วิทยานิพนธ์

การวิจัยและพัฒนาส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ของระบบสื่อสารในสถานีภาคพื้นดินสำหรับดาวเทียมอเนกประสงค์ขนาดเล็ก

Research and Development of Digital Baseband Processing for Ground Station Communication Module for SMMS

นายสัณห์ อุทยารัตน์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

พ.ศ. 2549

ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)

ปริญญา

วิศวกรรมไฟฟ้า	วิศวกรรมไฟฟ้า
สาขา	**ภาควิชา**

เรื่อง การวิจัยและพัฒนาส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ของระบบสื่อสารในสถานีภาคพื้นดินสำหรับดาวเทียมอเนกประสงค์ขนาดเล็ก

Research and Development of Digital Baseband Processing for Ground Station Communication Module for SMMS

นามผู้วิจัย นายสัณห์ อุทยารัตน์

ได้พิจารณาเห็นชอบโดย

ประธานกรรมการ .....
(รองศาสตราจารย์มงคล รักษาพัชรวงศ์, Ph.D.)

กรรมการ .....
(ผู้ช่วยศาสตราจารย์วัชรี วีรคเชนทร์, M.S.)

กรรมการ .....
(ผู้ช่วยศาสตราจารย์ศรีจิตรา มหาประดุณชัย, Ph.D.)

หัวหน้าภาควิชา .....
(อาจารย์ชูเกียรติ การะเกตุ, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....
(รองศาสตราจารย์วินัย อาจคงหาญ, M.A.)

คณบดีบัณฑิตวิทยาลัย

วันที่ 30 **เดือน** มีนาคม **พ.ศ.** 2549

วิทยานิพนธ์

เรื่อง

การวิจัยและพัฒนาส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ของระบบสื่อสารในสถานีภาคพื้นดินสำหรับดาวเทียมอเนกประสงค์ขนาดเล็ก

Research and Development of Digital Baseband Processing for Ground Station Communication Module for SMMS

โดย

นายสัณห์ อุทยารัตน์

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)
พ.ศ. 2549

ISBN 974-16-1406-3

สัณห์ อุทยารัตน์ 2549: การวิจัยและพัฒนาส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ของระบบสื่อสารในสถานีภาคพื้นดินสำหรับดาวเทียมอเนกประสงค์ขนาดเล็ก ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า) สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า ประธานกรรมการที่ปรึกษา: รองศาสตราจารย์ มงคล รักษาพัชรวงศ์, Ph.D. 103 หน้า
ISBN 974-16-1406-3

งานวิจัยและพัฒนาระบบสื่อสารในสถานีภาคพื้นดินสำหรับดาวเทียมอเนกประสงค์ขนาดเล็กนี้ เป็นการพัฒนาส่วนการทำงานในระดับชั้นฟิสิคัลสำหรับใช้ในการติดต่อสื่อสารระหว่างสถานีภาคพื้นดิน ที่ประกอบไปด้วยส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ ส่วนการเข้าจังหวะสัญญาณ ที่พัฒนาขึ้นด้วยเทคโนโลยี FPGA/VHDL ส่วนซอฟต์แวร์ประยุกต์ที่เป็นส่วนที่ติดต่อกับผู้ใช้งาน ถูกพัฒนาด้วยภาษา Visual C และสุดท้ายจะเป็นส่วนการติดต่อกับภาคสัญญาณวิทยุที่จะใช้อุปกรณ์มาตรฐานที่มีอยู่ในท้องตลาด นำมาประกอบกัน โดยผลที่ได้จากการทดสอบในส่วนสัญญาณดิจิตอลเบสแบนด์ จะสามารถรับส่งข้อมูลได้สูงถึง 1.5 Mbps และมีประสิทธิภาพในการเข้ารหัส ถอดรหัสสสัญญาณสูงถึง 6.5 dB และในส่วนการเข้าจังหวะเวลาสามารถแก้ไขเฟสตกค้าง เวลาที่ไม่ตรงกัน รวมถึงความถี่เลื่อนที่สามารถแก้ไขได้สูงถึง 4.5MHz ในขณะที่ดาวเทียมอเนกประสงค์นี้เกิดปัญหาความถี่เลื่อนเพียง 1.5MHz เท่านั้น และระบบต้นแบบที่พัฒนาขึ้นมายังสามารถติดต่อกับภาคสัญญาณวิทยุได้ที่ 65MHz สำหรับภาคส่งสัญญาณ และที่ 75MHz สำหรับภาครับสัญญาณ

ลายมือชื่อนิสิต ลายมือชื่อประธานกรรมการ 27 / 3 / 49

Sunt Uttayarath 2006: Research and Development of Digital Baseband Processing for Ground Station Communication Module of SMMS. Master of Engineering (Electrical Engineering), Major Field: Electrical Engineering, Department of Electrical Engineering. Thesis Advisor: Associate Professor Mongkol Raksapatcharawong, Ph.D. 103 pages. ISBN 974-16-1406-3

This research develops ground station communication module for small multi-mission satellite. It is to develop subsystems in physical layer for communicating ground station. And the subsystems consist of 4 parts: First is digital baseband processing module and second is recovery module. Both modules are developed by FGPA/VHDL technology. Next is application software that developed by Visual C. And the last is interfacing RF module that use COST (Commercial of the shelf) component. We test this baseband processing to verify the correctness of encoder and decoder, and to estimate the computation complexity by simulating the processing from application software and found this module can support data bit rate 1.3 Mbps and has coding gain 6.5 dB (compare with no coding). While recovery module is not only correct phase and timing offset but also doppler frequency that fix up to 4.5MHz. It is sufficient for SMMS that just has doppler frequency 1.5MHz. Finally, this communication module can interfacing RF Module at 65MHz in transmits module and 75MHz in receive module.

Sunt Uttayarath — Student's signature

Mongkol Raksapatcharawong — Thesis Advisor's signature

29 / 03 / 06

กิตติกรรมประกาศ

ข้าพเจ้าขอกราบขอบพระคุณ รองศาสตราจารย์ ดร. มงคล รักษาพัชรวงศ์ ประธานกรรมการที่ปรึกษา ผู้ช่วยศาสตราจารย์ วัชรี วีรคเชนทร์ และ ผู้ช่วยศาสตราจารย์ ดร. ศรีจิตรา มหาประคุณชัย ที่ได้ช่วยเหลือในการวางแผนงานวิจัยในวิทยานิพนธ์ฉบับนี้ ตลอดจนการให้คำปรึกษาแนะนำและตรวจแก้ไขข้อบกพร่องในวิทยานิพนธ์ให้สำเร็จลุล่วงไปด้วยดี

ข้าพเจ้าขอกราบขอบพระคุณ คุณพ่อบุญจิบ อุทยารัตน์ ที่สนับสนุนและให้กำลังใจ ขอบคุณเพื่อนๆในห้อง SCORPion Lab ทุกคนโดยเฉพาะ นาย เอกพล หิรัณยเอกภาพ รวมทั้งพี่ๆ เพื่อนๆ น้องๆทุกคน ที่มีส่วนช่วยแนะนำ แก้ไข ให้กำลังใจในการทำวิทยานิพนธ์ฉบับนี้จนสำเร็จลุล่วงไปด้วยดี

สุดท้ายนี้ข้าพเจ้าขอขอบพระคุณกระทรวงเทคโนโลยีสารสนเทศและการสื่อสาร ที่ให้การสนับสนุนการทำวิทยานิพนธ์ฉบับนี้ ผ่านโครงการวิจัยและพัฒนาต้นแบบระบบสื่อสารสำหรับสถานีภาคพื้นดินของดาวเทียมอเนกประสงค์ขนาดเล็ก และหากวิทยานิพนธ์ฉบับนี้มีข้อบกพร่องประการใด ข้าพเจ้ายินดีรับข้อเสนอแนะและขออภัยมา ณ ที่นี้ด้วย

สัณห์ อุทยารัตน์

มีนาคม 2549

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

การวิจัยและพัฒนาส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ของระบบสื่อสารในสถานีภาคพื้นดินสำหรับดาวเทียมอเนกประสงค์ขนาดเล็ก

Research and Development of Digital Baseband Processing for Ground Station Communication Module for SMMS

คำนำ

ปัจจุบันกิจการอวกาศมีความสำคัญต่อการพัฒนาเศรษฐกิจเป็นอย่างมาก หลายประเทศได้มีการพัฒนาที่รุดหน้าในการสร้างและพัฒนาวิทยาศาสตร์ทางด้านอวกาศและการใช้ประโยชน์จากอวกาศกันมากมาย ไม่ว่าจะเป็น ด้านการสื่อสาร ด้านอุตุนิยมวิทยา ด้านการสำรวจทรัพยากรและสิ่งแวดล้อม ด้านการศึกษา และด้านการสาธารณสุข เป็นต้น ซึ่งการพัฒนาดังกล่าวต้องอาศัยหลายๆ ปัจจัยเข้าไว้ด้วยกันไม่ว่าจะเป็น เงินทุน ทรัพยากรบุคคล และวัตถุดิบในการผลิต ซึ่งประเทศไทยเองนั้น ก็กล่าวได้ว่ามีความพร้อมการสร้างดาวเทียมเพื่อให้เป็นโครงสร้างพื้นฐานทางเทคโนโลยี อีกทั้งยังได้รับการสนับสนุนจากเจ้าของเทคโนโลยีที่มีประสบการณ์อย่างมากอย่างประเทศจีน ดังนั้นประเทศไทยได้เข้าร่วมดำเนินโครงการร่วมสร้างดาวเทียมอเนกประสงค์ขนาดเล็ก (Small Multi-Mission Satellite- SMMS) ในย่าน Ka-band ร่วมกับประเทศจีน และเปลี่ยนบทบาทของประเทศจาก “ผู้ใช้เทคโนโลยี” มาเป็น “ผู้พัฒนาเทคโนโลยี” อีกทั้งยังยกระดับประเทศในแง่ของการวิจัยและพัฒนานวัตกรรมใหม่ๆ เชิงวิศวกรรมศาสตร์อีกด้วย

โดยในงานวิจัยนี้ จะมุ่งเน้นที่จะพัฒนาส่วนการติดต่อสื่อสารบนสถานีภาคพื้นดินสำหรับอเนกประสงค์ขนาดเล็ก ประกอบด้วนการประมวลผลสัญญาณได้แก่ เฟรมข้อมูลและการเข้ารหัสถอดรหัส เพื่อเพิ่มความคงทนต่อสัญญาณรบกวน ส่วนการเข้าจังหวะสัญญาณที่เป็นหัวใจในภาครับสัญญาณ ส่วนการติดต่อกับภาคสัญญาณวิทยุที่ความถี่กลาง และสุดท้ายจะเป็นซอฟต์แวร์ประยุกต์ที่ใช้สำหรับส่งข้อมูลต่างๆ ไม่ว่าจะเป็นเสียง แฟ้มข้อมูล หรือวีดีโอ ซึ่งในสองส่วนแรกจะถูกพัฒนาด้วยเทคโนโลยี FPGA/VHDL ส่วนซอฟต์แวร์ประยุกต์จะถูกพัฒนาด้วยภาษา Visual C++ ทำงานอยู่บนคอมพิวเตอร์ และส่วนการติดต่อกับภาคสัญญาณวิทยุจะใช้อุปกรณ์มาตราฐานที่มีขายในท้องตลาด โดยมีการกำหนดค่าพารามิเตอร์ที่จำเป็นด้วยภาษา VHDL

วัตถุประสงค์

1. เพื่อพัฒนาต้นแบบของส่วนติดต่อสื่อสารข้อมูลที่ใช้ในการรับส่งข้อมูลผ่านดาวเทียมในย่าน Ka-band

2. ทำให้สามารถประยุกต์ระบบที่พัฒนาขึ้นมาเข้าไปใช้กับระบบสื่อสารอื่นๆ ได้

3. พัฒนาประเทศให้ทัดเทียมกับประเทศอื่นๆ ได้ในเทคโนโลยีทางด้านดาวเทียม

การตรวจเอกสาร

1. บทนำ

จากโครงการความร่วมมือระหว่างประเทศไทยและสาธารณรัฐประชาชนจีน ที่จะร่วมกันพัฒนาดาวเทียมอเนกประสงค์ขนาดเล็กในย่านความถี่ Ka ซึ่งเป็นย่านที่รองรับอัตราในการส่งข้อมูลได้มากถึง 600 Mbps ทำให้มีประสิทธิภาพที่จะสามารถที่จะพัฒนาให้ใช้เป็นเครือข่ายอินเตอร์เน็ต หรือรองรับโครงข่ายโทรศัพท์เคลื่อนที่ในยุคที่ 3 ได้ โดยประเทศไทยได้รับผิดชอบในส่วนการพัฒนาอุปกรณ์สำหรับการสื่อสารในย่าน Ka ซึ่งสื่อสารที่ความถี่ขาขึ้น 29.22 MHz และขาลง 18.72 MHz และจุดประสงค์หลักในส่วนของประเทศไทยก็คือ การทดลองทางวิทยาศาสตร์ต่างๆ เช่น การศึกษาสภาพฝนที่มีผลต่อการส่งข้อมูลผ่านดาวเทียม หรือ การทดสอบการเข้ารหัสต่างๆ ในการสื่อสารผ่านดาวเทียม เป็นต้น

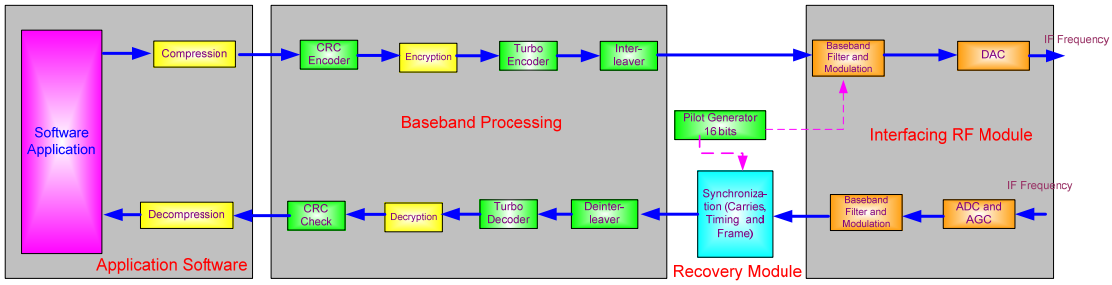
ในการพัฒนานี้จะเป็นการพัฒนาระบบสื่อสารของสถานีภาคพื้นดินสำหรับดาวเทียมอเนกประสงค์ขนาดเล็ก ซึ่งประกอบไปด้วย 4 ส่วนหลักๆ คือ ส่วนที่หนึ่งคือส่วนซอฟต์แวร์ประยุกต์ (Application Software) ที่ใช้สำหรับการติดต่อกับผู้ใช้งานต่างๆ ไม่ว่าจะเป็นการรับส่งไฟล์ข้อมูล เสียง หรือวีดีโอ รวมถึงยังสามารถตรวจวัดประสิทธิภาพของตัวเข้ารหัสหรือถอดรหัสสัญญาณได้ ส่วนที่สองคือส่วนการประมวลผลสัญญาณดิจิตอลเบสแบนด์ (Baseband Processing) ที่เป็นการพัฒนาตัวเข้ารหัสสัญญาณ ตัวถอดรหัสสัญญาณ ช่วยเพิ่มความคงทนของข้อมูลต่อสัญญาณรบกวนที่เกิดขึ้นจากการส่งผ่านข้อมูลในช่องสัญญาณสื่อสารไร้สาย ส่วนที่สามเป็นส่วนที่เป็นหัวใจในภาครับสัญญาณนั้นก็คือ ส่วนการเข้าจังหวะสัญญาณ (Synchronization/Recovery Module) ซึ่งจะเป็นการแก้ปัญหาในเรื่องการไม่เข้าจังหวะระหว่างภาครับกับภาคส่งสัญญาณไม่ว่าจะเป็นในเรื่องเฟส หรือเวลาที่ไม่เข้ากัน รวมถึงการแก้ปัญหาที่เกิดขึ้นจากวงโคจรที่ต่ำของดาวเทียมนั้นก็คือ ปัญหาความถี่เลื่อน (Doppler Frequency) และส่วนสุดท้ายจะเป็นการเชื่อมต่อกับส่วนสัญญาณวิทยุ (RF Module) ที่ความถี่กลาง 70 MHz

และในการพัฒนานี้ยังได้มีการเพิ่มส่วนการพัฒนาการบีบอัดข้อมูล (Compression Module) เพื่อช่วยให้สามารถรับส่งข้อมูลได้มากขึ้น และส่วนการเข้ารหัสลับข้อมูล (Data Encryption) เพื่อให้ข้อมูลมีความปลอดภัยมากยิ่งขึ้นสำหรับการรับส่งข้อมูลผ่านดาวเทียม โดยในส่วนแรกจะไป

รวมอยู่กับการพัฒนาซอฟต์แวร์ประยุกต์ และส่วนที่สองจะไปแทรกอยู่ในส่วนการพัฒนาส่วนประมวลผลดิจิตอลเบสแบนด์

2. การพัฒนาระบบการสื่อสารข้อมูลสำหรับสถานีภาคพื้นดิน

ในการพัฒนาส่วนการสื่อสารจะประกอบไปด้วย 4 ส่วนหลักๆ คือ ซอฟต์แวร์ประยุกต์เป็นส่วนที่ติดต่อกับผู้ใช้งาน โดยจะติดตั้งบนเครื่องคอมพิวเตอร์ หรือ Notebook ส่วนที่สองคือส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ ส่วนต่อไปคือส่วนการเข้าจังหวะสัญญาณ และส่วนสุดท้ายจะเป็นการเชื่อมต่อกับส่วนสัญญาณภาควิทยุ ตามภาพที่ 1



ภาพที่ 1 โครงสร้างระบบสื่อสารข้อมูลสำหรับสถานีภาคพื้นดิน

โดยในส่วนการพัฒนาส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์และส่วนการเข้าจังหวะสัญญาณ ทางผู้วิจัยได้คำนึงถึงความรวดเร็วในการพัฒนาต้นแบบ และตัวสถานีภาคพื้นดินไม่มุ่งเน้นเกี่ยวกับการประหยัดพลังงาน อีกทั้งต้องการความยืดหยุ่นสูงเพื่อสามารถจัดสรรทรัพยากรของระบบได้อย่างมีประสิทธิภาพ ซึ่งชิป FPGA (Field Programmable Gate Array) มีคุณสมบัติดังกล่าว และยังสามารถพัฒนาได้อย่างรวดเร็ว วงจรประมวลผลสัญญาณที่ซับซ้อนจะถูกสร้างขึ้นด้วยฮาร์ดแวร์ ขณะที่ส่วนคำนวณที่ซับซ้อนสามารถสร้างเป็น CPU ที่สามารถพัฒนาด้วยภาษา C ซึ่งส่วนประมวลผลนี้จะถูกพัฒนาขึ้นบน FPGA เพียงชิปเดียว และมีข้อสังเกตว่าในการใช้ชิป FPGA ไม่จำเป็นต้องผูกติดกับบริษัทใดบริษัทหนึ่ง เนื่องจากภาษา VHDL ค่อนข้างเป็นมาตรฐานและสามารถนำไปใช้กับชิป FPGA ของบริษัทอื่นๆ ได้อีกโดยไม่ต้องปรับเปลี่ยนมากเท่าไรนัก

และข้อดีอีกอย่างหนึ่งของการใช้งานชิป FPGA ก็คือการที่สามารถแก้ไขหรือเปลี่ยนแปลงฟังก์ชั่นการทำงานได้โดยง่าย ด้วยการ Reconfiguration จากซอฟต์แวร์ ทำให้สามารถที่จะทดสอบและทดลองการเข้ารหัสใหม่ๆ หรือการจัดการทรัพยากร รวมถึงการแก้ไขระบบในกรณีที่มีปัญหาได้ ซึ่งก็ถือว่าเป็นข้อได้เปรียบมากเมื่อเทียบกับการใช้งานแบบอื่นๆ เช่น ASIC หรือ DSP เป็นต้น

และในส่วนการเชื่อมต่อกับภาคสัญญาณวิทยุ เพื่อความรวดเร็วและสะดวกในการพัฒนาจะใช้อุปกรณ์มาตรฐานจากบริษัทชั้นนำ หรือจะเรียกได้อีกอย่างว่า COST (Commercial off the Shelf) ซึ่งก็ได้เลือกที่จะใช้อุปกรณ์จาก Analog Device ซึ่งจะมีด้วยกัน 4 ตัวคือ Up-converter, Down-converter, Analog to Digital Converter, และ Automatic Gain Control ที่ใช้สำหรับการติดต่อกับส่วนสัญญาณวิทยุที่ความถี่กลาง 70 MHz

2.1 ส่วนซอฟต์แวร์ประยุกต์ (Application Software Module)

2.1.1 ซอฟต์แวร์ประยุกต์

ในการพัฒนาซอฟต์แวร์ประยุกต์นี้จะพัฒนาด้วยภาษา C/C++ และทำงานบนคอมพิวเตอร์แบบพกพา (Notebook) โดยจะทำหน้าที่รับส่งข้อมูล ไม่ว่าจะเป็นเสียง ภาพ ไฟล์ข้อมูล หรือวีดีโอ จะรับส่งข้อมูลและพารามิเตอร์ต่างๆ เช่น พารามิเตอร์ที่ใช้สำหรับจำลองการทำงานของช่องสัญญาณ เป็นต้น ผ่าน USB 2.0 ไปยังส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ ซึ่งสามารถรองรับการทำงานได้สูงถึง 480 Mbps โดยในกรณีของดาวเทียมดวงนี้ตั้งเป้าหมายไว้เพียง 2 Mbps เท่านั้น ถือว่าสามารถรองรับการทำงานได้ตามที่ต้องการ และในอนาคตยังสามารถที่จะขยายอัตราในการรองรับข้อมูลให้สูงขึ้นได้ ซึ่งในย่านความถี่ของ Ka-band นี้สามารถรองรับได้สูงถึง 600 Mbps

อีกทั้งในการพัฒนาซอฟต์แวร์ประยุกต์นี้จะต้องคำนึงถึงความง่ายของผู้ใช้งานอีกด้วย ดังนั้นหน้าจอที่ใช้สำหรับการติดต่อจะต้องเห็นแล้วเข้าใจได้ในทันทีว่าควรจะใช้งานยังไง รวมถึงการทดสอบประสิทธิภาพของตัวเข้ารหัส ถอดรหัสสัญญาณที่จะแสดงให้เห็นในรูปแบบกราฟระหว่างค่า SNR กับ E_b / N_0 เพื่อให้สามารถตรวจสอบได้ว่าค่า Coding Gain เมื่อเปรียบเทียบ

กับการไม่เข้ารหัสแล้วจะมีขนาดเท่าไร และยังแสดงผลเป็นค่าเพื่อใช้ในการเปรียบเทียบอีกต่างหาก ตามภาพที่ 2

ภาพที่ 2 ตัวอย่างการรับส่งไฟล์ข้อมูล (ซ้าย) และการวัดประสิทธิภาพของตัวเข้ารหัส (ขวา)

2.1.2 ส่วนการบีบอัดข้อมูล (Data Compression)

การบีบอัดข้อมูลจะช่วยทำให้ข้อมูลที่จะถูกส่งออกไปมีขนาดเล็กลง และเป็นการเพิ่มอัตราในการส่งข้อมูลให้สูงขึ้น โดยการจัดการในลักษณะนี้ถูกเรียกว่า Source Coding มีด้วยกัน 2 ลักษณะคือ Lossy Compression และ Lossless Compression โดยแบบแรกเป็นพวกบีบอัดแล้วข้อมูลเสียหายก็คือเมื่อทำการ Decompression แล้วไม่สามารถนำข้อมูลกลับมาได้ เช่น JPEG, MPEG เป็นต้น และพวกที่สองก็จะเป็นพวกบีบอัดแล้วข้อมูลไม่หายไป หมายความว่าสามารถนำข้อมูลที่บีบอัดกลับมาเป็นก่อนไม่บีบอัดได้ การทำงานในลักษณะนี้นิยมใช้สำหรับข้อมูลที่มีความสำคัญ แต่ก็เปลืองทรัพยากรที่ใช้มากกว่าแบบแรก โดยเทคนิกพวกนี้จะได้แก่ Run-length, Huffman, Delta และ LZW เป็นต้น

ในโครงการนี้ทางคณะผู้วิจัยได้เลือกที่จะใช้ Lossless Compression เนื่องจากข้อมูลที่ใช้ส่งออกไปเป็นข้อมูลที่ต้องการความสมบูรณ์ในตัวเอง เช่นข้อมูลแหล่งทรัพยากรธรณี ข้อมูลภาพถ่ายต่างๆ เป็นต้น ซึ่งยอมรับไม่ได้กับการบีบอัดที่บีบไปแล้วข้อมูลสูญเสียไป โดยในนี้ก็มีอัลกอลิทึ่มอยู่มากมาย แต่ที่ใช้แพร่หลายก็จะเป็น Run-Length, Huffman และ LZW ดังนั้นจึงได้ทำการเปรียบเทียบ 3 แบบดังกล่าวเป็นดังต่อไปนี้

Run-Length จะเป็นการบีบอัดข้อมูลที่ง่ายที่สุด ซึ่งนับความยาวของข้อมูลที่เหมือนกัน และติดกันยกตัวอย่างดังนี้คือ ข้อมูล “000001111001101” จะเป็น “501420211011” โดยเทคนิกนี้เหมาะสมกับข้อมูลที่มีความซ้ำๆ กันมาก เช่นพวก Icon หรือ ลายเส้นแบบง่ายๆ เป็นต้น

แต่เทคนิกนี้บางครั้งเมื่อบีบอัดแล้ว อาจจะได้ข้อมูลที่มีขนาดใหญ่กว่าข้อมูลจริงก็ได้ ทำให้เกิดความไม่แน่นอนในการบีบอัดซักเท่าไร

Huffman code เทคนิกนี้เป็นเทคนิกที่ได้รับความนิยมใช้อย่างแพร่หลาย เนื่องจากมีประสิทธิภาพในการบีบอัดที่สูง ข้อมูลจะถูกแตกออกเป็นในลักษณะของกราฟต้นไม้ โดยแต่ละกิ่งจะมี code ที่เรียกว่า Pre-fix code อยู่ และเหมาะกับข้อมูลที่มีค่าน้ำหนัก เช่น ข้อความหรือประโยค โดยจะมีการให้น้ำหนักกับตัวอักษร (สระในภาษาอังกฤษจะมีโอกาสอยู่ในประโยคได้มากกว่าตัวอักษรตัวอื่นๆ ดังนั้นจะมีค่าน้ำหนักสูงกว่า) เป็นต้น และใช้วิธีการ entropy encoding มาใช้ในการคำนวณ สุดท้ายขนาดของข้อมูลจะมีขนาดที่เล็กลงตามค่าน้ำหนักที่จะคำนวณออกมา แต่ข้อเสียใหญ่ที่เกิดจากวิธีนี้คือการที่จะต้องรู้ค่าน้ำหนักเพื่อใช้ในการคำนวณ ซึ่งในทางปฏิบัติบางครั้งจะไม่มีทางรู้ค่าเหล่านี้ได้เลย

LZW (Lempel-Ziv) code เป็นเทคนิกที่ใช้กันแพร่หลายที่สุด โดยซอฟต์แวร์ตามท้องตลาดก็ได้นำวิธีนี้ไปใช้ เช่น WinZip เป็นต้น เนื่องด้วยเทคนิกนี้จะใช้วิธีสร้างดิกชั่นนารี่ขึ้นมาเก็บข้อมูล ถ้าเป็นข้อมูลใหม่ก็จะเก็บเอาไว้ในฐานข้อมูลไปเรื่อยๆ และแทนที่ด้วยรหัสแบบไบนารี่ ซึ่งถ้าเจอข้อมูลที่ซ้ำก็จะแทนที่ได้เลย โดยวิธีนี้สิ่งที่ต้องคำนึงก็คือขนาดของดิกชั่นนารี่ว่าจะมีขนาดเท่าไร และไปสัมพันธ์กับวิธีการค้นหาข้อมูลในฐานข้อมูลนั้นก็คือความเร็วในการสืบค้นข้อมูลนั้นเอง โดยปรกติขนาดของดิกชั่นนารี่จะใช้ขนาด 12 บิตข้อมูลหรือสามารถสืบค้นข้อมูลได้ 4,096 ตัว (สามารถที่จะปรับเปลี่ยนขนาดได้ไม่จำเป็นต้องยึดกับค่านี้เท่านั้น) ดังนั้นข้อเสียหลักก็คือขนาดของไฟล์ที่ใช้ในการบีบอัดจะต้องมีขนาดใหญ่พอสมควร ถ้าบีบอัดข้อมูลไม่มากก็จะทำให้ขนาดไฟล์ลดลงได้ไม่มาก ตามที่จะเห็นได้ในซอฟต์แวร์ WinZip เมื่อบีบอัดไฟล์ที่มีขนาดเล็กก็จะได้ไม่มากเท่าไร

และที่ได้กล่าวไปเป็นแค่ข้อดีและข้อเสียของการบีบอัดในแต่ละแบบที่นิยมใช้ในเชิงพาณิชย์ และยังมีการบีบอัดแบบอื่นๆ ที่ใช้กันอยู่ไม่ว่าจะเป็นมาตรฐาน CCSDS (เป็นมาตรฐานที่ใช้กับระบบการสื่อสารดาวเทียม) หรืองานวิจัยอื่นๆ ที่พัฒนาขึ้นมาใช้เฉพาะงานเหล่านั้น ซึ่งผู้วิจัยสามารถนำมาประยุกต์ใช้สำหรับการบีบอัดข้อมูลในระบบการสื่อสารสำหรับดาวเทียมอเนกประสงค์ขนาดเล็กได้

แต่ที่ได้กล่าวมาทั้งหมดการเลือกใช้การบีบอัดข้อมูลในเบื้องต้นขึ้นอยู่กับว่าระยะเวลาในการพัฒนาต้นแบบและงบประมาณ โดยจะมีอยู่ 2 แนวทาง โดยแนวทางแรกคือ จะ

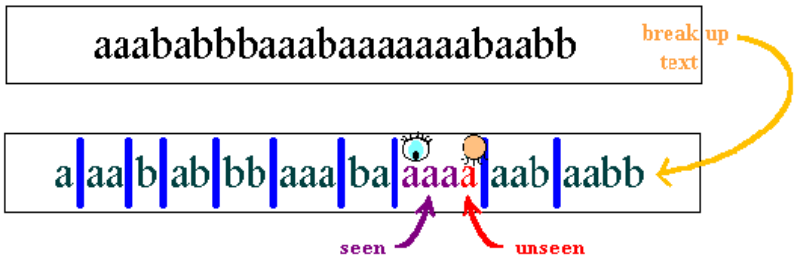
สามารถหาเทคนิกที่ไม่จำเป็นต้องเสียเงินสำหรับจ่ายค่าลิขสิทธิ์ และเป็นไลบารี่อยู่แล้ว เพื่อความสะดวกและรวดเร็วในการพัฒนาต้นแบบ หรืออีกแนวทางหนึ่งก็คือทำการพัฒนาเทคนิกการบีบอัดขึ้นมาเองโดยประยุกต์จากองค์ความรู้ที่มีอยู่ แต่ในกรณีนี้ก็เสียเวลาในการพัฒนาต้นแบบไปด้วย

ดังนั้นในขั้นแรกก็คือในขณะนี้จะดำเนินการไปในแนวทางแรกโดยหาไลบารี่ที่มีอยู่แล้วในท้องตลาด และไม่เสียเงินสำหรับค่าลิขสิทธิ์ และใช้ได้กับซอฟต์แวร์ประยุกต์ที่ทำงานด้วยภาษา C และถ้าไม่มีที่ไม่ต้องเสียเงินก็จะดำเนินการพัฒนาให้เป็นไปตามแนวทางที่สอง โดยคาดว่าจะเลือกใช้เทคนิก LZW มาใช้เนื่องจากมีความแน่นอนในการบีบอัด และไม่จำเป็นต้องรู้ค่าน้ำหนักของข้อมูลเพื่อใช้ตัดสินใจ เพียงแค่กำหนดค่าพารามิเตอร์ที่เหมาะสมกับขนาดของข้อมูล (การกำหนดขนาดของดิกชั่นนารี่) และพัฒนาเทคนิกในการสืบค้นข้อมูลให้มีความรวดเร็วที่สุดเท่าที่จะเป็นไปได้ เพื่อแสดงให้เห็นถึงเทคนิกการบีบอัดแบบ Lempel-Ziv จะใช้ตัวหนังสือสองตัวคือ a และ b สร้างรูปแบบข้อมูลดังตัวอย่างในภาพที่ 3

aaababbbaaabaaaaaaabaabb

ภาพที่ 3 รูปแบบข้อมูลที่สมมติขึ้นมา

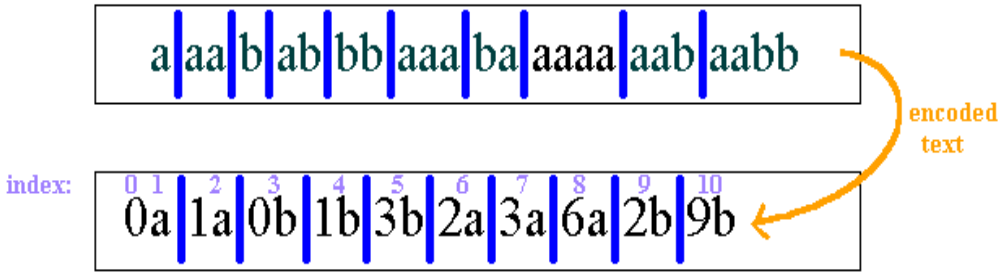
กฎในการทำคือ พยายามแยกรูปแบบข้อมูลเป็นกลุ่มตัวอักษรที่สั้นที่สุดที่ไม่เคยเห็นมาก่อน ตามกฎที่กล่าวข้างต้น จะเห็นว่าอักษรกลุ่มแรกคือ a กลุ่มที่สองต้องเป็น aa ถ้าทำอย่างนี้ไปเลื่อยๆจะได้ข้อมูลดังแสดงในภาพที่ 4 ก่อนที่จะทำการบีบอัด กลุ่มของตัวอักษรในขั้นตอนการแบ่งต้องมีการใส่ index กลุ่มของอักษร จาก 1 ถึง n ดังภาพที่ 5



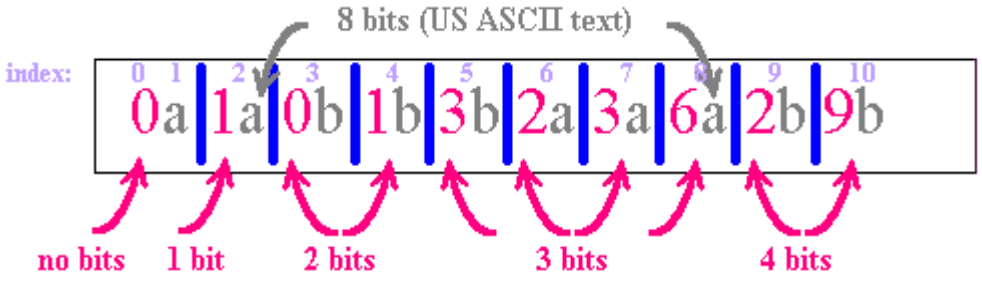
ภาพที่ 4 การแบ่งกลุ่มตัวอักษร

ภาพที่ 5 การใส่ Index

อักษรว่าง (จุดเริ่มต้นของตัวหนังสือ) มี index 0 กลุ่มที่มี index 1 คือ a ดังนั้น a รวมกับตัวเริ่มต้นจะมีหมายเลข 0a กลุ่มที่ 2 คือ aa มีหมายเลข 1a เพราะมีตัวอักษร a อยู่ด้วยซึ่งมี index 1 และอักษรตัวใหม่คือ a ดังแสดงในภาพที่ 6 หลังจากนั้นจะเป็นการคำนวณหาจำนวนบิตที่ต้องใช้ในการเข้ารหัสแบบนี้ ดังที่กล่าวข้างต้นกลุ่มของอักษรจะประกอบด้วยตัวเลขและตัวอักษร จำนวนบิตที่ต้องใช้ในกับตัวเลขกับ index n โดยส่วนมากจะเท่ากับจำนวนบิตที่ใช้กับ index ตัวที่ (n-1) ตัวอย่างเช่น จำนวนบิตที่ใช้กับ 6 ในกลุ่มที่ 8 จะเท่ากับ 3 เพราะมันต้องใช้ 3 [บิตกับ 7 (index ที่ n-1) ในระบบ binary ทุกๆตัวอักษรจะใช้ 8 bits เพราะใช้รูปแบบ ASCII ดังแสดงขั้นตอนในภาพที่ 7



ภาพที่ 6 การใส่หมายเลข



ภาพที่ 7 การคำนวณบิตที่ใช้งาน

หนึ่งในข้อดีของการบีบอัดแบบ Lempel-Ziv คือ ในการณีที่ตัวอักษรมีขนาดยาว จำนวนบิตที่ใช้ในการส่งจะมีขนาดเล็กมากเมื่อเทียบกับความยาวจริง ตัวอย่างเช่น ใช้ 12 bits ใน

การส่ง 2b แทนที่จะเป็น 24 bits (8+8+8) ในการใช้ส่งอันจริงคือ aab นั้นเอง และจำนวนบิตเฉลี่ยต่อ Symbol ไม่ว่าจะเป็นการบีบอัดแบบใดก็ตาม จะใช้โมเดล random independent symbol อย่างน้อยจะมีค่าเท่ากับ entropy โดย Huffman code สามารถทำได้เท่ากับ entropy แต่จะต้องรู้ค่าความน่าจะเป็นของแต่ละ symbol ซึ่งเป็นที่น่าสนใจที่ Lempel-Ziv ทำได้ดีพอๆ กัน โดยไม่ต้องการข้อมูลความน่าจะเป็น

2.2 ส่วนประมวลผลดิจิตอลเบสแบนด์ (Baseband Processing Module)

ส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์มีความจำเป็นในระบบสื่อสารไร้สายแบบดิจิตอล และถูกใช้อย่างแพร่หลายในระบบต่างๆ เช่น ระบบโทรศัพท์เคลื่อนที่ ระบบสื่อสารผ่านดาวเทียม ระบบสื่อสารข้อมูลคอมพิวเตอร์ เป็นต้น เนื่องจากเป็นส่วนที่เพิ่มความทนทานต่อสัญญาณรบกวนให้แก่ข้อมูล ซึ่งมีผลอย่างมากในช่องสัญญาณไร้สาย โดยเฉพาะอย่างยิ่งในระบบสื่อสารผ่านดาวเทียม เนื่องจากเป็นการสื่อสารระยะไกลที่มีทั้งการลดทอนสัญญาณที่สูงมาก และสัญญาณรบกวนอื่นๆ ที่อาจจะเกิดขึ้น ประกอบกับข้อจำกัดทางด้านพลังงานส่งจากดาวเทียม และที่สำคัญคือไม่สามารถเปลี่ยนแปลงแก้ไขส่วนที่อยู่บนดาวเทียมได้เลย การออกแบบส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ดังกล่าวที่สถานีภาคพื้นดินจึงมีความสำคัญมาก และเป็นปัจจัยที่สำคัญที่จะทำให้การสื่อสารเป็นไปตามข้อกำหนดที่ได้วางไว้ ในกรณีของดาวเทียมดวงนี้คือ 2 Mbps @ BER $\leq 10^{-6}$

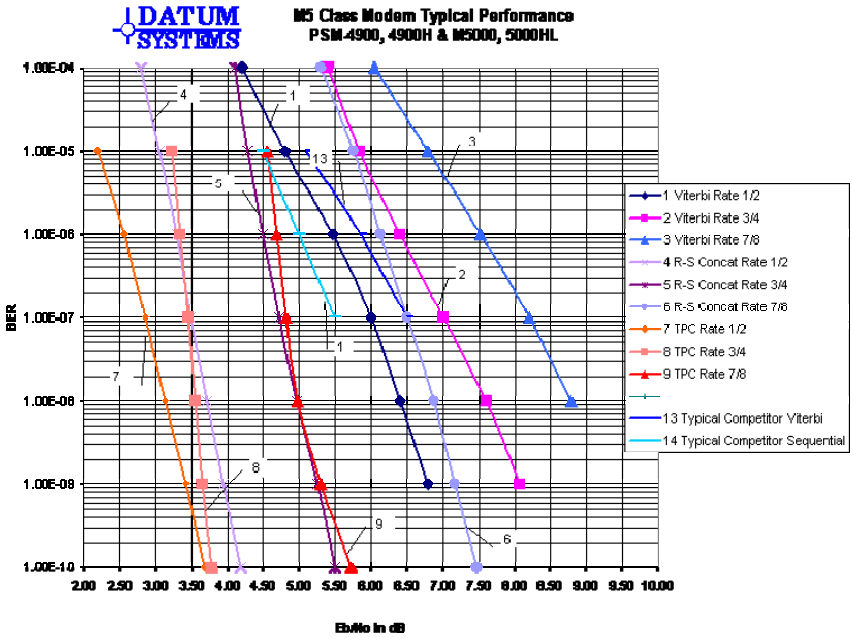
หลักการเบื้องต้นที่สำคัญของการประมวลผลสัญญาณดิจิตอลเบสแบนด์คือ การนำข้อมูลที่ได้รับมาทำการจัดรูปแบบเฟรม (Framing) เพื่อสามารถกำหนดขอบเขตของข้อมูลที่มีความสัมพันธ์กัน แล้วจึงนำการคำนวณทางคณิตศาสตร์เพื่อเพิ่มส่วนตรวจสอบความผิดพลาดของข้อมูล (CRC Generation) และทำการเพิ่มส่วนเข้ารหัสช่องสัญญาณ (Channel Coding) ที่ทำหน้าที่หลักในการเพิ่มความซ้ำซ้อนของข้อมูล เพื่อทำให้ภาครับแก้ไขข้อมูลที่ผิดพลาดจากความซ้ำซ้อนดังกล่าวได้ จากนั้นก็จะทำการสลับบิตข้อมูลที่ได้ (Interleaving) เพื่อแก้ไขปัญหาในกรณีที่เกิดสัญญาณรบกวน และอาจจะเกิดความผิดพลาดแบบต่อเนื่อง (burst error) ขึ้นได้ ซึ่งการสลับนี้จะทำให้ความต่อเนื่องนั้นถูกกระจายแบบสุ่ม และสามารถแก้ไขได้ด้วยส่วนถอดรหัสช่องสัญญาณที่ภาครับ จากนั้นจึงทำการมอดูเลตแบบดิจิตอล (Digital Modulation) และกรองสัญญาณ (Baseband Filter) เพื่อปรับรูปสัญญาณให้เหมาะสมและบรรเทาปัญหาการทับซ้อนกันรัหว่างกลุ่มข้อมูล (Inter-Symbol-Interference – ISI) แล้วจึงส่งเข้าส่วนแปลงสัญญาณจากดิจิตอลให้กลายเป็นอนาลอก

(Digital to Analog Converter – DAC) เพื่อขยายความถี่ให้ไปอยู่ที่ความถี่กลาง (IF Frequency) แล้วถึงจะส่งผ่านไปยังส่วนภาคสัญญาณวิทยุเพื่อส่งผ่านช่องสัญญาณต่อไป ตามที่แสดงไว้ในภาพที่ 1

ในภาครับก็จะทำกระบวนการย้อนกลับจากที่ได้กล่าวไว้ในข้างต้น เพียงแต่ว่าจะมีส่วนที่สำคัญอีกส่วนหนึ่งที่ใช้สำหรับการเข้าจังหวะสัญญาณเพิ่มเข้ามา เพื่อใช้ในการแก้ไขปัญหาการ Doppler ของสัญญาณ โดยจะต้องมีการปรับ Carrier และ Timing เพื่อให้ได้ข้อมูลออกมาในลักษณะของบิตข้อมูล จากนั้นจะต้องไปหาจุดเริ่มต้นของเฟรม โดยการหา Peak ของสัญญาณจาก Pilot ที่ได้ใส่แฝงเอาไว้ก่อนที่จะส่งข้อมูลออกไป โดยได้เลือกใช้การทำ Recursive Costas Loop และ Non data-aided early-late delay synchronizer ในการปรับ Carrier และ Timing ตามลำดับ โดยทั้ง 2 ตัวนี้จะกล่าวรวมกันเรียกว่า Bit Synchronize และจะใช้การ Correlation เพื่อหาจุดเริ่มของเฟรมข้อมูลเพื่อทำการถอดรหัสต่อไป โดยจะเรียกว่าการเข้าจังหวะเฟรมหรือ Frame Synchronization

จากที่ได้กล่าวไว้ข้างต้นส่วนประมวลผลสัญญาณนี้สามารถเรียกได้อีกชื่อหนึ่งว่าส่วนการเข้ารหัสช่องสัญญาณ (Channel Coding) เป็นหลักเพื่อทดสอบสมรรถะของระบบว่ามีความสามารถในการรองรับความผิดพลาดได้มากน้อยเท่าไร โดยจะจำลองการทำงานของช่องสัญญาณ BSC (Binary Symmetric Channel) ขึ้นมา เพื่อให้สามารถตรวจสอบความคงทนต่อสัญญาณรบกวนได้ สำหรับการทดสอบที่จะได้กล่าวต่อไป

โดยในเบื้องต้นได้เลือกที่จะใช้ CRC ขนาด 16 บิต ซึ่งตามปกติแล้วก็จะสามารถตรวจสอบข้อมูลว่ามีความผิดพลาดได้ถึง 10,000 บิต เพื่อใช้ในการตรวจสอบความผิดพลาดของข้อมูลในเฟรม ในส่วนของการเข้ารหัสช่องสัญญาณ ซึ่งในส่วนนี้จะต้องคำนึงถึง Coding Gain เพื่อให้มีความคงทนต่อช่องสัญญาณได้มาก อีกทั้งเป็นการช่วยให้ Link Budget ของดาวเทียมดีขึ้น ซึ่งจะสามารถดูได้จากกราฟ Eb/N0 กับ BER ตามภาพที่ 8



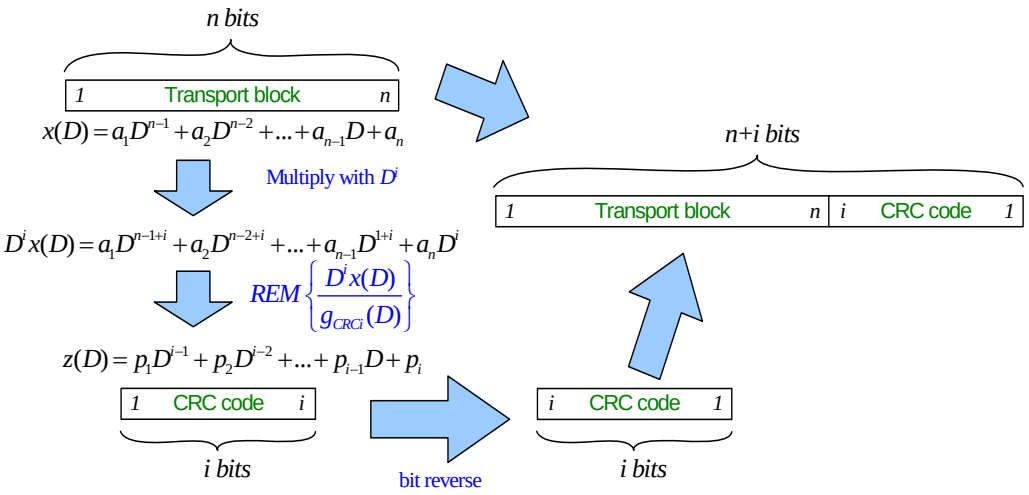
ภาพที่ 8 กราฟแสดงความสัมพันธ์ระหว่าง E_b/N_0 กับ SNR

