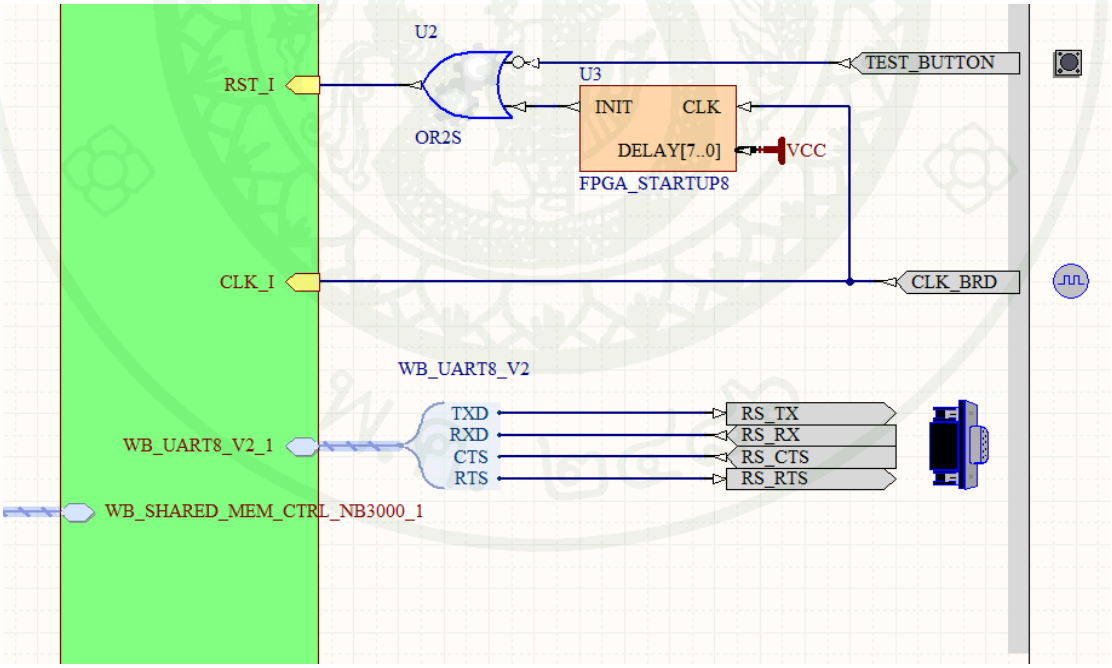
ตารางที่ 2 การเชื่อมต่อระหว่างพอร์ต Ethernet กับ ส่วนประมวลผล

ประเภทของการเชื่อมต่อ	ขาจากพอร์ตEthernet	ขาจากส่วนประมวลผล
Bus	ETH_THD	PHY_TXD
Signal	ETH_TXEN	PHY_TXEN
Signal	ETH_TXC	PHY_TXC
Bus	ETH_RXD	PHY_RXD
Signal	ETH_RXDV	PHY_RXDV
Signal	ETH_RXC	PHY_RXC
Signal	ETH_COL	PHY_COL
Signal	ETH_CRS	PHY_CRS
Signal	ETH_RESETB_E	PHY_RESETB

ตารางที่ 2 (ต่อ)

ประเภทของการเชื่อมต่อ	ขาจากพอร์ตEthernet	ขาจากส่วนประมวลผล
Signal	ETH_MDC	PHY_MDC
Signal	ETH_MDIO	PHY_MDOE
Signal	ETH_MDIO	PHY_MDO
Signal	ETH_MDIO	PHY_MDI
Signal	CLK_REF	CLK_MD

ในส่วนต่อมาจะเป็นการเชื่อมต่อระหว่างพอร์ต RS-232 กับ ส่วนประมวลผล,การเชื่อมต่อระหว่าง Test button ,และการเชื่อมต่อกับสัญญาณนาฬิกา สามารถเชื่อมต่อกันได้ดังภาพที่ 14 โดยการต่อปุ่ม Test button กับตัว FPGA Startup With 8 Bit Delay ต้องการให้หลังจากต้องการกดปุ่มรีเซต จะเกิดการนับจังหวะนาฬิกาไปจำนวน 8 bit ก่อนที่จะสั่งให้ RST_I ทำงาน



ภาพที่ 14 การเชื่อมต่อระหว่างส่วนประมวลผลและพอร์ต RS-232 ,Test Button, Clock Board

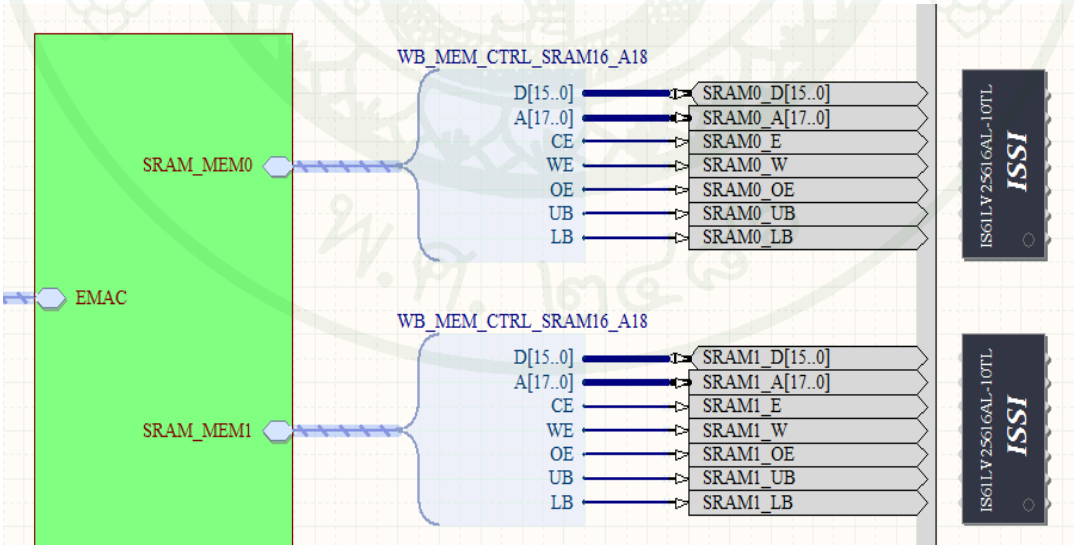
สำหรับการเชื่อมต่อระหว่างพอร์ต RS-232 จะเป็นการเชื่อมต่อระหว่าง UART กับ พอร์ต RS-232 ซึ่งเป็นการส่งข้อมูลแบบอนุกรม โดยเชื่อมต่อพอร์ต TXD ไปยัง RS_TX และเชื่อมต่อ

RXD กับ RS_TX เพื่อให้สามารถรับส่งข้อมูลได้โดยในตารางที่ 3 ได้แสดงการเชื่อมต่อระหว่าง UART กับพอร์ต RS-232 ไว้

ตารางที่ 3 การเชื่อมต่อระหว่างส่วนประมวลผลและพอร์ต RS-232 ,Test Button, Clock Board

ประเภทของการเชื่อมต่อ	ขาจากพอร์ต/ชื่อ	ขาจากส่วนประมวลผล
Signal	RS-232/Rs_TX	TXD
Signal	RS-232/Rs_RX	RXD
Signal	RS-232/Rs_CTS	CTS
Signal	RS-232/Rs_RTS	RTS
Signal	CLK_BRD	CLK_I
Signal	Test_BUTTON	RST_I

การเชื่อมต่อระหว่างหน่วยความจำแบบเอสแรม (SRAM) บนบอร์ดทดลองและส่วนประมวลผลสามารถทำได้ตามภาพที่ 15 หรือ ตารางที่ 4 โดย เอสแรมบนบอร์ดทดลองมี 2 ชิ้นจึงต้องทำการต่อเชื่อมประสานทั้ง 2 ชิ้น

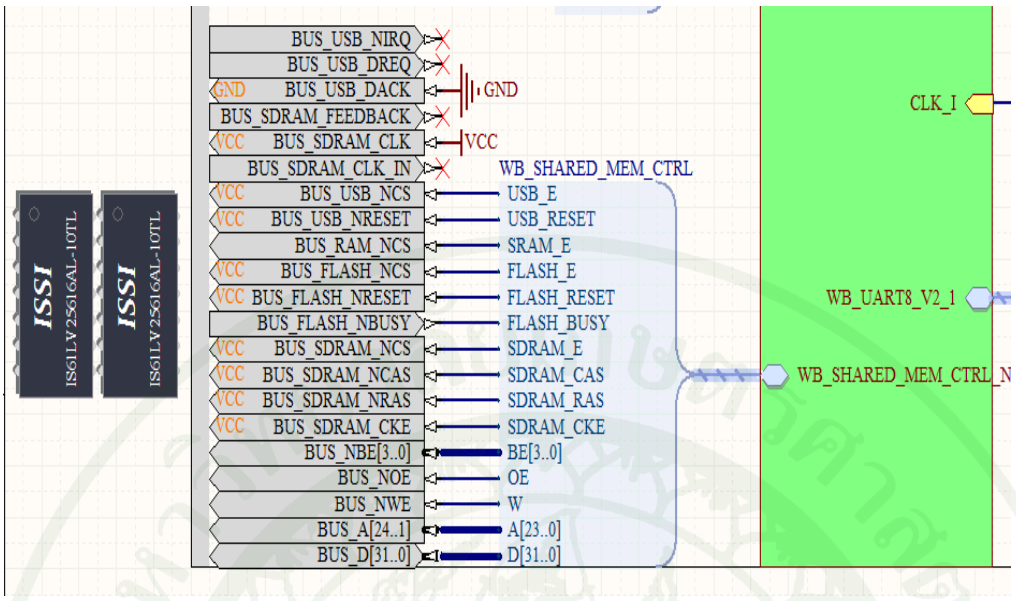


ภาพที่ 15 การเชื่อมต่อระหว่างเอสแรม (SRAM) และส่วนประมวลผล

ตารางที่ 4 การเชื่อมต่อระหว่างเอสแรม (SRAM) และส่วนประมวลผล

ประเภทของการเชื่อมต่อ	ขาจากพอร์ต Sram0/Sram1	ขาจากส่วนประมวลผล
Bus	D	SRAM0_D
Bus	A	SRAM0_A
Signal	CE	SRAM0_E
Signal	WE	SRAM0_W
Signal	OE	SRAM0_OE
Signal	UB	SRAM0_UB
Signal	LB	SRAM0_LB
Bus	D	SRAM1_D
Bus	A	SRAM1_A
Signal	CE	SRAM1_E
Signal	WE	SRAM1_W
Signal	OE	SRAM1_OE
Signal	UB	SRAM1_UB
Signal	LB	SRAM1_LB

ในการเชื่อมต่อระหว่างหน่วยความจำร่วม (Shared memory) และส่วนประมวลผลนั้นในงานวิจัยนี้เลือกใช้เอสแรม เท่านั้นจากระบบ Open bus ที่กำหนดไว้ก่อนแล้ว ดังนั้นจึงจำเป็นต้องเลือกที่จะต้องใช้งานในแต่ละพอร์ตให้ถูกต้องซึ่งสามารถจะเชื่อมต่อได้ตามภาพที่ 16 และ ตารางที่ 5 สำหรับพอร์ตที่ไม่ได้ใช้งานของหน่วยความจำร่วม (Shared memory) จำเป็นที่จะต้องทำการเชื่อมต่อด้วย หากไม่มีการเชื่อมต่อเวลาการทำการแปลโปรแกรม (compile) จะทำให้เกิดความผิดพลาดเกิดขึ้นเนื่องจากโปรแกรมตรวจพบว่าพอร์ตไม่ได้รับการเชื่อมต่อ



ภาพที่ 16 การเชื่อมต่อระหว่างหน่วยความจำร่วม (Shared memory) และส่วนประมวลผล

ตารางที่ 5 การเชื่อมต่อระหว่างหน่วยความจำร่วม (Shared memory) และส่วนประมวลผล

ประเภทของการเชื่อมต่อ	ขาจากพอร์ต Shared memory	ขาจากส่วนประมวลผล
Bus	BUS_D	D
Bus	BUS_A	A
Signal	BUS_NWE	W
Signal	BUS_NOE	OE
Bus	BUS_NBE	BE
Signal	BUS_SDRAM_CKE	SDRAM_CKE
Signal	BUS_SDRAM_NRAS	SDRAM_NRAS
Signal	BUS_SDRAM_NCAS	SDRAM_NCAS
Signal	BUS_SDRAM_NCS	SDRAM_E
Signal	BUS_FLASH_NBUSY	FLASH_BUSY
Signal	BUS_FLASH_NRESET	FLASH_RESET
Signal	BUS_FLASH_NCS	FLASH_E

คารางที่ 5 (ต่อ)

ประเภทของการเชื่อมต่อ	ขาจากพอร์ต Shared memory	ขาจากส่วนประมวลผล
Signal	BUS_RAM_NCS	SRAM_E
Signal	BUS_USB_NRESET	USB_RESET
Signal	BUS_USB_NCS	USB_E
Signal	BUS_SDRAM_CLK_IN	-
Signal	BUS_SDRAM_CLK	VCC
Signal	BUS_SDRAM_FEEDBACK	-
Signal	BUS_USB_DACK	GND
Signal	BUS_USB_DREQ	-
Signal	BuS_USB_NIRQ	-

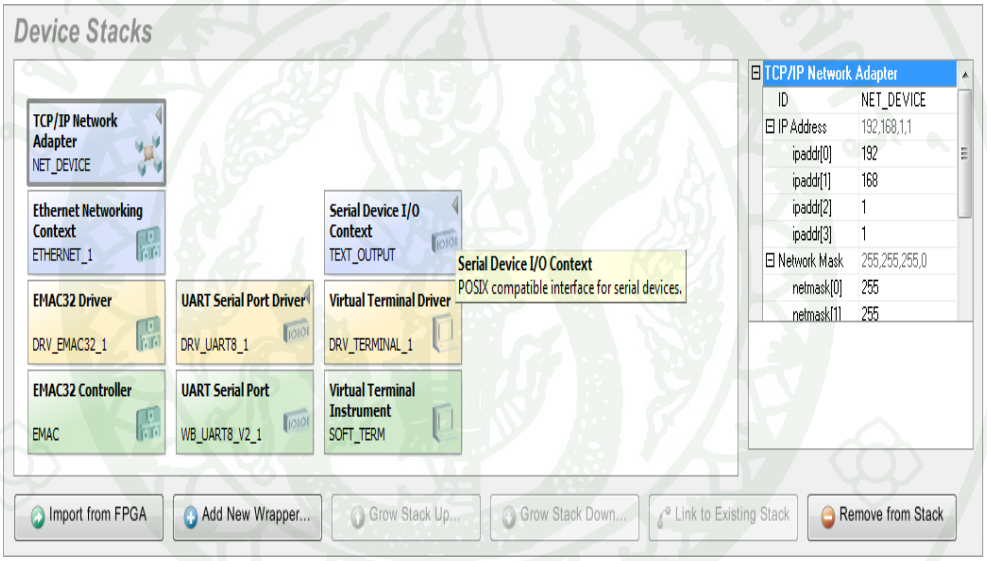
เมื่อทำการเชื่อมต่อทั้งหมดขั้นตอนต่อไปจะทำการแปลโปรแกรม (compile) ข้อมูลที่ได้ทำการเชื่อมต่อเพื่อหาส่วนที่ผิดพลาดและทำการแก้ไขให้มีความถูกต้องก่อนจะนำไปใช้งานจริง

2. การสร้างส่วนเชื่อมต่อระหว่างคอมพิวเตอร์ส่วนบุคคลและบอร์ดทดลอง FPGA

เมื่อทำการเชื่อมประสานระหว่างส่วนประมวลผลและส่วนอุปกรณ์เสร็จเรียบร้อยแล้ว ขั้นตอนต่อไปจะเป็นการกำหนดดีไวส์สแตก (Device stack) ซึ่งเป็นส่วนที่จะช่วยในการเขียนโปรแกรมสำหรับการเชื่อมต่อระหว่างคอมพิวเตอร์ส่วนบุคคลและบอร์ดทดลอง FPGA ในการทำการกำหนดลำดับขั้นการทำงานของอุปกรณ์จะทำได้โดยเริ่มจากการนำเข้าการออกแบบ FPGA ที่ได้สร้างไว้แล้ว มากำหนดลำดับขั้นของการทำงาน ซึ่งในแต่ละขั้นนั้นจะประกอบด้วยคำสั่งควบคุมในการทำงานได้ต่างกันออกไป โดยจะสามารถกำหนดคุณสมบัติของอุปกรณ์ในแต่ละชนิดให้ตามที่ต้องการได้ ในงานวิจัยชิ้นนี้จะใช้อุปกรณ์ 2 ชิ้นในการเชื่อมต่อคือ อีเทอร์เน็ต และ RS-232 โดย อีเทอร์เน็ตจะเชื่อมต่อกับคอมพิวเตอร์ส่วนบุคคลเพื่อนนำข้อมูลมาเข้ารหัสและใช้ส่งข้อมูลหลังจากถอดรหัสไปเปรียบเทียบผลที่ได้ และ RS-232 สำหรับเชื่อมต่อกันอุปกรณ์รับ-ส่งสัญญาณแบบไร้สาย เมื่อกำหนดคุณสมบัติต่าง ๆ แล้วลำดับต่อไปจะทำการเขียนโปรแกรมควบคุมภาษา C เพื่อเชื่อมต่อระหว่างคอมพิวเตอร์ส่วนบุคคลและบอร์ดทดลอง FPGA

2.1 การกำหนดค่าคุณสมบัติของอีเทอร์เน็ต(Ethernet)

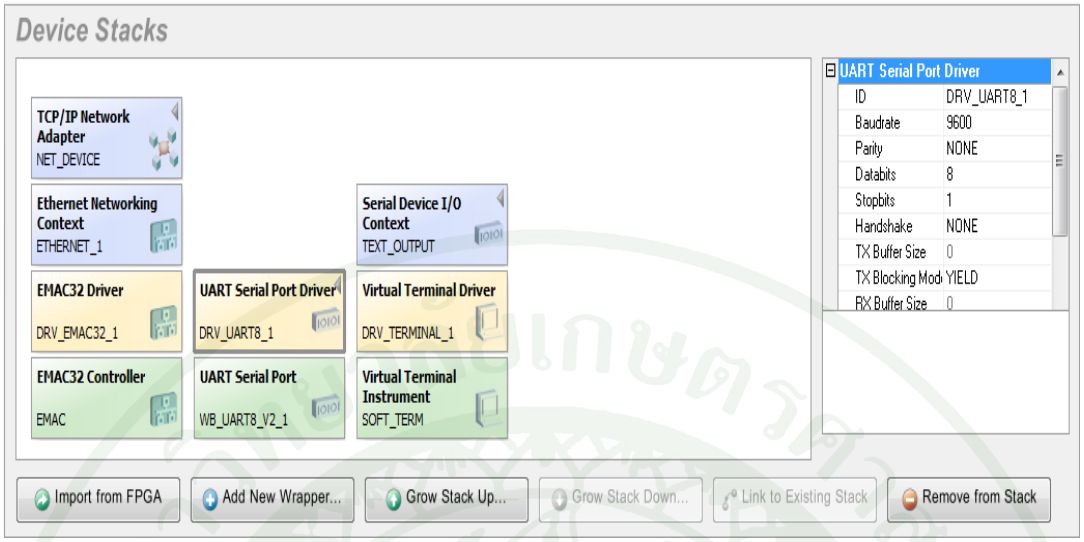
ในการกำหนดค่าคุณสมบัติของอีเทอร์เน็ต ในส่วนดีไวส์สแตก (Device Stack) จะสามารถกำหนดได้ตั้งแต่ลำดับขั้นของแมคแอดเดรส (Mac address) ในชั้นของ EMC32DRIVER จากภาพที่ 17 ซึ่งจะกำหนดให้เป็น 2,34,45,56,67,78 โดยแต่ละค่านั้นจะแทนด้วยเลขฐาน 16 และกำหนดให้การรับส่งค่าทำการบัฟเฟอร์ข้อมูลลงในเอสแรม สำหรับในชั้น TCP/IP Network Adaptor กำหนดให้ไอพีแอดเดรส (IP Address) ของบอร์ดทดลองเป็น 192.168.1.1 กำหนด Network Mask 255.255.255.0 เพื่อใช้งานต่อไป



ภาพที่ 17 การกำหนดคุณสมบัติสำหรับ Ethernet

2.2 การกำหนดค่าคุณสมบัติสำหรับ RS-232

การกำหนดค่าสำหรับ RS-232 นั้นจะเริ่มจากทำการกำหนด Baud rate ให้กับ UART Serial Port Driver โดยกำหนดที่ความเร็ว 9600 แบบไม่มีการใช้ parity check bit และกำหนดขนาดของข้อมูลเป็น 8 บิต จากนั้นทำการกำหนดค่า Stop bit เป็น 1 บิต โดยคุณสมบัติต่าง ๆ สามารถกำหนดได้ตามภาพที่ 18 สำหรับการฟังก์ชัน Handshake ของพอร์ตอนุกรมจะไม่มีการใช้งานสำหรับงานวิจัยนี้



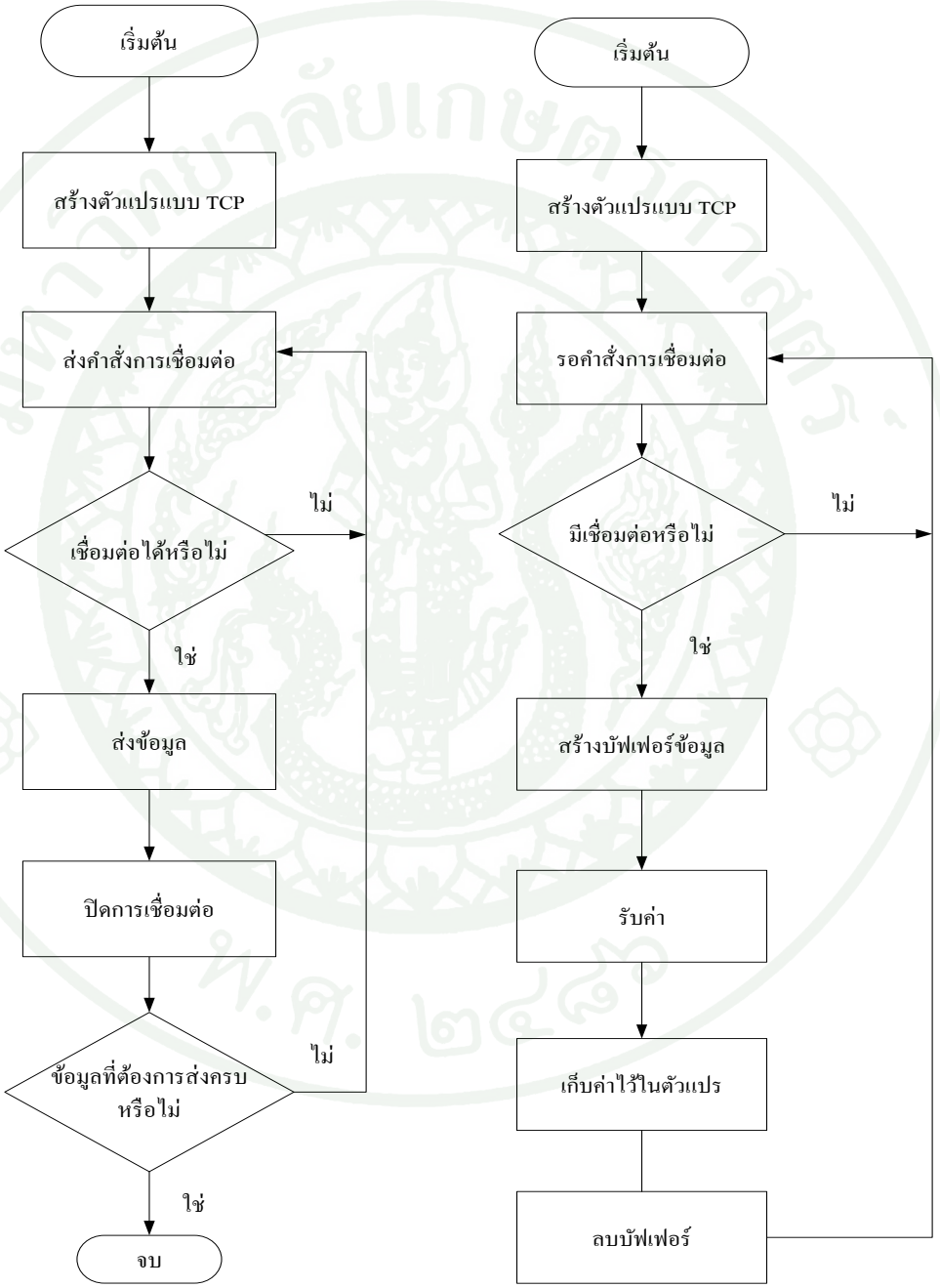
ภาพที่ 18 การกำหนดคุณสมบัติสำหรับ RS-232

