รายงานวิจัยฉบับสมบูรณ์

การเข้ารหัส-ถอดรหัสลับแบบ Chaotic สำหรับความปลอดภัยในการสื่อสารโดยอาศัยปรากฏการณ์ไม่เป็นเชิงเส้นในวงจรกรองสัญญาณดิจิตอล : การออกแบบและการสร้าง

Chaotic Crypto System for Communication Security Based on Nonlinear Phenomenon in Digital Filter : Design and Implementation

หัวหน้าโครงการวิจัย: ผศ.ดร. ศรวัฒน์ ชิวปรีชา

ผู้ช่วยวิจัย

นางสาว ณริสรา คงปรีชา

นาย ทศพล อโณทัยไพบูลย์

โครงการวิจัยงบประมาณเงินรายได้ ประจำปีงบประมาณ 2554

คณะวิศวกรรมศาสตร์

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญตาราง

หน้า

ตารางที่

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในระบบการสื่อสาร ความเป็นส่วนตัว (private) รวมทั้งความปลอดภัย (secure) ในการสื่อสารเป็นสิ่งสำคัญ ทั้งนี้เพื่อป้องกันมิให้ผู้ที่ไม่ได้มีส่วนเกี่ยวข้องสามารถรับรู้ข้อมูลข่าวสารนั้นได้ไม่ว่าจะโดยเจตนาหรือไม่เจตนา กระบวนการรักษาความเป็นส่วนตัวรวมทั้งรักษาความปลอดภัยในการสื่อสารจะเกี่ยวข้องกับเทคโนโลยีการปกปิดข้อมูลข่าวสารให้เป็นความลับ (Cryptography System) ในแง่ของการสื่อสารแบบอนาลอกวิธีการที่ใช้ในการในการปกปิดข้อมูลข่าวสารให้เป็นความลับมักจะเกี่ยวข้องกับการแปรเปลี่ยนของความถี่โดยอาจจะอาศัยหลักการทำ Frequency hopping เป็นต้น หรือทำการดัดแปลงสเปกตรัมของข้อมูลข่าวสารก่อนทำการส่งเข้าไปในระบบสื่อสารเลยก็เป็นได้ แต่โดยรวมทั่วไปวิธีการที่ใช้ในทางอนาลอกมักจะเรียกว่าการทำ scramble-descramble มากกว่าใช้คำว่า Cryptography ส่วนสำหรับวิธีการทางดิจิตอลซึ่งข้อมูลข่าวสารที่จะส่งเข้าไปในระบบสื่อสารก็อยู่ในรูปแบบของข้อมูลดิจิตอลด้วยนั้นวิธีการพื้นฐานที่สุดคือการทำกระบวนการที่เรียกว่าการทำ data scramble-descramble ไม่ว่าจะใช้วิธีการที่ง่ายสุดคือเพียงแค่ทำการสลับตำแหน่งของบิทข้อมูล (ซึ่งแน่นอนว่าความปลอดภัยของข้อมูลก็จะต่ำไปด้วย) หรือจะใช้ PN (Pseudo random noise) code ซึ่งเป็นรหัสแบบสุ่มทำการ modulo เข้ากับข้อมูลก่อนส่งเข้าไปในระบบสื่อสารก็ถือว่าเป็นรูปแบบหนึ่งของการเข้ารหัสลับแบบดิจิตอล แต่ด้วยวิธีการพื้นฐานดังกล่าวนี้หากมีผู้ไม่ประสงค์ดีทำการดักโจรกรรมข้อมูลข่าวสารไปได้ก็จะไม่เป็นการยากมากนักในการที่จะหาความสัมพันธ์หรือหาวิธีการแก้กลับเพื่อให้ได้ซึ่งข้อมูลจริงก่อนที่จะทำการปกปิดข้อมูลให้เป็นความลับ ดังนั้นจึงเป็นที่มาของกระบวนการเข้ารหัส-ถอดรหัสเพื่อปกปิดข้อมูลให้เป็นความลับ (Encryption-Decryption) ซึ่งคำว่า Cryptography ที่กล่าวข้างต้นจะเกี่ยวข้องโดยตรงกับกระบวนการเข้ารหัส-ถอดรหัสเพื่อปกปิดข้อมูลให้เป็นความลับที่กล่าวนี้

เทคนิควิธีการที่ใช้สำหรับการเข้ารหัส-ถอดรหัสเพื่อปกปิดข้อมูลให้เป็นความลับนั้นมีเทคนิควิธีการที่หลากหลายไม่ว่าจะอาศัยอัลกอริธึม (Algorithm) เช่น DES, AES, RSA หรือ Hash function เป็นต้น ซึ่งล้วนแล้วแต่เป็นวิธีการปกปิดข้อมูลให้เป็นความลับที่เรียกว่ามีระดับความปลอดภัยสูงและยากที่จะทำการถอดรหัสหากไม่รู้ถึงกุญแจที่ใช้ในการเข้ารหัส (ซึ่งมีทั้งรูปแบบที่เป็น private key และ public key) แต่ถึงแม้ algorithm ต่างๆ ซึ่งมีให้เลือกใช้มากมายนี้จะสามารถให้ความปลอดภัยของข้อมูลที่สูงก็ตามแต่สิ่งที่ตามมาก็คือความซับซ้อนในกระบวนการที่ใช้ในการเข้ารหัส-ถอดรหัสตามวิธีการที่กำหนดไว้ใน algorithm ต่างๆ ก็จะมีความซับซ้อนมากตามไปด้วย โดยเฉพาะอย่างยิ่งถ้าพิจารณาในแง่ของการออกแบบให้เป็นวงจรฮาร์ดแวร์หรือวงจรรวมที่ประยุกต์ใช้งานเฉพาะด้าน (Application Specific Integrated Circuits: ASIC) ด้วยแล้วนั้น ก็จะมีความซับซ้อนในการออกแบบสูง วงจรที่ได้ก็จะต้องใช้ทรัพยากรในการออกแบบที่มาก (เช่นจำนวนของ logic cells ในการออกแบบ AISC) ซึ่งก็จะส่งผลกับการบริโภคพลังงาน (Power consumption) ทำให้ระยะเวลาในการใช้งานอุปกรณ์ที่มีวงจรเข้ารหัส-ถอดรหัสนี้อยู่ร่วมด้วยนั้นสั้นเนื่องจากแบตเตอรี่หมด และถ้าพิจารณาถึงการใช้งานจริงโดยส่วนใหญ่ของ algorithm ต่างๆ ที่กล่าวมานั้นมักจะอยู่ในรูปแบบของ software เสียมากกว่าเพราะการเขียนโปรแกรมให้ทำงานเป็นการเข้ารหัส-ถอดรหัสนั้น ความซับซ้อนของ algorithm จะมีผลแค่ในส่วนของการทำงานของโปรแกรมซึ่งอาจใช้เวลานานเท่านั้น ผลของความซับซ้อนที่จะส่งผลถึงจำนวนทรัพยากรหรืออุปกรณ์ที่ใช้ และการบริโภคพลังงานก็จะไม่ได้นำมาคำนึงถึง ดังนั้นถ้าลักษณะการประยุกต์ใช้งานการเข้ารหัส-ถอดรหัสเพื่อปกปิดข้อมูลให้เป็นความลับนั้นมีเป้าหมายที่จะใช้งานในลักษณะของฮาร์ดแวร์ที่กินพลังงานต่ำ เช่นใช้ในระบบวิทยุสื่อสาร (ซึ่งการดักฟังวิทยุสื่อสารโดยทั่วไปนั้นทำได้ง่ายมาก) หรืออุปกรณ์อื่นที่ใช้งานในลักษณะเคลื่อนที่ (Mobile) ซึ่งใช้แหล่งพลังงานจากแบตเตอรี่ โดยที่ระดับความปลอดภัยของข้อมูลก็ยังอยู่ในระดับปานกลาง-สูงอยู่ หลักการที่จะนำเสนอในโครงการวิจัยชิ้นนี้จะช่วยตอบโจทย์ดังที่กล่าวมาทั้งหมดได้

หลักการที่จะนำเสนอในโครงการวิจัยนี้ยังคงเป็นการเข้ารหัส-ถอดรหัสเพื่อปกปิดข้อมูลให้เป็นความลับในรูปแบบการดำเนินการแบบดิจิตอลโดยมิได้ใช้ algorithm ใดดังที่ได้กล่าวมาก่อนหน้า แต่อาศัยปรากฏการณ์การเกิด Chaos (ความยุ่งเหยิง, ไม่มีระเบียบ, สับสนอลหม่าน) ซึ่งเกิดขึ้นในวงจรกรองสัญญาณดิจิตอลมาใช้เป็นกุญแจหลักในการออกแบบวงจรที่ใช้สำหรับเข้ารหัส-ถอดรหัสเพื่อปกปิดข้อมูลให้เป็น

ความลับ โดยเราจะเรียกวงจรที่นำเสนอนี้ว่า วงจรเข้ารหัส-ถอดรหัสลับแบบ Chaotic ด้วยพฤติกรรมของ Chaos ซึ่งยากที่จะคาดเดาถึงแนวโน้มของการเปลี่ยนแปลง ทำให้วงจรเข้ารหัส-ถอดรหัสลับแบบ Chaotic นี้ยังคงให้ระดับความปลอดภัยของข้อมูลที่อยู่ในระดับปานกลาง-สูงได้ การประยุกต์ใช้งานโดยส่วนมากของวงจรเข้ารหัส-ถอดรหัสลับแบบ Chaotic ที่มีการอ้างถึงในบทความก่อนหน้ามักจะนำไปใช้ในระบบสื่อสารแบบที่ต้องการความเป็นส่วนตัวในการสื่อสาร (private communication) นั่นคือจะถูกใช้งานในลักษณะสร้างความเป็นส่วนตัวในการใช้งานระบบสื่อสารเฉพาะกลุ่มโดยมิให้ผู้ที่ไม่เกี่ยวข้องรับรู้ข้อมูลข่าวสารได้ เช่นระบบสื่อสารที่ใช้ในองค์กร เป็นต้น ดังนั้นเพื่อให้มุมมองในการวิจัยและพัฒนาโครงการวิจัยนี้ชัดเจนยิ่งขึ้น ผู้วิจัยจะมองภาพของระบบสื่อสารแบบส่วนตัว (private communication) เป็นลักษณะของการประยุกต์ใช้งานเพื่อรักษาความปลอดภัยของข้อมูลข่าวสารในระบบวิทยุสื่อสารทางการทหารหรือเจ้าหน้าที่ตำรวจไม่ให้ถูกดักฟังจากศัตรูหรือฝ่ายตรงข้าม ซึ่งลักษณะการประยุกต์ใช้งานแบบนี้ระดับความปลอดภัยของข้อมูลสามารถที่จะอยู่ในระดับปานกลางได้ ซึ่งจะส่งผลดีกับความซับซ้อนซึ่งไม่มากเกินไปทำให้เมื่อทำการออกแบบให้เป็นชุดของวงจรเข้ารหัส-ถอดรหัส และนำไปใช้งานจริงก็จะไม่กินพลังงานจากแบตเตอรี่มากเกินไป

วงจรเข้ารหัส-ถอดรหัสลับแบบ Chaotic สำหรับระบบสื่อสารแบบส่วนตัวที่จะนำเสนอนี้อาศัยการเกิด Chaos ในวงจรกรองสัญญาณดิจิตอลมาเป็นกลไกหลักในการทำงาน วงจรกรองสัญญาณดิจิตอลซึ่งจัดว่าเป็นระบบแบบเชิงเส้นไม่แปรเปลี่ยนตามเวลา (Linear Time-invariant : LTI) นั้น เมื่อถูกนำมาทำการสร้าง (Implement) โดยเฉพาะอย่างยิ่งการนำมาสร้างเป็นฮาร์ดแวร์ซึ่งจะต้องทำการแทนทั้งสัญญาณ ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณ และการดำเนินการทางคณิตศาสตร์ ด้วยขนาดของความยาวคำ (wordlength) ที่มีขนาดจำกัด หรือกล่าวอย่างง่ายๆ ว่าการดำเนินการทั้งหมดของวงจรกรองสัญญาณเชิงเลขต้องอยู่บนระบบตัวเลขแบบ fixed-point ผลของ finite wordlength effect นี้ทำให้ในทางปฏิบัติวงจรกรองสัญญาณดิจิตอลซึ่งเป็นระบบ LTI เกิดปรากฏการณ์ที่ไม่เป็นเชิงเส้นขึ้น (Nonlinear phenomenon) และผลของ nonlinear phenomenon นี้เองที่ทำไปสู่การเกิด Chaos ขึ้นในวงจรกรองสัญญาณดิจิตอล และผลของการเกิด Chaos นี้เองก็ถูกนำมาใช้งานในการสร้างเป็นวงจรเข้ารหัส-ถอดรหัสลับแบบ Chaotic ดังนั้นลักษณะทางกายภาพของวงจรเข้ารหัส-ถอดรหัส ที่นำเสนอนี้ก็จะมีลักษณะเป็นวงจรกรองสัญญาณดิจิตอล (แต่ไม่ได้มีวัตถุประสงค์เพื่อ

เอาไว้กรองสัญญาณ) นั่นเอง สิ่งที่ใช้ในการกำหนดคุณลักษณะหรือสิ่งที่เปรียบเสมือนเป็นกุญแจ (key) ซึ่งใช้โดยทั่วไปในงานด้าน Cryptography ก็คือค่าสัมประสิทธิ์ของวงจรกรองสัญญาณนั่นเอง ซึ่งจะใช้ในกระบวนการเข้ารหัสเพื่อให้ออกมาเป็นข้อมูลลับ (cipher) และใช้ในกระบวนการถอดรหัสเพื่อให้ได้ข้อมูลต้นฉบับกลับมาด้วย และเนื่องจากโครงสร้างของวงจรเข้ารหัส-ถอดรหัสลับแบบ Chaotic เหมือนกันกับโครงสร้างของวงจรกรองสัญญาณดิจิตอล ดังนั้นเทคนิควิธีการต่างๆ ที่ใช้สำหรับการลดความซับซ้อนของโครงสร้างของวงจรกรองสัญญาณดิจิตอล (Low-complexity) ก็สามารถที่จะนำมาประยุกต์ใช้กับการออกแบบวงจรเข้ารหัส-ถอดรหัสนี้ได้ด้วย ส่งผลให้ได้วงจรเข้ารหัส-ถอดรหัสที่เป็น Low-power consumption อีกด้วย

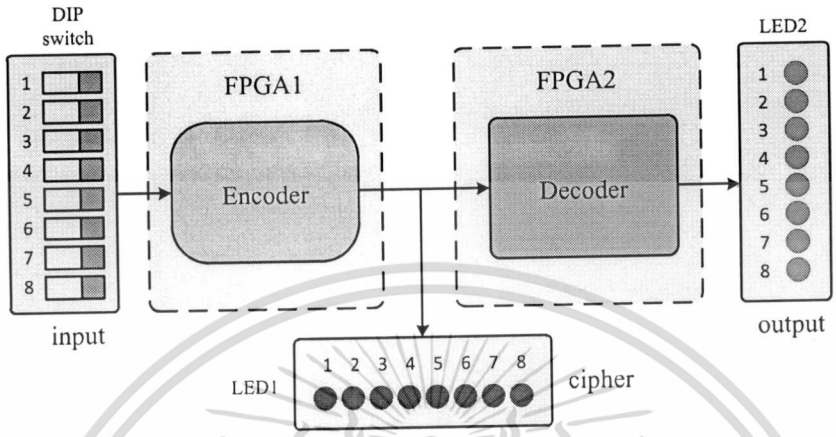
1.2 วัตถุประสงค์

1. ศึกษาหลักการเกิด Chaos ในวงจรกรองสัญญาณดิจิตอลและนำมาประยุกต์ใช้งานด้านความปลอดภัยของข้อมูลในการสื่อสารแบบส่วนตัว (Private Communication)
2. นำเสนอวิธีการออกแบบและโครงสร้างของวงจรเข้ารหัส-ถอดรหัสลับแบบ Chaotic เพื่อปกปิดข้อมูลให้เป็นความลับโดยอาศัย Chaos ที่เกิดขึ้นจากความไม่เป็นเชิงเส้นในวงจรกรองสัญญาณดิจิตอล
3. สร้างความเชื่องโยงและแสดงให้เห็นถึงการประยุกต์ใช้งานวงจรกรองสัญญาณดิจิตอลในงานด้านการรักษาความปลอดภัยของข้อมูลข่าวสารในระบบสื่อสาร (Digital filter ไม่ได้เอาไว้กรองสัญญาณแต่เพียงอย่างเดียว, เพราะ นศ. ส่วนใหญ่เมื่อเห็นคำว่า filter ก็มักจะมองแค่ใช้กรองสัญญาณเท่านั้น)
4. นำหลักการที่ใช้ในการสร้าง (Implementation) วงจรกรองสัญญาณดิจิตอลให้เป็นฮาร์ดแวร์มาใช้ในการสร้างต้นแบบฮาร์ดแวร์ของชุดวงจรเข้ารหัส-ถอดรหัส บนอุปกรณ์ FPGA และนำไปทดสอบกับสัญญาณจริง
5. องค์ความรู้ที่ได้จากโครงการวิจัยสามารถนำไปต่อยอดพัฒนาเป็นผลิตภัณฑ์เพื่อใช้งานจริงสำหรับระบบสื่อสารภายในหน่วยงานหรือองค์กร

1.3 ขอบเขตของโครงการวิจัย

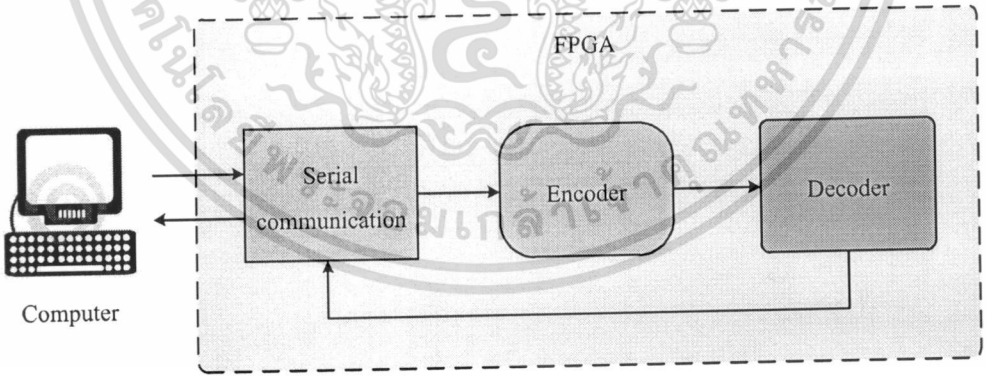
1. ทำการศึกษาถึงการเกิด Chaos ในวงจรกรองสัญญาณดิจิตอลซึ่งมีที่มาจาก finite wordlength effect ในระบบตัวเลขแบบ fixed-point ของวงจรกรองสัญญาณดิจิตอลที่นำมาสร้างเป็นวงจรหรือฮาร์ดแวร์จริงซึ่งต้องใช้จำนวนบิทในการแทนสัญญาณ ค่าสัมประสิทธ์ และตัวดำเนินการทางคณิตศาสตร์ ที่มีจำนวนจำกัด
2. ศึกษาหลักการออกแบบวงจรเข้ารหัส-ถอดรหัสลับแบบ Chaotic ซึ่งอาศัยการเกิด Chaos ในวงจรกรองสัญญาณดิจิตอล หลักการหาค่าสัมประสิทธิ์ของวงจรกรองสัญญาณที่ใช้เสมือนเป็นกุญแจสำหรับการเข้ารหัสและถอดรหัส ความเหมาะสมของโครงสร้างของวงจรกรองสัญญาณ คุณลักษณะของข้อมูลลับที่ได้ (Cipher) จากการเข้ารหัส
3. ทำการออกแบบวงจรเข้ารหัส-ถอดรหัสลับแบบ Chaotic และการจัดโครงสร้างของวงจรกรองสัญญาณที่ทำหน้าที่เป็นตัวเข้ารหัส-ถอดรหัส ทำการจำลองการทำงาน (Simulation) ด้วยคอมพิวเตอร์เพื่อดูผลที่ได้จากการเข้ารหัส-ถอดรหัสในเชิงของหลักการ ทำการวิเคราะห์ผลที่ได้ในแง่ความปลอดภัยของข้อมูลหลังจากผ่านการเข้ารหัสในเชิงสถิติโดยใช้ฟังก์ชั่นความหนาแน่นของความน่าจะเป็น (Probability density function : PDF) ในการบ่งบอกความหมาย
4. สร้างฮาร์ดแวร์ต้นแบบของวงจรเข้ารหัส-ถอดรหัสลับแบบ Chaotic โดยใช้ภาษา VHDL และสังเคราะห์เป็นวงจรพร้อมทั้งโปรแกรมลงบนอุปกรณ์ FPGA สำหรับทดสอบการทำงาน ซึ่งฮาร์ดแวร์ต้นแบบนี้จะเป็นการทำงานด้วยระบบตัวเลขแบบ fixed-point โดยแท้
5. ทดสอบการทำงานของฮาร์ดแวร์ต้นแบบเปรียบเทียบกับผลที่ได้จากการจำลองการทำงานว่ามีความ แตกต่างกันของ Chaos ที่เกิดขึ้นในส่วนของฮาร์ดแวร์จริงกับที่จำลองการทำงานหรือไม่ เนื่องจากจะส่งผลโดยตรงกับความถูกต้องในการทำงานของฮาร์ดแวร์จริง จากนั้นทำการทดสอบการทำงานของวงจรเข้ารหัส-ถอดรหัสเพื่อปกปิดข้อมูลให้เป็นความลับที่เป็นฮาร์ดแวร์จริงกับสัญญาณจริงเพื่อดูผลลัพธ์ที่ได้จากการเข้ารหัสรวมทั้งผลที่ได้จากการถอดรหัสกลับคืน วิเคราะห์ผลที่ได้ ทำการแก้ไขปรับปรุง

1.4 บล็อกไดอะแกรมของชิ้นงานที่พัฒนาขึ้นในโครงการวิจัย



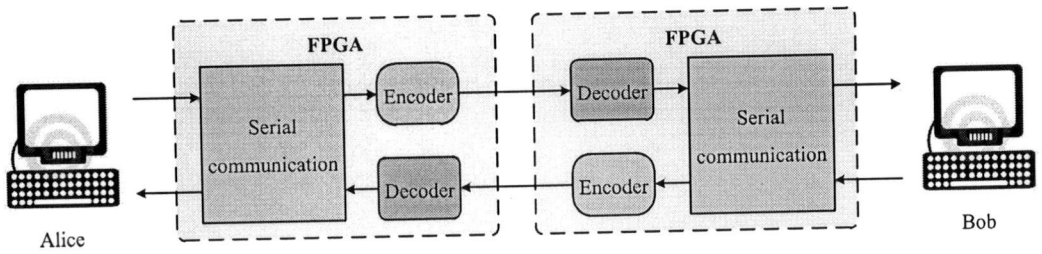
รูปที่ 1.1 บล็อกไดอะแกรมการเข้ารหัส-ถอดรหัสผ่านการรับส่งข้อมูลแบบขนานด้วย FPGA

จากรูปที่ 1.1 กำหนดให้อินพุตขนาด 8 บิต ป้อนข้อมูลโดยผ่าน DIP switch ไปยังวงจรเข้ารหัสของ FPGA1 ซึ่งอินพุตที่ผ่านการเข้ารหัสแล้ว (cipher) จะแสดงผลผ่าน LED1 จากนั้นจะส่งข้อมูลแบบขนานไปยังวงจรถอดรหัสของ FPGA2 และแสดงผลของเอาต์พุตที่ถูกถอดรหัสแล้วออกมาผ่าน LED2



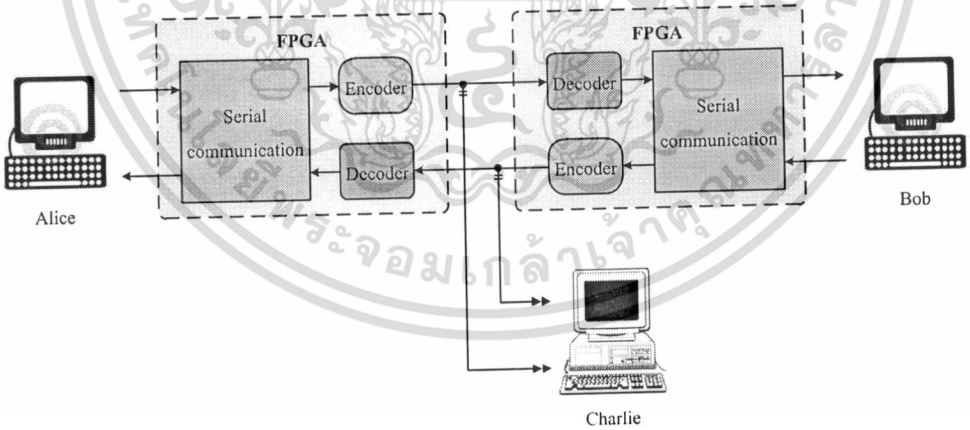
รูปที่ 1.2 บล็อกไดอะแกรมการเข้ารหัส-ถอดรหัสข้อมูลผ่านการรับส่งข้อมูลแบบอนุกรม
โดยใช้คอมพิวเตอร์เครื่องเดียว

จากรูปที่ 1.2 เป็นการเข้ารหัส-ถอดรหัสข้อมูลตัวอักษรด้วย FPGA ผ่านการรับส่งข้อมูลแบบอนุกรมกับคอมพิวเตอร์ที่เป็นทั้งตัวป้อนอินพุตและรับเอาต์พุตมาแสดงผล



รูปที่ 1.3 บล็อกไดอะแกรมการเข้ารหัส-ถอดรหัสข้อมูลผ่านการรับส่งข้อมูลแบบอนุกรมระหว่างเครื่องคอมพิวเตอร์สองเครื่อง