จะเห็นได้ว่าการเข้ารหัสแบบ Turbo เมื่อเปรียบเทียบกับการเข้ารหัสแบบอื่นๆ แล้วจะสามารถให้ Coding Gain ที่มากที่สุด ดังนั้นทางคณะผู้วิจัยจึงได้ทำเลือกที่จะใช้การเข้ารหัสแบบนี้ในการพัฒนา เพื่อให้เกิดความรวดเร็วมากที่สุดในการเลือกมาตรฐานที่ใช้ในการส่งข้อมูล ดังนั้นจะเลือกใช้มาตรฐานระบบโทรศัพท์เคลื่อนที่ยุคที่ 3 แบบ W-CDMA เป็นไปตามมาตรฐาน 3G TS25.212 และจากการศึกษาจากหลายๆ มาตรฐานเช่น CCSDS, Inmarsat เป็นต้น พบว่าช่องสัญญาณเหล่านี้มีสัญญาณรบกวนที่เป็นลักษณะต่อเนื่อง ก็คือถ้ามีความผิดพลาดเกิดขึ้นก็จะผิดแบบต่อเนื่อง และเป็นช่วงๆ ซึ่งส่วนใหญ่แล้วในการส่งข้อมูลผ่านดาวเทียมเองจะใช้ Reed-Solomon อย่างเดียวหรือ Convolution/Turbo ควบคู่ไปกับเทคนิก Interleaver ดังนั้นเพื่อให้สามารถลดความผิดพลาดให้ได้มากที่สุด จึงจำเป็นจะต้องมีการทำ Interleaver ให้กับข้อมูลด้วย และจะเป็นการเพิ่มประสิทธิภาพของอัตราในการเกิดความผิดพลาดให้น้อยลงอีกด้วย โดยรูปแบบการสลับบิตจะได้กล่าวในหัวข้อต่อไป

2.2.1 การเข้ารหัสและถอดรหัส CRC

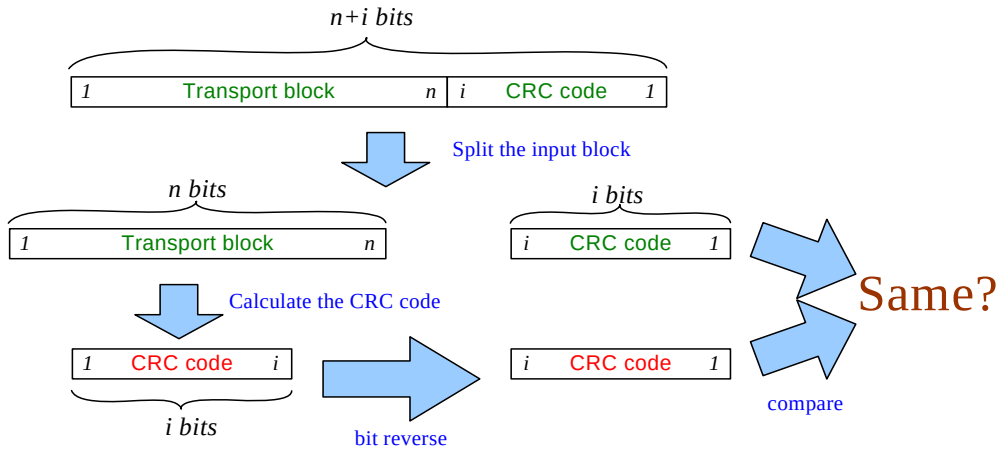
CRC (Cyclic Redundancy Check) จะเป็นการเข้ารหัสเพื่อที่จะใช้ในการตรวจสอบความถูกต้องของข้อมูลในแต่ละเฟรม โดยมีการเข้ารหัสเป็นไปตามสมการ $g_{CRC16}(D) = D^{16} + D^{12} + D^5 + 1$

รหัส CRC หาได้จาก $REM\left(\frac{D^i x(D)}{g_{CRC}(D)}\right)$ เศษเหลือจากการหารด้วยพหุนามตัวสร้าง (Generator Polynomial) โดย x(D) เป็นพหุนามของบล็อกข้อมูลอินพุต และ i คืออันดับ (order) ของพหุนามตัวสร้าง เมื่อได้รหัส CRC แล้วจึงทำการเรียงลำดับของรหัส CRC ที่ได้ใหม่และนำมาต่อกับบล็อกข้อมูล ซึ่งขั้นตอนการเพิ่มรหัส CRC แสดงดังภาพที่ 9



ภาพที่ 9 การเข้ารหัสสัญญาณ CRC

ส่วนการถอดรหัส CRC บล็อกอินพุตที่รับเข้ามาจะถูกนำมาแยกส่วนของข้อมูลและส่วนของรหัส CRC หลังจากนั้นนำบล็อกส่วนที่เป็นข้อมูลมาคำนวณหารหัส CRC และจึงทำการกลับตำแหน่งบิตของรหัส นำรหัสที่แยกไว้ในตอนแรกมาเปรียบเทียบ ถ้าหากเหมือนกันแสดงว่าบล็อกข้อมูลที่รับเข้ามาถูกต้อง โดยมีขั้นตอนดังภาพที่ 10

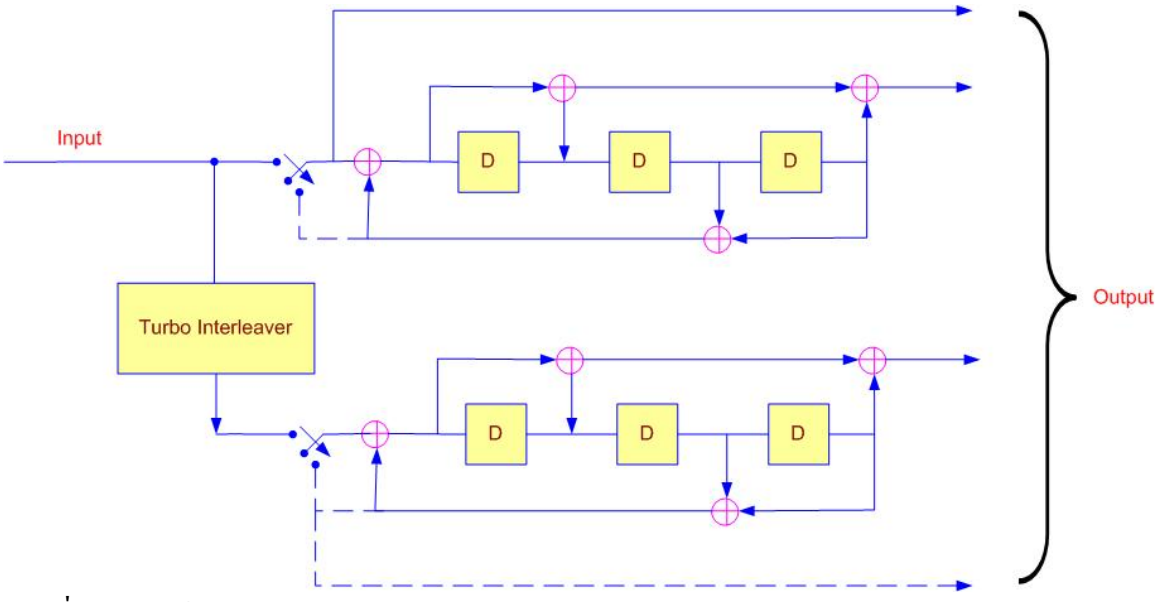


ภาพที่ 10 การถอดรหัสสัญญาณ CRC

โดยโครงสร้างของตัวเข้ารหัส และถอดรหัส CRC จะมีอยู่ 2 ส่วน ก็คือ FIFO (First In First Out) เพื่อช่วยในการเก็บข้อมูลเพื่อประมวลผล ในกรณีที่ตัวเข้ารหัสตัวถัดไปไม่พร้อมจะทำงาน และสองก็คือตัวเข้ารหัส และถอดรหัส สัญญาณ ซึ่งก็เป็นไปตามที่ได้กล่าวไว้ข้างต้น ทั้งนี้ได้มีการใช้ทรัพยากรบนชิป FPGA สำหรับส่วน CRC Encoder จะใช้ไป 77 Slices หรือ 173.25 Logic Cells และความถี่สูงสุดที่สามารถทำงานได้คือ 152.3 MHz ส่วน CRC Decoder จะใช้ไป 83 Slices หรือ 186.75 Logic Cells และความถี่สูงสุดที่สามารถทำงานได้คือ 181.3 MHz โดยในส่วนของ Decoder ที่ใช้ทรัพยากรมากกว่าเนื่องจากมีวงจรเปรียบเทียบอยู่นั้นเอง

2.2.2 การเข้ารหัส Turbo Code

จากที่ได้กล่าวไว้ในข้างต้นการเข้ารหัสที่ได้ทำการเลือกใช้จะเป็น Turbo และเพื่อความรวดเร็วในการทำงานก็จะใช้มาตรฐานโทรศัพท์เคลื่อนที่ในยุคที่ 3 ระบบ W-CDMA ซึ่งจะใช้การเข้ารหัส Turbo อัตรา 1/3 ตามภาพที่ 11 ประกอบด้วย 2 ส่วนหลักๆ คือ ส่วนการเข้ารหัสบิตข้อมูล และส่วนการสลับบิตของเทอร์โบ โดยเส้นปะในรูปจะทำงานเมื่อจบการเข้ารหัสทั้งหมดเพื่อทำการเคลียร์รีจิสเตอร์เมื่อจบการทำงานโดย 3 บิตรองสุดท้ายใช้สำหรับสายบน และ 3 บิตสุดท้ายจะเป็นสายล่าง



ภาพที่ 11 การเข้ารหัส Turbo Code

โดยขั้นตอนในการทำการสลับบิตของเทอร์โบ (Turbo Interleaver) จะเป็นไปดังต่อไปนี้

1) หาจำนวนแถวของเมตริกซ์(R) ให้ K คือ จำนวนอินพุตบิตที่เข้ามาทำการสลับบิตของเทอร์โบ ซึ่งอินพุตบิตนี้จะถูกนำไปเรียงลงในเมตริกซ์ โดยจะหาจำนวนแถวของเมตริกซ์(R) ได้จาก

$$R = \begin{cases} 5, \text{if } (40 \le K \le 159) \\ 10, \text{if } ((160 \le K \le 200) \text{ or } (481 \le K \le 530)) \\ 20, \text{if } (K = \text{any other value}) \end{cases}$$ ลำดับแถวจาก 0 ถึง R-1 จากบนลงล่าง

2) หาจำนวนคอลัมน์(C) ค่าจำนวนเฉพาะ (p) และค่าราก (v) จากตารางที่ 1

if $(481 \le K \le 530)$ then

p = 53 and C = p.

else

หาค่าจำนวนเฉพาะ (p) ที่น้อยที่สุดที่เป็นไปตามอสมการนี้ $K \le R \times (p+1)$,

และหาค่า C ดังนี้

$$C = \begin{cases} p-1 & if \quad K \le R \times (p-1) \\ p & if \quad R \times (p-1) < K \le R \times p \\ p+1 & if \quad R \times p < K \end{cases}.$$

end if

โดยคอลัมน์ของเมตริกซ์มีอันดับจาก 0 ถึง C-1 เรียงจากซ้ายไปขวา

ตารางที่ 1 ค่าจำนวนเฉพาะ P และค่าราก V

p	**v**	**p**	**v**	**p**	**v**	**p**	**v**	**p**	**v**
7	3	47	5	101	2	157	5	223	3
11	2	53	2	103	5	163	2	227	2
13	2	59	2	107	2	167	5	229	6
17	3	61	2	109	6	173	2	233	3
19	2	67	2	113	3	179	2	239	7
23	5	71	7	127	3	181	2	241	7
29	2	73	5	131	2	191	19	251	6
31	3	79	3	137	3	193	5	257	3
37	2	83	2	139	2	197	2		
41	6	89	3	149	2	199	3		
43	3	97	5	151	6	211	2		

3) เขียนลำดับอินพุตบิต $x_1, x_2, x_3, \ldots, x_K$ ลงในเมตริกซ์ $R \times C$ ตามลำดับ โดยเริ่มจาก y_1 ในคอลัมน์ 0 ของแถว 0

$$\begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_C \\ y_{(C+1)} & y_{(C+2)} & y_{(C+3)} & \cdots & y_{2C} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ y_{((R-1)C+1)} & y_{((R-1)C+2)} & y_{((R-1)C+3)} & \cdots & y_{R\times C} \end{bmatrix}$$

เมื่อ $y_k = x_k$ สำหรับ $k = 1, 2, \ldots, K$ และถ้า $R\times C > K$, จะต้องใส่ dummy bits ให้เต็มเมตริกซ์ $R\times C$ โดย dummy bits เหล่านี้จะถูกตัดออกจากเอาท์พุตของ rectangular matrix หลังจากการสลับบิตของเทอร์โบแล้ว

4) สร้างลำดับพื้นฐาน (Base sequence) $\langle s(j)\rangle_{j\in\{0,1,\cdots,p-2\}}$ สำหรับการสลับภายในแถวดังนี้

$$s(j) = (\nu \times s(j-1)) \bmod p \text{ , } j = 1, 2, \ldots, (p-2), \text{ และ } s(0) = 1$$

5) กำหนดให้ $q_0 = 1$ เป็นจำนวนเฉพาะตัวแรก ในลำดับ $\langle q_i\rangle_{i\in\{0,1,\cdots,R-1\}}$ และหาค่า prime integer q_i ในลำดับ $\langle q_i\rangle_{i\in\{0,1,\cdots,R-1\}}$ ให้เป็นค่าจำนวนเฉพาะที่น้อยที่สุดที่ทำให้ ห.ร.ม.$(q_i, p-1) = 1$, $q_i > 6$, และ $q_i > q_{(i-1)}$ สำหรับแต่ละ $i = 1, 2, \ldots, R-1$ เมื่อ ห.ร.ม. คือ หารร่วมมาก ซึ่งวิธีการหาค่านี้ไม่ได้ใช้การหา ห.ร.ม.โดยตรง แต่เราใช้วิธีการพิจารณาดังต่อไปนี้ โดย q_i เป็นจำนวนเฉพาะ เมื่อต้องการหา ห.ร.ม. $(q_i, p-1) = 1$ นั่นคือ

ถ้า $q_i > p-1$ หมายความว่า ห.ร.ม. $(q_i, p-1) = 1$ เป็นจริง

ถ้า $q_i < p-1$ แล้ว ถ้า $(p-1)/q_i$ ลงตัวนั่นคือ ห.ร.ม. $(q_i, p-1)$ ไม่เท่ากับ 1

ถ้า $(p-1)/q_i$ ไม่ลงตัวนั่นคือ ห.ร.ม. $(q_i, p-1)$ เท่ากับ 1

6) คำนวณหาค่าลำดับ $\langle r_i\rangle_{i\in\{0,1,\cdots,R-1\}}$ โดยให้ $r_{T(i)} = q_i$, $i = 0, 1, \ldots, R-1$, เมื่อ $\langle T(i)\rangle_{i\in\{0,1,\cdots,R-1\}}$ คือรูปแบบการสลับระหว่างแถวที่ถูกกำหนดเป็นดังตารางที่ 2 โดยขึ้นกับค่าจำนวนของอินพุตบิต K

ตารางที่ 2 รูปแบบการสลับระหว่างแถวสำหรับสลับบิตของเทอร์โบ

จำนวนบิตอินพุต K	**จำนวนแถว**	**รูปแบบการสลับ <$T(0), T(1), \ldots, T(R-1)$>**
$(40 \le K \le 159)$	5	<4, 3, 2, 1, 0>
$(160 \le K \le 200)$ or $(481 \le K \le 530)$	10	<9, 8, 7, 6, 5, 4, 3, 2, 1, 0>
$(2281 \le K \le 2480)$ or $(3161 \le K \le 3210)$	20	<19, 9, 14, 4, 0, 2, 5, 7, 12, 18, 16, 13, 17, 15, 3, 1, 6, 11, 8, 10>
K = ค่าอื่น ๆ	20	<19, 9, 14, 4, 0, 2, 5, 7, 12, 18, 10, 8, 13, 17, 3, 1, 16, 6, 15, 11>

7) ทำการสลับภายในแถวที่ i-th ($i = 0, 1, \ldots, R-1$) ดังนี้

if ($C = p$) then

$U_i(j) = s((j \times r_i) \bmod (p-1))$, $j = 0, 1, \ldots, (p - 2)$, and $U_i(p - 1) = 0$,

เมื่อ $U_i(j)$ คือ ตำแหน่งบิตเริ่มแรกของบิตที่ถูกสลับในคอลัมน์ที่ j ของแถวที่ i

end if

if ($C = p + 1$) then

$U_i(j) = s((j \times r_i) \bmod (p-1))$, $j = 0, 1, \ldots, (p - 2)$. $U_i(p - 1) = 0$, and $U_i(p) = p$,

เมื่อ $U_i(j)$ คือ ตำแหน่งบิตเริ่มแรกของบิตที่ถูกสลับในคอลัมน์ที่ j ของแถวที่ I, และ

if ($K = R \times C$) then

สลับ $U_{R-1}(p)$ กับ $U_{R-1}(0)$

end if

end if

if ($C = p - 1$) then

$U_i(j) = s((j \times r_i) \bmod (p-1)) - 1$, $j = 0, 1, \ldots, (p - 2)$,

เมื่อ $U_i(j)$ คือ ตำแหน่งบิตเริ่มแรกของบิตที่ถูกสลับในคอลัมน์ที่ j ของแถวที่ i

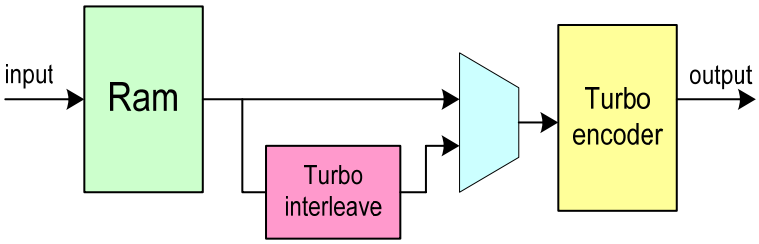
end if

8) ทำการสลับระหว่างแถวของเมตริกซ์ โดยใช้พื้นฐานรูปแบบของ $\langle T(i) \rangle_{i \in \{0,1,\cdots,R-1\}}$ เมื่อ $T(i)$ เป็นตำแหน่งแถวเริ่มแรกของแถวที่ i ที่ถูกสลับ

9) หลังจากการสลับภายในและระหว่างแถว บิตของเมตริกซ์ ที่ถูกสลับจะแทนด้วย y'_k โดยดึงเอาท์พุตออกตามแนวคอลัมน์ โดยตัดส่วน dummy bit ที่เติมในตอนแรกออกไป

$$\begin{bmatrix} y'_1 & y'_{(R+1)} & y'_{(2R+1)} & \cdots & y'_{((C-1)R+1)} \\ y'_2 & y'_{(R+2)} & y'_{(2R+2)} & \cdots & y'_{((C-1)R+2)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ y'_R & y'_{2R} & y'_{3R} & \cdots & y'_{C \times R} \end{bmatrix}$$

การทำงานของฟังก์ชั่นเทอร์โบที่ได้สร้างขึ้นมาตามภาพที่ 12 จะใช้ RAM เป็นตัวช่วยเก็บบิตข้อมูลที่เข้ามาให้ครบก่อน จากนั้นก็จะเริ่มการคำนวณ Turbo Interleaver ตามที่ได้กล่าวไว้ข้างต้น พร้อมๆ กับการทำการเข้ารหัสบิตข้อมูลตามภาพที่ 11 ไปทีละชุด โดยบิตเอาพุทจะออกมาทีละ 3 บิต จนกระทั่งบิตสุดท้ายหลังจากนั้นจะทำการเคลียร์รีจีสเตอร์อีก 6 บิต ซึ่งบิตเอาพุทจะมีขนาดเท่ากับ 3 เท่าของ (บิตอินพุท+6)



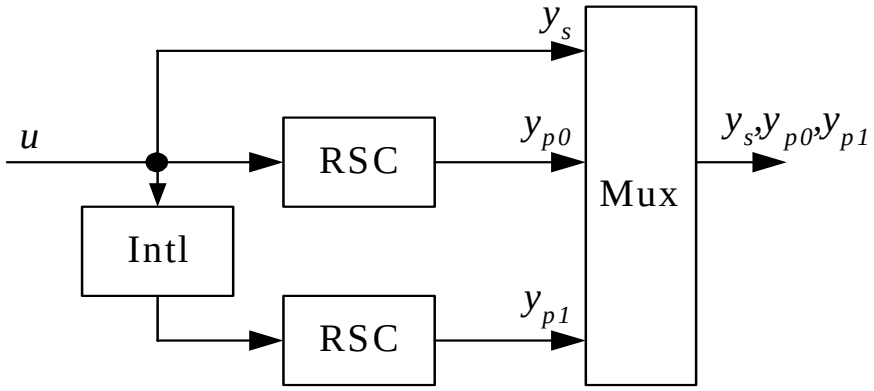
ภาพที่ 12 โครงสร้างตัวเข้ารหัส Turbo Encoder

โดยทรัพยากรที่ใช้ในการเข้ารหัส Turbo จะใช้ไป 161 Slices หรือประมาณ 362.25 Logic cells และใช้หน่วยความจำประมาณ 20 KB และความถี่สูงสุดที่ทำงานได้อยู่ที่ 168.81 MHz

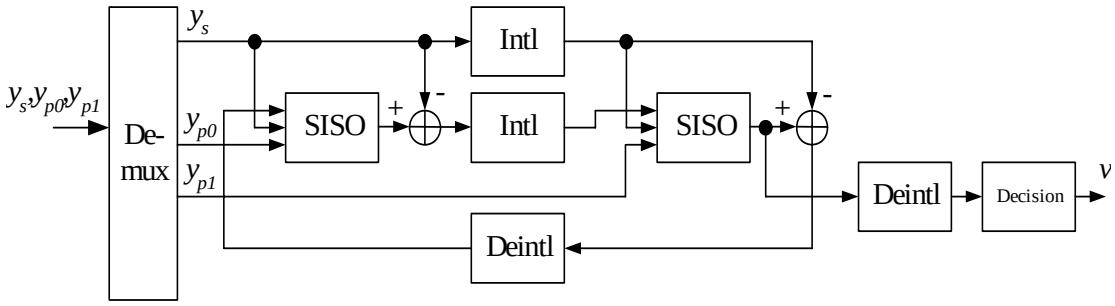
2.2.3 การถอดรหัส Turbo

การเข้ารหัสแบบเทอร์โบโค๊ดเป็นการเข้ารหัสที่มีการใช้ส่วนเข้ารหัสย่อยมากกว่าหนึ่งตัวมาสร้างคำรหัส (code word) โดยรหัสข้อมูลที่ส่งให้แก่ส่วนเข้ารหัสย่อยแต่ละตัวจะพยายามให้มีความสัมพันธ์กันต่ำ ซึ่งโดยปกตินิยมใช้การสลับบิตข้อมูลแบบกึ่งสุ่ม (pseudo-random Interleaver) คำรหัสที่ได้ เป็นรหัสที่ได้จากสายรหัสข้อมูลต้นฉบับ รวมเข้ากับสายรหัสพาริตี้ที่สร้างได้จากส่วนเข้ารหัสย่อยทั้งหมด ลักษณะการเข้ารหัสที่กล่าวมานี้จะใช้วิธีการถอดรหัสแบบวนรอบ (iteration) เพื่อดึงประสิทธิผลที่มีทั้งหมดออกมา วิธีการถอดรหัสแบบเทอร์โบโค๊ดเป็นการใช้ส่วนถอดรหัสย่อยหลายๆ ตัวมาช่วยในการถอดคำรหัสที่รับเข้ามา และแลกเปลี่ยนความผิดพลาดระหว่างส่วนถอดรหัสย่อยต่างๆ โดยผลที่ได้จะนำมาใช้ในการตัดสินใจเลือกรหัสข้อมูลที่ถูกต้องในภายหลัง ภาพรวมของการถอดรหัสเทอร์โบโค๊ด และเพื่อความรวดเร็วในการออกแบบและพัฒนาจะทำการเลือกใช้การเข้ารหัสถอดรหัสเทอร์โบตามมาตรฐาน 3G TS25.212

การเข้ารหัสเทอร์โบโค๊ดตามข้อกำหนดของ 3G TS25.212 เป็นการเข้ารหัสโดยมีส่วนเข้ารหัสย่อยแบบ RSC (Recursion Systematic Convolutional) จำนวน 2 ตัว และมีส่วนสลับบิตข้อมูลแบบกึ่งสุ่มจำนวน 1 ตัว ดังแสดงในภาพที่ 13 การถอดรหัสเทอร์โบโค๊ดนี้จึงจำเป็นต้องใช้ส่วนถอดรหัสย่อยแบบ SISO (Soft in Soft out) จำนวน 2 ตัว เพื่อใช้ถอดรหัสที่สร้างได้จากส่วนเข้ารหัสย่อยในแต่ละตัว และแลกเปลี่ยนข้อมูลเกี่ยวกับความผิดพลาดระหว่างสายรหัส ดังแสดงในภาพที่ 14 ในทางปฏิบัติสามารถสร้างส่วนถอดรหัสเพียงตัวเดียวให้ทำงานเป็นส่วนถอดรหัสย่อยแต่ละตัวในเวลาต่างกันได้ ซึ่งทำให้สามารถใช้ประโยชน์จากวงจรได้อย่างสูงสุด

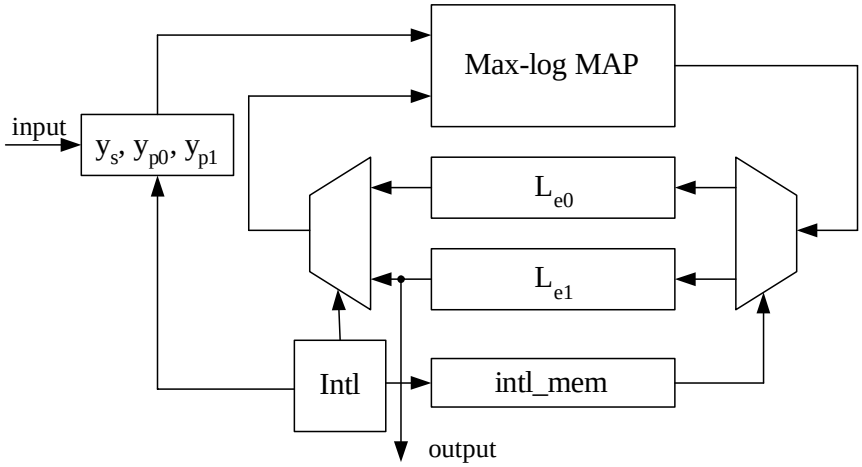


ภาพที่ 13 แผนภาพแสดงการทำงานของส่วนเข้ารหัสเทอร์โบโค๊ดแบบใช้ส่วนเข้ารหัสย่อยแบบ RSC จำนวน 2 ตัว



ภาพที่ 14 แผนภาพแสดงการทำงานของส่วนถอดรหัสเทอร์โบโค๊ดที่สร้างได้จากการเข้ารหัส

วงจรถอดรหัสเทอร์โบโค๊ดที่ออกแบบมีโครงสร้างโดยรวมแสดงในภาพที่ 15 ซึ่งมีส่วนประกอบหลักของการถอดรหัสเทอร์โบโค๊ด คือ ส่วนถอดรหัสย่อยซึ่งในส่วนเราจะใช้การถอดรหัสแบบ Max-Log MAP (Maximum A Posteriori), ส่วนสร้างตำแหน่งการสลับบิตข้อมูล หน่วยความจำ extrinsic หน่วยความจำคำรหัส และส่วนจักรกลสถานะ (Main State Machine)



ภาพที่ 15 โครงสร้างของวงจรถอดรหัส

ส่วนสร้างตำแหน่งการสลับบิตข้อมูลเริ่มการคำนวณหาพารามิเตอร์ที่ใช้ในการสร้างตำแหน่งในการสลับบิตข้อมูล หลังจากที่คำรหัสที่รับเข้ามาถูกนำมาเก็บไว้ในหน่วยความจำคำรหัสเรียบร้อยแล้ว สายรหัสต้นฉบับ y_s และสายรหัสพาริตี้ y_{p0} จะถูกลำเลียงให้แก่ส่วนถอดรหัสแบบ Max-Log MAP หลังจากคำนวณหาค่าแล้วจะลำเลียงนำค่าไปจัดเก็บในหน่วยความจำ extrinsic L_{e0} โดยตำแหน่งที่จัดเก็บจะได้จากส่วนสร้างตำแหน่งการสลับบิตข้อมูล เมื่อเสร็จสิ้นสายรหัสต้นฉบับ y_s และสายรหัสพาริตี้ y_{p1}พร้อมค่าที่จัดเก็บในหน่วยความจำ extrinsic L_{e0} จะถูกลำเลียงให้แก่ส่วนถอดรหัสแบบ Max-Log MAP โดยตำแหน่งจะขึ้นกับส่วนสร้างตำแหน่งการสลับบิตข้อมูล หลังจากคำนวณหาค่าแล้วจะลำเลียงมาจัดเก็บในหน่วยความจำ extrinsic L_{e1} โดยตำแหน่งที่อยู่ใน intl_mem เมื่อทำจนเสร็จสิ้นจะถือว่าเป็นหนึ่งรอบของการถอดรหัส หลังจากนั้นจะเริ่มรอบที่สองในลักษณะเดียวกันนี้ เมื่อจำนวนรอบของการถอดรหัสเพิ่มขึ้น ค่าอัตราบิตผิดพลาด (BER) มักมีค่าต่ำลง แต่ในทางกลับกันอัตราการไหลของข้อมูลที่ผ่านส่วนถอดรหัสก็มีค่าต่ำลง

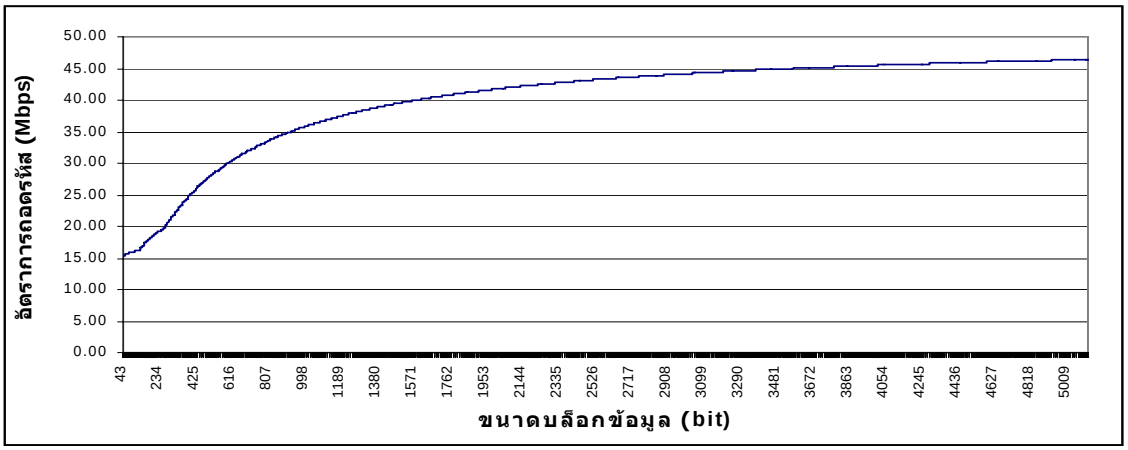
ส่วนถอดรหัสแบบ Max-Log MAP เป็นวิธีที่ปรับปรุงจาก MAP โดยแปลงให้อยู่ในโดเมนของล็อการิทึมซึ่งการกระทำคูณและหาร จะแปลงเป็นการกระทำบวกและลบ ทำให้มีความซับซ้อนน้อยลง แต่ก็ต้องแลกกับการสูญเสียค่าประสิทธิผล วิธีในการถอดรหัสมีหลักการง่ายๆ คือ การหาค่าโอกาสที่ข้อมูลที่ตำแหน่งนั้นมีค่าเป็นข้อมูลอะไร ซึ่งเมื่อเทียบกับการถอดรหัสด้วยวิธี Viterbi ซึ่งเป็นการหาคำรหัสที่ใกล้เคียงที่สุดแล้ว การถอดรหัสแบบ MAP จะให้ค่าอัตราบิตผิดพลาดต่ำกว่าแบบ Viterbi แต่ก็จำเป็นต้องใช้หน่วยความจำในการเก็บค่าความน่าจะเป็นเป็นจำนวนมาก และส่วนนี้เป็นส่วนที่ใช้จำนวนรอบของสัญญาณนาฬิกามากกว่าส่วนอื่นๆ จึงมีผลกับอัตราการถอดรหัสของส่วนถอดรหัสเทอร์โบโค๊ดเป็นอย่างมาก โดยตารางที่ 3 แสดงผลจากการจำลองโดยใช้ความถี่ของสัญญาณนาฬิกาที่ 50 MHz ซึ่งจะเห็นว่าขนาดบล็อกข้อมูลที่มีขนาดเล็กจะ

มีอัตราการถอดรหัสต่ำกว่าบล็อกข้อมูลที่มีขนาดใหญ่ และจะมีผลโดยรวมกับอัตราการถอดรหัสเมื่อนำไปใช้ในส่วนถอดรหัสเทอร์โบโค๊ด

ตารางที่ 3 ผลการจำลองของส่วนถอดรหัส Max-Log MAP @50 MHz

ขนาดบล็อกข้อมูล ที่เข้าส่วนถอดรหัส	เวลาที่ใช้ (ไมโครวินาที, μs)	อัตราการถอดรหัส (Mbps)
43 x 3	3.064	14.034
128 x 3	8.164	15.679
512 x 3	18.404	27.820
5,117 x 3	*110.300*	*46.392*

และภาพที่ 16 แสดงการอัตราการถอดรหัสของส่วนถอดรหัสแบบ Max-Log MAP โดยใช้ความถี่ของสัญญาณนาฬิกาที่ 50 MHz ของบล็อกข้อมูลขนาดต่างๆ ตั้งแต่ 43 บิต ถึง 5,117 บิต



ภาพที่ 16 อัตราการถอดรหัสของส่วนถอดรหัสแบบ Max-Log MAP ที่ความถี่สัญญาณนาฬิกา 50 MHz ของขนาดบล็อกข้อมูลต่างๆ

โดยทั้งหมดนี้ใช้ทรัพยากรไปทั้งหมด 2,491 Slices หรือ 5,605 Logic Cells Memory 72 KB และสัญญาณนาฬิกาสูงสุดที่สามารถทำงานได้คือ 96.1 MHz

2.2.4 Interleaver/Deinterleaver

ทำหน้าที่ในการสลับบิตข้อมูลออกเพื่อช่วยกระจายความผิดพลาดอันเนื่องมาจากช่องสัญญาณ ซึ่งความผิดพลาดที่จะเกิดขึ้นในช่องสัญญาณผ่านดาวเทียมจะเป็นในลักษณะของ Burst Error หรือความผิดพลาดแบบต่อเนื่อง ดังนั้นการกระจายข้อมูลออกจะช่วยให้ตัวถอดรหัส Turbo สามารถแก้ไขความผิดพลาดได้ทั้งหมด ถ้าไม่มีการกระจายข้อมูลออกจากกัน ในกรณีที่มีความผิดพลาดเกิดขึ้นแบบต่อเนื่อง ก็จะทำให้ตัวถอดรหัสตัดสินใจผิดพลาดไปด้วย ส่งผลให้เฟรมข้อมูลนั้นๆ เกิดความผิดพลาดขึ้นมาได้ จะเห็นได้ว่าการทำ Interleaver/Deinterleaver นั้นมีความสำคัญอย่างมากในการเข้ารหัส ถอดรหัสสัญญาณต่างๆ

ซึ่งวิธีที่ใช้กันอย่างแพร่หลายอยู่ 2 วิธี คือ แบบ Block Interleaver และแบบ Address Interleaver โดยแบบแรกจะเป็นในลักษณะ Write into Row Read from Column หรือเป็นการตั้งค่าบล็อกการทำงานเอาไว้ อาจจะเป็น 1,2,4,…,2^n โดย n จะเป็นค่า integer ซึ่งวิธีนี้จะสะดวกในการเข้ารหัสและถอดรหัส และวิธีที่สองจะอยู่ในเป็นการคำนวณหา address ในการอ่านข้อมูลออกไป ซึ่งจะใช้สำหรับการแยกข้อมูลออกจากกันให้มีความสัมพันธ์กันให้น้อยที่สุด เช่น การทำ Turbo Interleaver เป็นต้น

โดยได้ทำการเลือกใช้แบบแรกเนื่องจากมีความสะดวกในการสร้างฟังก์ชั่น และการคำนวณก็ไม่ซับซ้อนมาก ซึ่งจะการทำงานเป็นดังนี้

1) เลือกจำนวนคอลัมน์ C1 จากตารางที่ 4 ซึ่งขึ้นกับค่า 2^n โดยเรียงอันดับคอลัมน์ 0, 1, …, C1-1 จากซ้ายไปขวา

ตารางที่ 4 รูปแบบการสลับบิตของ interleaving

จำนวนคอลัมน์ C1	**รูปแบบการสลับบิต <$P1_{C1}(0)$, $P1_{C1}(1)$, …, $P1_{C1}(C1-1)$>**
1	<0>
2	<0,1>
4	<0,2,1,3>
8	<0,4,2,6,1,5,3,7>

2) หาค่าจำนวนแถวของเมตริกซ์, (R1) R1 = X_i / C1 โดยมีลำดับแถวของเมตริกซ์จาก 0, 1, …, R1-1 เรียงจากบนลงล่าง

3) เขียนลำดับอินพุตบิตลงในเมตริกซ์ขนาด R1 X C1 เรียงตามแนวแถว

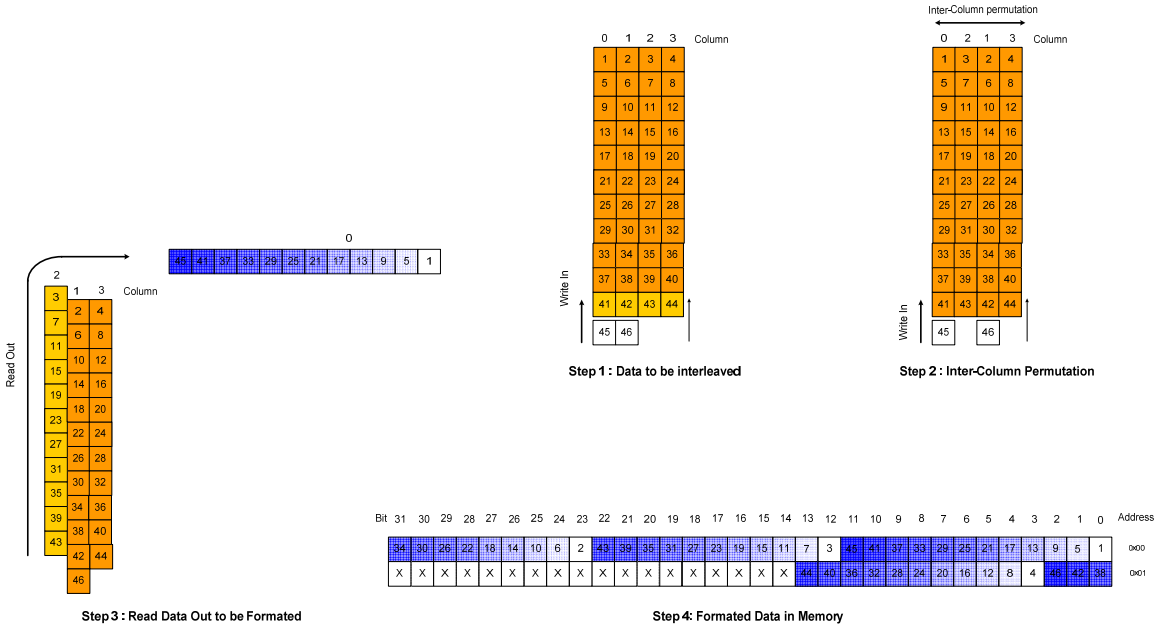
$$\begin{bmatrix} x_{i,1} & x_{i,2} & x_{i,3} & \cdots & x_{i,\mathrm{C1}} \\ x_{i,(\mathrm{C1}+1)} & x_{i,(\mathrm{C1}+2)} & x_{i,(\mathrm{C1}+3)} & \cdots & x_{i,(2\times\mathrm{C1})} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x_{i,((\mathrm{R1}-1)\times\mathrm{C1}+1)} & x_{i,((\mathrm{R1}-1)\times\mathrm{C1}+2)} & x_{i,((\mathrm{R1}-1)\times\mathrm{C1}+3)} & \cdots & x_{i,(\mathrm{R1}\times\mathrm{C1})} \end{bmatrix}$$

4) ทำการสลับระหว่างคอลัมน์สำหรับเมตริกซ์ โดยใช้รูปแบบการสลับ $\langle \mathrm{P1}_{\mathrm{C1}}(j) \rangle_{j\in\{0,1,\ldots,\mathrm{C1}-1\}}$ จากตารางที่ 4

5) อ่านลำดับเอาท์พุตบิตจากเมตริกซ์ตามแนวคอลัมน์ หลังจากสลับข้อมูลไปแล้ว

$$\begin{bmatrix} y_{i,1} & y_{i,(\mathrm{R1}+1)} & y_{i,(2\times\mathrm{R1}+1)} & \cdots & y_{i,((\mathrm{C1}-1)\times\mathrm{R1}+1)} \\ y_{i,2} & y_{i,(\mathrm{R1}+2)} & y_{i,(2\times\mathrm{R1}+2)} & \cdots & y_{i,((\mathrm{C1}-1)\times\mathrm{R1}+2)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ y_{i,\mathrm{R1}} & y_{i,(2\times\mathrm{R1})} & y_{i,(3\times\mathrm{R1})} & \cdots & y_{i,(\mathrm{C1}\times\mathrm{R1})} \end{bmatrix}$$

โดยทางคณะผู้วิจัยได้เลือกที่จะใช้การจำนวน Column เป็น 8 เพื่อความสะดวกในการสร้าง RAM มาจัดเก็บข้อมูล และจะเป็นไปตามตัวอย่างในภาพที่ 17 ดังนี้

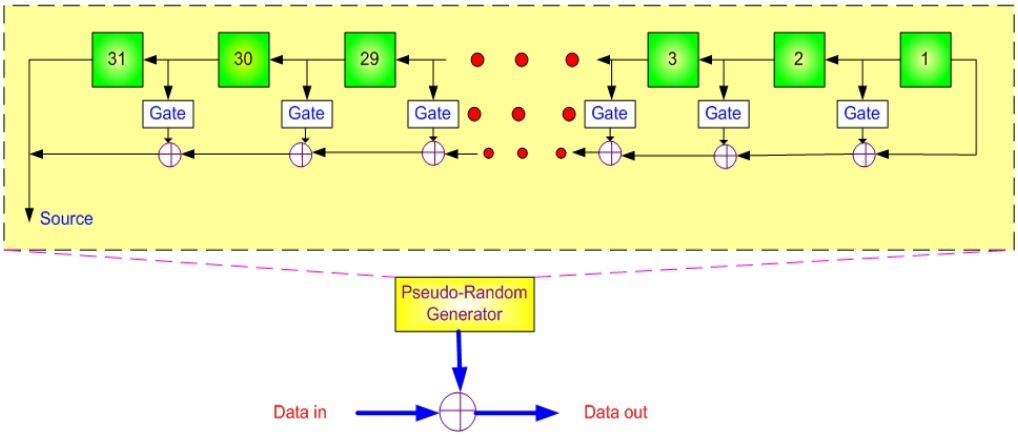


<u>ภาพที่ 17</u> ตัวอย่างการเข้ารหัส Interleaver/Deinterleaver

และในการทำงานนี้ได้ใช้ทรัพยากรไปทั้งหมด 51 Slices หรือ 115 Logic cells ส่วน Memory ประมาณ 3 KB และความถี่สูงสุดที่ทำงานได้ 144.8 MHz สำหรับ Interleaver ส่วน Deinterleaver จะใช้ทรัพยากรไป 56 Slices หรือ 126 Logic cells ส่วน Memory ประมาณ 3 KB และความถี่สูงสุดที่สามารถทำงานได้คือ 169.17 MHz

2.2.5 การเข้ารหัสความปลอดภัยของข้อมูล

การทำงานในส่วนนี้จะช่วยให้ข้อมูลที่ถูกส่งออกไปมีความปลอดภัยมากขึ้น โดยเทคนิกที่ใช้กันอย่างแพร่หลายในปัจจุบัน เช่น DES 192 บิต Hash Function หรือ Random-Sequence ต่างๆ เป็นต้น และเพื่อให้มีความสะดวกในสร้าง ทางคณะผู้วิจัยได้เลือกที่จะใช้การสร้าง Pseudo-Random-Sequence มาทำการ Scramble กับข้อมูลจะทำให้ปริมาณข้อมูลไม่ได้เพิ่มขึ้น ง่ายต่อการจัดการในการประมวลผลสัญญาณเดิมที่ได้พัฒนาไว้แล้ว ซึ่งรูปแบบจะเป็นไปตามภาพที่ 18



ภาพที่ 18 โครงสร้างของส่วนการเข้ารหัสสัญญาณความปลอดภัย

โดยค่า Polynomial จะเท่ากับ $x^{31}+x^{22}+x^{19}+x^{12}+x^{10}+x^{9}+x^{6}+x^{4}+1$ ดังนั้นค่าที่สำคัญก็คือค่าเริ่มต้น ให้กับตัว Generator ดังนั้นการจัดการจะเป็นในลักษณะที่ป้อน Password เข้ามาแล้วแปลง Password นั้นให้เป็นค่าเริ่มต้นได้ ซึ่งกระบวนการป้อน Password ทำที่ส่วนซอฟต์แวร์ประยุกต์ และซอฟต์แวร์ประยุกต์จะทำการแปลงรหัสผ่านนั้นๆ ให้เป็นค่าเริ่มต้น โดยกระบวนการแปลงค่านี้ในเบื้องต้นจะใช้การเข้ารหัส CRC-16 หรือ CRC-32 เป็นตัวจัดการ และรหัสผ่านจะต้องมีการตกลงให้แน่นอนทั้งสองฝั่งว่าเป็นค่าอะไร แนวคิดนี้คล้ายๆ กับการใช้ Public และ Private Key แต่ย่อยให้ง่ายกว่า เพื่อความรวดเร็วในการพัฒนาระบบ

การใช้งานการเข้ารหัสความปลอดภัยจะทำได้ทุกเมื่อที่ผู้ใช้ต้องการจะเข้ารหัส โดยทำการป้อนรหัสส่วนตัวที่ตกลงกันเอาไว้ แล้วต้องรออีกฝั่งทำการป้อนรหัสเพื่อยืนยัน เมื่ออีกฝั่งป้อนรหัสที่ถูกต้องแล้วก็จะสามารถใช้งานในส่วนนี้ได้ นั้นคือจะเข้ารหัสก็ต่อเมื่อทั้งสองฝั่งได้ใส่ Password ที่เหมือนกัน ถ้าใส่แค่ฝั่งใดฝั่งหนึ่งก้ไม่สามารถที่จะเข้ารหัสข้อมูลความปลอดภัยได้ และการใส่รหัสผ่านก็ไม่ได้ตายตัวว่าจะต้องเป็นรหัสนี้เท่านั้น สามารถปรับเปลี่ยนได้ตามแต่ฝู้ใช้จะตกลงกันเอาไว้ ซึ่งในอนาคตก็จะสามารถจำกัดการติดต่อหรือส่งข้อมูลกันเฉพาะได้ ยกตัวอย่างเช่น ถ้านาย ก ต้องการส่งข้อมูลให้นาย ข โดยไม่ให้นาย ค รู้ก็เพียงแต่ใส่ Password ที่รู้กันเฉพาะ 2 คน ถึงแม้ว่านาย ค จะรับข้อมูลนี้ด้วยแต่ก็ไม่มีชุดถอดรหัสที่ถูกต้อง จึงไม่สามารถทำอ่านข้อมูลได้ถูกต้อง และในทำนองเดียวกัน ถ้านาย ค จะส่งข้อมูลให้นาย ข โดยไม่ให้นาย ก รู้ก็จะส่ง Password ที่ตกลงกันไว้แล้ว 2 คน เป็นต้น