2.3 การเขียนโปรแกรมสำหรับเชื่อมต่อประสาน

ในการเขียนโปรแกรมสำหรับเชื่อมต่อประสานนั้นจะแบ่งออกเป็น 2 ส่วนคือการเขียนโปรแกรมภาษาซีเพื่อควบคุมการทำงานบนบอร์ดทดลอง และการเขียนโปรแกรมบน Matlab เพื่อรับ-ส่งข้อมูลจากคอมพิวเตอร์ส่วนบุคคล ในการเขียนโปรแกรมบน Matlab เริ่มจากการตั้งค่า IP address ของคอมพิวเตอร์ส่วนบุคคลให้เป็น 192.168.1.2 และ Subnet mask เป็น 255.255.255.0 เพื่อให้สามารถสื่อสารกับบอร์ดทดลองได้ หลังจากนั้นจะเริ่มการเขียนโปรแกรม Matlab โดยเริ่มจากการสร้างตัวแปรที่เป็น TCP/IP object ซึ่งจะเป็นตัวแปรที่ใช้สำหรับเก็บค่าต่าง ๆของประเภทการเชื่อมต่อแบบ TCP/IP ไว้ หลังจากนั้นทำการเปิดการเชื่อมต่อระหว่างบอร์ดทดลองและโปรแกรม Matlab ด้วยคำสั่ง fopen เพื่อให้ทดสอบดูว่าสามารถเปิดการเชื่อมต่อระหว่างบอร์ดทดลองและโปรแกรม Matlab ได้หรือไม่ หากเปิดการเชื่อมต่อได้แล้วก็จะสามารถส่งข้อมูลในรูปแบบต่าง ๆ ไปยังบอร์ดทดลองต่อไปได้ หากเปิดการเชื่อมต่อไม่ได้โปรแกรม Matlab จะทำการแสดงผลลัพธ์ที่เป็นความผิดพลาดออกมา

ในการเขียนโปรแกรมควบคุมบนบอร์ดทดลองจะทำการสร้างตัวแปรสำหรับเชื่อมต่อระหว่างพอร์ตอีเทอร์เน็ตไว้หลังจากที่สร้างตัวแปรเสร็จจะทำการสั่งให้บอร์ดทดลองรอจนกว่าจะได้รับการขอการเชื่อมต่อจากโปรแกรม Matlab เมื่อได้รับการขอการเชื่อมต่อแล้วบอร์ดทดลองจะทำการสร้างบัฟเฟอร์เพื่อรอรับข้อมูลที่จะส่งมาจาก Matlab เมื่อได้รับข้อมูลจะทำการเก็บ

ค่าไว้ในหน่วยความจำที่กำหนดไว้ เมื่อเก็บค่าข้อมูลแล้วบัฟเฟอร์ข้อมูลจะถูกลบออกไปเพื่อรอให้มีการเชื่อมต่อครั้งต่อไปจึงสร้างบัฟเฟอร์ขึ้นมาใหม่มารับค่าโดยการทำงานของการควบคุมทั้งสองฝั่งสามารถทำงานได้ตามภาพที่ 19



ภาพที่ 19 แผนผังการส่งข้อมูลจาก Matlab และ การรับข้อมูลของบอร์ดทดลอง

3. โปรแกรมการเข้ารหัสและการทดสอบ

การเข้ารหัสที่สนใจในงานวิจัยนี้เป็นการเข้ารหัสแบบต่อร่วมโดยจะแบ่งออกเป็น 2 ส่วน คือ ส่วนของการเข้ารหัสภายในและการเข้ารหัสภายนอก ปัจจัยที่ต้องคำนึงถึงเป็นอย่างมากในการสร้างโปรแกรมการเข้ารหัสคือขนาดของข้อมูลซึ่งต้องส่งผ่านจากรหัสภายในไปยังรหัสภายนอกจึงจำเป็นที่จะต้องมีการกำหนดขนาดของข้อมูลเพื่อความเหมาะสมในการเข้ารหัส

3.1 การเข้ารหัสภายนอก

การเข้ารหัสภายนอกที่สนใจในงานวิจัยชิ้นนี้คือ รหัสคอนโวลูชัน (3,2,2) จากสมการที่ (14) ซึ่งเป็นรหัสแบบนอนไบนารีที่กำหนดขนาดของสัญลักษณ์ไว้ที่ 102 บิตต่อสัญลักษณ์ จำนวน 14 สัญลักษณ์ แล้วจึงแยกสัญลักษณ์ออกมาเพื่อใช้เป็นข้อมูลสำหรับการทดสอบที่ต้องแยกสัญลักษณ์ออกมาเนื่องจากรหัสนั้นมีการรับข้อมูลเข้ามา 2 อินพุตต่อการเข้ารหัส 1 ครั้ง หลังจากทำการเข้ารหัสข้อมูลเสร็จต้องทำการจัดกับข้อมูลไว้ในหน่วยความจำด้วย เนื่องจากการเข้ารหัสเป็นแบบการเข้ารหัสที่มีความจำ ซึ่งผลลัพธ์ที่ได้จากการเข้ารหัสภายนอกแล้วจะได้ผลลัพธ์ออกมาเป็น 21 สัญลักษณ์ สำหรับการทดสอบเพื่อความถูกต้องของการเข้ารหัสภายในบอร์ดทดลอง FPGA จะทำได้โดยการเทียบผลลัพธ์กับโปรแกรม C++ เพื่อทดสอบความถูกต้อง

3.2 การเข้ารหัสภายใน

การเข้ารหัสภายในสำหรับงานวิจัยนี้จะเลือกทดสอบการเข้ารหัสด้วย 2 วิธีคือการเข้ารหัสด้วยรหัสบีซีเฮช (31,26) และรหัสรีดโซโลมอน (63,51) สำหรับการเข้ารหัสภายในแบบบีซีเฮช (31,26) ข้อมูลที่ผ่านเข้าไปทำการเข้ารหัสจะทำในระดับบิตแบ่งเป็นบล็อค บล็อคละ 26 บิต เมื่อทำการเข้ารหัสเสร็จเรียบร้อยจะไม่มีการเก็บค่าเนื่องจากเป็นรหัสที่ไม่มีความจำ ในการทดสอบผลลัพธ์จะทำการเปรียบเทียบความถูกต้องกับโปรแกรม C++ ในคอมพิวเตอร์ส่วนบุคคลต่อไป

การเข้ารหัสภายในแบบรีดโซโลมอน (63,51) จะเป็นการเข้ารหัสแบบนอนไบนารีบนพื้นฐาน $GF(2^6)$ กล่าวคืออินพุต 1 ค่ามีขนาด 6 บิต ต่อ 1 บล็อกข้อมูล จำนวนอินพุตทั้งหมดคือ 51 สัญลักษณ์ ผลลัพธ์ที่ได้ออกมา เป็น 63 สัญลักษณ์ โดยการทดสอบจะทำการแปรียบเทียบค่าที่ได้จากโปรแกรม C++ และ บอร์ดทดลอง

4. โปรแกรมการถอดรหัสและการทดสอบ

โปรแกรมถอดรหัสจะประกอบไปด้วย 2 ส่วนคือ ส่วนการถอดรหัสภายในแบบลิสออฟทูซอฟท์วิเทอร์บี,การถอดรหัสภายในด้วยวิธีการของ Berlekamp-Massey สำหรับการเข้ารหัสภายในรีดโซโลมอน (63,51) และการถอดรหัสภายนอกแบบ เวกเตอร์ซิมโบล

4.1 การถอดรหัสภายใน

การถอดรหัสภายในที่ใช้การถอดรหัสแบบลีสออฟทูวิเทอร์บี เป็นการถอดรหัสที่จะเลือกเก็บตัวเลือกที่ถูกต้องที่สุด 2 ตัวเลือกไว้ ที่ต้องทำการเก็บตัวเลือกที่ถูกต้องที่สุด 2 ตัวเลือกไว้นั้นเนื่องจากตัวเลือกที่ 2 สามารถที่จะนำไปช่วยคำนวนในการแก้ไขความผิดพลาดในการถอดรหัสภายนอกได้ สำหรับงานวิจัยชิ้นนี้จะเลือกทดลองทั้ง 2 แบบคือ การตัดสินใจแบบฮาร์ดที่จะใช้การตัดสินใจว่าข้อมูลในแต่ละบิต มีค่าเพียง2ระดับขั้นคือ 0 หรือ 1 เท่านั้น ส่วนการตัดสินใจแบบซอฟท์ นั้นจะมีตัดสินที่มีระดับของสัญญาณที่ละเอียดมากกกว่า 2 ระดับขั้น ในส่วนของการทำการถอดรหัสนั้นจะทำการรับข้อมูลที่มีขนาด 31 บิตแล้วส่งผ่านข้อมูลไปยังฟังก์ชันการการถอดรหัสแบบลิสออฟทูวิเทอร์บี หลังจากนั้นตัวฟังก์ชันจะคืนค่าออกมา 2 ค่าเป็นค่าทางเลือกลำดับแรก และค่าทางเลือกลำดับที่สองจะทำการเก็บไว้เพื่อส่งไปคำนวนในการถอดรหัสภายนอกต่อไป

ลำดับต่อมาจะเป็นการทดสอบการถอดรหัสรีดโซโลมอน (63,51) ด้วยวิธีการของ Berlekamp-Masseyโดยมีขนาด 6 บิตต่อ 1 สัญลักษณ์จำนวน 63 สัญลักษณ์ผลลัพธ์ที่ได้ 51 สัญลักษณ์ ในการทดสอบจะทำการเปรียบเทียบคำรหัสจากบอร์ดทดลอง FPGA และโปรแกรม C++ ว่ามีความถูกต้องของการถอดรหัสข้อมูลหรือไม่

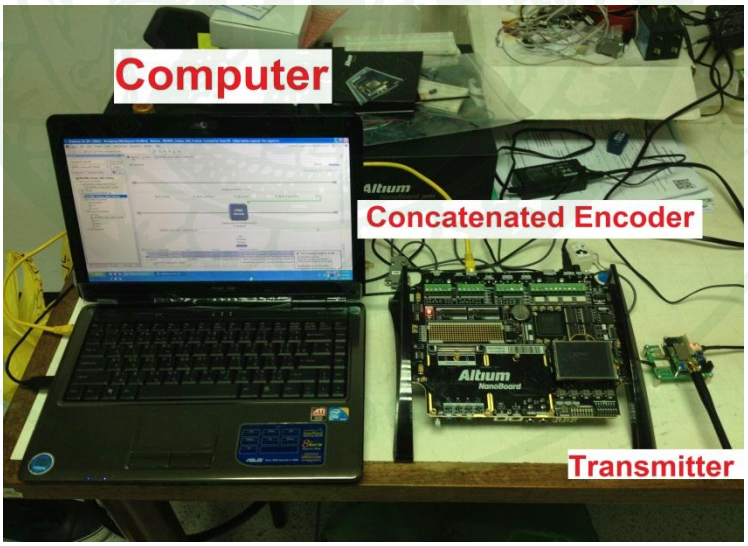
4.2 การถอดรหัสภายนอก

การถอดรหัสภายนอกจะทำการถอดรหัสด้วยวิธีการเวกเตอร์ซิมโบล (Vector Symbol Decoding) ซึ่งในส่วนแรกจะทำการรับค่าอินพุตเข้ามา 2 ค่าคือค่าทางเลือกลำดับแรกและค่าทางเลือกลำดับที่สอง จากนั้นเมื่อรับค่ามาจะทำการส่งผ่านข้อมูลทั้งสองชุดไปยังฟังก์ชัน เพื่อใช้ในการถอดรหัสข้อมูลต่อไป หลังจากส่งผ่านข้อมูลเข้ามาในฟังก์ชันแล้วจะทำการแก้ไขข้อมูลให้มีความถูกต้องด้วยวิธีการคำนวณซินโดรมเพื่อตรวจหาความผิดพลาด การใช้ตัวเลือกที่สองในการ

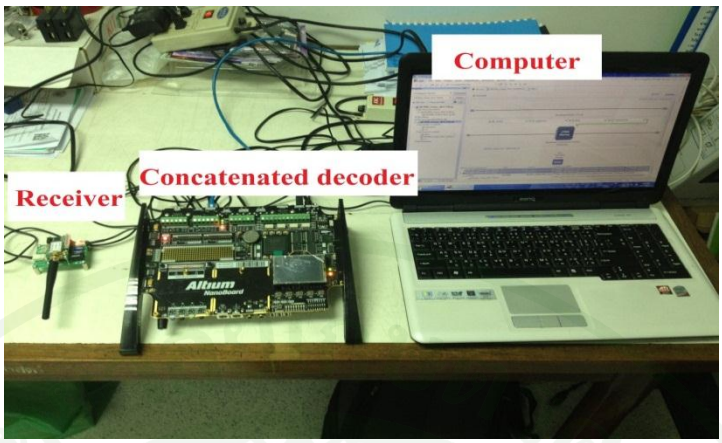
ช่วยแก้ไขข้อมูล การใช้ผลรวมศูนย์เพื่อแก้ไขข้อมูลให้ถูกต้องหลังจากแก้ไขข้อมูลเสร็จจะคืนค่าที่ทำการแก้ไขออกมา เมื่อเพื่อทำการดึงข้อมูลกลับให้ได้ข้อมูลต้นฉบับที่ส่งมา

5. ส่วนโมดูลรับ-ส่งสัญญาณไร้สาย

ส่วนรับส่งสัญญาณนั้นจะเลือกใช้การเชื่อมต่อกับพอร์ต RS-232 บนบอร์ดทดลอง FPGA โดยใช้อุปกรณ์ส่งสัญญาณ D-RFN96 จากบริษัท Sila Research โดยอุปกรณ์ชิ้นนี้เป็นอุปกรณ์ที่สามารถรับส่งสัญญาณแบบไร้สายผ่านพอร์ตอนุกรม RS-232 ที่อัตราบอร์ด 9600 บอร์ดต่อวินาที โดยใช้งานที่ช่วงความถี่ 430-433 MHz ซึ่งเป็นความถี่สาธารณะ สำหรับช่องความถี่ในการรับส่งที่สามารถตั้งค่าได้มีทั้งหมด 8 ช่อง โดยการติดต่อสื่อสารจะต้องตั้งค่าเพียง 1 ช่องความถี่ที่ต้องการใช้ของอุปกรณ์รับ-ส่งให้ตรงกันเพื่อให้อุปกรณ์สามารถสื่อสารได้ สำหรับงานวิจัยชิ้นนี้จะใช้ช่องความถี่ที่ 1 โดยการตั้งค่าบนโมดูลไว้เป็น 000 หรือการใส่จัมเปอร์ไว้บนโมดูลทั้งหมด สำหรับการเชื่อมต่อระหว่างบอร์ดทดลองแสดงในภาพที่ 20 สำหรับตัวส่งสัญญาณและในภาพที่ 21 สำหรับตัวรับสัญญาณ



ภาพที่ 20 ภาพการเชื่อมต่อระหว่างตัวส่งสัญญาณไร้สายและบอร์ดทดลอง

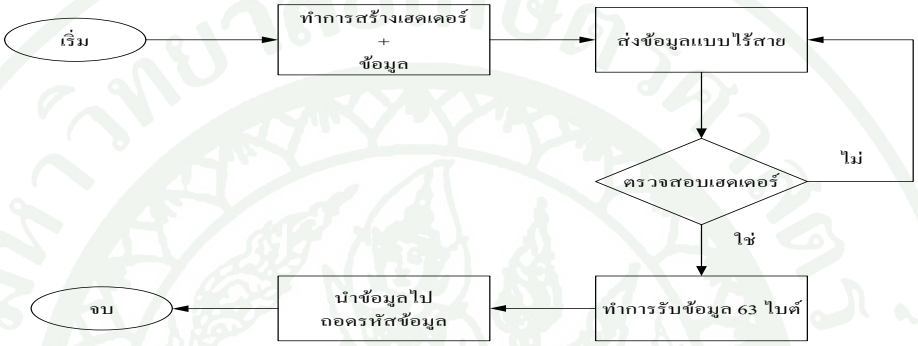


ภาพที่ 21 ภาพการเชื่อมต่อระหว่างตัวรับสัญญาณไร้สายและบอร์ดทดลอง

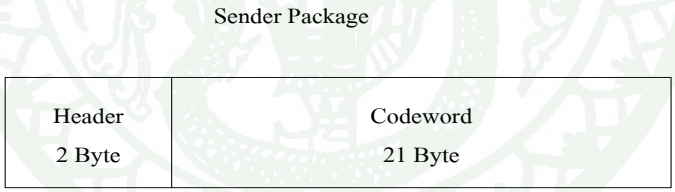
ในงานวิจัยนี้เลือกใช้อุปกรณรับส่ง D-RFN96 เนื่องจากโมดูลรับส่งสัญญาณไร้สายชนิดนี้เป็นโมดูลที่เหมาะสำหรับการส่งข้อมูลโดยตรงที่ไม่ต้องการนำข้อมูลไปเขียนโปรโตคอลในการรับส่งเองและไม่ต้องทำการเชื่อมต่อกันก่อน (Connectionless) ที่จะทำการรับส่ง ซึ่งในโมดูลบางตัวหากการเชื่อมต่อขาดหายไปข้อมูลที่รับ-ส่งกันก็จะขาดหายไปด้วย ซึ่งทำให้เหมาะกับงานวิจัยชิ้นนี้ที่เป็นการส่งคำรหัสที่ออกจากพอร์ตอนุกรมโดยตรง โดยโมดูลจะเสมือนทำหน้าที่แปลงข้อมูลจากพอร์ตอนุกรมเป็นคลื่นวิทยุในย่านความถี่ 433 MHz และทำการรับส่งแบบ สองทางครึ่งอัตรา (Half-Duplex) และมีการเชื่อมประสานใช้งานกับพอร์ตอนุกรม RS-232 โดยไม่ต้องทำการปรับค่าแรงดันซึ่งโมดูลจะทำการปรับค่าแรงดันด้วยชิพ MAX232 สำหรับพอร์ตอนุกรม RS-232 แรงดันเป็นส่วนสำคัญ ซึ่งโมดูลไร้สายทั่วไปส่วนใหญ่จะทำการรับข้อมูลที่แรงดันประมาณ 3.3 - 5.5 โวลต์ สำหรับโมดูลไร้สาย D-RFN96 นี้ยังมีส่วนของการใช้การตรวจสอบความผิดพลาดภายในโดยการใช้ CRC-16 เพิ่มเข้าไปในแพกเกจ แต่ในการใช้งานจริงการใช้รหัสช่องสัญญาณแบบต่อร่วมกับการตรวจสอบความผิดพลาดแบบต่าง ๆ นั้นก็จะช่วยเพิ่มประสิทธิภาพให้กับระบบได้

ในการศึกษาถึงประสิทธิภาพในการถอดรหัสข้อมูลในรหัสช่องสัญญาณแบบต่อร่วมและอุปกรณ์นี้สามารถรับข้อมูลที่เกิดความผิดพลาดเกิดขึ้นได้ ซึ่งในอุปกรณ์บางชนิดจำเป็นที่จะต้องทำการสร้างการเชื่อมต่อก่อนที่จะรับส่งข้อมูลและเมื่อรับข้อมูลที่ผิดพลาดจะทำการทิ้งข้อมูลที่ได้ไป แต่ว่าอุปกรณ์ชิ้นนี้ไม่จำเป็นต้องสร้างการเชื่อมต่อและรับความผิดพลาดที่เกิดขึ้นได้บ้าง สำหรับความผิดพลาดที่เกิดขึ้นจากการรับส่งข้อมูลผ่านอุปกรณ์ไร้สายในงานวิจัยนี้จะทำการทดสอบโดยการถอดเสาสัญญาณของอุปกรณ์และทำการรบกวนโดยการใช้สิ่งกีดขวางและทดสอบในระยะ 2 เมตร

สำหรับการรับ-ส่งข้อมูลด้วยอุปกรณ์แบบไร้สายจะเป็นไปตามขั้นตอนดังภาพที่ 22 จะทำการสร้างแพกเกจในการส่งดังภาพที่ 23 โดยมีเฮดเดอร์ที่มีขนาด 2 ไบต์ เพื่อใช้ระบุถึงข้อมูลที่จะไปถึงฝั่งรับโดยบอร์ดทดลอง FPGA จะทำการตรวจสอบข้อมูลว่าข้อมูลที่รับมาตรงกับเฮดเดอร์ที่อยู่บนฝั่งรับหรือไม่ ถ้าตรงกับฝั่งรับแล้วจะทำการเก็บข้อมูลจำนวน 21 ไบต์ จำนวน 3 ครั้งเพื่อนำไปถอดรหัสภายในต่อไป



ภาพที่ 22 แผนผังขั้นตอนในการส่งข้อมูล

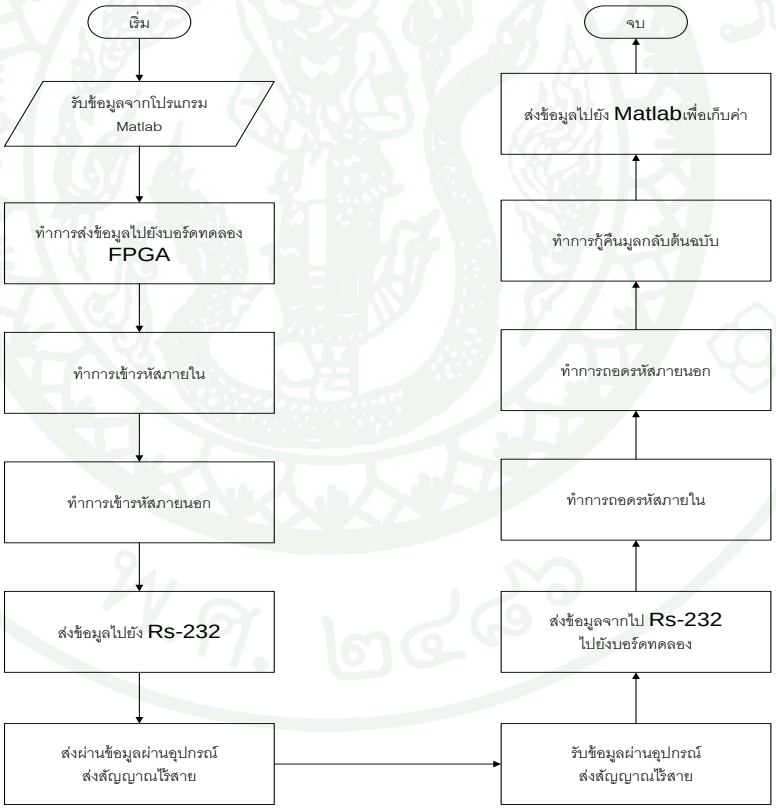


ภาพที่ 23 แพกเกจในการรับส่งข้อมูล

6. การทดสอบระบบรวมทั้งหมด

สำหรับการทดสอบระบบรวมจะใช้การเข้ารหัสภายนอกคือรหัสคอนโวลูชัน(3,2,2) ต่อร่วมกับการเข้ารหัสภายในคือรหัสรีดโซโลมอน (63,51) และ การถอดรหัสภายในโดยการใช้วิธีการของ Berlekamp–Massey สำหรับการเข้ารหัสภายในแบบรีดโซโลมอนต่อร่วมกับการถอดรหัสภายนอกแบบเวกเตอร์ซิมโบลดี โดยการใช้รหัสภายในแบบรหัสรีดโซโลมอน (63,51) สามารถแก้ไขความผิดพลาดได้ดีกว่ารหัสแบบบีซีเฮช (31,26) จึงเลือกวิธีการเข้ารหัสแบบรีดโซโลมอนสำหรับการทดสอบ

ในการทดสอบระบบบรหัสแบบต่อร่วมนี้จะมีลำดับขั้นตอนดังภาพที่ 24 โดยเริ่มต้นคอมพิวเตอร์ฝั่งส่งทำการส่งข้อมูลจากโปรแกรม Matlab ไปยังบอร์ดทดลอง FPGA ผ่านพอร์ตอีเทอร์เน็ต ซึ่งมีโปรแกรมการเข้ารหัสภายนอกและการเข้ารหัสภายในบรรจุไว้ จากนั้นทำการเข้ารหัสภายนอกและเข้ารหัสภายใน เมื่อทำการเข้ารหัสเสร็จสิ้นผลลัพธ์ที่ได้จะเป็นคำรหัส ต่อมาจะทำการส่งคำรหัสไปยังพอร์ต RS-232 ที่เชื่อมต่อกับอุปกรณ์ส่งสัญญาณไร้สาย ทางฝั่งรับรอรับสัญญาณจากฝั่งส่งเมื่อได้รับข้อมูลจากฝั่งส่งครบแล้วจะทำการผ่านค่าไปยังส่วนของการถอดรหัส ที่ประกอบไปด้วยการถอดรหัสภายในและการถอดรหัสภายนอกตามลำดับ จากนั้นดึงข้อมูลกลับต้นฉบับ ส่งข้อมูลผ่านไปยังพอร์ตอีเทอร์เน็ตไปยังโปรแกรม Matlab ทางฝั่งรับ ทำการเก็บค่าข้อมูลเมื่อส่งข้อมูลหมดทุกส่วนแล้วจึงทำการเปรียบเทียบค่าที่ได้รับจากฝั่งรับและฝั่งส่ง โดยรายละเอียดของขั้นตอนต่าง ๆ มีดังนี้