จากรูปที่ 1.3 เมื่อคอมพิวเตอร์ Alice ส่งข้อมูลตัวอักษรผ่านทางพอร์ตสื่อสารอนุกรมไปยังอุปกรณ์ FPGA จะผ่านวงจรสื่อสารอนุกรม (Serial Communication) ไปยังวงจรเข้ารหัส (Encoder) จากนั้นจะได้ข้อมูลที่ถูกเข้ารหัสแล้วส่งยังวงจรถอดรหัส (Decoder) ของฝั่งรับ วงจรนี้จะทำการถอดรหัสลับของข้อมูลและส่งข้อมูลผ่านไปยังวงจรสื่อสารอนุกรม (Serial Communication) เพื่อส่งข้อมูลผ่านไปยังพอร์ตสื่อสารอนุกรมของคอมพิวเตอร์ Bob ที่เป็นฝั่งรับ ในทางกลับกันเครื่องคอมพิวเตอร์ Bob ต้องการส่งข้อมูลกลับไปยังเครื่องคอมพิวเตอร์ Alice ก็จะผ่านขั้นตอนการทำงานข้างต้นเหมือนเดิมเช่นเดียวกัน



รูปที่ 1.4 บล็อกไดอะแกรมการเข้ารหัส-ถอดรหัสข้อมูลผ่านการรับส่งข้อมูลแบบอนุกรมระหว่างเครื่องคอมพิวเตอร์สองเครื่องโดยมีคอมพิวเตอร์อีกเครื่องหนึ่งดักจับข้อมูล

จากรูปที่ 1.4 คือการรับส่งข้อมูลแบบอนุกรมระหว่างคอมพิวเตอร์ 2 เครื่องเช่นเดียวกับรูปที่ 1.3 แต่จะมีคอมพิวเตอร์ Charlie เข้ามาดักจับข้อมูลในขณะที่คอมพิวเตอร์ Alice และคอมพิวเตอร์ Bob นั้นกำลังรับส่งข้อมูล เพื่อทดสอบความแข็งแกร่งของข้อมูลเมื่อถูกเข้ารหัส

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงทฤษฎีเคออส (Chaos theory) ที่จะใช้ในการอธิบายพฤติกรรมของวงจรกรองสัญญาณดิจิตอลที่เกิดความไม่เป็นเชิงเส้นจากการล้นแบบส่วนเติมเต็มสองซึ่งจะใช้ในการเข้ารหัสและถอดรหัสของสัญญาณ, ค่า Autocorrelationที่ใช้ในการตรวจสอบความเหมือนกันของสัญญาณ, ค่า Euclidean distanceและค่า PSNR (Peak Signal to Noise Ratio)ใช้วัดความแตกต่างระหว่างสัญญาณดั้งเดิมกับสัญญาณที่ได้ทำการถอดรหัส และ การสื่อสารข้อมูลแบบอนุกรม USB ซึ่งใช้สำหรับการติดต่อระหว่าง Computer และ FPGA

2.1 ทฤษฎีเคออส (Chaos theory)

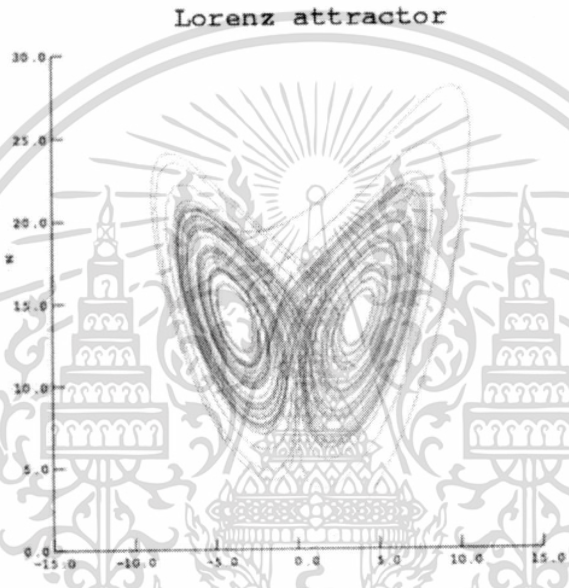
ทฤษฎีเคออสเป็นทฤษฎีที่อธิบายถึงลักษณะพฤติกรรมของระบบพลวัต (ระบบที่มีการเปลี่ยนแปลงตามเวลาที่เปลี่ยนไป) โดยลักษณะการเปลี่ยนแปลงของระบบที่เรียกว่าเคออติกนี้จะมีลักษณะที่ปั่นป่วนจนดูเหมือนสุ่มหรือไร้ระเบียบ (Random/Stochastic) แต่จริงๆ แล้วระบบเคออติกนี้เป็นระบบที่มีระเบียบ (Deterministic) ซึ่งในทางคณิตศาสตร์และฟิสิกส์ให้คำจำกัดความของระบบเคออติกว่าเป็นระบบไม่เชิงเส้น (Nonlinear system) ประเภทหนึ่งที่มีความไวต่อค่าเงื่อนไขเริ่มต้นสูงเปรียบได้กับประโยคที่ว่า "เด็ดดอกไม้สะเทือนถึงดวงดาว" หรือ "ผีเสื้อขยับปีกทำให้เกิดพายุ" (จาก "Butterfly Effect") จึงมีคนจำนวนไม่น้อยที่ตีความคำพูดนี้ในลักษณะของขนาดความรุนแรงของผลลัพธ์เท่านั้นซึ่งในความจริงแล้วระบบเคออติกไม่จำเป็นต้องแตกต่างกันในแง่ขนาดของผลลัพธ์เสมอไปแต่อาจแตกต่างกันในแง่ของพฤติกรรมการเปลี่ยนแปลงก็ได้ ตัวอย่างเช่น ถ้ามีระบบอยู่สองระบบแล้วกำหนดให้ค่าเงื่อนไขเริ่มต้นต่างกันเพียงเล็กน้อย การเปลี่ยนแปลงของระบบทั้งสองนั้นจะมีลักษณะที่คล้ายคลึงกันมากในขณะเริ่มต้น แต่เมื่อเวลาผ่านไปการเปลี่ยนแปลงนั้นแทบจะเรียกได้ว่าไม่มีอะไรที่เหมือนกันเลย

2.1.1 ประวัติของทฤษฎีเคออส

จุดเริ่มต้นของทฤษฎีเคออสนี้สามารถสืบย้อนกลับไปได้ถึงในช่วงปี พ.ศ. 2443 (ค.ศ.1900) จากการศึกษาปัญหาวงโคจรของวัตถุสามชิ้นในสนามแรงดึงดูดระหว่างกัน ซึ่งมีชื่อเรียกอย่างเป็นทางการว่า ปัญหาสามวัตถุ (Three-body problem) โดย อองรี ปวงกาเร(Henri Poincare) ซึ่งเขาได้ค้นพบว่าวงโคจรที่ศึกษานั้นอาจจะมีลักษณะที่ไม่ได้เป็นวงรอบ (Periodic) คือไม่ได้มีทางวิ่งซ้ำเป็นวงรอบยิ่งไปกว่านั้นวงโคจรก็ไม่ได้ขยายวงออกไปเรื่อยๆ คือมีลักษณะที่ลู่เข้าหาจุดใดๆต่อมาได้มีการศึกษาถึงปัญหาสมการเชิงอนุพันธ์ไม่เป็นเชิงเส้นที่เกี่ยวข้อง โดยที่ เบอร์คอฟ (G.D. Birkhoff)นั้นศึกษาปัญหาสามวัตถุคอลโมโกรอฟ (AndreyNikolaevich Kolmogorov) ศึกษาปัญหาความปั่นป่วน (หรือ เทอร์บิวเลนซ์) และปัญหาเกี่ยวกับดาราศาสตร์ ส่วนคาร์ทไรท์ (M.L. Cartwright) และ ลิตเติลวูด (J.E. Littlewood)นั้นศึกษาปัญหาทางวิศวกรรมการสื่อสารด้วยคลื่นวิทยุ สเมล (Stephen Smale) อาจเป็นนักคณิตศาสตร์คนแรกที่ทำการศึกษาถึงปัญหาทางด้านพลศาสตร์ของระบบไม่เป็นเชิงเส้น แต่ก็ได้มีการสังเกตพบพฤติกรรมความอลวนในการเคลื่อนที่ของของไหล และในการออสซิลเลทแบบไม่เป็นวงรอบของวงจรวิทยุซึ่งไม่มีทฤษฎีใดในขณะนั้นสามารถอธิบายพฤติกรรมเหล่านี้ได้ ความตื่นตัวในการพัฒนาทฤษฎีเคออสนี้เกิดขึ้นในช่วงกลางของศตวรรษที่ 20 เมื่อเป็นที่รู้กันว่าทฤษฎีของระบบเชิงเส้นนั้นไม่สามารถใช้อธิบายพฤติกรรมบางอย่าง แม้กระทั่งพฤติกรรมของระบบที่ไม่ซับซ้อนอย่างได้ และอีกปัจจัยหนึ่งที่ส่งผลให้การพัฒนาของทฤษฎีเคออสเป็นไปได้อย่างรวดเร็วก็คือ การใช้คอมพิวเตอร์ช่วยในการคำนวณทฤษฎีเคออส ซึ่งโดยส่วนใหญ่จะมีลักษณะที่เป็นการคำนวณค่าแบบซ้ำๆ จากสูตรคณิตศาสตร์จึงสามารถใช้คอมพิวเตอร์ช่วยในการคำนวณได้อย่างมีประสิทธิภาพ

เอ็ดเวิร์ด ลอเรนซ์ (Edward Lorenz) เป็นผู้ริเริ่มบุกเบิกทฤษฎีเคออส เขาได้สังเกตพฤติกรรมแบบเคออสในขณะทำการทดลองทางด้านการพยากรณ์อากาศในปี ค.ศ. 1961ลอเรนซ์ใช้คอมพิวเตอร์จำลองการทำงานแบบจำลองสภาพอากาศ ซึ่งในการคำนวณครั้งถัดมาเขาไม่ต้องการเริ่มจำลองการทำงานจากจุดเริ่มต้นใหม่เพื่อประหยัดเวลาในการคำนวณ เขาจึงใช้ข้อมูลในการคำนวณก่อนหน้านี้เพื่อเป็นค่าเริ่มต้น ปรากฏว่าค่าที่คำนวณได้มีความแตกต่างไปจากเดิมอย่างสิ้นเชิง เขาพบว่าสาเหตุเกิดจากการปัดเศษของค่าที่พิมพ์ออกมาจากค่าที่ใช้ในคอมพิวเตอร์ซึ่งมีค่าน้อยมากแต่สามารถนำไปสู่ความแตกต่างอย่างมากมายเรียกว่า ความไวต่อค่าเงื่อนไขเริ่มต้นสำหรับ คำว่า "butterfly effect" ซึ่งเป็นคำที่นิยมใช้เมื่อกล่าวถึงทฤษฎีเคออสนั้นมีที่มาไม่

ชัดเจนแต่เริ่มปรากฏแพร่หลายหลังจากการบรรยายของลอเรนซ์ในปี ค.ศ. 1972 ภายใต้ชื่อหัวข้อ "Doesthe Flap of a Butterfly's Wings in Brazil Set Off a Tornado in Texas?" นอกจากนี้แล้วยังอาจมีส่วนมาจากรูปที่ 2.1 ซึ่งเป็นรูปแนวโคจรของตัวดึงดูดลอเรนซ์ (Lorenz attractor) ที่มีรูปร่างคล้ายปีกผีเสื้อ ซึ่งเขาได้ตีพิมพ์ในบทความวิชาการก่อนหน้านี้ ส่วนคำว่า "Chaos" (เคออส) บัญญัติขึ้นโดยนักคณิตศาสตร์ประยุกต์ เจมส์ เอ ยอร์ค (James A. Yorke)



รูปที่ 2.1 แนวโคจรของตัวดึงดูดลอเรนซ์

2.1.2 นิยามของเคออส

คำว่า เคออส ตามความหมายในพจนานุกรมหมายถึง ความสับสนสับวุ่นวายความโกลาหล ความอลหม่าน แต่เคออสที่เราศึกษานั้นคือ เคออสในทางคณิตศาสตร์ หรือ Deterministic Chaos มีความหมายคือ สภาพหรือกระบวนการที่ไม่มีเสถียรภาพ (Unstable) หากมีการกระทบเพียงเล็กน้อย อาจจะทำให้เกิดสัญญาณที่ไม่เป็นเส้นตรง อาจจะคดเคี้ยว กวัดแกว่ง หรือในบางครั้งอาจจะเกิดการกระโดดฉับพลัน ดังนั้นผลลัพธ์ที่เกิดขึ้นจึงไม่สามารถคาดเดาหรือทำนายได้

พฤติกรรมแบบเคออส (Chaos behavior) เป็นพฤติกรรมที่ดูไร้ระเบียบเหมือนว่าเกิดขึ้นอย่างสุ่ม (Random) แต่ที่จริงเป็นพฤติกรรมที่กำหนดได้และสามารถอธิบายได้ด้วยสมการทางคณิตศาสตร์ดังนั้นพฤติกรรมแบบเคออสจึงแฝงไปด้วยความเป็นระเบียบ(Order)จุดที่เป็นประเด็นสำคัญของทฤษฎีเคออสคือระบบที่มีพฤติกรรมแบบเคออสจะไวต่อการเปลี่ยนแปลงของค่าเงื่อนไขเริ่มต้น (Sensitivityto initialconditions) จึงทำให้ไม่สามารถทำนายระบบเคออติกได้ในระยะยาว (Long-term unpredictable)

2.1.3 คุณลักษณะของเคออส

2.1.3.1 มีคุณสมบัติแบบไม่เป็นเชิงเส้น (Non-linearity)

ระบบแบบไม่เป็นเชิงเส้นผลลัพธ์ของระบบทั้งหมดจะไม่เท่ากับผลรวมของผลลัพธ์ของระบบย่อยๆ รวมกัน (โดยอาจจะมากหรือน้อยก็ได้) แต่มีข้อพึงระวังก็คือ การที่กล่าวว่าระบบเคออติกทุกระบบจะต้องเป็นระบบที่ไม่เป็นเชิงเส้นนั้นไม่ได้หมายความว่าระบบที่ไม่เป็นเชิงเส้นทุกระบบจะเป็นระบบเคออติกเสมอไป

2.1.3.2 ไม่ใช่ระบบที่เกิดขึ้นแบบสุ่มคือมีสมการอธิบาย (Deterministic)

คือเป็นระบบที่สามารถกำหนดได้หรือกล่าวอีกแบบหนึ่งก็คือในระบบเคออติกนั้น พฤติกรรมทั้งหลายจะเกิดขึ้นภายใต้กฎเกณฑ์ที่แน่นอน ดังนั้นเหตุการณ์ที่ไม่สามารถทำนายล่วงหน้าได้อย่างเช่นการทอดลูกเต๋าจึงไม่ใช่ความเป็นเคออสแต่เป็นการสุ่ม เพื่อป้องกันการเข้าใจผิดว่าระบบเคออติกเป็นระบบแบบสุ่มจึงมีคนเรียกระบบเคออติกว่า Deterministic chaotic

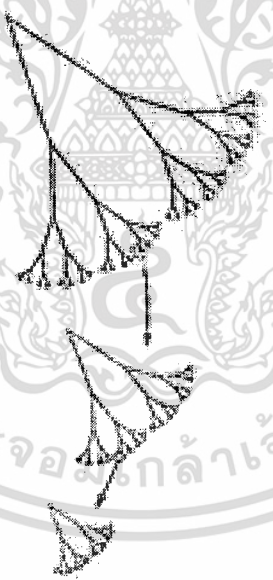
2.1.3.3 ไวต่อค่าเงื่อนไขเริ่มต้น (Sensitivity to initial conditions)

ฟังก์ชันการเริ่มต้นที่เงื่อนไขต่างกันเพียงนิดเดียวก็อาจจะทำให้ผลลัพธ์ของระบบในตอนสุดท้ายต่างกันอย่างมาก ซึ่งสาเหตุที่ทำให้ระบบเคออติกมีความไวต่อค่าเงื่อนไขเริ่มต้นนั้นก็เพราะว่ามันจะขยายความแตกต่างของผลลัพธ์ให้เพิ่มมากขึ้นอย่างรวดเร็วในระดับยกกำลัง (Exponential) ของเวลา

2.1.3.4 ไม่สามารถทำนายล่วงหน้าในระยะยาวได้ (Long-term prediction is impossible)

เป็นผลสืบเนื่องจากความไวต่อค่าเงื่อนไขเริ่มต้น เพราะการที่ระบบไวต่อค่าเงื่อนไขเริ่มต้นนั้นจะทำให้เราไม่สามารถรู้ว่าระบบที่เราสนใจอยู่นั้นจะเป็นอย่างไรในระยะยาว แต่อย่างไรก็ตามคุณสมบัติข้อนี้ไม่ได้แปลว่าการทำนายระยะสั้น (Short-term prediction) ของระบบแบบเคออติกจะเป็นสิ่งที่เป็นไปไม่ได้

นอกจากนี้ ระบบเคออสยังมีอีกหนึ่งคุณสมบัติคือ การแสดงลักษณะคล้ายกับตัวเอง (Self-similarity) หรือที่เรียกว่า แฟรกตัล (Fractal) ดังตัวอย่างในรูป 2.2โดยลักษณะนี้จะปรากฏขึ้นเมื่อเราพล็อตเส้นทางการเคลื่อนที่ของระบบในพิกัดที่บ่งบอกถึงสภาวะ อย่างไรก็ตามแฟรกตัลนี้ไม่ได้เป็นเงื่อนไขที่จำเป็นในการเกิดเคออสแต่อย่างใด เพียงแต่มักพบร่วมกันบ่อยครั้งเท่านั้น



รูปที่ 2.2 กิ่งไม้ที่มีความเป็นแฟรกตัล (Fractal)

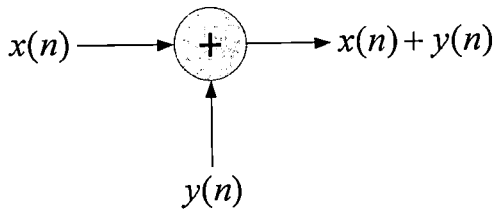
2.2 วงจรกรองดิจิตอล (Digital filter)

วงจรกรองดิจิตอลคือ กระบวนการเชิงเลข (Numerical procedure) ซึ่งเปลี่ยนลำดับของจำนวนๆ หนึ่งเข้าไปสู่อีกลำดับหนึ่งที่มีคุณสมบัติตามที่ต้องการ เช่น การลดสัญญาณรบกวน เป็นต้น โดยวงจรกรองดิจิตอลจะทำการเปลี่ยนลำดับสัญญาณอินพุต x(n) เป็นลำดับสัญญาณเอาต์พุต y(n) โดย n แสดงถึงดัชนี (Index) ของลำดับสัญญาณ ซึ่งโดยปกติจะเป็นเลขจำนวนเต็ม และลำดับสัญญาณเอาต์พุตของวงจรกรองดิจิตอลที่ต้องการจะขึ้นอยู่กับการประยุกต์ใช้งาน ตัวอย่างเช่น ในระบบเรดาห์วงจรกรองดิจิตอลจะถูกใช้ในการปรับปรุงการตรวจจับอากาศยาน เป็นต้น และได้มีการประยุกต์วงจรกรองดิจิตอลในการใช้งานด้านการเข้ารหัสเพื่อรักษาความลับของข้อมูล โดยทำให้วงจรกรองดิจิตอลเกิดการล้นแบบไม่เป็นเชิงเส้น

2.2.1 ส่วนประกอบของวงจรกรองดิจิตอล

วงจรกรองดิจิตอลประกอบด้วยการเชื่อมต่อของของอุปกรณ์พื้นฐาน 3 แบบด้วยกัน คือ ตัวบวก (Adder) ตัวคูณ (Multiplier) และตัวหน่วงเวลา (Unit delay) ซึ่งแสดงในรูปที่2.3โดยตัวบวกและตัวคูณเป็นอุปกรณ์พื้นฐาน ซึ่งมีในหน่วยคำนวณและตรรกะของคอมพิวเตอร์ ส่วนตัวหน่วงเวลาเป็นอุปกรณ์สำหรับการเข้าถึงค่าในอดีตของลำดับข้อมูล

โดยปกติตัวหน่วงเวลาจะถูกสร้างด้วยรีจีสเตอร์หน่วยความจำ สามารถเก็บค่าในปัจจุบันของลำดับข้อมูลในช่วงเวลาหนึ่ง ตัวหน่วงเวลาจะถูกแสดงด้วยกล่องสี่เหลี่ยมที่มีเครื่องหมาย z^{-1} ส่วนตัวล่วงหน้าเวลา (Unit advance) จะถูกใช้เมื่อต้องการค่าในอนาคตของลำดับข้อมูลโดยจะถูกแสดงด้วยกล่องสี่เหลี่ยมที่มีเครื่องหมาย zซึ่งจะถูกนำไปใช้กับงานบางอย่าง เช่น การประมวลผลสัญญาณภาพ (Image processing) ซึ่งข้อมูลทั้งหมดที่จะทำการประมวลผลมีพร้อมอยู่แล้วในขั้นตอนเริ่มต้นของการประมวลผล เพราะฉะนั้นตัวล่วงหน้าเวลาจะไม่สามารถใช้ได้กับงานบางอย่างเช่น งานที่ลำดับข้อมูลได้มาจากการสุ่มตัวอย่างของฟังก์ชันทางเวลา ซึ่งแต่ละตัวอย่างจะถูกประมวลผลทันที (Real-time processing) เมื่อรับข้อมูลเข้ามา ดังนั้นตัวล่วงหน้าเวลาจะใช้ไม่ได้เนื่องจากไม่สามารถรับค่าของข้อมูลในอนาคตเข้ามาได้



ตัวบวก (Adder)

ตัวคูณ (Multiplier)

ตัวหน่วงเวลา (Delay)

รูปที่ 2.3 ส่วนประกอบพื้นฐานของวงจรกรองดิจิตอล

2.2.2 ระบบตัวเลขในการประมวลผลของวงจรดิจิตอล

ในระบบคอมพิวเตอร์หรือโครงสร้างฮาร์ดแวร์ของระบบดิจิตอลอื่นๆเช่นวงจรกรองดิจิตอล ตัวเลขจะถูกแสดงด้วยการรวมกันของเลขฐานสอง (binary digit or bit) ที่มีจำนวนจำกัดซึ่งจะมีค่า 1 และ 0 โดยบิตเหล่านี้ปกติจะถูกจัดให้อยู่ในรูปแบบของไบต์ (byte) ซึ่งประกอบด้วย 8 บิต โดยในระบบดิจิตอลจะมีรูปแบบในการแสดงตัวเลขอยู่ 2 รูปแบบด้วยกัน รูปแบบแรกคือ ระบบจำนวนโดยตรง ซึ่งเป็นระบบตัวเลขที่มีจำนวนจุดทศนิยมคงที่และรูปแบบที่สองคือ ระบบจำนวนอิงดรรชนี ซึ่งเป็นระบบตัวเลขที่มีจำนวนตำแหน่งจุดทศนิยมลอยตัว ซึ่งในวงจรเข้ารหัสเดออติกแบบดิจิตอลจะใช้การคำนวณในรูปแบบของระบบจำนวนโดยตรง

2.2.2.1 ระบบจำนวนโดยตรง

ในระบบจำนวนโดยตรง จุดทวินิยมของเลขฐานสอง (binary point) ที่แบ่งระหว่างจำนวนเต็ม (integer) และจำนวนทศนิยม (fraction) จะถูกจำกัดให้คงที่อยู่โดยบิตแรกจะเรียกว่า(sign bit) ใช้ในการแสดงเครื่องหมายของตัวเลขโดยถ้าเป็นเครื่องหมายบวกจะแทนด้วย0

และถ้าเป็นเครื่องหมายลบจะแทนด้วย 1 ซึ่งขนาดของตัวเลขจะแสดงในรูปแบบการยกกำลังของเลข 2โดยหน้าจุดทศนิยมจะมีกำลังเป็นบวกรวมกำลัง 0 ด้วย และหลังจุดทศนิยมจะมีกำลังเป็นลบ ดังตัวอย่างการหาค่าจำนวนเต็มของเลขฐานสอง01.101_2ดังสมการที่ (2.1)

$$01.101_2 = \left(0\times 2^{1}\right)+\left(1\times 2^{0}\right)+\left(1\times 2^{-1}\right)+\left(0\times 2^{-2}\right)+\left(1\times 2^{-3}\right)=1.625_{10} \quad (2.1)$$

ความเที่ยงตรงของระบบตัวเลขจะถูกกำหนดโดยบิตที่อยู่ทางขวาสุดหรือบิตนัยสำคัญต่ำสุด (least significant bit) ส่วนขอบเขต (range) ของระบบตัวเลขจะนิยามโดยช่วงระหว่างจำนวนเลขลบมากที่สุดที่สามารถแสดงได้ ซึ่งจุดทศนิยมจะเป็นตัวกำหนดความเที่ยงตรง และ ขอบเขตของระบบตัวเลข เมื่อกำหนดจุดทศนิยมให้อยู่ทางขวาสุดรูปแบบบิตจะมีเพียงเลขจำนวนเต็มและไม่มีเลขทศนิยม

2.2.2.2 ส่วนเติมเต็มสอง

ส่วนเติมเต็มสองเป็นรูปแบบหนึ่งในการแสดงตัวเลขของระบบจำนวนโดยตรงเนื่องจากมีประสิทธิภาพในการคำนวณ โดยรูปแบบบิตของจำนวนเลขลบ จะได้มาจากขนาดของจำนวนเลขบวก หลังจากนั้นบิตแต่ละตัวจะถูกสลับค่า และบิต “1” จะถูกบวกเข้าไปกับบิตนัยสำคัญต่ำสุด