ในแง่ของการลักลอบการใช้ถอดรหัสจะเรียกว่าเป็นไปไม่ได้เลยที่จะสามารถถอดรหัสข้อมูลได้อย่างถูกต้องคือ ผู้ที่ลักลอบอย่างแรกที่จะต้องรู้ก็คือเทคนิกที่ได้เลือกใช้ว่าคืออะไร และถ้ารู้ก็จำเป็นต้องรู้ค่า Polynomials และค่าเริ่มต้นซึ่งสร้างมาจาก Password ที่ตกลงไว้ ถึงแม้ว่าจะรู้ Password ก็ไม่ทราบถึงกระบวนการแปลงให้เป็นค่าเริ่มต้นออกมาได้ และที่สำคัญอย่างยิ่งก็คือจังหวะแรกที่ Generator ส่งค่าออกมา (หมายความว่าจังหวะของสัญญาณที่ Generator ส่งค่าออกมา คืออาจจะส่งตั้งแต่ที่ได้รับค่าเริ่มต้นหรือจะส่งในขณะที่มีสัญญาณเข้ามาเท่านั้น หรือจะเป็นการส่งออกมาในขณะใดขณะหนึ่งระหว่างการประมวลผลสัญญาณ) จึงทำให้เกิดความยากในการทายว่าจะเป็นแบบไหน เนื่องจากมีความซับซ้อนในการเข้ารหัสในส่วนนี้มาก ดังนั้นผู้ใช้จะสามารถมั่นใจได้เลยว่าการส่งข้อมูลจะมีความปลอดภัย จะรู้ได้ก็ต่อเมื่อมีรหัสเท่านั้นเอง

จากที่ได้กล่าวมาทั้งหมดนี้จะเห็นได้ว่ากระบวนการถอดรหัสลับข้อมูลนั้นเป็นไปได้ยากมาก เนื่องจากกระบวนการ Scrambling Code สร้างจากการนำ user name และ password ซึ่งเป็น binary data มาผ่านเข้าใน shift register ซึ่งมี polynomial ดังนี้ $x^{31} + x^{22} + x^{19} + x^{12} + x^{10} + x^9 + x^6 + x^4 + 1$ เมื่อผ่านกระบวนการดังกล่าวจะได้ Scrambling Code ขนาด 32 บิตออกมา ซึ่ง Scrambling Code ที่ได้จะแตกต่างกันขึ้นอยู่กับ user name และ password ซึ่งจะนำไปใช้เพื่อให้ข้อมูลที่ส่งมีความปลอดภัย โดยข้อมูลก่อนที่จะส่งจะถูก scrambling ก่อน ดังนั้นผู้ที่ใช้โปรแกรมคนอื่นๆ หรือคนที่มาดักฟัง ก็ไม่สามารถเข้าใจข้อมูลที่ถูกส่งไปได้ การที่จะดักฟังได้ต้องรู้ scrambling code ซึ่งก็เป็นการยากที่จะเดาการเข้ารหัสได้ถูกเนื่องจาก scrambling code มีขนาด 32 bit ซึ่งให้มีรูปแบบที่แตกต่างกันถึง 4,294,967,296 แบบ โดยจะสามารถสรุปขั้นตอนในการสร้างค่าเริ่มต้นให้กับการเข้ารหัสลับข้อมูลได้ดังต่อไปนี้

- ผู้ใช้ Program ใส่ user name และ password ก่อนใช้โปรแกรมทุกครั้ง ซึ่งโปรแกรมจะเก็บข้อมูลนี้ไว้เพื่อนำไปสร้าง Scrambling Code โดย user name และ password จะถูกนำมาต่อกัน โดยทำการแปลงข้อมูลดังกล่าวให้เป็นข้อมูลไบนารี่

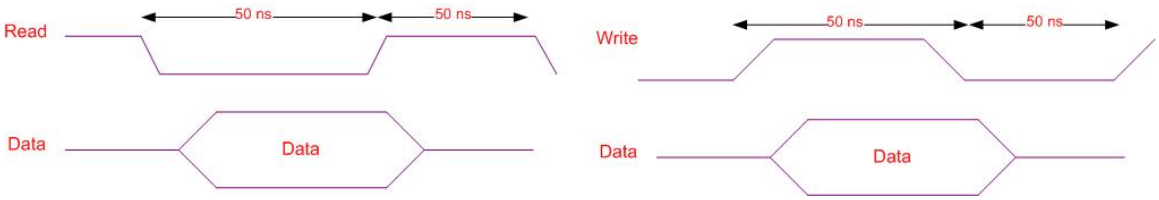
- นำ user name และ password ที่ต่อกันแล้วผ่านเข้าในฟังก์ชัน CRC 32 ซึ่งฟังก์ชันนี้จะมีการทำงานเสมือน shift register ซึ่งมี polynomial ดังนี้ $x^{31} + x^{22} + x^{19} + x^{12} + x^{10} + x^{9} + x^{6} + x^{4} + 1$ โดยการสร้าง scrambling code ทำได้โดยผ่านข้อมูล (user name และ password ที่ต่อกัน) เข้าใน shift register ทีละบิตจนหมดเมื่อใส่หมดแล้วจะได้ scrambling code ขนาด 32 บิตใน register เพื่อนำไปเป็นค่าเริ่มต้นให้กับการเข้ารหัสลับข้อมูล โดยการสร้างข้อมูลในส่วนนี้จะจัดการอยู่ในส่วนของซอฟต์แวร์ประยุกต์ และส่งเฉพาะค่านี้ไปให้แก่ส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ที่ทำงานอยู่บน FPGA เท่านั้น

จะเห็นได้ว่าการจัดการแบบนี้จะให้ความยืดหยุ่นอย่างมากในการเข้ารหัสลับ เนื่องจากเมื่อมีรหัสผู้ใช้ (User name) และรหัสผ่าน (Password) ที่แตกต่างออกไปไม่ว่าตัวใดตัวหนึ่งก็จะส่งผลให้ค่าเริ่มต้นของการเข้ารหัสลับก็จะไม่เหมือนกันไปด้วย และการจัดการแบบนี้ยังส่งผลดีต่อระบบในแง่ความปลอดภัย ยกตัวอย่างเช่น ถ้ามีผู้ไม่หวังดีเข้ามาลักลอบขโมยข้อมูลไปก็ไม่รู้รหัสผู้ใช้งาน และรหัสผ่าน ถึงแม้ว่าจะรู้แต่รหัสพวกนี้ก็จะสามารถเปลี่ยนแปลงไปได้เรื่อยๆ ตามแต่ความต้องการของผู้ใช้ และรู้กันเฉพาะผู้ติดต่อเท่านั้น เช่น นาย ก ส่งข้อมูลให้นาย ข โดยใช้รหัสแบบที่ 1 และแจ้งไปที่นาย ข ว่าใช้รหัสแบบที่ 1 ในครั้งแรก พอครั้งต่อไปนาย ก ส่งข้อมูลไปอีกครั้งโดยทำการเปลี่ยนรหัส ให้เป็นแบบที่ 2 ซึ่งสามารถกระทำได้ทันทีไม่จำเป็นต้องมาแก้ไขในระบบฐานข้อมูล ดังนั้นการเข้ารหัสก็จะเปลี่ยนแปลงไป ซึ่งจะทำให้ผู้ลักลอบไม่สามารถรับรู้รหัสผ่านที่แน่นอนได้เลย และในทำนองเดียวกันถ้านาย ข จะส่งข้อมูลให้นาย ก ก็ไม่จำเป็นต้องใช้รหัสเดิม คือสามารถใช้รหัสใหม่ได้เลยเพียงแค่ว่านาย ก จะต้องรู้ว่าใช้รหัสแบบไหนเท่านั้น เพื่อให้ทั้งสองคนสามารถรับส่งข้อมูลกันได้นั้นเอง เป็นต้น โดยแนวคิดเหล่านี้ก็จะคล้ายๆ กับการใช้ Public Key และ Private Key ตามที่ได้กล่าวไว้

สำหรับทรัพยากรที่ใช้ในการพัฒนาตัวเข้ารหัสนี้จะใช้จำนวน Logic cell ไป 54 cells และความถี่สูงสุดที่สามารถทำงานได้จะเท่ากับ 412 MHz โดยรูปแบบของอินพุท เอาพุท จะเป็นในลักษณะของ Frame, Clk, Data, และ Ready เพื่อให้เป็นไปในทิศทางเดียวกับส่วนประมวลผลสัญญาณอื่นๆ ที่ได้พัฒนา และในการถอดรหัสความปลอดภัยก็จะเป็นการทำงานซ้ำกระบวนการนี้ ก็จะได้ข้อมูลที่ถูกต้องกลับมา

2.2.6 ส่วนติดต่อกับซอฟต์แวร์ประยุกต์

การติดต่อกับบอร์ด USB นั้นจะใช้ในการรับส่งข้อมูลเพื่อไปเข้าคอมพิวเตอร์หรือ notebook โดยจะสามารถอ่านหรือเขียนได้ทีละ 1 Byte เท่านั้น ดังนั้นการทำงานในส่วนนี้จะต้องแยกออกเป็นสองส่วนใหญ่ๆ คือ ส่วนรับส่งข้อมูลกับบอร์ด USB (ทางคณะผู้วิจัยได้เลือกใช้ Ezy USB 1.0 ของบริษัท Astron Logic) ซึ่งเป็นการทำงานแบบ Asynchronies เป็นดังภาพที่ 19



ภาพที่ 19 สัญญาณในการรับส่งข้อมูล

โดยจะมีสัญญาณ Empty และ Full เป็นตัวบอกว่าข้อมูลในบอร์ด USB ว่างหรือเต็ม (บอร์ด USB จะใช้ FIFO เป็นตัวเก็บข้อมูล) และการทำงานได้ตั้งให้อยู่ในลักษณะ *Write before Read* นั้นก็คือ ถ้าสัญญาณ Empty เป็น “0” เมื่อไหร่ ก็จะทำการ Read ข้อมูลจาก FIFO ทันที และจะทำไปเรื่อยๆ จนกระทั้ง Empty เป็น “1” และในทำนองเดียวกัน ถ้ามีสัญญาณมาบอกให้เตรียมตัว Write ข้อมูล ก็จะ Write ไปเรื่อยๆ จนไม่มีสัญญาณมาบอกให้หยุด หรือ ถ้ามีสัญญาณ Full ก็จะทำการหยุด Write ชั่วคราว เพื่อรอให้คอมพิวเตอร์อ่านข้อมูลออกไปจาก FIFO และกรณีที่ทำการ Read ข้อมูลอยู่ และมีสัญญาณให้ Write มา ก็จะทำการ Write ข้อมูลให้ครบก่อน หลังจากนั้นก็จะกลับมา Read ข้อมูลต่อไป

และส่วนที่สองจะเป็นส่วนที่ใช้สำหรับแยกข้อมูลให้เป็นบิตข้อมูล และจัดเก็บข้อมูลให้อยู่ในลักษณะไบค์ข้อมูล โดยจะมีการสร้างสัญญาณเพื่อเริ่มต้นการทำงานของส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ นั้นคือ การเข้ารหัส CRC เป็นส่วนแรก และรับข้อมูลจาก CRC Checker เพื่อจัดเก็บข้อมูล โดยการทำงานในส่วนนี้จะมีความสัมพันธ์กับส่วนแรก เพื่อให้จังหวะในการอ่านหรือเขียนข้อมูลมีความถูกต้องแน่นอน

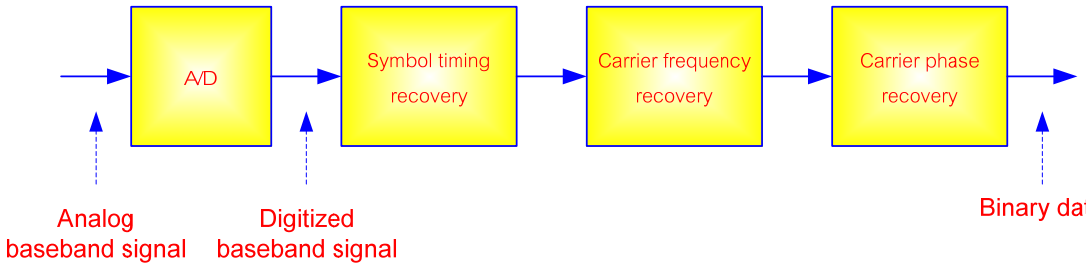
โดยใช้ทรัพยากรไป 188 Slices หรือ 423 Logic Cells และความถี่สูงสุดที่สามารถทำงานได้อยู่ที่ 189.95 MHz และส่วนนี้จะเป็นส่วนนอกสุดที่ใช้ในการรับส่งข้อมูล ดังนั้นส่วนที่สำคัญที่สุดก็คือ การเชื่อมต่อกับบอร์ด Ezy USB

2.3 ส่วนการเข้าจังหวะสัญญาณ (Recovery Module)

ส่วนการทำงานนี้เป็นส่วนที่คอยช่วยสนับสนุนให้ภาครับสัญญาณตัดสินใจว่าข้อมูลที่ได้รับมานั้นเป็นค่าอะไร โดยข้อมูลเหล่านี้ถูกรบกวนโดยสัญญาณรบกวนทำให้มีความผิดพลาดเกิดขึ้นระหว่างการส่งในช่องสัญญาณ ซึ่งส่วนการเข้าจังหวะนี้จะทำหน้าที่สร้างสัญญาณนาฬิกาเพื่อเข้าจังหวะเวลา (Symbol timing recovery) และสัญญาณพาหะ (Carrier recovery) สำหรับนำไปใช้ในการสร้างข้อมูลรับกลับคืน ถือว่าการทำงานในส่วนนี้เป็นหัวใจสำคัญในการประมวลผลสัญญาณดิจิตอลเบสแบนด์ในภาครับสัญญาณเลยก็ว่าได้

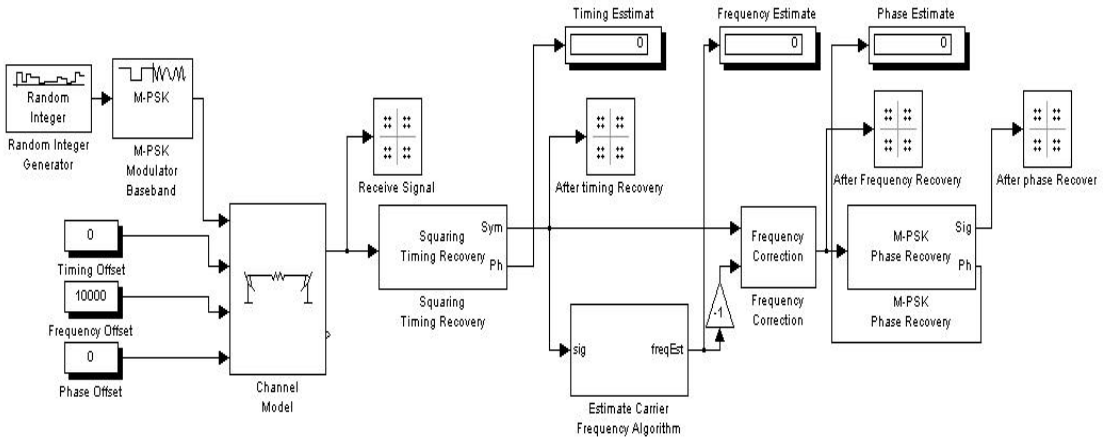
การเข้าจังหวะสัญญาณสามารถกระทำได้ทั้งทางอนาล็อกและทางดิจิตอล ซึ่งการเข้าจังหวะแบบดิจิตอลจะมีข้อดีทั้งด้านความยืดหยุ่น และค่าใช้จ่ายในการพัฒนาที่ถูกกว่า ประกอบกับการที่ผู้วิจัยได้ไปเห็นการออกแบบส่วนการเข้าจังหวะสำหรับดาวเทียมที่สถาบันวิจัย XISRT ณ ประเทศจีน และใช้จริงสำหรับระบบดาวเทียมทุกย่านความถี่ ใช้เทคโนโลยี FPGA/VHDL ซึ่งเป็นการพัฒนาทางด้านดิจิตอล (ในส่วนการพัฒนา Clock Recovery, Frequency Recovery และ AFC) และผู้วิจัยเองก็มีความเชี่ยวชาญในการพัฒนาทางด้านดิจิตอลมากกว่า จึงได้ตัดสินใจที่จะเลือกใช้การพัฒนาการเข้าจังหวะแบบดิจิตอล โดยการทำงานจะแบ่งออกเป็น 2 ส่วนใหญ่ๆ ก็คือการจำลองการเข้าจังหวะสัญญาณเพื่อหาค่าพารามิเตอร์ที่เหมาะสม (รวมถึงเทคนิคต่างๆ ที่จำเป็นต้องใช้) บนซอฟต์แวร์ MATLAB และการพัฒนาตัวเข้าจังหวะสัญญาณบนเทคโนโลยี FPGA/VHDL ก่อนที่จะกล่าวถึงรายละเอียดในการจำลองการเข้าจังหวะสัญญาณ จะกล่าวถึงลำดับในการเข้าจังหวะรวมถึงผลกระทบจากเข้าไม่เป็นจังหวะ

ในทางอุดมคติการประมาณจังหวะเวลาผิดพลาด (symbol timing errors), ความถี่ตกค้าง (carrier frequency offset) และเฟสตกค้าง (carrier phase offset) จะต้องนำมาใช้ในการประมาณร่วมพร้อมกัน แต่เนื่องจากความซับซ้อนในการจัดการ และการประประมวลผลสัญญาณอาจจะทำงานไม่ทันที่เวลาจริง ทำให้ในทางปฏิบัติส่วนใหญ่ การประมาณความถี่ (frequency estimation) ที่จำเป็นต้องรู้จังหวะเวลาข้อมูล (symbol timing) ส่วนการประมาณเฟส (phase estimation) ซึ่งต้องจำเป็นต้องรู้จังหวะเวลา (symbol timing) และความถี่ตกค้าง (frequency offset) ของข้อมูล ทำให้ไม่สามารถประมาณทั้งเฟสและความถี่พร้อมกันได้ ดังนั้นการเข้าจังหวะ (synchronization) ส่วนใหญ่จะเป็นไปตามลำดับดังนี้ คือ การเข้าจังหวะเวลาของข้อมูล (symbol timing), ความถี่ และเฟส ตามลำดับดังภาพที่ 20

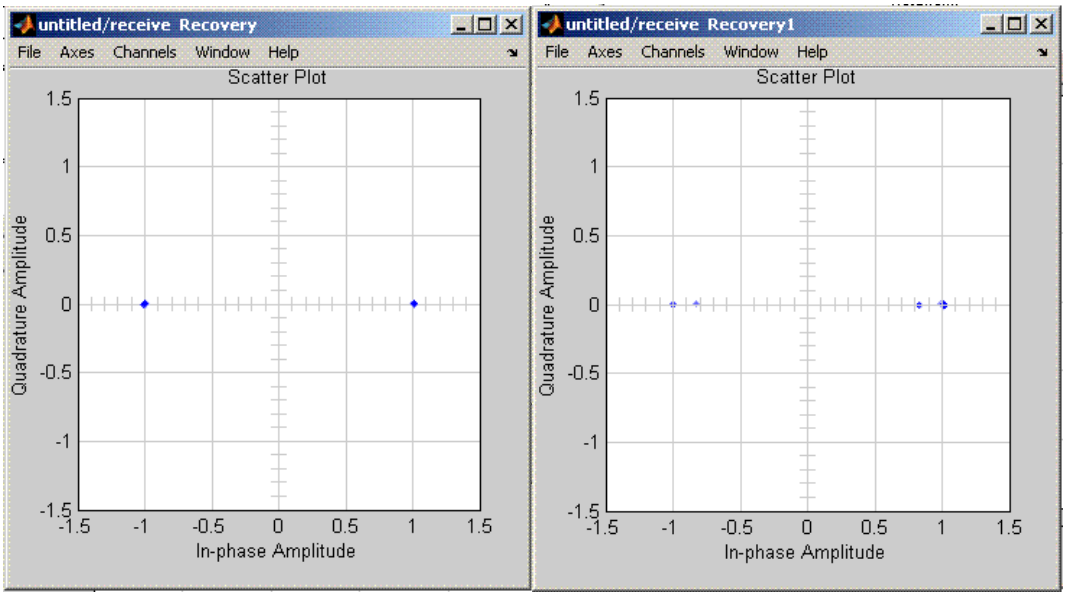


ภาพที่ 20 ลำดับการเข้าจังหวะข้อมูล

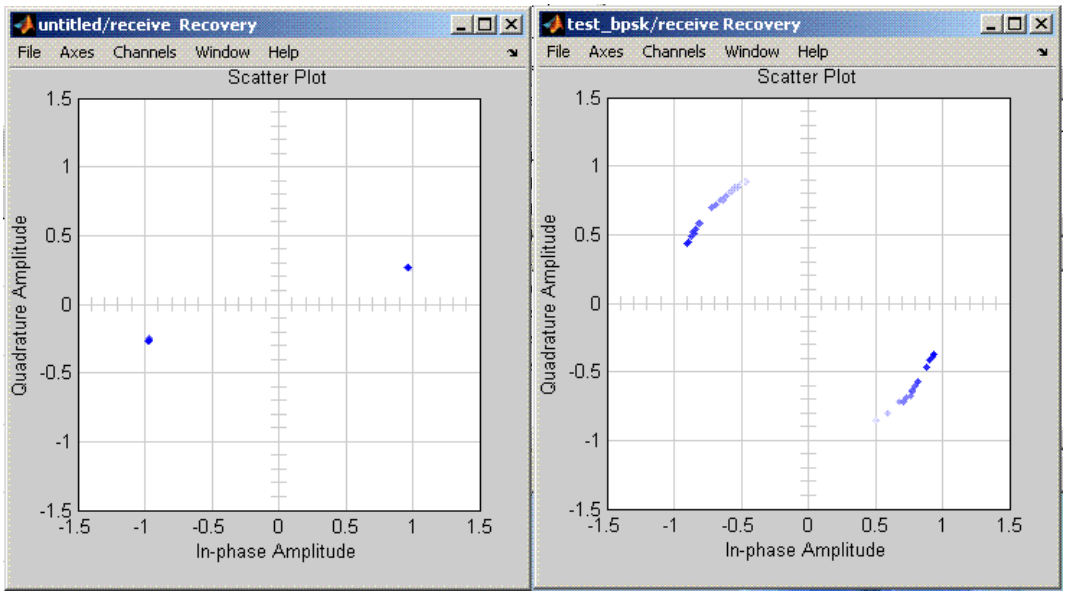
เบื้องต้นจะขอกล่าวถึงผลกระทบของการเข้าจังหวะที่ผิดพลาดในแต่ละส่วนก่อน เพื่อให้เห็นภาพกว้างว่าส่วนการแก้ไขเหล่านี้จะแก้ไขข้อมูลแบบไหนบ้าง โดยได้ทำการจำลองการทำงานของส่วนการเข้าจังหวะสัญญาณบนซอฟต์แวร์ MATLAB V 7.0 ตามภาพที่ 21 โดยปรับเปลี่ยนพารามิเตอร์ (Timing Offset, Frequency Offset, และ Phase Offset) ทีละค่าเพื่อตรวจสอบผลกระทบในส่วนต่างๆ โดยผลของการไม่เข้าจังหวะแบบต่างๆ จะมีผลให้สัญญาณที่ได้รับผิดพลาดในลักษณะที่แตกต่างกันไป สมมุติสัญญาณที่ใช้เป็นแบบ Binary Phase Shift Keying (BPSK) แสดงเป็น Signal constellation ดังรูปต่อไปนี้ ภาพที่ 22-1 คือภาพ Signal constellation ที่ได้รับที่เข้าจังหวะถูกต้องนั้นคือการที่กำหนดค่าพารามิเตอร์ทุกส่วนให้เป็นศูนย์ให้หมด ภาพที่ 22-2 เป็นผลจากการเข้าจังหวะเวลาผิดพลาด (symbol timing error) ซึ่งมีผลให้ชักข้อมูล (Sampling) ผิดตำแหน่ง (ไม่ได้ชักข้อมูลจากตำแหน่งที่ดีที่สุด) ภาพที่ 22-3 เป็นผลจากเฟสตกค้าง (phase offset) ทำให้ constellation ของสัญญาณ เลื่อนไปอยู่ผิดตำแหน่งนั้นก็คือการที่เฟสเปลี่ยนแปลงไปจากที่ควรจะเป็นทำให้สัญญาณที่ได้รับอยู่ผิดตำแหน่ง และภาพที่ 22 -4 เป็นผลจากความถี่ตกค้าง (frequency offset) ทำให้ constellation ของสัญญาณเกิดการหมุน ซึ่งก็คือ เฟสเปลี่ยนแปลงไปเมื่อเทียบกับเวลา



ภาพที่ 21 การจำลองการทำงานบน Software MATLAB



(1) Signal constellation เข้าจังหวะถูกต้อง (2) Signal constellation ที่จังหวะเวลาผิดพลาด



(3) Signal constellation ที่เฟสผิดพลาด (4) Signal constellation ที่ความถี่ผิดพลาด

ภาพที่ 22 ผลจากการเข้าจังหวะแบบต่างๆ

ในระบบสื่อสารดาวเทียมสัญญาณข้อมูลที่ได้รับจะถูกลดคุณภาพลงอันเนื่องมาจากหลายสาเหตุเช่น ผลของช่องสัญญาณ, ความถี่ตกค้างอันเนื่องมาจากภาคส่งและภาครับมีความถี่ไม่ตรงกันพอดี (เนื่องจากผลของความไม่แน่นอนของวงจรออสซิเลเตอร์), ความถี่ตกค้างจากผลของ Doppler เนื่องจากความเร็วสัมพัทธ์ระหว่างดาวเทียมและภาครับไม่เท่าเดิม โดยสามารถประมาณความถี่ Doppler ของสัญญาณที่ได้รับเป็น

$$f_d = f_c \frac{v(t)}{c} \cos(\alpha(t))$$

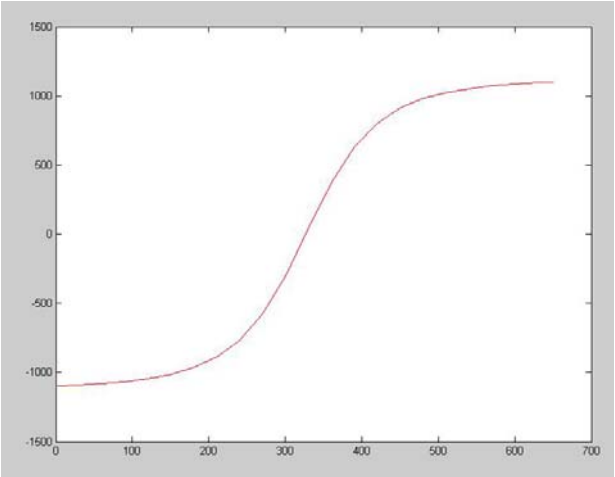
โดย v(t) คือ ความเร็วสัมพัทธ์ระหว่างดาวเทียมกับภาครับ,

จะเห็นได้ว่าผลของความถี่ Doppler ในระบบสื่อสารดาวเทียมนี้ จะทำให้สัญญาณที่ได้รับเกิดการเปลี่ยนแปลงของความถี่ Doppler ไปตลอดเวลาที่ดาวเทียมเคลื่อนที่ไป จึงทำให้ยิ่งยากต่อการเข้าจังหวะขึ้นไปอีก

โดยผลของความถี่ Doppler ทางจีนได้คำนวณไว้ในรายงาน “Preliminary design report for Ka Band experimental subsystem of small multi-mission satellite” ได้ผลการคำนวณความถี่ Doppler ไว้ดังตารางที่ 5 และสามารถวาดกราฟผลของความถี่ Doppler ได้ดังภาพที่ 23 ซึ่งทำให้เกิดความถี่ตกค้างสูงสุดประมาณ +/- 1.5 MHz

ตารางที่ 5 ผลการคำนวณความถี่ Doppler

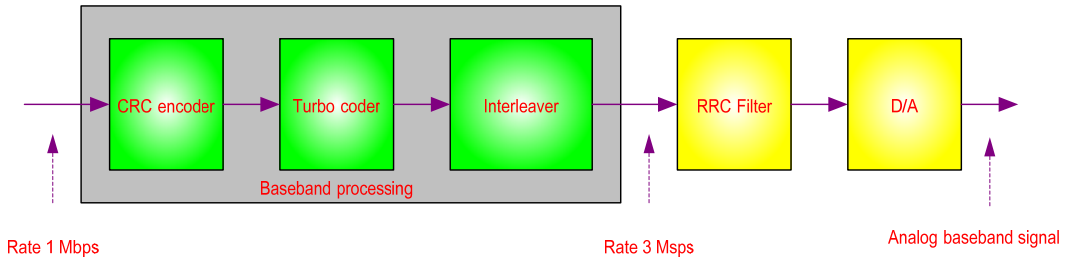
เวลา (s)	ผลรวมความถี่ Doppler (KHz)	เวลา (s)	ผลรวมความถี่ Doppler (KHz)
T0	-1097.383	T0+360s	374.466
T0+30s	-1091.052	T0+390s	631.664
T0+60s	-1084.346	T0+420s	802.457
T0+90s	-1066.910	T0+450s	910.386
T0+120s	-1045.566	T0+480s	978.769
T0+150s	-1013.735	T0+510s	1023.047
T0+180s	-965.383	T0+540s	1052.401
T0+210s	-890.195	T0+570s	1072.207
T0+240s	-771.033	T0+600s	1085.671
T0+270s	-583.242	T0+630s	1094.754
T0+300s	-306.471	T0+650.74s	1099.147
T0+330s	38.841		



ภาพที่ 23 ผลการประมาณความถี่ Doppler (KHz) เทียบกับเวลา (s)

แนวทางการสร้างวงจรการเข้าจังหวะนี้ในขั้นต้นจะนำไปจำลองการทำงานบน MATLAB เพื่อช่วยในการเลือกวงจรที่เหมาะสม และช่วยยืนยันความถูกต้องของวงจรที่สร้างว่าจะสามารถทำงานจริงได้ในระดับหนึ่งตามที่ได้กล่าวไป ซึ่งในขั้นต้นนี้จะใช้วงจรเข้าจังหวะเวลาข้อมูลแบบ Timing squaring phase ตัวประมาณความถี่ตกค้างจากเฟสที่หมุนไป (Phase increment) และตัวประมาณเฟสตกค้างแบบยกกำลัง M (M-power)

วงจรการเข้าจังหวะบางแบบจะมีความสัมพันธ์ไปถึงอัตราเร็วระดับ Symbol rate ของสัญญาณที่ได้รับ ดังนั้น จึงขอกล่าวถึงวงจรระดับเบสแบนด์เพื่ออ้างอิงความเร็วที่ใช้ในการส่งข้อมูล ภาพที่ 24 เป็นโครงสร้างวงจรเข้ารหัสเบสแบนด์ โดยความเร็วระดับข้อมูลดิบ ประมาณ 1 Mbps โดยผ่านการเข้ารหัส CRC, 1/3 Turbo encode และ Interleave อัตราเร็วข้อมูลจะเพิ่มขึ้นประมาณ 3 เท่า จะได้อัตราเร็วข้อมูลระดับ Symbol rate ประมาณ 3 Msps (Mega symbol per second) จากนั้นจึงถูกส่งไปสร้างรูปคลื่นสัญญาณที่ RRC filter และแปลงเป็นสัญญาณอนาล็อก แล้วจึงนำไปมอดูเลตต่อไป

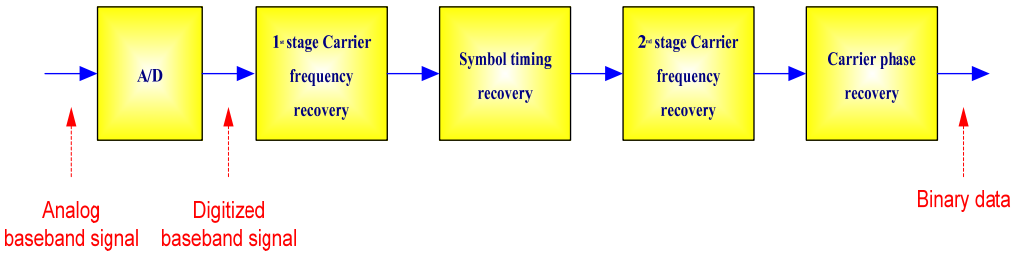


ภาพที่ 24 โครงสร้างส่วนประมวลผลเบสแบนด์

โดยตัวประมาณความถี่ตกค้างจากเฟสที่หมุนไปที่เลือกใช้มีข้อจำกัด คือ สามารถประมาณความถี่ตกค้างได้ไม่เกิน $\Delta f \le \frac{f_s}{2M}$ เมื่อ M แทนค่ามอดูเลต, f_s คือ ความถี่ของข้อมูล (Symbol rate) ประมาณ 3 Msps (ดังที่อธิบายในย่อหน้าที่ผ่านมา) โดยถูกมอเลตแบบ BPSK หรือ 2-PSK (M=2) ดังนั้นตัวประมาณความถี่ที่เลือกใช้จะสามารถประมาณความถี่ตกค้างได้ 0.75 MHz เท่านั้น แต่จากหัวข้อที่แล้วผลการคำนวณความถี่ตกค้างจากผล Doppler ที่ทางจีนคำนวณมาทำให้ความถี่เลื่อนไปถึง +/- 1.5 MHz ซึ่งเกินช่วงที่ตัวประมาณความถี่แบบที่เลือกสามารถทำงานได้อย่างถูกต้อง ทางแก้ไขที่เป็นไปได้มี 2 แนวทางคือ

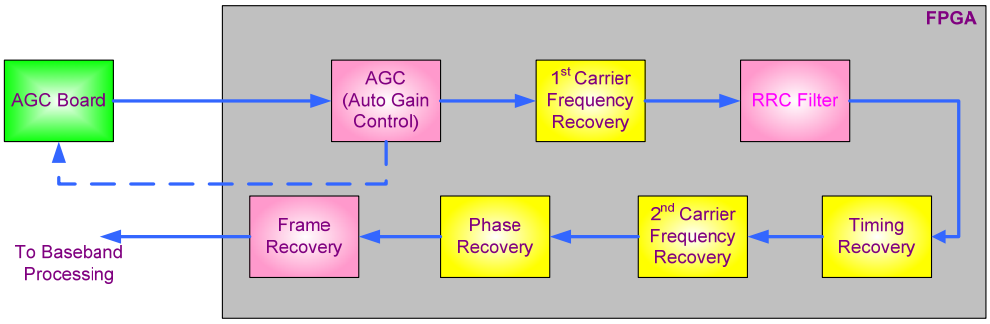
- เพิ่มอัตราเร็วข้อมูลที่ส่งมากขึ้น เพื่อทำให้สามารถเข้าจังหวะได้ทัน ซึ่งการเพิ่มความเร็วในการส่งข้อมูลก็ส่งผลกระทบต่อส่วนอื่น และในกรณีที่เกิดความถี่ตกค้างมากๆ จะมีผลให้การเข้าจังหวะเวลากับข้อมูลผิดพลาดไปด้วยทำให้ได้ผลการประมาณความถี่ผิดพลาดต่อไปอีก

- ใช้ตัวประมาณความถี่ 2 ชั้น คือ เพิ่มตัวประมาณความถี่แบบหยาบ เพื่อมาแก้ผลของความถี่ผิดพลาดไปก่อนในระดับหนึ่ง ซึ่งลักษณะวิธีการที่จะใช้ควรเป็นแบบที่ประมาณความถี่แบบไม่จำเป็นต้องรู้ผลของจังหวะเวลา ซึ่งจะช่วยต่อการเข้าจังหวะเวลาของข้อมูลได้ด้วย ดังนั้นจึงเลือกที่จะพัฒนาในแนวทางนี้ โดยมีลำดับการทำงานตามภาพที่ 25 และได้เลือกที่จะใช้การประมาณ FFT มาช่วยในการประมาณความถี่



ภาพที่ 25 โครงสร้างรวมวงจรเข้าจังหวะแบบตัวประมาณความถี่ตกค้าง 2 ชั้น

แนวคิดเบื้องต้นที่ได้นำเสนอไปนั้น ได้มีการนำแนวคิดนี้ไปออกแบบ และเลือกแนวทางที่เหมาะสมสำหรับจำลองการทำงานในซอฟต์แวร์ MATLAB V7.0 รวมถึงนำไปพัฒนาลงบนชิบ FPGA และหลังจากที่ได้พัฒนาด้วยเทคโนโลยี VHDL/FPGA แล้วพบว่าจำเป็นจะต้องแก้ไขวงจรบางส่วนให้เหมาะสมกับการประมวลผลสัญญาณ และแก้ไขให้ข้อมูลมีความผิดพลาดให้น้อยที่สุดตามภาพที่ 26 (ส่วนที่เพิ่มจะเป็นสีชมพู มี 3 ฟังก์ชั่น)



ภาพที่ 26 ส่วนการเข้าจังหวะสัญญาณ

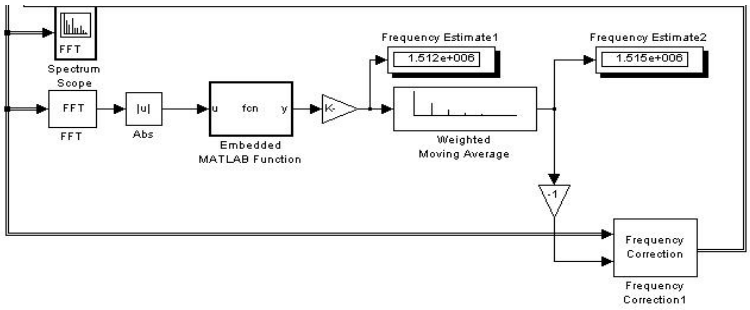
โดยการปรับปรุงแนวคิดเดิม เพิ่มส่วน Auto Gain Control เพื่อช่วยให้ข้อมูลที่เข้ามานั้นมีระดับที่คงที่ คือไม่สูงหรือต่ำจนเกินไปจน ADC (Analog to Digital Converter) ทำงานผิดพลาด นั้นคือเป็นการควบคุมสัญญาณที่เข้ามาไม่ให้สูงหรือต่ำจน ADC ทำงานผิดพลาดนั้นเอง และอีกส่วนวงจรนึงคือส่วนการกรองสัญญาณ RRC Filter เพื่อลดความผิดพลาดที่เกิดขึ้นจากช่องสัญญาณ โดยหลังจากที่ได้พัฒนาแล้วพบว่าในการกรองสัญญาณนี้ช่วยให้ส่วนการเข้าจังหวะสามารถทำงานได้ถูกต้องมากขึ้น คือจำนวนความผิดพลาดลดน้อยลงไปมาก สุดท้ายจะเป็นการแก้ไขความคลุมเครือของเฟสที่จะเกิดขึ้นหลังจากการเข้าจังหวะสัญญาณต่างๆ ที่ได้กล่าวมา ซึ่งจะใช้เทคนิก Correlation เพื่อหาช่วงของข้อมูลที่เป็นต้นเฟรมข้อมูลและตรวจสอบว่าอยู่ที่เฟสใด (ในที่นี้ใช้การมอดูเลตแบบ BPSK ดังนั้นจะมีอยู่ 2 เฟสคือ 0 กับ 180 องศา)

สำหรับส่วนอื่นๆ จะใช้แนวคิดเดิมที่เคยเสนอไว้ ดังนี้ การประมาณความถี่แบบหยาบจะใช้วงจร FFT เพื่อช่วยการประมาณความถี่แบบละเอียดมีความถูกต้องมากขึ้น การประมาณเวลาจะใช้แนวคิด Timing Square Phase โดยจะช่วยหาสัญญาณที่แรงที่สุดในแต่ละ Symbol การประมาณความถี่แบบละเอียด (ช่วยแก้ Doppler Frequency) และการประมาณเฟสตกค้าง (แก้เฟสที่อาจจะมีการตกค้างที่เกิดจากช่องสัญญาณ) จะใช้วิธีการ M-Power และจะมีรายละเอียดดังต่อไปนี้

2.3.1 ส่วนการประมาณความถี่แบบหยาบ (Rough Frequency Recovery)

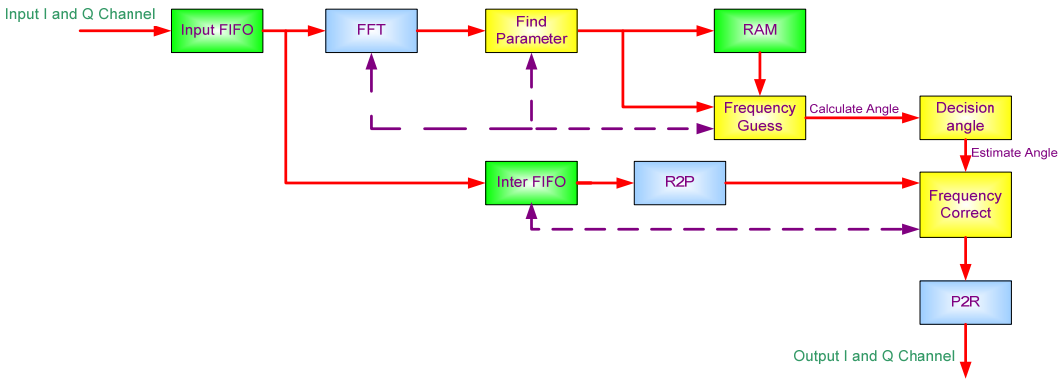
การประมาณความถี่แบบหยาบนี้จะใช้เทคนิค FFT (Fast Fourier Transform) ซึ่งจะสามารถประมาณความถี่ได้ระหว่าง $-\frac{f_s}{2}$ ถึง $\frac{f_s}{2}$ โดยค่า f_s จะมีค่าเท่ากับ 12 MHz (จากการคำนวณในครั้งก่อน) ดังนั้นการประมาณความถี่จะทำได้ไม่เกิน $\pm 6MHz$ เลยทีเดียว และถือว่าการ

ประมาณความถี่นี้จะช่วยให้การประมาณความถี่แบบละเอียดปรับอีกเล็กน้อยเท่านั้นก็จะได้ข้อมูลที่มีความผิดพลาดน้อยที่สุด



ภาพที่ 27 การจำลองการทำงาน FFT บนซอฟต์แวร์ MATLAB V7.0

ซึ่งหลังจากการจำลองการทำงานในครั้งก่อนตามภาพที่ 27 จะพบว่าส่วนที่มีผลกับความแม่นยำของการประมาณความถี่จะขึ้นอยู่กับจำนวนข้อมูลใน FFT ว่าจะให้มีขนาดเป็นเท่าไร (ยิ่งสูงยิ่งดี) แต่ก็ต้องแลกกับความซับซ้อน และการใช้ทรัพยากรภายในที่สูงขึ้นเป็นเงาตามตัว โดยในเบื้องต้นก็ได้เลือกที่จะใช้ 1024 จุดในการทำงานของ FFT และอีกส่วนนึงที่มีผลกับการประมาณความถี่ ก็คือการเลือกค่าที่จะมาปรับความถี่ในแต่ละรอบการทำงาน และโดยรวมจะต้องไม่ให้ความถี่ที่คำนวณได้แตกต่างกันมากจนเกินไป ซึ่งถ้าแตกต่างกันมากจนเกินไปก็จะทำให้เกิดความผิดพลาดขึ้นมาได้ ทั้งๆ ที่ข้อมูลในช่วงนั้นไม่ผิดพลาดเลยก็ได้ ดังนั้นจึงเป็นส่วนสำคัญอย่างยิ่งในการพัฒนา จึงทำให้จะต้องมีการประมาณกันถึง 2 รอบ ก่อนที่จะทำการปรับความถี่ โดยรอบแรกก็จะได้มาจากการคำนวณในแต่ละรอบ และรอบที่ 2 จะเป็นการประมาณที่ได้จากการคำนวณในแต่ละรอบมาปรับไม่ให้ค่าเปลี่ยนแปลงมากจนเกินไป (ในที่นี้ใช้การประมาณรอบที่สอง 8 ค่า) โดยการพัฒนาจะเป็นไปตามภาพที่ 28

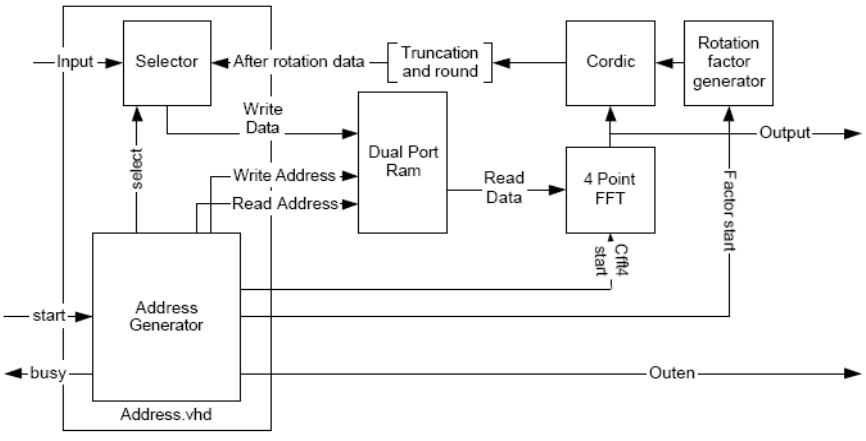


ภาพที่ 28 โครงสร้างการประมาณความถี่แบบหยาบ

โดยหลักการแล้วจะทำการคำนวณ FFT เพื่อนำไปหาค่าความถี่ โดยการหาค่าความถี่จะทำการหาได้จากค่าสูงสุดที่เกิดขึ้นของข้อมูล แล้วนำไปประมาณหาค่าที่ขอบซ้าย และขอบขวาของข้อมูลที่ออกมาจาก FFT หลังจากนั้นก็จะเฉลี่ยค่าที่ได้ทั้ง 2 ค่า เพื่อเป็นความถี่ที่ควรจะปรับ แต่อย่างที่ได้กล่าวไปแล้วนั้นการใช้ความถี่ที่คำนวณได้อาจจะทำให้เกิดข้อผิดพลาดขึ้นมาได้ จึงจำเป็นต้องมีการประมาณความถี่อีกชั้นนึงก่อน ด้วยการหาค่าเฉลี่ยของความถี่ที่เข้ามา เพื่อไม่ให้เกิดความผิดพลาดในการประมาณของข้อมูลที่ติดกันมากจนเกินไป แต่วิธีนี้ถ้าใช้จำนวนที่มาเฉลี่ยมากเกินไป อาจจะเกิดกรณีที่ค่าแรกๆ อาจจะเกิดความผิดพลาดหรือประมาณผิดพลาดมากและค่าที่เข้ามาต่อๆ ไป ประมาณได้ถูกต้อง แต่ต้องมาเฉลี่ย ทำให้ค่าในช่วงต้นๆ เกิดการประมาณที่ผิดพลาดขึ้นมาได้ถึงแม้ว่าจะไม่สูงมาก ซึ่งจำเป็นต้องเลือกจำนวนที่จะเฉลี่ยไม่มากและไม่น้อยจนเกินไป โดยในเบื้องต้นได้เลือกใช้ 8 ค่ามาเฉลี่ยกัน ซึ่งข้อเสียที่เกิดขึ้นนั้นจะสามารถไปแก้ไขได้ด้วยการประมาณความถี่แบบละเอียดอีกทีนึง

เพื่อให้เกิดความรวดเร็วในการทำงานฟังก์ชั่นบางส่วนจะนำมาจาก Open Core ซึ่งมีอยู่แล้วและไม่เสียค่าใช้จ่ายสำหรับค่าลิขสิทธิ์ เช่น R2P หรือ P2R ซึ่งเป็น CORDIC แปลงจากระบบ Rectangular เป็นระบบ Polar (ขนาดกับมุมที่หมุนไป) หรือกลับกันก็ได้ ซึ่งสามารถหาอ่านในรายละเอียดได้จาก (Cordic Open Core Specification) และอีกฟังก์ชั่นนึงคือ FFT (Fast Fourier Transform) โดยได้เลือกใช้ 1024 จุด ในเบื้องต้น และรายละเอียดการทำงานภายในจะเป็นไปตามภาพที่ 29