ภาพที่ 24 ขั้นตอนการส่ง-รับข้อมูลของระบบ

สำหรับรายละเอียดการเชื่อมต่อมีดังนี้ เริ่มต้นจะทำการสร้างอ๊อบเจกต์ TCP/IP ขึ้นมาในโปรแกรม Matlab สำหรับการสร้างอ๊อบเจกต์ชนิดนี้เพื่อการสร้างการสื่อสารข้อมูลแบบ TCP/IP กับบอร์ดทดลองผ่านพอร์ตอีเทอร์เน็ตโดยตัวอย่างการสร้างคือ

T = tcipip(‘192.168.1.1’,9090,’LocalPort’,9090) โดยมีพารามิตเตอร์ต่าง ๆ ดังนี้

1. พารามิเตอร์ ‘192.168.1.1’ จะเป็นค่าแอดเดรสของบอร์ดทดลองที่จะทำการเชื่อมต่อกับโปรแกรม Matlab
2. พารามิเตอร์ 9090 เป็นพอร์ตที่ทำการติดต่อทั้งนี้จำเป็นต้องมีการเปิดกำหนดพอร์ตเพื่อเป็นช่องทางเฉพาะในการติดต่อของบอร์ดทดลอง
3. พารามิเตอร์ ‘LocalPort’ เป็นค่าคุณสมบัติถึงการกำหนดพอร์ตของคอมพิวเตอร์ที่ใช้ติดต่อกับบอร์ดทดลอง
4. พารามิเตอร์ 9090 เป็นการเปิดพอร์ตของคอมพิวเตอร์เพื่อเป็นช่องทางในการเชื่อมต่อระหว่างคอมพิวเตอร์และบอร์ดทดลอง

เมื่อทำการสร้างอ๊อบเจกต์ในบอร์ดทดลองแล้วต่อมาจะต้องการเปิดอ๊อบเจกต์ขึ้นมาเพื่อการใช้งานและเชื่อมต่อกับบอร์ดทดลอง สำหรับการเปิดอ๊อบเจกต์จะใช้คำสั่ง fopen(T) โดยพารามิเตอร์ T ในวงเล็บจะเป็นอ๊อบเจกต์แบบ TCP/IP ที่ได้ทำการสร้างและกำหนดค่าไว้แล้ว ซึ่งหากสามารถทำการเชื่อมต่อระหว่างบอร์ดทดลองและโปรแกรม Matlab ได้อ๊อบเจ๊กต์ T จะมีสถานะ Open และบอร์ดทดลองจะขึ้นคำว่า accept new connection หลังจากทำการเชื่อมต่อเรียบร้อยแล้วจะสามารถส่งข้อมูลผ่านคำสั่ง fwrite ซึ่งเป็นการส่งค่าข้อมูลไปยังอ๊อบเจกต์ต่าง ๆ ได้ยกตัวอย่างเช่น fwrite(T,a(1:182),’char’) สามารถอธิบายพารมิตเตอร์ต่าง ๆ ได้ดังนี้

1. พารามิเตอร์ T เป็นพารามิเตอร์ที่ได้จากการใช้คำสั่ง fopen เท่านั้น
2. พารามิเตอร์ a(1:182) เป็นพารามิเตอร์สำหรับการส่งข้อมูลจากตัวอย่างเป็นการส่งข้อมูลแบบในตัวแปล a โดย มีขนาด 182 ไบต์ไปยังพารามิเตอร์ t แบบ TCP/IP
3. พารามิเตอร์ ‘char’ เป็นพารามิเตอร์ที่บอกว่าข้อมูลที่ต้องการส่งเป็นรูปแบบไหน สำหรับในการทดสอบจะส่งในรูปแบบของตัวอักษรหรือสัญลักษณ์ที่มีขนาด 8 บิต

หลังจากได้รับการเชื่อมต่อจากคอมพิวเตอร์แล้วทางบอร์ดทดลองจะสร้างบัฟเฟอร์และรอรับข้อมูลด้วยฟังก์ชัน netconn_recv โดยเมื่อได้รับข้อมูลแล้วจะคืนค่า ERR_OK ซึ่งเมื่อทำการรับค่าแล้วจะทำการนำบัฟเฟอร์มาใช้โดยฟังก์ชัน netbuf_data ฟังก์ชันนี้จะนำข้อมูลไปเก็บไว้ในอะเรย์ที่เราสร้างขึ้นเพื่อนำไปใช้เข้ารหัสโดยจะสามารถเช็คความยาวของข้อมูลที่ได้รับมาด้วย หลังจากทำการเก็บค่าจากบัฟเฟอร์ไว้ในข้อมูลแบบอะเรย์แล้วจะทำการเคลียร์บัฟเฟอร์เพื่อรอรับข้อมูลใหม่ด้วยฟังก์ชัน netbuf_delete หลังจากนั้นจะทำการดึงค่าข้อมูลในอะเรย์ของข้อมูลมาเข้ารหัสโดยในการทดสอบนี้จะทำการรับค่าทีละ 10 สัญลักษณ์และเคลียร์ข้อมูลอีก 4 สัญลักษณ์สัญลักษณ์ละ 102

บิต ก่อนเข้ารหัสจะทำการแปลงข้อมูลที่รับมาทำเป็นบิตอะเลย์ด้วยฟังก์ชัน char2bit หลังจากนั้นจะนำไปเข้ารหัสภายนอกแบบคอนโวลูชัน (3,2,2) โดยจะทำการเข้ารหัสทีละ 2 สัญลักษณ์ด้วยฟังก์ชัน encoder_outer เมื่อทำการเข้ารหัสภายนอกเสร็จแล้วจะนำข้อมูลแบบบิตอะเลย์มาทำการแปลงเป็นรูปแบบสัญลักษณ์แบบรีดโซโลมอนโดยจะแปลงเป็น 6 บิตต่อสัญลักษณ์จำนวน 51 สัญลักษณ์ผ่านฟังก์ชัน outer2rs และทำการเข้ารหัสภายในแบบรีดโซโลมอน (63,51) ด้วยฟังก์ชัน main_rs เมื่อทำการเข้ารหัสช่องสัญญาณเสร็จแล้วจะทำการส่งข้อมูลไปยังพอร์ต RS-232 ที่มีการเชื่อมต่อกับโมดูลไร้สายด้วยคำสั่ง send_rs232 จะทำการเข้ารหัสไปจนข้อมูลในอะเลย์ข้อมูลที่รับมาครับแล้วจึงไปนำข้อมูลในบัฟเฟอร์ที่รับรับข้อมูลมาเพิ่ม

ในส่วนของฝั่งรับจะต้องทำการเปิดการเชื่อมต่อระหว่างบอร์ดทดลองกับโปรแกรม Matlab ซึ่งจะทำในลักษณะเดียวกันกับฝั่งส่ง เมื่อทำการเชื่อมต่อเรียบร้อยแล้วทางฝั่งรับจะรอรับข้อมูลจากพอร์ตอนุกรม RS_232 โดยฟังก์ชัน rx โดยจะเก็บข้อมูลมาเรื่อย ๆ จนครบ 7 คำรหัสคำรหัสละ 63 สัญลักษณ์ สัญลักษณ์ละ 6 บิต เมื่อได้ข้อมูลมาครบแล้วจะทำการถอดรหัสแบบด้วย Berlekham-Massey ด้วยฟังก์ชัน main_rs หลังจากถอดรหัสแล้วข้อมูลที่ได้จะทำการแปลงเป็นบิตอะเลย์เพื่อนำไปใช้สำหรับการถอดรหัสภายนอกแบบเวกเตอร์ซิมโบลด้วยฟังก์ชัน rs2vsd จากนั้นจะนำข้อมูลแบบบิตอะเลย์ไปถอดรหัสแบบเวกเตอร์ซิมโบลด้วย main_vsd เมื่อทำการถอดรหัสด้วยเวกเตอร์ซิมโบลเรียบร้อยแล้วจะต้องนำข้อมูลที่ได้จากการถอดรหัสไปดึงข้อมูลกลับเนื่องจากในการเข้ารหัสแบบคอนโวลูชัน (3,2,2) ที่ใช้เป็นการเข้ารหัสแบบไม่เป็นระบบ กล่าวคือคำรหัสที่ได้จะไม่ปรากฏข้อมูลในส่วนของข้อมูลออกมาจึงต้องนำไปทำการดึงข้อมูลกลับด้วยวงจรการกู้คืนข้อมุลกลับก่อนถึงจะได้ข้อมูลที่ถูกต้อง เมื่อทำการดึงข้อมูลกลับแล้วจะนำข้อมูลที่เป็นบิตอะเลย์ไปทำเป็นข้อมูลในรูปแบบของตัวอักษรขนาด 8 บิตเพื่อให้เป็นรูปแบบเดียวกันกับทางฝั่งส่งผ่านฟังก์ชัน bit2char ผลลัพธ์ที่ได้จะเป็นอะเลย์ข้อมูลแบบเดียวกันกับฝั่งส่งที่รับมาจากโปรแกรม Matlab ในขั้นตอนสุดท้ายจะทำการส่งข้อมูลไปยังโปรแกรม Matlab ด้วยคำสั่ง netconn_write โดยจะเขียนข้อมูลทั้งหมดเท่ากับทางฝั่งส่งที่ส่งมาในโปรแกรม Matlab ที่ฝั่งรับจะรับด้วยคำสั่ง fread(t) โดยพารามิเตอร์ในวงเล็บคือ t ที่เป็นอ๊อบเจกต์แบบ TCP/IP เมื่ออ่านข้อมูลที่คืนมาได้แล้วผลลัพธ์ที่ได้จะอยู่ในตัวแปลที่ชื่อ ans

ผลและวิจารณ์

ผล

1. ผลการสังเคราะห์วงจรสำหรับตัวเข้ารหัสและถอดรหัส

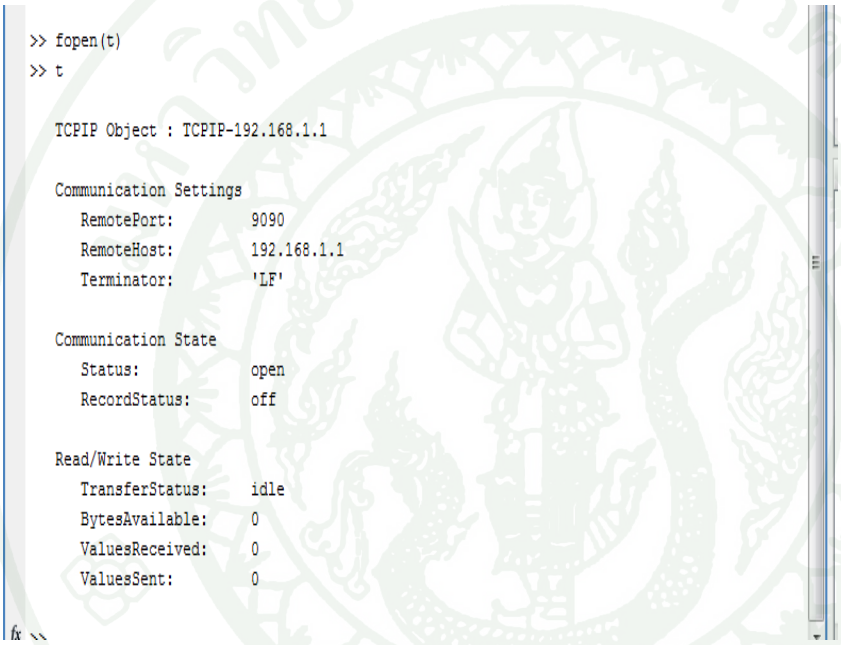
ผลจากการสังเคราะห์วงจรสำหรับตัวเข้ารหัสและถอดรหัสบน Nanoboard 3000 รุ่น XC3S1400AN-4FGG676C ประกอบไปด้วย 4-input LUTs-Logic 22,528 ตัว ใช้งานไป 6,509 ตัว คิดเป็น 28 % จากทั้งหมด I/0 pins 502 ตัวใช้งานไป 183 ตัวคิดเป็น 36 % ของทั้งหมด Slice Flip Flops จำนวน 22,528 ใช้งานไปทั้งหมด 3,198 ตัวคิดเป็น 14 % ของทั้งหมด การใช้งาน Slices ซึ่งเป็นบล็อกที่ประกอบไปด้วย LUTs และ Flipflop เพื่อใช้สำหรับสร้างเป็นลอจิกเกตรูปแบบต่าง ๆ ได้ใช้งานไปประมาณ 37 % และสำหรับการใช้งาน LUTs สำหรับเป็นเส้นทางในการเชื่อมต่อระหว่างลอจิกบล็อกใช้งานไป 344 ตัว เมื่อรวมกับ การใช้งานที่เป็น หน่วยความจำ 256 ตัวและที่เป็นลอจิกอีก 6253 ตัว จึงใช้งานรวมเป็น 6853 หรือประมาณ 30 % สำหรับจำนวน IOBs (Input-Output Blocks) ซึ่งเป็นส่วนที่ใช้เชื่อมต่อกับอินพุตเอท์พุตภายนอกใช้ไป 36% โดยแสดงผลการสังเคราะห์อยู่ในตารางที่ 4

ตารางที่ 6 ตารางการสังเคราะห์การใช้งานทรัพยากรบน FPGA

Logic Utilization	Used	Total	% usage
Number of slice Flip Flops	3,198	22,528	14%
Number of 4 input LUTs	6,509	22,528	28%
Logic distribution			
Number of occupied Slices:	4,194	11,264	37%
Total Number of 4 input LUTs	6,853	22,528	30%
Number used as logic	6,253		
Number used as a route-thru	344		
Number used for Dual Port RAMs	256		
Number of bonded IOBs	183	502	36%

2. ผลการทดสอบการรับ-ส่งข้อมูลกับบอร์ดทดลอง

การรับส่งข้อมูลด้วยโปรแกรม Matlab จะเริ่มจากการทำการขอทำการเชื่อมต่อด้วยคำสั่ง fopen ซึ่งหากสามารถทำการเชื่อมต่อได้แล้วจะแสดงภาพดังรูปที่ 25 ซึ่งแสดงว่าโปรแกรม Matlab ใช้การทำการเชื่อมต่อนี้จะทำการเชื่อมต่อกับ IP address 192.168.1.1 พอร์ต 9090 โดยที่ Status อยู่ในสถานะ Open ดังภาพที่ 25



ภาพที่ 25 แสดงการเชื่อมต่อระหว่าง อีเทอร์เน็ต และบอร์ดทดลอง

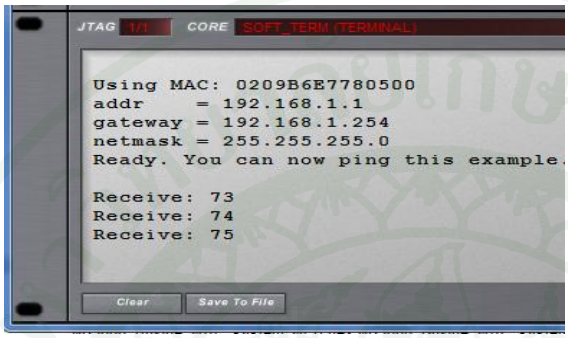
ส่วนบอร์ดทดลองจะใช้ IP address เป็น 192.168.1.1 Netmask เป็น 255.255.255.0 เพื่อทำการเชื่อมต่อกับ Matlab หลังจากทำการเชื่อมต่อกันได้แล้วจะทำการทดสอบโดยการส่งตัวเลย ผ่านคำสั่ง fwrite ไปยังบอร์ดทดลอง จากนั้นแสดงผลในส่วน Soft terminal ที่เป็นส่วนแสดงผลของบอร์ดทดลอง Nanoboard 300 ผ่านทางโปรแกรม Altium Designer

หลังจากรับค่าแล้วจะคืนค่าที่ได้รับกลับมายังโปรแกรม Matlab การอ่านค่าที่ได้รับมาจะใช้คำสั่ง fread เพื่ออ่านค่ามาเก็บไว้ใน Matlab ผลลัพธ์ทั้งหมดที่ได้สำหรับการทดสอบครั้งนี้แสดงดังภาพที่ 26 โดยจากภาพที่ 26 (ก) ทดลองลองการส่งค่าไปทั้งหมด 3 ค่า 73 74 75 จากโปรแกรม Matlab ไปยังบอร์ดทดลอง (ข) รับค่าภายในบอร์ดทดอลงและแสดงผล จากภาพ (ค) บอร์ดทดลองจะส่งค่ากลับไปยังโปรแกรม Matlab อ่านค่าที่ได้ซึ่งเป็นการอ่านค่าที่เก็บไว้ภายในบอร์ดทดลอง

(ก)

```
>> fwrite(t,73,'int16')
>> fwrite(t,74,'int16')
>> fwrite(t,75,'int16')
>>
```

(ข)



(ค)

```
>> fread(t)
Warning: The specif

ans =

     0
    73
     0
    74
     0
    75

>>
```

ภาพที่ 26 ก.) การส่งค่าข้อมูลไปยังบอร์ดทดลอง
ข.) การรับค่าข้อมูลภายในบอร์ดทดลอง
ค.) การอ่านค่ากลับจากโปรแกรม Matlab

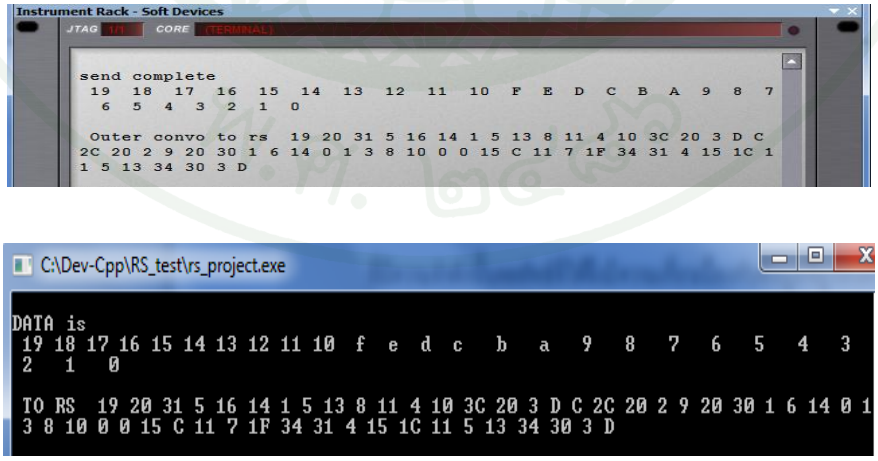
3. ผลการทดสอบความถูกต้องของการเข้ารหัส

ในการทดสอบการเข้ารหัสนั้นจะแบ่งออกเป็นการทดลอง 3 ส่วนด้วยกันกล่าวคือ การทดสอบการเข้ารหัสภายนอก การทดสอบการเข้ารหัสภายใน การทดสอบการเข้ารหัสแบบต่อร่วม ซึ่งการทดสอบจะทำการผ่านข้อมูลที่สร้างขึ้นเพื่อให้เห็นความเปลี่ยนแปลงที่เกิดขึ้นแล้วนำไปเทียบกับทฤษฎีที่เขียนขึ้นด้วยโปรแกรมภาษา C++

3.1 การทดสอบการเข้ารหัสภายนอก

เนื่องจากการเข้ารหัสภายนอกนั้นจะเข้ารหัสแบบคอนโวลูชัน (3,2,2) ซึ่งเป็นรหัสที่มีความจำดังนั้นผลลัพธ์ที่ได้จึงมีความเกี่ยวเนื่องกับข้อมูลที่ถูกส่งผ่านด้วยฟังก์ชันก่อนข้อมูลตัวปัจจุบันด้วยโดยในส่วนเริ่มต้นนั้นจะต้องทำการตั้งค่าหน่วยความจำให้เป็นบิต 0 ทั้งหมดก่อนจึงรับข้อมูลเข้ามายังฟังก์ชันการเข้ารหัส การทดลองจะทำการเข้ารหัสทั้งหมด 14 สัญลักษณ์ แบ่งเป็นส่วนของข้อมูล 5 สัญลักษณ์ต่อ 1 อินพุตและการเคลียร์หน่วยความจำอีก 2 สัญลักษณ์ต่อ 1 อินพุต ซึ่งจะได้ผลลัพธ์ออกมาทั้งหมด 21 สัญลักษณ์ โดยตัวอย่างผลที่ได้จากบอร์ดทดลองนั้นจะทำการเปรียบเทียบกับโปรแกรมที่เขียนขึ้นตามทฤษฎีโดยภาษา C++ ตัวอย่างผลลัพธ์จะแสดงในภาพที่ 27

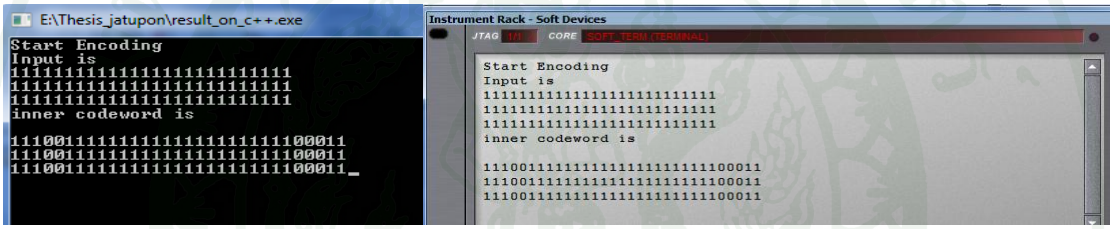
ซึ่งจากภาพที่ 27 จะเป็นการส่งข้อมูลไปเข้ารหัสภายนอกแบบ (3,2,2) คอนโวลูชัน โดย ข้อมูลแรกเป็น 19 18 17 16 15 14 13 12 11 10 F E D จำนวนทั้งหมด 13 ไบต์หรือ 104 บิต แสดงออกมาในรูปของฐาน 16 ซึ่งข้อมูลที่รับมาจะนำไปใช้เพียง 102 บิตโดยจะไม่นำข้อมูล 2 บิตสุดท้ายไปใช้งานและ ข้อมูลที่สองเป็น C B A 9 8 7 6 5 4 3 2 1 0 จำนวนทั้งหมด 13 ไบต์แสดงในรูปของฐาน 16 นำข้อมูลไปใช้ 102 บิตเช่นเดียวกัน ยกตัวอย่างเช่น 19 คือข้อมูล 00011001 และ 18 คือ 00011000 เป็นต้น เมื่อได้รับข้อมูลอย่างครบถ้วนแล้วจะนำข้อมูลไปเข้ารหัส โดยข้อมูลที่รับมาจะมีทั้งหมด 102 x 3 = 306 บิต แปลงเป็นข้อมูลจำนวน 51 สัญลักษณ์ แต่ละสัญลักษณ์มีขนาด 6 บิตต่อสัญลักษณ์เพื่อนำข้อมูลไปเข้ารหัสแบบรีดโซโลมอนต่อไป



ภาพที่ 27 ตัวอย่างผลจากการเข้ารหัสภายนอกแบบคอนโวลูชัน(3,2,2)บนบอร์ดทดลองและโปรแกรม C++

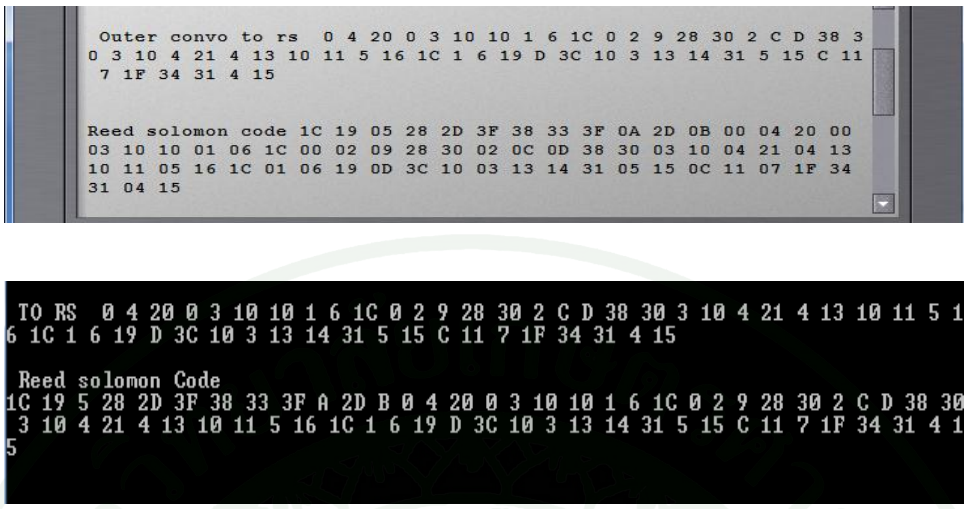
3.2 การทดสอบความถูกต้องของการเข้ารหัสภายใน

การเข้ารหัสภายในทำการเข้ารหัสแบบบีซีเฮส (31,26) และรหัสแบบรีดโซโลมอน (63,51) ซึ่งเป็นการเข้ารหัสที่ไม่มีความจำ ดังนั้นในการเข้ารหัสจำไม่จำเป็นที่จะต้องสร้างบล็อกข้อมูลเพื่อเก็บคำรหัสที่ได้ไว้ใช้ครั้งต่อไป โดยการทดสอบด้วยข้อมูลที่เป็น 1 ทั้งหมดจำนวน 26 บิต ไปเข้ารหัสแบบบีซีเฮช (31,26) ผลลัพธ์ที่ได้จะได้ทั้งหมด 31 บิต ตัวอย่างผลลัพธ์ที่เกิดขึ้นหลังจากการเข้ารหัสบนบอร์ดทดลองจะแสดงในภาพที่ 28 ซึ่งให้ผลลัพธ์อยู่ในรูปของเลขฐาน 2 และตัวอย่างการเข้ารหัสภายในบนโปรแกรม C++ จะแสดงในภาพที่ 29 จากการเปรียบเทียบผลลัพธ์ที่ได้แสดงให้เห็นว่า ผลลัพธ์มีความถูกต้อง