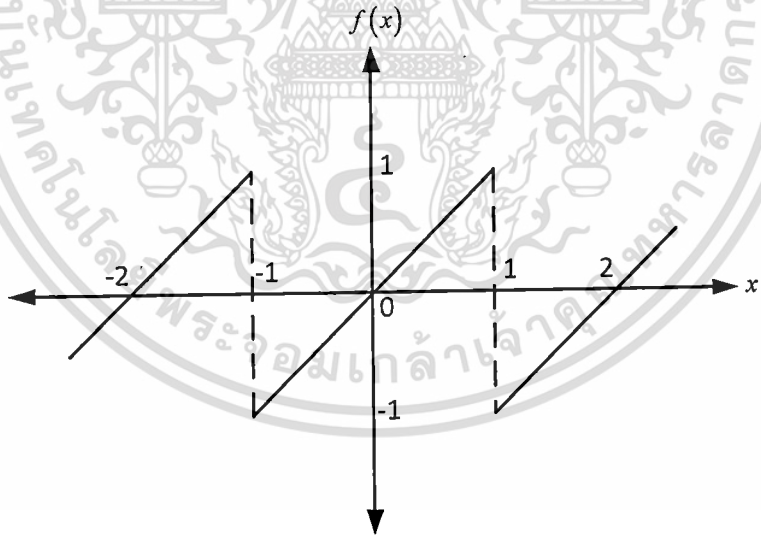
2.2.3 ความคลาดเคลื่อนจากการใช้ระบบจำนวนโดยตรง

เมื่อนำระบบจำนวนโดยตรงมาใช้แทนค่าสัญญาณหรือนำมาใช้ในการประมวลผลของวงจรกรองดิจิตอลจะมีความคลาดเคลื่อนเกิดขึ้น โดยจะยกตัวอย่างการแปลงเลขฐานสิบไปเป็นเลขฐานสองของจำนวนลบมาอธิบาย ซึ่งเลขจำนวนลบที่จะทำการแปลงได้แก่ -6.86_{10} เมื่อทำการแปลงขนาดของเลขฐานสิบไปเป็นเลขฐานสองได้ $6.86_{10} = 0110.1110_2$ จากนั้นทำการสลับค่าบิตต่อบิตได้1001.0001_2แล้วทำการบวกค่า0.0001_2เข้าไปจะได้ว่า $-6.86_{10} = 1001.0010_2$ (เท่ากับ -6.875_{10}) ซึ่งจะเห็นว่าค่าที่ได้ออกมานั้นมีความคลาดเคลื่อนไปจากค่าเดิม โดยความคลาดเคลื่อนที่เกิดขึ้นอาจเกิดจากความผิดพลาดจากการจัดระดับสัญญาณ(quantization error) หรือความคลาดเคลื่อนที่เกิดจากการล้น (overflow) แต่ในที่นี้จะให้ความสัมพันธ์เฉพาะความคลาดเคลื่อนจากการล้นเท่านั้น

2.2.3.1 ความคลาดเคลื่อนจากการล้น

การล้น หมายถึงการที่ผลลัพธ์ที่ได้จากการประมวลผล มีค่ามากเกินกว่าขอบเขตที่สามารถแทนค่าในระบบจำนวนโดยตรงได้ โดยการล้นที่จะกล่าวนี้เป็นการล้นที่เกิดจากการบวกโดยจะพิจารณาการล้นที่เกิดกับผลลัพธ์ของการบวกเลขในระบบส่วนเติมเต็มสอง ซึ่งตัวอย่างของการเกิดการล้นมีดังนี้ ให้ตัวตั้งเป็น 0100 (เท่ากับ 4) และตัวบวก 0100 เช่นกัน ซึ่งเมื่อบวกกันควรจะได้ผลลัพธ์เป็น 8 จึงจะถูกต้องแต่ผลลัพธ์ที่ได้ไม่สามารถแทนค่าได้ในขอบเขตระหว่าง -8 ถึง 7 ดังนั้นการล้นจึงเกิดขึ้น ซึ่งผลลัพธ์ที่ได้จากการเกิดการล้น คือ 1000 (เท่ากับ -8)

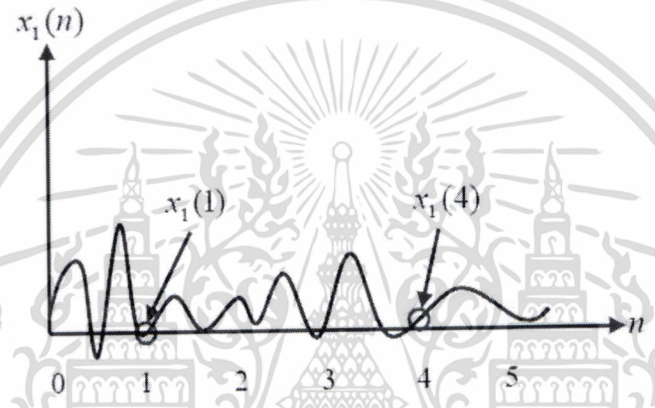
จากการเกิดการล้นแบบส่วนเติมเต็มสอง สามารถนำมาพล็อตเป็นกราฟได้ดังรูปที่ 2.4 ซึ่งเป็นกราฟแสดงค่าที่เกิดจากการบวกเลขฐานสอง 8 บิต โดยแกนนอนแสดงค่าที่ถูกต้องที่เกิดจากการบวก ส่วนแกนตั้งแสดงค่าที่เกิดจากการล้น ซึ่งทั้งสองแกนได้แปลงเป็นเลขฐานสิบจากกราฟจะเห็นว่าการล้นแบบส่วนเติมเต็มสองจะทำให้ผลลัพธ์ที่ได้เปลี่ยนไปมาก ดังนั้นเมื่อเกิดการล้นวงจรกรองดิจิตอลจะได้ผลลัพธ์ที่ผิด



รูปที่ 2.4 การล้นแบบส่วนเติมเต็มสอง

2.3 อัตสหสัมพันธ์ (Autocorrelation)

อัตสหสัมพันธ์ (Autocorrelation) เป็นสิ่งบอกถึงความเหมือนกันหรือความคล้ายคลึงกันระหว่างสัญญาณสุ่มตัวแปรเดียวกัน ตัวอย่างเช่นการหาค่าสหสัมพันธ์ของ $x_1(1)$ และ $x_1(4)$ ดังรูปที่ 2.5

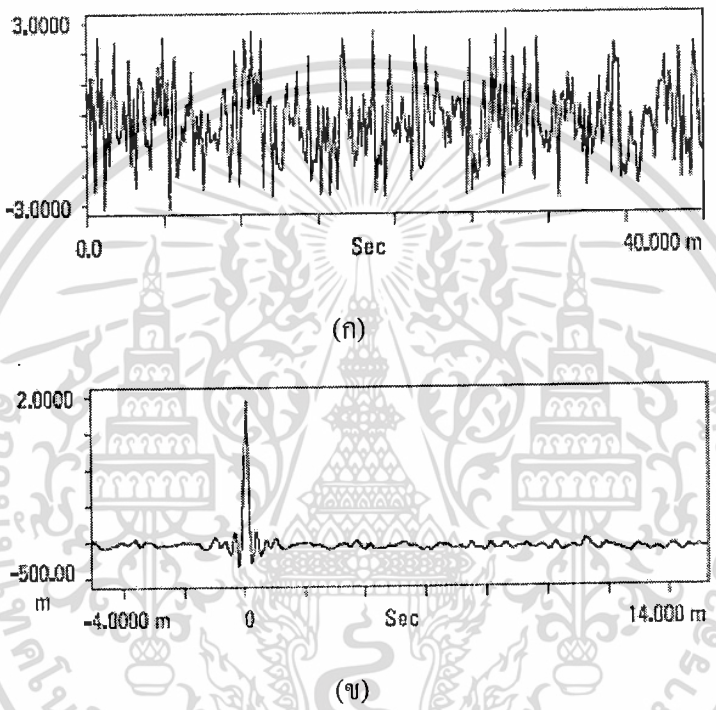


รูปที่2.5 ค่า $x_1(n)$ ที่เวลา n ต่างกัน

จากรูปที่ 2.5 การหาค่าสหสัมพันธ์ $r_{xx}(n,m)$ นี้เป็นการหาความสัมพันธ์ระหว่างสัญญาณ $x_1(1)$และ $x_1(4)$ แต่ทั้งคู่เป็น x_1 เหมือนกัน ดังนั้นเราจึงเรียก $r_{xx}(n,m)$ ว่าเป็นอัตสหสัมพันธ์ (Autocorrelation) ของ x_1 ซึ่งสามารถหาค่าอัตสหสัมพันธ์ได้จากสมการที่ (2.2)

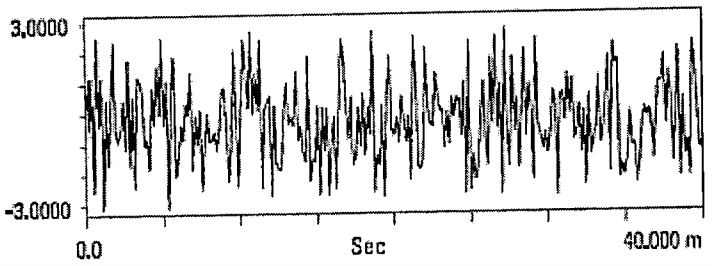
$$r_{xx}(n,m) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} x_i(n)\, x_i(m) \tag{2.2}$$

ยกตัวอย่างดังรูปที่ 2.6(ก) เป็นสัญญาณของ Random noise แบบหนึ่ง ซึ่งมีลักษณะไม่เหมือนกันเลยเมื่อเปรียบเทียบสัญญาณในช่วงเวลาต่างๆ กันหรือ time shift ต่างๆ กัน ดังนั้นลักษณะกราฟ Autocorrelation ในรูปที่ 2.6 (ข) ที่ได้จึงมีลักษณะมียอดสูงสุดในลักษณะ spike ในช่วงเวลา time shift เป็นศูนย์

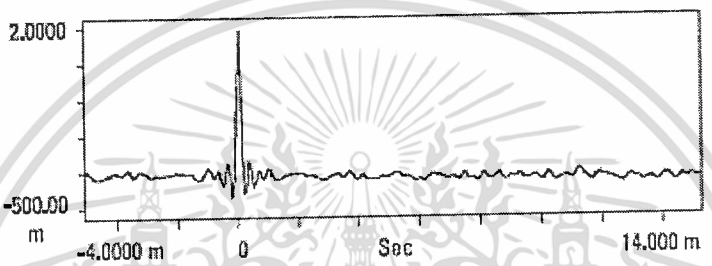


รูปที่ 2.6(ก) สัญญาณรบกวนที่มีลักษณะไม่เป็นคาบ (Random noise)
(ข) กราฟ Autocorrelation ของสัญญาณ

และเมื่อพิจารณากรณีสัญญาณเทียมอย่าง Pseudo random noise ซึ่งมีลักษณะมีการซ้ำกันของคาบเวลาเป็นช่วงๆ และเมื่อ time shift เท่ากับคาบเวลาที่ซ้ำกันนั้น เส้นกราฟของ Autocorrelation จะปรากฏpeak ซ้ำเช่นเดียวกัน ดังรูปที่ 2.7



(ก)



(ข)

รูปที่ 2.7(ก) สัญญาณรบกวนที่มีลักษณะเป็นคาบ (Pseudo random noise)

(ข) กราฟ Autocorrelation ของสัญญาณ

จากกรณีตัวอย่างทั้งสองทำให้เราทราบว่ากราฟ Autocorrelation จะมีลักษณะมีคาบเวลาเกิดซ้ำในทำนองเดียวกันและมีคาบเวลาเท่ากันกับการเกิดคาบเวลาซ้ำของสัญญาณที่พิจารณาและเมื่อพิจารณากรณีสัญญาณเทียมอย่าง Pseudo random noise ซึ่งมีลักษณะมีการซ้ำกันของคาบเวลาเป็นช่วงๆ

2.4 ค่าระยะห่างยูคลีดีน (Euclidean distance) และค่า PSNR (Peak Signal to Noise Ratio)

2.4.1 ค่าระยะห่างยูคลีดีน (Euclidean distance)

ใช้ในการเปรียบเทียบความแตกต่างระหว่างสัญญาณดั้งเดิมกับสัญญาณที่ผ่านการถอดรหัสโดยจะใช้สมการที่ (2.3)

$$D = \sqrt{\sum_{k=1}^{n}\left\{\left(x(k)-z(k)\right)^2\right\}} = \sqrt{((x(1)-z(1))^2+(x(2)-z(2))^2+...(x(n)-z(n))^2)} \tag{2.3}$$

โดย $x(k)$ คือสัญญาณดั้งเดิม และ $z(k)$ คือสัญญาณที่ผ่านการถอดรหัสแล้ว

2.4.2 ค่า PSNR (Peak Signal to Noise Ratio)

Peak Signal to Noise Ratio (PSNR) เป็นค่ามาตรฐานที่บ่งบอกถึงคุณภาพที่เปลี่ยนระหว่างสองรูปภาพ ซึ่งใช้ในการเปรียบเทียบที่สภาวะต่างๆ กันดังสมการที่ (2.4)

$$\mathrm{PSNR} = 20\log_{10}\left(\frac{b}{\mathrm{RMSE}}\right) \tag{2.4}$$

เมื่อตัวแปร b คือค่าสูงสุดที่เป็นไปได้ของพิกเซลในภาพ ในที่นี้ก็คือค่าสูงสุดที่ข้อมูลขนาด 8บิตสามารถแสดงได้ นั่นคือ 255

ค่า Root Mean Square Error (RMSE) และค่า Mean Square Error (MSE)สามารถคำนวณได้จากสมการที่ (2.5) และ (2.6) ตามลำดับ

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}} \tag{2.5}$$

$$\mathrm{MSE} = \frac{1}{N}\times\left(SUM_{ij}\left|Org_{ij}-Wmk_{ij}\right|^2\right) \tag{2.6}$$

เมื่อ Org_{ij} คือค่าพิกเซลของภาพต้นฉบับ Wmk_{ij} คือค่าพิกเซลของภาพที่ถูกถอดรหัสแล้ว N คือจำนวนพิกเซลทั้งหมดภายในภาพที่ได้มาจากผลคูณระหว่างขนาดพิกเซลทางกว้างและขนาดพิกเซลทางยาว

2.5 อุปกรณ์ FPGA และภาษา VHDL

2.5.1 ประเภทของอุปกรณ์ FPGA

ในปัจจุบันมี FPGA อยู่ 4 ชนิดที่วางขายอยู่ในท้องตลาดได้แก่ Symmetrical Array, Row-Based, Hierarchical PLD และ Sea-of-Gates ซึ่งแต่ละชนิดก็มีลักษณะการเชื่อมต่อภายในและการโปรแกรม ที่แตกต่างกันไป นอกจากนี้ในการแบ่งประเภทของ FPGA อาจแบ่งได้ตามเทคโนโลยีที่ใช้ในการโปรแกรม ซึ่งมีอยู่ 2 แบบคือ การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพของตัวชิพ และการโปรแกรมโดยการใช้หน่วยความจำ

2.5.1.1การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพ

1) Fuse เป็นวิธีการโปรแกรมที่สามารถทำได้เพียงครั้งเดียว ซึ่งหลังจากที่โปรแกรมแล้วจุดเชื่อมต่อจะขาดจากกัน

2) Anti-Fuse เป็นวิธีการโปรแกรมที่คล้ายกับแบบ Fuse แต่ต่างกันที่หลังจากทำการโปรแกรมแล้วจุดเชื่อมต่อจะเชื่อมถึงกัน

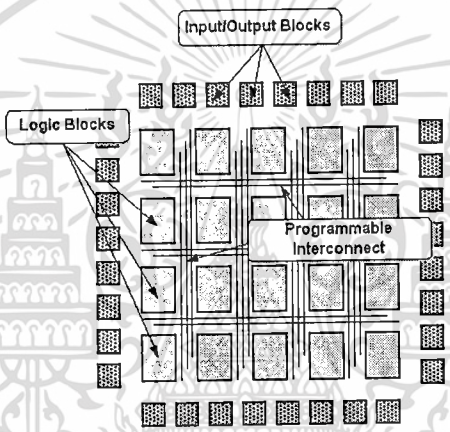
2.5.1.2การโปรแกรมโดยใช้หน่วยความจำ

1) EEPROM Based FPGA: FPGA ที่ใช้การโปรแกรมแบบนี้มักเรียกว่า CPLD ซึ่งเทคโนโลยีที่ใช้จะเหมือนกับ EEPROM ทำให้มีความจุของเกทต่ำ โดยทั่วไปจะน้อยกว่า 20,000เกท แต่ข้อดีของ EEPROM Based FPGA คือสามารถเก็บข้อมูลที่โปรแกรมลงไปได้โดยไม่จำเป็นต้องมีไฟเลี้ยง และในการโปรแกรมจะใช้ทรานซิสเตอร์ 1ตัวต่อ 1บิต ซึ่งการโปรแกรมสามารถทำได้ประมาณ 10,000ครั้ง

2) SRAM Based FPGA: FPGA แบบนี้จะใช้เทคโนโลยีในการโปรแกรมเหมือนกับ SRAM (Static RAM) ทำให้สามารถโปรแกรมซ้ำได้โดยไม่จำกัดจำนวนครั้ง นอกจากนี้ยังมีความจุของเกทในระดับปานกลางถึงสูงมาก (ประมาณ 10,000 - 1,000,000เกท) ซึ่งข้อดีของ SRAM Based FPGA

คือใช้เวลาในการโปรแกรมน้อย (ระดับ nsec) การโปรแกรมทำได้ง่ายเทียบได้กับการเขียน SRAM ทั่วไปและเหมาะสำหรับการออกแบบวงจรที่มีความสลับซับซ้อน ส่วนข้อเสียคือไม่สามารถเก็บโปรแกรมในภาวะที่ไม่มีไฟเลี้ยงได้ดังนั้น FPGA ชนิดนี้จึงมักใช้ควบคู่กับ ROM เพื่อเก็บโปรแกรมและทำการโหลดโปรแกรมลงในตัวชิพในขณะที่เริ่มต้นใช้งาน

2.5.2 โครงสร้างภายในของ FPGA



รูปที่ 2.8 สถาปัตยกรรมของ FPGA โดยทั่วไป

จากรูปที่ 2.8 Logic Blocks เป็นส่วนประกอบหลักที่ใช้งานสำหรับการออกแบบวงจรดิจิตอลด้วย FPGA ซึ่งโครงสร้างภายในของ Logic Blocks นั้นก็จะมีลักษณะที่แตกต่างจากของ PLD ประการแรกคือแทนที่จะทำการสร้าง SOP (Sum-of-Products) expression ด้วย AND gatesplane แล้วตามด้วย OR gates plane โดยปกติแล้ว FPGA จะใช้หลักการบนพื้นฐานของ LUT (Lookup Table) ประการถัดไปคือ ในFPGA มีจำนวนของ flip-flops สำหรับการใช้งานที่มากกว่าใน CPLD อย่างมาก นอกจากนั้นใน FPGA เองยังมีการเพิ่มเติม features ต่างๆสำหรับการใช้งานเพิ่มเติมเข้าไว้อย่างมากด้วย เช่น SRAM Memory, Clock Multiplication, PCI Interface และ FPGA บางตัวยังได้ใส่ blocks สำหรับการใช้งานเฉพาะ (Dedecated Blocks) เข้าไปด้วย เช่น Multipliers, DSPs (Digital Signal Processors) และMicroprocessors (เช่น Power PC และ

MicroBlazeสำหรับ Xilinx FPGA หรือ NIOS II สำหรับ Altera FPGA) ทำให้การใช้งาน FPGA มีความสะดวกและหลากหลายมากขึ้นจนบางครั้งการเชื่อมต่อของ FPGA กับ Microprocessor ภายนอก สำหรับการใช้งานบางอย่างไม่จำเป็น เนื่องจากเราสามารถใช้ Microprocessor ที่อยู่ภายใน FPGA ได้เลย และยังสามารถออกแบบวงจรอื่นๆ ที่จำเป็นต้องใช้งานร่วมกันในระบบที่ต้องการออกแบบด้วยภาษา HDL และสังเคราะห์เป็นวงจรที่ใช้งานร่วมกันอยู่ภายใน FPGA เช่นกัน ซึ่งเป็นที่มาของคำว่า SoC (System on Chip) หรือ SoPC (System on Programmable Chip) คือระบบทั้งระบบอยู่ใน Chip FPGA เพียงตัวเดียว ข้อแตกต่างพื้นฐานอีกอย่างหนึ่งของ FPGA กับ CPLD ก็คือ CPLD จะมีลักษณะเป็น Non-volatile คือไฟดับแล้วข้อมูลที่เป็น Configuration ของวงจรไม่หายในขณะที่ FPGA ส่วนใหญ่จะเป็นแบบไฟดับแล้วข้อมูลจะหายเนื่องจากเทคโนโลยี FPGA ใช้จะเป็น SRAM-based ดังนั้นในการนำ FPGA มาใช้งานจริงเพื่อป้องกันการไฟดับแล้วข้อมูลเสียหาย จึงทำการเก็บข้อมูล Configuration ไว้ในหน่วยความจำภายนอก เช่น SPROM (Serial PROM)เป็นต้น

2.5.3 การออกแบบโดยใช้ภาษาอธิบายพฤติกรรมของฮาร์ดแวร์

ในการออกแบบวงจรดิจิตอลนั้นสามารถทำได้โดยการวาดวงจร (Schematic) หรือใช้ภาษาอธิบายพฤติกรรม (Hardware Description Language) ของฮาร์ดแวร์จากที่ได้กล่าวไปแล้วในบทที่ 1 ในกรณีของการออกแบบวงจรด้วย ASIC ชนิด Full Custom ผู้ออกแบบจะต้องเขียนวงจรด้วย Schematic จากนั้นจะนำวงจรที่ ออกแบบไว้ไปทำการจำลองการทำงาน (Simulate) ซึ่งหากผลออกมาเป็นที่พอใจก็จะต้อง Layout เป็นชั้นสาร และในการออกแบบ ASIC ชนิดนี้ผู้ออกแบบจำเป็นจะต้องทราบถึงเทคโนโลยีที่ใช้ในการสร้างด้วย หลังจากได้ Layout ที่สมบูรณ์แล้วจึงจะส่งไปเข้ากระบวนการสร้างไอซีหรือ Fabrication เพื่อสร้างเป็นชิพไอซีออกมา แต่ในการออกแบบวงจรด้วย FPGA โดยการใช้ Schematic หรือใช้ภาษาอธิบายการทำงานของวงจรจะทำ ได้สะดวกกว่า เนื่องจากวิธีการนี้ผู้ออกแบบไม่จำเป็นต้องคำนึงถึงเทคโนโลยีที่จะใช้สร้างไอซี และที่สำคัญ การออก แบบโดยวิธีนี้สามารถแก้ไขโมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่า เพราะไม่ต้องวาดวงจร ใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยีสำหรับภาษาที่ใช้ สำหรับอธิบายพฤติกรรมของฮาร์ดแวร์ที่ใช้กันก็มี VHDL, AHDL และ Verilog เป็นต้น ส่วนรายละเอียดของขั้นตอนในการออกแบบสามารถอธิบายได้ดังนี้

2.5.3.1 การสังเคราะห์วงจร (Logic Synthesis)

ในขั้นตอนนี้จะใช้ซอฟต์แวร์ในการสังเคราะห์วงจร (Synthesis Tools) ทำการสังเคราะห์พฤติกรรม ของวงจรที่ได้จากการออกแบบด้วย Schematic หรือ VHDL ซึ่งต้องทำการตรวจสอบด้วยว่าซอฟต์แวร์ นั้นสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการหรือไม่ ตัวอย่างเช่น FPGA ของบริษัท XILINX และบริษัท ALTERA จะมีซอฟต์แวร์หลายตัวที่สามารถใช้ได้ เช่น Max Plus II ในขั้นตอนนี้ซอฟต์แวร์สังเคราะห์วงจรจะทำการแปลงโค้ด VHDL และทำการ Optimize เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ในการสังเคราะห์วงจรนั้น วงจรระดับเกท (Gate Level) จะไม่เหมาะสมกับโครงสร้างที่มีอยู่ในอุปกรณ์ FPGA ดังนั้นในการ Optimize ซอฟต์แวร์สังเคราะห์วงจรจะต้องทำการ Optimize ให้ได้เป็นวงจรที่ประกอบ ด้วยกลุ่มของลอจิกที่เหมาะสมกับอุปกรณ์ FPGA นั้นๆ จึงทำให้ผลที่ได้มีประสิทธิภาพและในขั้นตอนการสังเคราะห์วงจรนี้ ผู้ออกแบบสามารถกำหนดข้อบังคับสำหรับโมเดลแต่ละตัวได้ เช่น ข้อบังคับในเรื่องเวลา (Timing Constraints) หรือข้อบังคับในเรื่องของพื้นที่ (Area) หรือกำหนดชนิดและตำแหน่งของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอน Optimize เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนดส่วนสำคัญในการ Optimize คือการเทียบ (Mapping) โมเดลให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้างและสถาปัตยกรรมภายในอุปกรณ์ FPGA เมื่อทำการสังเคราะห์วงจรเสร็จแล้วซอฟต์แวร์การสังเคราะห์วงจรก็จะมีการรายงานผลว่าโมเดลที่ออกแบบไปนั้นเป็นอย่างไร เช่นมีค่าความหน่วง (Delay) เท่าใดใช้ทรัพยากรต่างๆ ใน FPGA อะไรบ้างเมื่อมาถึงขั้นตอนนี้ผู้ออกแบบก็จะทราบว่าโมเดลเป็นไปตามข้อบังคับหรือไม่ ถ้าไม่ก็สังเคราะห์ใหม่จนกว่าจะเป็นไปตามที่กำหนด