ในส่วนฟังก์ชั่นการทำงานส่วนอื่นๆ ก็จะเป็นฟังก์ชั่นมาตรฐานที่ใช้ในการจัดการในเรื่องหน่วยความจำ นั้นก็คือ FIFO หรือ RAM และสุดท้ายฟังก์ชั่นการคำนวณต่างๆ ที่พัฒนาขึ้นมาเองเพื่อรองรับการทำงานสำหรับส่วนการประมาณความถี่แบบหยาบนั้นเอง



ภาพที่ 29 โครงสร้างภายใน FFT

การประมาณความถี่แบบหยาบนี้เป็นการประมาณความถี่เพื่อให้ความผิดพลาดที่เกิดขึ้นจากความถี่ที่ Doppler ไปนั้นลดลง หรือประมาณให้การประมาณความถี่แบบละเอียดนั้นสามารถที่จะประมาณความถี่ได้ เนื่องจากแนวทางของการประมาณความถี่แบบละเอียดไม่สามารถที่จะประมาณความถี่ Doppler ได้ถึง 1.5 MHz แต่สูงสุดของวงจรประมาณความถี่แบบละเอียดทำได้แค่ 750 MHz เท่านั้นในทางทฤษฎี ดังนั้นจึงมีความจำเป็นต้องสร้างการประมาณความถี่แบบหยาบเข้ามาช่วยนั้นเอง ซึ่ง แนวทางนี้จะช่วยขยายการประมาณความผิดพลาดอันเนื่องมาจาก Doppler Frequency ได้สูงถึง 6 MHz ตามที่ได้กล่าวไปแล้วนั้นเอง

และสุดท้ายทรัพยากรที่ใช้จะใช้ไป 2198 Slices หรือ 4946 Logic Cells ใช้ RAM ไป 18 KB และความถี่สูงสุดของวงจรที่สามารถทำงานได้คือ 73.73 MHz

2.3.2 ส่วนการกรองสัญญาณแบบ Root Raise Cosine (RRC Filter)

ส่วนการกรองสัญญาณนี้เป็นส่วนที่เพิ่มเข้ามาเพื่อช่วยในการกรองสัญญาณความถี่สูงออก ซึ่งจะทำให้ข้อมูลลดความผิดพลาดลงไป และเหตุผลสำคัญที่นำมาวางไว้ระหว่างการประมาณความถี่แบบหยาบ (Rough Frequency Recovery) กับการเข้าจังหวะเวลา (Timing Recovery) นั้น เนื่องจากการที่จะนำไปวางไว้ในต่ำแหน่งก่อนที่จะประมาณความถี่แบบหยาบ จะทำให้สัญญาณนั้นถูกลดทอนออกไปในช่องความถี่สูง ซึ่งจะส่งผลให้การประมาณนั้นผิดพลาดจึงต้องนำมาวางไว้หลังการประมาณความถี่นี้ และในเหตุผลในการวางไว้ก่อนหน้าการเข้าจังหวะเวลาเนื่องจากการเข้าจังหวะเวลาเป็นการเข้าจังหวะเพื่อหาสัญญาณที่แรงที่สุดในแต่ละ Symbol ดังนั้นถ้าสัญญาณที่เข้ามามีความผิดพลาดอาจจะทำให้การเลือกสัญญาณมีความผิดพลาดขึ้นมาได้ ดังนั้นเพื่อเป็นการช่วยฟังก์ชั่นการเข้าจังหวะเวลาให้สามารถทำงานได้ถูกต้องยิ่งขึ้น จึงมีความจำเป็นอย่างยิ่งยวดที่จะกรองความผิดพลาดออกไปบางส่วน ถึงแม้ว่าจะมีความผิดพลาดอยู่บ้าง แต่ก็ยังดีกว่าที่จะไม่ได้จัดการความผิดพลาดเลย และเทคนิกที่นิยมใช้ก็จะเป็นพวกตัวกรองสัญญาณความถี่ต่ำ (Lowpass Filter)

และจากการออกแบบระบบต้นแบบส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ที่ได้พัฒนาขึ้นมานั้น ได้เลือกที่จะใช้เทคนิก RRC Filter ซึ่งเป็นที่นิยมในทางปฏิบัติอย่างแพร่หลายในกระบวนการช่วยการทำการเข้าจังหวะสัญญาณ และง่ายต่อการพัฒนาในเชิงปฏิบัติ ดังนั้นพารามิเตอร์ที่สำคัญของการสร้างตัวกรองสัญญาณนี้มีอยู่ 2 ค่าคือ ค่า Rolloff Factor (α) และ

จำนวนของค่าสัมประสิทธิ์ (N) ซึ่งจะสัมพันธ์กับความกว้างของช่องสัญญาณ (Bandwidth) และค่าประสิทธิ์ภาพ หรือ BER Performance โดยจะมีความสัมพันธ์กันดังนี้

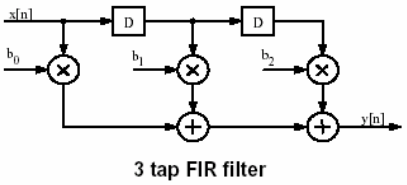
Bandwidth required $\uparrow$ as $\alpha \downarrow$ BER performance sensitivity $\uparrow$ as $\alpha \downarrow$

Bandwidth required $\downarrow$ as N $\uparrow$ BER performance $\uparrow$ as N $\uparrow$

เมื่อได้ศึกษาและค้นคว้าต่างๆ พบว่าระบบการสื่อสารผ่านดาวเทียมส่วนใหญ่แล้ว (อ้างอิงตามมาตรฐาน CCSDS และระบบดาวเทียมทั่วไป) พบว่าจะใช้ค่า Rolloff Factor ประมาณ 0.35-0.4 และจำนวนสัมประสิทธ์ จะใช้อยู่ที่ประมาณ 30-64 ค่า โดยในเบื้องต้นนี้ได้เลือกที่จะผู้วิจัยได้เลือกที่จะใช้ค่า α = 0.35 และค่า N = 33 ตามที่ได้จำลองไว้บนซอฟต์แวร์ MATLAB V7.0 และหลักการในการสร้างตัวกรองสัญญาณจะใช้ตัว MAC (Multiply and Accumulate) ซึ่งทำงานตามสมการดังนี้

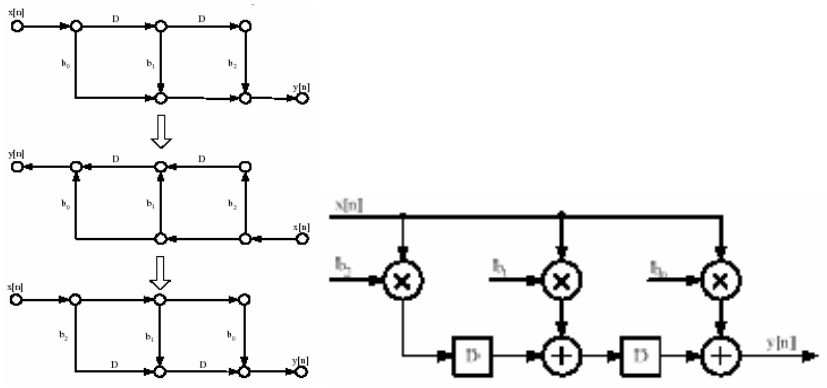
$$y[n] = \sum_{k=0}^{M-1} b_k \cdot x[n-k]$$

โดย b_k แทนค่าสัมประสิทธิ์ (Coefficient) ของ RRC filter และ x[n] คือ อินพุตที่รับเข้ามา ซึ่งสามารถแปลงเป็นวงจรดังภาพที่ 30 ซึ่งมี Time delay ใน Critical path (t_c) = (M-1) * $t_{add} + t_{mult}$



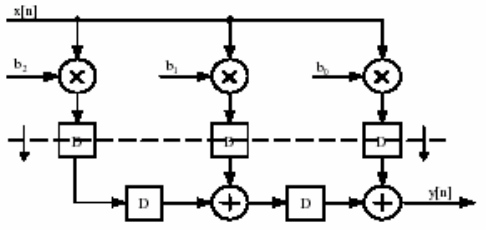
ภาพที่ 30 วงจร RRC filter แบบทั่วๆไป

เมื่อนำมาแปลงวงจรเป็น Signal Flow Graphs (SFG) และนำ (SFG) มาย้ายตำแหน่ง (Transposition of SFGs) แล้วได้ผลลัพธ์การแปลงแสดงดังภาพที่ 31



ภาพที่ 31 Transposition of SFGs

จากผลการแปลง SFGs จะเห็นว่า $t_c = t_{add} + t_{mult}$ จะเห็นว่า delay time ลดลงอย่างมาก ซึ่งสามารถลด delay time ลงได้อีก เมื่อเราแทรกรีจิสเตอร์ไว้ เพื่อพักค่าผลคูณก่อน แสดงวงจรดังภาพที่ 31 จะได้ $t_c = t_{mult}$



ภาพที่ 32 วงจร RRC filter เมื่อถูกแปลง SFG และเพิ่มรีจิสเตอร์

ซึ่งในภาพที่ 32 เป็นวงจรที่จะนำมาใช้ในการสร้างวงจร RRC filter โดยลักษณะของภาครับที่ไม่จำเป็นต้องทำการ Down Sampling ลงมาเนื่องจากข้อมูลทุกค่าจะถูกนำไปตัดสินใจในฟังก์ชั่นการเข้าจังหวะเวลา โดยการทำ RRC Filter นี้จะเป็นแค่การจำกัดช่วงของข้อมูลให้มีความผิดพลาดน้อยลงนั้นเอง และเมื่อความผิดพลาดของข้อมูลน้อยลง การเข้าจังหวะเวลาก็จะทำงานได้ถูกต้องมากขึ้นเป็นเงาตามตัวนั้นเอง และค่าสัมประสิทธิ์ที่ใช้ในการพัฒนาจะเป็นไปตามตารางที่ 6

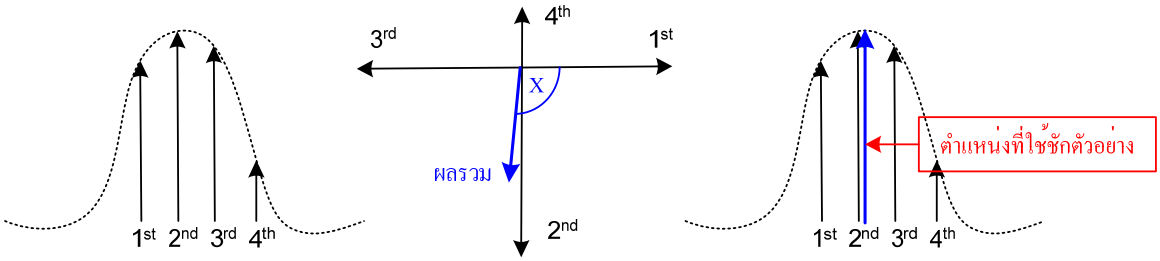
ตารางที่ 6 ตารางค่าสัมประสิทธิ์ที่ใช้ในการพัฒนาวงจร RRC Filter

ตำแหน่งค่าสัมประสิทธ์		ค่าสัมประสิทธ์
0	32	0
1	31	0
2	30	0
3	29	0
4	28	-1
5	27	0
6	26	1
7	25	2
8	24	2
9	23	-1
10	22	-4
11	21	-5
12	20	-2
13	19	6
14	18	17
15	17	27
16		31

และทรัพยากรที่ใช้สำหรับการพัฒนาตัวกรองสัญญาณนี้จะใช้ไป 302 Slices หรือ 680 Logic cells หน่วยความจำใช้ไป 3 KB และความถี่สูงสุดที่สามารถทำงานได้เท่ากับ 158.94 MHz

2.3.3 การเข้าจังหวะเวลา (Timing Recovery)

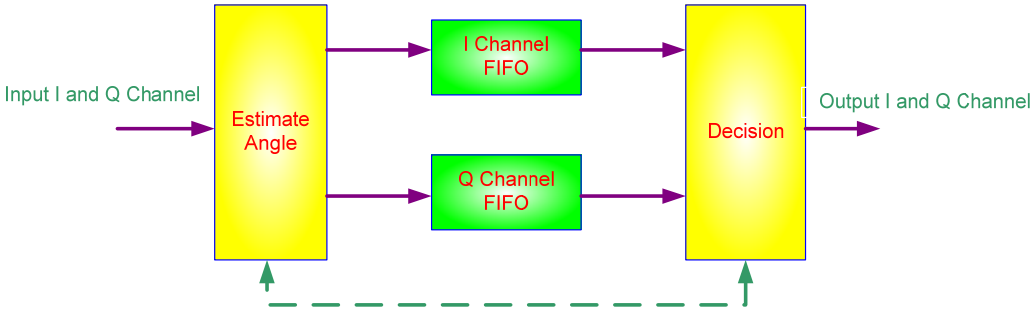
การเข้าจังหวะเวลานี้จะเลือกใช้แบบ Timing Square Phase ซึ่งเป็นการนำสัญญาณใน 1 Symbol มาหมุนไปตามแกน และรวมผลเพื่อหามุมที่คาดว่าจะมีสัญญาณแรงที่สุด โดยมุมที่ได้จะนำไประบุตำแหน่งของสัญญาณอีกทีนึง โดยแนวคิดการเข้าจังหวะจะเป็นไปตามภาพที่ 33



ภาพที่ 33 ขั้นตอนในการเข้าจังหวะเวลา

โดยในทางปฏิบัติแล้วนั้นการที่จะระบุตำแหน่งในแต่ละครั้งควรนำผลประมาณตำแหน่งข้อมูลจากหลายๆ Symbol มาเฉลี่ยกันเพื่อใช้ประมาณตำแหน่งในการชักตัวอย่างที่ดีที่สุด กันความผิดพลาดเนื่องจากการ Re-Sampling และ การเฉลี่ยสัญญาณไม่ละเอียดพอ จะเป็นไปตามตัวอย่างดังนี้

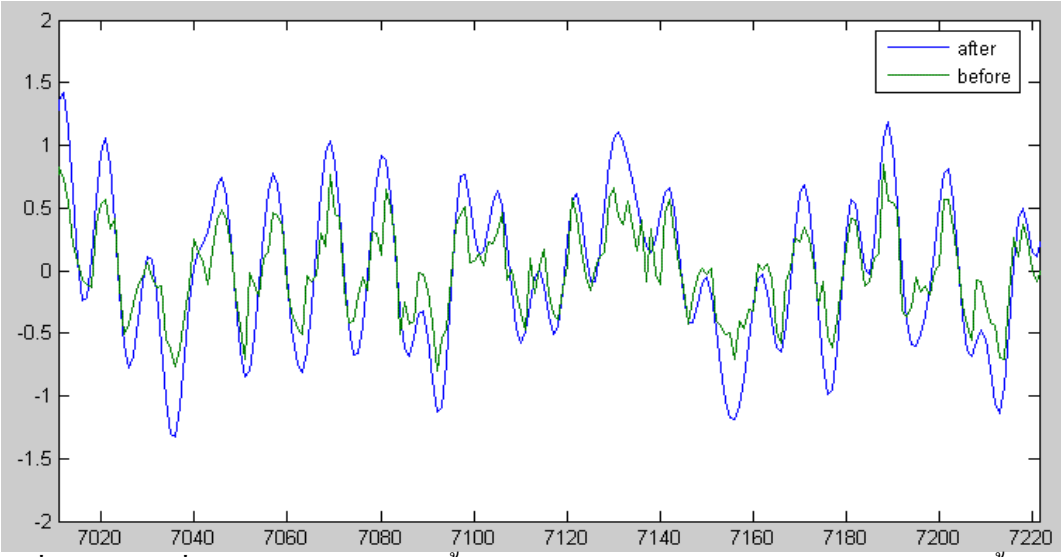
ในกรณีที่รับข้อมูลแบบ 4 samples/symbol จะนำข้อมูลทั้ง 4 หมุนไปอยู่ที่มุมต่างๆ 0, 270, 180 และ 90 องศา ดังภาพที่ 33 (กลาง) จากนั้นนำสัญญาณไปรวมกัน จะได้เวกเตอร์ผลรวม (เวกเตอร์เส้นสีน้ำเงินในภาพที่ 33 (กลาง)) ซึ่งสามารถนำมุมของเวกเตอร์ที่ได้ (X) มาระบุตำแน่งข้อมูลที่คาดว่าดีที่สุดในการชักตัวอย่างข้อมูลได้ ดังแสดงในภาพที่ 33 (ขวา)



ภาพที่ 34 โครงสร้างการทำงานฟังก์ชั่น Timing Recovery

จากภาพที่ 34 จะเห็นได้ว่าการทำงานจะแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ การประมาณมุมเพื่อหาตำแหน่งที่ดีที่สุด และการตัดสินใจเลือกข้อมูล โดยการประมาณมุมจะเป็นไปตามแนวคิดที่ได้กล่าวไปในข้างต้น ซึ่งจะมีการเรียกใช้ฟังก์ชั่น R2P (Rectangular to Polar) ที่เป็น Open Core ที่ได้กล่าวเอาไว้ในหัวข้อการประมาณความถี่แบบหยาบ เพื่อช่วยในการหามุม ส่วนการตัดสินใจเลือกข้อมูลที่ออกมานั้นจะใช้การประมาณหาข้อมูลที่ใกล้เคียงมากที่สุด เพื่อให้ง่ายต่อการพัฒนา รวมถึงไม่ซับซ้อนและใช้ทรัพยากรน้อยเท่าที่จะเป็นไปได้ ซึ่งการประมาณนี้จะใช้ถึง 16 ระดับ ถือว่ามากพอสมควร เหมาะสมกับการตัดสินใจ

และเพื่อให้เห็นภาพที่ได้กล่าวไว้ในหัวข้อที่ผ่านมาเกี่ยวกับความจำเป็นที่จะต้องมีการกรองสัญญาณก่อนเข้า การเข้าจังหวะเวลา จะเห็นได้ว่าถ้าอินพุทที่เข้ามาผิดเพื้อนไป การประมาณมุมก็จะผิดพลาดไป โดยบางค่าก็อาจจะไม่ใช่ค่าที่ดีที่สุดในการประมาณ แต่ถ้ามีส่วนตัวกรองนี้อินพุทที่เข้าก็จะมีความต่อเนื่องหรืออาจจะมีความผิดพลาดน้อยลง ทำให้การประมาณที่ออกมามีความถูกต้องมาก ดังที่จะเห็นได้ตามภาพที่ 35 โดยเส้นสีเขียวจะเป็นอินพุทที่ไม่ได้ผ่านตัวกรองสัญญาณ จะเห็นได้ว่าสัญญาณมีความผิดเพื้อนมากกว่าเส้นสีน้ำเงินซึ่งผ่านตัวกรองสัญญาณแล้ว ดังนั้นความผิดพลาดที่เกิดขึ้นก็จะลดน้อยลงในกรณีที่ผ่านตัวกรองสัญญาณก่อนเข้าการเข้าจังหวะเวลานี้



ภาพที่ 35 อินพุทที่เข้าการเข้าจังหวะเวลาทั้งแบบไม่มี Filter (สีเขียว) และมี RRC Filter (สีน้ำเงิน)

และทรัพยากรที่ใช้จะมีดังนี้คือ จำนวนเซลล์ที่ใช้ 552 Slices หรือ 1242 Logic cells หน่วยความจำใช้ไปทั้งหมด 5 KB และความถี่สูงสุดที่สามารถทำงานได้คือ 173.93 MHz

2.3.4 การประมาณความถี่แบบละเอียด (Delicate Frequency Recovery)

การประมาณความถี่แบบละเอียดนี้จะช่วยแก้ไขความถี่เลื่อนที่เกิดขึ้นซึ่งได้ทำการจำกัดให้อยู่ในช่วงที่แคบลงจากความถี่ที่เลื่อนมาแล้วจากการประมาณความถี่แบบหยาบ ดังนั้นการประมาณความถี่ในขั้นตอนนี้จะมีการคำนวณที่แม่นยำมากกว่าเดิม โดยแนวคิดในการหาตัวประมาณความถี่ตกค้างจากเฟสที่หมุนไป ซึ่งจะมีข้อจำกัดอยู่ที่ $\Delta f \le \frac{f_s}{2M}$ หรือประมาณ 750 kHz และข้อเสียของแนวคิดนี้ซึ่งก็คือ ถ้าเฟสที่เปลี่ยนมีขนาดใหญ่เกินไป (จนทำตัดสินตำแหน่งการหมุน

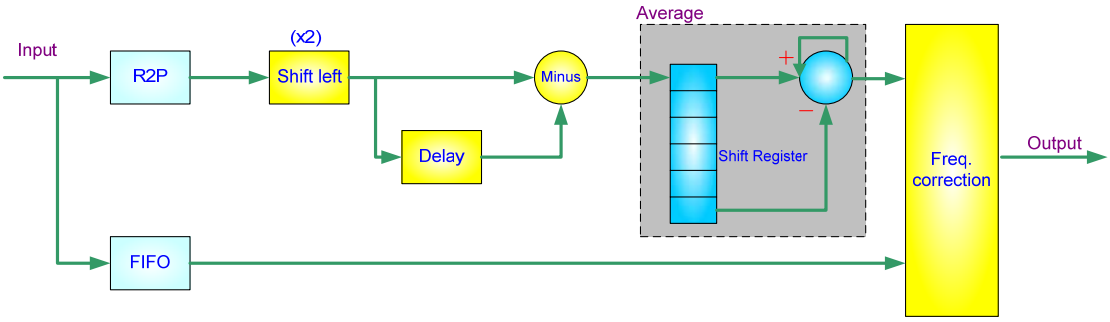
ผิดพลาด) จะมีผลให้การประมาณความถี่ตกค้างผิดพลาดได้ ข้อเสียนี้ได้ถูกตัวประมาณความถี่แบบหยาบกำจัดทิ้งไปในทันที เนื่องจากตัวประมาณความถี่แบบหยาบจะช่วยให้เฟสเปลี่ยนแปลงไปไม่มากคือมีขนาดไม่ใหญ่จนเกินไป ทำให้การทำงานในส่วนนี้มีความถูกต้องมาก และการทำงานในส่วนต่างๆ จะเป็นไปดังนี้

1. นำสัญญาณที่ได้รับไปยกกำลัง M (M-power) เพื่อกำจัดการมอดูเลตออกไป ซึ่งจะข้อมูลส่งแบบ BPSK หรือ 2-PSK (M=2) โดยจะมองสัญญาณที่ได้รับให้อยู่ในรูปแบบเชิงขั้ว $Re^{j\theta}$, หรือ $R\angle\theta$ (สัญญาณ BPSK จะอยู่ในรูป $R\angle 0$ หรือ $R\angle 180$) ดังนั้นเมื่อนำสัญญาณเหล่านี้ไปยกกำลัง 2 มุมก็เปลี่ยนเป็น 2 เท่าของมุมเดิม จะได้ผลลัพธ์ คือ สัญญาณทั้งหมดจะหมุนมาอยู่ในตำแหน่งที่ 0 องศา ซึ่งก็คือ การกำจัดการมอดูเลต นั่นเอง

2. นำผลที่ได้ไปคำนวณหาเฟสที่เปลี่ยนไประหว่างสัญญาณตัวนี้กับตัวที่แล้ว และหารเฟสที่ได้ด้วย 2 (กำจัดผลจากการยกกำลัง M=2 ในข้อ 1 ออกไป) แล้วนำไปเทียบกับเวลา ซึ่งก็จะได้ผลเป็นมุมที่เปลี่ยนไปต่อเวลา ซึ่งก็คือ ความถี่ $f = \frac{d\theta}{dt}$ ซึ่งควรเฉลี่ยผลของความถี่ตกค้างจากหลายๆ Symbol เพื่อใช้ในการประมาณและนำผลความถี่ตกค้างที่ได้ไปหมุนสัญญาณกลับเพื่อชดเชยหรือหักล้างผลของความถี่ตกค้าง

โดยการประมาณความถี่แบบละเอียดจะมีสมมติฐานที่สำคัญคือ การทำงานในส่วนการเข้าจังหวะเวลาจะต้องมีความถูกต้องแม่นยำ ดังนั้นเอาพุทที่ออกมาจากการเข้าจังหวะเวลาจำเป็นต้องถูกต้องในส่วนใหญ่ จึงจะทำให้ตัวประมาณความถี่แบบละเอียดสามารถทำงานได้ถูกต้องแน่นอน แต่ในความเป็นจริงจะพบว่าข้อมูลที่ส่งเข้ามาในการประมาณนี้ไม่ได้ถูกต้องทั้งหมด ดังนั้นการประมาณความถี่ก็จะเกิดข้อผิดพลาดขึ้นมาเล็กน้อยเท่านั้น

การแก้ไขสัญญาณที่ผิดพลาดนี้ก็ขึ้นอยู่กับการออกแบบวงจรนี้ให้มีการสะสมข้อมูลเพื่อนำมาหมุนเฟสมีขนาดเป็นเท่าไร ถ้าเลือกใช้น้อยเกินไปก็จะเกิดความผิดพลาดมาก ถ้าเลือกมากเกินไปก็จะมีความซับซ้อนในการพัฒนา และเปลืองทรัพยากร ดังนั้นส่วนที่สำคัญที่สุดในวงจรนี้คือการเลือกพารามิเตอร์ที่เหมาะสมสำหรับใช้ในการสะสมข้อมูล โดยในเบื้องต้นได้เลือกใช้การสะสมข้อมูลเพื่อหาเฟสที่หมุนไปจำนวน 128 ค่า (ได้ทำการทดสอบค่าที่น้อยกว่านี้แล้วพบว่าเกิดความผิดพลาดขึ้นที่ไม่สามารถยอมรับได้ และถ้าใช้สูงกว่านี้ก็จะเปลืองทรัพยากร)



ภาพที่ 36 โครงสร้างส่วนการประมาณความถี่แบบละเอียด

จากภาพที่ 36 จะเห็นได้ว่าอินพุทที่เข้ามาทั้ง I และ Q จะถูกแปลงให้อยู่ในรูปของ Polar เพื่อนำไปประมาณเฟส โดยจะมีการคูณข้อมูลด้วย 2 เนื่องจากต้องการที่จะกำจัดการมอดูเลชั่น ให้ข้อมูลหมุนมาอยู่ที่ตำแหน่ง 0 หลังจากจะเป็นการเฉลี่ยเฟสที่ได้มาเพื่อนำไปประมาณเป็นความถี่เลื่อน (Doppler Frequency) ที่ยังตกค้างอยู่อีกเล็กน้อย สุดท้ายก็จะเป็นแก้ไขเฟสที่เปลี่ยนแปลงไปตามความถี่ ตามรูปที่แสดงไว้ในข้างต้น

การใช้ทรัพยากรจะเป็นดังนี้คือ จะใช้เซลล์ไป 821 Slices หรือ 1847 Logic cells หน่วยความจำใช้ไป 3 KB และความถี่สูงสุดที่สามารถทำงานได้คือ 168.47 MHz

2.3.5 การประมาณเฟสตกค้าง (Phase Recovery)

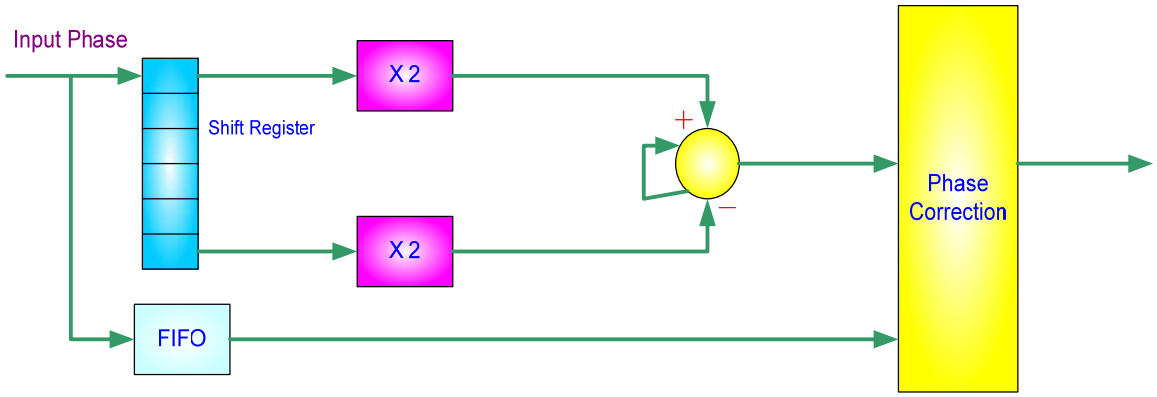
การประมาณเฟสตกค้างทางคณะผู้วิจัยได้เลือกที่จะใช้แนวคิดแบบ M-Power ซึ่งแนวคิดในการทำงานจะเป็นดังต่อไปนี้คือ

- ในขั้นแรกนำข้อมูลไปยกกำลัง M (M-power) เพื่อกำจัดการมอดูเลตออกไป (เหมือนกับที่อธิบายในตัวประมาณความถี่ตกค้างจากเฟสที่หมุนไป) คือ ถ้าข้อมูลส่งแบบ BPSK หรือ 2-PSK (M=2) โดยจะมองสัญญาณที่ได้รับให้อยู่ในรูปแบบเชิงขั้ว $Re^{j\theta}$, หรือ $R\angle\theta$ (สัญญาณ BPSK จะอยู่ในรูป $R\angle 0$ หรือ $R\angle 180$) ดังนั้นเมื่อนำสัญญาณเหล่านี้ไปยกกำลัง 2 มุมก็เปลี่ยนเป็น 2 เท่าของมุมเดิม

- ผลลัพธ์ คือ สัญญาณทั้งหมดจะหมุนมาอยู่ในตำแหน่งที่ 0 องศา แล้วนำค่าเฟสที่ได้หาร 2 ก็จะได้ค่าเฟสตกค้างที่เหลืออยู่ ซึ่งควรนำผลเฟสตกค้างจากหลายๆ Symbol มาเฉลี่ยกันเพื่อใช้มาหักล้างเฟสตกค้างออกจากสัญญาณที่ได้รับ

จะเห็นได้ว่าการทำเฟสตกค้างนี้จะคล้ายๆ กับการประมาณความถี่แบบละเอียด โดยเฟสที่ตกค้างจะเกิดจากจังหวะของภาครับกับภาคส่งที่ไม่ได้เข้าจังหวะ (Synchronous) กันทำให้เกิดเฟสขึ้น โดยแนวคิดนี้ส่วนที่ยากที่สุดก็คือการเฉลี่ยเฟส เนื่องจากเฟสที่ได้จะมีทั้งที่เป็นบวกและลบ ทำให้ในการพัฒนาฟังก์ชั่นนี้จะต้องมีการพิจารณาเทคนิคการเฉลี่ยเฟสว่าจะใช้แบบคิดเครื่องหมาย หรือแบบที่ไม่คิดเครื่องหมาย

ซึ่งแนวทางแก้ไขในส่วนนี้ก็คือการพิจารณาทั้ง 2 แบบคู่ขนานกันไปร่วมกับการพิจารณาข้อมูลว่าอยู่ในช่วงไหนของแกน และค่าเฟสที่ได้ก่อนหน้านั้นเป็นค่าอะไร เพื่อให้เฟสตกค้างที่คำนวณได้มีความต่อเนื่อง ไม่กลับไปกลับมา แต่ข้อเสียที่สำคัญของการพัฒนาการประมาณเฟสตกค้างนี้ก็คือ ข้อมูลที่ตัดสินใจออกมาจะเกิดการกลับเฟสหรือจะเรียกได้ว่าการเกิดความคลุมเครือของเฟส สำหรับระบบต้นแบบส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ที่พัฒนานี้ได้เลือกใช้ การมอดูเลตแบบ BPSK (Binary Phase Shift Keying) ซึ่งจะมีอยู่ 2 ค่าคือ $R\angle 0$ หรือ $R\angle 180$ โดยได้แก้ไขในฟังก์ชั่นการเข้าจังหวะเฟรมข้อมูล (Frame Recovery) อยู่ในหัวข้อถัดไป



ภาพที่ 37 โครงสร้างการประมาณเฟสตกค้าง

จากภาพที่ 37 จะพบว่าเฟสที่เข้ามาจะนำมาเฉลี่ยและกำจัดการมอดูเลตออกไป เพื่อหาเฟสตกค้างที่แท้จริง โดยส่วนที่สำคัญก็คือการหาค่าเฉลี่ยที่ได้กล่าวไว้ข้างต้นนั้นเอง ดังนั้นความซับซ้อนของวงจรนี้จะอยู่ที่ส่วนนี้นั้นเอง

และทรัพยากรที่ใช้สำหรับการประมาณเฟสตกค้างจะใช้เซลล์ไป 189 Slices หรือ 425 Logic cells หน่วยความจำใช้ไปทั้งหมด 3 KB และความถี่สูงสุดที่สามารถทำงานได้เท่ากับ 91.18 MHz

2.3.6 การเข้าจังหวะเฟรม (Frame Recovery)

การเข้าจังหวะเฟรมจะเป็นเทคนิกเพื่อแก้ไขความคลุมเครือของเฟสโดยที่การแก้ไขนี้จะทำให้เกิดการกลับเฟสในกรณีที่เฟสเป็น 180 องศา ซึ่งเทคนิกที่ใช้ก็คือการเพิ่มต้นเฟรมที่แน่นอนเข้าไปทุกๆ ระยะ โดยการเพิ่มต้นเฟรมจะเพิ่มทุกๆ 1024 บิตเพื่อให้สัมพันธ์กับการทำงานของ FFT ในการประมาณความถี่แบบหยาบ ซึ่งจะมีหลักการดังนี้

เฟสแบบใดที่ใกล้เคียงกับรูปแบบบิตที่ใส่ไว้มากกว่า เฟสนั้นก็คือเฟสที่ถูกต้องนั่นเอง (โดยในกรณี BPSK นี้ ก็อาจแก้เฟสคลุมเครือไป 0 หรือ 180 องศาเท่านั้น) ซึ่งวิธีการแบบนี้จะเป็นการใช้ Correlation ในการแก้ปัญหา โดยจะดูช่วงที่ข้อมูลสูงกว่าค่าที่ตั้งเอาไว้ (Threshold) เช่นค่าสูงสุดเป็น 100 ตั้งเอาไว้ที่ 80 ถ้าสัญญาณที่ทำ Correlation ออกมาแล้วได้ 79 ก็แสดงว่าไม่ใช้ข้อมูลที่ต้องการ แต่ถ้าสูงกว่าหรือเท่ากับ 80 ก็แสดงว่าข้อมูลที่เข้ามาถูกต้อง ในกรณีที่เป็นเฟส 0 แต่ถ้าเป็นเฟส 180 ก็จะตรวจสอบที่ 0 หรือค่าอาจจะอยู่ประมาณ 20 เป็นต้น

ดังนั้นทุกๆ 1024 บิตก็จะตรวจสอบทีนึง และยังจำเป็นต้องตรวจหาข้อมูลต้นเฟรมด้วยเนื่องจากข้อมูลจะเข้ามาเรื่อยๆ ไม่สามารถรู้ได้ว่าข้อมูลบิตไหนเป็นบิตเริ่มต้น การเข้าจังหวะเฟรมจึงแบ่งเป็น 2 ระดับคือ การเข้าจังหวะเพื่อหาขอบเขตเฟรมในตอนแรก (ตั้งเอาไว้ที่ 1024 บิต) และเมื่อหาขอบเขตได้แล้วก็จะไปหาว่าเฟรมข้อมูลนั้นเป็นค่าเริ่มต้นของข้อมูลหรือไม่ เพื่อที่จะนำข้อมูลทั้งหมดไปเข้าส่วนการถอดรหัสข้อมูลดิจิตอลเบสแบนด์ต่อไป

สรุปได้ว่าการแบ่งเฟรมจะแบ่งออกเป็นเฟรมย่อยๆ 16 เฟรม โดยในแต่ละเฟรมจะมีทั้งหมด 1024 บิต (Header + Data) ดังนั้นส่วนที่สำคัญก็คือส่วน Header โดยจะแบ่งออกเป็น 2 ส่วนเพื่อหาขอบเขต มี 64 บิต และตำแหน่งเริ่มต้น จะใช้ไป 8 บิต ส่วนข้อมูลในแต่ละเฟรมย่อยก็จะเหลืออยู่ 952 บิต ต่อไปจะกล่าวถึงเหตุผลในการเลือกใช้บิตในการหาขอบเขตว่าทำไมถึงใช้ 64 บิต

จากการศึกษาพบว่ายิ่งให้มี Header มากเท่าไรก็จะมีความผิดพลาดที่จะเข้าจังหวะผิดน้อยลง และการที่เลือกใช้ถึง 64 บิตเนื่องจากข้อมูลที่ได้จำเป็นต้องมีความถูกต้องเกือบ 100% เพื่อให้สามารถทำการถอดรหัสได้ถูกต้องไม่ผิดพลาด และข้อมูลในการจัดการเลือกรูปแบบอ้างอิงมาจากข้อมูลในอินเตอร์เน็ตที่ได้กล่าวถึงในมาตรฐานต่างๆ ที่นิยมใช้ในการวัด Telemetry Receiver และ PCM ซึ่งสอดคล้องกับระบบต้นแบบที่กำลังพัฒนาอยู่นี้ตามภาพที่ 38

OPTIMUM FRAME SYNC PATTERNS

# BITS	OCTAL	HEX	FALSE-LOCK PROBABILITY
7	540	B0	5.7 E-1
8	560	B8	4.2 E-1
9	560	B80	2.9 E-1
10	6700	DC0	1.8 E-1
11	5560	B70	9.1 E-2
12	6540	B60	5.1 E-2
13	72600	EB00	2.8 E-2
14	71500	E680	1.5 E-2
15	73120	ECA0	6.6 E-3
16	727100	EB90	3.5 E-3
17	746500	F3500	1.7 E-3
18	746500	F3500	8.2 E-4
19	7631200	F9940	3.8 E-4
20	7336100	EDE20	2.2 E-4
21	7351300	EE9600	1.1 E-4
22	7466500	F36A00	4.9 E-5
23	75346400	F5CD00	2.5 E-5
24	76571440	FAF320	1.3 E-5
25	762670400	F96E200	6.4 E-6
26	764654200	FA6B100	3.1 E-6
27	765514600	FAD3300	1.6 E-6
28	7536263000	F5E5980	8.0 E-7
29	7536315000	F5E66800	4.1 E-7
30	7657146400	FAF33400	2.1 E-7
31	77467650200	FE6FA840	*
32	77465450200	FE6B2840	*

ภาพที่ 38 Optimum Frame sync Patterns

และทรัพยากรที่ใช้ในการพัฒนาฟังก์ชั่นนี้จะใช้เซลล์ไป 372 Slices หรือ 837 Logic cells หน่วยความจำใช้ไปทั้งหมด 3 KB และความถี่สูงสุดที่สามารถทำงานได้เท่ากับ 70.34 MHz

2.3.7 ส่วนการควบคุมกำลังงาน (Automatic Gain Control)

ในส่วนการทำงานนี้จะเป็นการควบคุมระดับสัญญาณไม่ให้สูง หรือต่ำจนเกินไปเพื่อให้สัญญาณที่เข้ามาคงที่ และช่วยให้ฟังก์ชั่นการเข้าจังหวะสัญญาณอื่นๆ สามารถทำงานได้เต็มประสิทธิภาพ นั้นก็คือ การประมาณความถี่แบบหยาบจะมีความแน่นอนมากยิ่งขึ้น เนื่องจาก ส่วนการทำงานนี้จะให้สัญญาณที่มีความแรงอยู่ในระดับสูง ก็จะช่วยให้การประมาณ FFT ในฟังก์ชั่นต่อไปคำนวณค่าได้ดีขึ้นนั้นเอง และควบคุมกำลังงานนี้จะใช้เทคนิก M-Power เพื่อให้ได้ค่าที่จะต้องไปตัดสินใจ โดยค่าๆ นี้จะถูกนำไปสร้างเป็นพารามิเตอร์สำหรับควบคุมในส่วน AGC Board ที่เป็นอนาลอก (รายละเอียดจะกล่าวในหัวข้อ AD8321 AGC)

ดังนั้นการทำงานจะเป็นการวัดสัญญาณที่เข้ามาไปทีละ 32 บิท เพื่อตรวจสอบช่วงของข้อมูลว่าเปลี่ยนแปลงไปมากน้อยขนาดไหน และค่าที่ออกมาจะใช้ไปเพิ่มระดับของสัญญาณที่จะใช้ในการควบคุมพารามิเตอร์สำหรับบอร์ด AGC อีกทีหนึ่ง

ทรัพยากรที่ใช้ในการพัฒนาฟังก์ชั่นนี้จะใช้เซลล์ไป 106 Slices หรือ 239 Logic cells และความถี่สูงสุดที่สามารถทำงานได้เท่ากับ 218.95 MHz

2.4 ส่วนการติดต่อกับภาคสัญญาณวิทยุ (Interfacing with RF Module)

อุปกรณ์ที่ใช้สำหรับเชื่อมต่อกับส่วนสัญญาณวิทยุจะแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ ฝั่งส่งข้อมูล และฝั่งรับข้อมูล จากศึกษาและออกแบบระบบต้นแบบส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์สำหรับสื่อสารกับดาวเทียมดวงนี้ จะเลือกที่จะใช้การมอดูเลตแบบ BPSK ซึ่งก็ถือว่าเพียงพอกับการรองรับอัตราในการส่งข้อมูล (ด้วยอัตรา 2 Mbps) และขนาดของช่องสัญญาณที่มีขนาด 15 MHz และเชื่อมต่อกับส่วนสัญญาณวิทยุที่ 70 MHz (สำหรับการทดสอบสัญญาณ ถ้าใช้งานจริงจะเป็น 65 MHz ที่ความถี่ขาขึ้น และ 75 MHz สำหรับความถี่ขาลง) ดังนั้นอุปกรณ์ที่เลือกใช้จะเป็นอุปกรณ์ที่สามารถรองรับเงื่อนไขนี้ได้ และที่สำคัญคือจะต้องหาได้โดยง่ายในท้องตลาด หรือที่เรียกกันอีกนัยนึงว่า COST (Commercial Of the Shelf) และราคาเหมาะสมกับงบประมาณที่มีอยู่จำกัด

จากการสอบถามไปยังบริษัทต่างๆ และการค้นหาทางอินเตอร์เน็ต เพื่อให้ได้ตามเงื่อนไขที่มีอยู่พบว่าในฝั่งส่งจะเลือกใช้ AD9856 Quadrature Digital Upconverter ของบริษัท Analog Device ซึ่งสามารถทำตามเงื่อนไขที่ตั้งไว้ได้โดยจะทำการย้ายความถี่จาก Baseband ไปที่ความถี่กลาง 65 หรือ 70 MHz ได้ รวมทั้งแปลงข้อมูลจากดิจิตอลให้เป็นอนาล็อก

ในส่วนฝั่งรับจะเลือกใช้อุปกรณ์ 2 ตัวเพื่อให้ได้ตามเงื่อนไขก็คือตัว ADC (Analog to Digital Converter) ที่เลือกใช้ AD6644 ตาม และ Digital Receive Signal Processor AD6620 ตาม ซึ่งทั้ง 2 ตัวก็เป็นของบริษัท Analog Device โดยทั้งคู่จะทำหน้าที่แปลงสัญญาณให้อนาล็อกให้เป็นดิจิตอล และทำการดีมอดูเลตสัญญาณพร้อมทั้ง Down-Converter สัญญาณลงมาจากความถี่กลาง 70 หรือ 75 MHz ให้เป็นสัญญาณที่เบสแบนด์เพื่อนำเข้าบอร์ดประมวลผล FPGA

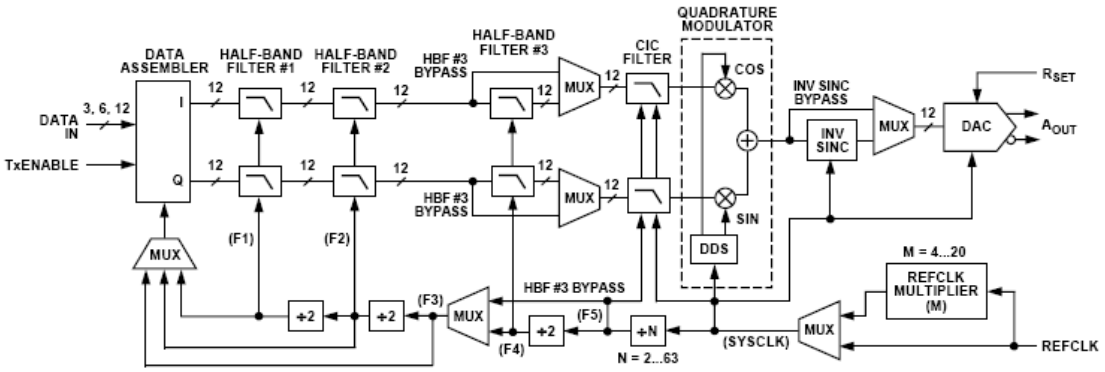
และขอเพิ่มเติมลงไปว่าในฝั่งรับจำเป็นต้องมีการควบคุมสัญญาณเพื่อไม่ให้สัญญาณที่เข้ามาสูงหรือต่ำเกินไป ซึ่งถ้าเป็นเช่นนั้นก็อาจจะทำให้การเข้าจังหวะอาจจะเกิดคามผิดพลาดขึ้นมาได้ ดังนั้นจึงต้องมีส่วน AGC (Automatic Gain Control) ซึ่งในส่วนอุปกรณ์อนาล็อกAD8321 AGC

ในการควบคุมสัญญาณอนาล็อก โดยสัญญาณควบคุมจะได้มาจากการประมาณกำลังงานจากส่วนประมวลผลในบอร์ด FPGA (ในส่วน AGC) อีกทีหนึ่ง ตามที่ได้กล่าวไว้ในหัวข้อที่ผ่านมา

ตามที่จะได้กล่าวไว้ในข้างต้นนี้ จะเห็นได้ว่าในการพัฒนาส่วนเชื่อมต่อกับภาคสัญญาณวิทยุที่จะเป็นส่วนสุดท้ายสำหรับการพัฒนาส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ จะใช้อุปกรณ์มาตรฐานที่มีขายอยู่ตามท้องตลาด โดยจะแบ่งเป็น 2 ส่วนใหญ่ๆ ก็คือส่วนแรกภาคส่งสัญญาณที่จะเชื่อมต่อกันที่ความถี่กลาง 65 MHz (สำหรับติดต่อจริงกับดาวเทียม) ที่จะใช้อุปกรณ์แค่ 1 ตัวคือ AD9856 Quadrature Digital Upconverter ที่จะทำการมอดูเลตแบบ BPSK และย้ายความถี่ไปที่ความถี่กลาง ส่วนที่สองคือภาครับสัญญาณที่เชื่อมต่อที่ความถี่กลาง 75 MHz (สำหรับติดต่อจริงกับดาวเทียม) โดยจะใช้อุปกรณ์ 3 ตัวคือ AD8321 Automatic Gain Control ทำหน้าที่ปรับความแรงของสัญญาณไม่ให้ต่ำหรือสูงเกินไป AD6644 Analog to Digital Converter แปลงสัญญาณอนาลอกที่รับมาเป็นสัญญาณดิจิตอล และสุดท้าย AD6620 Digital Receive Signal Processor ใช้ในการย้ายความถี่ลงมาให้เป็นสัญญาณเบสแบนด์ เพื่อที่ส่วนการเข้าจังหวะสัญญาณ (Signal Recovery) จะได้นำไปตัดสินใจได้ และทำงานร่วมกันเพื่อให้สามารถรับสัญญาณที่มาจากส่วนสัญญาณวิทยุได้

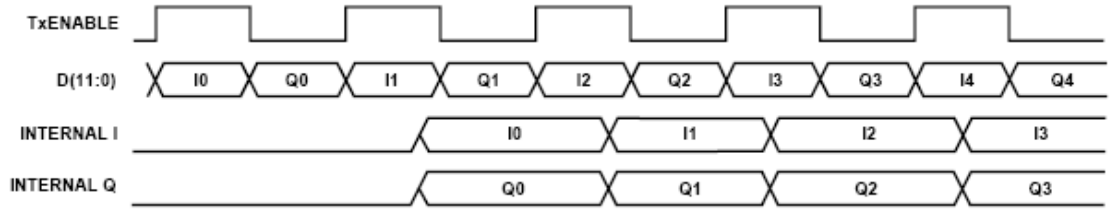
2.4.1 AD9856 Quadrature Digital Upconverter