ภาพที่ 28 ตัวอย่างผลจากการเข้ารหัสภายในแบบบีซีเฮช(31,26)บนโปรแกรม c++ และบนบอร์ดทดลอง

สำหรับการทดสอบการเข้ารหัสรีดโซโลมอนแสดงในภาพที่ 30 โดยจะต้องทำการป้อนข้อมูลจำนวนไม่เกิน 6 บิตต่อ 1 สัญลักษณ์เนื่องจากรหัสรีดโซโลมอนอยู่บนพื้นฐาน GF(2^6) จากภาพที่ 31 ข้อมูลที่จะนำมาเข้ารหัสคือ 0 จนถึง 15 ซึ่งข้อมูลแสดงอยู่ในรูปฐาน 16 ยกตัวอย่างเช่น 0 = 000000 กรณี 4 = 001000 กรณี 20 = 100000 เป็นต้น จากนั้น ผลลัพธ์จากการเข้ารหัสข้อมูลทั้งหมด 51 สัญลักษณ์ จะได้ข้อมูลเป็น 63 สัญลักษณ์ที่แสดงในรูปฐาน 16 เช่นเดียวกัน ยกตัวอย่างเช่น ผลลัพธ์ 1C =011100 และ 19 = 011001 เป็นต้น โดยการเปรียบเทียบโปรแกรม C++ และบนบอร์ดทดลองให้ผลลัพธ์ที่เหมือนกันแสดงในภาพที่ 29



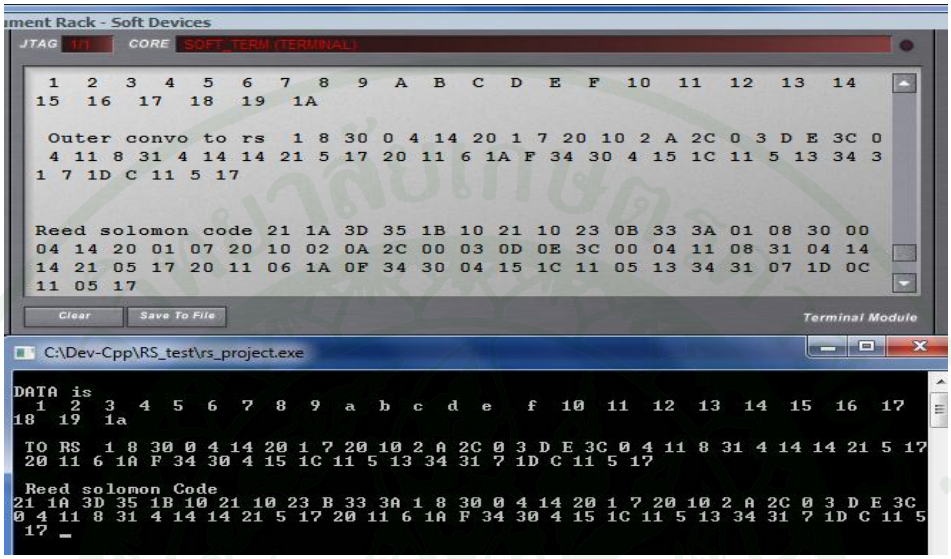
ภาพที่ 29 ตัวอย่างผลจากการเข้ารหัสภายในแบบรีดโซโลมอนบนบอร์ดทดลองและโปรแกรมภาษา C++

3.3 การทดสอบความถูกต้องของการเข้ารหัสแบบต่อร่วม

การทดสอบความถูกต้องของการเข้ารหัสแบบต่อร่วมจะใช้รหัสแบบรีดโซโลมอนเป็นรหัสภายในต่อร่วมกับรหัสคอนโวลูชันที่เป็นรหัสภายนอก เพื่อใช้สำหรับการทดสอบในช่องสัญญาณจริงต่อไป โดยวิธีการทดสอบจะส่งข้อมูลผ่านขนาด 102 บิตนำไปทำการเข้ารหัสภายในจำนวน 7 สัญลักษณ์ต่อ 1 อินพุตประกอบไปด้วย ข้อมูล 5 สัญลักษณ์และการเคลียร์หน่วยความจำ 2 สัญลักษณ์ รวมทั้งหมด 14 สัญลักษณ์ รวมบิตที่ทดสอบทั้งหมด 1428 บิต และผลลัพธ์ที่ได้มีขนาด 2646 บิต โดยแสดงตัวอย่างผลลัพธ์ที่ได้บนบอร์ดทดลอง ในภาพที่ 30 โดยการป้อนข้อมูลอินพุตขนาด 102 บิต 2 ค่าเข้าไปในรหัสแบบคอนโวลูชัน (3,2,2) และแปลงผลลัพธ์ที่ได้ให้อยู่ในรูปข้อมูล 6 บิตต่อ 1 สัญลักษณ์ จากนั้นป้อนข้อมูลเข้าไปในตัวเข้ารหัสภายในแบบรีดโซโลมอน (63,51) ผลลัพธ์ที่ได้ออกมาจะมีขนาด 63 สัญลักษณ์ สัญลักษณ์ละ 6 บิต

สำหรับตัวอย่างผลลัพธ์ที่ได้บนโปรแกรม C++ ในภาพที่ 30 ข้อมูลที่รับมาคือ 1 จนถึง 1A จำนวนทั้งหมด 26 ไบต์ โดยแบ่งข้อมูลชุดแรกเป็น 1 2 3 4 5 6 7 8 9 A B C D และข้อมูลชุดที่สองเป็น E F 10 11 12 13 14 15 16 17 18 19 1A โดยแต่ละชุดข้อมูลอินพุตจะทำการเก็บข้อมูล 102 บิตจาก 104 บิตต่อสัญลักษณ์ เพื่อนำไปเข้ารหัสคอนโวลูชัน (3,2,2) หลังจากเข้ารหัสภายนอกแล้วจะทำการแปลงข้อมูลเป็นเลขฐาน 6 จำนวน 51 สัญลักษณ์สัญลักษณ์ละ 6 บิต จากภาพที่ 32 คือ 1

จนถึง 17 และนำข้อมูลไปเข้ารหัสภายในแบบรีดโซโลมอน (63,51) ผลลัพธ์ที่ได้คือ 21 จนถึง 17 ทั้งหมด 63 สัญลักษณ์

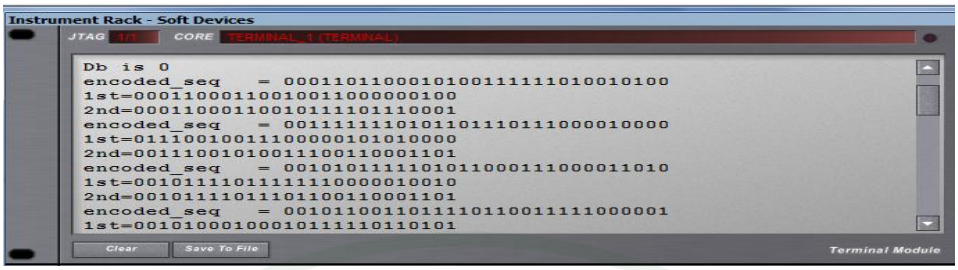


ภาพที่ 30 ตัวอย่างผลจากการเข้ารหัสแบบต่อร่วมบนโปรแกรมภาษา C++ และบอร์ดทดลอง FPGA

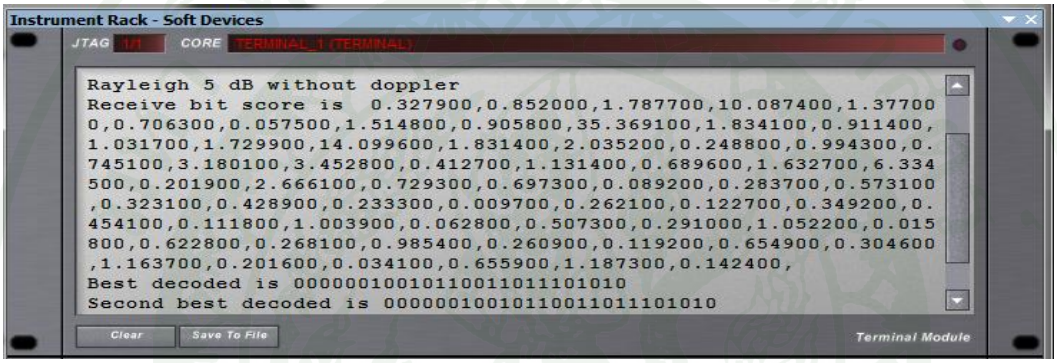
4. ผลการทดสอบความถูกต้องและประสิทธิภาพของการถอดรหัส

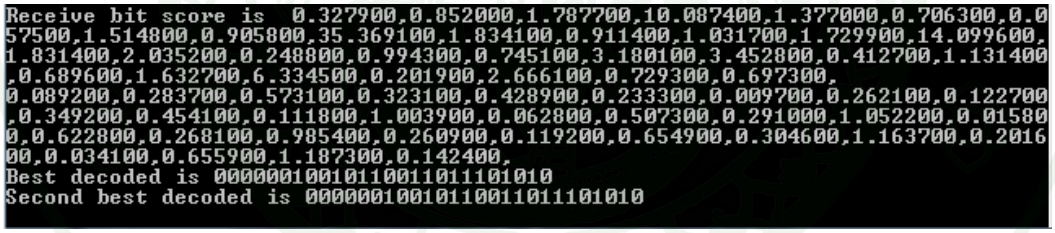
4.1 การทดสอบความถูกต้องและประสิทธิภาพของการถอดรหัสภายใน

การถอดรหัสภายในจะใช้วิธีการของลิฟออฟทูวิเทอร์บี (List-of-2 Viterbi Algorithm) ซึ่งจะเก็บค่าข้อมูลที่ถอดรหัสไว้ทั้ง 2 ตัวเลือก การทดสอบจะทำทั้งการตัดสินใจแบบฮาร์ด (Hard decision) และการตัดสินใจแบบซอฟท์ (Soft decision) โดยในภาพที่ 31 จะแสดงตัวอย่างผลการตัดสินใจแบบฮาร์ด (Hard Decision) ในบอร์ดทดลอง และในภาพที่ 32 จะแสดงตัวอย่างของผลการตัดสินใจแบบซอฟท์ (Soft Decision) ในบอร์ดทดลองและบนโปรแกรม C++ สำหรับการถอดรหัสแบบซอฟท์วิเทอร์บีจะทำการรับค่าที่เป็นบิตสกอร์เข้ามาเพื่อคำนวณผลลัพธ์ที่ได้จากภาพที่ 32 จะแสดงออกมาในรูปของเลขฐาน 2 โดยมีค่าทางเลือกแรกและค่าทางเลือกที่สองแสดงในรูปที่ 32 เมื่อเปรียบเทียบผลลัพธ์พบว่าได้ค่าทั้ง 2 ทางเลือกเท่ากัน



ภาพที่ 31 แสดงตัวอย่างผลการถอดรหัสแบบ Hard Viterbi ในบอร์ดทดลอง



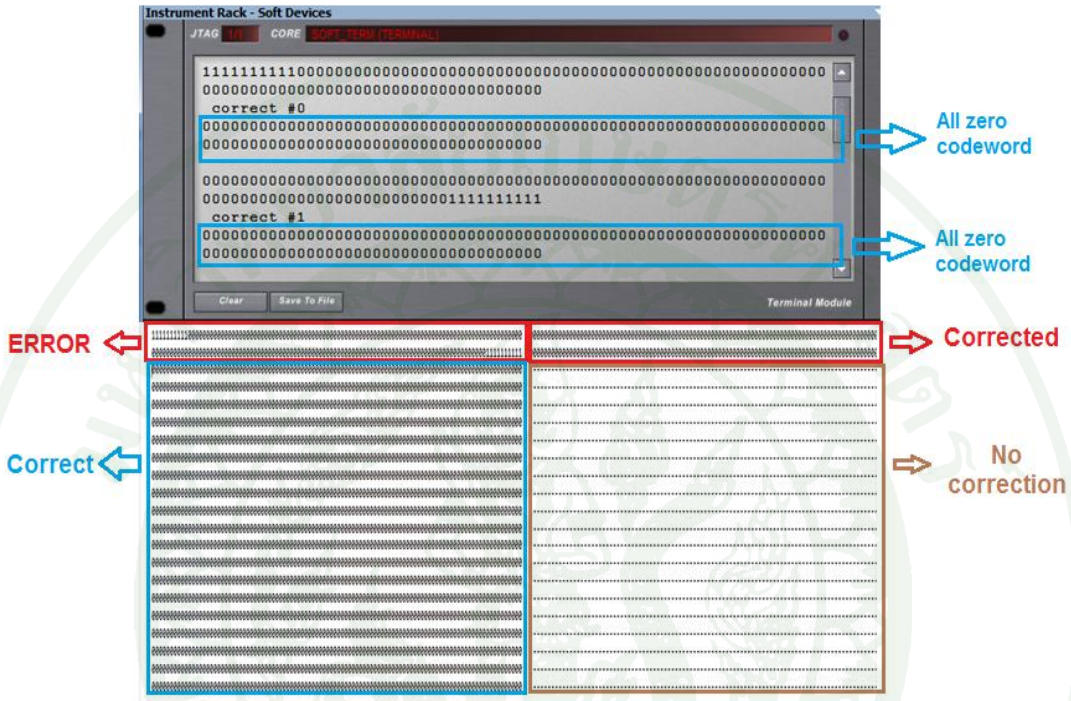


ภาพที่ 32 แสดงตัวอย่างของผลการถอดรหัสแบบ Soft Viterbi ในบอร์ดทดลอและบนโปรแกรม C++ (Jatupon and *et.al.*, 2012)

4.2 การทดสอบความถูกต้องและประสิทธิภาพของการถอดรหัสภายนอก

รหัสภายนอกนั้นจะใช้การถอดรหัสแบบเวกเตอร์ซิมโบล (Vector Symbol Decoding) โดยขนาดสัญลักษณ์ที่ใช้ทดสอบนั้นมีขนาดเท่ากับ 102 บิต ต่อสัญลักษณ์ ซึ่งการทดสอบจะทำการสร้างข้อมูลที่มีค่าผิดพลาดเกิดขึ้น แล้วนำมาเป็นข้อมูลอินพุตให้กับบอร์ดทดลอง หลังจากทำการถอดรหัสแบบเวกเตอร์ซิมโบล แล้วจึงนำผลลัพธ์มาเปรียบเทียบกันดังภาพที่ 33 โดยการทดสอบจะเป็นการสร้างคำรหัสที่เป็น 0 ทั้งหมดป้อนเข้าไปความผิดพลาดที่เกิดขึ้นในสัญลักษณ์แรกสำหรับ 10 บิตแรกและสัญลักษณ์ที่ 2 สำหรับ 10 บิตสุดท้าย บอร์ดทดลองสามารถแก้ไขความผิดพลาดให้มี

ความถูกต้องได้จากภาพที่ 33 โดยส่วนที่เป็น 0 หมด คือค่าที่เป็นผลลัพธ์ออกมาจากการแก้ไขความถูกต้องบนบอร์ดทดลอง

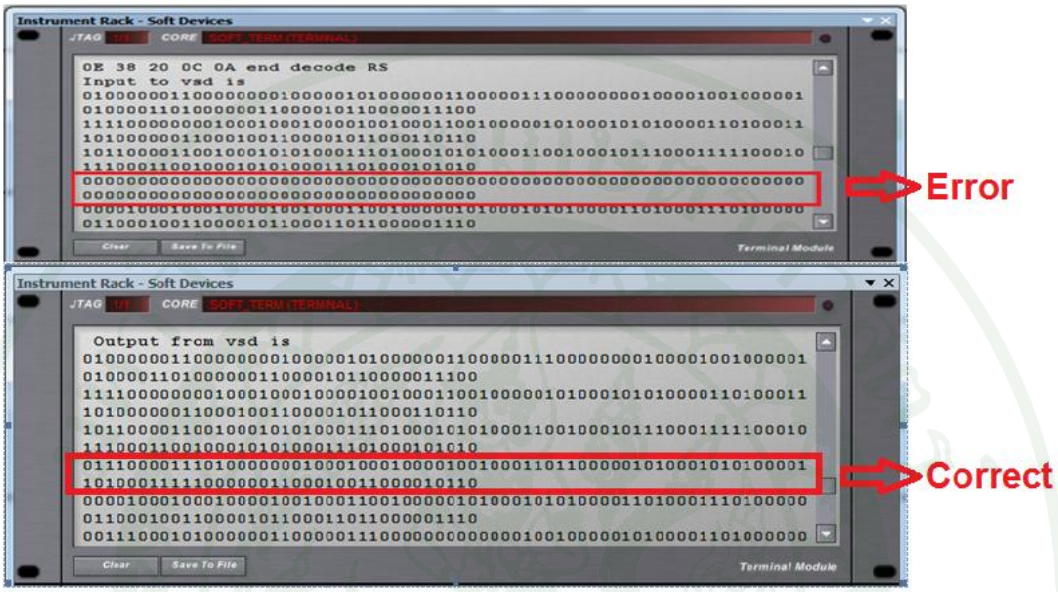


ภาพที่ 33 ผลลัพธ์ที่ได้จากการถอดรหัสแบบ VSD ในบอร์ดทดลองและบนโปรแกรม C++

จากภาพที่ 33 จะพบว่าในสัญลักษณ์ที่ 1 และสัญลักษณ์ที่ 2 มีค่าความผิดพลาดเกิดขึ้นส่วนในสัญลักษณ์ที่เหลือมีความถูกต้องอยู่แล้ว ซึ่งในโปรแกรมภาษา C++ จะทำการแก้ไขความผิดให้มีความถูกต้องตามภาพที่ 34 ใน 2 สัญลักษณ์แรกทำการแก้ไขความผิดพลาดและสัญลักษณ์ที่เหลือจะไม่ทำการแก้ไข โดยเทียบได้จากข้อมูลในหน้าต่างของบอร์ดทดลองที่ข้อมูลในสัญลักษณ์แรกมีความผิดพลาดจึงทำการแก้ไขให้ถูกต้อง ถัดมาสัญลักษณ์ที่สองมีความผิดพลาดจึงทำการแก้ไขให้มีความถูกต้องเช่นกัน

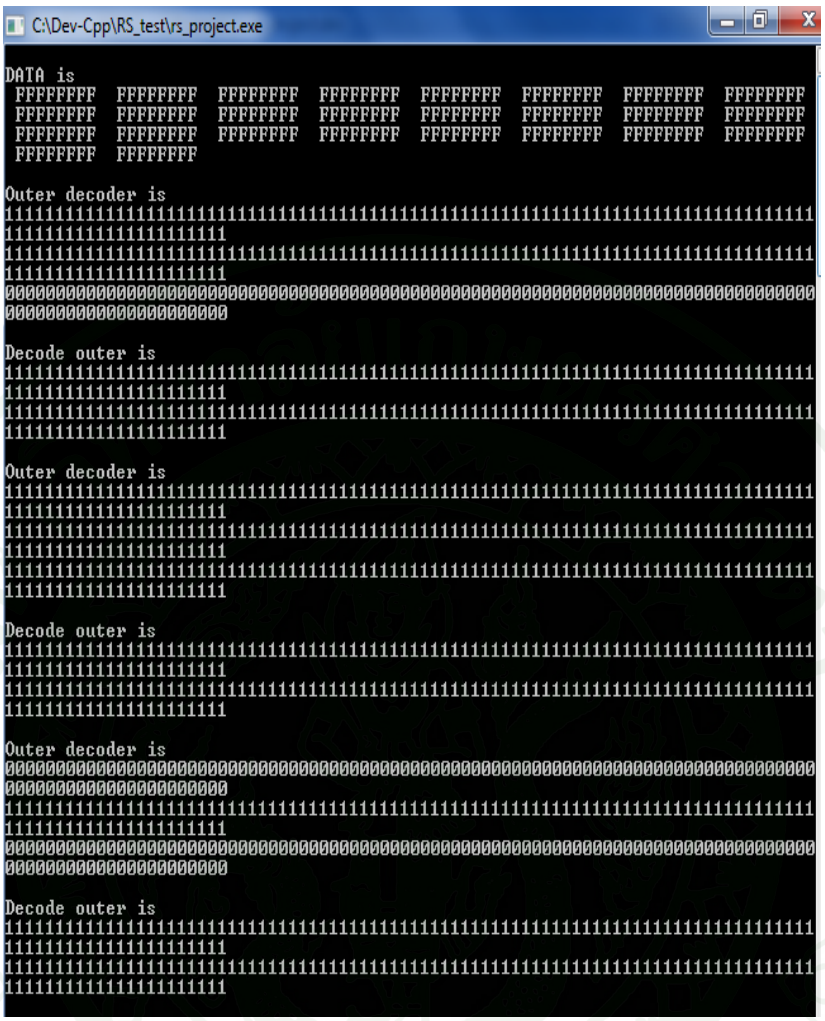
สำหรับกรณีที่ส่งข้อมูลแบบเข้ารหัสภายนอกผ่านไปยังการทำการถอดรหัสแบบเวกเตอร์ซิมโบลและทำการสร้างความผิดพลาดในสัญลักษณ์ที่ 4 ในกรณีนี้ผลลัพธ์ที่ได้สามารถแก้ไขความผิดพลาดได้เช่นกันโดยผลลัพธ์แสดงในภาพที่ 34 โดยรูปด้านบนเป็นรูปที่แสดงผลลัพธ์ที่ผิดพลาดและรูปด้านล่างเป็นการแก้ไขความผิดพลาดของคำรหัสให้มีความถูกต้องโดยในกรอบด้านบนใน

ภาพที่ 34 ภาพบนจะเป็นข้อมูลที่มีความผิดพลาดและกรอบด้านล่างภาพที่ 34 จะเป็นข้อมูลที่ถูกแก้ไขแล้ว



ภาพที่ 34 ภาพการแก้ไขความถูกต้องของเวกเตอร์ซิมโบลดีโคดดิง

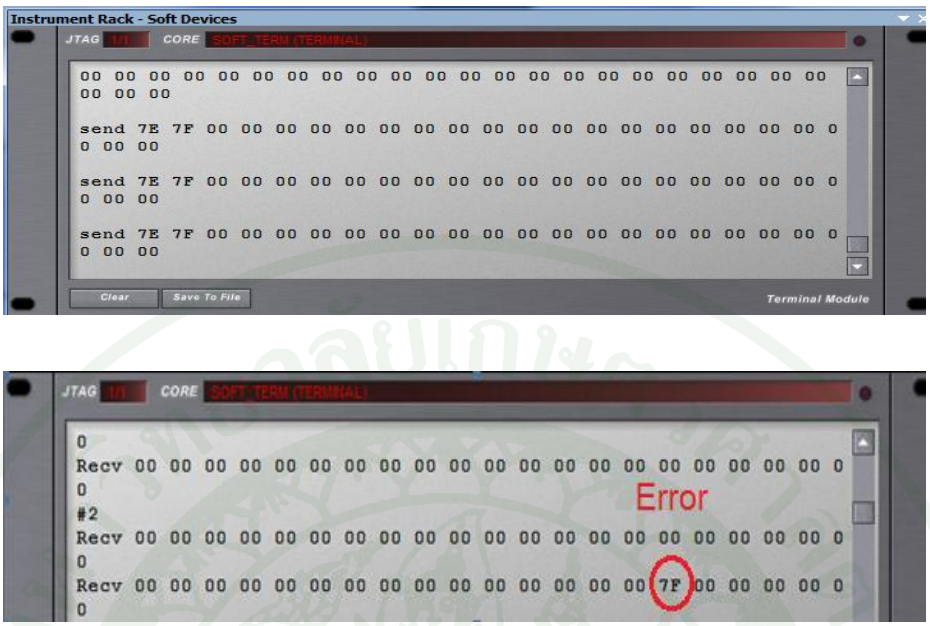
หลังจากทดสอบการถอดรหัสภายนอกเพื่อทำการตรวจสอบผลความถูกต้อง ขั้นตอนต่อไปจะทำการทดสอบการดึงข้อมูลกลับต้นฉบับ สำหรับการทดสอบเบื้องต้นจะนำคำรหัสที่ทำการเข้ารหัสแบบคอนโวลูชัน (3,2,2) มาทำการดึงข้อมูลกลับซึ่งจากภาพที่ 35 มาจากการป้อนข้อมูลที่เป็น 1 ทั้งหมดเข้าไปทำการเข้ารหัสแบบคอนโวลูชัน (3,2,2) และนำคำรหัสที่ได้ผ่านเข้าไปทดสอบกับการทำงานบนโปรแกรม C++ ผลลัพธ์ที่ได้จากบนโปรแกรม C++ ให้ผลลัพธ์ที่เหมือนกับการทดสอบการทำงานบนบอร์ดทดลอง โดยสามารถดึงข้อมูลกลับจากรหัสภายนอกได้อย่างถูกต้อง