2.5.3.2 การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นการแบ่งวงจรที่ได้จากการสังเคราะห์เป็นส่วนย่อยๆสำหรับลงใน CLBs, IOBs หรือองค์ประกอบอื่นๆ ภายในอุปกรณ์ FPGA สำหรับเกณฑ์ที่ใช้ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกันมีจำนวนสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้ เพื่อลดความหนาแน่นในตอนทำการเชื่อมต่อสัญญาณ (routing) ในขั้นตอนนี้จะใช้ซอฟต์แวร์ทำโดยซอฟต์แวร์จะเทียบส่วนประกอบของวงจรเช่น เกท (gate), ฟลิป-ฟลอบ (flip-flop) ลงในทรัพยากรต่างๆ ที่มีอยู่ในอุปกรณ์ FPGAหลังจากทำขั้นตอนนี้เสร็จแล้วผู้ออกแบบสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ไปเท่าไรส่วนข้อมูลทางเวลานั้นผู้ออกแบบจะทราบเฉพาะความหน่วงภายในแต่ละส่วนเท่านั้น หรือที่เรียกว่าความหน่วงลอจิก

(logic delay) ส่วนซอฟต์แวร์จะรวมเอาซอฟต์แวร์ย่อยอื่นๆอีกเพื่อให้การทำ PPR (Partitioning Placement & Routing) เป็นไปอย่างต่อเนื่อง

2.5.3.3 การวางอุปกรณ์ (Placement)

ขั้นตอนนี้เป็นการเลือกทำเลที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (Partitioning) มาแล้วว่าควรจะอยู่ ณ ตำแหน่งไหนในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ดีที่สุด เช่นวงจรส่วนไหนควรอยู่ใกล้กัน เพื่อจะได้ค้นหาเส้นทางได้ (route) ง่ายหรือช่วยลดความหน่วง จะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือ Router ทำการค้นหาเส้นทางสัญญาณได้ไม่หมด การวางอุปกรณ์ที่ดีควรวางส่วนต่างๆ ให้อยู่ใกล้กันโดยเฉพาะส่วนที่มีการเชื่อมต่อสัญญาณด้วยกันนอกจากนั้นการกำหนดตำแหน่งขา I/O (I/O pin) ตามตำแหน่งขา I/O ของ FPGA บนแผ่น PCB ก็จะมีผลโดยตรงเลยคือซอฟต์แวร์จะวาง I/O ลงในตำแหน่งที่ผู้ออกแบบกำหนด ซึ่งบางครั้งตำแหน่งที่กำหนดไปไม่เหมาะสม ดังนั้นการกำหนดขา I/O ควรกำหนดตำแหน่งให้เหมาะสมหรือไม่ก็ให้ซอฟต์แวร์จัดการเอง

2.5.3.4 การเชื่อมต่อสัญญาณ (Routing)

ในขั้นตอนนี้เป็นการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่างๆภายในอุปกรณ์ FPGA ขั้นตอนนี้จะทำต่อเนื่องจากการวางอุปกรณ์ ในกรณีที่ทำการวางอุปกรณ์ไว้ไม่ดีซอฟต์แวร์ก็จะทำการเชื่อมต่อสัญญาณได้ไม่หมด (เนื่องจากจำนวนทรัพยากรสำหรับเชื่อมต่อสัญญาณนั้นมีอยู่จำกัด) หรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับผู้ออกแบบสามารถทำขั้นตอนนี้ได้โดยใช้ซอฟต์แวร์หรือผู้ออกแบบจะทำการเชื่อมต่อสัญญาณด้วยตนเองก็ได้ แต่ทางที่ดีควรใช้ซอฟต์แวร์ทำดีกว่านอกจากนั้นการกำหนดข้อบังคับทางเวลาจะช่วยให้ผลที่ได้จากการเชื่อมต่อสัญญาณดีขึ้นได้

2.5.3.5 ความหน่วงด้านเวลา (Delay)

ในการทำ FPGA นั้นความหน่วงที่เกิดขึ้นเป็นความหน่วงที่เกิดจากการวางตำแหน่ง (layout) ของอุปกรณ์ ซึ่งผู้ออกแบบไม่สามารถเข้าไปแก้ไขได้ แต่สามารถทำให้มีความหน่วงน้อยที่สุดได้ สำหรับความหน่วงที่เกิดขึ้นนั้นแยกได้เป็นสองประเภทคือ

1) ความหน่วงลอจิก (Logic delay)

เป็นความหน่วงภายในองค์ประกอบของอุปกรณ์ FPGA เอง

2) ความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณ (Routing delay)

เป็นความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณระหว่างองค์ประกอบภายในอุปกรณ์ FPGA โดยปกติแล้ว ค่าความหน่วงลอจิกไม่ควรเกิน 50% ของค่าความหน่วงที่ยอมรับได้ เพราะความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณมักจะมีค่ามากกว่าค่าความหน่วงลอจิก ดังนั้นในการวางอุปกรณ์ และเชื่อมต่อสัญญาณ ผู้ออกแบบควรกำหนดข้อบังคับทางเวลาเพื่อให้ซอฟต์แวร์ได้ทำงานอย่างมีประสิทธิภาพเพิ่มขึ้นและเพื่อให้ได้ผลลัพธ์ที่ดีขึ้นค่าความหน่วงที่ได้หลังจากการวางอุปกรณ์ และเชื่อมต่อสัญญาณแล้วจะมีค่าความหน่วงที่ค่อนข้างแน่นอน ซึ่งผู้ออกแบบสามารถทราบได้ว่าโมเดลที่ออกแบบนั้นเป็นไปตามข้อกำหนดหรือไม่

2.5.3.6 การจำลองการทำงานของวงจร (Simulation)

ในขั้นตอนนี้เป็นขั้นตอนที่สำคัญอีกขั้นตอนหนึ่ง เพราะเป็นขั้นตอนที่ผู้ออกแบบตรวจสอบฟังก์ชันการทำงานของโมเดลว่าถูกต้องหรือไม่มีข้อผิดพลาดตรงไหนเพื่อจะได้ทำการแก้ไขให้ถูกต้อง ในขั้นตอนนี้จะมีซอฟต์แวร์ที่ใช้สำหรับทำการจำลองการทำงานของวงจรที่ใช้อยู่ เช่น Model Simของบริษัท Model Technology หรือ Max Plus II ของบริษัท Altera ในการจำลองการทำงานของวงจร ควรทำทุกครั้งหลังจากที่มีการทำแต่ละขั้นตอนหลักเสร็จแล้ว เพื่อจะได้ทราบว่าข้อผิดพลาดของโมเดลเกิดขึ้นตอนไหนจะได้แก้ไขข้อผิดพลาดตรงขั้นตอนนั้นๆ ได้เลยไม่ต้องมาคอยตรวจหาขั้นตอนที่ทำให้เกิดข้อผิดพลาดนั่นคือการทำการจำลองการทำงานของวงจร ต้องทำทั้งหลังการเขียนโค้ด, การสังเคราะห์วงจร และการทำ PPR การจำลองการทำงานของวงจรหลังจากที่เขียนโค้ดเสร็จแล้วนั้น ผู้ออกแบบสามารถทราบได้แค่โมเดลทำงานถูกต้องหรือไม่เท่านั้น (functional test) ยังไม่สามารถตรวจสอบการทำงานในเชิงเวลาได้ถูกต้อง ในการจำลองการทำงานของวงจรหลังจากที่สังเคราะห์เป็นวงจรแล้ว เพื่อตรวจสอบว่าฟังก์ชันการทำงานยังคงถูกต้องหรือไม่ และค่าความหน่วงที่เกิดขึ้นเป็นไปตามข้อบังคับหรือไม่มีข้อผิดพลาดเกิดขึ้นหรือไม่ถ้ามีจะแก้ไขให้ถูกต้อง

ในการจำลองการทำงานของวงจรหลังจากที่ทำการวางอุปกรณ์ การเชื่อมต่อสัญญาณ (Post layout simulation) แล้วก็มีความสำคัญเช่นกันเพราะผลที่ได้จากการจำลองการทำงานของวงจรในตอนนี้ จะเป็นผลลัพธ์ของโมเดลเลย ซึ่งผู้ออกแบบนอกจากจะตรวจสอบฟังก์ชันการทำงานแล้วยังต้องตรวจสอบคุณสมบัติอื่นๆ เช่น ความหน่วงที่ได้จากการทำ PPR ในรูปแบบค่าความหน่วงมาตรฐาน (Standard Delay Format: SDF) ว่าตรงตามที่กำหนดหรือไม่ หรือ

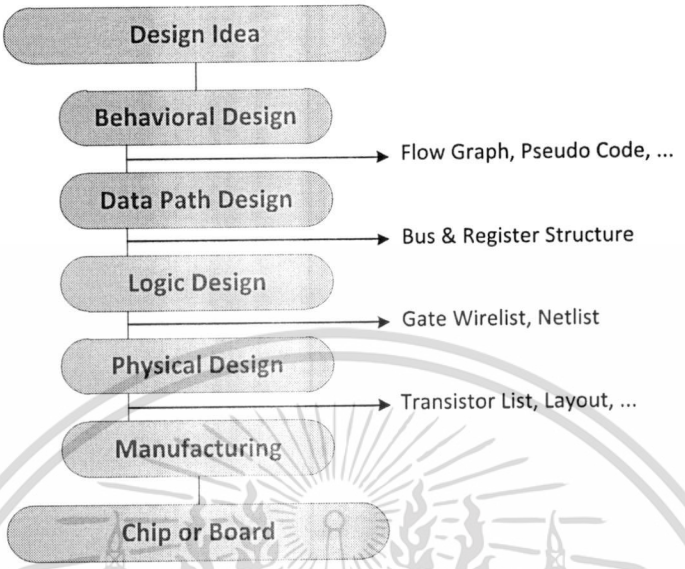
ตรวจสอบว่าวงจรรวม สามารถใช้งานที่ความถี่สูงสุดเท่าไรนั่นเอง ในการจำลองการทำงานของวงจรควรใช้ซอฟต์แวร์ตัวเดียวกันตลอดเพื่อจะได้เปรียบเทียบผลที่ได้จากขั้นตอนต่างๆ

2.5.3.7 การโปรแกรมอุปกรณ์ FPGA (Configuration)

หลังจากที่โมเดลผ่านขั้นตอนต่างๆจนกระทั่งผ่านการทำ PPR (Partitioning, Placement & Routing) แล้วนั้น ถึงขั้นตอนนี้ก็สามารถที่จะดาวน์โหลด (download) ลงในอุปกรณ์ FPGA ได้แล้ว ในการดาวน์โหลดนี้ก่อนอื่นต้องแปลงแบบวงจรรวมที่ได้เป็นข้อมูลวงจร (configuration data) ซึ่งอยู่ในรูปของบิตสตรีม (bit stream) ก่อนแล้วจึงดาวน์โหลดลงไปเพื่อให้อุปกรณ์ FPGA มี ฟังก์ชันการทำงานตามโมเดลที่ผู้ออกแบบต้องการ ซึ่งในขั้นตอนนี้จะใช้วิธีที่แตกต่างกันออกไปสำหรับ อุปกรณ์ FPGA ของแต่ละบริษัทผู้ผลิตคือในกรณีที่เป็นอุปกรณ์ FPGA ชนิดที่ต้องโปรแกรมโดยวิธี SRAM นั้น ในการใช้งานผู้ออกแบบจะต้องเก็บข้อมูลวงจรไว้ในหน่วยความจำประเภท EPROM หรือ serial PROM ด้วยเพื่อจะใช้งานสะดวกขึ้น คือในการใช้งานโมเดลครั้งต่อไปไม่ต้องดาวน์โหลดข้อมูลวงจรจากเครื่องคอมพิวเตอร์อีก เพราะมีข้อมูลวงจรเก็บอยู่ในหน่วยความจำอยู่แล้วแต่กรณีที่อุปกรณ์ FPGA เป็นชนิดที่โปรแกรมโดยใช้วิธี EPROM หรือ Anti fuseก็ไม่จำเป็นต้องมีหน่วยความจำสำหรับเก็บข้อมูลวงจรเพราะว่าอุปกรณ์ FPGA ชนิดนี้เมื่อดาวน์โหลดข้อมูลวงจรลงไปข้อมูลที่ดาวน์โหลดลงไปก็ยังคงอยู่ในอุปกรณ์ FPGA และครั้งต่อไปก็ใช้งานโมเดลที่ออกแบบไว้ได้เลย

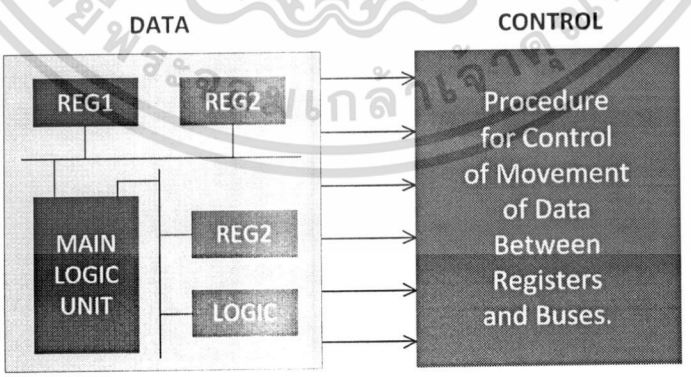
2.5.4 การออกแบบระบบดิจิตอล

ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวความคิดเบื้องต้นจนกระทั่งได้ออกมาเป็นอุปกรณ์ฮาร์ดแวร์ที่ใช้งานได้จะต้องผ่านขั้นตอนต่างๆ มากมาย และในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์ในแต่ละขั้น ก่อนเข้าสู่กระบวนการออกแบบในขั้นต่อไป รูปที่ 2.9 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิตอลทั่วไป ขั้นแรกผู้ออกแบบจะกำหนดแนวความคิดในการออกแบบแล้วทำการพัฒนาให้สามารถนำมาใช้ได้อย่างสมบรูณ์ ซึ่งภาย ในขั้นตอนนี้ผู้ออกแบบจำเป็นต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบซึ่งอาจจะเป็นผังงานแสดงแบบหรือ รหัสคำสั่งเทียม (Pseudo code) ก็ได้



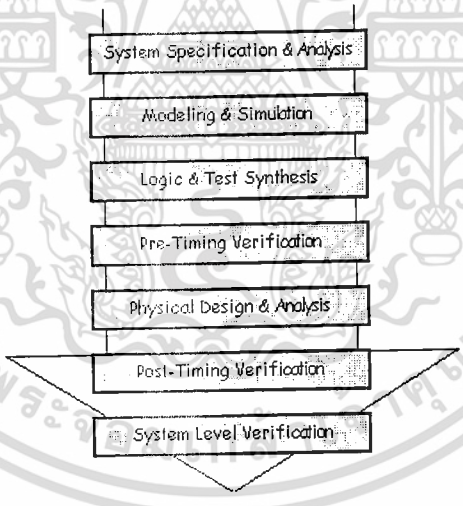
รูปที่ 2.9 ขั้นตอนการออกแบบระบบดิจิตอล

ขั้นตอนต่อไปเป็นการออกแบบระบบเส้นทางของข้อมูล ผู้ออกแบบจะกำหนดส่วนประกอบของรีจิสเตอร์และวงจรลอจิก ที่จำเป็นทั้งหมดเพื่อนำมาประกอบเป็นระบบที่สมบูรณ์ โดยแต่ละองค์ประกอบสามารถเชื่อมต่อกันด้วยบัสหนึ่งหรือสองทิศทาง (Unidirectional or Bidirectional Bus) ส่วนกระบวนการในการควบคุมการเคลื่อนย้ายข้อมูลระหว่างรีจิสเตอร์และวงจรลอจิกจะขึ้นอยู่กับพฤติกรรมของระบบที่กำหนดไว้ดังรูปที่ 2.10



รูปที่ 2.10 การออกแบบระบบเส้นทางของข้อมูล

Top-Down Design เป็นการออกแบบวงจรและระบบดิจิตอลสมัยใหม่ (Modern Digital System Design) ที่อาศัยการออกแบบด้วยภาษาอธิบายการทำงานของฮาร์ดแวร์ (HDL, Hardware Description Language) เป็นหลักในแต่ละขั้นตอนของการออกแบบจะมีการตรวจสอบและจำลองการทำงานเป็นขั้นๆ ไป ดังนั้นเมื่อเกิดข้อผิดพลาดหรือปัญหาใดๆ ขึ้น ก็จะสามารถทำการแก้ไขได้ทันที มิต้องรอจนถึงขั้นสุดท้ายดังในกรณีของ Bottom-Up Design ถึงจะรู้ได้ว่าสิ่งที่ออกแบบมามีความผิดพลาดอยู่หรือไม่นั่นหมายความว่าด้วยวิธีการออกแบบแบบ Top-Down Design ผู้ออกแบบสามารถมั่นใจได้ว่าผลสุดท้ายที่ได้จากการออกแบบ (ซึ่งอยู่ในระดับล่างสุด) สามารถทำงานได้อย่างถูกต้องแน่นอนทำให้โดยรวมแล้วการออกแบบด้วยวิธีการนี้ช่วยประหยัดเวลาได้อย่างมากเมื่อเทียบกับการออกแบบแบบ Bottom-Up Design และสอดคล้องกับการออกแบบวงจรระบบดิจิตอลด้วยภาษาอธิบายการทำงานของฮาร์ดแวร์ดังเช่นที่ใช้งานกันอยู่ในการออกแบบระบบดิจิตอลในปัจจุบันขั้นตอนการออกแบบแบบ Top-Down Design สามารถแสดงได้ดังรูปที่ 2.11



รูปที่ 2.11 ขั้นตอนการออกแบบจากบนลงล่าง

โดยจากรูปที่ 2.11 สามารถอธิบายได้ดังนี้ขั้นตอนการออกแบบจะเริ่มจากการกำหนดขอบเขตการทำงานของระบบที่ต้องการออกแบบและทำการวิเคราะห์การทำงานของระบบ (System Specification and Analysis) จากนั้นจะทำการสร้างแบบจำลอง (Modeling) ซึ่งใน

กรณีนี้จะหมายถึงการออกแบบตัววงจรหรือระบบด้วยภาษา HDL จากนั้นก็จะจำลองการทำงาน (Simulation) เพื่อดูว่าตัววงจรหรือระบบที่เราออกแบบหรือโมเดลมันขึ้นมานั้นมีฟังก์ชั่นการทำงานถูกต้องอย่างที่เรากำหนดไว้ในตอนแรกหรือไม่ถ้าไม่ก็จะทำการแก้ไขให้ถูกต้อง ณ เวลานี้เลย แต่ถ้าผลการจำลองการทำงานถูกต้องอยู่แล้ว ก็จะผ่านไปยังขั้นตอนต่อไปคือการสังเคราะห์วงจรและทดสอบวงจรที่ได้จากการสังเคราะห์ (Logic and Test Synthesis, Pre-Timing Verification)ในขั้นตอนนี้จะเป็นการนำตัวภาษา HDL ที่เราได้เขียนขึ้นและผ่านการทดสอบฟังก์ชั่นการทำงานแล้วมาทำการสังเคราะห์ให้กลายเป็นวงจรจริงๆ

ในขั้นตอนนี้จะต้องมีการใช้ Software ที่เรียกกันว่า Synthesis Tools เข้ามาช่วยและจะต้องมีการอ้างตัว Target Technology ให้ Synthesis Tools รู้เพื่อใช้ในการอ้างอิง Library ของอุปกรณ์ที่จะนำมาสังเคราะห์เป็นวงจรจริงๆ จากนั้นทำการทวนสอบการทำงานอีกครั้ง (Pre-Timing Verification) เพื่อยืนยันว่าวงจรที่ได้จากการสังเคราะห์ยังคงทำงานได้ถูกต้องเหมือนเดิม การทวนสอบหรือจำลองการทำงานอีกครั้งในขั้นตอนนี้บางครั้งเรียกกันว่าการทำ Gate Level Simulation คือจะคิดค่า delay time ที่เกิดขึ้นใน gate ต่างๆที่สังเคราะห์ให้กลายเป็นวงจรระบบขึ้นมาด้วยแต่ยังไม่คิดค่า delay time ที่เกิดในสาย routing สัญญาณต่างๆ

เมื่อผ่านขั้นตอนนี้ก็จะไปยังขั้นตอน Physical Design and Analysis โดยขั้นตอนนี้จะเป็นการนำวงจรที่ได้จากการสังเคราะห์ในขั้นตอนที่ผ่านมาทำการจัดโครงสร้างของวงจรให้มีความเหมาะสมกับลักษณะทาง physical จริงของตัว target devices จริงซึ่งอาจหมายถึงอุปกรณ์จำพวก PLD (Programmable Logic Device),CPLD(Complex PLD) หรือFPGA (Field Programmable Gate Array) หรืออาจจะหมายถึง VLSI chip จริงที่จะทำการ fabricate กับทางโรงงาน (Foundry) ก็ได้การจัดโครงสร้างดังกล่าวเพื่อให้เหมาะกับ physical ของตัว devices จริงจะประกอบไปด้วยขั้นตอนย่อยๆ 3 ขั้นตอนที่เรียกว่า PPR (Partioning, Place and Routing) คือทำการแบ่งวงจรที่ได้จากการสังเคราะห์ให้เป็นส่วนๆ ที่เหมาะกับ physical จริงๆภายใน target devices จากนั้นก็จะทำการเลือกตำแหน่งที่เหมาะสมสำหรับวางชิ้นส่วนของวงจรเหล่านั้นลงไปแล้วทำการเชื่อมต่อส่วนต่างๆ ของวงจรเข้าด้วยกันด้วยสายเชื่อมต่อสัญญาณอีกครั้งหนึ่งจากนั้นก็จะทำการทวนสอบอีกครั้งหนึ่งที่เรียกว่า Post-timing Verification หรือบางครั้งเราจะเรียกกันว่า Post Layout Simulation ซึ่งจะเป็นการจำลองในขั้นสุดท้ายที่คิดทั้งผลของ delay time ที่เกิดขึ้นใน gate ต่างๆ และ delay time ที่เกิดขึ้นในสาย routing สัญญาณต่างๆ ที่เกิดขึ้นจริงภายใน devices และขั้นตอนสุดท้ายคือ System Level Verification ซึ่งก็คือการทดสอบจริงกับตัววงจรระบบที่ได้

จาการออกแบบโดยอาจจะทำการทดสอบกับระบบแวดล้อมภายนอกอื่นๆที่จำเป็นต้องใช้งานร่วมกันด้วยก็ได้

2.5.5 ภาษา VHDL

ความซับซ้อนและขนาดของระบบดิจิตอลในปัจจุบันได้เพิ่มมากขึ้นทุกขณะส่งผลให้มีการนำคอมพิวเตอร์เพื่อช่วยในการออกแบบหรือ CAD มาใช้ในขบวนการออกแบบฮาร์ดแวร์เพิ่มขึ้นเช่นกัน อีกทั้งอุปกรณ์และวิธีการออกแบบใหม่ๆ ก็ถูกพัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้กับนักออกแบบมากขึ้นด้วย สำหรับภาษาบรรยายอุปกรณ์ฮาร์ดแวร์ (HDL : Hardware Description Language) ก็เป็นเครื่องมืออย่างหนึ่งที่ได้รับการพัฒนามาอย่างต่อเนื่อง เพื่อช่วยให้การปรับปรุงขบวนการออกแบบระบบดิจิตอลเป็นไปอย่างมีประสิทธิภาพ

VHDL ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC: Very High Speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (High Level Language) ที่ใช้สำหรับการออกแบบฮาร์ดแวร์ในระบบดิจิตอล ตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (Hierarchy) และ สามารถเขียนได้หลายรูปแบบด้วยเหตุผลนี้จึงทำให้ภาษา VHDL เป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุดคือ แนวความคิดที่จะแก้ปัญหาลงไปทีละขั้นจนถึงขั้นตอนของการสร้างวงจรจริง และตัวภาษาก็เปิดโอกาสให้วิศวกรได้พัฒนาและจำลองการทำงานของรูปแบบฟังก์ชันการทำงานของวงจรอย่างสังเขป โดยยังไม่ต้องคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้น VHDL ยังเป็นภาษาที่สนับสนุนลักษณะต่างๆ ของระบบดิจิตอลที่มีความซับซ้อนได้ทั้งหมด ดังนั้นภาษาVHDL จึงเป็นภาษาที่น่าสนใจในการศึกษาและนำไปใช้งานเป็นอย่างยิ่ง วิวัฒนาการของภาษา VHDL เริ่มต้นประมาณปี ค.ศ. 1981 เมื่อกระทรวงกลาโหมสหรัฐอเมริกา หรือ DoD (Department of Defense) ได้พยายามปรับปรุงอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการ ทางทหาร ให้มีความทันสมัยมากขึ้นประกอบกับเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์มีการพัฒนาไปอย่างรวดเร็วดังจะเห็นได้จากการนำวงจรดิจิตอลหลายๆ วงจรมาทำการผลิตอยู่บนแผ่นซิลิกอนที่มีพื้นที่เพียง 1 - 2 ตารางเซนติเมตรเท่านั้น ซึ่งเป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้นตลอดจนความน่าเชื่อถือ ในการทำงานและความคงทนต่อสภาพ แวดล้อมสูง แต่เนื่องจากในขณะนั้นขั้นตอนของการออกแบบ การผลิต และการตรวจสอบวงจรต้นแบบ เป็นขบวนการที่ต้องใช้วิศวกร และเวลาในดำเนินการมาก ฉะนั้นทาง DoDจึงจัดตั้งโครงการขึ้นมาเพื่อศึกษาวิธีการที่ช่วยในการ