อุปกรณ์นี้ใช้สำหรับเชื่อมต่อในภาคส่งสัญญาณ โดยจะทำหน้าที่มอดูเลตสัญญาณจากช่องสัญญาณ In-Phase และ Quadrature ให้กลายเป็น BPSK และย้ายความถี่ไปที่ความถี่กลาง โดยจะมีโครงสร้างตามภาพที่ 39



ภาพที่ 39 AD9856 Block Diagram

สิ่งที่จำเป็นต้องคำนึงถึงเป็นอันดับแรกก็คือจังหวะเวลาที่จะใช้ในการรับข้อมูล ไม่ว่าจะเป็น Data in, Tx enable, และ REFCLK ซึ่งการกำหนดค่าเหล่านี้เพื่อใช้งานก็จะต้องมาจากความถี่ที่จะใช้ส่งข้อมูลออกมาจากส่วนดิจิตอลเบสแบนด์ โดยข้อมูลที่ถูกส่งออกมาจากส่วนดิจิตอลเบสแบนด์จะมีขนาด 3.33 MHz (เกี่ยวพันกับส่วนภาครับที่จะใช้ทดสอบ และข้อมูลที่ได้จากการเข้ารหัสสัญญาณ) ในแต่ละช่องสัญญาณ ดังนั้นค่า Tx enable ก็จะต้องออกที่ความถี่ 3.33 MHz เนื่องจากมีอยู่ 2 ช่องสัญญาณ (ในช่องสัญญาณ In-Phase จะเป็นข้อมูลที่ใช้ในการรับส่งข้อมูลจริง ส่วน Quadrature ก็จะใช้วิธีการส่ง “0” ไปให้แก่ตัวบอร์ด) และจะมีลักษณะในการส่งข้อมูลดังภาพที่ 40



ภาพที่ 40 Bit Input mode, Alternate Tx Enable Timing

สำหรับส่วน REFCLK จะสัมพันธ์กับความถี่กลาง (IF Frequency) ซึ่งในที่นี้จะใช้ 65 MHz ดังนั้น REFCLK ต้องมีขนาดเป็น 2 เท่าหรือมากกว่านั้น (130 MHz) เนื่องจาก REFCLK จะใช้สำหรับการชักสัญญาณ (Sampling) ในวงจร Quadrature Modulator จึงจำเป็นต้องมีขนาดความถี่มากกว่าหรือเท่ากับ 2 เท่าของความถี่ขาออกนั้นเอง โดยในเบื้องต้นก็จะใช้ที่ความถี่ 200 MHz แต่เนื่องจากตัวอุปกรณ์ที่ขายอยู่ตามท้องตลาดไม่มี หรือถ้ามีก็จะมีราคาแพง ดังนั้นจึงจะใช้วิธีการคูณความถี่ หรือใช้บล๊อก REFCLK Multiplier (M) ตามภาพที่ 40 โดย REFCLK จะใช้ 10 MHz และค่า M เป็น 20

และในการกำหนดค่าสำหรับตัวแปร N หรือค่า CIC Interpolate rateเพื่อให้ความถี่ขาเข้าสัมพันธ์กับความถี่ขาออก จะเป็นไปตามสมการดังต่อไปนี้

$$REFCLKxM = 4xHxNxDatain$$

โดย ค่า H จะเป็น 1 ต่อเมื่อ Half-Band Fitter 3 Bypass หรือ 2 ต่อเมื่อ Half-Band Filter 3 enable

M จะมีค่าตามค่า REFCLK Multiplier ที่กำหนดไว้

REFCLK เป็นค่า REFCLK ที่มาจากตัวบอร์ด

N คือ CIC interpolation rate $(2 \leq N \leq 63)$

และจากการกำหนดค่าพารามิเตอร์ที่ผ่านมาแล้ว จะพบได้ว่าสามารถหาค่า N ได้เท่ากับ 15 ในกรณีที่บายพาส Half-Band Filter และเท่ากับ 4 ในกรณีที่ไม่บายพาส ซึ่งก็จะเลือกใช้ในกรณีที่บายพาส ไปก่อนในเบื้องต้น

และก่อนที่จะสามารถส่งข้อมูลไปให้ AD9856 ทำการประมวลผลสัญญาณได้นั้นจะต้องมีการกำหนดค่าพารามิเตอร์เหล่านี้เข้าไปให้แก่อุปกรณ์ ซึ่งจะทำได้ 2 วิธีคือการโปรแกรมผ่านสาย Parallel ที่จะทำงานผ่านคอมพิวเตอร์ หรือสองคือการโปรแกรมผ่านสาย Serial ที่จะต้องสั่งผ่านบอร์ดประมวลผลเอง และด้วยเหตุผลที่ว่าบนคอมพิวเตอร์เองจะต้องให้มีการทำงานแค่ซอฟต์แวร์ประยุกต์เท่านั้น ดังนั้นจึงจำเป็นที่จะต้องโปรแกรมผ่าน Serial ด้วยบอร์ด FPGA โดยจะต้องพัฒนาซอฟต์แวร์สำหรับโปรแกรมพารามิเตอร์เอง โดยตำแหน่งของข้อมูลที่จะใช้ในการกำหนดพารามิเตอร์จะเป็นไปตามตารางที่ 7

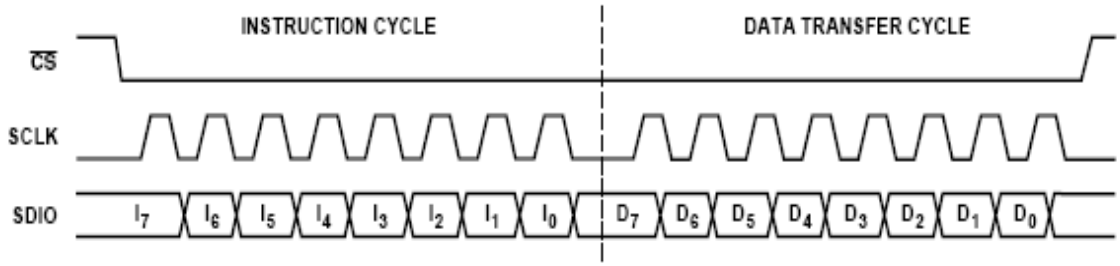
ตารางที่ 7 Serial Control Bus Register

Address (Hex)	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
00	SDO Active	LSB First	REFCLK Mult.(4)	REFCLK Mult.(3)	REFCLK Mult.(2)	REFCLK Mult.(1)	REFCLK Mult.(0)	Reserved
01	CIC Gain	Continuous Mode	Full Sleep Mode	Single tone Mode	Bypass Inverse Sinc Filter	Bypass REFCLK Mult.	Input Format Select (1)	Input Format Select (0)
02	Frequency Tuning Word (7:0)							
03	Frequency Tuning Word (15:8)							
04	Frequency Tuning Word (23:16)							
05	Frequency Tuning Word (31:24)							
06	N (5)	N (4)	N (3)	N (2)	N (1)	N (0)	Spectral Inversion	Bypass Third Half-Band Filter
07	Gain Control Bit(7:0)							

โดยค่าอื่นๆ ที่ไม่ได้กำหนดไว้ก่อนหน้านี้จะเหมือนกับค่า Default ที่บอร์ดได้กำหนดเอาไว้ ยกเว้นค่าและค่า Input format Select จะมีค่าเป็น “10” สำหรับข้อมูลขนาด 12 bit ที่ได้เลือกใช้ และ Frequency Tuning Word (FTW) ที่จะมีค่าตามสมการดังต่อไปนี้

$$fout = (FTWxSYSCLK) / 2^{32}$$

สำหรับการนำค่าเหล่านี้ไปใส่ไว้ใน Address ตามตารางที่ 7 ก็จะต้องมีการกำหนดค่าให้แก่ Address ก่อน หลังจากนั้นก็จะเป็นค่าพารามิเตอร์ตาม Address เหล่านั้น โดยจะมีรูปแบบตามภาพที่ 41



ภาพที่ 41 Serial Port Writing Timing

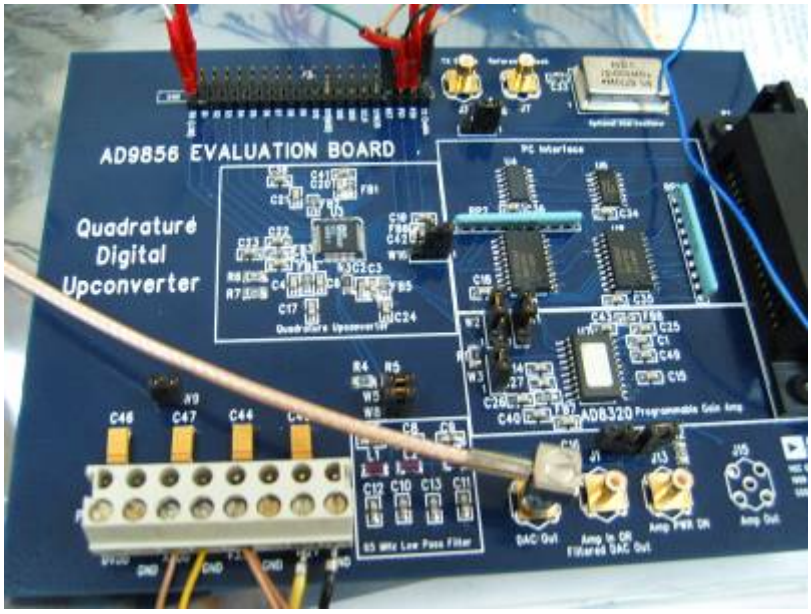
โดยการกำหนดในแต่ละส่วนสำหรับ I7 ถึง I0 ก็จะมีรูปแบบเป็นไปตามตารางที่ 8 และหลังจากนั้นก็จะเป็นการกำหนดค่าพารามิเตอร์ตามที่ได้กล่าวไป โดยส่วนที่สำคัญคือค่า SCLK ที่จะต้องขึ้นเฉพาะที่จะใช้เท่านั้น คือจะมีค่าแค่ช่วงส่งคำสั่งเท่านั้น ช่วงอื่นๆ จะไม่มีค่า (นั้นคือกำหนดให้เป็น 0) รวมถึงช่วงระหว่าง Instruction Cycle กับ Data Transfer Cycle ก็จะต้องเป็น “0” เท่านั้นไม่ยังงั้นก็จะทำให้เกิดความผิดพลาดขึ้นมาได้

ตารางที่ 8 Instruction Byte Information

MSB	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**LSB**
R/W	N1	N2	A4	A3	A2	A1	A0

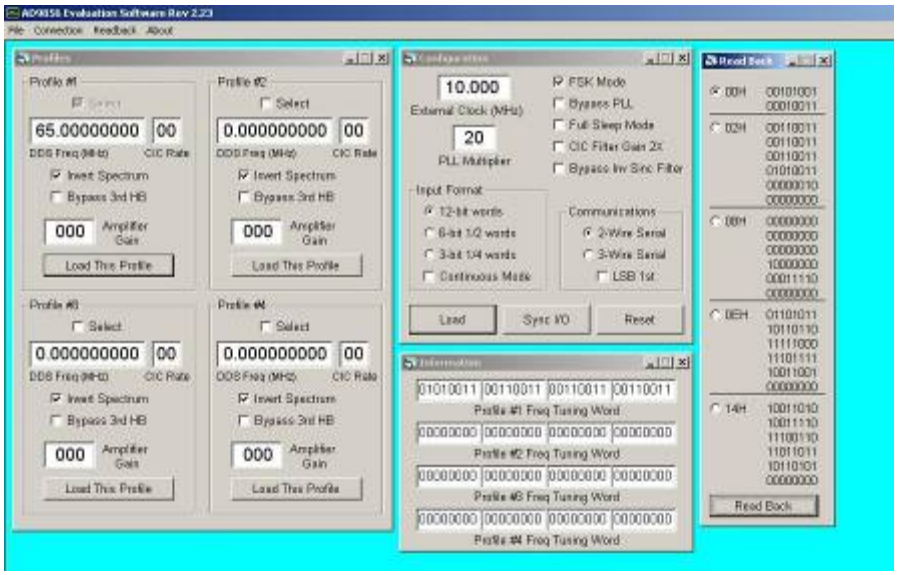
จากตารางที่ 8 ค่า N1 และ N2 จะกำหนดให้เป็น 0 ทั้งคู่สำหรับการส่งข้อมูลทีละ Bytes และ A4 ถึง A0 จะเป็นค่า Address ในตารางที่ 7 อีกส่วนที่ต้องกำหนดสำหรับการติดตั้ง

ค่าพารามิเตอร์ก็คือ การกำหนดค่า Profiles ที่ได้เลือกใช้ Profiles 1 (ข้อกำหนดจากตัวอุปกรณ์ถ้าใช้ 1 ก็จะใส่ค่า “0” เข้าไปที่ขา PS0 และ PS1 ด้วย) โดยภาพที่ 42 จะแสดงถึงรูปบอร์ดและขาที่ใช้สำหรับการเชื่อมต่อกับส่วนอื่นๆ

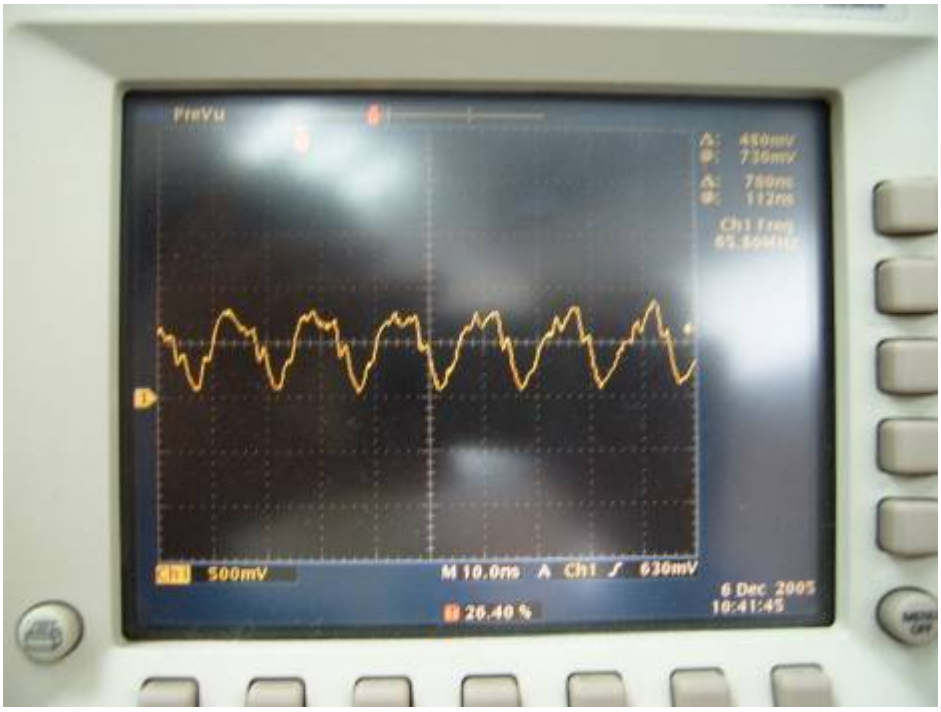


ภาพที่ 42 บอร์ด AD9856 Quadrature Digital Upconverter

สำหรับการทดสอบสัญญาณที่ได้จากบอร์ด AD9856 นี้จะแบ่งออกเป็น 2 ส่วนใหญ่ๆ ก็คือ ส่วนแรกการทดสอบสัญญาณผ่านพอร์ตพาราแรล โดยจะมีซอฟต์แวร์ที่ใช้สำหรับการโปรแกรมพารามิเตอร์ที่ได้กล่าวไปนี้ โดยจะทดสอบแบบ Single Tone Mode ที่จะทำให้เห็นสัญญาณได้ที่ความถี่ที่ต้องการได้ทันที แค่เพียงใส่ค่าพารามิเตอร์เข้าไปเท่านั้น ตามภาพที่ 43 และสัญญาณที่ออกมาจะได้ตามภาพที่ 44 ซึ่งจะได้อยู่ในช่วง 65 MHz ที่ได้กำหนดไว้

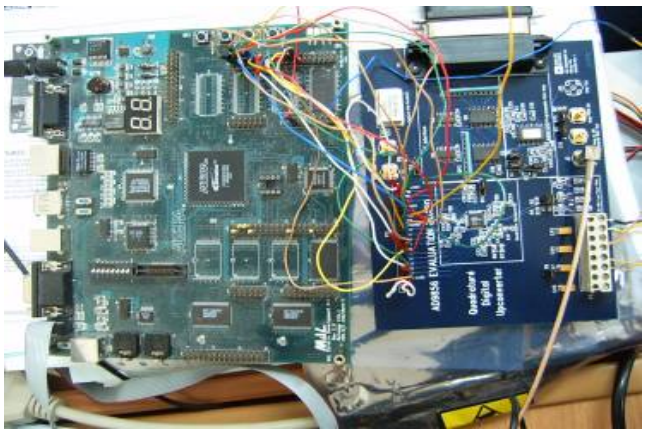


ภาพที่ 43 ซอฟต์แวร์สำหรับโปรแกรมพารามิเตอร์ลงบอร์ด AD9856



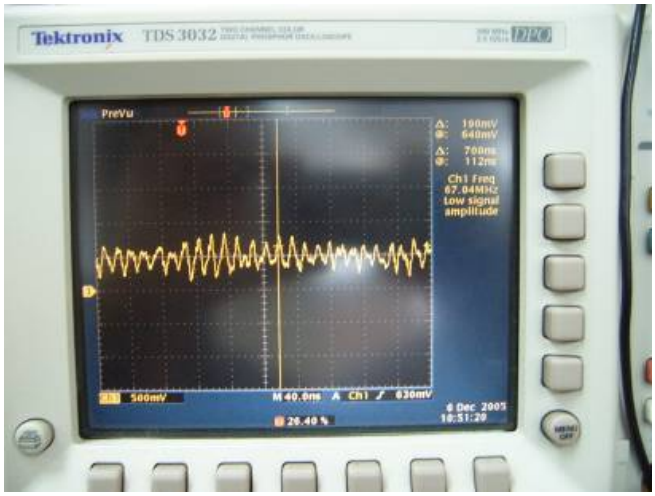
ภาพที่ 44 เอาพุตที่ออกจากการทดสอบในขั้นตอนที่ 1

ในขั้นที่ 2 จะเป็นการติดต่อแบบ Serial โดยพัฒนาซอฟต์แวร์สำหรับกำหนดค่าพารามิเตอร์ด้วยภาษา VHDL ผ่านเข้าไปในบอร์ด FPGA เพื่อไปเชื่อมต่อกับส่วนย้ายความถี่และมอดูเลตของบอร์ดพัฒนา AD9856 ซึ่งการพัฒนาจะเป็นไปตามที่ได้กล่าวมาในข้างต้นในเรื่องการโปรแกรมค่าพารามิเตอร์ผ่านพอร์ต Serial และภาพที่ 45 แสดงให้เห็นถึงการเชื่อมต่อระหว่างบอร์ด FPGA กับ AD9856

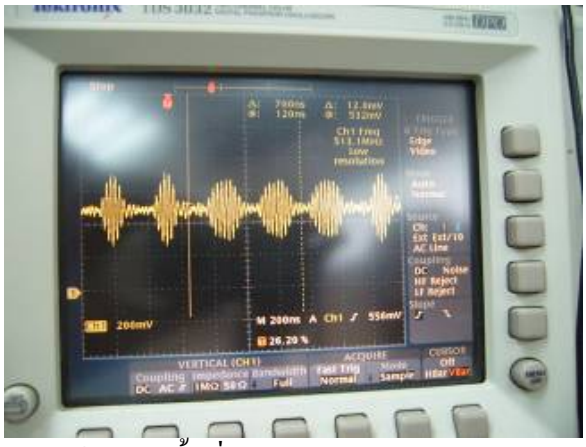


ภาพที่ 45 การเชื่อมต่อระหว่างบอร์ด FPGA กับ AD9856

สัญญาณที่ออกมาจากการทดสอบโดยภาพที่ 46 จะเป็นลักษณะของการทดสอบแบบ Single Tone เพื่อตรวจสอบว่าสามารถโปรแกรมพารามิเตอร์ถูกต้องหรือไม่ และภาพที่ 47 จะเป็นการทดสอบการส่งข้อมูลจริงจากบอร์ด FPGA

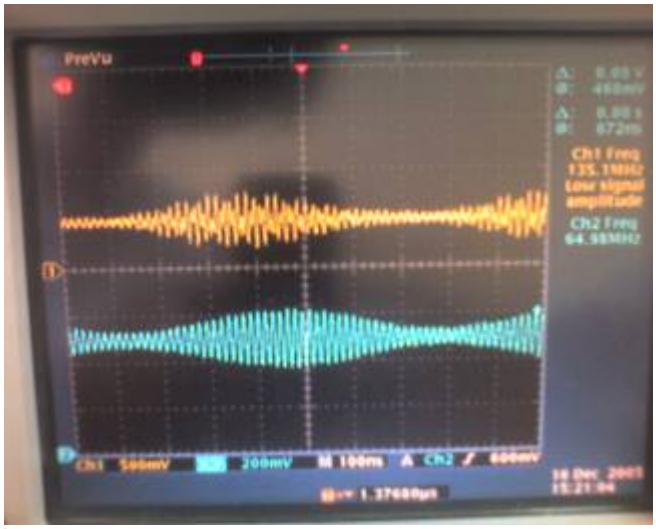


ภาพที่ 46 การทดสอบแบบ Single Tone ในขั้นที่ 2



ภาพที่ 47 การทดสอบการส่งข้อมูลในขั้นที่ 2

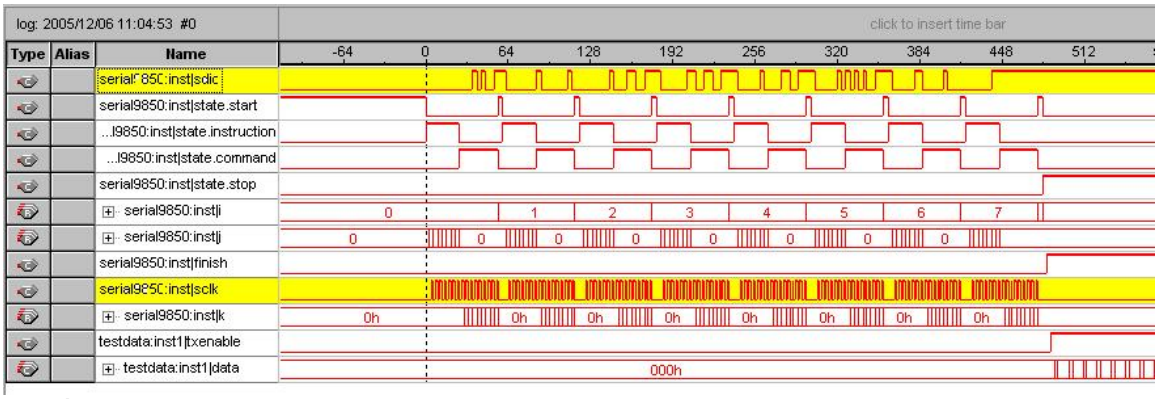
สังเกตได้ว่าข้อมูลที่ถูกมอดูเลตออกมาในภาพที่ 47 จะมีรูปร่างที่ไม่ค่อยสวยจะมีสัญญาณรบกวนค่อนข้างมาก แต่เนื่องจากว่าตัวบอร์ด AD9856 จะมีส่วนที่มีตัวกรองสัญญาณจะทำให้สัญญาณที่ออกมามีรูปร่างที่สวยขึ้นในภาพที่ 48



ภาพที่ 48 สัญญาณที่ออกมาจาก AD9856 สีเหลืองคือก่อนเข้าตัวกรองสัญญาณ สีฟ้าคือหลังจากออกจากตัวกรองสัญญาณ

และสุดท้ายในหัวข้อนี้จะขอกล่าวถึงอุปสรรคที่พบนั้นก็คือ การเชื่อมต่อสายที่ไม่มั่นคงเพียงพอในแต่ละสายสัญญาณ (มีอยู่กว่า 30 เส้น) โดยได้แก้ไขปัญหาเรื่องการเชื่อมต่อสายระหว่างบอร์ด และอีกปัญหานึงที่พบก็คือ การโปรแกรมค่าพารามิเตอร์ที่ผิดพลาดไม่ตรงจังหวะนั้นคือ จังหวะการให้ค่า SCLK กับจังหวะการให้ค่า SDIO ที่กล่าวไปในข้างต้นไม่ตรงกัน มีเกินหรือขาดไปบ้าง จึงได้ทำการแก้ไขปรับปรุงจนกระทั่งจังหวะของทั้งสองสัญญาณเป็นไปตามมาตรฐานของบอร์ด AD9856 โดย จะแสดงถึงการให้จังหวะสัญญาณของทั้งสองเส้นและตรวจจับจาก

ซอฟต์แวร์ ChipScope Pro V7.1 ในภาพที่ 49 ซึ่งเป็นซอฟต์แวร์ที่ตรวจจับสัญญาณที่ทำงานบนบอร์ด FPGA



ภาพที่ 49 การตรวจจับสัญญาณจาก ChipScope Pro V7.1

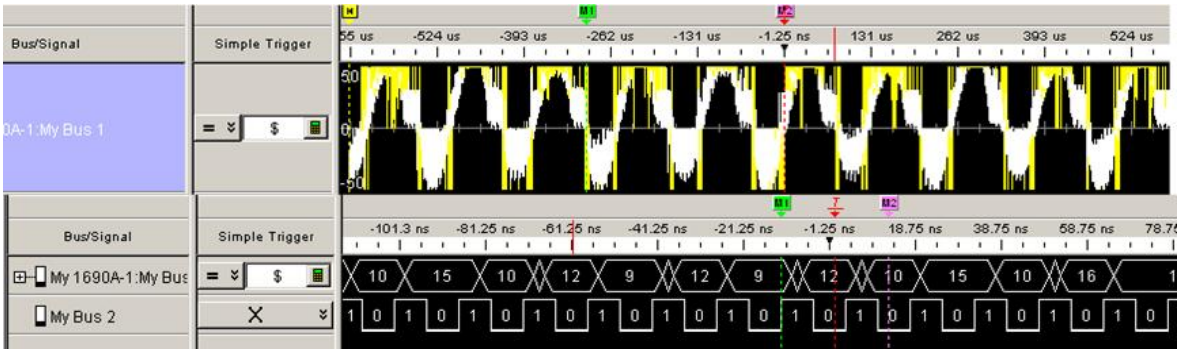
2.4.2 AD6644 Analog to Digital Converter

สำหรับการทำงานในส่วนนี้จะรับสัญญาณที่มีความถี่ 75 MHz โดยมีการชักสัญญาณที่ 66.66 MHz ซึ่งบอร์ด AD6644 ตามภาพที่ 50 จะรับสัญญาณอนาลอกเข้ามาและแปลงให้อยู่ในรูปของดิจิตอลออกมา

ภาพที่ 50 AD6644 Analog to Digital Converter

โดยการรับสัญญาณจะรับที่ขา Ain และสัญญาณดิจิตอล 12 บิท พร้อมกับสัญญาณนาฬิกา 66.66 MHz จะออกมาที่ขา Parallel ทางด้านซ้ายของภาพที่ 50 ซึ่งสัญญาณที่

ออกมาจากขา Parallel นี้ได้ถูกวัดจาก Logic Analyzer ได้ดังภาพที่ 51 ซึ่งจะเห็นได้ว่าสัญญาณที่ออกมาจากการทดสอบเป็นรูป Sine Wave ถูกต้องตามที่คาดไว้ เพียงแต่อัตราการชักสัญญาณไม่ตรงกับสัญญาณที่ได้รับมา ดังนั้นสัญญาณที่ออกมาจึงไม่ค่อยสวยนัก



ภาพที่ 51 สัญญาณที่จับได้จาก Logic Analyzer

ซึ่งในการชักสัญญาณนี้จะไปสัมพันธ์กับการรับข้อมูลในส่วนต่อไปนั้นก็คือการย้ายความถี่ลงมาเป็นสัญญาณดิจิตอลเบสแบนด์ (รายละเอียดอยู่ในหัวข้อ AD6620 Digital Receive Signal Processor)

ดังนั้นการพัฒนาในส่วนนี้จะเป็นแค่ส่วนการแปลงให้สัญญาณที่รับมาเป็นอนาลอกเปลี่ยนเป็นดิจิตอลเท่านั้น โดยมีการชักสัญญาณที่ 66.66 MHz และที่ชักสัญญาณที่ความถี่นี้เนื่องจาก สัญญาณนาฬิกาที่ความถี่ 75 MHz นั้นไม่มีขายทั่วไปตามท้องตลาด หรือถ้าขายก็จะมีราคาแพงเกินไป จึงจำเป็นต้องใช้ที่ความถี่นี้ แต่สิ่งที่เกิดขึ้นก็จะสามารถแก้ไขได้ใน AD6620 ต่อไป

อุปสรรคที่พบในการพัฒนาในส่วนนี้ที่สำคัญก็คือสัญญาณที่ออกมา ในครั้งแรกที่เห็น Oscilloscope ไม่ทราบว่าจะมีความถูกต้องหรือไม่ จึงได้ปรับเปลี่ยนมาตรวจสอบด้วย Logic Analyzer ซึ่งก็จะเป็นไปตามที่ได้กล่าวมาในข้างต้น และอีกสิ่งนึงที่มีปัญหาก็คือ การเลือกสัญญาณนาฬิกาว่าควรจะเลือกที่เท่าไร โดยในความตั้งใจแรกคือ จะเลือกที่จะใช้ที่ความถี่ 60 MHz แต่เนื่องจากประสบปัญหาที่สำคัญก็คือ ของที่จะใช้ขาดตลาด และหาซื้อไม่ได้ จึงต้องจำเป็นอย่างยิ่งที่จะใช้ที่ 66.66 MHz และที่ไม่เลือกใช้ที่ 75 MHz ก็เป็นไปตามเหตุผลที่กล่าวไว้ข้างต้น รวมถึงสัญญาณนาฬิกาที่ 75 MHz ก็ไม่สามารถนำมาใช้ได้ที่ AD6620 ด้วย เพราะว่าตัวบอร์ด AD6620 รับสัญญาณได้ที่ความถี่สูงสุด 67 MHz

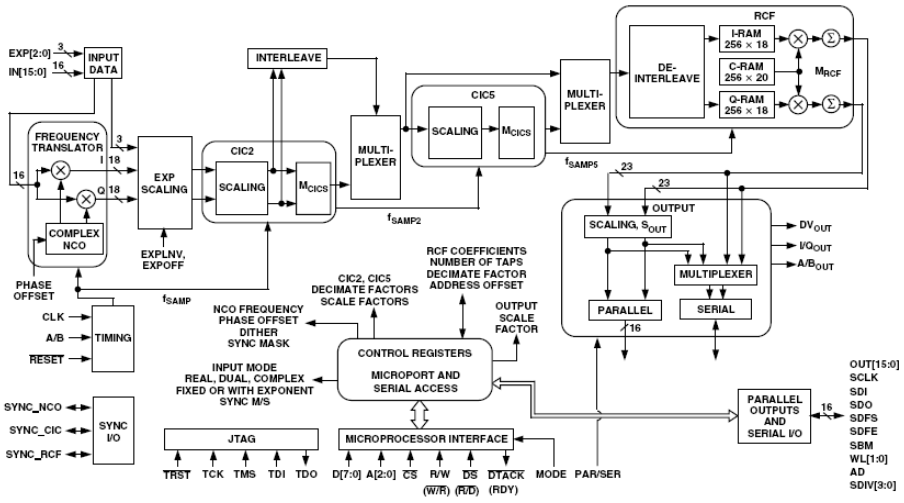
2.4.3 AD6620 Digital Receive Signal Processor

ในส่วนนี้จะทำหน้าที่อย่างเดียวเท่านั้นก็คือ การย้ายความถี่ลงมาที่เบสแบนด์เท่านั้น โดยสัญญาณที่ได้รับมาจะถูกแยกออกมาเป็นสัญญาณ In-phase และ Quadrature ช่องละ 16 บิตสลับกันไป ดังนั้นส่วนที่สำคัญที่สุดก็คือส่วนการย้ายความถี่นั้นเอง ที่สามารถจะเปลี่ยนแปลงพารามิเตอร์ต่างๆ ตามที่ต้องการได้ โดยการทำงานของบอร์ด AD6620 นี้จะเป็นไปดังภาพที่ 52

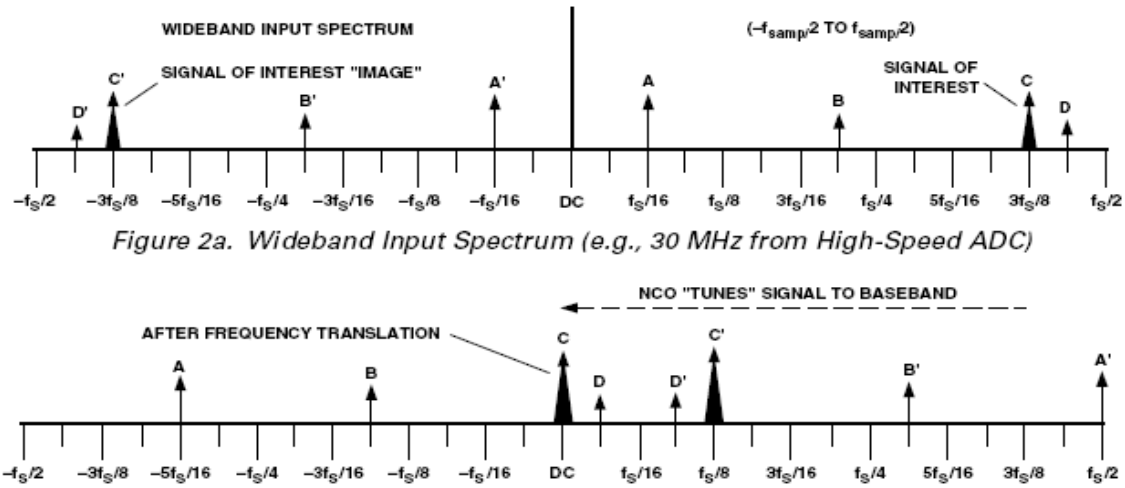
ส่วนที่สำคัญของ AD6620 นี้จะมีอยู่ด้วยกัน 4 ส่วนคือ Frequency Translator, CIC2, CIC5, และ RCF ซึ่งส่วนที่สำคัญที่สุดก็คือส่วน Frequency Translator ที่จะใช้สำหรับการย้ายความถี่ลงมาที่เบสแบนด์ตามที่ต้องการ โดยเทคนิกที่สำคัญที่บอร์ดนี้ใช้ก็คือ การเลื่อนความถี่และหมุนไปในตำแหน่งที่ต้องการตามภาพที่ 53 ส่วนอีก 3 ส่วนที่เหลือจะสามารถคำนวณค่าพารามิเตอร์สำหรับแต่ละส่วนได้จากซอฟต์แวร์ Digital Filter ที่ได้มาพร้อมกับบอร์ด

ใน Frequency Translator จะมีการคำนวณค่า Frequency ที่ปรับนั้นจะเป็นไปตามสมการข้างล่างนั้น จะเห็นได้ว่าส่วนที่สำคัญก็คือ Fch ที่จะเป็นความถี่ที่เข้ามาที่บอร์ด AD6620 ซึ่งจะได้ขนาด 8.34 MHz ตามที่แสดงไว้ในภาพที่ 54

$$NCO\_FREQ = 2^{32}\, x \bmod\left(\frac{Fch}{Fsamp},1\right)$$

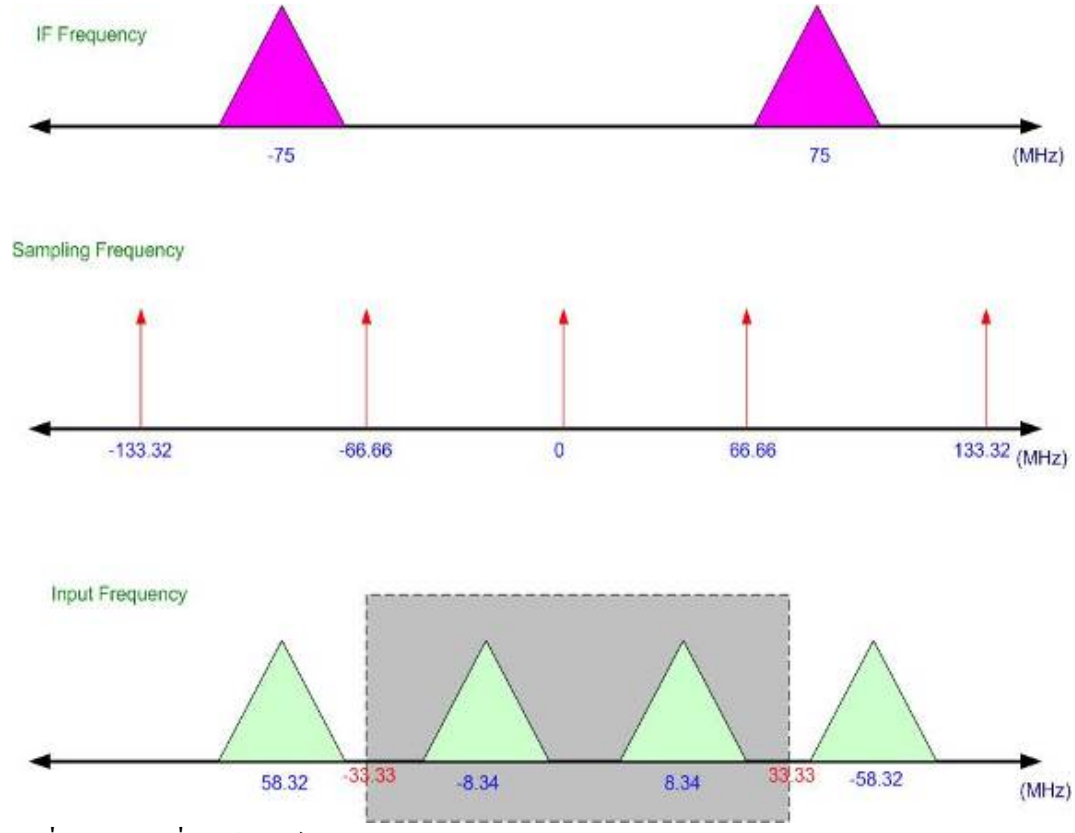


ภาพที่ 52 Block Diagram of AD6620



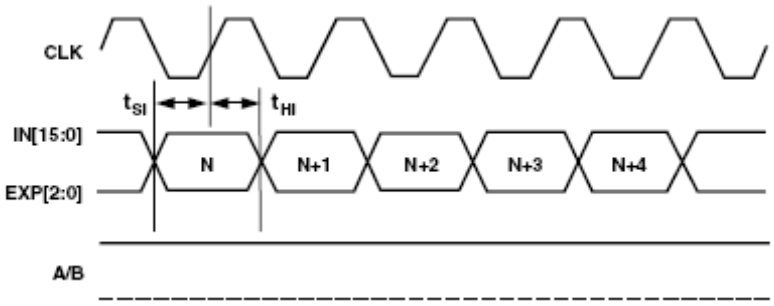
ภาพที่ 53 Frequency Translation Technique

ดังนั้นสัญญาณที่เข้ามาจะมีขนาดอยู่ในช่วงของความถี่ที่จะชักสัญญาณ (Sampling Frequency) หารด้วย 2 ตามภาพที่ 53 ซึ่งจากหัวข้อที่ผ่านมาก็คือ 66.66 MHz เป็นความถี่ที่จะชักสัญญาณ ดังนั้นช่วงความถี่อยู่ที่ -33.33 MHz ถึง 33.33 MHz นั้นเอง โดยที่ความถี่58.32 MHz ที่เกิดขึ้นมาก็จะไม่เกี่ยวข้องกับการย้ายความถี่ในส่วนนี้ไปด้วย พิจารณาแค่ในช่วงความถี่ 8.34 MHz เท่านั้น



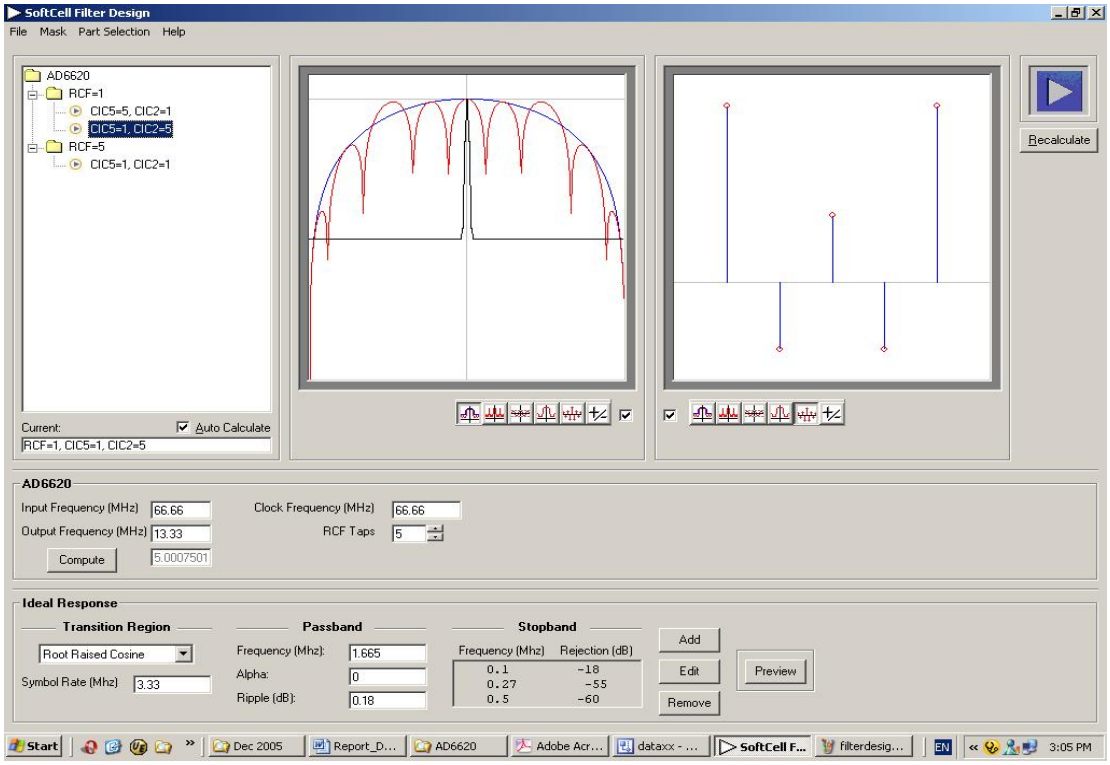
ภาพที่ 54 ความถี่ขาเข้าบอร์ด AD6620

ที่กล่าวมาทั้งหมดนี้จะเป็นลักษณะของความถี่ที่เข้ามา แต่สิ่งที่สำคัญไม่แพ้กันก็คือ รูปแบบของสัญญาณที่เข้ามาที่บอร์ด AD6620 ซึ่งจะไปเชื่อมต่อให้เข้ากับ AD6644 ซึ่งจะเห็นได้ว่าขาสัญญาณ Data input มีขนาดไม่เท่ากัน วิธีแก้ไขจึงกำหนดให้ 4 บิตบนของบอร์ด AD6620 เป็น “0” ไป รวมถึงส่วนขา A/B ก็ใส่ Vcc เข้าไปด้วยเพื่อให้สามารถใช้งานร่วมกันได้ โดยรูปแบบของสัญญาณจะเป็นไปตามภาพที่ 55



ภาพที่ 55 Full Rate Input Timing

ส่วนอีก 3 ส่วนที่ได้กล่าวถึงไปในข้างต้นนั้นจะใช้ซอฟต์แวร์ Digital Filter สำหรับออกแบบค่าพารามิเตอร์ต่างๆ ให้เหมาะสมตามภาพที่ 56



ภาพที่ 56 การคำนวณพารามิเตอร์ต่างๆ ที่ใช้งานสำหรับ AD6620

ซึ่งเมื่อคำนวณออกมาแล้วพบว่าสามารถเลือกได้หลายทางเลือก แต่ว่าที่ใช้งานได้กับบอร์ด AD6620 มีแค่กรณีเดียวเท่านั้นคือ ค่า CIC2 = 5, CIC5 = 1, และ RCF = 1 นั้นเอง และที่ใช้ Output Frequency 13.33 MHz เนื่องจากทางภาครับสัญญาณดิจิตอลเบสแบนด์ใช้การเลือกค่าทุกๆ 4 Sample เพื่อตัดสินใจหาข้อมูลที่ดีที่สุด (มาจากส่วนการเข้าจังหวะสัญญาณ (Signal Recovery)) ดังนั้นเมื่อต้องส่งทุก ๆ 3.33 MHz ภาครับก็จะต้องรับที่ 13.33 MHz นั้นเอง

เมื่อได้พารามิเตอร์ที่จะใช้ทั้งหมดแล้ว สิ่งนึงที่สำคัญมากก็การนำพารามิเตอร์เหล่านี้ไปไว้ในตำแหน่งในฐานข้อมูลของบอร์ด โดยจะมีอยู่ 2 ทางในการนำค่าเหล่านี้เข้าไปคือโปรแกรมผ่านทางพอร์ต Parallel ซึ่งสะดวกต่อการใช้งาน แต่ไม่เหมาะสมกับการทำงานของระบบเนื่องจากจะต้องกำหนดผ่านคอมพิวเตอร์เท่านั้น และอีกทางนึงก็คือการโปรแกรมผ่านทางพอร์ต Serial ซึ่งต้องทำผ่านบอร์ด FPGA คล้ายๆ กับการกำหนดในบอร์ด AD9856 นั้นเอง แต่ทว่าการโปรแกรมนี้จะซับซ้อนมากกว่ามาก โดยรายละเอียดจะกล่าวถึงต่อไป

ก่อนที่จะกล่าวถึงรายละเอียดการโปรแกรมพารามิเตอร์ผ่านพอร์ต Serial ขอกล่าวถึงค่าพารามิเตอร์ทั้งหมดที่จำเป็นต้องโปรแกรมลงไปแสดงไว้ในตารางที่ 9 โดยที่ต้องสนใจก็จะเป็นค่า NCO SYNC CONTROL REGISTER, NCO_FREQ, NCO_PHASE_OFFSET, INPUT/CIC2 SCALE REGISTER, CIC5 SCALE REGISTER, OUTPUT/RCF CONTROL REGISTER, และ RCF ADDRESS OFFSET REGISTER ที่เป็นค่าที่เปลี่ยนแปลงได้ ส่วนค่าอื่นๆ ก็จะยึดตามค่าที่กำหนดไว้ใน Specification ของบอร์ด AD6620 และค่าเหล่านี้ก็ได้เลือกเอาไว้แล้วตามที่ได้อธิบายข้างต้น

ตารางที่ 9 Control Register and RAM Address in AD6620

Address	Bit Width	Name	Notation	Description
000–0FF	20	RCF Coefficient RAM		RCF Coefficient RAM
100–1FF	36	RCF Data RAM		RCF Data RAM
200–27F	0	Reserved		Reserved
300	8	MODE CONTROL REGISTER		0: SOFT_RESET[1] 1: Diversity Channel Real Input Mode 2: Single Channel Complex Input Mode 3: Sync Master/Slave[2] (Master = 1, Slave = 0) 7–4: Reserved
301	3	NCO CONTROL REGISTER		0: NCO Bypass (Bypass = 1, Active = 0) 1: Enable Phase Dither 2: Enable Amplitude Dither 7–3: Reserved
302	32	NCO SYNC CONTROL REGISTER	SYNC_MASK	Write: Sync Mask Shadow Read: Sync Mask
303	32	NCO_FREQ	NCO_FREQ	Channel Frequency for NCO Tuning
304	16	NCO PHASE_OFFSET		NCO Phase Offset
305	8	INPUT/CIC2 SCALE REGISTER		2–0: S_{CIC2} 3: Reserved 4: ExpInv 7–5: : ExpOff
306	8	$M_{CIC2} - 1$	$M_{CIC2} - 1$	CIC2 Decimation Minus One
307	5	CIC5 SCALE REGISTER	S_{CIC5}	4–0: S_{CIC5} 7–5: Reserved
308	8	$M_{CIC5} - 1$	$M_{CIC5} - 1$	CIC5 Decimation Minus One
309	4	OUTPUT/RCF CONTROL REGISTER	S_{OUT}	2–0: Output Scale Factor 3: Unique B Flag (Normal Mode = 0, Unique B Mode = 1) 7–4: Reserved
30A	8	$M_{RCF} - 1$	$M_{RCF} - 1$	RCF Decimation Minus One
30B	8	RCF ADDRESS OFFSET REGISTER	RCF_{OFF}	Filter Coefficient Address Offset
30C	8	$N_{TAPS} - 1$	$N_{TAPS} - 1$	Number of Taps Minus One
30D	8	Reserved (Should Be Written 0)		Reserved (Should Be Written 0)

NOTES
[1]This bit is set high on $\overline{\text{RESET}}$. The chip is held into SOFT_RESET until it is written low.
[2]This bit is set low on $\overline{\text{RESET}}$. This keeps multiple AD6620 SYNC Masters from driving each other.