ภาพที่ 35 ตัวอย่างผลลัพธ์ที่ได้จากการดึงข้อมูลกลับจากโปรแกรม C++

5. ผลการทดสอบการรับ-ส่งสัญญาณแบบไร้สาย

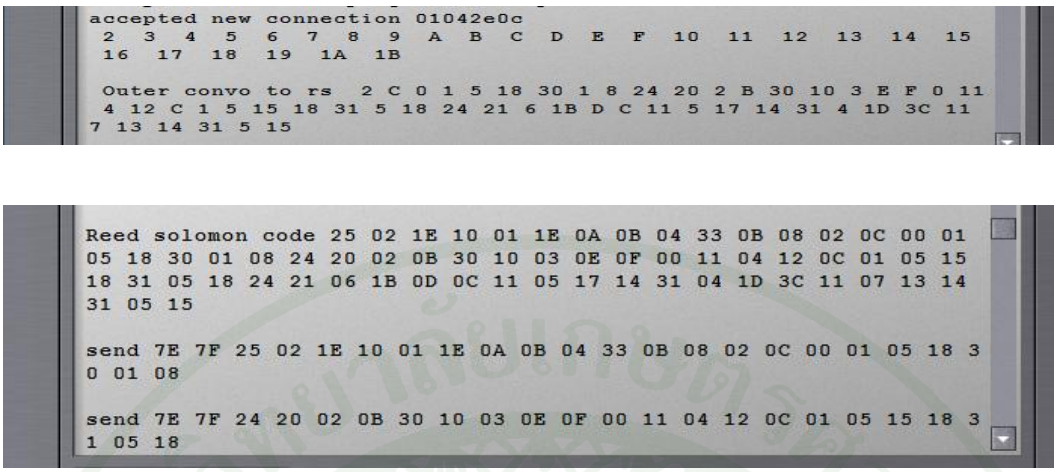
สำหรับการทดสอบการรับส่งสัญญาณไร้สายจะทำการส่งข้อมูลจากอุปกรณ์ส่งสัญญาณไปยังอุปกรณ์ฝั่งรับสัญญาณ จากนั้นทำการดูข้อมูล การส่งสัญญาณจากฝั่งส่งจะทำการส่งข้อมูลที่เป็น 0 หมดเพื่อทดสอบว่ามีความผิดพลาดเกิดขึ้นในฝั่งรับหรือไม่ซึ่งในการส่งข้อมูลที่เป็น 0 ทั้งหมดทำให้ง่ายต่อการตรวจสอบดังภาพที่ 36 สำหรับผลลัพธ์ที่ได้จากการส่งสัญญาณผิดพลาดแสดงในภาพที่ 36 ภาพล่างโดยส่งข้อความที่เป็น0ทั้งหมดแต่ได้รับข้อความที่เป็น 7F เข้ามา



ภาพที่ 36 ภาพแสดงการรับส่งข้อมูลแบบไร้สายที่มีความผิดพลาดเกิดขึ้น

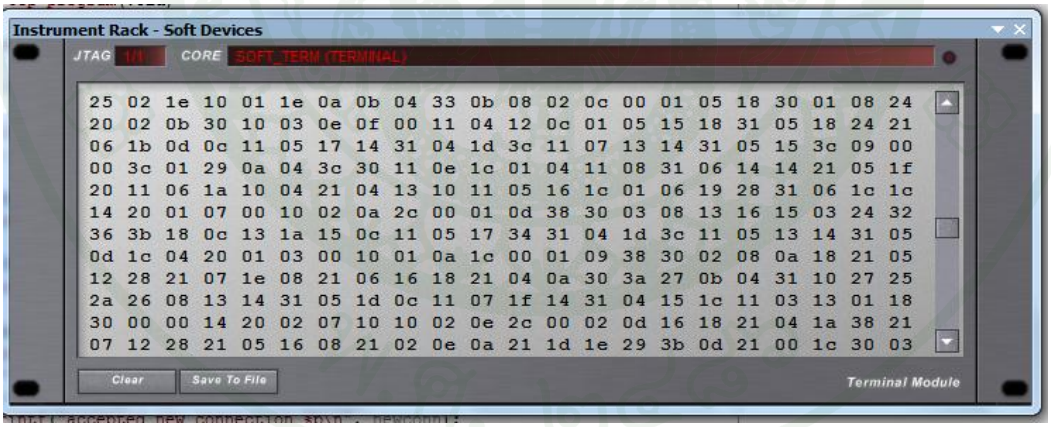
6. ผลการทดสอบระบบ

ผลการทดสอบระบบการเข้ารหัสแบบต่อร่วมเบื้องต้นจะเป็นการทดสอบระหว่างบอร์ดทดลองและบอร์ดทดลองโดยผ่านอุปกรณ์ต่อรับสั่งสัญญาณแบบไร้สายเพื่อทดสอบประสิทธิภาพของรหัส โดยเริ่มจากการส่งชุดข้อมูลจากฝั่งส่งดังภาพที่ 37 พบว่า สามารถรับข้อมูลจากพอร์ตอีเทอร์เน็ตและแสดงในรูปของเลขฐาน 16 และสามารถเข้ารหัสแบบต่อร่วมเพื่อส่งผ่านพอร์ต RS-232 ไปยังฝั่งรับในภาพที่ 38 สามารถที่จะรับข้อมูลอย่างถูกต้องเช่นกัน จากภาพที่ 37 ส่วนแรกจะรับข้อมูล 2 3 4 5 6 7 8 9 จนถึง 1B ส่วนลำดับถัดมาจะเป็นการเข้ารหัสแบบคอนโวลูชัน (3,2,2) จะให้ข้อมูล เป็น 2 C 0 1 5 18 จนถึง 15 และนำข้อมูลที่ได้ไปเข้ารหัสแบบรีดโซโลมอน (61,53) ก่อนที่จะส่งข้อมูลทั้งหมดที่เป็นคำรหัสแบบต่อร่วมไปยังพอร์ต RS-232 เพื่อให้ส่งผ่านอุปกรณ์ส่งสัญญาณต่อไป ข้อมูลที่ส่งมาจะประกอบไปด้วย Header 7E 7F ด้วยซึ่งในส่วนนี้เป็นการกำหนดให้ฝั่งรับอ่านค่าจาก Header ก่อนเพื่อรับข้อมูลอีก 21 ไบต์ที่ตามมา โดยข้อมูลที่ส่งไปชุดแรกคือ 25 จนถึง 08 ชุดที่สองคือ 24 จนถึง 18 และชุดที่ 3 คือ 24 จนถึง 15



ภาพที่ 37 ตัวอย่างการทดลองการเข้ารหัสช่องสัญญาณแบบต่อร่วมและส่งข้อมูลจากฝั่งส่ง

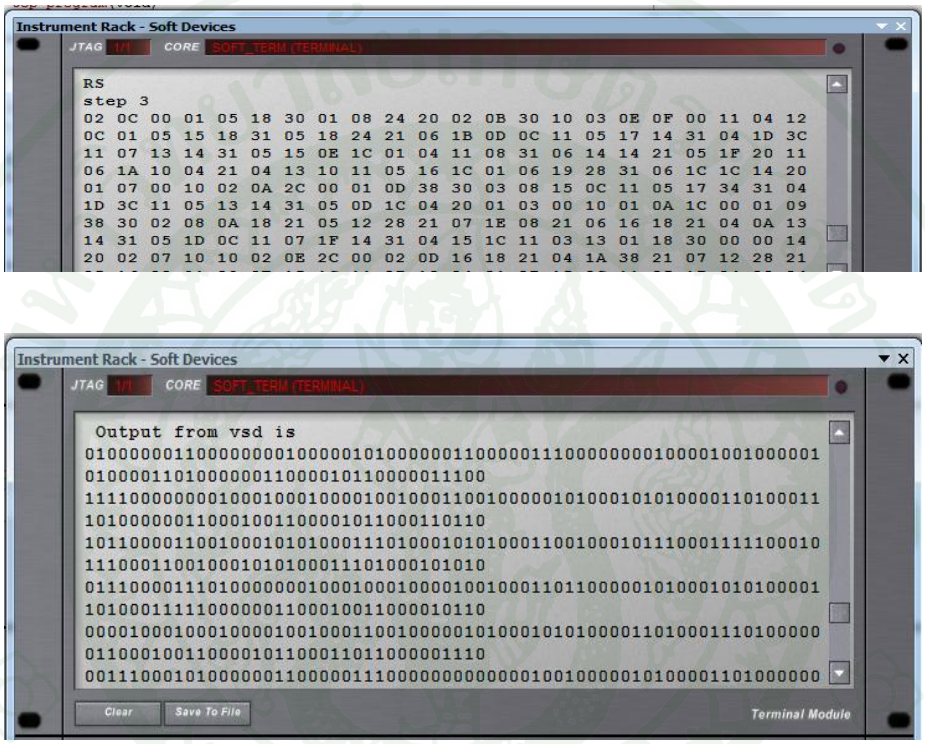
สำหรับภาพที่ 38 จะเป็นข้อมูลทั้งหมดที่บอร์ดทดลองรับค่ามาเพื่อนำข้อมูลทั้งหมดไปถอดรหัสแบบต่อร่วมโดยรับข้อมูลชุดแรกมาคือ 25 จนถึง 08 และทำการรับค่าทั้งหมดที่เป็นข้อมูลเข้ามาจำนวน 63 x 7 = 441 ค่า โดยจากภาพที่ 38 แสดงบางส่วนของการรับข้อมูลเท่านั้น



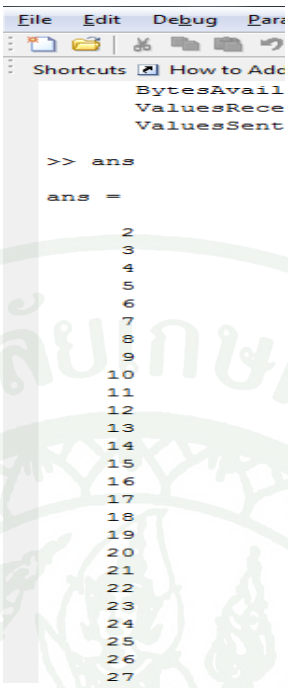
ภาพที่ 38 ตัวอย่างการรับข้อมูลทางฝั่งรับเพื่อถอดรหัสรีดโซโลมอน (63,51)

สำหรับภาพที่ 39 จะเป็นการถอดรหัสข้อมูลของรีดโซโลมอนซึ่งเมื่อถอดรหัสข้อมูลที่รับมาทั้งหมด 441 สัญลักษณ์จะทำให้ได้ข้อมูลเพื่อนำไปถอดรหัสแบบเวกเตอร์ซิมโบล โดยเปรียบเทียบข้อมูลที่ส่งมาก่อนจะเข้ารหัสแบบรีดโซโลมอนจากภาพที่ 37 คือ 2 ไปจนถึง 15 พบว่าข้อมูลที่ได้มีค่าที่ตรงกับภาพที่ 39 หลังจากนั้นข้อมูลที่ได้ทำการแปลงเป็นข้อมูล 102 บิตต่อสัญลักษณ์จำนวน 21 สัญลักษณ์เพื่อนำไปถอดรหัสด้วยการถอดรหัสแบบเวกเตอร์ซิมโบลผลลัพธ์ที่

ได้จากการถอดรหัสจากเวกเตอร์ซิมโบลแสดงในภาพที่ 39 ภาพล่างโดยแสดงบางส่วนเท่านั้น เมื่อได้ข้อมูลที่ทำการแก้ไขความผิดพลาดจากเวกเตอร์ซิมโบลแล้วจะทำการดึงข้อมูลกลับและส่งไปยังโปรแกรม Matlab โดยข้อมูลจากภาพที่ 40 เปรียบเทียบกับภาพที่ 37 ข้อมูลกับฝั่งส่งคือ 2 ถึง 1B ทางฝั่งรับรับได้คือ 2 ถึง 27 โดย 1B มีค่า 27 ในรูปแบบของเลขฐาน 10



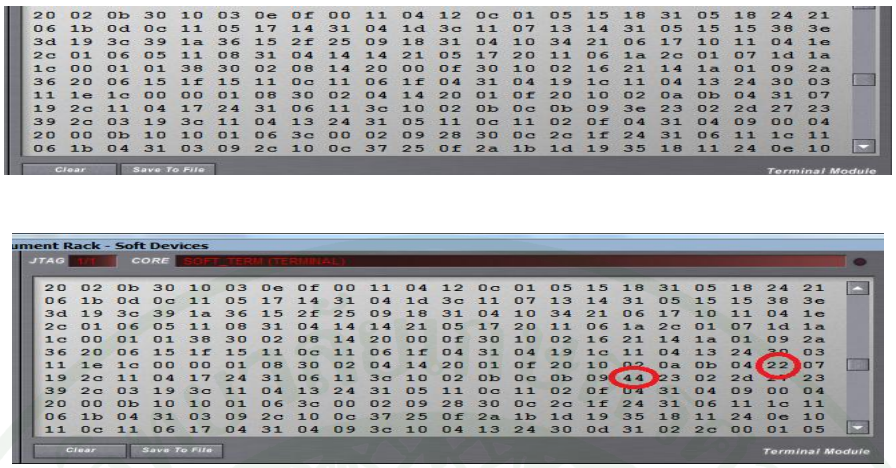
ภาพที่ 39 ตัวอย่างการถอดรหัสแบบต่อร่วมแบบรีดโซโลมอน (63,51) และการถอดรหัสแบบเวกเตอร์ซิมโบลดีโคดดิง



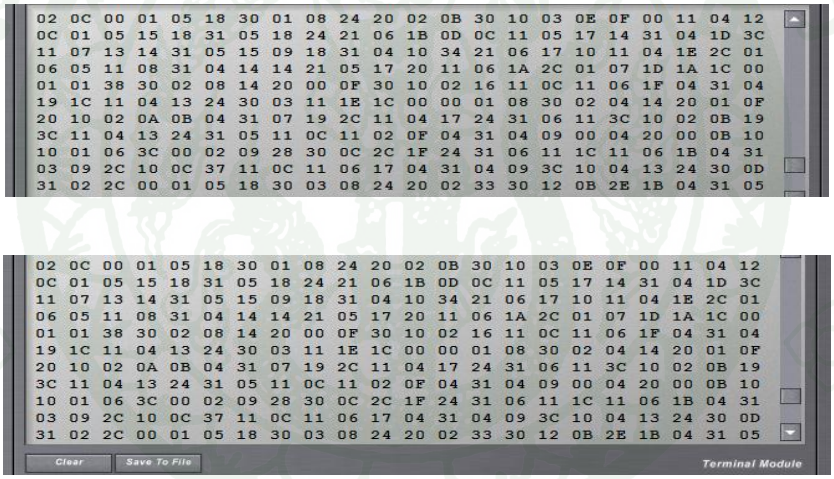
ภาพที่ 40 ตัวอย่างการรับข้อมูลของโปรแกรม Matlab จากบอร์ดทดลอง

ขั้นตอนต่อไปเป็นการทดสอบการถอดรหัสข้อมูลที่มีความผิดพลาดเกิดขึ้น โดยส่งข้อมูลที่เป็นชุดเรียงกันจากการทดสอบพบว่าสามารถกวนข้อมูลได้บางส่วนจนเกิดเป็นความผิดพลาดออกมาสำหรับความผิดพลาดแสดงในภาพที่ 41 โดยข้อมูลที่ผิดพลาดเกิดขึ้นแสดงเป็นเลขฐาน 16 คือ 44 และ 22 ซึ่งข้อมูลที่ถูกต้องควรเป็น 3E และ 31 ตามลำดับหลังจากฝั่งรับทำการรับข้อมูลแล้วจะส่งเข้าไปทำการแก้ไขความผิดพลาดต่อไป

จากภาพที่ 42 จะเป็นการเปรียบเทียบผลลัพธ์จากการแก้ไขความผิดพลาดที่เกิดขึ้นจากช่องสัญญาณไร้สายจริงโดยการรบกวน ผลลัพธ์ที่ได้พบว่าข้อมูลที่แก้ไขความผิดพลาดที่เกิดขึ้นให้เป็นข้อมูลที่ถูกต้องได้และส่งข้อมูลกลับไปยังโปรแกรม Matlab และเปรียบเทียบผลได้อย่างถูกต้อง



ภาพที่ 41 เปรียบเทียบความผิดพลาดที่เกิดขึ้นกับข้อมูลที่ไม่มีความผิดพลาด



ภาพที่ 42 เปรียบเทียบผลลัพธ์การแก้ไขคำรหัสระหว่างข้อมูลที่ไม่มีความผิดพลาดและมีความผิดพลาดเกิดขึ้น

วิจารณ์

1. การออกแบบระบบภายในบอร์ดทดลอง

จากการออกแบบระบบโดยใช้ซอฟท์คอร์โปรเซสเซอร์ TSK3000a ทำให้ระบบสามารถทำงานได้ตามต้องการโดยมีการทำงานร่วมกับพอร์ตอีเทอร์เน็ตสำหรับการรับส่งข้อมูลระหว่างบอร์ดทดลอง FPGA และคอมพิวเตอร์ซึ่งเลือกใช้เพื่อรองรับการใช้งานด้านเน็ตเวิร์คในอนาคต และมีการทำงานร่วมกับพอร์ต RS-232 เพื่อใช้ส่งข้อมูลให้กับส่วนรับ-ส่งสัญญาณไร้สาย สำหรับการสังเคราะห์วงจรของการออกแบบระบบภายนนบอร์ดทดลองใช้ทรัพยากรที่อยู่ใน FPGA ประมาณ 40% โดยระบบสามารถทำงานได้ตามทฤษฎีและการออกแบบระบบเพิ่มเติมในทรัพยากรที่เหลือสามารถทำได้เพื่อเพิ่มประสิทธิภาพให้กับการเข้ารหัสและถอดรหัส โดยระบบทั้งการเข้ารหัสและถอดรหัสใช้การออกแบบเดียวกันซึ่งทรัพยากรที่ใช้ไม่มีความเปลี่ยงแปลง ถึงแม้การถอดรหัสจะมีความซับซ้อนมากกว่าการเข้ารหัส

2. การติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ส่วนบุคคลและบอร์ดทดลง FPGA

ในการรับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ส่วบุคคลและบอร์ดทดลอง FPGA โดยการใช้งานอีเทอร์เน็ต สามารถทำได้อย่างมีประสิทธิภาพเนื่องจากการรับส่งข้อมูลระหว่างอุปกรณ์ทั้ง 2 มีการใช้งาน TCP/IP โปรโตคอลซึ่งอยู่ในลำดับขั้นของแอพพลิเคชันเลเยอร์ (Application Layer) ทำให้สะดวกต่อการทดลอง นอกจากนั้นโปรแกรม Matlab ยังสามารถถควบคุมการทำงานโดยคำสั่งด้วยวิธีการที่หลากหลายเพื่อตอบสนองความต้องการของผู้ใช้งานได้อย่างแม่นยำ สำหรับการทำงานของส่วนติดต่อสื่อสารของบอร์ดทดลอง FPGA นั้นก็จะติดต่อสื่อสารได้ในระดับเดียวกันกับโปรแกรม Matlab จึงทำให้การติดต่อสื่อสารเป็นไปได้อย่างดี แต่ว่าในการรับส่งข้อมูลจากโปรแกรม Matlab และบอร์ดทดลองยังต้องมีการเพิ่มการหน่วงเวลาเข้าไปเนื่องจากบอร์ดทดลองต้องอ่านข้อมูลและแสดงผลจึงทำให้ในกรณีที่ส่งข้อมูลมาเมื่อไม่มีการหน่วงเวลาจะทำให้ข้อมูลที่รับมาบางส่วนไม่สามารถเก็บไว้ในบัพเฟอร์ได้

3. การเข้ารหัสแบบต่อร่วมบนบอร์ดทดลอง

ในการเข้ารหัสแบบต่อร่วมบนบอร์ทดลองได้เข้ารหัสแบบคอนโวลูชัน (3,2,2) ต่อร่วมกับรหัสแบบรีดโซโลมอน (63,51) และการทดสอบการเข้ารหัสภายในแบบบีซีเฮช (31,26) ใช้โปรแกรมภาษา C เขียนลงในบอร์ดทดลอง FPGA และรับค่าอินพุตเข้ามาทดสอบการเข้ารหัสและทำการทดสอบความถูกต้องเทียบกับโปรแกรม C++ ที่เขียนขึ้นตามทฤษฎี ผลการทำงานในแต่ละส่วนสามารถที่จะทำงานได้ตรงตามทฤษฎีและจากการทดสอบระบบการเข้ารหัสพบว่า ผลลัพธ์ที่ได้มีความถูกต้องซึ่งได้มีการแสดงผลลัพธ์แล้วในส่วนของผลการทดลอง

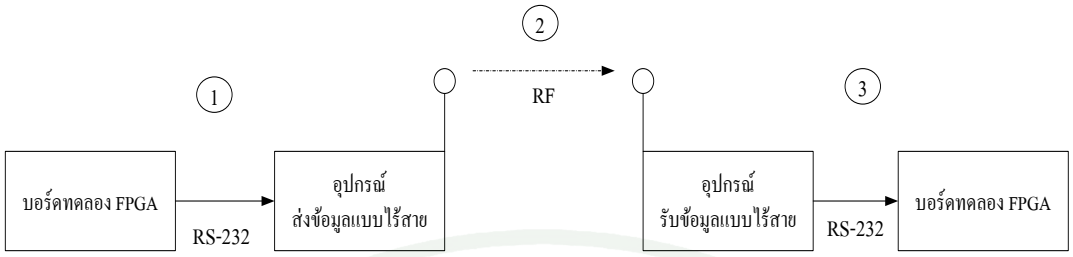
4. การถอดรหัสแบบต่อร่วมบนบอร์ดทดลอง

ในการทดลองการถอดรหัสแบบต่อร่วมระหว่างการถอดรหัสด้วยวิธีการของ Berlekhamp-Massey กับการถอดรหัสแบบเวกเตอร์ซิมโบล โดยการใช้การเขียนโปรแกรมภาษา C ลงบนบอร์ดทดลอง FPGA ในการถอดรหัสรีดโซโลมอนที่ต่อร่วมกับการถอดรหัสแบบเวกเตอร์ซิมโบล สามารถให้ผลลัพธ์ที่ถูกต้องตามทฤษฎี และได้ทดสอบเพิ่มเติมสำหรับการถอดรหัสแบบลิสออฟทูวิเทอร์บี ซึ่งนำเอาผลลัพธ์มาเปรียบเทียบกับโปรแกรม C++ ที่เขียนขึ้นตามทฤษฎีพบว่า ให้ผลลัพธ์ถูกต้อง และได้มีการแสดงผลลัพธ์ไปในส่วนของผลการทดลองแล้ว

5. การรับ-ส่งสัญญาณแบบไร้สาย

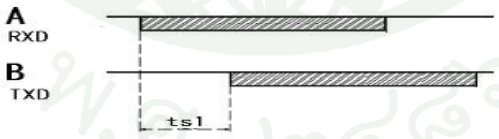
สำหรับการรับส่งสัญญาณสามารถรับส่งข้อมูลได้อย่างครบถ้วนด้วยการใช้เสาสัญญาณถึงแม้ในกรณีที่มีสิ่งกีดขวางบังอยู่ก็ตาม สำหรับในกรณีที่ทำการถอดเสาสัญญาณและปรับทิศทางให้อุปกรณ์ฝั่งส่งและรับไปอยู่ในทิศทางที่ไม่อยู่ในแนวเดียวกันจะทำให้เกิดความผิดพลาดเกิดขึ้น ซึ่งการทดสอบทำในความเร็วที่อัตราบอร์ดที่ 9600 บอร์ดต่อวินาทีแต่ต้องมีการทำการหน่วงเวลาซึ่งการหน่วงเวลาเกิดจากปัจจัยต่าง ๆ ดังนี้

ในขั้นตอนการส่งข้อมูลจากบอร์ดทดลองไปยังอุปกรณ์รับ-ส่งข้อมูลแบบไร้สายและรับข้อมูลจากอุปกรณ์รับ-ส่งข้อมูลแบบไร้สายส่งข้อมูลไปยังบอร์ดทดลองผ่านพอร์ตอนุกรม RS-232 สามารถทำได้ตามแผนภาพดังภาพที่ 43