พัฒนา วงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอล ให้สามารถนำไปผลิตได้เร็วขึ้น ซึ่งโครงการดังกล่าวมีชื่อว่า "Very High Speed Integrated Circuits" หรือ VHSIC โดยในระยะแรกนั้นโครงการนี้ถือเป็นความลับทาง ด้านความมั่งคงของประเทศ และอยู่ภายใต้ความควบคุมดูแลของ United States International Traffic and Arms Regulations (ITAR) สำหรับมาตรฐานของภาษาที่ใช้บรรยายพฤติกรรมวงจรหรือฮาร์ดแวร์ของระบบ สำหรับโครงการ VHSIC ที่ DoDได้ให้ไว้สามารถสรุปได้ดังนี้

1) ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถเข้าใจได้ทั้งมนุษย์และเครื่องคอมพิวเตอร์โดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก
2) สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้
3) ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

ฉะนั้นภาษาดังกล่าวนี้จึงจัดเป็นภาษาโปรแกรมระดับสูง เช่นเดียวกับภาษาปาสคาล หรือภาษาซี ซึ่งในทางวิศวกรรม ภาษาที่ใช้ในการออกแบบฮาร์ดแวร์นี้เรียกว่า "Hardware Description Language" หรือ HDL ในตอนเริ่มแรกนั้น DoDได้มอบหมายให้บริษัทไอบีเอ็ม เท็กซัสอินสตูเมนท์ และอินเตอร์เมทริกซ์ เป็นผู้ศึกษาและพัฒนาโครงการ ซึ่งการดำเนินงานเป็นไปอย่างต่อเนื่อง จนกระทั่งในปี ค.ศ.1985 ทาง ITAR ได้ยกเลิกข้อจำกัดในการถ่ายทอด เทคโนโลยีทางทหารออกจากโครงการนี้ ดังนั้นภาษา VHDL จึงเริ่มเป็นที่รู้จักกันโดยทั่วไป และประมาณปี ค.ศ. 1987IEEE ได้ทำการกำหนดมาตรฐานของภาษานี้เป็น IEEE 1076-1987 และมีชื่อเรียกว่า VHDL ซึ่งมาตรฐานนี้ได้รับ การปรับปรุงจนเป็นมาตรฐาน IEEE 1076-1993 หรือ VHDL 1993 เนื่องจากในขณะนั้น DoD เป็นลูกค้ารายใหญ่ ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์ ดังนั้นจึงมีผู้รับโครงการต่างๆ จาก DoD ไปดำเนินการวิจัยและพัฒนา เป็นจำนวนมาก และเพื่อให้ทุกโครงการอยู่ในมาตรฐานเดียวกันหมด ดังนั้นทาง DoD จึงได้กำหนดว่าทุกๆโครงการต้องเขียนอยู่ในรูปของภาษา VHDLเท่านั้น ซึ่งทำให้ DoD สามารถนำโครงการเหล่านี้ไปจำลองกับเครื่องคอมพิวเตอร์ได้หลายๆระบบ

2.5.5.1 องค์ประกอบพื้นฐานของภาษา VHDL

รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบของ VHDL จะประกอบไปด้วยส่วนกำหนดการเชื่อมต่อ (interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม (architecture)โดยในการบรรยายการเชื่อมต่อจะขึ้นต้นด้วยคำว่า ENTITY แล้วตามด้วยชื่อขององค์ประกอบจากนั้นตามด้วยคำว่า IS และถัดมาจะเป็นการบรรยายถึงพอร์ตการติดต่อ อินพุต - เอาต์พุตขององค์ประกอบ ในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้บรรยายหน้าที่การทำงานขององค์ประกอบ โดยหน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณอินพุต–เอาต์พุตและพารามิเตอร์อื่นๆที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูปที่ 2.12 และสำหรับการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังจากคำว่า BEGIN เป็นต้นไป

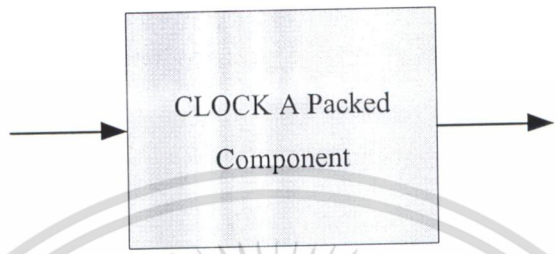
```
ENTITY component_name IS
  Input and output ports
  Physical and other parameters
END [component_name];

ARCHITECTURE identifier OF component_name IS
  [declaration]
BEGIN
  Specification of the functionality of component
in terms of its input lines and as influenced
by physical and other parameters
END [identifier];
```

รูปที่ 2.12 การกำหนดการเชื่อมต่อและสถาปัตยกรรมของภาษา VHDL

1) การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ โดยในระดับนี้ต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบภายนอกอื่นๆ ดังตัวอย่างในรูปที่ 2.13 ซึ่งเป็นบล็อกไดอะแกรมและการบรรยายการเชื่อมต่อขององค์ประกอบสำหรับตัวจ่ายสัญญาณนาฬิกาในบรรทัดแรกของการบรรยายการเชื่อมต่อเป็นการกำหนดชื่อขององค์ประกอบซึ่งกำหนดเป็นclock_ component ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ภายในวงเล็บส่วน IN

และ OUTเป็นการกำหนดโหนดของสัญญาณให้เป็นอินพุตหรือเอาต์พุตและ BIT เป็นการแสดงชนิดของข้อมูล




```
ENTITY component_name IS
   PORT (en : IN BIT; ck : OUT BIT);
END [clock_name];
```

รูปที่ 2.13 บล็อกไดอะแกรมและการบรรยายการเชื่อมต่อของ clock_ component

2) การกำหนดรูปแบบการบรรยาย

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ ซึ่งในการบรรยายสามารถกำหนดค่าของสัญญาณเอาต์พุตในเทอมของอินพุต หรือในรูปขององค์ประกอบอื่นๆ หรือทั้งสองอย่างรวมกันก็ได้ ดังตัวอย่างการบรรยายของ clock_component ในรูปที่ 2.14 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมโดยมี en เป็นอินพุตและ ck เป็นเอาต์พุตPROCESS เป็นคำที่ใช้ในการเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม และภายในโปรเซสกำหนดให้ periodic เป็นตัวแปรที่มีค่าเริ่มต้นเป็น "0" ถ้าสัญญาณ en มีค่าเป็น "1"' จะทำให้ตัวแปร periodic ถูกคอมพลีเมนท์ (complement) และส่งค่าให้กับ ck ซึ่งเป็นสัญญาณเอาต์พุต และสำหรับคำสั่ง WAIT จะเป็นการกำหนดให้สัญญาณมีคาบเวลาเท่ากับ 1us

```
ARCHITECTURE behavioral OF clock_component IS

BEGIN
  PROCESS
      VARIABLE periodic : BIT = '0';PORT(en:IN BIT;ck:OUT BIT);
  BEGIN
      IF en = '1' THEN
      periodic := Not periodic ;
      END IF ;
      ck <= periodic ;
      WAIT FOR 1 US;
  END PROCESS ;
END behaviorai ;
```

รูปที่ 2.14 การบรรยายเชิงพฤติกรรมของclock _ component

3) หน่วยการออกแบบแพ็กเกจ

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อยที่เป็นประโยชน์ต่อการเขียนรูปแบบการบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนของแพ็กเกจซึ่งหน่วยการออกแบบต่างๆเช่น หน่วยการออกแบบ entity หน่วยการออกแบบสถาปัตยกรรมหรือ หน่วยการออกแบบแพ็กเกจอื่นๆสามารถเรียกข้อมูลเหล่านี้ไปใช้ได้นอกจากนั้นสิ่งที่นิยมทำกันมากคือการนำรูปแบบมาตรฐานต่างๆเช่นอุปกรณ์มาตรฐาน (ไอซีตระกูล 74XX เป็นต้น) มาเก็บไว้ในรูปของแพ็กเกจที่ทุกคนสามารถเข้าถึงได้ ตามปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (package declaration)และส่วนของบอดี้แพ็กเกจ (package body) เนื่องจากแพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน

package declaration ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจ สำหรับนำไปใช้ภายนอกตัวของแพ็กเกจเอง ถ้ามีการประกาศสิ่งใดๆ ในส่วนของบอดี้แพ็กเกจ แต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่าและพฤติกรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ entity คือ จุดเชื่อมต่อหรือ พอร์ตที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศชนิด (type) หรือสัญญาณ เช่นเดียวกับส่วน

บอดี้แพ็กเกจที่ไม่จำเป็นต้องมีส่วนของการประกาศแพ็กเกจ แต่แพ็กเกจนั้นไม่สามารถนำไปใช้จากรูปแบบอื่นได้โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้วในส่วนของการประกาศแพ็กเกจจะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึงการกำหนดค่าคงที่ต่างๆอันได้แก่ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อนในส่วนของการประกาศแพ็กเกจ และถูกกำหนดค่าในส่วนของบอดี้แพ็กเกจฉะนั้นในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการประกาศชื่อที่เป็นโปรแกรมย่อย หรือค่าคงที่ การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในรูปที่ 2.15

```
PACKAGE BODY package_name IS
    declarative part
END package_name ;
```

รูปที่ 2.15 โครงสร้างของบอดี้แพ็กเกจ

หน่วยการออกแบบ configuration ดังที่ทราบกันแล้วว่าระบบดิจิตอลรูปแบบหนึ่งไม่ว่าจะเป็นอะไรก็ตามจะสามารถมีหน่วยการออกแบบ entity ได้ เพียงหนึ่งเดียวเท่านั้น ซึ่งในหน่วยการออกแบบ entity หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมีหน่วยการออกแบบ configuration มาเพื่อกำหนดการใช้ configuration ของการประกอบ entity กับหน่วยการออกแบบสถาปัตยกรรมหน่วยใดๆ เข้าด้วยกัน

โปรแกรมย่อย การใช้ฟังก์ชันและโพรซีเจอร์ใน VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไป ค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อย อาจจะมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่นถ้าใช้ฟังก์ชันแทนการกระทำในสมการบูลีน ก็จะมีผลต่อวงจรลอจิกจริงๆ ในขณะที่ถ้าใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูล หรือในการคำนวณค่าการหน่วงเวลาแล้ว ก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์ รูปที่ 2.16 แสดงการใช้โพรซีเจอร์เพื่อเปลี่ยนข้อมูลชนิด 8 บิตเป็นค่าจำนวนเต็ม และรูปที่ 2.17 แสดงการใช้ฟังก์ชันโดยกำหนดให้ X เป็นตัวแปรชนิดบิต แทนการกระทำในสมการบูลีน

```
TYPE byte IS ARRAY (7 downto 0) OF BIT ;

PROCEDURE byte_to_integer (ib: IN byte : oi : OUT INTEGER) IS
    VARIABLE result : INTEGER := 0;
BEGIN
    FOR I IN 0 TO 7 LOOP
        IF ib(i) = '1' THEN
        Result := result + 2**I ;
        END IF ;
    END LOOP;
    Oi :=result ;
END byte_to_integer
```

รูปที่ 2.16 การใช้โพรซีเจอร์

```
FUNCTION f(a,b,c : BIT) RETURN BIT IS
    VARIABLE x : BIT;
BEGIN
    x := ((NOT a) AND (NOT b) AND c) ;
    RETURN x ;
END f ;
```

รูปที่ 2.17 การใช้ฟังก์ชัน

โอเปอร์เรเตอร์ การบรรยายเชิงพฤติกรรมในภาษา VHDL มีตัวดำเนินการหรือโอเปอร์เรเตอร์ทางลอจิกและคณิตศาสตร์เช่นเดียวกับภาษาซอฟต์แวร์ทั่วไปดังรูปที่ 2.18

PREDEFIND OPERATORS	
LOGICAL OPERATORS	:NOT AND OR NAND NOR XOR
OPERAND TYPE	:BIT BOOLEAN
RESULT TYPE	:BIT BOOLEAN
RELATIONAL OPERATORS	:= /= <<= >>=
OPERAND TYPE	:any type
RESULT TYPE	:Boolean
ARITHMATIC OPERATORS	:+-*//** MOD REM ABS
OPERAND TYPE	:INTEGER REAL physical
RESULT TYPE	:INTEGER REAL physical
CONCANTENATION OPERATOR	:&
OPERAND TYPE	:ARRAY of any type
RESULT TYPE	:ARRAY of any type
RESULT TYPE	:ARRAY of any type

รูปที่ 2.18 ตัวดำเนินการใน VHDL

เวลาและความพร้อมเพรียง ในวงจรอิเล็กทรอนิกส์ อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ (always active) และจะมีเรื่องของเวลาเข้ามาเกี่ยวข้องในทุกๆ เหตุการณ์ที่เกิดขึ้นเสมอ VHDL เป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถบรรยายรูปแบบและการพ้องกันของเวลา สำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้องการบรรยายการทำงานที่อยู่ภายในส่วนของการบรรยายสถาปัตยกรรม จะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็นแบบลำดับคำสั่งก็ตาม ซึ่งหากมีหลายๆ โปรเซสอยู่ภายในโครงสร้างเดียวกัน ทุกๆ โปรเซสก็จะทำงานไปพร้อมๆ กันด้วย

สัญญาณและตัวแปรสัญญาณ มีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ ที่ใช้ในการส่งผ่านข้อมูลและมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยการกำหนดค่าให้กับสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาใน

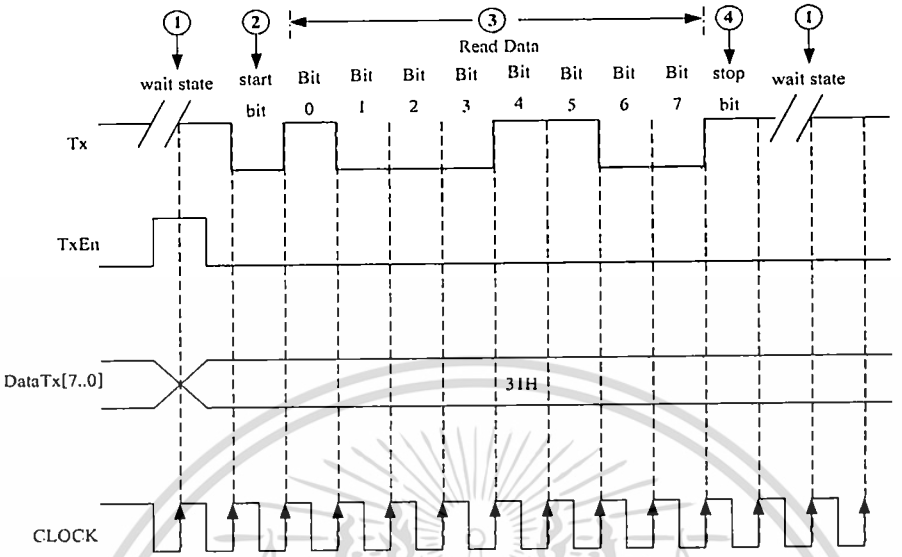
การส่งผ่านค่าของสัญญาณ เช่น w <= a AFTER 12ns หมายถึงการกำหนดค่าสัญญาณ a ให้กับ w หลังจากเวลาผ่านไป 12ns ในทางตรงข้ามตัวแปรมีลักษณะเป็นเสมือนตัวกลาง ที่ใช้ในการส่งผ่านข้อมูล และไม่มีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วย ซึ่งตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งเช่นใน ฟังก์ชันโพรซีเจอร์ และโปรเซส สำหรับการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์ :=

2.6 การสื่อสารข้อมูลแบบอนุกรม

วงจรสื่อสารข้อมูลแบบอนุกรมนี้ จะมีการกำหนดค่าอัตราบอดที่ 9600Hz โดยมีความกว้างของข้อมูลเท่ากับ 8 บิต มีการใช้บิตเริ่มต้นและบิตสิ้นสุด และไม่มีการใช้บิตพาริตี้ ในส่วนรายละเอียดของวงจรภายในจะประกอบด้วย 2 ส่วนคือ ภาคการส่งข้อมูลและภาคการรับข้อมูล

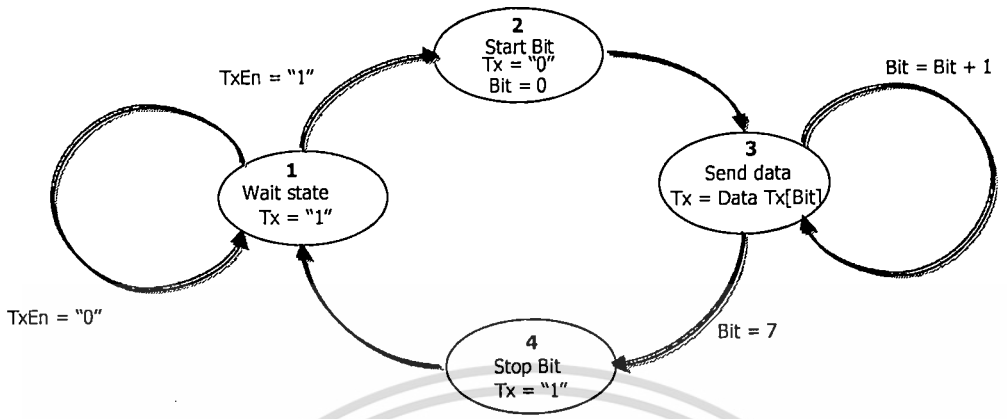
2.6.1 ภาคส่งข้อมูลแบบอนุกรม

เป็นการออกแบบ FPGA ให้ทำหน้าที่เป็นตัวส่งข้อมูลแบบอนุกรมให้แก่อุปกรณ์ต่างๆ เช่น คอมพิวเตอร์, ไมโครคอนโทรลเลอร์ เป็นต้น โดยรูปแบบการส่งข้อมูลแบบอนุกรมแสดงดังรูปที่ 2.19



รูปที่ 2.19 รูปแบบการส่งข้อมูลแบบอนุกรม

จากรูปที่ 2.19 สัญญาณ Tx เป็นสัญญาณขนาด 1บิต ใช้สำหรับส่งข้อมูลแบบอนุกรม ซึ่งในการออกแบบวงจรส่งข้อมูลแบบอนุกรมนั้น จะต้องมีสัญญาณนาฬิกาเป็นสัญญาณอ้างอิงเพื่อใช้กำหนดความเร็วในการส่งข้อมูล ซึ่งค่าความเร็วนี้จะต้องตรงกับทางภาครับด้วย เช่น ถ้าต้องการส่งข้อมูลด้วยความเร็ว 9600Hz ทางด้านตัวส่งจะต้องกำหนดให้สัญญาณนาฬิกาให้มีค่าเท่ากับ 9600Hz ด้วยซึ่งความเร็วในการส่งข้อมูลนี้ก็คือ อัตราบอด (Baud Rate) นั่นเอง ส่วนสัญญาณ TxEn จะเป็นสัญญาณเปิดการส่งข้อมูล โดยถ้า TxEn มีค่าเป็นลอจิก 0แสดงว่าในขณะนั้นยังไม่มีการส่งข้อมูลใดๆ ออกไป และถ้าสัญญาณที่ขา TxEn มีค่าเป็นลอจิก 1จะทำให้ Tx มีค่าเป็นลอจิก 0เพื่อเป็นการบอกทางภาครับว่าจะมีการเริ่มส่งข้อมูลออกไป และหลังจากนั้นจะทำการส่งข้อมูลออกไปทาง Txซึ่งข้อมูลที่ต้องการส่งในรูปแบบอนุกรม (DataTx[7..0]) จะถูกส่งออกไปทาง Tx โดยจะส่งบิตที่มีความสำคัญต่ำสุด (Bit 0) ออกไปก่อน แล้วจึงส่งบิตถัดไปตามออกมาจนครบทั้ง 8บิต จากนั้นสัญญาณ Tx จึงมีสถานะเป็นลอจิก 1เพื่อบอกทางภาครับว่าเป็นการสิ้นสุดการส่งข้อมูล และยังสามารถนำวิธีการส่งข้อมูลแบบอนุกรมมาเขียนเป็น State Diagram ได้ดังรูปที่ 2.20



รูปที่ 2.20 State Diagram ของการส่งข้อมูลแบบอนุกรม

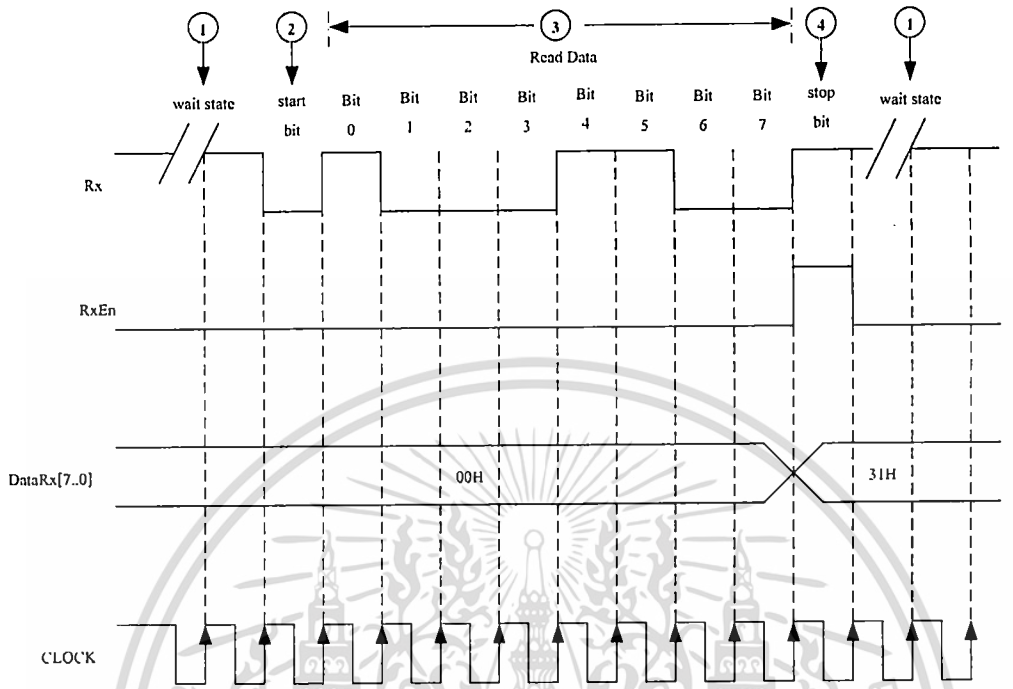
จาก State Diagram เมื่อ TxEn มีค่าเป็นลอจิก 0 จะทำให้ Tx มีค่าเท่ากับลอจิก1 และจะยังอยู่ใน State 1 จนกว่า TxEn จะมีค่าเป็นลอจิก 1 จึงจะเข้าสู่ State 2

ใน State 2 สัญญาณ Tx จะมีค่าเป็นลอจิก 0 เพื่อเป็นการกำหนดบิตเริ่มต้นการส่งข้อมูล หลังจากนั้นจะเข้าสู่ State 3ซึ่งเป็นการส่งข้อมูลออกไปทีละบิต โดยจะเริ่มส่งข้อมูลบิตที่ 0 ออกไปก่อนและวนส่งข้อมูลออกไปจนครบ 8 บิต จึงจะหลุดเข้าสู่ State 4

ใน State 4จะทำการกำหนดให้ Tx มีค่าเป็นลอจิก 1 ซึ่งเป็นการกำหนดบิตสิ้นสุดการส่งข้อมูล และหลังจากนั้นการทำงานจะกลับเข้าสู่ State 1ใหม่อีกครั้ง

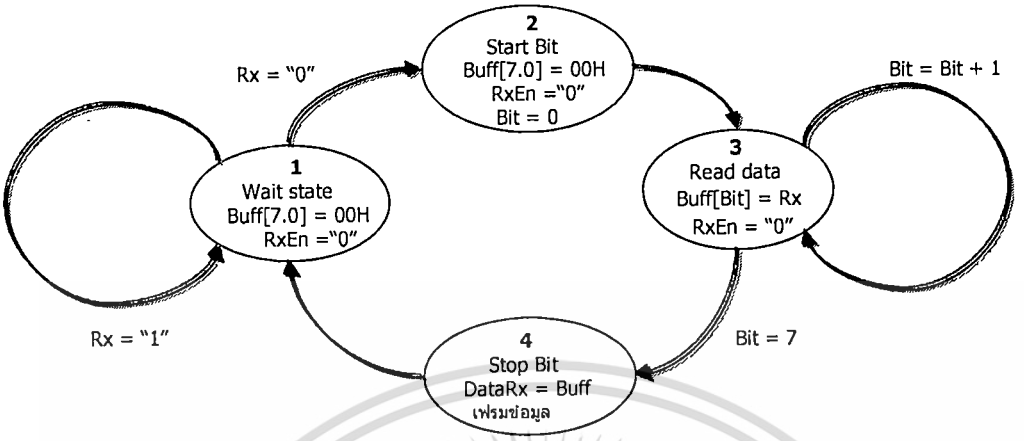
2.3.2 ภาครับข้อมูลแบบอนุกรม

สำหรับภาครับข้อมูลแบบอนุกรมจะมีการทำงานคล้ายกับทางภาคส่ง ซึ่งข้อมูลแบบอนุกรมที่รับเข้ามาได้จะมีลักษณะดังรูปที่ 2.21