ต่อไปจะกล่าวถึงการป้อนค่าส่วนการป้อนค่าพารามิเตอร์ให้ AD6620 โดยป้อน 3-Bit Address (Ext_A[2:0]) และ 8-Bit Data (Ext_D[7:0]) ให้กับ External Interface Registers เพื่อใช้ในการเข้าถึง Control Register ภายในของ AD6620

External Interface Registers ตามตารางที่ 10 ประกอบด้วย 8 Address คือ

- Ext_A = 0-4 แทน DR0-DR4 เป็นรีจิสเตอร์ข้อมูล ซึ่งความยาวของข้อมูลแต่ละชนิดอาจใช้จำนวนบิตไม่เท่ากันที่ได้กล่าวไว้ในตารางที่ 9
- Ext_A = 5 ไม่ใช้งาน
- Ext_A = 6 แทน Low Address Register (LAR) A[7:0] ใช้กำหนด Address
- Ext_A = 7 แทน 1-0: Address Mode Register (AMR) A[9:8] ใช้กำหนด Address ดังนี้

5-2: ไม่ใช้งาน

6: Read increment ใช้เลื่อน Address เพิ่มขึ้นอัตโนมัติ หลังจากมีการอ่านข้อมูลภายใน

7: Write increment ใช้เลื่อน Address เพิ่มขึ้นอัตโนมัติหลังจากมีการเขียนข้อมูลภายใน

ตารางที่ 10 External Interface Register

Ext_A[2:0]	Name	Comment
000	Data Register 0 (DR0)	D[7:0]
001	Data Register 1 (DR1)	D[15:8]
010	Data Register 2 (DR2)	D[23:16]
011	Data Register 3 (DR3)	D[31:24]
100	Data Register 4 (DR4)	D[35:32]
101	Reserved	Reserved
110	Low Address Register (LAR)	A[7:0]
111	Address Mode Register (AMR)	1-0: A[9:8]
		5-2: Reserved
		6: Read Increment
		7: Write Increment

การเข้าถึงรีจิสเตอร์ภายใน เริ่มต้นด้วยการป้อนค่า Address ผ่านทาง AMR (Address Mode Register) A[9:8] และ LAR (Low Address Register) A[7:0] เพื่อเลือกชนิดของพารามิเตอร์ที่ต้องการกำหนด แล้วจึงป้อนค่าข้อมูลตามชนิดของข้อมูลนั้นๆ ตามตารางที่ 10 และได้อธิบายการป้อนพารามิเตอร์ด้วยการยกตัวอย่างดังต่อไปนี้

ตัวอย่างการป้อนค่าลงใน NCO frequency register (32-bits wide) โดยสมมุติว่าต้องการป้อนค่า NCO_FREQ ที่มีค่าเป็น 0xCBEFEFFF โดย NCO register จะอยู่ที่ Internal Address (A[9:0]) = 0x303

1. เซ็ทค่า AMR (A[9:8]) = 0x03 ด้วยการป้อน Ext_A[2:0] = 7 และ Ext_D[7:0] = 0x03
2. เซ็ทค่า LAR (A[7:0]) = 0x03 ด้วยการป้อน Ext_A[2:0] = 6 และ Ext_D[7:0] = 0x03

เนื่องจาก NCO_FREQ มีความกว้าง 32 บิต ดังนั้นจึงเซ็ทค่าผ่าน DR3 จนถึง DR0 เท่านั้น

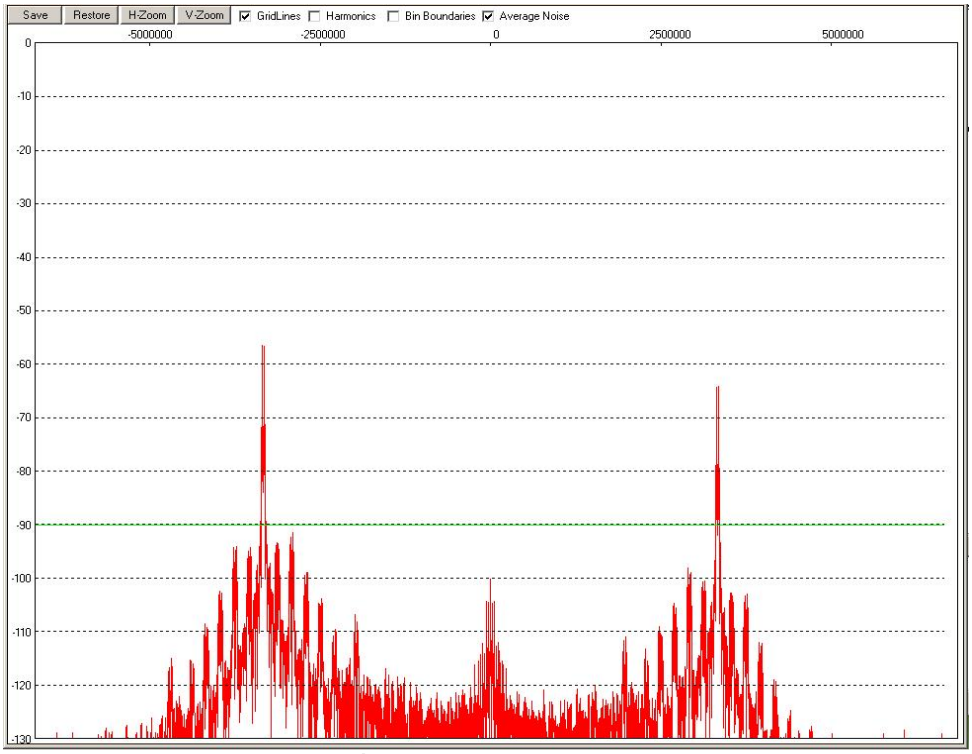
3. เซ็ทค่า DR3 = 0xCB ด้วยการป้อน Ext_A[2:0] = 3 และ Ext_D[7:0] = 0xCB
4. เซ็ทค่า DR2 = 0xEF ด้วยการป้อน Ext_A[2:0] = 2 และ Ext_D[7:0] = 0xEF
5. เซ็ทค่า DR1 = 0xEF ด้วยการป้อน Ext_A[2:0] = 1 และ Ext_D[7:0] = 0xEF
6. เซ็ทค่า DR0 = 0xFF ด้วยการป้อน Ext_A[2:0] = 0 และ Ext_D[7:0] = 0xFF

ซึ่งการเขียน DR0 จะทำให้ข้อมูลเหล่านี้ถูกส่งไปยังรีจิสเตอร์ภายใน นั่นหมายความว่าต้องเขียน DR3, DR2, DR1 ไปก่อน จากนั้นจึงเขียน DR0 ซึ่งจะทำให้ไม่สามารถที่จะเขียนต่ำแหน่งของข้อมูลที่ DR0 ก่อนได้มิฉะนั้นข้อมูลที่ DR อื่นๆ จะหายไปไม่สามารถส่งได้นั้นเอง

ในขณะที่ป้อนค่าเหล่านี้สิ่งนึงที่ต้องคำนึงก็คือ จำเป็นจะต้องกำหนดพอร์ต S/P ให้เป็น “0” เพื่อที่จะเป็นขา Serial และเมื่อป้อนค่าเสร็จแล้ว ก็จะต้องเปลี่ยนให้เป็น “1” เพื่อให้เป็นขา Parallel เพื่อที่จะส่งค่าเอาพุทไปยังส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ในบอร์ด FPGA นั้นเอง จะเห็นได้จาก ที่เป็นบอร์ดพัฒนา AD6620 ที่ขาทั้งสองส่วนนี้ใช้งานร่วมกัน

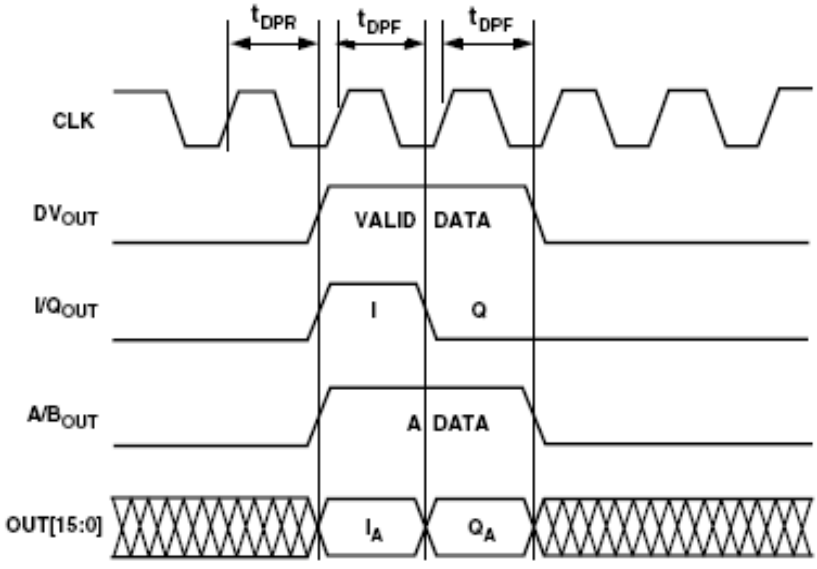
ภาพที่ 57 AD6620 Digital Receive Signal Processor

โดยพัฒนา VHDL สำหรับควบคุมการโปรแกรมค่าพารามิเตอร์ตามที่ได้อธิบายไว้ในข้างต้น และก็ได้มีการทดสอบผ่านพอร์ต Parallel (หรือจะเรียกว่า Micro port) ควบคู่กันไป และได้สัญญาณออกมาดังภาพที่ 58 ซึ่งก็สามารถทำงานได้ถูกต้อง



ภาพที่ 58 สัญญาณดิจิตอลเบสแบนด์ที่ออกมาจาก AD6620

และสุดท้ายจะเป็นส่วนเอาพุทที่ออกมาจากบอร์ดพัฒนานี้ โดยทางผู้วิจัยได้เลือกใช้แบบ Single-Channel Mode เนื่องจากข้อมูลที่เข้าบอร์ดมาในลักษณะช่องสัญญาณเดียวที่มีข้อมูลทั้ง In-Phase และ Quadrature รูปแบบจะเป็นตามภาพที่ 59 และได้มีการจับสัญญาณจาก Logic Analyzer ก็ออกมาเป็นไปตามภาพที่ 60



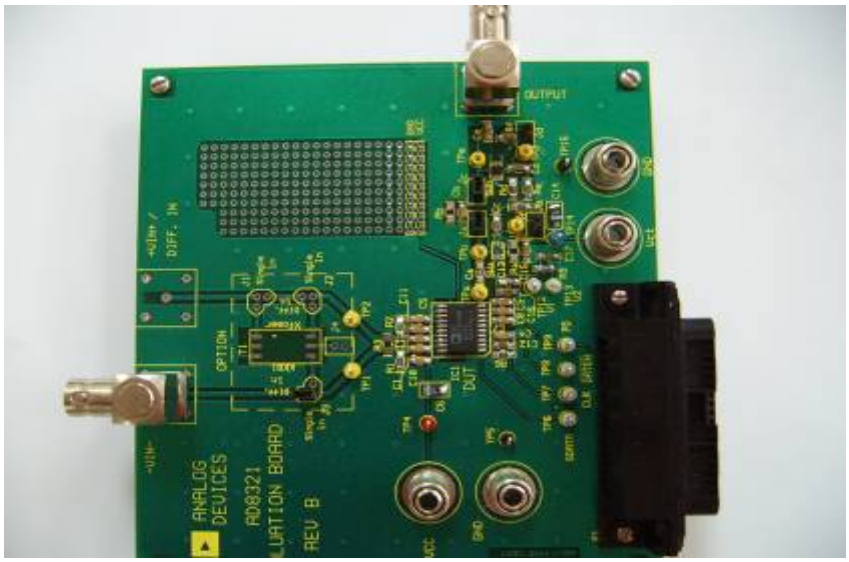
ภาพที่ 59 Parallel Output Data Timing (Single-Channel Mode)

ภาพที่ 60 ลักษณะของสัญญาณเอาพุทที่ได้ออกมาจาก AD6620

ดังนั้นที่บอร์ด FPGA ก็ต้องรับข้อมูลที่เป็นไปตามจังหวะดังกล่าว และแยกออกมาเป็นช่องสัญญาณ I และ Q แค่ช่องละ 6 บิท เนื่องจากส่วนการเข้าจังหวะรับข้อมูลแค่ 6 บิทเท่านั้น แต่ข้อมูลที่เข้ามามีทั้งหมด 16 บิท ซึ่งจะตัด 10 บิทล่างออก ใช้เฉพาะบิทบนเท่านั้น

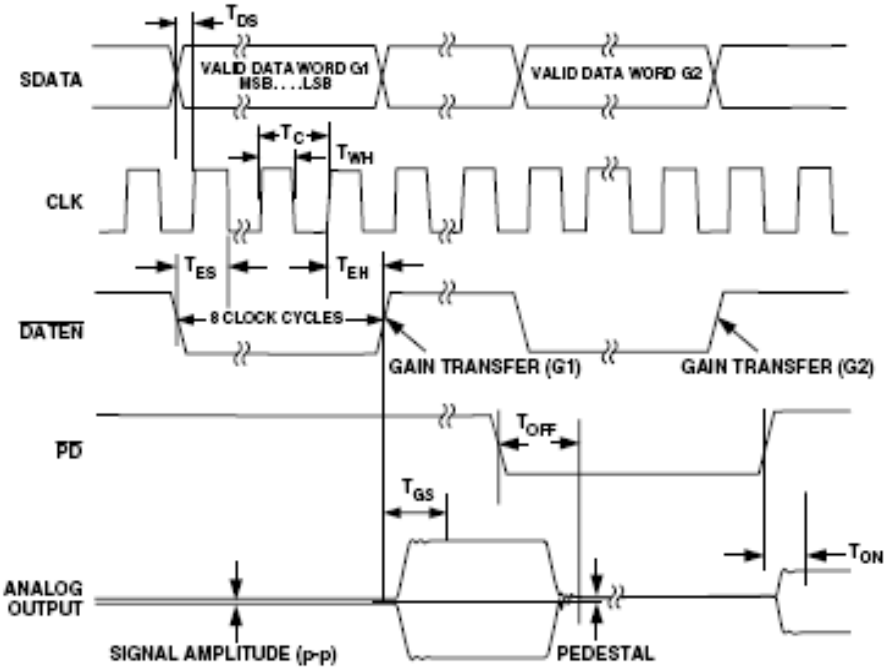
2.4.4 AD8321 Gain Programmable CATV Line Driver

บอร์ดพัฒนานี้จะใช้สำหรับการปรับพลังงาน หรือ Gain ไม่ให้สูงเกินไป หรือต่ำเกินไป เนื่องจากระบบที่พัฒนาขึ้นมานี้ ในส่วนการส่งข้อมูลเพื่อไปประมวลผลสัญญาณที่ส่วนการเข้าจังหวะสัญญาณนั้นจะใช้แค่เพียง 6 บิตในแต่ละช่องสัญญาณ จากที่ได้ออกมาเป็นสัญญาณดิจิตอล 16 บิต ดังนั้นเพื่อให้ 6 บิตที่ได้มานี้เป็นสัญญาณที่มีคุณภาพ และไม่ทำให้เกิดความผิดพลาดจนมากเกินไป จึงจำเป็นที่จะใช้ตัวบอร์ดพัฒนา AD8321 ในภาพที่ 61



ภาพที่ 61 AD8321 Gain Programmable CATV Line driver

โดยหลักการของตัวบอร์ดพัฒนานี้แล้วจะเห็นได้ว่าจะใช้การควบคุมสัญญาณอยู่ 4 สัญญาณคือ SDATA, CLK, DATAEN, PD โดยป้อนค่าอินพุทเข้าไปในลักษณะของสัญญาณอนาลอก ซึ่งรูปแบบของจังหวะในการป้อนจะเป็นดังภาพที่ 62 โดยส่วนที่ SDATA จะไปเกี่ยวพันกับค่าที่ใช้ในการปรับ Gain ค่า PD จะใช้สำหรับเอาพุทเป็นสัญญาณที่บ่งบอกให้เอาพุทออกจากบอร์ด AD8321 ส่วน DATAEN จะใช้ควบคู่กับ SDATA

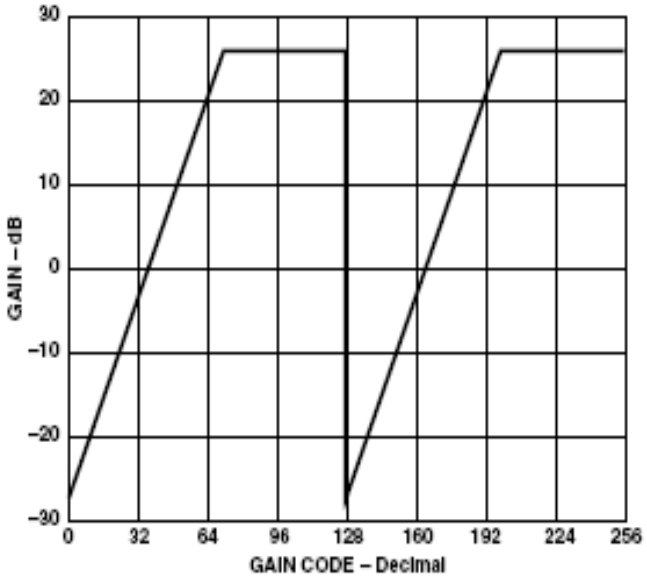


ภาพที่ 62 Serial Interface Timing

จากที่ได้กล่าวไปในข้างต้นเกี่ยวกับค่า SDATA ซึ่งจะไปสัมพันธ์กับ Gain เป็นไปตามสมการข้างล่างนี้

$A_V = 26\ dB - ((71 - CODE) \times 0.7526\ dB)$ for $CODE \leq 71$
$A_V = 26\ dB$ for $71 \leq CODE \leq 127$
$A_V = 26\ dB + ((199 - CODE) \times 0.7526\ dB)$ for $128 \leq CODE \leq 199$
$A_V = 26\ dB$ for $199 \leq CODE \leq 255$

โดยค่า CODE ก็คือค่า SDATA (7:0) ที่จะเก็บค่าในช่วงที่ DATAEN เป็น “1” 8 CLK ติดต่อกัน และจากค่า Gain นี้สามารถดูความสัมพันธ์ได้จาก ที่เป็นกราฟแสดงความสัมพันธ์ระหว่าง Gain (dB) กับ Gain Control (หรือ Code ในสมการข้างต้น)



ภาพที่ 63 Linear-In dB Gain VS. Gain Control

ดังนั้นแนวทางการพัฒนาในส่วนนี้จะพัฒนาซอฟต์แวร์ด้วยภาษา VHDL ที่ทำงานบนบอร์ด FPGA ให้เป็นไปตามจังหวะที่ได้กล่าวไป โดยส่วนที่นำมาควบคุมจะใช้เทคนิกการหาช่วงกำลังงานที่ยอมรับได้ก่อนที่จะเข้าไปในส่วนการเข้าจังหวะ เพื่อนำข้อมูลมาประมวลผลกลับ (Loop Back) ไปที่ AD8321 ให้ปรับความแรงของสัญญาณตาม รวมถึงในส่วนของ AD6644 Digital to Analog Converter ที่จะมีส่วน Over Flow Bit ที่จะสามารถนำมาช่วยปรับลด Gain ที่เข้ามาได้อีกด้วย

อุปกรณ์และวิธีการ

อุปกรณ์

1. คอมพิวเตอร์ส่วนบุคคล Pentium 4 , 2.4 GHz , 1GB DDR-RAM, 60 GB HDD
2. โปรแกรม Quartus และ Xilinx ISE 7.1
3. โปรแกรม ModelSim
4. โปรแกรม Matlab
5. บอร์ดพัฒนา FPGA Stratix EP1S25F672C6 ของบริษัท Altera และบอร์ดพัฒนา FPGA Xilinx
6. บอร์ดพัฒนาจากบริษัท Analog Device AD9856, AD8321, AD6644, และ AD6620

วิธีการ

1. พัฒนาฟังก์ชันบล็อกแต่ละฟังก์ชัน

ฟังก์ชันบล็อกของส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ในระบบดาวเทียมสื่อสารอเนกประสงค์ขนาดเล็กนี้จะประกอบไปด้วย CRC, Turbo Code, Interleaver สำหรับส่วนประมวลผลดิจิตอลเบสแบนด์ซึ่งอ้างอิงกับมาตรฐาน 3GPP รวมถึงการพัฒนาส่วนเชื่อมต่อกับซอฟต์แวร์ประยุกต์ โดยที่กล่าวมาทั้งหมดจะถูกพัฒนาขึ้นด้วยเทคโนโลยี FPGA/VHDL ส่วนซอฟต์แวร์ประยุกต์จะถูกพัฒนาขึ้นด้วย Visual C/C++

2. การทดสอบการเข้ารหัสและถอดรหัสสัญญาณ

เป็นการทดสอบความถูกต้องในการประมวลผลและการทำงานร่วมกันของส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ ซึ่งประกอบด้วย CRC, Channel coding (Turbo code) และ Interleave แล้วป้อนกลับไปให้ตัวถอดรหัส โดยเลือกทดสอบประสิทธิภาพของตัวเข้ารหัสถอดรหัสผ่านการจำลองช่องสัญญาณแบบ BSC (Binary Symmetric Channel) รวมถึงการรับส่งข้อมูลกับส่วนซอฟต์แวร์ประยุกต์

3. <u>การทดสอบส่วนการเข้าจังหวะสัญญาณ</u>

เป็นการทดสอบการเข้าจังหวะสัญญาณเพื่อแก้ไขปัญหาที่เกิดจากสภาพแวดล้อมต่างๆ รวมถึงการไม่เข้าจังหวะกันระหว่างฝั่งส่งและฝั่งรับสัญญาณ รวมถึงส่วนที่เกิดปัญหามากที่สุดสำหรับการสื่อสารผ่านดาวเทียมที่มีวงโคจรต่ำก็คือ การเกิดปัญหาความถี่เลื่อน อีกทั้งยังมีการแก้ปัญหาระดับของสัญญาณที่ไม่เท่ากันทำให้เกิดความผิดพลาดในการประมาณค่าได้ง่าย ด้วยการควบคุมกำลังงานของสัญญาณไม่ให้สูงเกินไปหรือต่ำเกินไป โดยแบ่งได้เป็นดังนี้คือ

3.1 <u>พัฒนาฟังก์ชันการเข้าจังหวะสัญญาณ</u>

ที่พัฒนาจะเป็นการเข้าจังหวะสัญญาณโดยแบ่งได้เป็น 7 ฟังก์ชั่นนั้นก็คือ การประมาณความถี่แบบหยาบ ที่จะใช้การประมาณ FFT เข้ามาช่วยในการทำงานในการคำนวณความถี่ สองจะเป็นตัวกรองสัญญาณ ที่จะช่วยลดความผิดพลาดออกไปส่วนหนึ่ง และสามจะเป็นการเข้าจังหวะเวลาเพื่อแก้ปัญหาในเรื่องความไม่ตรงของเวลาในการรับส่งข้อมูล และสี่จะเป็นตัวประมาณความถี่แบบละเอียด ซึ่งจะทำให้ความถี่เลื่อนลดลงไปหรือหมดไปในกรณีที่ไม่มากเท่าไรจากนั้นตัวที่ห้าจะเป็นตัวประมาณเฟสตกค้าง และหกจะเป็นตัวเข้าจังหวะเฟรมเพื่อแก้ปัญหาที่เรียกว่าความกำกวมของเฟส และสุดท้ายจะเป็นตัวควบคุมสัญญาณไม่ให้สูงหรือต่ำจนเกินไป

3.2 <u>การทดสอบการทำงานของส่วนเข้าจังหวะสัญญาณ</u>

การทดสอบสัญญาณการเข้าจังหวะด้วยซอฟต์แวร์ต่างๆ ไม่ว่าจะเป็น Matlab เพื่อตรวจสอบความเป็นไปได้ของการประมาณความถี่ และอีกส่วนหนึ่งจะเป็นการทดสอบการทำงานด้วย ซอฟต์แวร์ ModelSim เพื่อทดสอบความถูกต้องของซอฟต์แวร์ที่ได้พัฒนาขึ้นมา และประมาณความถี่ที่สามารถแก้ไขได้ รวมถึงเฟสและเวลาด้วย

4. <u>การทดสอบการทำงานของส่วนติดต่อภาคสัญญาณวิทยุ</u>

เป็นการทดสอบการทำงานของบอร์ดพัฒนาที่ได้เลือกใช้จากบริษัท Analog Device ต่างๆ รวมถึงการควบคุมพารามิเตอร์ต่างๆ ในทุกๆ ส่วนของบอร์ดพัฒนา โดยมีการทดสอบการป้อนสัญญาณลงไปและตรวจสอบการนำกลับมาของข้อมูล (นั้นคือการทดสอบการทำงานแบบย้อนกลับ)

5. <u>การทดสอบการประมวลผลวนกลับ (Loopback)</u>

เป็นการทดสอบการเข้ารหัส ถอดรหัส และการทำงานร่วมกัน ระหว่างส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ ส่วนการเข้าจังหวะสัญญาณ รวมถึงส่วนติดต่อภาคสัญญาณวิทยุ โดยให้มีการประมวลผลย้อนกลับที่ความถี่กลาง ซึ่งจะใช้การป้อนข้อมูลจากซอฟต์แวร์ประยุกต์และประมวลผลย้อนกลับมาที่เดิมนั้นเอง

6. <u>สถานที่ทำการทดลองและระยะเวลาในการทดลอง</u>

สถานที่ทำการทดลอง คือ SCORPion Lab ชั้น 6 ห้อง 603/R3 ตึก 60 ปี คณะวิศวกรรมศาสตร์ มหาวิทยาลัย เกษตรศาสตร์ แขวงลาดยาว เขตจตุจักร กรุงเทพฯ 10900 รวมระยะเวลาที่ใช้ในการวิจัยทั้งสิ้น 18 เดือนตั้งแต่เดือน ธันวาคม 2547 ถึง มีนาคม 2549

ผลและวิจารณ์

<u>การทดสอบระบบการสื่อสารข้อมูลสำหรับสถานีภาคพื้นดิน</u>

การทดสอบระบบสื่อสารข้อมูลที่จะกล่าวนี้ จะเป็นการทดสอบกระบวนการทำงานต่างๆ ในส่วนสื่อสารข้อมูล ไม่ว่าจะเป็นส่วนดิจิตอลเบสแบนด์และซอฟต์แวร์ประยุกต์ ส่วนการเข้าจังหวะสัญญาณ และสุดท้ายคือการทดสอบส่วนภาคติดต่อสัญญาณวิทยุ ซึ่งก็จะเป็นการทดสอบรวมทั้งระบบด้วย

ดังนั้นเพื่อให้เกิดความเข้าใจในการทดสอบในแต่ละส่วนการทำงาน จึงขอแบ่งการทดสอบออกเป็น 3 ส่วนใหญ่ๆ คือ การทดสอบต้นแบบส่วนประมวลผลดิจิตอลเบสแบนด์และซอฟต์แวร์ประยุกต์ การทดสอบส่วนเข้าจังหวะสัญญาณ และสุดท้ายเป็นการทดสอบรวม (มีภาคติดต่อกับสัญญาณวิทยุ) โดยใน 2 ส่วนแรกการทำงานอยู่บนบอร์ด FPGA แต่ส่วนการติดต่อกับภาคสัญญาณวิทยุจะใช้อุปกรณ์มาตรฐาน และติดต่อกับส่วน FPGA อีกทีนึง และส่วนซอฟต์แวร์ประยุกต์ที่ทำงานอยู่บนคอมพิวเตอร์หรือ Notebook นั้นก็จะติดต่อผ่านบอร์ด Ezy-USB 1.1 ซึ่งจะติดต่อกับคอมพิวเตอร์ผ่านพอร์ต USB และติดต่อกับ FPGA ผ่านพอร์ตพาราแรลทีละ 8 บิท

ส่วนที่สำคัญที่สุดของการทดสอบนี้ก็คือการที่ส่งไฟล์ข้อมูลต่างๆ จากซอฟต์แวร์ประยุกต์มาทำการประมวลผลดิจิตอลเบสแบนด์ เพื่อความคงทนต่อสัญญาณรบกวนในช่องสัญญาณ และส่งไปที่ส่วนภาคติดต่อกับสัญญาณวิทยุ ที่ความถี่ประมาณ 70 MHz (สำหรับการทดสอบตามที่ได้กล่าวไว้ในข้างต้น) และรับข้อมูลกลับมาโดยมีการปรับขนาดของสัญญาณให้คงที่ การแปลงสัญญาณให้เป็นดิจิตอล และย้ายความถี่ลงมาที่เบสแบนด์ เพื่อทำการเข้าจังหวะต่างๆ ไม่ว่าจะเป็นเวลาที่ไม่ตรงกัน ความถี่เลื่อน หรือเฟสตกค้างที่เกิดขึ้นมา หลังจากนั้นก็จะเป็นการถอดรหัสสัญญาณและส่งข้อมูลกลับไปที่ซอฟต์แวร์ประยุกต์เพื่อแสดงผลออกหน้าจอที่ติดต่อกับผู้ใช้งาน

1. <u>การทดสอบส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์</u>

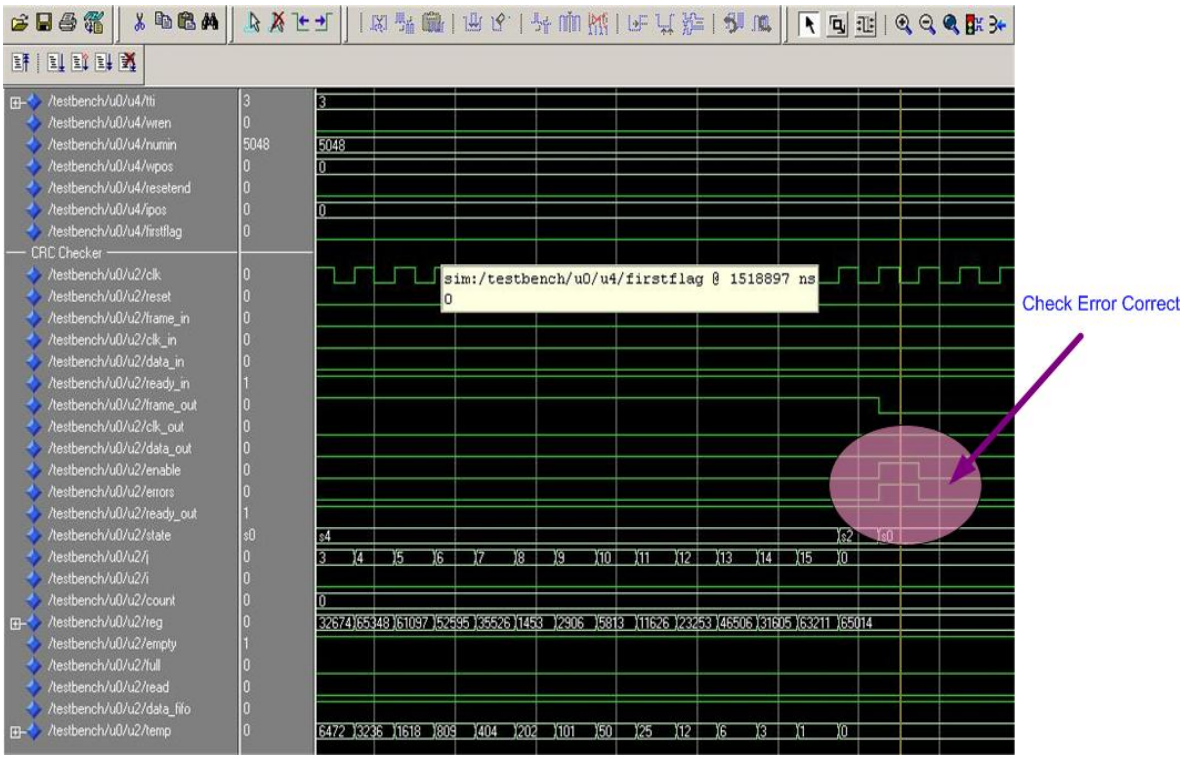
การทดสอบต้นแบบส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์จะแบ่งการทำงานได้เป็น 3 เฟสใหญ่ๆ เพื่อให้สามารถทดสอบการทำงานได้ทันที่เวลาจริง และมีความถูกต้อง ดังต่อไปนี้

- เฟสที่ 1 ทดสอบการประมวลผลสัญญาณในแต่ละฟังก์ชั่นการทำงาน
- เฟสที่ 2 ทดสอบการประมวลผลสัญญาณแบบวนกลับ (Loop Back)
- เฟสที่ 3 การพัฒนาต้นแบบการประมวลผลสัญญาณดิจิตอลเบสแบนด์

ในการทดสอบในแต่ละเฟส ก็จะพบความไม่แน่นอนในการจำลองการทำงานบนซอฟต์แวร์กับการทดสอบกับฮาร์ดแวร์จริง ซึ่งความไม่แน่นอนนี้เกิดจาก LPM (Library of Parameterized Modules) เช่น FIFO และ Memory ที่แสดงผลแตกต่างกัน และการส่งข้อมูลระหว่างบอร์ดต่างๆ ซึ่งทำให้สัญญาณที่ส่งออกไปเกิดความลดทอน หรือเกิดสัญญาณรบกวนขึ้น โดยปัญหาเหล่านี้จะได้กล่าวถึงอีกครั้งอย่างละเอียดในส่วนต่อไป

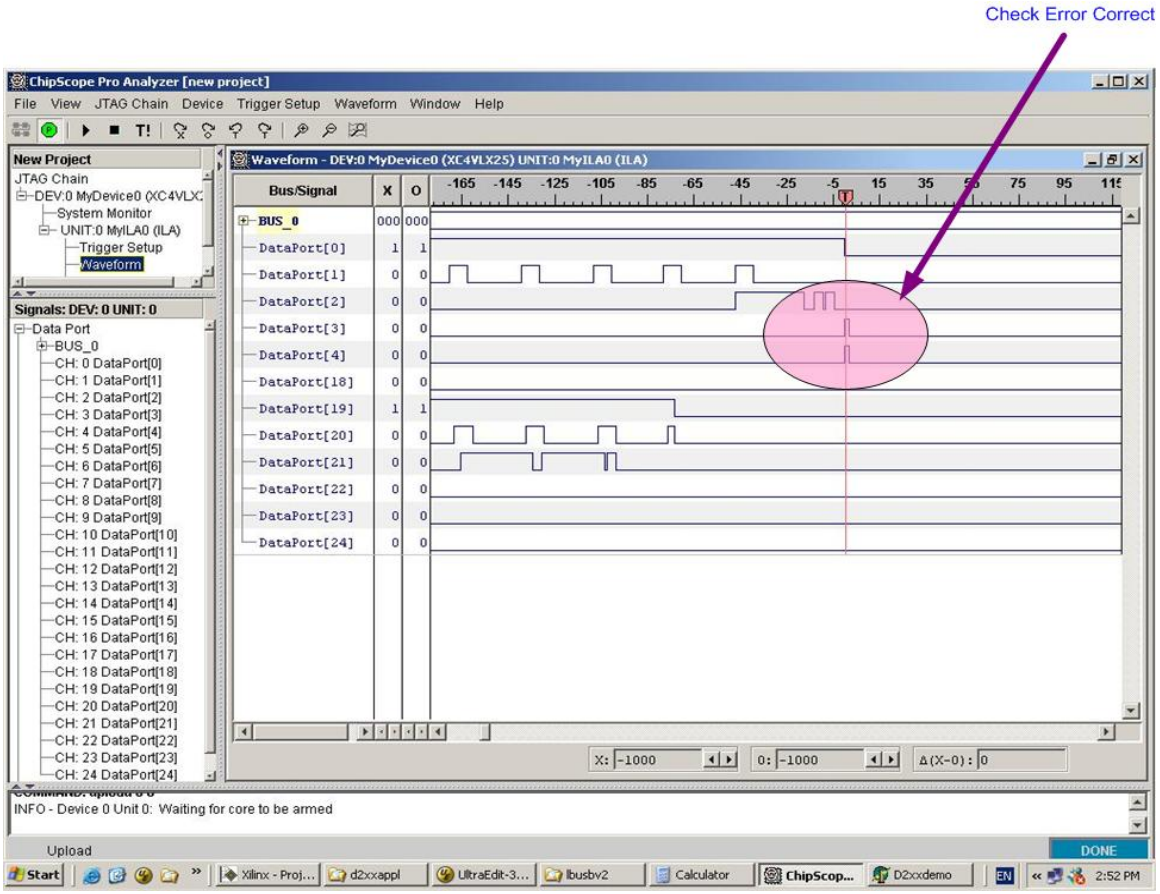
1.1 การทดสอบการประมวลผลสัญญาณในแต่ละฟังก์ชั่น

ในการทดสอบการประมวลผลสัญญาณจะแบ่งการทดสอบเป็นการจำลองการทำงาน เพื่อทดสอบความถูกต้องของฟังก์ชั่นต่างๆ เป็นคู่ๆ แล้วก็ค่อยๆ เพิ่มเข้ามาจนครบการประมวลผลสัญญาณดิจิตอลเบสแบนด์ นั้นคือ จะทดสอบ CRC Encoder คู่กับ CRC Checker แล้วก็ค่อยๆ เพิ่ม Turbo Encoder กับ Turbo Decoder จากนั้นก็จะเป็น Interleaver กับ Deinterleaver และสุดท้ายจะเป็นการใส่วงจร Noise ที่สร้างจำลองช่องสัญญาณ BSC (Binary Symmetric Channel) เข้าไปเพื่อให้สามารถทดสอบในเฟสที่สองได้ โดยจะใช้โปรแกรม Modelsim V6.0 ตรวจสอบสัญญาณ และปัญหาที่พบมากในการแก้ไขในแต่ละวงจรก็คือ การอ่านข้อมูลจาก FIFO และจาก RAM ไม่ถูกต้อง การกำหนดสัญญาณที่ผิดพลาด และการกำหนด State การทำงานในแต่ละวงจรที่ผิดพลาด ทำให้วงจรไม่สามารถที่จะทำงานได้ (สัญญาณค้าง) ซึ่งเสียเวลาในการตรวจสอบและแก้ไขเพื่อให้สามารถทำงานได้ถูกต้องตามภาพที่ 64 ซึ่งแสดงให้เห็นว่าสามารถตรวจสอบความผิดพลาดที่ CRC Checker ได้ถูกต้อง นั้นคือ ไม่มีความผิดพลาดเกิดขึ้น (ขา Enable และ Errors ขึ้น “1” พร้อมกัน)



ภาพที่ 64 ผลการ Simulation Waveform ของส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์

และหลังจากนั้นได้นำ Code VHDL ซึ่งตรวจสอบการทำงานที่ถูกต้องแล้วมาทดสอบประมวลผลจริงบนบอร์ด FPGA โดยจะใช้โปรแกรม ChipScope Pro 7.1 ซึ่งสามารถดึงสัญญาณจากฮาร์ดแวร์ได้ และตามหลักการแล้วจะการทดสอบในส่วนนี้จะต้องได้ผลตามการทดสอบการจำลองการทำงาน แต่ก็พบปัญหานั้นก็คือ Timing ของ LPM ต่างๆ เช่น FIFO เมื่อ Reset แล้วจะเสียเวลาไปประมาณ 5 Clock ถึงจะสามารถทำงานได้ตามการจำลองไว้บน Modelsim ดังนั้นทำให้ผู้วิจัยจำเป็นต้องแก้ไขสัญญาณบางตัวให้มีค่าหน่วงให้มากขึ้น เพื่อให้สามารถทำงานได้ถูกต้องตามที่ควรจะเป็นจนสุดท้ายถูกต้องตามภาพที่ 65



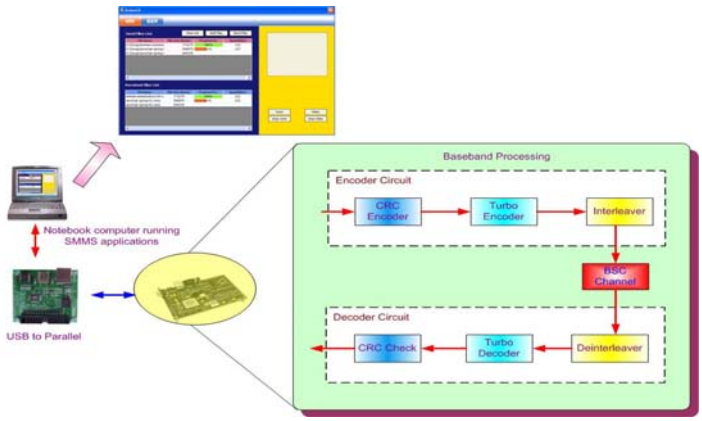
ภาพที่ 65 สัญญาณที่ได้จากการประมวลผลสัญญาณบนบอร์ด FPGA

หลังจากนั้นจะเป็นการเขียนโปรแกรมเพื่อให้สามารถรับส่งข้อมูลกับบอร์ด EZ-USB โดยหลักการตามที่ได้กล่าวไว้ในหัวข้อการพัฒนา และส่วนที่สำคัญของโปรแกรมนี้ คือในการ Read/Write ข้อมูลกับบอร์ด USB ซึ่งทำให้ความเร็วในการส่งข้อมูลเหลืออยู่เท่ากับ ความเร็วของสัญญาณนาฬิกา÷20 และจังหวะการเปลี่ยนแปลง Read/Write โดยในขั้นต้นใช้วิธี Write แทรก Read นั้นคือถ้ากำลังอ่านข้อมูลจาก FIFO ใน USB อยู่แล้วมีสัญญาณให้ Write ข้อมูลก็จะมาทำ Write ให้เสร็จสิ้นก่อนแล้วถึงจะกลับไป Read Process ได้ แต่เมื่อนำมาทดสอบกับซอฟต์แวร์ประยุกต์แล้วผลปรากฏว่าจะทำให้ซอฟต์แวร์ทำงานค้าง จึงได้แก้ไขให้ทำอย่างหนึ่งอย่างใดให้เสร็จทั้ง Process ก่อนถึงจะไปทำอีก Process ได้หมายความว่า เมื่อ Read ข้อมูลจาก FIFO ก็จะทำงานให้เสร็จก่อน ถึงแม้ว่าจะมีสัญญาณให้ Write เข้ามา ดังนั้น Write Process จะทำงานได้ก็ต่อเมื่อวงจร Read ทำงานเสร็จสิ้นแล้วนั้นเอง ซึ่งในส่วนนี้ก็สามารถทำงานได้ถูกต้องตามจังหวะของสัญญาณที่ใช้ในการเชื่อมต่อระหว่างทั้งสองบอร์ด

และสุดท้ายเมื่อได้พัฒนาในส่วนฟังก์ชั่นในแต่ละส่วนเรียบร้อบแล้วก็จะเป็นการทดสอบการทำงานในเฟสที่สอง ซึ่งเป็นการทดสอบการประมวลผลย้อนกลับ จะกล่าวโดยละเอียดในหัวข้อถัดไปดังนี้

1.2 การประมวลผลดิจิตอลเบสแบนด์แบบย้อนกลับ

ในการทดสอบการทำงานในส่วนนี้เป็นการทดสอบว่าฟังก์ชั่นที่พัฒนาในเฟสแรกสามารถทำงานได้ต่อเนื่องและมีความถูกต้องหรือไม่ โดยจะแบ่งการทดสอบออกเป็นสองส่วนก็คือการทดสอบการส่งไฟล์ข้อมูล กับการทดสอบประสิทธิภาพในการส่งข้อมูล โดยการประมวลผลสัญญาณจะเป็นแบบย้อนกลับ (Loopback) ในบอร์ด FPGA 1 บอร์ดนั้นเอง ตามภาพที่ 66

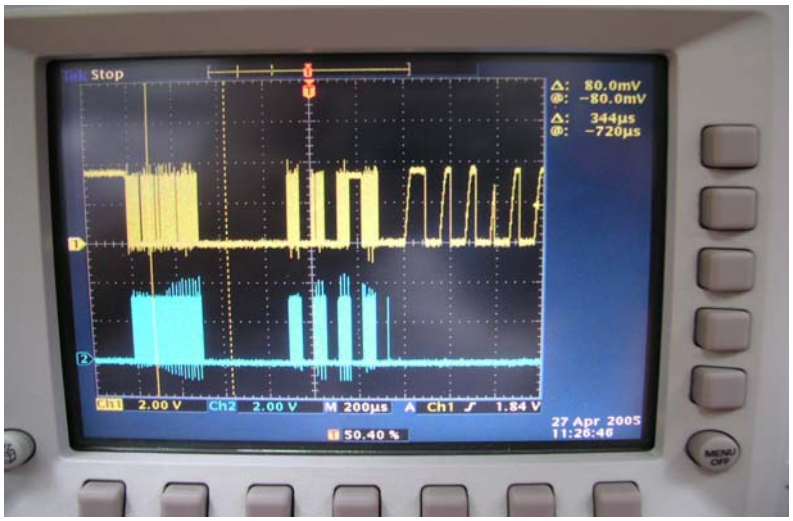


ภาพที่ 66 การทดสอบการประมวลผลสัญญาณดิจิตอลเบสแบนด์แบบวนกลับ (Loopback)

โดยในการประมวลผลสัญญาณในเฟสที่สองนี้จะพบปัญหาที่เกี่ยวข้องกับสายสัญญาณที่ไว้สำหรับการเชื่อมสัญญาณระหว่างบอร์ด เนื่องจากการส่งข้อมูลระหว่างบอร์ดมีความถี่ในการส่งข้อมูลที่สูง และอาจเกิดจากการรบกวนจากสายสัญญาณข้างเคียง ทำให้สัญญาณผิดเพี้ยนไป หรือทำให้สัญญาณมีการกระเพื่อมขึ้นมา โดยเฉพาะเมื่อจับสัญญาณจาก Logic Analyzer แล้วจะพบว่ามีการอ่านข้อมูลเกิน ทำให้เกิดการเลื่อนเฟรม และรับข้อมูลไม่ครบ ซึ่งได้แก้ไขดังต่อไปนี้

ในขั้นแรกได้ลองเปลี่ยนตัวสังเคราะห์วงจรจาก XST ในโปรแกรม Xilinx เป็น Leonardo Spectrum ผลปรากฏว่าสามารถ Read/Write ข้อมูลได้ถูกแต่ว่าวงจรถอดรหัส Turbo ทำงานผิดพลาด นั้นคือถอดรหัสผิดหมด ดังนั้นวิธีนี้จึงไม่สามารถแก้ปัญหาได้ เนื่องจากผิดพลาดที่อื่นต่อไป จึงได้เปลี่ยนตัวสังเคราะห์วงจรให้กลับมาเป็น XST เหมือนเดิมซึ่งสามารถทำงานได้ถูกต้อง แต่มีสัญญาณรบกวน

จากนั้นได้ต่อตัวต้านทาน (R) และตัวเก็บประจุ (C) เข้าไปในสายสัญญาณเฉพาะขา Read และ Write เพื่อกรองสัญญาณรบกวนออกไป ผลปรากฏว่าสัญญาณมีความคมไม่มีสัญญาณรบกวนตามภาพที่ 67 ข้อมูลไม่มีการเลื่อนและถูกต้องทุกบิตข้อมูล