ภาพที่ 43 ตัวอย่างขั้นตอนการหน่วงเวลาในการรับส่งข้อมูลของอุปกรณ์

สำหรับในขั้นตอนที่ 1 ในการส่งข้อมูลจากบอร์ดทดลองผ่านพอร์ตอนุกรม RS-232 แต่ละครั้งจะต้องส่งข้อมูลที่เป็นบิตเริ่ม จากนั้นจึงทำการส่งบิตข้อมูลที่เป็นออกไปซึ่งในแต่ละบิตที่ทำการส่งออกไปจะต้องใช้เวลา (1 / อัตราบอร์ด) สำหรับในการทดสอบที่ใช้ความเร็วที่ 9600 baud/sec ทำให้ในแต่ละบิตที่ส่งจะใช้เวลาในการส่ง 104 us หลังจากที่ส่งบิตข้อมูลแล้วจะต้องส่งบิตหยุดอีก 1 บิตเพื่อเป็นการบอกว่าสิ้นสุดบิตข้อมูลแล้ว ดังนั้นสำหรับในการส่งข้อมูล 1 ไบต์จะต้องใช้เวลาในการส่งประมาณ 1 ms เพื่อให้ส่งข้อมูลจากฝั่งส่งหรือบอร์ดทดลองไปยังฝั่งรับหรือบอร์ดทดลองไร้สาย สำหรับในการเริ่มส่งข้อมูลชุดถัดไปจำเป็นต้องมีการหน่วงเวลาเพื่อให้บิตเริ่มจากข้อมูลชุดถัดไปอยู่ในช่วงเวลาที่ใกล้กันจนเกินไปจนเกิดการตัดสินใจที่ผิดพลาด เนื่องจากการสื่อสารข้อมูลผ่าน RS-232 เป็นการสื่อสารข้อมูลแบบไม่ประสานเวลา (Asynchronous) จะต้องใช้บิตในการบอกจุดเริ่มต้นขอข้อมูลหากไม่สามารถแยกได้ว่าบิตเริ่มต้นส่งมาได้ถูกต้องจะทำให้ข้อมูลที่ส่งไปมีความผิดพลาดเกิดขึ้น



Air rate (bps)	Time ts1 (ms)	Air Rate (bps)	Time ts1 (ms)
19200	17	2400	76
9600	24	1200	152
4800	43		

ภาพที่ 44 ตัวอย่างช่วงเวลาในการรับ-ส่งข้อมูลระหว่างแพจเกจ

ที่มา : Synergy electronic supply co., Ltd. (2009)

สำหรับใน ขั้นตอนที่ 2 เป็นขั้นตอนในการรับ-ส่งข้อมูลจากอุปกรณ์รับ-ส่งข้อมูลไร้สาย ข้อมูลที่ได้รับมาจากบอร์ดทดลองทางฝั่งส่งจะถูกนำมาเพิ่มเติมเป็นแพจเกจเพื่อให้ข้อมูลสามารถรับส่งกันได้กับอุปกรณ์ทางฝั่งรับที่มีการตั้งค่าให้ตรงกัน ซึ่งในส่วนนี้จะต้องมีการเพิ่มการทำการหน่วงเวลาในตารางดังภาพที่ 44 ซึ่งเป็นตัวอย่างค่าการหน่วงเวลาสำหรับการส่งข้อมูลแบบไร้สาย จากตารางในภาพที่ 44 เป็นค่าการหน่วงเวลาระหว่างข้อมูลแต่ละชุดที่ต่อกัน สำหรับงานวิจัยนี้ใช้ความเร็วในการส่งข้อมูลที่ 9600 bps ซึ่งมีการหน่วงเวลาประมาณ 24 ms เป็นอย่างน้อย

สำหรับในขั้นตอนที่ 3 เป็นการรับข้อมูลที่ส่งมาจากอุปกรณ์รับ-ส่งไร้สายไปยังบอร์ดทดลอง โดยสำหรับบอร์ดทดลองจะทำการสร้างบัฟฟอร์รับข้อมูลจากอุปกรณ์รับส่งข้อมูลแบบไร้สาย เมื่อข้อมูลได้รับข้อมูลจะทำการเก็บค่าข้อมูลไว้และดึงข้อมูลที่ได้ไปใช้งาน สำหรับในการเก็บค่าข้อมูลและแสดงผลบนบอร์ดทดลองจะมีช่วงเวลาในการทำกระบวนการในการเก็บค่าและแสดงค่า โดยกระบวนการที่ทำบนบอรทดทดลองมีดังนี้คือการเก็บข้อมูลเพื่อนำไปใช้งานและกระบวนการในการเคลียร์บัฟเฟอร์เพื่อให้บัฟเฟอร์สามารถรับข้อมูลใหม่เข้ามาได้ซึ่งในแต่ละกระบวนการจะมีช่วงเวลาในการทำงาน

ดังนั้นจากขั้นตอนทั้ง 3 ขั้นจึงเกิดการหน่วงเวลาเกิดขึ้นโดยในการรับ-ส่งข้อมูลให้มีประสิทธิภานจะต้องนำการหน่วงเวลาทั้ง 3 ส่วนไปคำนวณเพื่อให้ข้อมูลที่ส่งไปยังฝั่งรับสามารถส่งข้อมูลได้ครบถ้วนโดยไม่มีการสูญหายของข้อมูลหรือมีความผิดพลาดเกิดขึ้น

ในกรณีนี้หากมีการปรับปรุงขั้นตอนการส่งจะทำให้สามารถส่งข้อมูลได้เร็วขึ้นและลดการหน่วงเวลาลง สำหรับในงานวิจัยนี้เป็นการทดสอบการเข้ารหัสและถอดรหัสช่องสัญญาณซึ่งอุปกรณ์ในท้องตลาดนั้นจะมีการเพิ่มส่วนของการแก้ไขและตรวจสอบความผิดพลาดเข้าไปเพื่อให้เป็นไปตามมาตรฐานในการรับ-ส่งข้อมูลและเพิ่มความน่าเชื่อถือของอุปกรณ์ ทำให้ผลลัพธ์ที่ได้จากการทดลองจะไม่สามารถนำไปเทียบกับทฤษฎีได้ แต่ว่าในการใช้งานจริงการเพิ่มรหัสช่องสัญญาณแบบต่อร่วมเพื่อทำงานร่วมกับอุปกรณ์รับ-ส่งข้อมูลที่มีการเข้ารหัสช่องสัญญาณอยู่แล้วสามารถทำได้เพื่อให้ประสิทธิภาพของการรับ-ส่งข้อมูลสูงขึ้น

6. การทดสอบทั้งระบบ

ในการทดสอบระบบรวมทั้งหมดสามารถทำงานได้อย่างถูกต้องทั้งการเข้ารหัสและถอดรหัสแบบต่อร่วมและการส่งส่งผ่านข้อมูลจากคอมพิวเตอร์ฝั่งส่งผ่านบอร์ดทดลองไปยังฝั่งรับ

ในกรณีที่เกิดความผิดพลาดขึ้นรหัสแบบต่อร่วมสามารถช่วยในการแก้ไขความผิดพลาดได้ตามทฤษฎี แต่ในงานวิจัยนี้ไม่ได้ทำการทดลองในการส่งข้อมูลขนาดใหญ่และความเร็วสูงมากในการทำงาน สำหรับเป้าหมายของงานวิจัยนี้เพื่อใช้สำหรับการทดสอบทฤษฎีเบื้องต้นและทดลองบนบอร์ดทดลอง FPGA ที่ใช้ซอฟท์คอร์โปรเซสเซอร์ในการควบคุม แต่ว่าการรับส่งข้อมูลมีการหน่วงเวลาในขั้นตอนของการส่งข้อมูลจากบอร์ดทดลองก่อนไปยังอุปกรณ์ไร้สายจึงทำให้การรับส่งข้อมูลยังช้า สำหรับการหน่วงเวลาเพื่อทำให้ข้อมูลที่ส่งและรับสามารถรับข้อมูลได้อย่างแน่นอนไม่เกิดการหายไปของข้อมูล สำหรับการรับส่งข้อมูลจากคอมพิวเตอร์และบอร์ดทดลอง FPGA นั้นจะเลือกส่งข้อมูลครั้งละ 10 สัญลักษณ์และสัญลักษณ์สำหรับเคลียร์หน่วยความจำอีก 4 สัญลักษณ์ซึ่งสามารถรับส่งข้อมูลได้และใช้ร่วมกับการเข้าและถอดรหัสได้

สรุปและข้อเสนอแนะ

สรุป

ในงานวิจัยชิ้นนี้ต้องการสร้างและทดสอบระบบบรหัสแบบต่อร่วมลงบนซอฟท์คอร์โปรเซสเซอร์โดยส่งผ่านช่องสัญญาณแบบไร้สาย สำหรับการเข้ารหัสแบบต่อร่วมได้ทำการทดลองด้วยรหัสภายนอกแบบรหัสคอนโวลูชัน (3,2,2) ต่อร่วมกับรหัสภายในแบบรีดโซโลมอน (63,51) และทดสอบเพิ่มเติมด้วยรหัสบีซีเฮช (31,26) สำหรับตัวถอดรหัสภายในใช้วิธีการของ Beklekham-Massey สำหรับรีดโซโลมอนและทดสอบเพิ่มเติมกับการถอดรหัสซอฟท์ลิสวิเทอร์บี สำหรับการถอดรหัสภายนอกใช้การถอดรหัสแบบเวกเตอร์ซิมโบล

ในการออกแบบสำหรับระบบบรหัสต่อร่วมใช้งานซอฟท์คอร์โปรเซสเซอร์สำหรับเป็นส่วนควบคุมและใช้ภาษา C ในการเขียนโปรแกรมควบคุม อุปกรณ์ที่ต่อร่วมกับบอร์ดทดลองประกอบไปด้วยพอร์ตอีเทอร์เน็ตสำหรับรับส่งข้อมูลจากคอมพิวเตอร์ พอร์ต RS-232 สำหรับรับ-ส่งข้อมูลจากบอร์ดทดลองไปยังอุปกรณ์รับ-ส่งสัญญาณแบบไร้สาย และส่วนของหน่วยความจำขนาด 1 Mb จำนวน 2 ชิ้น ผลการทดสอบสามารถทำการรับส่งข้อมูลจากพอร์ตอีเทอร์เน็ทและพอร์ต RS-232 ได้

จากผลทดสอบการทำงานของการเข้ารหัสแบบต่อร่วม สามารถเข้ารหัสได้ถูกต้องเป็นไปตามทฤษฎีสำหรับรหัสคอนโวลูชัน (3,2,2) รหัสบีซีเฮช (31,26) และรหัสรีดโซโลมอน (63,51) โดยรหัสภายนอกแบบคอนโวลูชัน (3,2,2) เป็นการเข้ารหัสแบบนอนไบนารีการเข้ารหัสครั้งละ 102 บิตจำนวน 14 สัญลักษณ์ และการเข้าภายในแบบไบนารีบีซีเฮช (31,26) ทำการเข้ารหัสเป็นบิตครั้งละ 26 บิต สำหรับในการทดสอบระบบรวมเลือกใช้รหัสรีดโซโลมอน (63,51) โดยมีขนาดสัญลักษณ์เป็น 6 บิตต่อสัญลักษณ์จำนวน 51 สัญลักษณ์

ในส่วนของการทดสอบการถอดรหัสจะแบ่งเป็น 2 ส่วนคือ การถอดรหัสภายในโดยเลือกใช้ลิฟออฟทูวิเทอร์บีสำหรับการเข้ารหัสแบบบีซีเฮช (31,26) โดยทดสอบทั้งการตัดสินใจแบบฮาร์ดและแบบซอฟท์ สำหรับรหัสรีดโซโลมอนที่ใช้ในการทดสอบในการส่งผ่านช่องสัญญาณไร้สายจะใช้การถอดรหัสด้วยวิธีการของ Berlekamp-Massey ซึ่งผลลัพธ์ที่ได้ออกมาจะนำไปใช้กับการถอดรหัสแบบเวกเตอร์ซิมโบลที่เป็นการถอดรหัสภายนอก สำหรับการถอดรหัสแบบเวกเตอร์

ซิมโบลเลือกใช้ขนาดสัญลักษณ์ 102 บิตเพื่อถอดรหัสคอนโวลูชัน (3,2,2) ผลลัพธ์ที่ได้สามารถแก้ไขข้อมูลที่เกิดความผิดพลาดได้ตามทฤษฎีและถอดรหัสข้อมูลได้อย่างถูกต้อง

ในการทดสอบสำหรับการรับส่งสัญญาณแบบไร้สายเพื่อส่งข้อมูลเลือกใช้อุปกรณ์ส่งสัญญาณ D-RFN96 มีการมอดูเลตแบบ GFSK (Gaussian Frequency shift keying) ที่อยู่ในช่วงความถี่ 433 MHz ส่วนภาครับจะใช้อุปกรณ์เดียวกันเพื่อรับสัญญาณโดยการส่งข้อมูลจะรับข้อมูลจากพอร์ต RS-232ของฝั่งส่งเพื่อส่งไปยังฝั่งรับ โดยการส่งข้อมูลจะทำการส่งข้อมูล 23 ไบต์ ประกอบไปด้วยข้อมูล 21 ไบต์และเฮดเดอร์ 2 ไบต์ซึ่งสามารถส่งข้อมูลได้และมีความผิดพลาดขึ้นเพื่อทดสอบรหัสแบบต่อร่วมต่อไปแต่ว่าได้มีการทำการหน่วงเวลาเพื่อให้ข้อมูลที่ส่งมาสามารถที่จะเก็บข้อมูลได้ทางฝั่งรับซึ่งหากทำการหน่วงเวลาน้อยเกินไปข้อมูลบางส่วนจะขาดหายไป

ในการทดสอบทั้งระบบสามารถทำงานได้เป็นไปตามทฤษฎีและทดสอบการทำงานบนบอร์ดทดลองที่ควบคุมด้วยซอฟท์คอร์โปรเซสเซอร์ที่สร้างบนบอร์ดทดลอง FPGA และส่งข้อมูลผ่านช่องสัญญาณ สำหรับในงานวิจัยนี้ได้ทำการทดสอบเบื้องต้นพบว่ารหัสแบบต่อร่วมสามารถทำงานได้อย่างถูกต้องบนช่องสัญญาณจริง

ข้อเสนอแนะ

สำหรับการใช้งานซอฟท์คอร์โปรเซสเซอร์สามารถกำหนดที่อยู่ของข้อมูลได้เพื่อช่วยในการระบุตำแหน่งที่ชัดเจน สำหรับการการใช้โปรแกรมภาษาซียังใช้ในระดับพื้นฐาน หากสามารถใช้เทคนิคต่าง ๆ ในการเขียนโปรแกรมจะทำให้ประสิทธิภาพของระบบดีขึ้นกว่าเดิม

ประสิทธิภาพของการทำงานของซอฟท์คอร์โปรเซสเซอร์มีความเร็วในการทำงานน้อยกว่าการออกแบบด้วย VHDL ดังนั้นการทำงานบางอย่างควรใช้ทั้งการควบคุมโดยใช้ภาษา VHDL และการใช้ซอฟท์คอร์โปรเซสเซอร์เพื่อเพิ่มประสิทธิภาพให้มากขึ้น

ถึงแม้ว่าการใช้งานพอร์ตอนุกรม RS-232 นั้นมีความเร็วที่ช้าแต่แลกมาด้วยความซับซ้อนที่น้อยกว่าของระบบ ทำให้การเลือกใช้พอร์ตอนุกรม RS-232 เหมาะสำหรับมาทำงานเพื่อทดสอบระบบเบื้องต้น แต่อย่างไรก็ตามพอร์ตต่าง ๆ ที่มีบนบอร์ดทดลองยังมีให้ใช้อย่างหลากหลาย เช่น พอร์ต RS-485 ซึ่งเป็นพอร์ตอนุกรมอีกแบบที่น่าสนใจเนื่องจากมีความเร็วสูงกว่าพอร์ต RS-232 , ระยะของสายส่งยังไกลกว่าพอร์ต RS-232และกำจัดปัญหาที่เกิดจากสัญญาณกราวด์ของอุปกรณ์ฝั่ง

รับและฝั่งส่งไม่เท่ากันได้ สำหรับการใช้พอร์ตอนุกรมนี้จะใช้ในงานอุตสาหกรรมที่ต้องการการสื่อสารกันเพื่อเชื่อมต่อกันกับหลายอุปกรณ์ พอร์ต USB ซึ่งเป็นอุปกรณ์ที่มีความเร็วที่สูงและยังมีอุปกรณ์ออกมารองรับมากขึ้นซึ่งการหาอุปกรณ์แบบไร้สายที่มีการเชื่อมต่อกับพอร์ต USB จะทำได้ง่ายแต่ว่าในบางอุปกรณ์จะต้องมีการทำโปรโตคอลเพื่อให้ข้อมูลที่อ่านจากบอร์ดทดลองกับอุปกรณ์สามารถสื่อสารกันได้จึงเพิ่มความซับซ้อนให้กับระบบมากขึ้น

สำหรับอุปกรณ์รับ-ส่งสัญญาณแบบไร้สายที่เลือกใช้งาน RS-232 เนื่องจากสามารถหามาทดสอบได้ง่ายจากท้องตลาดซึ่งอุปกรณ์รองรับการสื่อสารแบบอนุกรม RS-232 และได้สร้างตัวควบคุมในการทำงานให้เพื่อพร้อมสำหรับการใช้งานได้ทันที อย่างไรก็ตามยังมีอุปกรณ์ที่น่าสนใจในการทดสอบเพื่อเพิ่มความเร็วของการรับ-ส่งสัญญาณและการใช้งานที่หลากหลายขึ้นเช่น Zigbee สำหรับอุปกรณ์ชนิดนี้สามารถสร้างเป็นเน็ตเวิร์คได้และมีความเร็วที่สูงในระดับหนึ่งซึ่งการสร้างเน็ตเวิร์คจะช่วยเพิ่มแง่มุมในการทดสอบระยะของอุปกรณ์และการใช้งานรหัสช่องสัญญาณบนเน็ตเวิร์คแบบใหม่อีกด้วย หรือการใช้งานชิพการส่งสัญญาณไร้สายโดยตรงเช่น CC1100,CC2500 ของบริษัท Texus Instruments ซึ่งโดยส่วนมากจะทำการเชื่อมต่อกับอุปกรณ์ภายนอกแบบ SPI ซึ่งการใช้ชิพประเภทนี้จะต้องทำการสร้างวงจรในการรับ-ส่งข้อมูลจากบอร์ดทดลองไปยังตัวชิพเองและต้องทำมีการทำการกำหนดค่าพารามิเตอร์ต่าง ๆ บนตัวชิพเองเพื่อให้สามารถรับส่งข้อมูลได้เร็ว ซึ่งในส่วนนี้จะต้องมีความรู้ความเข้าใจในการออกแบบระบบการรับส่งข้อมูลแบบไร้สายและความรู้ด้านอิเล็กโทรนิคเป็นอย่างสูงเพื่อให้อุปกรณ์เหล่านี้สามารถทำงานได้อย่างเต็มประสิทธิภาพ

เอกสารและสิ่งอ้างอิง

เกรียงศักดิ์ หงษ์ชุมแพ. 2544. **มาตรฐานอินเตอร์เน็ตและโปรโตคอล**. พิมพ์ครั้งที่ 1. บริษัทศูนย์การศึกษาทางไกล(ประเทศไทย)จำกัด, กรุงเทพ.

พิสิฐ วนิชชานันท์, ปิยะ โควินท์ทวีวัฒน์, อุศนา ตัณฑุลเวศม์, เกียรติศักดิ์ ศรีพิมานวัฒน์, กำพล วรดิษฐ์, ปรมินทร์ แสงวงษ์งาม, ดิสพล ฉ่ำเฉียวกุล และ สัญฉกร วุฒิสิทธิกุลกิจ. 2552. **ทฤษฎีรหัสช่องสัญญาณ**. พิมพ์ครั้งที่ 1. สถาบันวิจัยและพัฒนาอุตสาหกรรมโทรคมนาคม, กรุงเทพ.

พงษ์พิสุทธิ์ นรดี. 2553. **ต้นแบบเครื่องถอดรหัสภายในสำหรับตัวถอดรหัสคอนโวลูชันภายนอก.** วิทยานิพนธ์ปริญญาโท, มหาวิทยาลัยเกษตรศาสตร์.

รัชนนท์ อินทรสกุล. 2551. **การออกแบบและสร้างระบบเพื่อใช้สำหรับการเข้ารหัสและถอดรหัสสัญลักษณ์นอนไบนารีด้วยบอร์ด FPGA**. วิทยานิพนธ์ปริญญาโท, มหาวิทยาลัยเกษตรศาสตร์.

อรรถภัทร์ สืบเนื่อง. 2552. **ชิ้นงานต้นแบบของเครื่องถอดรหัสด้วยวิธีเวกเตอร์ซิมโบลดีโคดดิงสำหรับรหัสแบบคอนโวลูชัน (3, 2, 2) ที่มีชุดข้อมูลขาเข้า 2 ตัวเลือก บนบอร์ดทดลอง FPGA.** วิทยานิพนธ์ปริญญาโท, มหาวิทยาลัยเกษตรศาสตร์

Aditya, G., A. S. Raj., R. P. Behera., N.Murali and S.A. V. Murty. 2011. Design & development of soft-core processor Based remote Terminal units for Nuclear Reactors. pp.1-4**. Field-programable Technology (FPT)**. 12-14 December 2011

Bose, R. C. and D. K. Ray-Chaudhuri. 1960. On a class of error correcting binary group codes. **Informantion and Control**. 3: 68-79.

Carvalhosa, A., P. Machado., A. Sousa and J.C. Alves. 2009. Soft core robot with joint wheel motion controller. pp.3168-3173. **Industrial Electronics Conference (IECON).** 3-5 November 2009.

Do-Hoon. K., K. M. Kim and C. Lee. 2012. An application of MB-OFDM UWB technology for wireless speaker systems. **IEEE Transections on Consumer Electronics.** 58: 1169-175.

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

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

Hocquenghem, A. 1959. Codes corecteurs k'erreurs. **Chiffres**. 2: 147-156.

John, F. W. 2000. **Digital Design Principles and Practices**. 3rd edition ed. Prentice Hall,Upper Saddle River, NJ.

Lin, S. and D. J. Costello Jr. 2004. **Error Control Coding**. 2nd edition ed. Pearson Education, Upper Saddle River, NJ.

Lionel, B. and N. Dominique. 2007. Choice and implementation of a reed Solomon code for power low data rate communication sysytems. **Radio and Wireless Symposium**, pp. 365-368.

Mehra, R., G. Sini and S. Singh., 2011 FPGA based high speed BCH encoder for wireless communication applications. pp.576-579. **Communication Systems and Network Technologies(CSNT)**. 3-5 June 2011.

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

Mizuochi, T., K. Yoshiaki., M. Yoshikuni. Inoue. And *et al.* 2009. Experimental demonstration of concatenated LDPC and RS codes by FPGA emulator. pp.1302 1304, **Photonics Technology Letters.**

Synergy Electronic Supply co., Ltd. 2008. Jz863 micro power wireless data module user's manual**.** User manual. Available Source: http://www.synes.co.th, January 20, 2013.

Thonchai, J., V. Suktalordcheep and U. Tuntoolavest. 2013. Lab prototype of list-of-2 soft viterbi decoder for a BCH inner code in a generalized concatenated coding system. **The International Conference on Information and Communication Technology for Embeded Systems (ICICTES)**. 24-26 January 2013, Samutsongkhram, Thailand.

Tiwari, B. and M. Rajesh. 2012. Design and implementation of Reed-Solomon decoder for 802.16 networking using FPGA. pp. 1-5. **Signal and Processing Computer and Control (ISPCC)**. 15-17 March 2012.

Tuntoolavest, U. and P. Noradee. 2010. Lab prototype of a list-of-2 viterbi decoder a diversity inner decoder for the outer vector symbol decoder. pp. 973-977. **International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON)**. 19-21 May 2010.

Yi, H. C., C. L. Chu., C. C. Yeh and K. F. Lin. 2012. FPGA implementation and verification of Reed-Solomon (63,47,8) Code in SDR system. pp. 100-103. **Instrumentation Measurement Computer Communication and Control (IMCC)**. 8-10 December 2012.

Yi-Min, L., J. Y. Wu., C.C. Lin and H. C. Chang. 2009. A long block length bch decoder for DVB-S2 application. pp.171-174. **International Symposium on Integrated Circuits (ISIC'09).** 14-16 December 2009.

Zhi, c., L. Miao and J. Shen. 2012 Design of SINS/GPS integrated navigation system based on dual NiosII soft-core. pp. 198-202. **International Conference on Advanced Computational Intelligence (ICACI).** 18-20 October 2012.

ภาคผนวก

VHDL Design of a Convolutional Concatenated Encoding System

Usana Tuntoolavest* and Jatupon Thonchai
Department of Electrical Engineering, Faculty of Engineering
Kasetsart University, Bangkok, 10900, Thailand
Tel: +66-2-9428555 ext 1565, Fax: +66-2-942-8555 ext 1550
E-mail: fengunt@ku.ac.th and g5314501171@ku.ac.th

Abstract—Previous work showed the lab-prototype of a 32-bit nonbinary convolutional encoder on an FPGA board. This paper proposes the design of the overall concatenated encoding system, of which both the inner and the outer codes are convolutional. Its application is to be the encoding system for the LVA-VSD (List Viterbi Algorithm–Vector Symbol Decoding) coding system for wireless fading channels. LVA-VSD is a concatenated coding system with a diveristy inner decoder. Our system integrates the outer encoder and more than one parallel inner encoders in the same unit for faster processing. In this design, a new symbol size (64-bit) is added as an option and a two-stage multiplexing of the inner encoded sequences is designed particularly for burst error channels. The Simulation was done with VHDL language using Xilinx Spartan-3AN Evaluation Kit FPGA platform. Results show that the encoding process work exactly as designed.