รูปที่ 2.21 รูปแบบการรับข้อมูลแบบอนุกรม

จากรูปที่ 2.21 สัญญาณ Rx เป็นสัญญาณขนาด 1 บิตใช้สำหรับรับข้อมูลแบบอนุกรมที่ส่งมาจากตัวส่ง ในการออกแบบภาครับข้อมูลจะต้องมีสัญญาณอ้างอิงเพื่อกำหนดความเร็วในการรับข้อมูลซึ่งสอดคล้องกับทางภาคส่ง เช่น ถ้าทางภาคส่งใช้อัตราเร็วในการส่งข้อมูลเท่ากับ 9600Hz ทางภาครับจะต้องกำหนดอัตราเร็วในการรับข้อมูลให้เป็น 9600Hz ด้วยเช่นกัน ในส่วนของการทำงานเริ่มจากสัญญาณ Rx จะมีค่าเป็นลอจิก 1 ซึ่งหมายถึงยังไม่มีการส่งข้อมูลออกมา และการทำจะรอจนกระทั่ง Rx จะมีค่าเป็นลอจิก 0 นั่นแสดงว่าทางภาคส่งเริ่มส่งข้อมูลมาแล้ว หลังจากนั้นจะทำการอ่านข้อมูลเข้ามาเก็บไว้ที่ละบิตจนครบทั้ง 8 บิต และตรวจสอบสัญญาณ Rx ว่าเป็นลอจิก 1 หรือไม่ หากเป็นลอจิก 1 แสดงว่าสิ้นสุดการส่งข้อมูล จากนั้นจะมีการกำหนดสัญญาณ DataRx[7..0] ให้มีค่าเท่ากับสัญญาณที่รับมาได้ และกำหนดให้สัญญาณ RxEn มีค่าเป็นลอจิก 1 เพื่อเป็นสัญญาณกระตุ้นให้วงจรภายนอกนำข้อมูล DataRx[7..0] ไปใช้งาน จากการทำงานดังกล่าวนี้เราสามารถนำมาเขียนเป็น State Diagram ได้ดังรูปที่ 2.22



รูปที่ 2.22 State Diagram ของการรับข้อมูลแบบอนุกรม

จาก State Diagram ในรูปที่ 2.22 การทำงานขณะเริ่มต้นจะอยู่ใน State 1ถ้าสัญญาณ Rx มีค่าเป็นลอจิก 1 การทำงานจะไม่มีการเปลี่ยน Stateและจะได้ค่า RxEnเป็นลอจิก 0 และ Buff[7..0] ซึ่งเป็น Buffer สำหรับเก็บข้อมูลที่อ่านมาได้จากสัญญาณ Rx มีค่าเป็น "00000000"

การทำงานจะเข้าสู่ State2 เมื่อสัญญาณ Rx มีค่าเป็นลอจิก 0 ซึ่งเมื่อเข้ามาที่ State2 แสดงว่าได้มีการรับบิตเริ่มต้นข้อมูลเข้ามาแล้ว โดยใน State2 นี้จะยังคงกำหนดให้ Buff[7..0] มีค่าเท่ากับ "00000000" RxEnมีค่าเป็นลอจิก 0 และ bit ซึ่งเป็นตัวแปรนับจำนวนบิตมีค่าเท่ากับ 0 และหลังจากนั้นจะเข้าสู่ State 3 โดยอัตโนมัติ

ภายในState 3 จะมีการอ่านข้อมูลจาก Rx เข้ามาเก็บไว้ใน Buffer จนครบ 8 บิตและกำหนดให้สัญญาณ RxEn มีค่าเป็นลอจิก 0 จากนั้นจะเข้าสู่ State 4 ซึ่งจะมีการโอนย้ายข้อมูลจาก Buff มาเก็บไว้ใน DataRx ซึ่งเป็นข้อมูลขนาด8 บิตทั้งหมดที่รับมาได้ และกำหนดให้ RxEn มีค่าเป็นลอจิก 1 เพื่อเป็นสัญญาณกระตุ้นให้วงจรภายนอกอื่นๆ รับทราบว่าได้รับข้อมูลมาครบทั้ง 8 บิตแล้ว หลังจากนั้นการทำงานจะกลับมาที่ State 1 เพื่อรอรับข้อมูลอีกครั้ง

บทที่ 3

การออกแบบและการจัดทำโครงการวิจัย

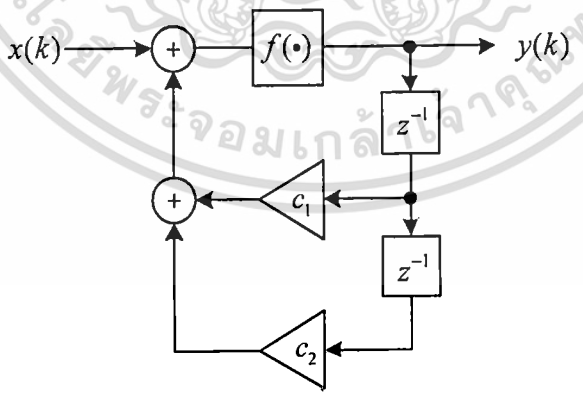
บทที่ 3 นี้เป็นการศึกษาพฤติกรรมเคออสที่เกิดขึ้นในวงจรกรองสัญญาณดิจิตอล ซึ่งเป็นการศึกษาในเชิงของทฤษฎี รวมถึงการจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสโดยใช้โปรแกรม MATLAB จากนั้นจำลองการทำงานและออกแบบวงจรเข้ารหัสและถอดรหัสโดยใช้ภาษา VHDL

3.1 การออกแบบ

3.1.1 การออกแบบการทดลองโดยใช้โปรแกรม MATLAB

3.1.1.1 วงจรเข้ารหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสองชนิดอิมพัลส์ไม่จำกัด (Infinite Impulse Response Second Order Filter: IIR 2^{nd} order filter)

โดยวงจรกรองมีโครงสร้างแบบโดยตรง (Direct form) ดังรูปที่ 3.1



รูปที่ 3.1 โครงสร้างของวงจรเข้ารหัสแบบ IIR 2^{nd} order filter

ซึ่งหากยังไม่พิจารณาถึงฟังก์ชั่น $f(\bullet)$ สามารถเขียนเป็นสมการผลต่างสืบเนื่อง (Difference equation) ในรูปแบบของสมการป้อนกลับ (recursive equation) ดังสมการที่ (3.1)

$$y(k)=x(k)+c_1y(k-1)+c_2y(k-2) \tag{3.1}$$

จากสมการที่ (3.1) สามารถเขียนเป็นฟังก์ชั่นถ่ายโอน (Transfer function) ได้ดังสมการที่ (3.2)

$$Y(z)=X(z)+c_1Y(z)z^{-1}+c_2Y(z)z^{-2}$$

$$Y(z)\left(1-c_1z^{-1}-c_2z^{-2}\right)=X(z)$$

$$H(z)=\frac{Y(z)}{X(z)}=\frac{1}{\left(1-c_1z^{-1}-c_2z^{-2}\right)} \tag{3.2}$$

จากสมการที่ (3.2) จะเห็นได้ว่ามีตำแหน่งของโพล (Pole) อยู่ 2 ตัวซึ่งสามารถหาได้จาก

$$1-c_1z^{-1}-c_2z^{-2}=0$$

$$z^2-c_1z-c_2=0$$

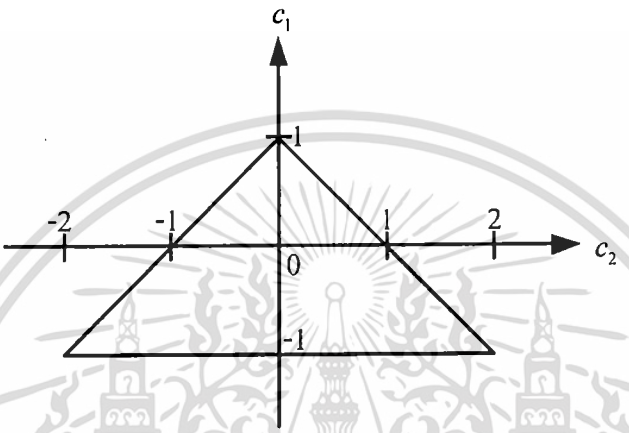
ดังนั้น

$$p_1=\frac{c_1+\sqrt{c_1^{\,2}+4c_2}}{2} \tag{3.3}$$

และ

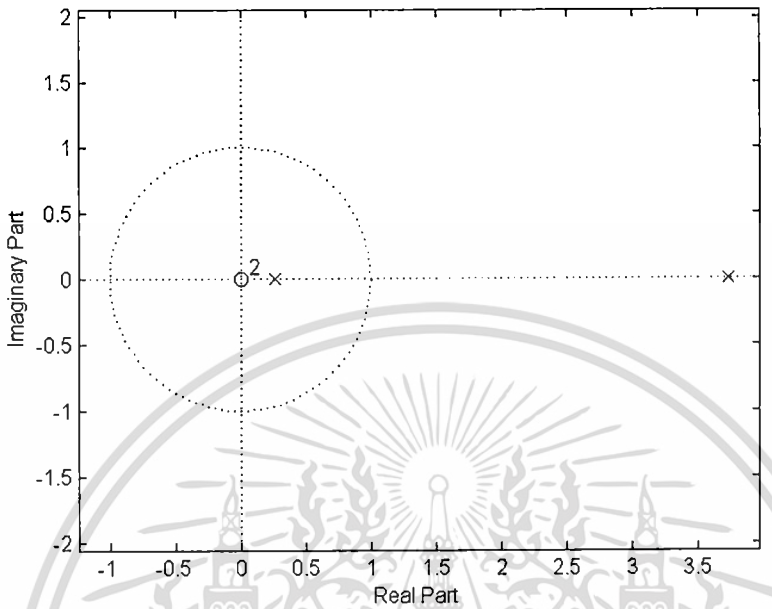
$$p_2=\frac{c_1-\sqrt{c_1^{\,2}+4c_2}}{2} \tag{3.4}$$

จากเงื่อนไขของการทำให้ระบบไม่มีเสถียรภาพ จะต้องมีค่าสัมประสิทธิ์ของวงจรกรองสัญญาณดิจิตอลอย่างน้อย 1 ตัว อยู่ภายนอกขอบเขตพื้นที่สามเหลี่ยมเสถียรภาพ ดังรูปที่ 3.2



รูปที่ 3.2 ขอบเขตพื้นที่สามเหลี่ยมเสถียรภาพ

หากเลือกค่าสัมประสิทธิ์อย่างน้อย 1 ตัวให้อยู่ภายนอกพื้นที่เสถียรภาพ แล้วจะพบว่าค่าโพลของฟังก์ชั่นถ่ายโอนจะมีค่ามากกว่าหนึ่ง ซึ่งทำให้อยู่นอกพื้นที่วงกลมหนึ่งหน่วย แสดงให้เห็นว่าวงจรกรองสัญญาณดิจิตอลนี้จะไม่มีเสถียรภาพ เช่น ถ้าเลือกค่า $c_1 = 4$ และ $c_2 = -1$ แล้วจะได้ตำแหน่งของโพลของวงจรเข้ารหัสดังต่อไปนี้ $p_1 = \frac{4+\sqrt{4^2-4}}{2} \approx 3.7321$ และ $p_2 = \frac{4-\sqrt{4^2-4}}{2} \approx 0.2679$ ดังแสดงในรูปที่ 3.3



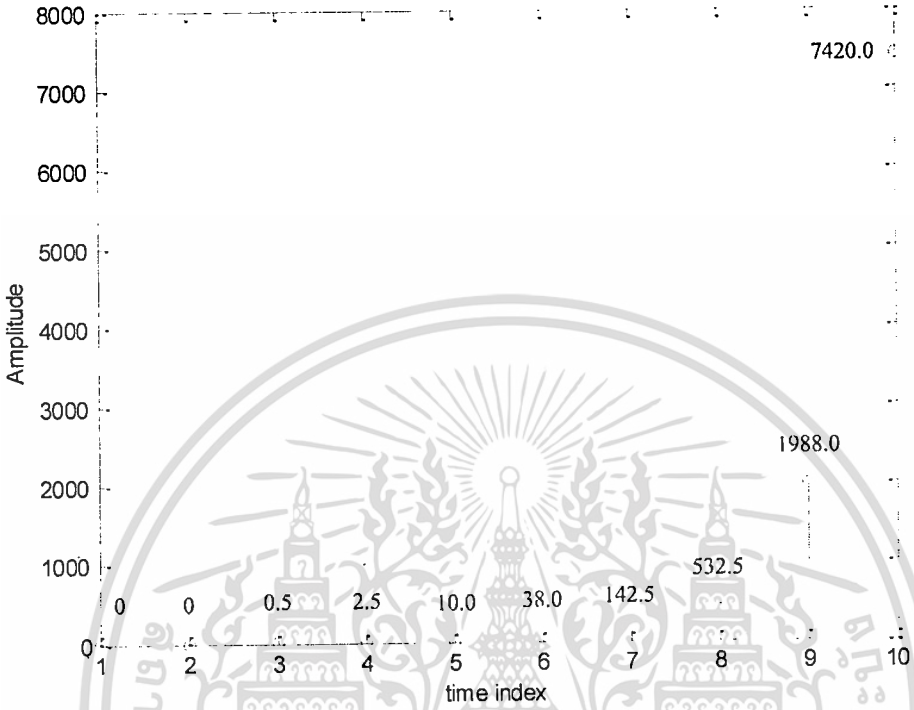
รูปที่ 3.3 ตำแหน่งโพลของวงจรเข้ารหัสแบบ IIR 2^{nd} order filter
กรณี $c_1 = 4$ และ $c_2 = -1$

หากทำการจำลองพฤติกรรมการทำงานของวงจรเข้ารหัสโดยไม่มีฟังก์ชั่น $f(\bullet)$ ตามสมการที่ (3.1) แล้วกำหนดค่าเงื่อนไขเริ่มต้นในการจำลองการทำงานโดยไม่มีการป้อนอินพุตเข้าไปยังวงจร (Zero input) ซึ่งจะกำหนดดังต่อไปนี้

กรณี $c_1 = 4$ และ $c_2 = -1$ แล้วให้ $x(k) = 0.5$ เนื่องจากในการจำลองการทำงานโดยใช้โปรแกรม MATLAB จำเป็นต้องกำหนดค่าเงื่อนไขเริ่มต้นคือ $y(1) = 0$, $y(2) = 0$ โดยกำหนดให้ $k = 3:10$ จะได้ผลดังรูปที่ 3.4 และเมื่อเทียบกับการคำนวณจากสมการ (3.1) นั่นคือ $y(k) = x(k) + c_1 y(k-1) + c_2 y(k-2)$ จะได้ผลลัพธ์ดังต่อไปนี้

$$y(3) = x(3) + 4y(2) - y(1) = 0.5 + 4(0) - 0 = 0.5$$

$$y(4) = x(4) + 4y(3) - y(2) = 0.5 + 4(0.5) - 0 = 2.5$$

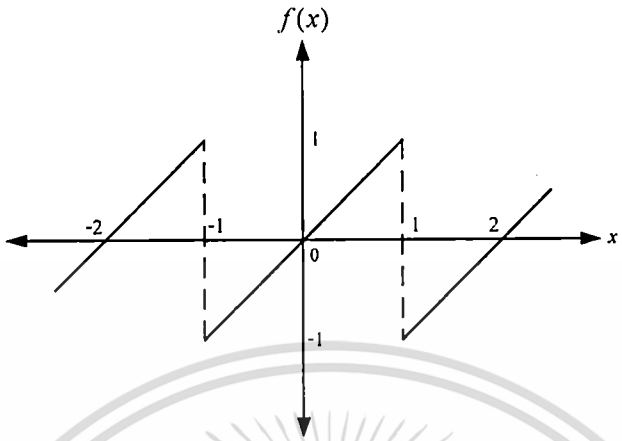


รูปที่ 3.4 เอาต์พุตของวงจรเข้ารหัสแบบ IIR 2^{nd} order filter กรณียังไม่ผ่านฟังก์ชัน $f(\bullet)$

เมื่อตรวจสอบจากผลการรันโปรแกรมที่ได้ทำการเขียนขึ้นพบว่าผลที่ได้จากการคำนวณและผลที่ได้จากการรันโปรแกรมมีค่าตรงกันดังนี้

$$y(k) = 0, 0.5, 2.5, 10.0, 38.0, 142.5, 532.5, 1988.0, 7420.0$$

จะเห็นได้ว่าค่าของ $y(k)$ จะมีค่าที่เพิ่มขึ้นเรื่อยๆซึ่งแสดงให้เห็นว่าวงจรเข้ารหัสนี้จะไม่มีเสถียรภาพ แต่เมื่อนำระบบดังกล่าวไปสร้างเป็นอุปกรณ์ฮาร์ดแวร์ จำเป็นจะต้องกำหนดความยาวของข้อมูลให้มีความยาวที่จำกัด (Finite word-length) เนื่องจากมีการกำหนด fixed-point ซึ่งจะทำให้เกิดการล้นขึ้น ดังนั้นในการจำลองการทำงานจึงต้องอาศัยฟังก์ชั่น $f(\bullet)$ เพื่อจำลองการล้นของข้อมูล โดยฟังก์ชั่น $f(\bullet)$ จะมีคุณลักษณะ (Characteristic) ดังรูปที่ 3.5



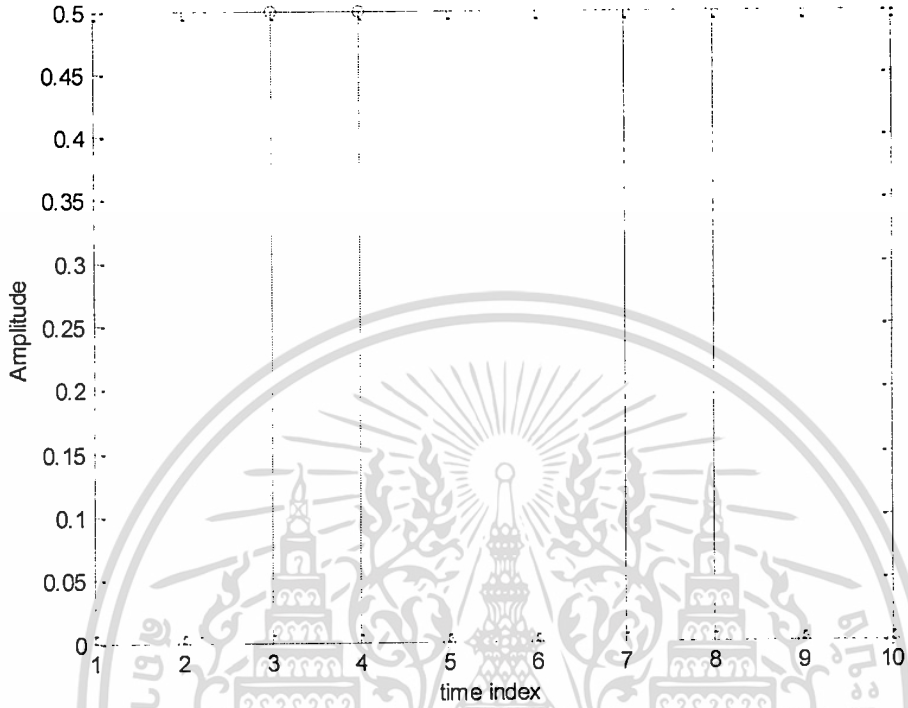
รูปที่ 3.5 คุณลักษณะของฟังก์ชั่น $f(x)=\left[(x+1)\bmod 2\right]-1$

ฟังก์ชั่นของ $f(\bullet)$ จะมีสมการคือ $f(x)=\left[(x+1)\bmod 2\right]-1$ ดังนั้น หลังจากเพิ่มฟังก์ชันนี้ลงไปแล้วค่า $y(k)$ จะเกิดการเปลี่ยนแปลงตามสมการของฟังก์ชัน $f(\bullet)$ ยกตัวอย่างเช่นที่ $y(4)=0.5$ เมื่อผ่านฟังก์ชัน $f(\bullet)$ จะกำหนดให้เป็นค่าใหม่คือ $\hat{y}(4)$ จะได้

$$\hat{y}(k)=\left[(y(k)+1)\bmod 2\right]-1$$

$$\hat{y}(4)=(0.5+1)\bmod 2-1=1.5\bmod 2-1=1.5-1=0.5$$

ตรวจสอบผลการทำงานของวงจรเข้ารหัสเมื่อมีฟังก์ชั่น $f(\bullet)$ โดยกำหนด $c_1=4$ และ $c_2=-1$ แล้วให้ $x(k)=0.5$ กำหนดค่าเงื่อนไขเริ่มต้นคือ $y(1)=0$, $y(2)=0$ โดยกำหนดให้ $k=3:10$ จะได้ผลดังรูปที่ 3.6



รูปที่ 3.6 ค่าเอาต์พุตจากการจำลองการทำงานของวงจรเข้ารหัสแบบ IIR 2nd order filter เมื่อมีฟังก์ชั่น $f(\bullet)$

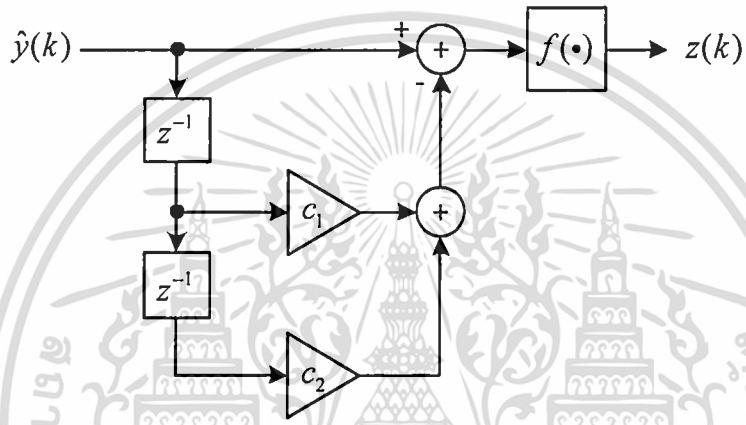
โดยตรวจสอบค่าจากการรันโปรแกรมได้ดังนี้

$$\hat{y}(k) = 0\ ,0\ ,0.5000\ ,0.5000\ ,0\ ,0\ ,0.5000\ ,0.5000\ ,0\ ,0$$

เมื่อพิจารณาจะพบว่าค่าที่ได้จากการคำนวณจะมีค่าที่ตรงกับในรูปที่ 3.6 ดังนั้นจึงสามารถบอกได้ว่า เมื่อกำหนดให้ค่าสัมประสิทธิ์อย่างน้อยหนึ่งตัวให้มีค่าอยู่นอกขอบเขตพื้นที่สามเหลี่ยมเสถียรภาพแล้วจะทำให้วงจรเข้ารหัสซึ่งเป็นวงจรกรองแบบอิมพัลส์ไม่จำกัดเกิดตำแหน่งของโพลอยู่ภายนอกขอบเขตพื้นที่วงกลมหนึ่งหน่วย (Unit cycle) ซึ่งทำให้วงจรไม่มีความเสถียรภาพทำให้ค่าที่ได้จากวงจรจะมีค่าเพิ่มขึ้นเรื่อยๆ จากนั้นจะใช้ฟังก์ชั่น $f(\bullet)$ เพื่อทำให้เกิดการล้น

3.1.1.2 วงจรถอดรหัสที่ใช้วงจรกรองสัญญาณดิจิตอลอันดับสองชนิด อิมพัลส์จำกัด (Finite Impulse Response Second Order Filter)

จะมีโครงสร้างที่เป็นส่วนกลับ (Inverse) ของวงจรเข้ารหัสวงจรเข้ารหัสแบบ IIR 2^{nd} order filter ดังรูปที่ 3.7



รูปที่ 3.7 โครงสร้างของวงจรถอดรหัสแบบ FIR 2^{nd} order filter

จากรูปที่ 3.7 เมื่อยังไม่พิจารณาฟังก์ชั่น $f(\bullet)$ สามารถเขียนสมการผลต่างสืบเนื่องได้ดังสมการที่ (3.5) และฟังก์ชั่นถ่ายโอนได้ดังต่อไปนี้

$$z(k) = \hat{y}(k) - c_1\hat{y}(k-1) - c_2\hat{y}(k-2) \tag{3.5}$$

จากสมการที่ (3.5) สามารถเขียนเป็นฟังก์ชั่นถ่ายโอน (Transfer function) ได้ดังสมการที่ (3.6)

$$Z(z) = \hat{Y}(z) - c_1\hat{Y}(z)z^{-1} - c_2\hat{Y}(z)z^{-2}$$

$$H(z) = \frac{Z(z)}{\hat{Y}(z)} = 1 - c_1z^{-1} - c_2z^{-2} \tag{3.6}$$

จากสมการที่ (3.5) จะเห็นได้ว่ามีโครงสร้างแบบไม่ป้อนกลับ (Non-recursive) ซึ่งเป็นลักษณะทั่วไปของวงจรกรองสัญญาณดิจิตอลชนิดอิมพัลส์จำกัด และจากสมการที่ (3.6) ฟังก์ชั่นถ่ายโอนของวงจรถอดรหัสจะมีตำแหน่งของซีโร่ (zero) ดังต่อไปนี้

$$1-c_1z^{-1}-c_2z^{-2}=0$$

$$z^2-c_1z-c_2=0$$

ดังนั้น

$$z_1=\frac{c_1+\sqrt{c_1^{\,2}+4c_2}}{2} \tag{3.7}$$

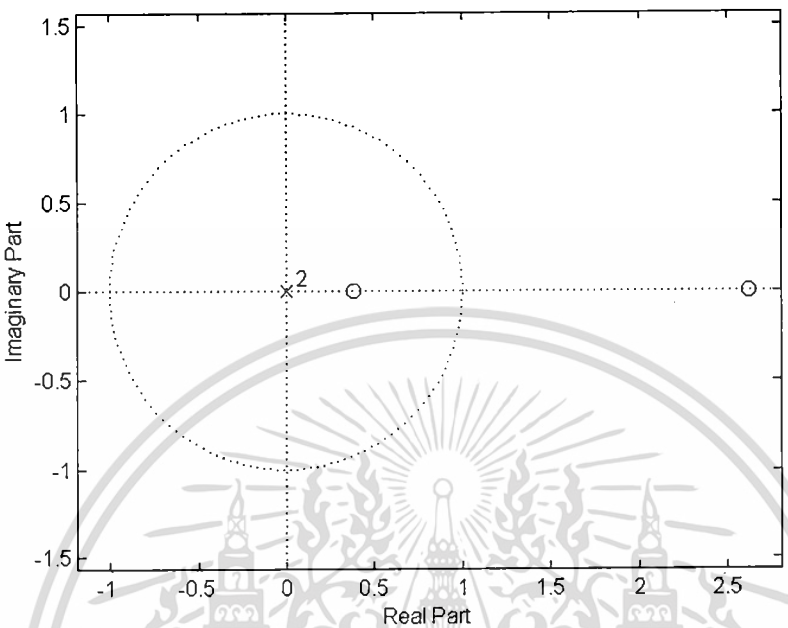
และ

$$z_2=\frac{c_1-\sqrt{c_1^{\,2}+4c_2}}{2} \tag{3.8}$$

ค่าสัมประสิทธิ์ของวงจรถอดรหัสจะต้องใช้ค่าเดียวกันกับวงจรเข้ารหัส ดังนั้นจึงกำหนดค่าสัมประสิทธิ์ให้เหมือนกับตัวอย่างก่อนหน้านี้คือให้ค่า $c_1=4$ และ $c_2=-1$ แล้วจะได้ว่า

$$z_1=\frac{4+\sqrt{4^2-4}}{2}\approx 3.7321 \qquad \text{และ} \qquad z_2=\frac{4-\sqrt{4^2-4}}{2}\approx 0.2679$$