ภาพที่ 67 สัญญาณ Read (สีฟ้า) และ Data (สีส้ม) หลังจากต่อ RC แล้ว

และได้ทำการทดสอบประสิทธิภาพของตัวเข้ารหัส ถอดรหัสตามที่ได้กล่าวในหัวข้อการพัฒนาซอฟต์แวร์ ซึ่งผลปรากฏว่าเมื่อ E_b / N_0 มากขึ้นแต่กลับเกิดความผิดพลาดที่สูงขึ้นซึ่งไม่น่าจะเกิดขึ้นมาได้ดังนั้นจึงได้ทดสอบวงจรว่าทำงานได้ถูกต้องหรือไม่ โดยในขั้นแรกได้ตรวจสอบวงจรโปรแกรม Probability of Error และวงจรสร้างสัญญาณรบกวน แต่ก็ไม่พบความผิดพลาดใดๆ ทั้งสองวงจรก็ทำงานถูกต้องตามที่ควรจะเป็น จึงได้ย้อนมากลับมาทดสอบในวงจรถอดรหัส โดยคาดว่าน่าจะเป็นที่ Turbo Decoder ที่ทำงานไม่ครบ Iterative เนื่องจากค่า Likelihood Threshold นั้นน้อยเกินไป หมายความว่าถ้าค่า Threshold มีค่าน้อยเกินไป เมื่อวงจร MAP Decoder ทำงานเสร็จสิ้นในแต่ละรอบก็จะให้ค่า Likelihood ออกมาเปรียบเทียบ ซึ่งค่านี้จะบ่งบอกถึงความเชื่อมั่นต่อบิตข้อมูลว่าเป็นบิตที่ถูกต้องหรือไม่ ดังนั้นถ้าค่า Likelihood ที่ใช้เปรียบเทียบน้อยเกินไป

ค่าที่ออกจากวงจรในรอบแรกๆ ก็อาจจะเกินค่านี้ และจบการทำงานของวงจรทันที แทนที่จะทำงานให้ครบตามจำนวนรอบที่ได้กำหนดไว้ ทำให้การถอดรหัสมีความผิดพลาดขึ้น ดังนั้นเพื่อเป็นการแก้ไขในความผิดพลาดที่อาจจะเกิดขึ้นอีกเมื่อ E_b / N_0 มีค่าสูงๆ ก็จะกำหนดค่า Likelihood Threshold ให้มีค่าที่สูงที่สุดที่เป็นไปได้ เพื่อให้วงจร MAP ทำงานครบตามจำนวนที่ได้กำหนดไว้ (นั้นคือกำหนดเอาไว้ที่ค่าที่สูงที่สุด) และผลปรากฏว่า BER ที่เกิดขึ้นในการทดสอบเป็นไปตามทฤษฎีดังที่ได้กล่าวไว้ในส่วนการพัฒนาซอฟต์แวร์ประยุกต์

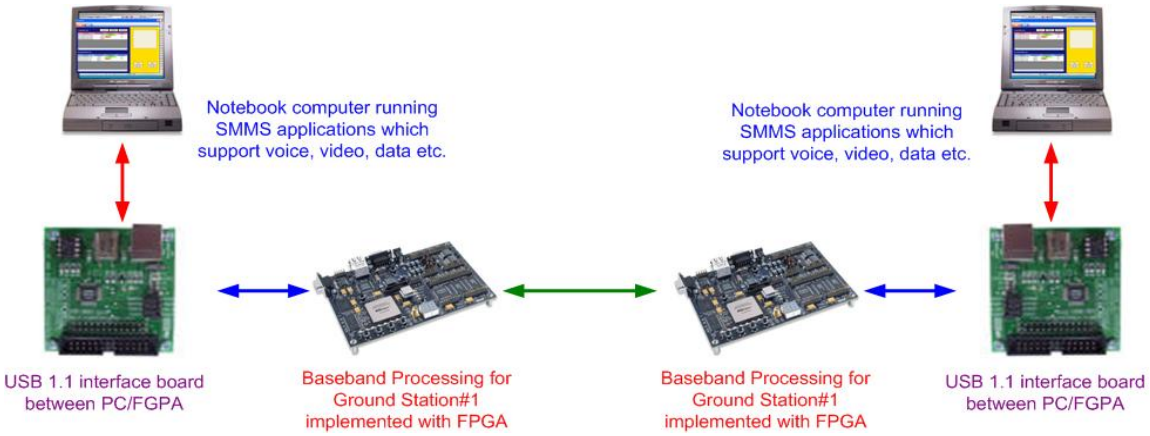
รูปร่างของบอร์ดต้นแบบสำหรับการประมวลผลแบบวนกลับได้ทำออกมา 3 เวอร์ชั่นโดยพัฒนาให้มีความสวยงาม และง่ายต่อการแก้ไขปรับปรุงให้มีสามารถขยายไปทดสอบในเฟสต่อไปได้ โดยไม่ต้องแก้ไขมากตามภาพที่ 68

และสุดท้ายจะเป็นทรัพยากรที่ใช้ในการประมวลผลแบบวนกลับโดยใช้ไป 3864 Slice หรือ 8694 Logic Cells ใช้หน่วยความจำรวม 99 Kbytes และความถี่สูงสุดที่สามารถทำงานได้คือ 141.24 MHz แต่ในความเป็นจริงแล้วที่ใช้ได้ถึง 141.24 MHz (ค่านี้เป็นค่าสัญญาณนาฬิกาที่อินพุทของวงจรลดทอนสัญญาณ) เนื่องจากต่อวงจรลดความถี่สัญญาณนาฬิกาเอาไว้เพื่อให้วงจร Turbo decoder สามารถทำงานได้ (ที่ความถี่สูงสุด 96.1 MHz) ซึ่งที่จริงแล้ววงจรจะทำงานได้สูงสุดไม่เกิน 96.1 MHz เท่านั้น

ภาพที่ 68 บอร์ดต้นแบบสำหรับการประมวลผลสัญญาณแบบวนกลับ

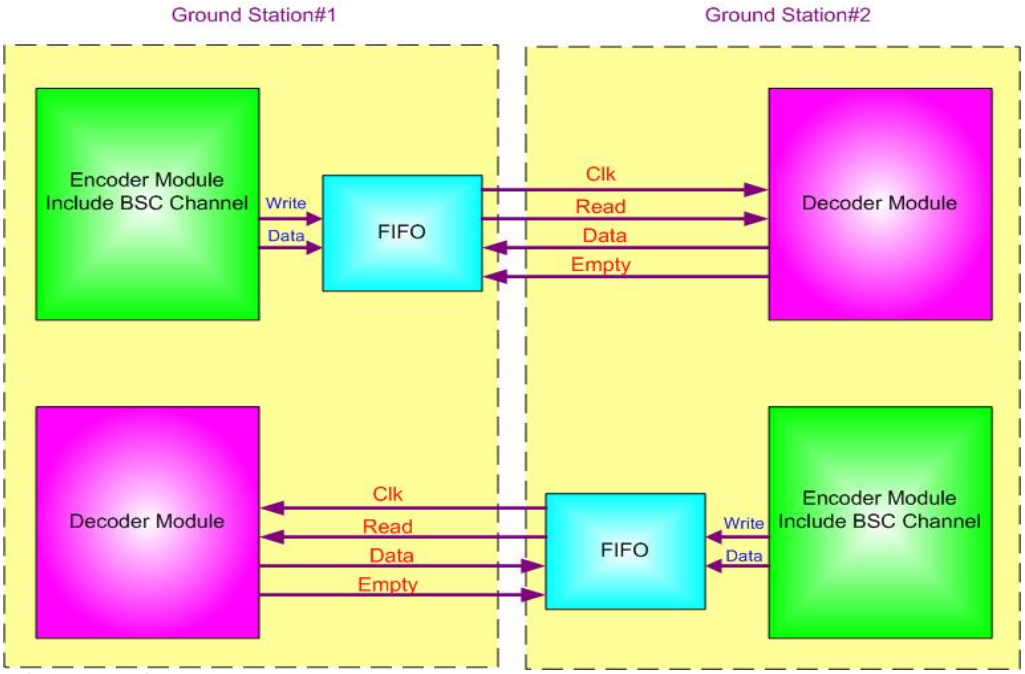
1.3 การพัฒนาต้นแบบการประมวลผลสัญญาณดิจิตอลเบสแบนด์

การพัฒนาในเฟสนี้จะเป็นการพัฒนาต้นแบบออกมา เพื่อจำลองการทำงานโดยเบื้องต้นของสถานีภาคพื้นดิน 2 สถานีตามภาพที่ 69 โดยจะมีการรับส่งข้อมูลจากซอฟต์แวร์ประยุกต์ที่รันบน Notebook หรือคอมพิวเตอร์ และส่งข้อมูลมาที่ส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์เพื่อเพิ่มความคงทนต่อสัญญาณรบกวน และจะจัดส่งออกไปให้แก่ภาควิทยุต่อไป แต่ในเบื้องต้นจะเป็นแค่การประมวลผลสัญญาณดิจิตอลเบสแบนด์เท่านั้น และจะมีการทดสอบหา BER (Bit Error Rate) ด้วยการจำลองช่องสัญญาณ BSC (Binary Symmetric Channel) ขึ้นมาเพื่อตรวจสอบ Coding Gain ที่ได้จากตัวเข้ารหัส ถอดรหัสที่ได้เลือกใช้ในการพัฒนา



ภาพที่ 69 ต้นแบบส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์

สิ่งที่ต้องคำนึงถึงเป็นอย่างมากในการทำงานในเฟสนี้ก็คือจะเลือกส่งสัญญาณใดและสายสัญญาณที่มีคุณภาพ เพื่อเชื่อมต่อระหว่างบอร์ด ปัญหาแรกนั้นเนื่องจากทั้งสองบอร์ดไม่ได้ใช้สัญญาณนาฬิกาอันเดียวกันเหมือนในเฟสที่สอง ดังนั้นก่อนที่จะส่งออกไปที่สายสัญญาณจำเป็นต้องมี FIFO เพื่อใช้ในการส่งข้อมูลโดยแยกสัญญาณนาฬิกาในการอ่านและเขียน เพื่อที่ข้อมูลที่ส่งออกไปจะได้เข้าจังหวะกับอีกบอร์ดนึง ในส่วนปัญหาที่สองเนื่องจากจะมีการส่งสายสัญญาณไปอีกบอร์ดนึงถึง 4 เส้น ในการทดสอบได้ต่อสายสัญญาณแต่ละเส้นคู่กับสายกราวด์ เพื่อให้ลดสัญญาณรบกวนที่มากวนจากเส้นข้างเคียง ดังนั้นฟังก์ชั่นการทำงานจะถูกแยกออกเป็น 2 โมดูลก็คือโมดูลเข้ารหัส (รวมถึงการจำลองช่องสัญญาณด้วย) กับโมดูลถอดรหัส โดยมี FIFO เป็นตัวกลางในการส่งข้อมูลระหว่างทั้งสองโมดูลตามภาพที่ 70



ภาพที่ 70 การเชื่อมต่อระหว่างบอร์ด

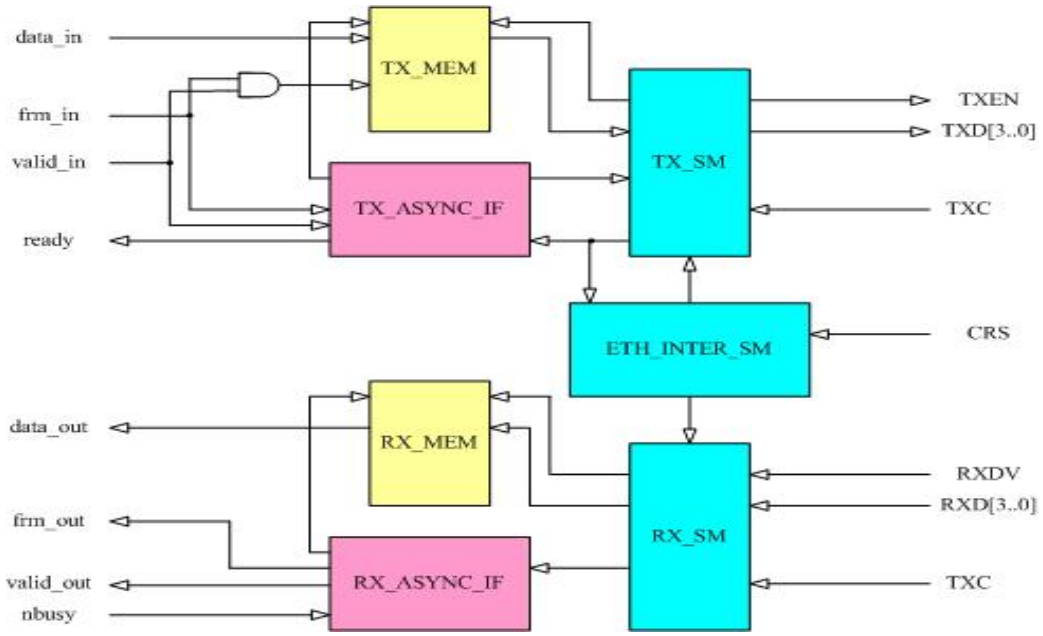
เมื่อได้ทดสอบตามที่ได้กล่าวมาทั้งหมดนี้จะสามารถรับส่งข้อมูลได้ แต่ว่าจะมีปัญหาเกิดขึ้นที่ฝั่งรับข้อมูลจะมีข้อมูลผิดอยู่เป็นบางบิต ซึ่งได้ค้นพบว่าน่าจะมีข้อมูลค้างอยู่ใน FIFO ซึ่งวิธีแก้ไขก็จะใช้การ Clear ข้อมูลใน FIFO ทุกๆ การส่งข้อมูลครบ 1 เฟรม ก็จะทำให้ระบบสามารถทำงานได้ถูกต้องรับส่งข้อมูลกันได้แบบ 2 ทิศทาง แต่เมื่อทดสอบความถูกต้องไปเรื่อยๆ ก็พบปัญหาในการส่งข้อมูล คือเกิดความผิดพลาดของข้อมูลอันจะเกิดขึ้นจากการกวนกันของสัญญาณที่ส่งออกมาที่พอร์ตที่ได้กำหนดไว้ในข้างต้น ดังนั้นจึงจำเป็นต้องปรับเปลี่ยนการส่งข้อมูลระหว่างบอร์ดออกไป โดยจะมีแนวทางแก้ปัญหาอยู่ 2 แนวทางนั้นก็คือการส่งข้อมูลให้เป็นแบบ LVDS (Low-Voltage Differential Signaling) กับการส่งข้อมูลผ่านพอร์ต Ethernet

การส่งข้อมูล LVDS จะเป็นการส่งข้อมูลที่นิยมกันมากในการส่งระยะทางไกล เนื่องจากจะมีการส่งข้อมูลที่อยู่ในลักษณะที่เป็นบวกกับลบ (การส่งสัญญาณออกไป 2 เส้นคู่กันโดยเส้นแรกจะส่งสัญญาณจริงออกไป และอีกเส้นจะส่งแบบกลับทิศทาง) เพื่อที่จะทำให้มีการหักลบสัญญาณที่รบกวนเข้ามาในฝั่งรับออกไป โดยจะใช้การส่งแบบนี้เฉพาะสัญญาณที่รับส่งระหว่างบอร์ดเท่านั้น มีทั้งหมด 8 คู่สัญญาณ ซึ่งพอร์ตที่กำหนดไว้มีไม่เพียงพอ จึงจำเป็นต้องใช้พอร์ตเพิ่มเป็นพอร์ตแบบ P160 และซอฟต์แวร์ของ Xilinx เองก็สามารถที่จะทำสัญญาณให้ออกมาเป็นแบบ LVDS ได้จึงทำให้เสียเวลาไม่มากในการเปลี่ยนแปลงแนวทางในการดำเนินการ และจากการทดสอบผลปรากฏว่าสามารถส่งข้อมูลได้จริง มีความถูกต้องแต่เมื่อส่งข้อมูลในปริมาณที่มากขึ้น มี

ข้อมูลบางเฟรมหายไป หรือมีความผิดพลาดเกิดขึ้นมาในเฟรมข้อมูล ได้มีการพยายามในการลดความเร็วในการส่งข้อมูล หรือทดสอบว่าขาสัญญาณเสียหรือไม่ แต่ก็ไม่ได้ผลที่ดีขึ้น เลยเปลี่ยนไปใช้การส่งแบบ Ethernet

ในการส่งแบบ Ethernet จะมีข้อจำกัดที่สำคัญนั้นคือ พอร์ตที่ให้มาจะไม่มีการควบคุมในระดับ MAC มาให้ซึ่งเป็นตัวควบคุมให้การส่งมีความถูกต้องแน่นอน (นั้นคือมีแต่การส่งในระดับ Physical มาให้เท่านั้น) จึงจำเป็นที่จะต้องพัฒนาการควบคุมการส่งสัญญาณเองเป็นไปตามมาตรฐานที่ได้ให้มากับบอร์ดพัฒนา ดังนั้นจะไม่ใช้รูปแบบที่ได้กล่าวไว้ แต่จะเป็นการส่งสัญญาณเข้า Module Ethernet ตามภาพที่ 71 โดยในสัญญาณในฝั่งทางซ้ายของรูป จะเป็นสัญญาญที่ใช้ในการส่งข้อมูลกับส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ ส่วนอีกฝั่งจะเป็นสัญญาณที่ส่งไปให้ชิปที่ควบคุมการทำงานของ Ethernet ดังนั้นการกำหนดขาสัญญาณในส่วนของการเชื่อมต่อระหว่างบอร์ดจะเปลี่ยนไปเป็นการเชื่อมต่อกับชิปที่ควบคุมการทำงานของพอร์ต Ethernet ซึ่งเป็นไปตามมาตรฐานที่กำหนดไว้ โดยชิปที่ควบคุมจะเป็น BCM5221 ซึ่งใช้สำหรับควบคุมการทำงานของ Ethernet โดยเฉพาะ

ดังนั้นในการทดสอบ Ethernet ก็จะแบ่งการทดสอบออกเป็น 2 ส่วนนั้นคือการทดสอบการรับส่งข้อมูลผ่าน Ethernet และส่วนที่สองจะเป็นการนำส่วนแรกไปเชื่อมต่อกับส่วนประมวลผลสัญญาณ ซึ่งก็ได้ผลออกมาถูกต้องตามที่ต้องการ จึงสรุปได้ว่าการส่งข้อมูลสำหรับการพัฒนาต้นแบบนี้จะใช้การส่งข้อมูลผ่านพอร์ต Ethernet โดยข้อมูลที่ส่งในสายสัญญาณจะถูกเข้ารหัส Line Coding แบบ 4B5B ซึ่งชิป BCM5221 จะเป็นตัวแปลงการส่งสัญญาณนี้เอง และแปลงกลับให้อยู่ในลักษณะของบิทข้อมูลอีกที



ภาพที่ 71 การส่งข้อมูลแบบ Ethernet

และระบบต้นแบบที่พัฒนาขึ้นจะถูกนำไปใส่ไว้ในกล่องเพื่อความคงทนและสวยงาม ดังแสดงในภาพที่ 72 ซึ่งจะเห็นได้ว่าสามารถเคลื่อนย้ายได้โดยง่าย เหมาะสมกับความต้องการให้สถานีภาคพื้นดินนี้เป็นสถานีแบบเคลื่อนที่

ภาพที่ 72 ภาพถ่ายต้นแบบส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ที่พัฒนาขึ้น

สุดท้ายได้ทำการปรับปรุงทรัพยากรที่ใช้ให้มีขนาดเล็กลง โดยลดขาที่ไม่จำเป็นต้องตรวจสอบออกไป ทำให้ทรัพยากรที่ใช้มีขนาด 3707 Slices หรือ 8341 Logic cells ใช้หน่วยความจำไป 99 Kbytes และความถี่สูงสุดที่สามารถทำงานได้คือ 153.112 MHz แต่ที่จริงวงจรจะสามารถทำงานที่ความถี่สูงสุดที่ 96.1 MHz ตามที่ได้กล่าวไว้ในหัวข้อที่แล้ว

2. <u>การทดสอบการเข้าจังหวะสัญญาณ</u>

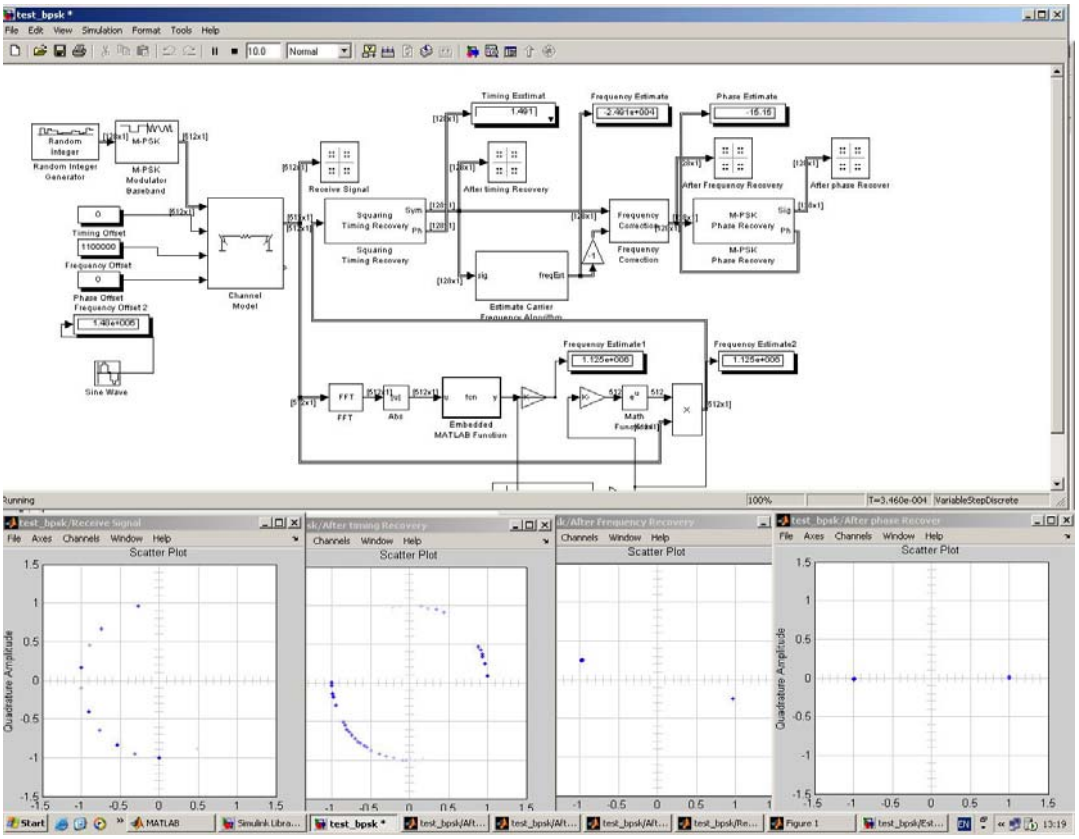
ดังที่ได้กล่าวไว้ข้างต้น การทดสอบการเข้าจังหวะว่าสามารถแก้ไขความผิดพลาดที่เกิดจากความผิดพลาดจากเวลาที่ไม่ตรงกัน เฟสที่ตกค้าง อันเนื่องมาจากความไม่ตรงกันของสัญญาณนาฬิกาในภาคส่งและภาครับ และปัญหาที่รุนแรงที่สุดจากการโครจรดาวเทียมวงโคจรต่ำ นั้นก็คือปัญหาความถี่เลื่อน (Doppler Frequency) ซึ่งจากการคำนวณจากประเทศจีนพบสูงถึง 1.5 MHz ตามที่ได้กล่าวไว้ในหัวข้อการเข้าจังหวะสัญญาณ ดังนั้นการทดสอบส่วนการเข้าจังหวะนี้จะถูกแบ่งออกเป็น 2 เฟสใหญ่ๆ เพื่อความเข้าใจดังนี้

- เฟสที่ 1 จะเป็นการพัฒนาส่วนฟังก์ชั่นการทำงานในแต่ละส่วนรวมทั้งทดสอบความถูกต้อง และการปรับปรุงพารามิเตอร์ต่างๆ ให้เหมาะสม

- เฟสที่ 2 เป็นการทดสอบระบบรวมของการเข้าจังหวะสัญญาณ โดยจะทดสอบร่วมกับซอฟต์แวร์ MATLAB V7.0 และ ModelSim รวมถึงการปรับปรุงประสิทธิภาพในแต่ละฟังก์ชั่นการทำงาน ซึ่งในการพัฒนาในเฟสนี้จะถูกนำไปสู่การทดสอบในขั้นสุดท้าย

2.1 <u>การพัฒนาฟังก์ชั่นการทำงานของส่วนการเข้าจังหวะสัญญาณ</u>

การพัฒนาในเฟสแรกนี้จะเป็นการพัฒนาให้ฟังก์ชั่นที่อธิบายไว้ในหัวข้อที่ผ่านมาสามารถทำงานได้ถูกต้อง ตามแนวทางที่ตั้งเอาไว้ โดยจะมีการจำลองการทำงานในแต่ละส่วนออกมาด้วยซอฟต์แวร์ MATLAB V7.0 ตามภาพที่ 73 เพื่อดูผลลัพธ์ต่างๆ ออกมาก่อนเพื่อเพิ่มความเข้าใจและการดำเนินการต่างๆ รวมถึงปรับปรุงให้เหมาะสมกับการพัฒนาบนเทคโนโลยี FPGA/VHDL

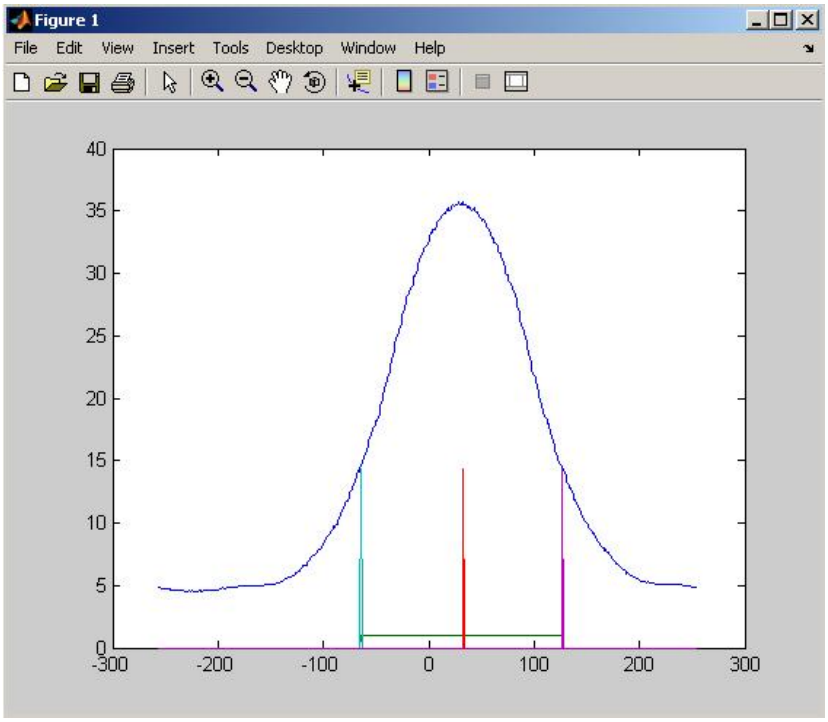


ภาพที่ 73 ผลการจำลองการทำงานในการเข้าจังหวะสัญญาณ

สำหรับการพัฒนาในแต่ละฟังก์ชั่นความยากอยู่ที่การเลือกใช้พารามิเตอร์ให้เหมาะสม เพื่อให้สามารถเข้าจังหวะสัญญาณได้ถูกต้อง ซึ่งพารามิเตอร์ต่างๆ ได้กล่าวถึงไว้แล้วในหัวข้อที่ผ่านมา และการทดสอบก็จะตรวจสอบข้อมูลร่วมกับซอฟต์แวร์ MATLAB ที่ใช้ในการตรวจสอบข้อมูล และ ModelSim ที่ใช้สำหรับการจำลองการทำงานของภาษา VHDL หรือจะเรียกได้ว่าเป็นการสังเคราะซอฟต์แวร์ที่พัฒนาขึ้นมาบนคอมพิวเตอร์เพื่อดูความถูกต้องของฟังก์ชั่นการทำงานต่างๆ ต่อไปจะขอยกตัวอย่างซัก 2 ตัวอย่างสำหรับการแก้ไขพารามิเตอร์เพื่อให้เหมาะสมกับการเข้าจังหวะสัญญาณ

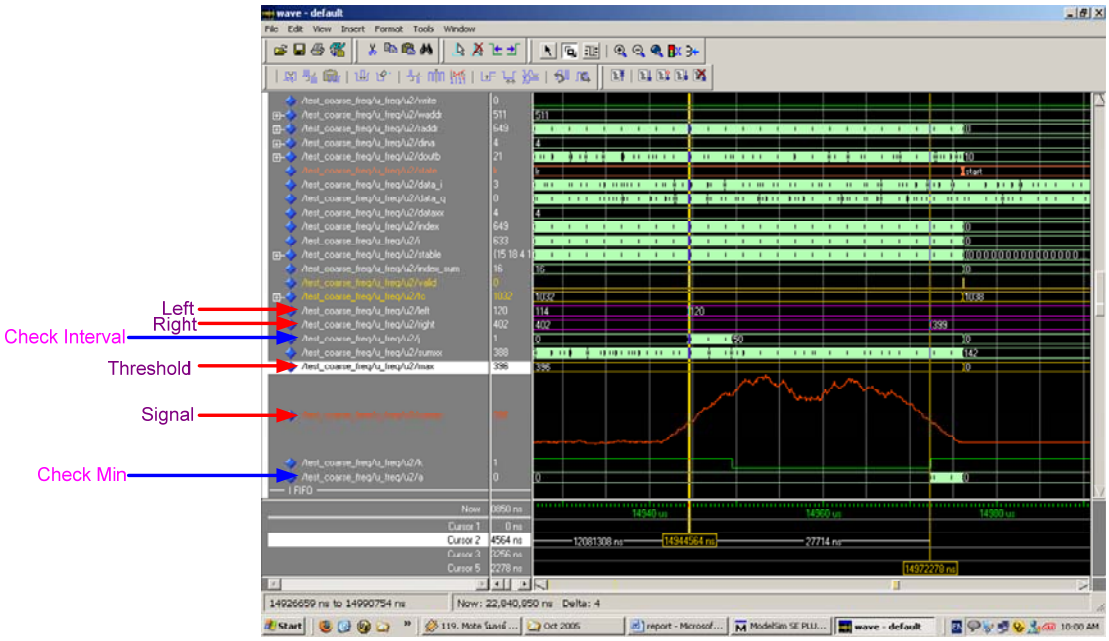
ตัวอย่างแรกจะเป็นการเลือกขอบซ้ายและขอบขวาของสัญญาณ ที่จะนำไปประมาณเป็นความถี่สำหรับแก้ไขในการประมาณความถี่แบบหยาบ ซึ่งการตัดสินใจนี้จะส่งผลถึงความผิดพลาดของข้อมูล ถ้าเลือกไม่ดีความผิดพลาดก็จะสูงมาก ซึ่งการเลือกขอบซ้ายจะเลือกด้วยการหาค่าที่มากกว่าค่าตัดสินใจที่ได้ตั้งเอาไว้ก่อนหน้า ส่วนขอบขวาจะเลือกด้วยการหาค่าของสัญญาณต่อไปที่น้อยกว่าค่าที่ตัดสินใจ และจะเห็นได้ว่าเมื่อทำการเลือกขอบซ้ายเสร็จแล้วจำเป็นต้องเว้นระยะห่างเอาไว้ซักช่วงนึงเพื่อกันไม่ให้เกิดค่าสัญญาณที่อาจจะเกิดการตกลงมา

ก่อนที่จะพ้นจุดสูงสุด ตามภาพที่ 74 ซึ่งถ้าตัดสินใจไปเลยก็จะทำให้เกิดการประมาณที่ผิดพลาดมาก ดังนั้นจึงจำเป็นต้องมีการทิ้งระยะห่างไปให้พ้นช่วงจุดสูงสุดก่อนที่จะตัดสินใจเลือกขอบขวาต่อไป



ภาพที่ 74 การจำลองการเลือกขอบซ้ายและขอบขวาในฟังก์ชั่นการประมาณความถี่แบบหยาบ

และการเลือกขอบขวาก็จะต้องมีการตรวจสอบว่าสัญญาณต่อจากที่เลือกไว้นั้นต่ำกว่าระดับที่ตัดสินใจแน่นอนเพื่อกันไม่ให้สัญญาณสูงกลับไปแล้วดิ่งกลับลงมาอีกที ซึ่งถ้าเกิดกรณีนั้นก็จะต้องมาเลือกกันใหม่นั้นเอง การที่ต้องมีทั้งระยะห่างในการเลือกระหว่างทั้งสองขอบและการตรวจสอบค่าหลังจากเลือกขอบขวานั้นระยะนึง ก็เป็นจำเป็นต้องมีการตรวจสอบเพื่อหาค่าที่เหมาะสม เพื่อนำไปคำนวณเป็นความถี่ Doppler สำหรับใช้แก้ไขค่าในการประมาณความถี่แบบหยาบ โดยภาพที่ 75 จะเป็นการสังเคราะห์การทำงานของการประมาณความถี่แบบหยาบโดยตรวจสอบหาขอบซ้ายและขอบขวาบนซอฟต์แวร์ ModelSim



ภาพที่ 75 การหาขอบซ้ายและขอบขวาในการสังเคราะห์บน ModelSim

และอีกตัวอย่างจะเป็นการเลือกค่ามุมเพื่อใช้ในการตัดสินใจสำหรับการหาสัญญาณที่แรงที่สุดในส่วนการทำการเข้าจังหวะเวลา (Timing Recovery) โดยการเลือกนี้จะต้องไม่ทำให้ค่าที่ใช้เปลี่ยนแปลงมากจนเกินไป ซึ่งถ้าเราเลือกให้ค่ามุมที่ใช้ในการตัดสินใจมีการเปลี่ยนแปลงมากจนเกินไปก็จะก่อให้เกิดความผิดพลาดขึ้นมา ถึงแม้ว่าจะมีการกรองสัญญาณความถี่สูงออกไปแล้วในฟังก์ชั่น RRC Filter แต่ก็ยังมีความผิดพลาดเกิดขึ้นอยู่ ซึ่งตัวมุมที่ใช้ในการตัดสินใจในการเลือกสัญญาณจะเป็นพารามิเตอร์ที่ส่งผลอย่างมากต่อความผิดพลาดที่เกิดขึ้น ดังนั้นจึงจำเป็นต้องมีการประมาณมุมที่คำนวณออกมาได้อีกชั้นนึง เพื่อให้เกิดความเปลี่ยนแปลงของมุมให้น้อยที่สุด โดยการประมาณนี้จะใช้วิธีการเฉลี่ยค่าตามที่ได้อธิบายไว้ในข้างต้นนั้นเอง

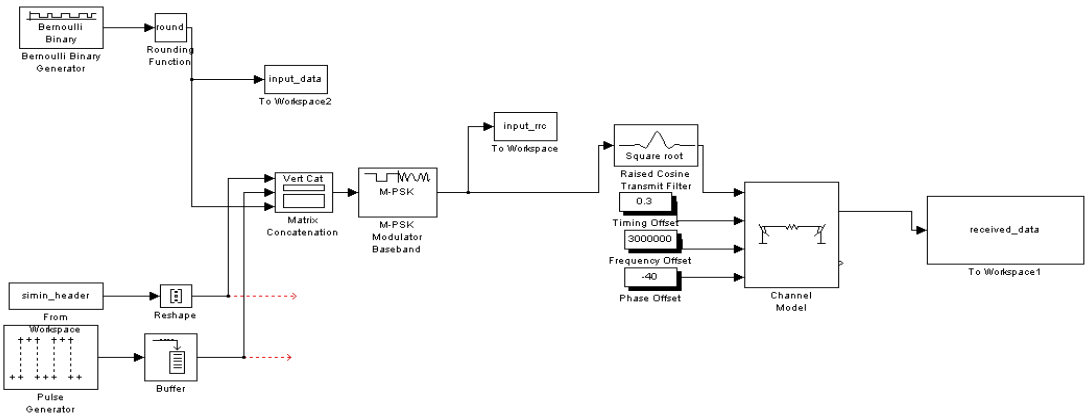
อีกส่วนนึงเพื่อให้การพัฒนาเป็นไปอย่างรวดเร็วตามปธิณาน FCB (Faster, Cheaper และ Better) ก็คือฟังก์ชั่นย่อยๆ บางส่วนจะใช้ของที่มีอยู่แล้วในท้องตลาดและฟรี ซึ่งเป็น Open Core อยู่แล้วได้แก่ Rectangular to Polar (R2P), Polar to Rectangular (P2R), และ Fast Fourier Transform (FFT) ซึ่งฟังก์ชั่นพวกนี้สามารถนำมาประกอบเข้าได้ทันที

สรุปแล้วในการพัฒนาในเฟสแรกจะเป็นการพัฒนาฟังก์ชั่นการทำงานในแต่ละส่วนเพื่อให้สามารถทำงานได้ถูกต้องตามแนวทางที่ดำเนินการไว้ โดยปรับเปลี่ยนพารามิเตอร์ให้เหมาะสมที่สุดกับการเข้าจังหวะ ในเฟสต่อไปจะเป็นการทดสอบระบบการเข้าจังหวะสัญญาณเพื่อตรวจสอบว่าระบบที่พัฒนาขึ้นมาทนต่อสภาพช่องสัญญาณได้มากที่สุดเท่าไร ซึ่งเป้าหมายของ

โครงการนี้ตั้งไว้ให้สามารถรับความถี่เลื่อนหรือ Doppler Frequency ได้ 1.5 MHz จะได้กล่าวต่อไปในหัวข้อถัดไป

2.2 การทดสอบระบบการเข้าจังหวะสัญญาณ

การทดสอบในเฟสนี้จะเป็นการตรวจสอบความสามารถในการแก้ไขความผิดพลาดอันเนื่องมาจากช่องสัญญาณในลักษณะต่างๆ ไม่ว่าจะเป็นผลที่เกิดขึ้นจาก ความถี่เลื่อน (Doppler Frequency) การเลื่อนของเวลา และการเลื่อนเฟส โดยอินพุทจะถูกสร้างจากซอฟต์แวร์ MATLAB ดังภาพที่ 76 และการจำลองการทำงานของฟังก์ชั่นที่พัฒนาขึ้นมานั้นจะถูกทดสอบด้วยซอฟต์แวร์ ModelSim เพื่อนำข้อมูลไปเปรียบเทียบกับอินพุทที่ได้มา



ภาพที่ 76 การสร้างข้อมูลอินพุทเพื่อทดสอบการเข้าจังหวะ

และการรวมฟังก์ชั่นก็จะเป็นไปตามภาพที่ 26 ที่มีลำดับดังนี้คือ การประมาณความถี่แบบหยาบ ตัวกรองความถี่ การเข้าจังหวะเวลา การประมาณความถี่แบบละเอียด การประมาณเฟสตกค้าง และสุดท้ายสร้างเฟรมข้อมูล โดยการประมาณความถี่แบบหยาบจะทำให้การเลื่อนความถี่ที่เกิดจากความถี่เลื่อนนั้นแคบลงมาและเปลี่ยนแปลงเล็กน้อย เพื่อให้การประมาณความถี่แบบละเอียดทำงานได้ถูกต้องมากขึ้น ฟังก์ชั่นที่สองในลำดับจะเป็นตัวกรองสัญญาณที่จะช่วยลดความผิดพลาดของข้อมูลตามที่จะเห็นได้กล่าวไว้ข้างต้น และช่วยให้การเข้าจังหวะเวลาสามารถเลือกมุมในการตัดสินใจได้ถูกต้องมากขึ้น ซึ่งส่วนการเข้าจังหวะสัญญาณที่เป็นฟังก์ชั่นต่อมาก็จะมาช่วยแก้ไขปัญหาความแรงของสัญญาณที่เกิดขึ้นในช่องสัญญาณที่อาจจะมีมากน้อยไม่เท่ากันในแต่ละ Symbol โดยฟังก์ชั่นนี้จะทำการเลือกสัญญาณที่แรงที่สุดในแต่ละ Symbol เพื่อให้เกิดความถูกต้องมากที่สุด และฟังก์ชั่นต่อไปก็จะเป็นการประมาณความถี่แบบละเอียด ซึ่งก็จะมาแก้ไขปัญหาความถี่เลื่อนเหมือนกับการประมาณแบบหยาบ และตัวต่อไปก็จะเป็นการแก้ไขเฟสที่

ตกค้างในฟังก์ชั่นการประมาณเฟสตกค้าง สุดท้ายจะเป็นตัวช่วยแก้ไขความคลุมเครือของเฟสที่เกิดขึ้นมา

ในการเข้าจังหวะสัญญาณจะไม่ได้รับประกันว่าจะสามารถแก้ไขความผิดพลาดที่เกิดจากช่องสัญญาณได้ 100% แต่จะเป็นแค่การทำให้ความผิดพลาดของข้อมูลที่เกิดขึ้นมาไม่ว่าจะนั้นลดลงไปให้เหลือน้อยที่สุด เนื่องจากการเข้าจังหวะสัญญาณเป็นแค่การแก้ไขปัญหาบางส่วนที่เกิดจากช่องสัญญาณเช่น ความถี่เลื่อน เฟสตกค้าง และการเลื่อนของเวลา เป็นต้นเท่านั้น ดังนั้นความผิดพลาดก็จะสามารถเกิดขึ้นได้หลังจากเข้าจังหวะสัญญาณแล้ว แต่เพียงว่าความผิดพลาดที่เกิดขึ้นนั้นจะต้องอยู่ในสัดส่วนที่ต่ำพอที่ตัวถอดรหัสข้อมูลจะสามารถแก้ไขได้ โดยความสามารถของตัวถอดรหัสสามารถรับความผิดพลาดได้ประมาณ 15% หรือถ้าข้อมูลบางแบบอาจจะสามารถรับได้ถึง 20% ทีเดียว ซึ่งที่จริงอาจจะอยู่สูงกว่านั้นขึ้นอยู่กับรูปแบบของความผิดพลาดถ้าเกิดติดๆกันมากจนเกินไปก็ไม่สามารถที่จะแก้ไขได้ แต่ว่าสิ่งที่ดีที่สุดคือ การที่จะทำให้ข้อมูลที่ออกมาจากการเข้าจังหวะสัญญาณมีความผิดพลาดให้อยู่ในสัดส่วนที่น้อยที่สุดเท่าที่จะเป็นไปได้ จึงเป็นเหตุผลหลักในเฟสแรกที่จะต้องมีการปรับพารามิเตอร์ให้เหมาะสมมากที่สุด ซึ่งจะช่วยให้ความผิดพลาดที่เกิดขึ้นมานั้นมีค่าน้อยที่สุด

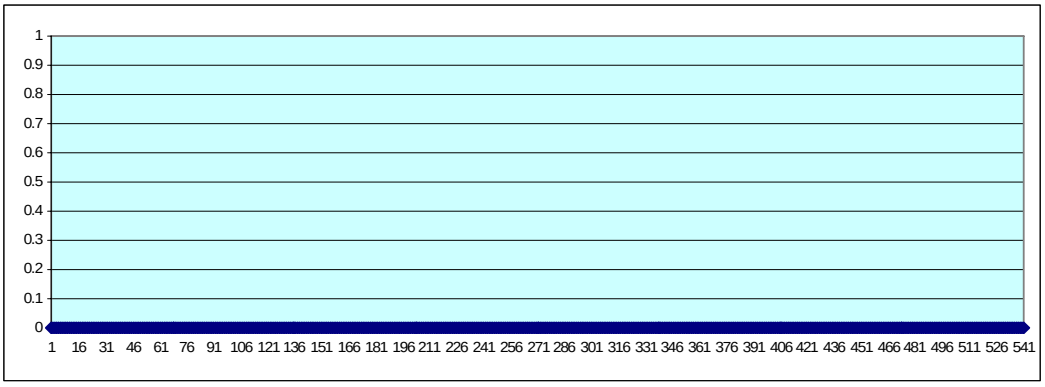
การทดสอบระบบการเข้าจังหวะสัญญาณนี้จะทดสอบเป็นลำดับขั้นเพื่อให้เกิดความเข้าใจการทำงานในแต่ละส่วนโดยได้แบ่งออกเป็น 3 ขั้นดังนี้

- ขั้นที่ 1 เป็นการทดสอบตัวกรองสัญญาณ การเข้าจังหวะเวลา การประมาณความถี่แบบละเอียด และการประมาณเฟสตกค้างเข้าด้วยกัน เพื่อให้เห็นถึงความสามารถของการเข้าจังหวะแค่ชั้นเดียว
- ขั้นที่ 2 เป็นการทดสอบทั้งหมดยกเว้นการเข้าจังหวะเฟรม (Frame Recovery) เพื่อให้เห็นภาพของการกลับเฟส 0 หรือ 180 ตามที่ได้กล่าวไว้
- ขั้นที่ 3 เป็นการทดสอบรวมทั้งระบบเพื่อตรวจสอบว่าสามารถสร้างเฟรมข้อมูลเพื่อนำไปถอดรหัสได้หรือไม่

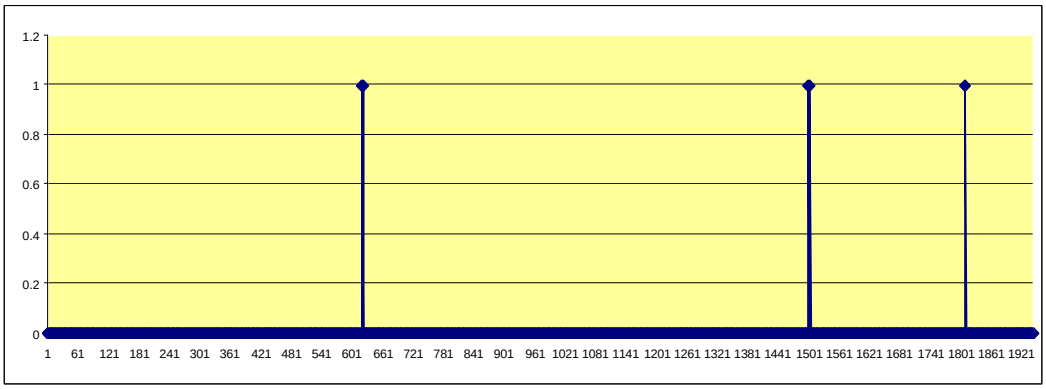
การทดสอบทุกขั้นตอนจะเป็นการสังเคราะห์การทำงานบน ModelSim จะเปรียบเทียบข้อมูลกับซอฟต์แวร์ MATLAB ที่สร้างข้อมูลอินพุท

ในขั้นแรกจะทดสอบการประมวลผลสัญญาณ 4 ฟังก์ชั่นตามที่ได้กล่าวไว้ในเบื้องต้น และจะมีการทดสอบที่ความถี่เลื่อน 10 kHz, 100kHz, 350kHz, 500kHz, และ 700kHz ซึ่ง

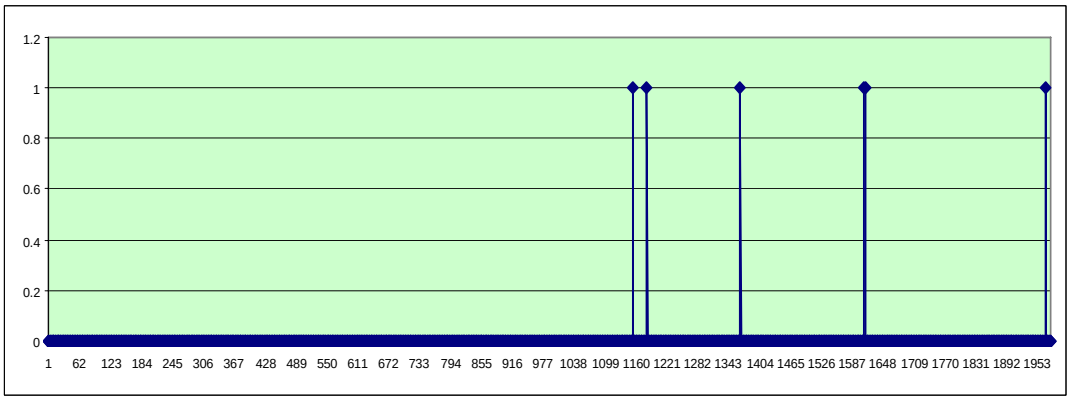
ผลการทดสอบจะอยู่ตามภาพที่ 77 (ก) ถึง (จ) ตามลำดับ โดยค่าการเลื่อนเวลาจะเป็น 0.2 ค่าเฟสตกค้าง 20 และ SNR (Signal to Noise Ratio) 15 dB ตามทฎษฎีแล้วการทำงานในทั้ง 4 ฟังก์ชั่นนี้จะต้องแก้ไขความถี่เลื่อนได้สูงถึง 750 kHz ทีเดียว