Keywords- outer convolutional encoder; VHDL; LVA-VSD; convolutional concatenated code; Vector Symbol Decoding

I. INTRODUCTION

The structure of concatenated codes was first proposed by Forney in 1966 [1] as a way to construct long good code from shorter codes. A simple single-level concatenated code consists of two layers of codes: an inner binary code and an outer nonbinary code. The most widely used outer codes are Reed-Solomon codes [2] because they are nonbinary codes with many good properties, especially the maximum minimum-distance property. For the inner codes, both block codes and convolutional codes are used since the decoding methods for good binary codes are well known such as using shift register circuit for decoding binary cyclic codes [3] or using Viterbi algorithm [4] for decoding binary convolutional codes. An example of a well known concatenated code is the one recommended by CCSDS (Consultative Committee for Space Data Systems) for Telemetry Channel Coding [5]. This code uses the outer (255,223) Reed-Solomon code with the inner (2,1,6) convolutional code.

In 1990, Metzner and Kapturowski [6] proposed a general decoding algorithm that can be applied for any linear block or convolutional codes that use large nonbinary symbols. This algorithm is called "Vector Symbol Decoding" or VSD. With this algorithm, it became easy to use convolutional codes as the outer code of concatenated codes. VSD for nonsystematic convolutional codes was presented by Tuntoolavest and Metzner in [7]. One interesting property of VSD is that it can easily select the correct input symbol from a list of alternative choices for that symbol. This idea was first proposed for block VSD in [8] and convolutional VSD in [7]. The details of the VSD principle, algorithm and some implementations can be found in [9]. Due to this ability to select the correct symbol from a list, it is beneficial for the concatenated code that employs VSD to use a diversity inner decoder. The diversity inner decoder is that decoder that can provide more than one decoded sequence to the outer decoder. These set of decoded sequences should also be ordered based on their likelihood. A decoder that can provide list of decoded sequences in the order of their likelihood is the List Viterbi decoder proposed by Seshadri and Sundberg [10] in 1994. Since List Viterbi algorithm is most suitable with convolutional codes, the obvious choice for the inner code is therefore a convolutional code.

Previous work [11] implemented a nonbinary convolutional encoder on an FPGA (Field Programmable Gate Array) board using a USB module as the interface between the board and a computer. The inner encoder was not included. In this paper, we design the overall encoding part of a concatenated coding system that integrates a convolutional outer encoder and three parallel convolutional inner encoders. Instead of fixing the symbol size at 32-bit only, this new design allows the user to select different symbol sizes. In addition, the encoding speed is tripled by using the integrated three parallel inner encoders instead of one separated inner encoder. This design is also assumed to be used with the new interface, namely, Ethernet. In the previous design using USB module, the input came in 8-bit at a time and there was a need to assemble 4 bytes into one 32-bit symbol.

With the assumption of using Ethernet, the input can come in as a file and can be buffered inside the board before the encoding process. By planning to buffer the input data sequence and encoded sequence inside the board, it is much more flexible to allow different symbol sizes. This option will be beneficial in the future investigation on the effect of chosen appropriate symbol sizes based on the average length of burst errors of a wireless fading channel.

Section II covers the encoding of convolutional codes and the encoding of concatenated codes with inner and outer convolutional codes. Section III describes the design of the

Supported by Kasetsart University Research and Development Institute

proposed system. Section IV explains the method. Section V shows the simulation results. Section VI and VII discusses and concludes the results respectively.

II. BACKGROUND

An (n,k,m) convolutional encoder encodes each set of k-bit input into an n-bit output. The encoder can be considered as a sequential circuit with memory of m units. The parameters k ,n and m are usually small integers. For this paper, we use a (3,2,2) convolutional code over GF(2^{32}) or over GF(2^{64}) as the outer code and a (2,1,4) convolutional code over GF(2) as the inner code. As explained in [11], the encoding process of a nonbinary encoder is the same as that of the binary encoder except that the exclusive-OR (XOR) operations are performed on the nonbinary symbols instead of the 1-bit binary symbol. Each memory unit also contains 32 or 64 bits according to the selected symbol size.

The encoding process of the proposed convolutional concatenated code with (n_1,k_1,m_1) outer convolutional code and (n_2,k_2,m_2) inner convolutional code is as follows:

1. Divide the input data sequence into symbols.
2. Each set of k_1 symbols are input into the outer encoder. The outer encoder then performs the appropriate XOR operations at the symbol level based on the structure of the code and then output n_1 symbols for each k_1 input symbols. The details and example of the encoding at the symbol level has been shown in [11].
3. Then each output symbol of the outer encoder is input to the inner encoder. The inner encoder performs the appropriate XOR operations at the bit level and output n_2 bits for each k_2 input bits.
4. The inner encoder for our proposed system will clear the memory after the end of the input bits in each input symbols. This is different from the CCSDS standard code that the inner encoder treats that sequence of input symbol as one long binary input sequence to the inner encoder described in [5].
5. All outputs of the inner encoder are then multiplexed and transmitted as the overall concatenated codeword.

Note that the termination or the memory clearing in step 4is necessary if VSD is to be used as the outer decoder.

III. SYSTEM SETUP

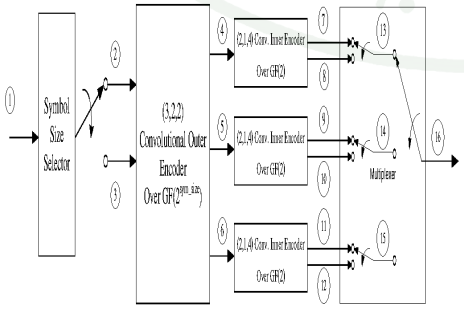


Fig. 1 Overview of the proposed encoding system

The block diagram of the proposed encoding system is shown in Fig. 1. This diagram consists of four main parts. The first part is the symbol size selector, which the user can select different symbol sizes. The second part is the outer encoder, which is a (3,2,2) nonbinary convolutional encoder with the transfer function matrix $G_{322}(D)$.

$$G_{322}(D)=\begin{bmatrix} 1+D+D^2 & D^2 & 1 \\ D & 1+D^2 & 1+D+D^2 \end{bmatrix} \qquad (1)$$

The third part consists of the three parallel binary inner encoders. Each inner encoder has the same transfer function matrix, which is

$$G_{214}(D) = [(1+D+D^4)\ (1+D+D^3+D^4)] \qquad (2)$$

The fourth part is the multiplexing of all inner decoder outputs in the appropriate order into the overall concatenated codeword. It is seen from fig. 1 that the multiplexing is done in two stages. The first stage is the multiplexing of the two branches of each inner encoder. This is done at the bit level since the inner encode is a binary encoder. The second stage is the multiplexing of the three inner encoded codewords. This is done at the inner codeword level because each encoded codeword corresponds to each encoded branch of the outer encoder. An example of the multiplexing process with the bit values at position ① to ⑯ is shown in comparison with the simulation results in section V.

By keeping the structure of the outer code, the burst errors occur from wireless fading channel will not spread out over many outer encoded symbols. Consequently, the outer decoding with VSD will be more effective. If the multiplexing for both stages is done at the bit level, it will be equivalent to performing interleaving with a depth of 6 (from the 6 inner outer branches). This may be suitable for some applications, but it is not good for VSD. This is because each burst error will be likely to affect more symbols causing more error symbols at the outer code level.

Fig. 2 shows the details of the input and output lengths for the second and third parts. These details are illustrated for an example of a 640-bit input data sequence with the selected

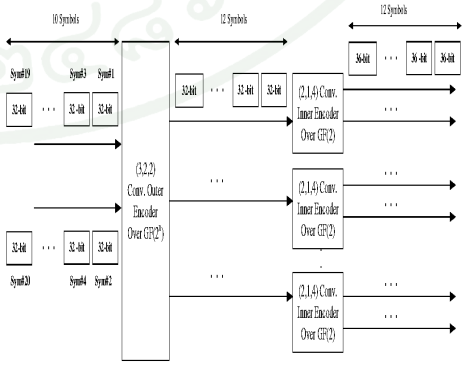


Fig. 2 Details of the input and output sequences for the outer and inner encoding parts

symbol size of 32-bit. In this example, there are a total of twenty 32-bit symbols input into the outer encoder. Since the outer encoder is the (3,2,2) code with $k_1 = 2$, ten of the symbols are input to the upper branch and another ten are input to the lower branch. The orders in which the symbols are input into the outer encoder are shown in Fig. 2. After the outer encoding operation, the outer encoder will output three encoded sequences. Each sequence consists of ten 32-bit symbols plus the two tail symbols that come from the clearing of two memory units ($m_1 = 2$) inside the outer encoder. Each of these symbols is then input into the inner encoder.

There are three inner encoders in this design, so three output sequences can be input into the three inner encoders simultaneously. Each 32-bit symbol then becomes a 32-bit input sequence of the (2,1,4) inner encoder. The inner encoder will encode each symbol separately and clear the memory after the end of each symbol. This clearing of memory can be viewed as adding four "0" bits at the end of the input sequence. Therefore, the input sequence may be considered as containing 36 bits (32 data bits + 4 zero bits).

Since $k_2 = 1$ and $n_2 = 2$, there is one input branch and two output branches from each inner encoder. Each of the two encoded sequences consists of 36 bits. Each inner codeword will then consists of 72 bits after the multiplexing at the bit level of the two inner encoded sequences. The total number of outer symbols remains the same.

IV. METHOD

The program was written with a Hardware Description Language (HDL) called VHDL (Very High Speed Integrated circuit HDL) language [12, 13]. Sample parts of the program are shown in fig. 4. The program accepted 128 bits at a time. The 128-bit sequence is equivalent to four 32-bit symbols or two 64-bit symbols. The VHDL code is fig 3. is from the main program. The parts marked "xxxx" indicated omitted codes when the code idea can be deducted from the context. The main program is divided as follows:

1. Select the symbol size (in the "select mode").
2. Save the input sequence into buffers (in the "split").
3. Encode the outer code with the function: encoder_32bit or encoder_64bit (in the "encode").
4. Send out concatenated encoded sequence.

The omitted code for 32-bit symbol mode is similar to the 64-bit symbol mode. The difference is that the four 32-bit symbols must be input two symbols at a time. Therefore, they are input in two sets. Additional programming is needed to check the end of the two sets and begin with the next 128-bit sequence.

The encoder function is shown in Fig. 4 and the procedures can be divided as follows:

1. The input sequence encoded by the outer encoder first. This outer encoder performs the appropriate XOR operations based on the code structure (in "encode 322").
2. Each of the three encoded symbols is added k_2m_2 zero bits (= 4 bits in this case) at the end (in "add zero bits for inner encoder").

```
architecture Behavioral of scale is
begin
--------Select  mode for symbol size ------------------
Size_selec_encode :process(mode,CLK,address)
--variable output_buff_32:std_logic_vector(215 downto 0);
variableoutput_buffer:std_logic_vector(407 downto 0);
variable memory_32:std_logic_vector(139 downto 0);
variable memory_64:std_logic_vector(267 downto 0);
begin
ifCLK'event and CLK='1' then
if (mode="01" ) then  ------input = 01 mode 32 bit------
xxxxxxxxxxxxxxxxxxxxxxxx
end if;
elsif  (mode="11") then ------------- mode  64 bit---------
-------------------------split-----------------------------------
input_buffer64_1<=input1(127 downto 64);
input_buffer64_2<=input1(63 downto 0);
-------------------------encode-----------------------------------
encoder_64bit(input2_buffer1,input2_buffer2,memory_64,next_symbol,
memory_64,output_buffer);
------------------send out concatenated encoded seqeunce-----------------
case address is
when 15=> output1<=output_buffer(407downto 376);
when 16=> output1<=output_buffer(375downto 344);
xxxxxxxxxxxxxxxxxxxxxxxx
when 26=> output1<=output_buffer(55downto 24);
when 27=> output1<=("--------"&output_buffer(23downto 0));
when 100 =>donext<='0';
when others => empty<='1';
end case;end if;end process;end Behavioral;
```

Fig. 3 Sample parts of the main VHDL code

```
procedure encoder_64bit
  (symbol_input1,symbol_input2:in std_logic_vector(63 downto
0);memory_64_in : in std_logic_vector(267 downto 0);
-------------------  Encode 322  ----------------------------
symbol_outer1 := symbol_input1 xor mem_outer1 xor mem_outer2 xor
mem_outer3;
symbol_outer2 := symbol_input2 xor mem_outer2 xor mem_outer4;
symbol_outer3 := symbol_input1 xor symbol_input2 xor mem_outer3
xor mem_outer4;
              mem_outer2 := mem_outer1;
              mem_outer4 := mem_outer3;
              mem_outer1 := symbol_input1;
              mem_outer3 := symbol_input2;
---------------------  add zero bits for inner  encoder--------------------
xxxxxxxxxxxxxxxxxxxxxxxx
---------------------- encode 214 symbol 1 -----------------------------
for i in 0 to 67 loop
bit_inner1(i) := symbol_outer_tail1(i) xor mem_inner11 xor
mem_inner14;
bit_inner2(i) := symbol_outer_tail1(i) xor mem_inner11 xor
mem_inner13 xor mem_inner14;
              mem_inner14 := mem_inner13;
              mem_inner13 := mem_inner12;
              mem_inner12 := mem_inner11;
              mem_inner11 := symbol_outer_tail1(i);
---------------------- encode 214 Symbol 2 ----------------------------
xxxxxxxxxxxxxxxxxxxxxxxx
-------------------------encode 214 Symbol 3 -----------------------------
xxxxxxxxxxxxxxxxxxxxxxxx
---------------Multiplexer at bit level-----------------
symbol_inner1((2*i)+1 downto 2*i) :=bit_inner1(i)&bit_inner2(i);
symbol_inner2((2*i)+1 downto 2*i) :=bit_inner3(i)&bit_inner4(i);
symbol_inner3((2*i)+1 downto 2*i) :=bit_inner5(i)&bit_inner6(i);
end loop;
---------------- Multiplexer at inner codeword level ---------------
symbol_output:=symbol_inner1&symbol_inner2&symbol_inner3;
```

Fig. 4 Sample parts of the encoder function for the 64-bit symbol

3. The three sequences are then encoded by the inner encoder three inner encoders (in “encode 214 symbol 1,2,3”).
4. The outputs from two branches of an inner encoder are multiplexed at the bit level to obtain the inner encoded sequences (in “Multiplexer at bit level”).
5. The three inner encoded sequences are in multiplexed at the inner codeword level to obtain the concatenated codeword (in “Multiplexer at inner codeword level”).

Again, the parts marked “xxxx” indicated omitted codes when the code idea can be deducted from the context. Note also that when the procedure “encoder_64bit” is finished once, the output is from only one 128-bit input sequence. The actual input sequence for this proposed system is usually much longer than 128-bit and we simply repeat this procedure many times to obtain the whole concatenated codeword. Recall that the inner encoder needs additional k_2m_2 zero bits to clear its memory. The outer encoder also needs k_1 m_1 zero symbols to clear its memory at the end of the long input sequence.

V. RESULTS

The Xilinx Spartan-3AN Evaluation Kit is selected as the simulation platform. A simulation result for 64-bit symbols is shown in fig. 5 and fig 6. To verify this result, the input and output sequences at each step are shown in Table 1. The input sequence is assumed to be an all “1” 128-bit sequence. This can be seen in the first waveform “input[127:0]” in fig. 5, 6 and the “input sequence” in Table 1, which shows the value in hexadecimal number of “fff…ff” for 32 digits. In fig. 5, 6 and table 1, the leftmost digit is the most significant digit and the rightmost digit is the least significant digit in hexadecimal numbers. Note also that the most significant bit is input first into the encoding system.

The second waveform “mode” shows that the 64-bit symbol (mode “11”) is selected. The third waveform is the clock signal “CLK”. The fourth waveform is “address”. This indicates which inner output is to be transmitted out at that time. Recall from fig. 3 that the addresses for 64-bit case are from 15 (binary “0..01111”) to 27 (binary “0..011011”). Consider the output waveform starting in the time interval that the address is 15 (shown as “1111”). This output waveform started with the delay of ½ clock cycle.

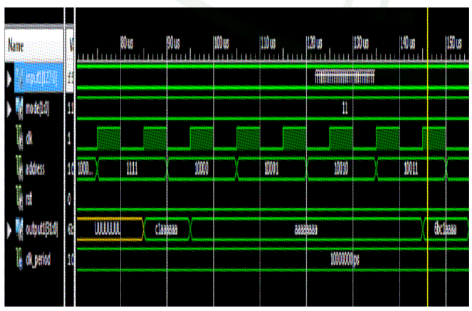


Fig. 5 The first part of the simulation result from the encoding of a 128-bit data sequence with all “1” bits

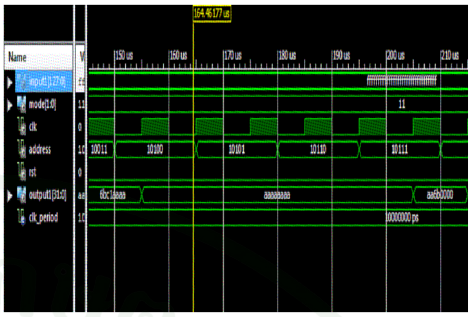


Fig. 6 The second part of the simulation result from the encoding of a 128-bit data sequence with all “1” bits

Table 1 An example of input and output sequences at each position of fig. 1

Position	Description	Hexadecimal Values
①	Input sequence	ffff ffff ffff ffff ffff ffff ffff ffff
②	Input Symbol (1st)	ffff ffff ffff ffff
③	Input Symbol (2nd)	ffff ffff ffff ffff
④	Encoded Sym (1st)	ffff ffff ffff ffff
⑤	Encoded Sym (2nd)	ffff ffff ffff ffff
⑥	Encoded Sym (3rd)	0000 0000 0000 0000
④*	Input inner seq (1st)	ffff ffff ffff ffff 0
⑤*	Input inner seq (2nd)	ffff ffff ffff ffff 0
⑥*	Input inner seq (3rd)	0000 0000 0000 0000 0
⑦	Encoded inner seq (1st)	8fff ffff ffff ffff 7
⑧	Encoded inner seq (2nd)	9000 0000 0000 0000 9
⑨	Encoded inner seq (3rd)	8fff ffff ffff ffff 7
⑩	Encoded inner seq (4th)	9000 0000 0000 0000 9
⑪	Encoded inner seq (5th)	0000 0000 0000 0000 0
⑫	Encoded inner seq (6th)	0000 0000 0000 0000 0
⑬	Inner codeword (1st)	c1aa aaaa aaaa aaaa aaaa aaaa aaaa aaaa 6b
⑭	Inner codeword (2nd)	c1aa aaaa aaaa aaaa aaaa aaaa aaaa aaaa 6b
⑮	Inner codeword (3rd)	0000 0000 0000 0000 0000 0000 0000 0000 00
⑯	Output sequence	c1aa aaaa aaaa aaaa aaaa aaaa aaaa aaaa 6bc1 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aa6b 0000 0000 0000 0000 0000 0000 0000 0000 00

④* means the same position as ④ with four added “0” bits to clear the inner encoder memory

This output waveform is corresponding to the output sequence in Table 1. Due to limited space, the first part of the output waveform is shown in fig. 5 and the second part is shown in fig. 6. In addition, the zero output value at the end of the waveform is omitted. The value “aaaaaaaa” in the middle of the output waveform is actually consisted of 24 hexadecimal numbers “a”, which equal to 96 bits. It is displayed this way since the values do not change for these three time intervals. The results in fig. 5 and 6 are the same as in table 1 and the VHDL code works exactly as designed. Note that the output sequence in table 1 is only the first part of the concatenated codeword since the input data sequence for this application is usually much longer than 128-bit.

VI. Discussions

By programming with the VHDL language, the simulation can be done with the platform of Xilinx Spartan-3AN Evaluation Kit FPGA board. The sample simulation result in fig. 5 and the example in table 1 show that the VHDL design for the proposed encoding system work properly as designed. With the option of the size selector, the VHDL code for the 32-bit symbol is actually a little more complicated than the one for 64-bit shown in fig. 3. This is because it needs to divide each block of 128-bit input sequence into two sets of two 32-bit symbols instead of only one set of two 64-bit symbols. Therefore, additional VHDL code is needed to keep track of which symbols set it is using and when to begin the new block of 128-bit. Nevertheless, the result and sample part of VHDL code are shown for a 64-bit symbol case since it is the additional option in this new design.

This convolutional concatenated encoding system was designed to be compatible with the LVA-VSD concatenated coding system that uses three parallel inner decoders. It is also designed particularly for wireless fading channel where the main type of errors is burst error. This leads to the use of three parallel inner encoders and the two-stage multiplexing process. The multiplexer is designed this way to keep the structure of the inner code and the outer code separated. This contains the burst errors from fading channel in fewer outer symbols than multiplexing all six inner encoded branches at bit level. This is because the latter case is the same as interleaving the outer together, which will spread out into many more outer encoded symbols. Consequently, the number of symbol errors that VSD must correct is fewer with the two-stage multiplexer.

VII. Conclusions

The presented encoding design integrates the outer encoder and the three parallel inner encoders with the option of symbol size selector. This design is for the LVA-VSD concatenated coding system and particularly for channels with burst errors. The authors plan to implement this design in an FPGA board with the Ethernet interface and RAM in the near future.

Acknowledgment

The authors would like to thank Mr. Rachanon Intharasakul for his VHDL program, which is the seed for our VHDL program in this paper.

References

[1] G.D. Forney, Jr., Concatenated Codes, MIT Press, Cambridge Mass., 1966.

[2] I.S. Reed and G. Solomon, "Polynomial codes over certain finite fields," SIAM Journal on Applied Mathematics, vol.8, 1960, pp. 300-304.

[3] S. Lin and D. J. Costello Jr., Error Control Coding, 2nd edition, Pearson Education, Upper Saddle River, NJ, 2004.

[4] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans. Inf. Theory, vol. IT-13, pp. 260-269, Apr 1967.

[5] Consultative Committee for Space Data Systems, Telemetry Channel Coding, Recommendation for Space Data System Standards, CCSDS 101.0-B-6, Blue Book, Issue 6, Oct. 2002.

[6] 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.

[7] 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.

[8] J.J. Metzner, "Vector symbol decoding with list inner symbol decisions," IEEE Trans. Comm., vol.51, Issue3, pp.371–380, Mar 2003.

[9] Usana Tuntoolavest, Vector Symbol Decoding for Wireless Fading Channels, VDM publishing, 184 pages, 2009, ISBN-10: 3639132866, ISBN-13: 978-3639132861

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

[11] U. Tuntoolavest and R. Intharasakul, "Nonbinary Convolutional Encoder for Vector Symbol Decoding on FPGA board." 2006 10th Inter. Conf. on Comm. Tech. Proceedings (ICCT2006), vol. 1, pp.716-719, November 27-30, 2006, Guilin, China.

[12] D. L. Perry, VHDL: programming by example. 4th edition. McGraw-Hill, Boston, 2002.

[13] A. P. Pedroni, Circuit design with VHDL. The MIT Press, 2004.

Lab Prototype and Performance Investigation of List-of-2 Soft Viterbi Decoder for a BCH Inner Code

Jatupon Thonchai, Vasin Suktalordcheep and Usana Tuntoolavest*
Department of Electrical Engineering
Kasetsart University, Bangkok, Thailand
Tel: +66-2-7970999 ext 1565, Fax: +66-2-7970999 ext 1550
E-mail: g5314501171@ku.ac.th, g5314501317@ku.ac.th, fengunt@ku.ac.th

***Abstract*—**List-of-2 soft Viterbi algorithm (VA) decoder provides a list of two possible decoded sequences in the order of their likelihood. It can decode convolutional codes and any block codes represented in a trellis diagram. It is an important component of a generalized concatenated coding system that allows the inner and outer codes to be any combination of convolutional and block codes with some modifications. This paper implemented the list-of-2 soft VA and the concatenated encoder in a Nanobaord3000 with Xilinx Spartan 3AN FPGA chip. The results from the C++ simulation and the lab prototype were exactly the same in various fading channel conditions. Thus, the lab prototype worked properly as designed. The results showed that the soft VA was better than the hard VA as expected. The presence of Doppler increased the decoding error probability. The Rician fading channel provided noticeable improvement compared to the Rayleigh channel.

Keywords—list-of-2 soft Viterbi, VSD, prototype, inner decoder, generalized concatenated codes, fading channels.

I. Introduction

Serial concatenated codes were first proposed by Forney in 1966 [1]. It was originally called "concatenated codes". Since turbo codes [2] were proposed and used widely, the original one was sometimes called serial concatenated codes to distinguish them from turbo codes that were sometimes referred to as parallel concatenated codes [3]. The outer code of a serial concatenated code was usually a Reed-Solomon (RS) code [4] as described in CCSDS standards [5]. RS codes are block codes with nonbinary symbols from GF(q) [6]. The inner code is usually a binary block or a binary convolutional code. The inner decoder depends on the inner code selected. The outer decoder is a RS code decoder.