แสดงตำแหน่งของซีโร่ดังรูปที่ 3.8



รูปที่ 3.8 ตำแหน่งซีโร่ของวงจรถอดรหัสแบบ FIR 2^{nd} order filter

เมื่อพิจารณาตำแหน่งของซีโร่จากรูปที่ 3.8 แล้วนำไปเปรียบเทียบกับตำแหน่งของโพลในรูปที่ 3.3 จะพบว่าค่าโพลของวงจรเข้ารหัสกับค่าซีโร่ของวงจรถอดรหัส นั้นจะมีหักล้าง คือ ตำแหน่งโพลของวงจรเข้ารหัสจะมีค่าเท่ากับตำแหน่งซีโร่ของวงจรถอดรหัส ดังนั้นจึงทำให้ตำแหน่งของโพลและซีโร่ของวงจรทั้งสองหักล้างกัน จึงส่งผลให้ในการเข้ารหัสและถอดรหัสจะต้องมีค่าสัมประสิทธิ์ที่เหมือนกันทั้งวงจรเข้ารหัสและวงจรถอดรหัส

จากสมการที่ (3.5) เมื่อจำลองการทำงานโดยแยกเป็น 2 กรณีคือ

1. ไม่มีฟังก์ชั่น $f(\bullet)$ ดังนั้นเมื่อนำอินพุตคือ $\hat{y}(k)$ มาทำการคำนวณหาค่า $z(k)$ และกำหนดค่าสัมประสิทธิ์ชุดเดียวกันคือ $c_1 = 4$ และ $c_2 = -1$ จะได้

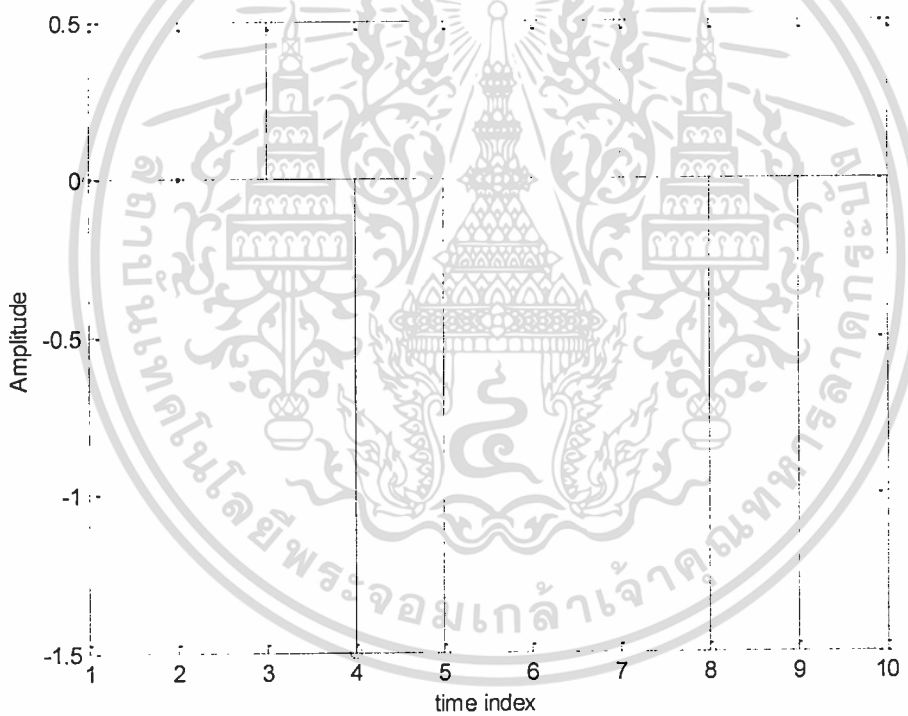
$$z(k) = \hat{y}(k) - c_1\hat{y}(k-1) - c_2\hat{y}(k-2)$$

$$z(3) = \hat{y}(3) - 4\hat{y}(2) - (-1)\hat{y}(1) = 0.5 - 4(0) + (0) = 0.5$$

$$z(4) = \hat{y}(4) - 4\hat{y}(3) - (-1)\hat{y}(2) = 0.5 - 4(0.5) + 0 = -1.5$$

และเมื่อจำลองการทำงานจะได้ค่าของ $z(k)$ ดังนี้

$$z(k) = 0,\ 0,\ 0.5,\ -1.5,\ -1.5,\ 0.5,\ 0.5,\ -1.5,\ -1.5,\ 0.5$$



รูปที่ 3.9 ค่าเอาต์พุตของวงจรถอดรหัสแบบ FIR 2nd order filter
กรณียังไม่ผ่านฟังก์ชัน $f(\bullet)$

จากรูปที่ 3.9 จะเห็นได้ว่าค่าที่ตำแหน่ง $z(1)=z(2)=0$ จะมีค่าเท่ากับศูนย์เนื่องจากการกำหนดให้ค่า $\hat{y}(1)=\hat{y}(2)=0$ ดังนั้นค่าที่ถอดรหัสออกมาได้ก็จะมีค่าเป็นศูนย์ด้วยเช่นกัน

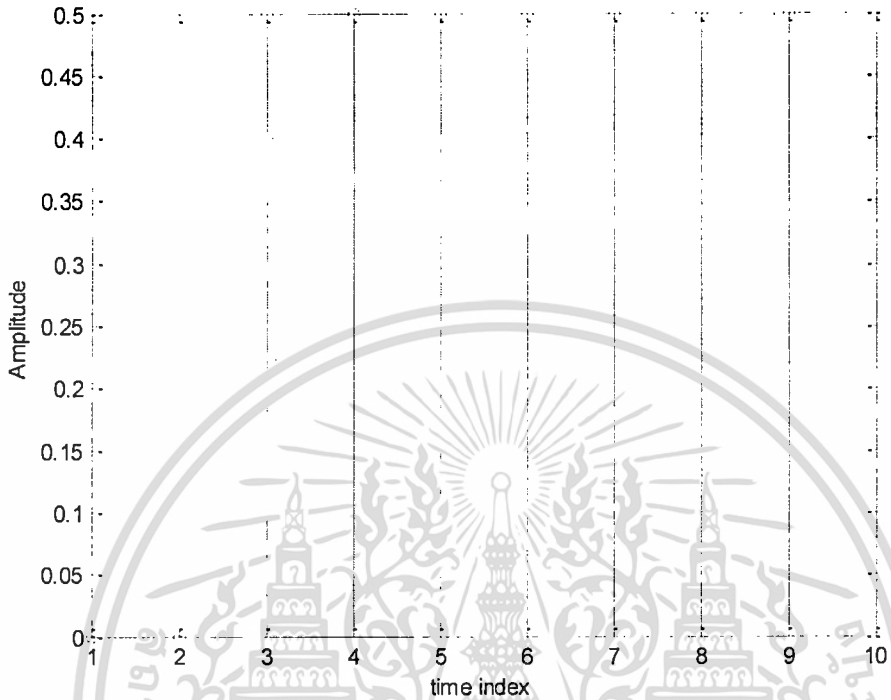
สังเกตค่าของ $z(k)$ ที่ตำแหน่งต่างๆ จากการคำนวณจะมีค่าเท่ากับค่าที่ได้จากการรันโปรแกรม ดังนั้นจึงสามารถสรุปได้ว่า หากกำหนดค่าสัมประสิทธิ์ของทั้งสองวงจรให้มีค่าเท่ากันแล้ว จะสามารถทำการเข้ารหัสและถอดรหัสได้

2. เพิ่มฟังก์ชั่น $f(\bullet)$ เข้าไป จากนั้นหาค่า $z(k)$ ได้ดังต่อไปนี้

$$z(3)=f\{\hat{y}(3)-4\hat{y}(2)-(-1)\hat{y}(1)\}$$
$$=f\{0.5-4(0)+(0)\}$$
$$=(0.5+1)\bmod 2-1=1.5-1$$
$$z(3)=0.5$$

$$z(4)=f\{\hat{y}(4)-4\hat{y}(3)-(-1)\hat{y}(2)\}$$
$$=f\{2.5-4(0.5)+0\}$$
$$=(0.5+1)\bmod 2-1=1.5-1$$
$$z(4)=0.5$$

สังเกตค่าของ $z(k)$ ที่ตำแหน่งต่างๆ จากการคำนวณจะมีค่าเท่ากับค่าที่ได้จากการรันโปรแกรม ดังรูปที่ 3.10



รูปที่ 3.10 ค่าเอาต์พุตของวงจรถอดรหัสแบบ FIR 2^{nd} order filter กรณีที่มีฟังก์ชั่น $f(\bullet)$

และเมื่อตรวจสอบจากผลการรันโดยใช้โปรแกรมจะได้ดังนี้

$$z(k) = 0,\ 0,\ 0.5,\ 0.5,\ 0.5,\ 0.5,\ 0.5,\ 0.5,\ 0.5,\ 0.5$$

จะเห็นได้ว่าค่าเอาต์พุตของวงจรถอดรหัสแบบ FIR 2^{nd} order filter : $z(k)$ เท่ากับค่าอินพุตของวงจรเข้ารหัสแบบ IIR 2^{nd} order filter : $x(k)$

หมายเหตุ สำหรับการหารเอาเศษหรือที่เรียกว่า Modulo operation นั้น ในโปรแกรม MATLAB จะเขียนด้วย $\mathrm{mod}(x,y)$ คือ นำค่า x มา mod ด้วย y หรือเขียนโดยทั่วไปได้คือ $x \bmod y$ ซึ่งจะมีวิธีการคิดแบ่งเป็น 2 กรณีดังนี้

1. กรณีที่เป็นค่าบวก เช่น

$3 \bmod 2 = 1$ คือ นำ 3 หารด้วย 2 จะได้เศษเกินมา 1 ซึ่งก็คือคำตอบ

$$\begin{array}{r} 1 \\ 2\overline{)3} \\ \underline{2} \\ \boxed{1} \end{array}$$

$3.5 \bmod 2 = 1.5$ คือนำ 3.5 หารด้วย 2 จะได้เศษเกินมา 1.5 ซึ่งก็คือคำตอบ

$$\begin{array}{r} 1 \\ 2\overline{)3.5} \\ \underline{2} \\ \boxed{1.5} \end{array}$$

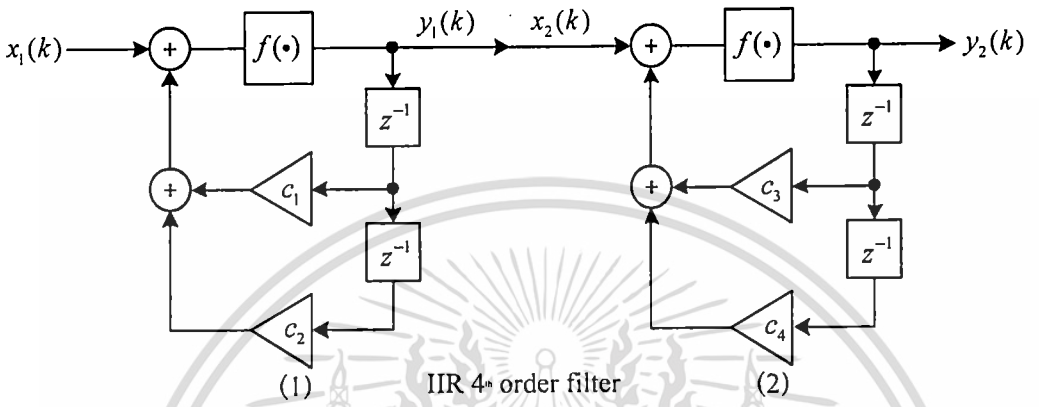
2. กรณีที่เป็นค่าลบ เช่น

$-3 \bmod 2 = 1$ วิธีการหาคือเราจะให้หาว่าค่าถัดไปที่ 2 หารลงตัวได้ นั่นคือ -4 แต่ค่าที่นำมา mod คือ -3 ซึ่งยังขาดอีก $|-4-(-3)| = |1|$ (ให้มองเป็นขนาด) คำตอบก็คือ 1

$-7.5 \bmod 2 = 0.5$ ซึ่งค่าถัดไปที่ 2 หารลงตัวคือ -8 ดังนั้น ค่าที่ได้คือ $|-8-(-7.5)| = |0.5|$ คำตอบก็คือ 0.5

คำตอบที่ได้จากการ mod จะมีค่าเป็นบวกเสมอ ดังเช่น $-7.5 \bmod 2 = 0.5$ และ $7.5 \bmod 2 = 1.5$

3.1.1.3 วงจรเข้ารหัสที่ใช้วงจรกรองสัญญาณดิจิตอลอันดับสี่ชนิดผลตอบสนองอิมพัลส์ไม่จำกัด (Infinite Impulse Response Fourth Order Filter)



รูปที่ 3.11 โครงสร้างของวงจรเข้ารหัสแบบ IIR 4th order filter

โดยการนำวงจรเข้ารหัสแบบ IIR 2nd order filter สองตัวมาต่อกัน ซึ่งจะทำการจำลองพฤติกรรมการทำงานของวงจรเข้ารหัสแบบ IIR 4th order filter โดยแบ่งเป็น 2 ส่วน ดังนี้

1) วงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (1)

โดยกำหนดค่าสัมประสิทธิ์ $c_1 = 4$ และ $c_2 = -1$ แล้วให้ $x_1(k) = 0.5$ ซึ่งสมการในการคำนวณ กรณียังไม่ผ่านฟังก์ชั่น $f(\cdot)$ คือ

$$\tilde{y}_1(k) = x_1(k) + c_1 y_1(k-1) + c_2 y_1(k-2)$$

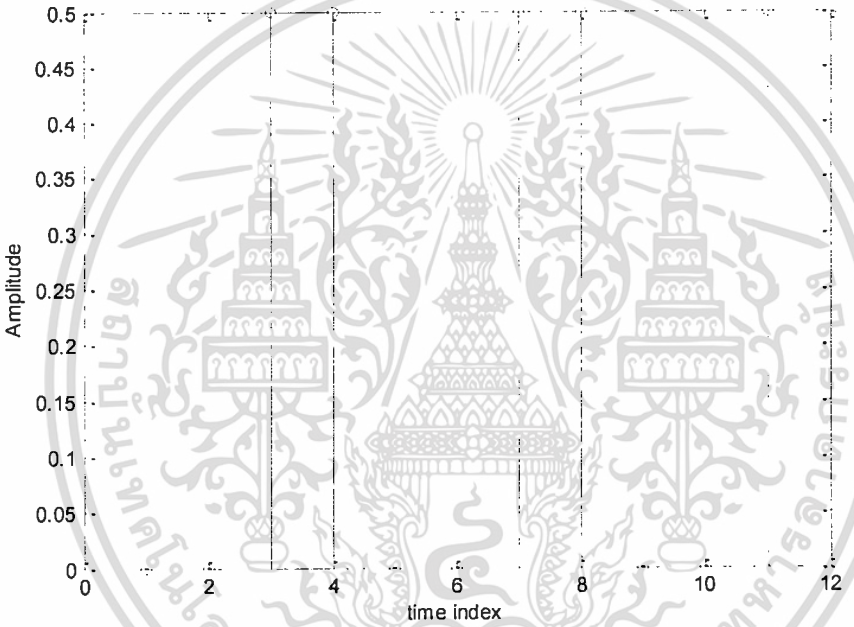
กรณีผ่านฟังก์ชั่น $f(\cdot)$ แล้วจะได้

$$y_1(k) = \left[\left(\tilde{y}_1(k) + 1\right) \bmod 2\right] - 1$$

แต่เนื่องจากในการจำลองการทำงานโดยใช้โปรแกรม MATLAB จำเป็นต้องกำหนดค่าเงื่อนไขเริ่มต้นเป็น $y_1(1)=0$, $y_1(2)=0$ และให้ $k=3:12$ จะได้ผลจากการรันโปรแกรมดังนี้

$$y_1(k)=0\ ,0\ ,0\ ,0.5\ ,0.5\ ,0\ ,0\ ,0.5\ ,0.5\ ,0\ ,0\ ,0.5\ ,0.5$$

ดังแสดงในรูปที่ 3.12



รูปที่ 3.12 ค่าเอาต์พุต $y_1(k)$ จากการจำลองการทำงานของวงจรเข้ารหัสแบบ IIR 2^{nd} order filter ตัวที่ (1) เมื่อมีฟังก์ชั่น $f(\bullet)$

2) วงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (2)

โดยกำหนดค่าสัมประสิทธิ์ $c_3 = 6$ และ $c_4 = -2$ แล้วให้ $y_1(k) = x_2(k)$ ซึ่งสมการในการคำนวณ กรณียังไม่ผ่านฟังก์ชั่น $f(\bullet)$ คือ

$$\tilde{y}_2(k) = x_2(k) + c_3 y_2(k-1) + c_4 y_2(k-2)$$

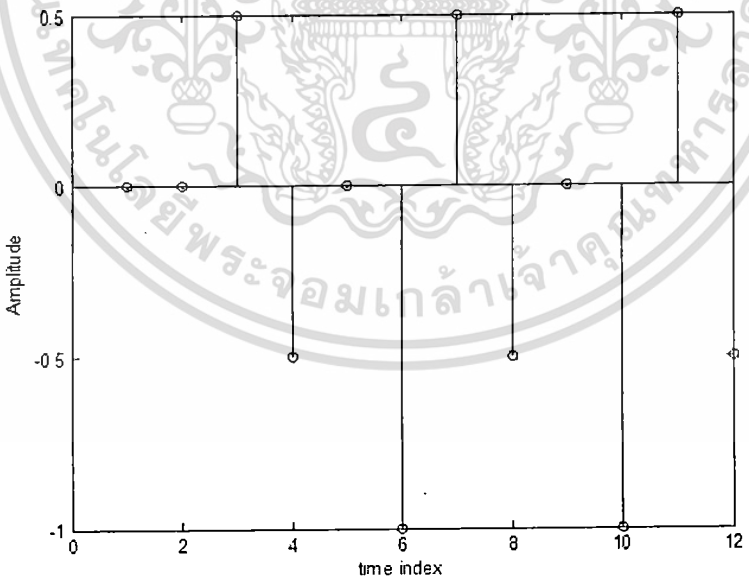
กรณีผ่านฟังก์ชั่น $f(\bullet)$ แล้วจะได้

$$y_2(k) = \left[(\tilde{y}_2(k) + 1) \bmod 2 \right] - 1$$

แต่เนื่องจากในการจำลองการทำงานโดยใช้โปรแกรม MATLAB จำเป็นต้องกำหนดค่าเงื่อนไขเริ่มต้นเป็น $y_2(1) = 0$, $y_2(2) = 0$ และให้ $k = 3:12$ จะได้ผลจากการรันโปรแกรมดังนี้

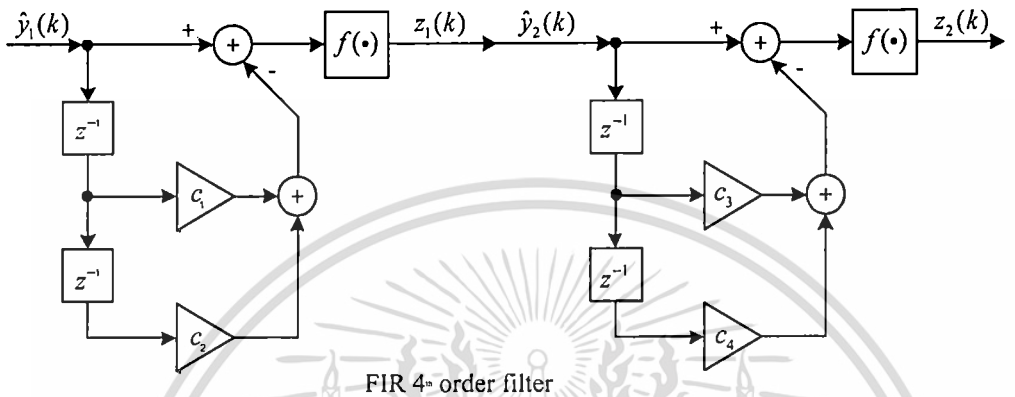
$$y_2(k) = 0, 0, 0, 0.5, -0.5, 0, -1, 0.5, -0.5, 0, -1, 0.5, -0.5$$

ดังแสดงในรูปที่ 3.13



รูปที่ 3.13 ค่าเอาต์พุต $y_2(k)$ จากการจำลองการทำงานของวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (2) เมื่อมีฟังก์ชั่น $f(\bullet)$

3.1.1.4 วงจรถอดรหัสที่ใช้วงจรกรองสัญญาณดิจิตอลอันดับสี่ชนิดผลตอบสนองอิมพัลส์จำกัด (Finite Impulse Response Fourth Order Filter)



FIR 4th order filter

Decoder

รูปที่ 3.14 โครงสร้างของวงจรถอดรหัสแบบ FIR 4th order filter

โดยการนำวงจรเข้ารหัสแบบ FIR 2nd order filter สองตัวมาต่อกัน ซึ่งจะทำการจำลองพฤติกรรมการทำงานของวงจรถอดรหัสแบบ IIR 4th order filter โดยแบ่งเป็น 2 ส่วน ดังนี้

1) วงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (1)

โดยกำหนดค่าสัมประสิทธิ์ $c_1 = 6$ และ $c_2 = -2$ แล้วให้ $y_2(k) = \hat{y}_1(k)$ ซึ่งสมการในการคำนวณ กรณียังไม่ผ่านฟังก์ชั่น $f(\bullet)$ คือ

$$\tilde{z}_1(k) = \hat{y}_1(k) - c_3\hat{y}_1(k-1) - c_4\hat{y}_1(k-2)$$

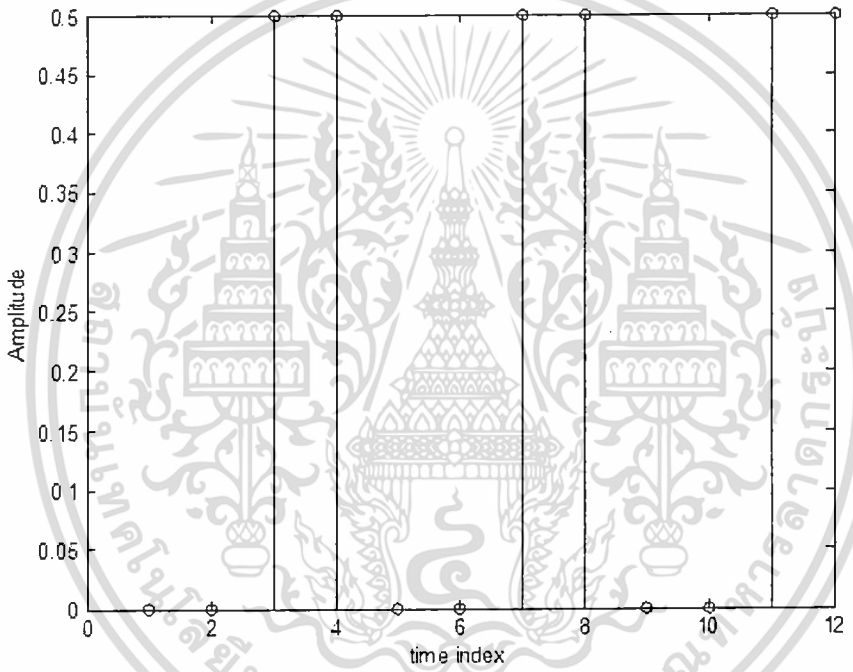
กรณีผ่านฟังก์ชั่น $f(\bullet)$ แล้วจะได้

$$z_1(k) = \left[\left(\tilde{z}_1(k)+1\right) \bmod 2\right] - 1$$

แต่เนื่องจากในการจำลองการทำงานโดยใช้โปรแกรม MATLAB จำเป็นต้องกำหนดค่าเงื่อนไขเริ่มต้นเป็น $\hat{y}_1(1)=0$, $\hat{y}_1(2)=0$ และให้ $k=3:12$ จะได้ผลจากการรันโปรแกรมดังนี้

$$z_1(k)=0\ ,0\ ,0\ ,0.5\ ,0.5\ ,0\ ,0\ ,0.5\ ,0.5\ ,0\ ,0\ ,0.5\ ,0.5$$

ดังแสดงในรูปที่ 3.15



รูปที่ 3.15 ค่าเอาต์พุต $z_1(k)$ จากการจำลองการทำงานของวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (1) เมื่อมีฟังก์ชั่น $f(\bullet)$

2) วงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (2)