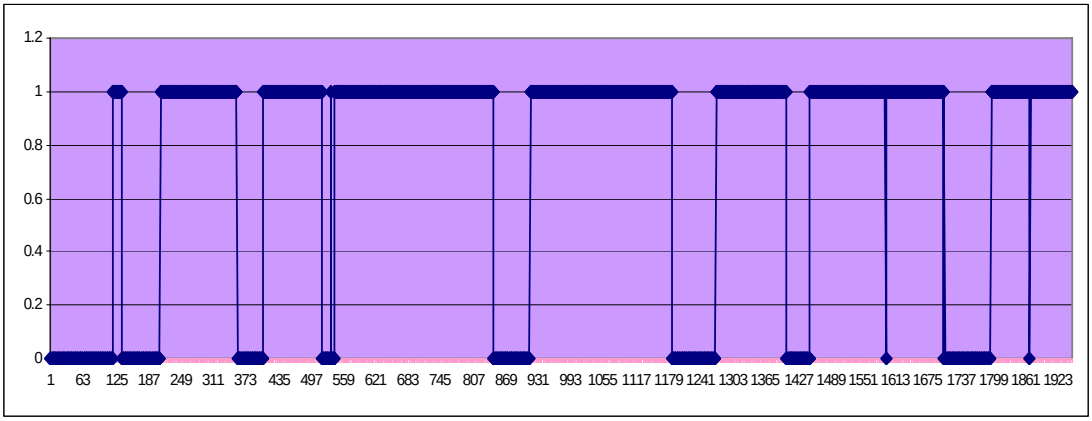
(ก) การทดสอบที่ความถี่เลื่อน 10 kHz ขั้นที่ 1



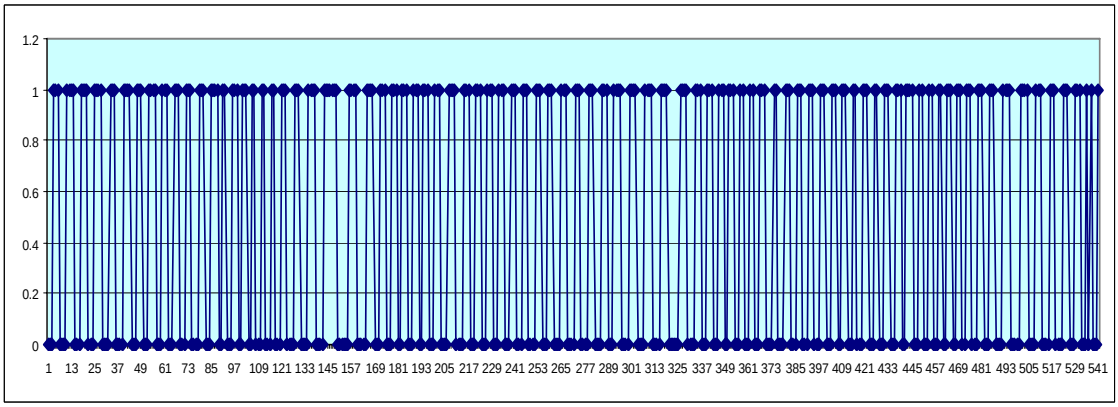
(ข) การทดสอบขั้นที่ 1 ที่ความถี่เลื่อน 100 kHz



(ค) การทดสอบขั้นที่ 1 ที่ความถี่เลื่อน 350 kHz



(ง) การทดสอบขั้นที่ 1 ที่ความถี่ 500 kHz

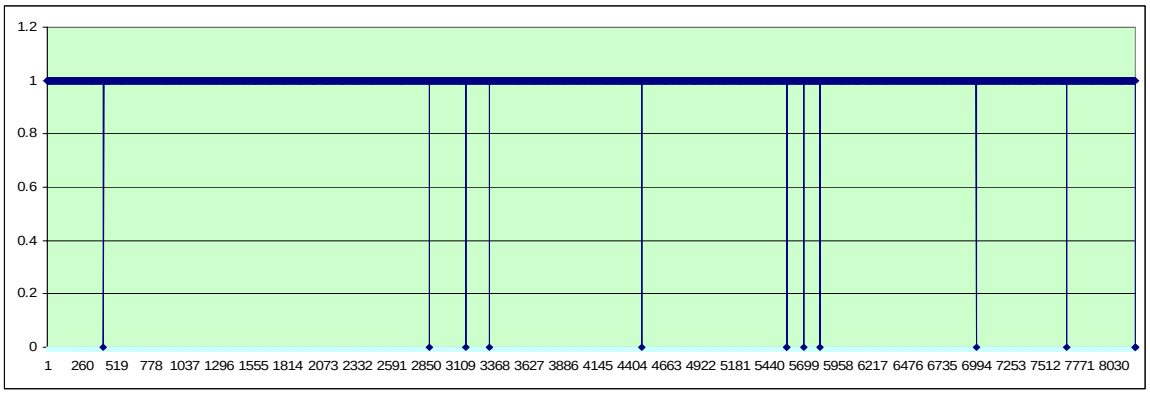


(จ) การทดสอบขั้นที่ 1 ที่ความถี่ 700 kHz

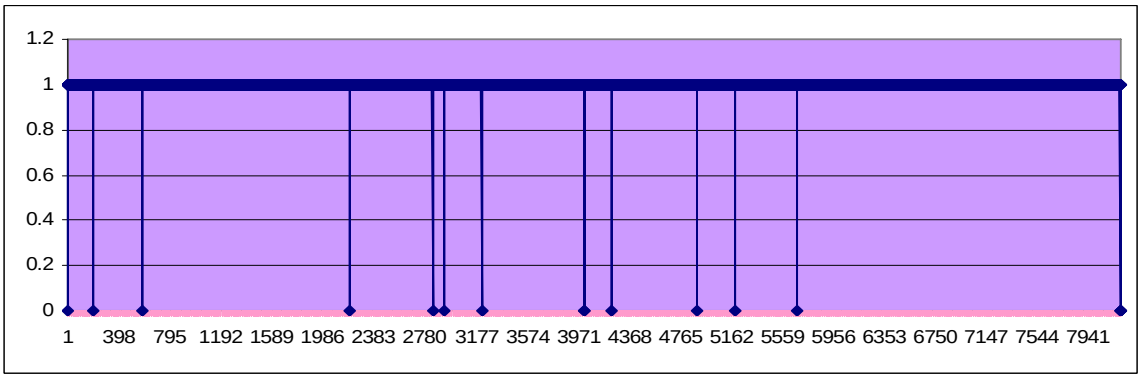
ภาพที่ 77 การทดสอบที่ความถี่ต่างๆ ในขั้นตอนที่ 1

ผลการทดสอบจะเห็นได้ว่าที่ความถี่ 10 kHz จะไม่มีความผิดพลาดเกิดขึ้นเลย และเฟสจะเป็น 0 พอเพิ่มไปเป็น 100 kHz จะเกิดความผิดพลาด 0.41% และเฟสก็ยังเป็น 0 อยู่ จากนั้นก็เพิ่มเป็น 350 kHz ความผิดพลาดจะเกิดขึ้นประมาณ 0.3% ซึ่งช่วงนี้ก็ตัวถอดรหัสสัญญาณก็ยังสามารถที่จะแก้ไขความผิดพลาดที่เกิดขึ้นมาได้ และที่เกิดผิดพลาดขึ้นมาเนื่องจากส่วนการทำงานนี้จะแก้ความผิดพลาดได้เฉพาะอย่างเท่านั้น ไม่สามารถแก้ความผิดพลาดที่เกิดจากช่องสัญญาณได้ทั้งหมด พอเพิ่มความถี่ขึ้นเป็น 500 kHz และ 700 kHz ก็จะพบความผิดพลาดที่สูงขึ้นเป็น 28% และ 48% ตามลำดับ และสังเกตได้ว่าที่ความถี่ 500 kHz จะมีการเปลี่ยนเฟสไปมาระหว่าง 0 กับ 180 อีกด้วย ซึ่งจะเห็นได้ว่าความสามารถของการประมาณความถี่แบบละเอียดในทางปฏิบัติจะไม่ถึง 750 kHz ทำได้แค่ 500 kHz ก็เกิดความผิดพลาดที่ไม่สามารถที่จะยอบรับได้แล้ว ซึ่งนี่ก็เป็นเหตุผลหลักอีกอันนึงที่จะต้องมีการประมาณความถี่แบบ 2 ชั้นด้วย เนื่องจากเป้าหมายของโครงการตั้งไว้ถึง 1.5 MHz ทีเดียว

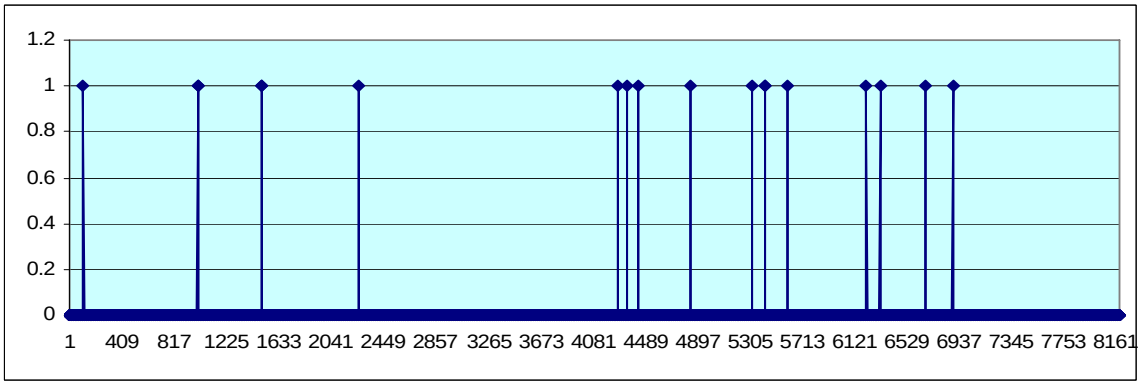
จากที่ได้กล่าวอ้างไปในข้างต้นทำให้การทดสอบจะดำเนินไปในขั้นที่ 2 ด้วยการเพิ่มการประมาณแบบหยาบเข้ามาในระบบการเข้าจังหวะสัญญาณ เพื่อเพิ่มประสิทธิภาพการประมาณความถี่เลื่อน โดยผลการทดสอบตาม และค่าพารามิเตอร์อื่นๆ ก็จะยังเหมือนเดิม ซึ่งตามทฤษฎีแล้วเมื่อเพิ่มการประมาณความถี่แบบหยาบเข้าไปจะสามารถแก้ไขได้สูงไม่เกิน 6 MHz แต่ในความเป็นจริงแล้วจะสามารถทำงานที่จะแก้ไขความถี่ได้ไม่เกิน 1.5 MHz เนื่องจากในภาคสัญญาณวิทยุได้จำกัดช่องสัญญาณที่ 6 MHz (คือตั้งแต่ -3 MHz ถึง 3 MHz)



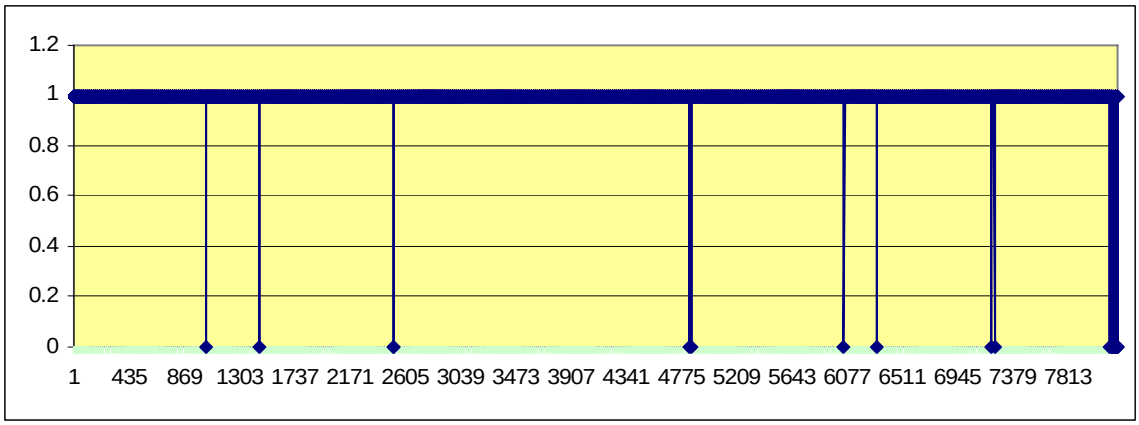
(ก) การทดสอบขั้นที่ 2 ที่ความถี่ 350 kHz



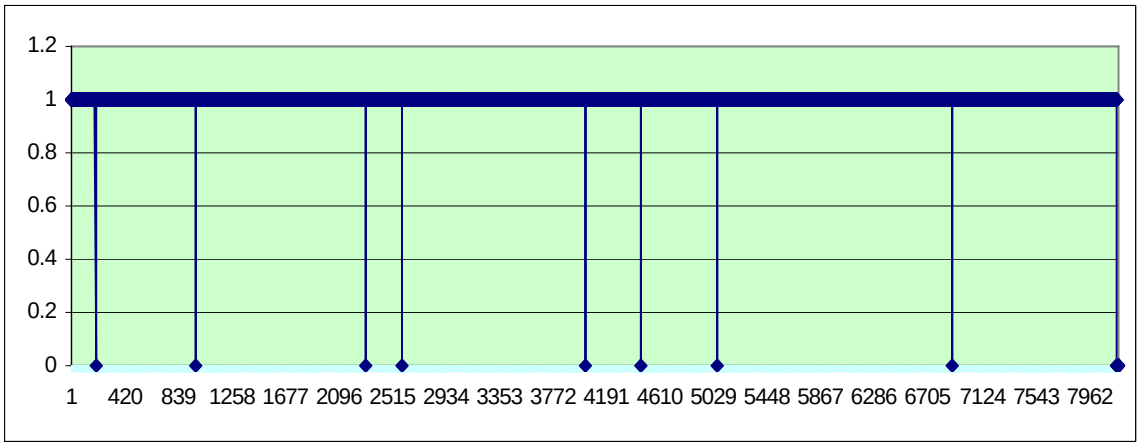
(ข) การทดสอบขั้นที่ 2 ที่ความถี่ 500 kHz



(ค) การทดสอบขั้นที่ 2 ที่ความถี่ 700 kHz



(ง) การทดสอบขั้นที่ 2 ที่ความถี่ 1 MHz



(จ) การทดสอบขั้นที่ 2 ที่ความถี่ 3 MHz

ภาพที่ 78 การทดสอบที่ความถี่ต่างๆ ในขั้นที่ 2

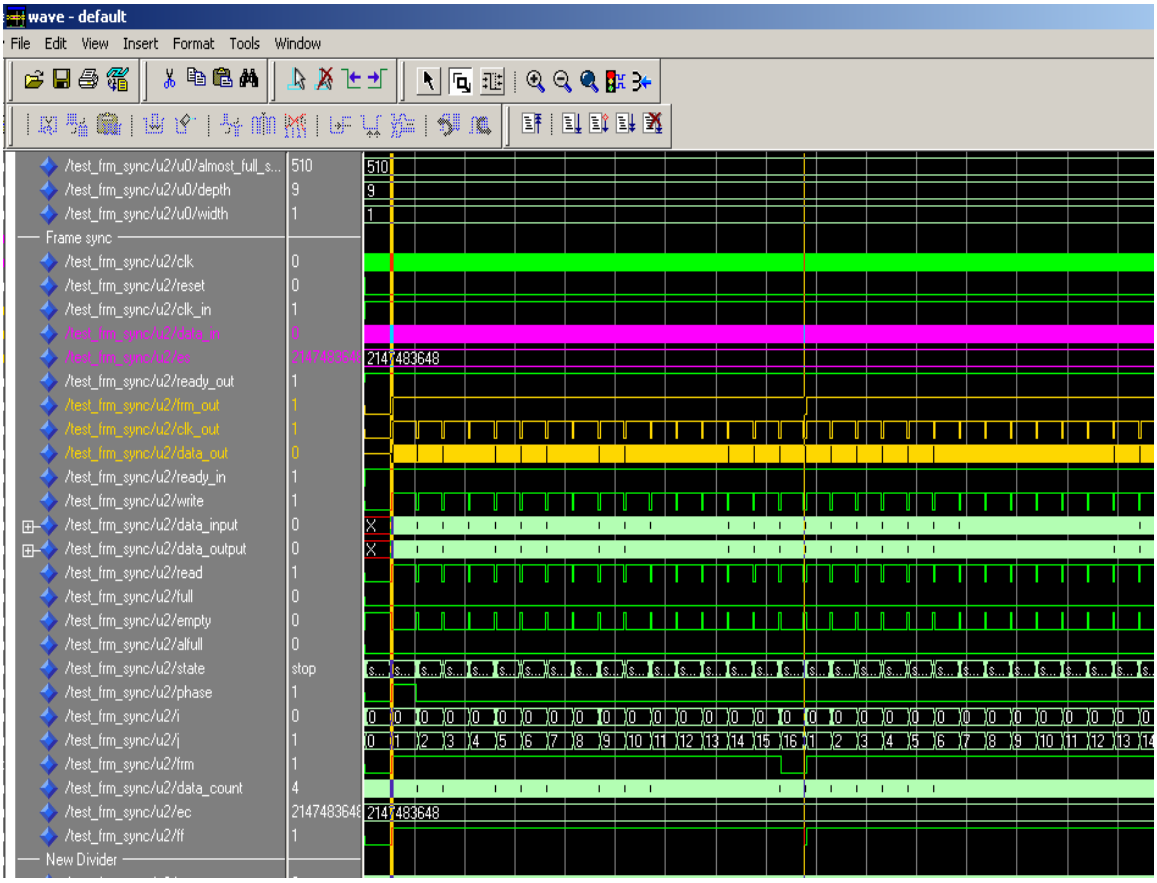
จะเห็นได้ว่าการทดสอบเมื่อรวมเข้ากับส่วนการประมาณความถี่แบบหยาบแล้ว สัดส่วนความผิดพลาดที่เกิดขึ้นมาจะไม่ถึง 1% เลยในทุกการทดสอบ (ผลการทดสอบดังนี้คือ ที่ 350 kHz จะมีความผิดพลาด 0.17%, ที่ 500 kHz จะมีความผิดพลาด 0.15%, 700 kHz จะมีความผิดพลาด 0.22%, 1 MHz จะมีความผิดพลาด 0.42%, และ 3 MHz จะมีความผิดพลาด 0.13%) จะเห็นได้ว่าจะมีการกลับเฟสเป็น 180 ให้เห็นทั้งหมด ยกเว้นที่ความถี่ 700 kHz เท่านั้นที่เป็นเฟส 0 ซึ่งความผิดพลาดที่เกิดขึ้นเป็นความผิดพลาดที่สามารถยอมรับได้ และสามารถที่จะแก้ไขได้ที่ส่วนการถอดรหัสข้อมูล

และความสามารถของการประมาณความถี่แบบหยาบจะถูกจำกัดไว้ที่สูงสุดไม่เกิน 1.5 MHz เนื่องด้วยขนาดของช่องสัญญาณสามารถใช้ได้ที่ 6 MHz แต่ก็ไม่มีผลอะไรเพราะว่า

ความถี่สูงสุดในการเลื่อนความถี่จะอยู่ที่ 1.5 MHz ถึงแม้ว่าความสามารถของตัวประมาณความถี่แบบหยาบนี้จะสามารถทำได้สูงสุดไม่เกิน 4.5 MHz โดยประมาณก็ตาม

ผลที่ได้จากขั้นที่ 2 เป็นที่น่าพอใจสำหรับการเข้าจังหวะสัญญาณ โดยความผิดพลาดทีเกิดขึ้นมาก็อยู่ในช่วงที่สามารถจะแก้ไขให้ถูกต้องได้ในการถอดรหัสสัญญาณ แต่มีส่วนนึงที่จำเป็นต้องมีการแก้ไขก็คือ การแก้ความคลุมเครือของเฟส (Phase Ambiguity) ซึ่งจะถูกทดสอบในขั้นที่ 3 และในขั้นนี้จะจะเป็นการสร้างเฟรมข้อมูลเพื่อนำไปเชื่อมต่อกับส่วนถอดรหัสข้อมูลต่อไป

ตามที่ได้กล่าวขั้นตอนนี้จะเพิ่มส่วนการเข้าจังหวะเฟรม (Frame Recovery) ซึ่งจะทำหน้าที่ในการกลับเฟส 180 ให้กลับมาเป็น 0 และสร้างสัญญาณเฟรมข้อมูลเพื่อเชื่อมต่อกับส่วนการถอดรหัส โดยจะได้รูปแบบของเฟรมข้อมูลเป็นไปตามภาพที่ 79



ภาพที่ 79 การสร้างเฟรมเพื่อเชื่อมต่อกับส่วนถอดรหัส (รูปแบบเฟรม สีเหลือง)

ในการทดสอบระบบรวมทั้งหมดแล้วพบว่าความผิดพลาดของข้อมูลจะน้อยกว่าหรือเท่ากับของเดิมที่ทดสอบไว้ในขั้นที่ 2 ที่น้อยกว่าได้เนื่องจากความผิดพลาดอาจจะไปตกอยู่ในช่วงของการตรวจสอบเฟรมที่ยอมให้ผิดพลาดได้เล็กน้อยทำให้สัดส่วนความผิดพลาดที่เกิดขึ้นน้อยลงกว่าเดิม

สรุปแล้วในการทดสอบในขั้นตอนนี้สามารถทำงานได้ถูกต้องตามความต้องการและแก้ความคลุมเครือของเฟสได้ รวมทั้งยังมีความสามารถในการเชื่อมต่อกับส่วนประมวลผลสัญญาณต่อไปได้ หลังจากนี้จะเป็นการทดสอบระบบในขั้นตอนสุดท้าย นั้นก็คือการเชื่อมต่อร่วมกับส่วนการเชื่อมต่อกับภาคสัญญาณวิทยุ ซึ่งได้เลือกใช้อุปกรณ์มาตรฐานในท้องตลาดนั้นเอง

3. <u>การทดสอบระบบรวม</u>

จากที่กล่าวมาทั้งหมดข้างต้นเป็นการอธิบายการทำงานในแต่ละส่วนของส่วนการเชื่อมต่อกับภาคสัญญาณวิทยุเท่านั้น โดยในส่วนภาคส่งก็ได้มีการเชื่อมต่อกับส่วนประมวลผลดิจิตอลเบสแบนด์ สามารถทำงานได้จริงที่ 65 MHz ตามที่ได้กล่าวไปแล้ว และส่วนภาครับจำเป็นต้องใช้บอร์ดพัฒนาถึง 3 ตัวคือ AD8321, AD6644, และ AD6620 เพื่อเชื่อมต่อกับส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ในบอร์ด FPGA และจะต้องรับสัญญาณได้ที่ความถี่ 75 MHz ดังนั้นในแผนการเชื่อมต่อออกเป็น 3 เฟสใหญ่ๆ ดังนี้คือ

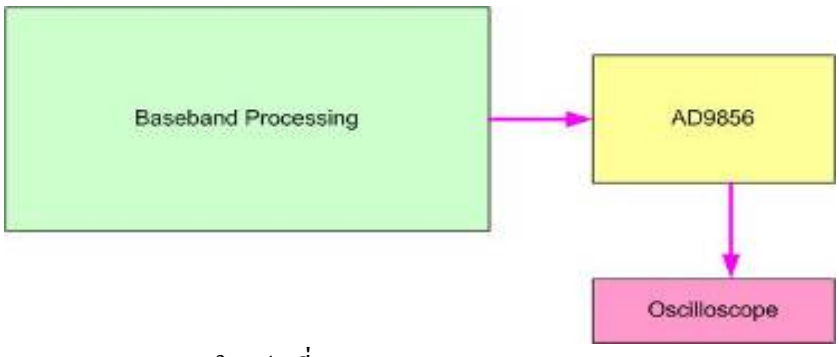
- เฟสที่ 1 การทดสอบการประมวลผลสัญญาณและส่งสัญญาณที่ความถี่ 65 MHz เป็นการทดสอบส่วนภาคส่งสัญญาณเท่านั้น

- เฟสที่ 2 การทดสอบการประมวลผลสัญญาณและรับสัญญาณที่ความถี่ 75 MHz โดยจะเป็นการทดสอบแบบประมวลผลย้อนกลับ

3.1 <u>การทดสอบการประมวลผลสัญญาณและส่งสัญญาณที่ความถี่ 65 MHz</u>

ในการทดสอบนี้ตามที่ได้กล่าวไว้ในหัวข้อ AD9856 ที่แบ่งการทำงานออกเป็น 2 ส่วนคือ การทดสอบแบบ Single Tone ที่ตรวจสอบความถูกต้องของพารามิเตอร์ต่างๆ ว่าสามารถทำงานได้จริงที่ 65 MHz ตามที่จะเห็นได้จากภาพที่ 44 ที่เป็นสัญญาณที่วัดได้จาก Oscilloscope และส่วนที่สองเป็นการทดสอบส่งข้อมูลออกเข้าบอร์ด AD9856 จากการโปรแกรมค่าพารามิเตอร์

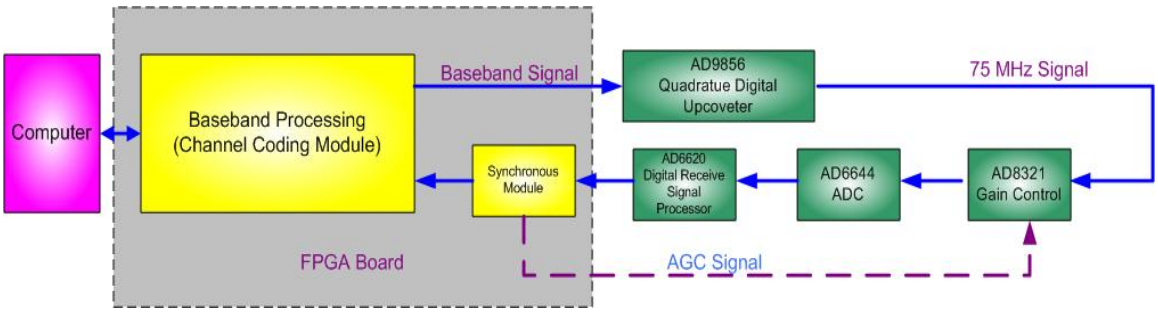
สำหรับการจัดการในบอร์ด AD9856 ผ่านบอร์ด FPGA ที่พัฒนาขึ้นด้วยภาษา VHDL เป็นไปตามภาพที่ 80 และผลที่ได้ก็จะเป็นไปตามภาพที่ 47 ที่เป็นสัญญาณหลังจากถูกทำการ Up-Converter ขึ้นมาจากสัญญาณเบสแบนด์ให้กลายเป็นสัญญาณความถี่กลาง 65 MHz และภาพที่ 48 ที่เป็นสัญญาณที่ถูกการกรองสัญญาณความถี่สูงออกไปแล้ว



ภาพที่ 80 แนวทางการทดสอบในเฟสที่ 1

3.2 การทดสอบการประมวลผลกลับสำหรับภาครับสัญญาณที่ความถี่ 75 MHz

ในส่วนนี้จะเป็นการพัฒนาเพื่อตรวจสอบสัญญาณในภาครับว่าจะสามารถทำงานได้ถูกต้องหรือไม่ โดยได้วางการทดสอบไว้ตามภาพที่ 83 ซึ่งจะเป็นการส่งข้อมูลจากซอฟต์แวร์ประยุกต์มาให้แก่ส่วนดิจิตอลเบสแบนด์ประมวลผลสัญญาณและส่งออกไปที่ความถี่ 75 MHz และประมวลผลย้อนกลับมาที่บอร์ด AD8321, AD6644, AD6620, และ FPGA ตามลำดับ โดยในบอร์ด FPGA ก็จะมาเข้าที่ส่วนการเข้าจังหวะสัญญาณ เพื่อแก้ไขปัญหาต่างๆ เช่น เวลาไม่ตรง เฟสไม่ตรง ความถี่เลื่อน และใช้ปรับพลังงานไม่ให้สูงเกินหรือน้อยเกินไป



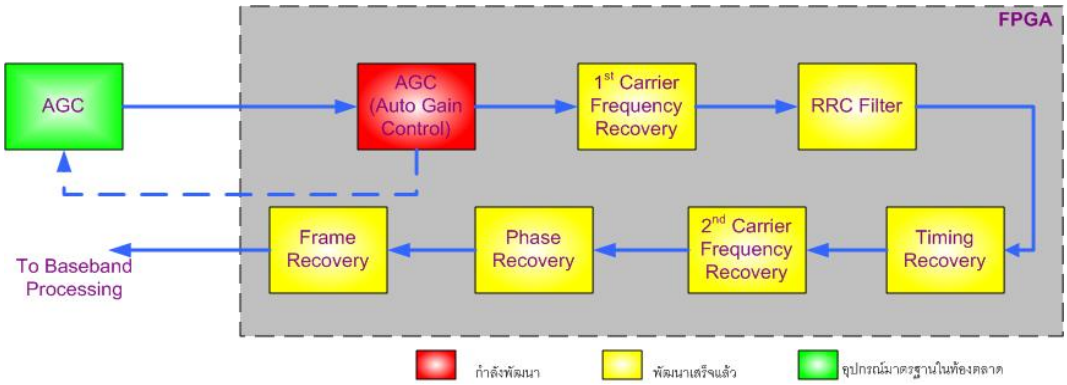
ภาพที่ 81 แนวทางทดสอบในเฟสที่ 2

โดยภาคส่งสัญญาณจะนำมาจากในเฟสที่ 1 เพียงเปลี่ยนแปลงพารามิเตอร์ให้ส่งออกมาที่ 75 MHz แทนไปก่อนเพื่อให้สามารถทดสอบแบบประมวลผลย้อนกลับได้

ในภาครับได้ทำการทดสอบการทำงานของ AD6644 และ AD6620 ซึ่งได้กล่าวไว้ในหัวข้อข้างต้นไปแล้ว โดยในส่วน AD6620 ก็จะเป็นการโปรแกรมพารามิเตอร์เริ่มต้นให้แก่บอร์ดผ่านทางพอร์ตพาราแรล (ที่ใช้สำหรับค่าเอาพุทของบอร์ด AD6620) ดังนั้นในส่วนนี้จึงจำเป็นต้องมีการใช้งานร่วมกันระหว่างการโปรแกรมค่าในเริ่มต้น และการนำเอาพุทออกมาใช้ ซึ่งมีการโปรแกรมขาเพิ่มเติม เช่นขา S/P ที่ใช้สำหรับการควบคุมการทำงานของการโปรแกรมพารามิเตอร์ ในกรณีที่เป็น “0” และเป็นเอาพุทในกรณีที่เป็น “1” รวมถึงการควบคุมขาเพิ่มเติมไม่ว่าจะเป็นขา WI สำหรับกำหนดขอบเขตของข้อมูลที่จะโปรแกรมไปในแต่ละครั้ง (โดยที่ใช้จะเป็นค่า “00” หรือทีละ 16 บิทข้อมูล) และค่าที่สำคัญอีกค่าหนึ่งก็คือค่า SBM ที่เป็นตัวกำหนดสัญญาณนาฬิกาว่าจะใช้แบบภายใน (SBM = ‘1’) หรือภายนอก (SBM = “0”) สำหรับการทำงานนี้จะเลือกเป็นแบบภายนอก โดยสัญญาณที่นำไปเข้าสู่ส่วนการเข้าจังหวะสัญญาณจะเป็นไปตามภาพที่ 60 แต่ในที่สุดแล้ว ก็ได้เลือกที่จะใช้การโปรแกรมผ่านพอร์ต Micro ซึ่งที่ได้เลือกใช้แบบนี้เนื่องจากการทำงานผ่านพอร์ตพาราแรล แล้วในที่สุดจะเกิดปัญหาในการโปรแกรมพารามิเตอร์ที่จำเป็นต้องมีการ Set up Reset ให้ขึ้น “1” ในการเปลี่ยนค่าขาสัญญาณ S/P ซึ่งจะส่งผลให้ค่าพารามิเตอร์บางส่วนที่ได้โปรแกรมลงไปนั้นหายไปด้วยนั้นเอง

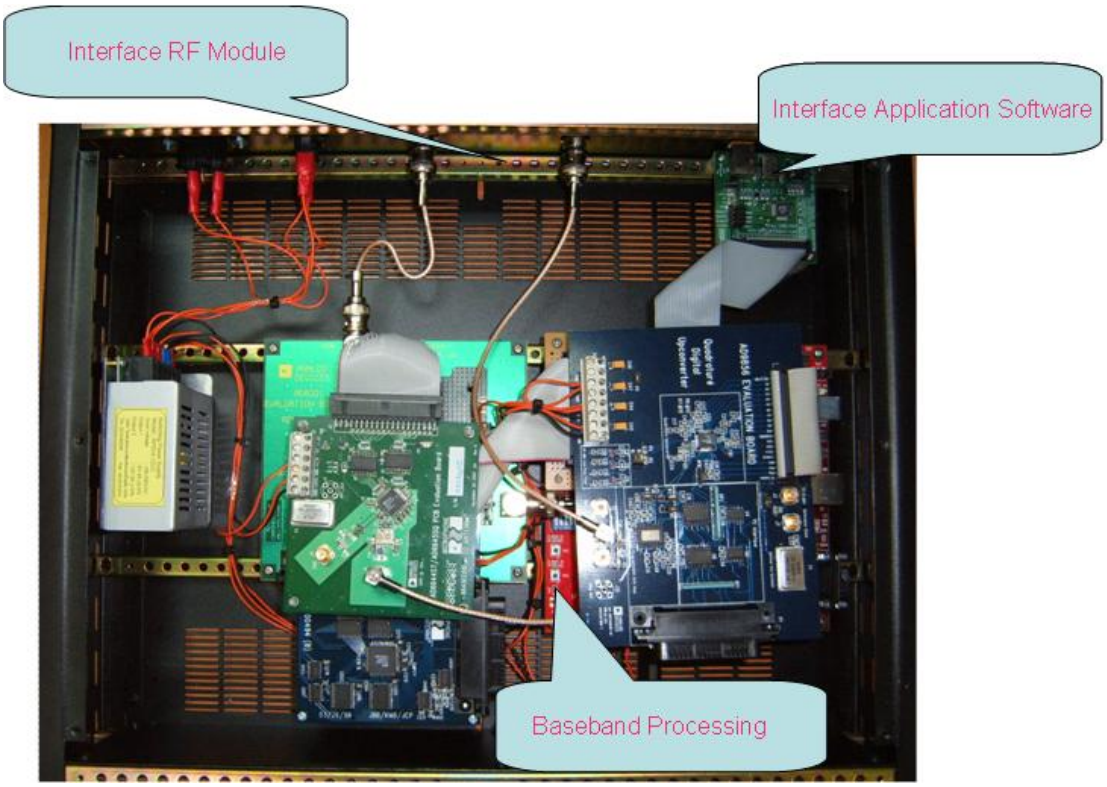
สำหรับบอร์ด AD6644 ก็ไม่ได้มีการเปลี่ยนแปลงเพิ่มเติมจากที่ได้เคยกล่าวไว้ในหัวข้อการพัฒนาส่วนภาคติดต่อกับสัญญาณวิทยุ ซึ่งจะรับข้อมูลอินพุทที่ความถี่ 75 MHz แต่ทำการชักสัญญาณที่ความถี่ 66.66 MHz ตามที่ได้กล่าวไว้แล้ว ซึ่งเหตุผลหนึ่งที่ใช้สัญญาณในการชักสัญญาณที่ความถี่นี้ เนื่องจากหาได้ง่ายในท้องตลาด ราคาไม่แพงเมื่อเทียบกับสัญญาณที่ความถี่ 75 MHz ซึ่งจะทำให้สัญญาณเป็นสัญญาณดิจิตอลเบสแบนด์ได้เลยทันที และอีกเหตุผลหนึ่งก็คือว่าตัวบอร์ด AD6644 ที่ได้เลือกใช้นั้นสามารถสนับสนุนการชักสัญญาณที่ความถี่สูงสุด 67 MHz เท่านั้นเอง โดยผลลัพธ์ในส่วนนี้จะเป็นไปตามภาพที่ 51

ส่วนบอร์ด AD8321 ที่ใช้สำหรับการปรับพลังงานนั้นการพัฒนาซอฟต์แวร์สำหรับการปรับพลังงานด้วยภาษา VHDL เป็นไปตามภาพที่ 82 ซึ่งจะตรวจสอบจากระดับสัญญาณที่เข้ามา เพื่อใช้ในการปรับ Gain



ภาพที่ 82 แนวทางการพัฒนาสำหรับบอร์ด AD8321

จะเห็นได้ว่าในการพัฒนาในขั้นตอนนี้เป็นการพัฒนาขึ้นเพื่อนำไปทดแทน หรือแทนที่ส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์ของจีน โดยการเชื่อมต่อจะเป็นในลักษณะของสัญญาณอนาลอกที่ความถี่ 65 MHz ในทางภาคส่งสัญญาณ และที่ความถี่ 75 MHz ที่ภาครับสัญญาณ ตามภาพที่ 83 ที่เป็นต้นแบบส่วนประมวลผลดิจิตอลเบสแบนด์ที่ได้พัฒนาขึ้นมา โดยมีความสามารถในการรับส่งข้อมูลได้สูง 1 Mbps และมี Coding Gain ที่ 6.5 dB รวมถึงสามารถติดต่อภาคสัญญาณวิทยุได้ที่ความถี่ 65 MHz ในภาคส่งสัญญาณและ 75 MHz ในภาครับสัญญาณ



ภาพที่ 83 ต้นแบบส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์

การพัฒนานี้เน้นไปในเรื่องการทำงาน 2 ส่วนก็คือ ส่วนแรกความสามารถในการรับส่งข้อมูลซึ่งทางจีนเองได้แค่ 512 Kbps ในขณะที่การพัฒนานี้สามารถทำได้ถึง 1 Mbps และส่วนที่สองคือประสิทธิภาพของตัวเข้ารหัส ซึ่งทั้งการพัฒนาของทั้งจีนและทางนี้เองก็ได้เท่ากันก็คือ 6.5 dB (โดยการพัฒนาของจีนเองใช้ทั้ง Reed Solomon และ Convolution Code แต่การพัฒนานี้ใช้เพียงแค่ Turbo Code เท่านั้น) ดังนั้นสามารถที่จะพัฒนาปรับปรุงประสิทธิภาพโดยการผสมการเข้ารหัส ถอดรหัสสัญญาณให้ซับซ้อนขึ้นได้ ก็จะสามารถมีประสิทธิภาพที่สูงกว่า แต่ในการพัฒนาในที่นี้จะใช้แค่เพียง Turbo Code ก็เพียงพอแล้ว

และอีกส่วนนึงก็คือความสามารถในการแก้ไขปัญหาความถี่เลื่อนที่ตัวต้นแบบที่ได้พัฒนาขึ้นมานั้นสามารถแก้ไขได้ถึง 4.5MHz เพียงพอต่อความต้องการของดาวเทียมอเนกประสงค์นี้ อีกทั้งยังสามารถแก้ไขเวลาและเฟสที่ตกค้างได้อีกด้วย โดยการแก้ไขทั้งหมดนี้ยังเกิดความผิดพลาดขึ้นอยู่บ้าง แต่ก็เล็กน้อยมากสามารถนำไปแก้ได้ในส่วนดิจิตอลเบสแบนด์ ซึ่งความผิดพลาดที่เกิดขึ้นมานั้นเป็นความผิดพลาดที่เกิดจากพวกสัญญาณรบกวนต่างๆ นั้นเอง

สรุปและข้อเสนอแนะ

สรุป

1. การรับส่งข้อมูลจะได้อัตราที่เร็วที่สุดที่ 1.3 Mbps และที่เร็วมากกว่านี้ไม่ได้ทั้งๆ ที่สามารถทำงานได้สูงกว่านี้ เนื่องจากข้อมจำกัดของ Hardware Ez-USB 1.1

2. ซอฟต์แวร์ที่ใช้งานทำให้เข้าใจง่ายต่อผู้ใช้งาน ไม่ว่าจะเป็นการรับส่งไฟล์ข้อมูล เสียง หรือแม้แต่การทดสอบประสิทธิภาพของตัวเข้ารหัสถอดรหัสสัญญาณด้วย

3. การทดสอบประสิทธิภาพ ผลปรากฏว่า Turbo Code ที่เลือกใช้จะได้ Coding Gain สูงถึง 6.5 dB (เปรียบเทียบกับที่ไม่ได้เข้ารหัส)

4. ในการพัฒนาความปลอดภัยจองข้อมูลจะใช้แนวคิด Public Key และ Private Key ในการพัฒนา โดยสามารถปรับเปลี่ยน Key ที่ใช้ได้เองตามความต้องการของผู้ใช้งาน

5. การเข้าจังหวะสัญญาณสามารถแก้ไขความถี่เลื่อนได้สูงถึง 4.5 MHz (เพียงพอกับความต้องการของดาวเทียมดวงนี้ที่ 1.5 MHz) และยังแก้ไขปัญหาความผิดพลาดที่เกิดจากเฟสและเวลาที่ไม่ตรงกันของเครื่องรับและเครื่องส่งได้

6. ในการแก้ไขความผิดพลาดที่เกิดจากเฟสตกค้าง เวลาที่ไม่ตรงกัน และความถี่เลื่อนนี้ จะไม่สามารถแก้ไขความผิดพลาดที่เกิดขึ้นทั้งหมดได้เนื่องจากว่า ความผิดพลาดที่เกิดขึ้นมีเหตุผลอื่นๆ ด้วยเช่น ความผิดพลาดที่เกิดจากสัญญาณรบกวน เป็นต้น ทำให้จะเห็นได้ว่าการเข้าจังหวะสัญญาณเมื่อแก้ไขความผิดพลาดไปแล้วยังมีความผิดพลาดให้เห็นอยู่ แต่ความผิดพลาดที่เหลือนี้นั้นสามารถแก้ไขได้ด้วยส่วนการถอดรหัสสัญญาณ ซึ่งจากมาตรฐานของตัวถอดรหัสแล้วพบว่าสามารถแก้ไขได้สูงถึง 20 เปอร์เซ็นต์ทีเดียวในบางรูปแบบของข้อมูล

7. ตัวต้นแบบระบบสื่อสารข้อมูลนี้สามารถทำงานได้โดยส่งสัญญาณที่ความถี่กลาง 65 MHz ในภาคส่ง และที่ 75 MHz ที่ภาครับ โดยสามารถรับส่งข้อมูลได้เองจากซอฟต์แวร์ประยุกต์

ข้อเสนอแนะ

1. ส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์สำหรับสถานีฐานที่พัฒนาขึ้นนี้ ยังขาดส่วนควบคุมที่สามารถติดต่อกับเลเยอร์บนซึ่งจะต้องจัดการเกี่ยวกับโปรโตคอลสื่อสารข้อมูล เพื่อที่จะตอบโต้คำสั่งรับการร้องขอสร้างช่องสัญญาณ และจัดการกับพารามิเตอร์ต่างๆ หรือถ้าเกิดความผิดพลาดขึ้นมาในการประมวลผลสัญญาณว่าควรจะจัดการยังไง

2. ในกรณีที่ต้องการขยายระบบให้สามารถทำงานได้ทั้งสถานีจำเป็นต้องมีการเพิ่มเติมในส่วนสัญญาณวิทยุ สายอากาศ รวมถึงส่วนติดตามและควบคุมการทำงานของดาวเทียมอีก ซึ่งราคาสำหรับภาคส่งสัญญาณในย่านความถี่ Ka-Band เป็นราคาที่แพงมาก จำเป็นต้องมีงบประมาณที่มากกว่าในปัจจุบัน

3. ในการนำต้นแบบส่วนประมวลผลสัญญาณดิจิตอลเบสแบนด์นี้ไปใช้งานแทนที่หรือทดแทนของจีนนั้น อาจจะไม่สามารถนำไปประกอบเข้ากับชุดส่วนสัญญาณวิทยุ และส่วนการควบคุมและติดตามดาวเทียมได้ เนื่องจากไม่ทราบถึงการเชื่อมต่อที่ใช้งานจริง จำเป็นต้องมีการตกลงในเรื่องดังกล่าว ก่อนที่จะนำไปใช้งานจริง

เอกสารและสิ่งอ้างอิง

โครงการวิจัยและพัฒนาระบบสื่อสารข้อมูลของสถานีภาคพื้นดินสำหรับดาวเทียมอเนกประสงค์ขนาดเล็ก, **กระทรวงเทคโนโลยีสารสนเทศและการสื่อสาร.**

สัณห์ อุทยารัตน์, ลูกา เนตรเนรมิตร และมงคล รักษาพัชรวงศ์. 2005. การวิจัยและพัฒนาระบบสื่อสารในสถานีภาคพื้นดินสำหรับดาวเทียมอเนกประสงค์ขนาดเล็ก. **งานประชุมวิชาการครั้งที่ 43 มหาวิทยาลัยเกษตรศาสตร์**, กรุงเทพ, 3 กุมภาพันธ์ 2548

Telemetry Channel Coding. Blue Book, Issue 6, **CCSDS**. Houston Texas USA. October 2002.

Man Young Rhee. 1989. Error Correction Coding Theory. **McGraw-Hill Pubilshing** 1989.

Emmanuel C. Ifeachor and Barrie W. Jervis. 1993. Digital Signal Processing: A Practical Approach. **Addison-Wesley**. 1993.

Baines, R. and D. Baines. 2003**.** A Total Cost Approach to Evaluating Different Reconfigurable Architectures for Baseband Processing in Wireless Receivers. **Communications Magazine, IEEE ,** 1 (41): 105 -113.

Zuo Peng. 2005. Preliminary Design Report for KABES Subsystem of Small Multi-Mission Satellite. Xi'An Institue of Space Radio Technology, **CAST**, June 2005.

Hiranya-ekaparb, A., L. Netneramit, N. Pimpuch, S. Sae-wong, S. Uttayarath, S. Netirojjanakul, W. Veerakachen, and M. Raksapatcharawong. 2005. Loopback Functional Test of 3G—WCDMA Baseband Processing, **The International Conference in ECTI**, Pattaya, May 2005.

picoChip Design Ltd, Design Programming & verification of 3G Basestations. Available source: http://www.picochip.com

Sae-wong, S., M. Raksapatcharawong and A. Hiranya-ekaparb. 2002.

An FPGA-based Baseband Processing Blocks in 3GPP Transport Channels, **The 2nd International Symposium on Communications and Information Technology**, Pattaya, October 2002.

_____ . 2002. An Efficient Implementation of an FPGA-based Viterbi Decoder for

3GPP Systems. 2002. **The 2nd International Symposium on Communications and Information Technology**, Pattaya, October 2002.

Uttayarath, S., N. Pimpuch, S. Netirojjanakul and M. Raksapatcharawong.

2003. Implementing 3GPP Baseband Processing for Mobile Station Using TMS320C6416 DSP, **The 3rd International Symposium on Communications and Information Technology**, Songkhla, September 2003.

The 3rd Generation Partnership Project (3GPP). 2001. TS 25.211 Physical channels and mapping of transport channels onto physical channels (FDD). Available source: http://www.3gpp.org

_____. 2001. TS 25.212 Multiplexing and channel coding (FDD).

Available source: http://www.3gpp.org