In 2011, the concept of a generalized concatenated decoder for serial concatenated codes was presented [7]. The decoding principle proposed could be applied to any combinations of block and convolutional codes for inner and outer codes. In other words, the inner code can be either of the two types of codes and the outer code can also be either of the two types. This was possible because the principle of Vector Symbol Decoding (VSD), the outer decoding algorithm, can be applied to both types of codes with some modifications in the implementation [8]. Moreover, list-of-2 Viterbi decoding, the inner decoding algorithm, can be used to decode block codes as well as convolutional codes by representing the block codes in the trellis diagram [6]. The list Viterbi algorithm (LVA) was presented by Seshadri and Sundberg in [9]. LVA provided an ordered list of possible decoded sequences for each received sequences based on the likelihood. In 2011, a soft decoding for binary cyclic codes was proposed, but it was a "light soft version of permutation decoding" [10]. It was not a Viterbi decoding and no list decoding was used. The soft Viterbi decoding combined with list decoding was described and analyzed in 2011 [11]. Metzner presented the algorithm of VSD with list inner decisions in [12]. VSD with list for various structures of convolutional codes was analyzed in [13]. LVA was mentioned as a way to provide list inner decisions for VSD in [14]. LVA with only two choices (list-of-2 VA) was selected as the inner decoder for VSD in [15].

In [11], the concept of this coding system was presented with the performance comparison of inner decoder only. Specially, the comparison was done for the algebraic decoding, hard Viterbi and list-of-2 soft VA decoding of a BCH code in the AWGN (Additive white Gaussian noise) channel only. It was concluded from the results that the algebraic decoding and the hard decision VA provided basically the same decoding error probability. However, the soft list-of-2 VA provided much lower decoding error probability. In [16], the lab prototype of list-of-2 VA was implemented for a convolutional inner code, while the one in this paper is for a block inner code. The selected board was also different. The prototype in [16] was programmed with VHDL (VHSIC hardware description language). The current prototype was programmed with C, which makes it much easier to modify.

In this paper, we implemented the hard decision and the list-of-2 soft decision VA in an FPGA (Field Programmable Gate Array) board, which was the Nanoboard 3000. We also implemented other components of the generalized concatenated coding system, which were a convolutional outer encoder and the BCH inner encoder in a board. We demonstrated an example of a convolutional outer code instead of a block code because standard serial concatenated codes usually used RS codes, which were nonbinary block codes as the outer codes. Therefore, if a convolutional outer code was shown, it would

Supported by Kasetsart University Research and Development Institute

emphasize the generality of this coding system. For the block outer code case such as RS codes, the outer encoder will be the nonbinary block encoder such as the shift register circuits and the decoder will be VSD for block codes. We also extend the performance investigation of the inner decoder from AWGN in [11] to fading channels in this paper. These models were more realistic for its wireless applications. The wireless fading channels were Rayleigh and Rician fading channels with AWGN. The Doppler effect was also investigated for both fading models. For the performance investigation, all-zero code words was assumed for simplicity since this assumption is valid with no loss of generality.

In the near future, the practical outer decoder of this concatenated coding system will be added to complete the system. This outer decoder will use the Vector Symbol Decoder (VSD) with list decoding [8,12]. In addition, another function to map the decoded sequence to its corresponding data sequence needs to be included. This conversion is not trivial for a nonsystematic convolutional code. Since the nonsystematic convolutional code provided better performance than the systematic one, it was preferable and the outer decoder would need to include this mapping.

II. SYTEM DESIGN

A. *Block Diagram of the System*

The presented concatenated code consists of an inner block code and the outer convolutional code as shown in fig. 1. The modulation, demodulation and the channel were modeled with Matlab. The encoders and the decoders had been previously simulated with C++. In this paper, the inner and outer encoders as well as the hard Viterbi and the list-of-2 soft Viterbi decoders were implemented in a nanoboard3000 with an FPGA chip.

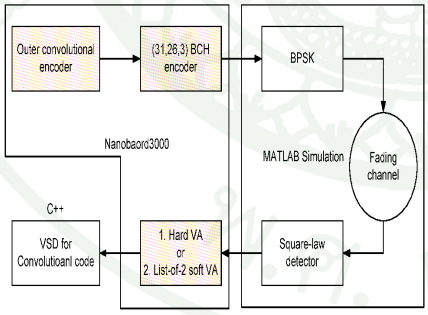


Figure 1. The concatenated coding system

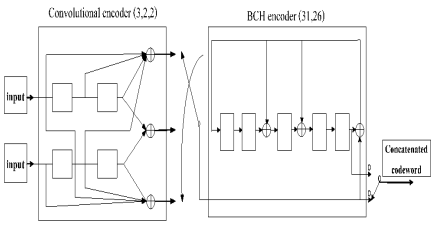


Figure 2. The structure of the selected concatenated encoder

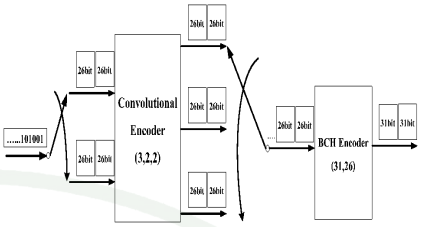


Figure 3. The inputs and outputs of the inner and outer encoders

In the next phase, the outer decoder (VSD) will be implemented in the same board. The standard decoder for a BCH code is the algebraic decoder. In addition, Viterbi can be used to decode block codes that can be represented by a trellis diagram [6]. Therefore, both hard decision and list-of-2 soft decision Viterbi were also used as the inner decoder. The fading channels were Rayleigh and Rician channels with and without Doppler effect.

The inner code was a single error correcting binary BCH code of length 31. The outer code was a (3,2,2) nonbinary convolutional code. The detailed structure of the selected inner and outer encoders is illustrated in fig. 2. Each memory block of the (3,2,2) nonbinary code contains 26 bits, but that of the binary BCH code contains 1 bit. Fig. 3 shows the input, output and the interface inside the concatenated encoders. It can be seen that each set of two 26-bit symbols were encoded by the outer encoder into three 26-bit symbols. These three 26-bit symbols were then multiplexed and input to the inner encoder. Each 26-bit sequence was encoded into a 31-bit inner code word. The encoded sequence was modulated by a binary phase shift keying (BPSK) scheme as described in fig. 1 and transmitted through a fading channel.

B. *Hardware Aspect*

The hardware was designed with Altium Designer. The FPGA board used was the Nanoboard 3000. The FPGA chip was the Xilinx Spartan 3AN device (XC3S1400AN-4FG676C). This chip consisted of 140,000 gates. It contained the TSK-3000A 32-bit RISC (Reduced Instruction Set Computer) processor as shown in fig. 4. The open bus was

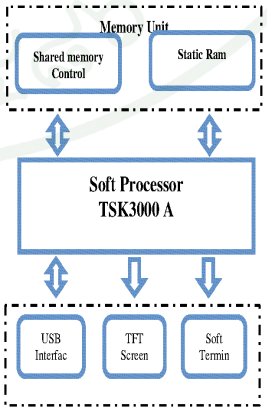


Figure 4. System design in hardware

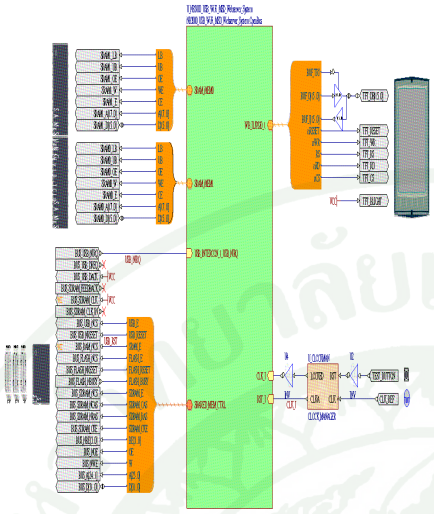


Figure 5. The schematic diagram of the system

designed with "soft" processor. The soft processor was chosen because it was flexible, low cost and faster to implement. This is because it behaved as a CPU (central processing unit). We could write the algorithm in C and downloaded into this soft processor. The authors planned to implement other inner coding system and compare their performance in the next phase. The design in fig. 4 shows that the soft processor was connected to a 1-megabit independent SRAM (Static Random-Access Memory) and a 1-megabit common bus memory.

In addition, the input/output ports of the processor were connected to a USB port, a soft terminal and a colour TFT-LCD (Thin film transistor-liquid crystal display) panel. The soft terminal and the TFT-LCD panel were for the input output display. All selected components were linked together in the schematic diagram illustrated in fig. 5 to allow the flow of signals between them and the processor. The left hand side of the processor was connected with the independent memory and the common bus memory. The right hand side of the processor was connected to the TFT-LCD panel, the clock generator and the test button.

III. METHOD

First, all parts of the coding system were modeled by either Matlab or Visual C++. These helped with the hardware test. Then the hardware was designed as described in section II. The C++ program was modified and simplified into a C program with only basic functions to meet the requirement of the processor in the board. This new program was input into the Nanoboard3000.

A. Simulations

The simulations were done using the combination of Matlab and C++ programming. Specifically, the fading channels were modeled with Matlab. The fading channels in consideration were the Rayleigh and Rician fading channel with or without the Doppler effect. In addition, the BCH inner encoder and the algebraic decoder were modeled with Matlab since these functions were readily available. The C++ programming was used for the outer encoder, the list Viterbi inner decoder and the outer VSD decoder since they were not standard encoder and decoders. The modulation scheme was the BPSK. The demodulation was done with a square-law detector.

The Doppler shift (f_d) can be computed from [16]

$$f_d = -f_c \frac{v \cdot \cos(\gamma)}{c_0} \qquad (1)$$

where f_c is the carrier frequency

v is the velocity that the receiver moves away from the transmitter

c_0 is the light speed and equal to 3×10^8 m/s

and γ is the angle between the transmitter and the receiver

The maximum Doppler shift occurs when γ is 0. In the simulation, we assumed maximum Doppler shift. The carrier frequency used was 2.1 GHz, which is the standard frequency for the third generation mobile communication system (3G). The velocities considered were 80 km/hr and 120 km/hr. From the computation, the 80 km/hr case caused the Doppler shift of 155.54 Hz. The 120 km/hr case caused the Doppler shift of 233.31 Hz.

B. FPGA Implementations

Fig. 6 shows the setup of the nanoboard in operation. The board is connected to the computer via a USB (Universal Serial Bus) port. For the experiment, the input file was in a flash drive USB that was connected to the board. The flash drive was used to test that the inner decoder worked properly as designed. To test the function of the board, we separated the test of the encoding and the decoding part. For the encoding part, the function was tested by several pilot data sequences. The encoded sequences were compared to the correct encoded sequences from the coding theory.

For the inner decoder part, the same received sequences obtained at the output of the square-law detector were input into the inner decoder with C++ program and the inner decoder inside the board. The results were then compared for hard decision and list-of-2 soft VA in various fading channel conditions.

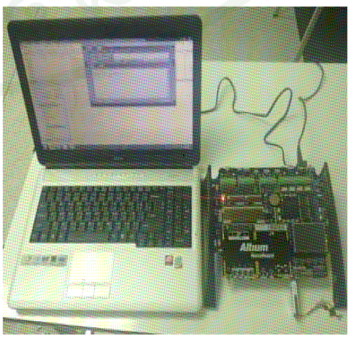


Figure 6. Actual hardware setup during tests.

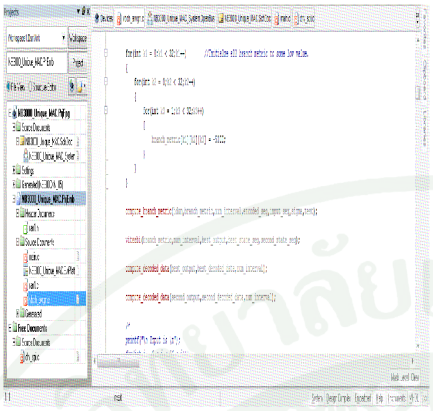


Figure 7. The code explorer window in Altium designer showing the Viterbi functions

In fig. 7, the inner decoder functions are shown in the Altium designer code explorer window. The detailed programming in each function was not demonstrated due to the lack of space. The concept of the list-of-2 soft VA had been explained in detail in [11].

IV. Results

The encoding functions of the system on the board were tested by inputting several pilot data sequences into the board and check whether the encoded sequences were properly encoded. Fig. 8 shows an example of the pilot data sequence and the encoded sequence on the soft terminal of the board. The encoding functions were straightforward and the encoded sequence matched with the expected results from the theory.

To ensure that the Viterbi program written in C++ worked correctly, its decoding error probability was compared with the algebraic decoding function in Matlab. The modulation scheme was BPSK and the demodulation was readily available with Matlab function for the hard decision case. The channel was modeled as a Rayleigh fading channel with AWGN. The result in fig. 9 confirms that they are approximately the same. However, the decoding error probability was high since the inner code can correct only 1 error in a 31-bit received sequence. Since the decoding error probability is in the small range of 0.1-1 in fig. 9, the y-axis was shown in linear scale.

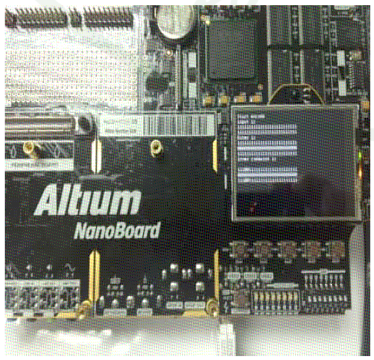


Figure 8. An example of the concatenated encoding result shown in the TFT-LCD display on the Nanoboard 3000 board.

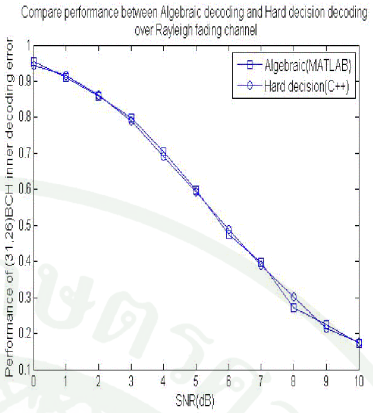


Figure 9. Decoding error probability of the algebraic and the hard VA decoders for the (31,26,3) BCH code in a Rayleigh channel

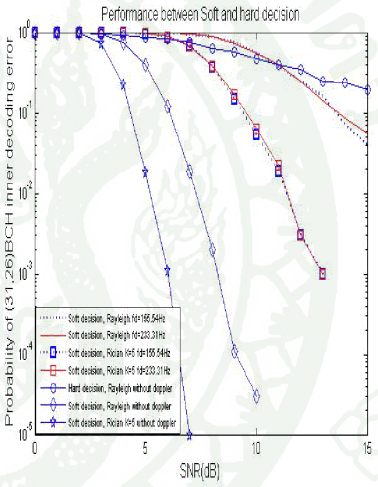


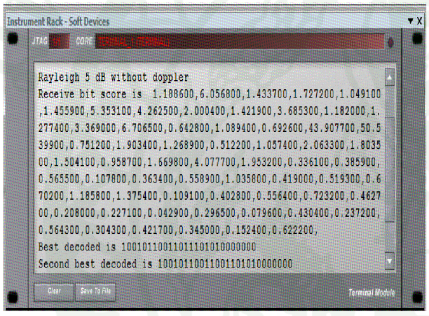
Figure 10. Decoding error probability of the soft and hard VA decoder for the (31, 26, 3) BCH code in fading channels

Recall that list-of-2 VA provides two possible decoded sequences, which can be called "the first choice" and "the second choice". Fig. 10 shows the comparison between the hard and soft decision VA. Only the decoding error probability of the first choice was shown since they were compared in several channel conditions. It is clear that the soft decision VA provided much better performance than the hard decision VA. Consequently, only the soft VA was investigated further in details. Fig. 10 shows the performance of soft VA in Rayleigh and Rician channels with and without Doppler effect. It is seen that the fading channels with no Doppler effect provided significantly better performance than the ones with Doppler effect.

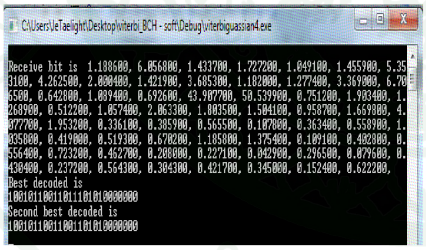
The Rician channel resulted in much lower decoding error than the Rayleigh channel both when there was a Doppler effect and when there was no Doppler effect. This is as expected since the Rayleigh channel is a special case of Rician channel when the Rician factor k = 0, which means that there is no line-of-sight path. For the channel with the Doppler effect and the carrier frequency of 2.1 GHz, the f_d of 155.54 Hz is for

the velocity of 80 km/hr case and the f_d of 233.31 Hz is for the velocity of 120 km/hr case. The performance for both Doppler frequency cases was approximately the same for both Rayleigh and Rician channels.

The results from the lab prototype were exactly the same as the results from the C++ simulation. Thus, fig. 10 represents the results from both hardware and software simulations since their graphs completely overlapped. To demonstrate the results from the lab prototype, fig. 11 and 12 show the soft terminal display in comparison with C++ output for several cases. In details, fig. 11 shows an example of a decoded sequence in two displays for the Rayleigh fading channel with AWGN that had SNR of 5 dB and no Doppler effect. The first one is the soft terminal display of the hardware. The second one is the C++ output display. It is clear that the results from the hardware and the software in fig. 11 were exactly the same. Notice that there were two possible decoded sequences in the display because the decoder is a list-of-2 decoder. Both decoded sequences will be the input to the outer decoder.



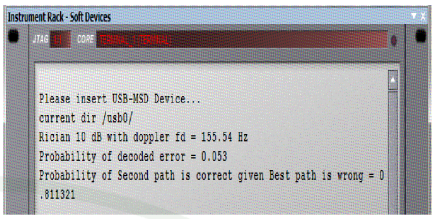
a) Soft terminal display of the prototype



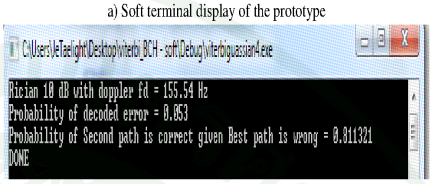
b) The C++ output display

Figure 11. Example of a decoding result in Rayleigh fading channel with SNR = 5 dB and no Doppler.

Fig. 12 shows a decoding error probability in the Rician fading channel with SNR = 10 dB and the Doppler frequency of f_d = 155.54 Hz. The number of trials used was 1,000 trials. Two probabilities were shown in the display. The probability that the first choice is wrong is considered the decoding error probability of the decoder. This is because the first choice is the main decoded sequence. If the inner decoder is used by itself, the decoding will fail when the first choice is wrong. However, when VSD with list inner symbol decision is used as the outer decoder, it can replace many wrong first choices with the correct second choices. This second choice can improve the performance of VSD. Therefore, the probability that the second choice is correct given that the first choice is wrong is of interest.



a) Soft terminal display of the prototype



b) The C++ output display

Figure 12. Example of a decoding error probability in Rician fading channel with SNR = 10 dB and f_d = 155.54 Hz.

Table 1. Resource utilization of the prototype

Logic Utilization	Used	Total	% Usage
Number of Slice Flip Flops	5,951	22,528	26%
Number of 4 input LUTs	9,743	22,528	43%
Logic Distribution			
Number of occupied Slices:	7,072	11,264	62%
Total Number of 4 input LUTs	10,273	22,528	45%
Number used as logic	9,473		
Number used as a route-thru	530		
Number used as 16x1 RAMs	14		
Number used for Dual Port RAMs	256		
Number of bonded IOBs	194	502	62
Number of BUFGMUXs	4	24	16
Number of DCMs	1	8	12
Number of MULT18X18SIOs	10	32	31
Number of RAMB16BWEs	20	32	62

The resource utilization is shown in table 1. This FPGA board operated at 50 MHz. Since there are a lot of logic units left in this board, we plan to implement the outer decoder on the same board.

V. Discussions

The FPGA implementation is a natural step for testing proposed system already tested in software simulation. The main objective is to show that the generalized concatenated coding system can be implemented. In this paper, only the encoders and the inner decoder were implemented. In the near future, other inner coding systems and the outer decoder will also be implemented.

The decoding error probabilities of the hardware and the software for this inner decoder were exactly the same when the same received sequences were used as the input. This confirmed that the lab prototype worked properly as designed. It also emphasized that the list Viterbi is not too complex to be implemented in hardware. The results also showed that the list-of-2 soft VA provided significantly better performance than the hard VA. Therefore, only list-of-2 soft VA will be considered in a generalized concatenated coding system.

The performance investigation was done for various channel conditions. It is clear that the Rician channels provided better performance than the Rayleigh channels. In addition, the Doppler effect increased the probability of decoding errors in both the Rician and Rayleigh channels. For this particular code, different Doppler frequencies resulted in almost the same decoding error probability for both Rayleigh and Rician fading channels. This was because each inner code word had a length of only 31 bits, which is relatively short. It was also a single-error-correcting BCH code, which has low correction capability, even when the soft decision decoding was employed. This inner code word was not powerful enough to correct the burst errors due to the fast fading caused by the Doppler effect. Consequently, both Doppler frequencies caused almost no difference in the inner decoder performance even though the higher Doppler frequency caused shorter fading periods and increased the number of crossover between the fade and non-fade. More powerful and longer inner codes may improve the performance of the inner decoder for fading channels with Doppler effect.

VI. Conclusions

This paper has presented the implementation of a concatenated encoding system and the list-of-2 soft VA in a Nanobaord3000 with the Xilinx Spartan 3AN FPGA chip. (XC3S1400AN-4FG676C). The performance investigation for the list-of-2 soft VA for the inner BCH code was done for fading channels with and without Doppler effects.

This list-of-2 soft VA is an important component of a generalized concatenated coding system that allows the inner and outer codes to be any combination of convolutional code(s) and block code(s). The actual algorithms and implementations depended on the selected codes, but the main principle and algorithm are the same. The test results showed that the prototype worked properly as designed. The soft processor was selected instead of FPGA implementation with hardware language like VHDL (VHSIC hardware description language) because we planned to test different inner coding systems to see which one will be most suitable for the proposed generalized concatenated coding system. This presented prototype will be used in the complete system in a near future.

Acknowledgment

The authors would like to thank Mr. Chanothai Chaiwan for the help with the C++ program of the list-of-2 VA for the selected BCH code.

References

[1] G.D. Forney, Jr., Concatenated Codes, MIT Press, Cambridge Mass., 1966.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in Proc. ICC'93, pp. 1064–1070, May 1993, Geneva, Switzerland.

[3] S. Benedetto and G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes," IEEE Trans on Inf. Theory, vol. 42, issue: 2, pp. 409 – 428, 1996.

[4] I.S. Reed and G. Solomon, "Polynomial codes over certain finite fields," Journal of Society for Industrial and Applied Mathematics, vol. 8, pp. 300-304, 1960

[5] CCSDS 130.1-G-1, TM synchronization and channel coding-summary of concept and rationales, green book, 89 pages, June 2006.

[6] Lin and D. J. Costello Jr., Error Control Coding, 2nd edition, Pearson Education, Upper Saddle River, NJ, 2004.

[7] U. Tuntoolavest and J. Thonchai, "VHDL design of a convolutional concatenated encoding system," in Proc. of ICICTES 2011, pp.140-144, Jan 27-29, 2011, Pattaya, Chonburi, Thailand.

[8] U. Tuntoolavest, Vector Symbol Decoding for Wireless Fading Channels, VDM publishing, 184 pages, 2009.

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

[10] I. Chana, H. Allouch and M. Belkasmi, "An efficient new soft-decision decoding algorithm for binary cyclic codes," in Proc. of ICMCS, pp. 823-828, April 2011, Ouarzazate, Morocco.

[11] U. Tuntoolavest, V. Suktalordcheep and C. Chaiwan, "List-of-2 soft decision Viterbi inner decoder for a generalized concatenated coding system," in Proc. of ECTI-CON 2011, pp. 264-267, May 17-19, 2011, Khon Kaen, Thailand.

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

[13] U. Tuntoolavest and C. Chaiwan, "On adjusting vector symbol decoding for many different nonbinary convolutional codes," Kasetsart Journal (Natural Science), vol. 46, no. 2, pp. 305-317, March-April 2012.

[14] U. Tuntoolavest and J.J. Metzner, "Vector symbol decoding with list symbol decisions and outer convolutional codes for reliable communications," Integr comput-aided engineer Journal, vol. 9, no. 2, 2002, pp. 101-116, IOS Press.

[15] 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), supplement issue, vol. 41, no 5, p 364-370, January-December 2007.

[16] U. Tuntoolavest and P. Noradee, "Lab prototype of a list-of-2 Viterbi decoder: a diversity inner decoder for the outer vector symbol decoder," in Proc. of ECTI-CON 2010, pp. 973-977, May 19-21, 2010, Chiang Mai, Thailand

[17] A.F. Molisch, Wireless communications, John Wiley & sons, 622 pages, 2005.

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นาย จตุพล ท่นไชย
วัน เดือน ปี ที่เกิด	วันที่ 10 พฤษภาคม 2531
สถานที่เกิด	อำเภอเมือง จังหวัดเชียงราย
ประวัติการศึกษา	วศ.บ.(วิศวกรรมไฟฟ้า) มหาลัยวิทยาลัยเกษตรศาสตร์
ตำแหน่งหน้าที่การงานปัจจุบัน	-
สถานที่ทำงานปัจจุบัน	-
ผลงานดีเด่นและรางวัลทางวิชาการ	-
ทุนการศึกษาที่ได้รับ	-