โดยกำหนดค่าสัมประสิทธิ์ $c_3=4$ และ $c_4=-1$ แล้วให้ $z_1(k)=\hat{y}_2(k)$ ซึ่งสมการในการคำนวณ กรณียังไม่ผ่านฟังก์ชั่น $f(\bullet)$ คือ

$$\tilde{z}_2(k)=\hat{y}_2(k)-c_3\hat{y}_2(k-1)-c_4\hat{y}_2(k-2)$$

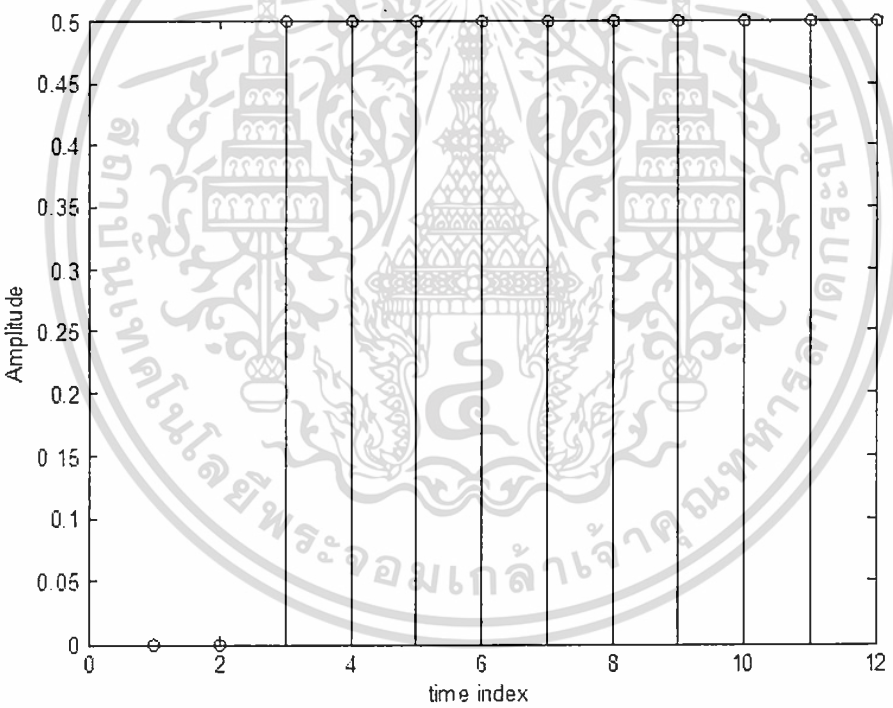
กรณีผ่านฟังก์ชั่น $f(\bullet)$ แล้วจะได้

$$z_2(k)=\left[\left(\tilde{z}_2(k)+1\right)\bmod 2\right]-1$$

แต่เนื่องจากในการจำลองการทำงานโดยใช้โปรแกรม MATLAB จำเป็นต้องกำหนดค่าเงื่อนไขเริ่มต้นเป็น $y_2(1)=0$, $y_2(2)=0$ และให้ $k=3:12$ จะได้ผลจากการรันโปรแกรมดังนี้

$$z_2(k)=0\ ,0\ ,0\ ,0.5\ ,0.5\ ,0.5\ ,0.5\ ,0.5\ ,0.5\ ,0.5\ ,0.5\ ,0.5\ ,0.5$$

ดังแสดงในรูปที่ 3.16

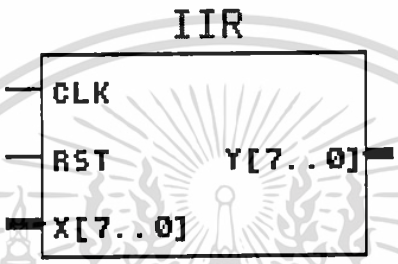


รูปที่ 3.16 ค่าเอาต์พุต $z_2(k)$ จากการจำลองการทำงานของวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (2) เมื่อมีฟังก์ชั่น $f(\bullet)$

3.1.2 การออกแบบในส่วนของ FPGA

โครงสร้างภายในส่วนของ FPGA ที่ใช้สำหรับสร้างวงจรเข้ารหัสและถอดรหัสเป็นดังนี้

3.1.2.1 วงจรเข้ารหัสที่ใช้วงจรกรองสัญญาณดิจิตอลอันดับสองชนิดผลตอบสนองอิมพัลส์ไม่จำกัด (IIR 2^{nd} order filter)



รูปที่ 3.17 วงจรเข้ารหัสแบบ IIR 2^{nd} order filter ใน FPGA

จากส่วนของการจำลองการทำงานโดยโปรแกรม MATLAB ก่อนหน้านี้ ใช้ค่าสัมประสิทธิ์ $c_1 = 4$ กับ $c_2 = -1$ ดังนั้นในการสร้างวงจรเข้ารหัสนี้จะใช้ค่าสัมประสิทธิ์ชุดเดียวกัน ซึ่งโครงสร้างที่ใช้ในการสร้างมีดังรูปที่ 3.1

โดยการจำลองการทำงานของวงจรเข้ารหัสกำหนดเงื่อนไขดังนี้

1. สมการผลต่างสืบเนื่องของวงจรเข้ารหัส

$$y(k) = f\{x(k) + C_1 y(k-1) + C_2 y(k-2)\}$$

2. อินพุต $x(k) = 0.5$

3. กำหนดเงื่อนไขเริ่มต้น $y(-1) = 0, y(-2) = 0$

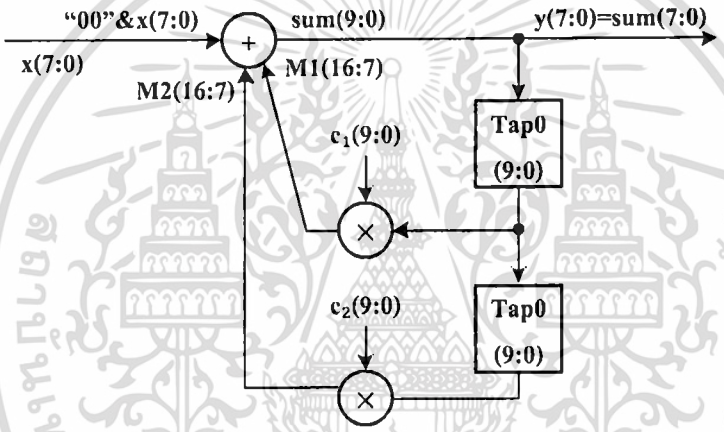
4. สมการฟังก์ชันที่ทำให้เกิดการล้น

$f(\bullet)$ คือ $f(x) = (x+1) \bmod 2 - 1$

5. โดยกำหนดรูปแบบตัวเลขของอินพุตและเอาต์พุตคือ x.xxx xxxx เป็นการกำหนด fixed-point เพื่อทำให้เกิดการล้น

6. ค่าสัมประสิทธิ์จะถูกแทนด้วยรูปแบบตัวเลขคือ xxx.xxx xxxx โดยเพิ่ม 2 บิตหน้าตามค่าสัมประสิทธิ์ที่กำหนด
7. ในการออกแบบจริงจึงต้องมีการเพิ่มความยาวข้อมูลของอินพุตอีก 2 บิตหน้าเพื่อใช้ในการประมวลผล และเอาต์พุตก็จะตัด 2 บิตหน้าทิ้งเช่นกัน

โครงสร้างที่จะใช้ในการออกแบบวงจรเข้ารหัสโดยแสดงขนาดความยาวของข้อมูลโดยละเอียด ซึ่งในรูปที่ 3.18 แสดงโครงสร้างของวงจรเข้ารหัสขนาด 8 บิต



รูปที่ 3.18 โครงสร้างแบบละเอียดที่ใช้สร้างวงจรเข้ารหัสแบบ IIR 2nd order filter

หลังจากที่ Tap0 ที่มีจำนวน 10 บิต คูณ c_1 ที่มีจำนวน 10 บิตเช่นเดียวกัน จะได้ค่าผลคูณ (M1) จำนวน 20 บิต โดยจะเลือกเพียงใช้ 10 บิต คือบิตที่ 7-16 ของ M1 เนื่องจากเป็นค่าผลลัพธ์ที่ได้ตาม format ที่กำหนดไว้ ส่วนบิตที่เหลือไม่จำเป็นต้องใช้ในการคำนวณสามารถตัดทิ้งได้ ซึ่งในการออกแบบฮาร์ดแวร์จริงไม่จำเป็นต้องใช้ฟังก์ชัน $f(\bullet)$ เพราะพฤติกรรมการสั้นสามารถเกิดขึ้นได้ด้วยตัวฮาร์ดแวร์เอง

ยกตัวอย่างการทดลองโดยกำหนดค่าสัมประสิทธิ์ของวงจรกรองสัญญาณดิจิตอลเป็น $c_1 = 4 = 100.0\ 0000$, $c_2 = -1 = 111.0\ 0000$ และกำหนดให้ค่า $x(k) = 0.5 = 000.1\ 0000$ สำหรับทุกค่าของ k โดยให้ $y(-1) = 0, y(-2) = 0$ แล้ว

กำหนดให้เอาต์พุตมี format เป็น $x.xxx\ xxxx$ โดยให้บิตแรกเป็นบิตเครื่องหมาย ซึ่งทั้งการคำนวณค่าเอาต์พุต $y(k)$ ของวงจรกรองสัญญาณดิจิตอลที่จำลองการทำงานจากโปรแกรม MATLAB และ วงจรกรองสัญญาณดิจิตอลในทางปฏิบัตินั้นจะแสดงควบคู่กันไปเพื่อให้สามารถเห็นผลลัพธ์ได้อย่างชัดเจน แสดงดังในตารางที่ 3.1

ตารางที่ 3.1 ผลการคำนวณจากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรเข้ารหัสแบบ IIR 2^{nd} order filter

ฐานสิบ	ฐานสองแบบ 2's complement
$k=0;\ I(0)=x(0)+c_1\ y(-1)+c_2\ y(-2)$ $=0.5+(4\times0)+(-1\times0)=0.5$ $\therefore y(0)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000+ $\Leftarrow 0.5$ 000.000 0000+ $\Leftarrow 0$ 000.000 0000 $\Leftarrow 0$ 00[0.100 0000] $\Leftarrow 0.5$
$k=1;\ I(1)=x(1)+c_1\ y(0)+c_2\ y(-1)$ $=0.5+(4\times0.5)+(-1\times0)=-1.5$ $\therefore y(1)=[(-1.5+1)\bmod 2]-1=0.5$	000.100 0000+ $\Leftarrow 0.5$ 010.000 0000+ $\Leftarrow 2$ 000.000 0000 $\Leftarrow 0$ 00[0.100 0000] $\Leftarrow 0.5$
$k=2;\ I(2)=x(2)+c_1\ y(1)+c_2\ y(0)$ $=0.5+(4\times0.5)+(-1\times0.5)=2$ $\therefore y(2)=[(2+1)\bmod 2]-1=0$	000.100 0000+ $\Leftarrow 0.5$ 010.000 0000+ $\Leftarrow 2$ 111.100 0000 $\Leftarrow -0.5$ 11[0.000 0000] $\Leftarrow 0$
$k=3;\ I(3)=x(3)+c_1\ y(2)+c_2\ y(1)$ $=0.5+(4\times0)+(-1\times0.5)=0$ $\therefore y(3)=[(0+1)\bmod 2]-1=0$	000.100 0000+ $\Leftarrow 0.5$ 000.000 0000+ $\Leftarrow 0$ 111.100 0000 $\Leftarrow -0.5$ 00[0.000 0000] $\Leftarrow 0$
$k=4;\ I(4)=x(4)+c_1\ y(3)+c_2\ y(2)$ $=0.5+(4\times0)+(-1\times0)=0.5$ $\therefore y(4)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000+ $\Leftarrow 0.5$ 000.000 0000+ $\Leftarrow 0$ 000.000 0000 $\Leftarrow 0$ 00[0.100 0000] $\Leftarrow 0.5$

ตารางที่ 3.1 ผลการคำนวณจากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรเข้ารหัสแบบ IIR 2nd order filter (ต่อ)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=5;\ I(5)=x(5)+c_1\ y(4)+c_2\ y(3)$ $=0.5+(4\times0.5)+(-1\times0)=2.5$ $\therefore y(5)=[(2.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐0.5 010.000 0000+ ⇐2 000.000 0000 ⇐0 01[0.100 0000] ⇐0.5
$k=6;\ I(6)=x(6)+c_1\ y(5)+c_2\ y(4)$ $=0.5+(4\times0.5)+(-1\times0.5)=2$ $\therefore y(6)=[(2+1)\bmod 2]-1=0$	000.100 0000+ ⇐0.5 010.000 0000+ ⇐2 111.100 0000 ⇐–0.5 11[0.000 0000] ⇐0
$k=7;\ I(7)=x(7)+c_1\ y(6)+c_2\ y(5)$ $=0.5+(4\times0)+(-1\times0.5)=0$ $\therefore y(7)=[(0+1)\bmod 2]-1=0$	000.100 0000+ ⇐0.5 000.000 0000+ ⇐0 111.100 0000 ⇐–0.5 11[0.000 0000] ⇐0
$k=8;\ I(8)=x(8)+c_1\ y(7)+c_2\ y(6)$ $=0.5+(4\times0)+(-1\times0)=0.5$ $\therefore y(8)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐0.5 000.000 0000+ ⇐0 000.000 0000 ⇐0 00[0.100 0000] ⇐0.5
$k=9;\ I(9)=x(9)+c_1\ y(8)+c_2\ y(7)$ $=0.5+(4\times0.5)+(-1\times0)=2.5$ $\therefore y(9)=[(2.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐0.5 010.000 0000+ ⇐2 000.000 0000 ⇐0 01[0.100 0000] ⇐0.5

ตรวจสอบผลการคำนวณโดยการใช้ค่าที่คำนวณได้เทียบกับค่าที่ได้จากโปรแกรม MATLAB ซึ่งมีโค้ดของโปรแกรมดังรูปที่ 3.19 และกำหนดให้ค่าเงื่อนไขเริ่มต้นอยู่ที่ตำแหน่ง $y(-2) \Rightarrow y(1) = 0$ และ $y(-1) \Rightarrow y(2) = 0$ ดังนั้นจึงเริ่มคำนวณที่ $y(0) = y(3)$ ขึ้นไป

```
clear all;
clc;
% coeefficient Encoder
c1=4;
c2=-1;
% initial value
y(1)=0;
y(2)=0;
%%%% Encoder %%%%
for k=3:12;
    x(k)=0.5;
    I=x(k)+c1*y(k-1)+c2*y(k-2);
    y(k)=mod(I+1,2)-1;

end;
disp(y);
```

รูปที่ 3.19 โค้ดของโปรแกรม MATLAB ที่ใช้ในการจำลองการทำงานของวงจรเข้ารหัสแบบ IIR 2^{nd} order filter

ผลการคำนวณจากโปรแกรมได้ดังนี้

Columns 1 through 7

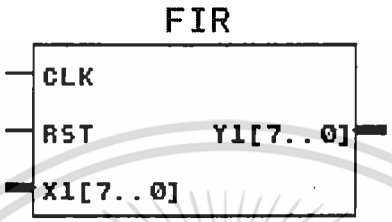
0 0 0.5 0.5 0 0 0.5

Columns 8 through 12

0.5 0 0 0.5 0.5

3.1.2.2 วงจรถอดรหัสที่เป็นวงจรกรองสัญญาณดิจิตอลอันดับสองชนิดผลตอบสนองอิมพัลส์จำกัด (FIR 2nd order filter)

มีโครงสร้างภายใน FPGA ดังรูปที่ 3.20



รูปที่ 3.20 วงจรถอดรหัสแบบ FIR 2nd order filter ใน FPGA

ในการจำลองการทำงานโดยใช้โปรแกรม MATLAB นั้นจะต้องมีฟังก์ชัน overflow ($f(\bullet)$) เพื่อทำให้เกิดการล้นขึ้นในวงจรกรองสัญญาณดิจิตอล ซึ่งโครงสร้างที่ใช้ในการสร้างมีดังรูปที่ 3.7

โดยการจำลองการทำงานของวงจรเข้ารหัสกำหนดเงื่อนไขดังนี้

1. สมการผลต่างสืบเนื่องของวงจรเข้ารหัส

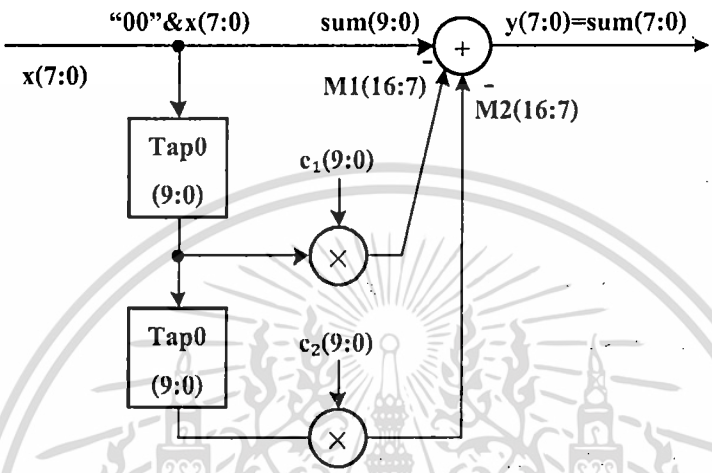
$$z(k) = f\{\hat{y}(k) - c_1\hat{y}(k-1) - c_2\hat{y}(k-2)\}$$

2. กำหนดเงื่อนไขเริ่มต้น $\hat{y}(-1) = 0, \hat{y}(-2) = 0$
3. สมการฟังก์ชันที่ทำให้เกิดการล้น

$f(\bullet)$ คือ $f(x) = (x+1) \bmod 2 - 1$

4. โดยกำหนดรูปแบบตัวเลขของอินพุตและเอาต์พุตคือ x.xxx xxxx เป็นการกำหนด fixed-point เพื่อทำให้เกิดการล้น
5. ค่าสัมประสิทธิ์จะถูกแทนด้วยรูปแบบตัวเลขคือ xxx.xxx xxxx โดยเพิ่ม 2 บิตหน้าตามค่าสัมประสิทธิ์ที่กำหนด
6. ในการออกแบบจริงจึงต้องมีการเพิ่มความยาวข้อมูลของอินพุตอีก 2บิตหน้าเพื่อใช้ในการประมวลผล และเอาต์พุตก็จะตัด 2 บิตหน้าทิ้งเช่นกัน

โครงสร้างที่จะใช้ในการออกแบบวงจรถอดรหัสโดยแสดงขนาดความยาวของข้อมูลโดยละเอียด ซึ่งในรูปนี้แสดงโครงสร้างของวงจรเข้ารหัสขนาด 8 บิตรูปที่ 3.21



รูปที่ 3.21 โครงสร้างแบบละเอียดภายในของวงจรถอดรหัสแบบ FIR 2nd order filter

หลังจากที่ Tap0 ที่มีจำนวน 10 บิต คูณ c_1 ที่มีจำนวน 10 บิตเช่นเดียวกัน จะได้ค่าผลคูณ (M1) ซึ่งมีจำนวน 20 บิต โดยจะเลือกใช้ 10 บิต คือบิตที่ 7-16 ของ M1 เนื่องจากเป็นค่าผลลัพธ์ที่ได้ตาม format ที่กำหนดไว้ ส่วนบิตที่เหลือไม่จำเป็นต้องใช้ในการคำนวณสามารถตัดทิ้งได้

ยกตัวอย่างการทดลองโดยกำหนดค่าสัมประสิทธิ์ของวงจรกรองสัญญาณดิจิตอลเป็น $c_1 = 4 = 100.0\ 0000$ และ $c_2 = -1 = 111.0\ 0000$ กำหนดค่า $\hat{y}(k) = y(k)$ ที่ตำแหน่งของ k ใดๆ และให้ $\hat{y}(-1) = 0, \hat{y}(-2) = 0$ โดยกำหนดให้เอาต์พุตมี format เป็น $x.xxx\ xxxx$ โดยให้บิตแรกเป็นบิตเครื่องหมาย ซึ่งทั้งการคำนวณค่าเอาต์พุต $z(k)$ ของวงจรกรองสัญญาณดิจิตอลที่จำลองการทำงานจากโปรแกรม MATLAB และ วงจรกรองสัญญาณดิจิตอลในทางปฏิบัตินั้นจะแสดงควบคู่กันไปเพื่อให้สามารถเห็นผลลัพธ์ได้อย่างชัดเจน แสดงดังในตารางที่ 3.2

ตารางที่ 3.2 ผลการคำนวณจากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรถอดรหัสแบบ FIR 2nd order filter

ฐานสิบ	ฐานสองแบบ 2's complement
$k=0;\ I(0)=\hat{y}(0)-c_1\ \hat{y}(-1)-c_2\ \hat{y}(-2)$ $=0.5-(4\times 0)-(-1\times 0)=0.5$ $\therefore z(0)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000 + ⇐0.5 000.000 0000 + ⇐0 000.000 0000 ⇐0 00[0.100 0000] ⇐0.5
$k=1;\ I(1)=\hat{y}(1)-c_1\ \hat{y}(0)-c_2\ \hat{y}(-1)$ $=0.5-(4\times 0.5)-(-1\times 0)=-1.5$ $\therefore z(1)=[(-1.5+1)\bmod 2]-1=0.5$	000.100 0000 + ⇐0.5 110.000 0000 + ⇐−2 000.000 0000 ⇐0 11[0.100 0000] ⇐0.5
$k=2;\ I(2)=\hat{y}(2)-c_1\ \hat{y}(1)-c_2\ \hat{y}(0)$ $=0-(4\times 0.5)-(-1\times 0.5)=-1.5$ $\therefore z(2)=[(-1.5+1)\bmod 2]-1=0.5$	000.000 0000 + ⇐0 110.000 0000 + ⇐−2 000.100 0000 ⇐0.5 11[0.100 0000] ⇐0.5
$k=3;\ I(3)=\hat{y}(3)-c_1\ \hat{y}(2)-c_2\ \hat{y}(1)$ $=0-(4\times 0)-(-1\times 0.5)=0.5$ $\therefore z(3)=[(0.5+1)\bmod 2]-1=0.5$	000.000 0000 + ⇐0 000.000 0000 + ⇐0 000.100 0000 ⇐0.5 00[0.100 0000] ⇐0.5
$k=4;\ I(4)=\hat{y}(4)-c_1\ \hat{y}(3)-c_2\ \hat{y}(2)$ $=0.5-(4\times 0)-(-1\times 0)=0.5$ $\therefore z(4)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000 + ⇐0.5 000.000 0000 + ⇐0 000.000 0000 ⇐0 00[0.100 0000] ⇐0.5
$k=5;\ I(5)=\hat{y}(5)-c_1\ \hat{y}(4)-c_2\ \hat{y}(3)$ $=0.5-(4\times 0.5)-(-1\times 0)=-1.5$ $\therefore z(5)=[(-1.5+1)\bmod 2]-1=0.5$	000.100 0000 + ⇐0.5 110.000 0000 + ⇐2 000.000 0000 ⇐0 11[0.100 0000] ⇐0.5

ตารางที่ 3.2 ผลการคำนวณจากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรถอดรหัสแบบ FIR 2nd order filter (ต่อ)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=6;\ I(6)=\hat{y}(6)-c_1\ \hat{y}(5)-c_2\ \hat{y}(4)$ $=0-(4\times0.5)-(-1\times0.5)=-1.5$ $\therefore z(6)=[(-1.5+1)\bmod 2]-1=0.5$	000.000 0000 + ⇐0 110.000 0000 + ⇐−2 000.100 0000 ⇐0.5 11[0.100 0000] ⇐0.5
$k=7;\ I(7)=\hat{y}(7)-c_1\ \hat{y}(6)-c_2\ \hat{y}(5)$ $=0-(4\times0)-(-1\times0.5)=0.5$ $\therefore z(7)=[(0.5+1)\bmod 2]-1=0.5$	000.000 0000 + ⇐0 000.000 0000 + ⇐0 000.100 0000 ⇐0.5 00[0.100 0000] ⇐0.5
$k=8;\ I(8)=\hat{y}(8)-c_1\ \hat{y}(7)-c_2\ \hat{y}(6)$ $=0.5-(4\times0)-(-1\times0)=0.5$ $\therefore z(8)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000 + ⇐0.5 000.000 0000 + ⇐0 000.000 0000 ⇐0 00[0.100 0000] ⇐0.5
$k=9;\ I(9)=\hat{y}(9)-c_1\ \hat{y}(8)-c_2\ \hat{y}(7)$ $=0.5-(4\times0.5)-(-1\times0)=-1.5$ $\therefore z(9)=[(-1.5+1)\bmod 2]-1=0.5$	000.100 0000 + ⇐0.5 110.000 0000 + ⇐−2 000.000 0000 ⇐0 11[0.100 0000] ⇐0.5

ตรวจสอบผลการคำนวณโดยการใช้ค่าที่คำนวณได้เทียบกับค่าที่ได้จากโปรแกรม MATLAB ซึ่งมีโค้ดของโปรแกรมดังรูปที่ 3.22

```
clear all;
clc;
% coeefficient Encoder
c1=4;
c2=-1;
% coeefficient Decoder
c3=4;
c4=-1;
% initial value
y(1)=0;
y(2)=0;
%%%% Encoder %%%%
for k=3:12;
    x(k)=0.5;
    I=x(k)+c1*y(k-1)+c2*y(k-2);
    y(k)=mod(I+1,2)-1;


end;
        disp('Y value'); disp(y);
%%%% Decoder %%%%
yr=y;
for k=3:12;
        J=yr(k)-(c3*yr(k-1) + c4*yr(k-2));
        z(k)=mod(J+1,2)-1;

end;
disp('Z Value'); disp(z);
```

รูปที่ 3.22 โค้ดของโปรแกรม MATLAB ที่ใช้ในการจำลองการทำงานของวงจรถอดรหัสแบบ FIR 2nd order filter

จากโปรแกรม MATLAB จะได้ผลลัพธ์ดังต่อไปนี้

Columns 1 through 7

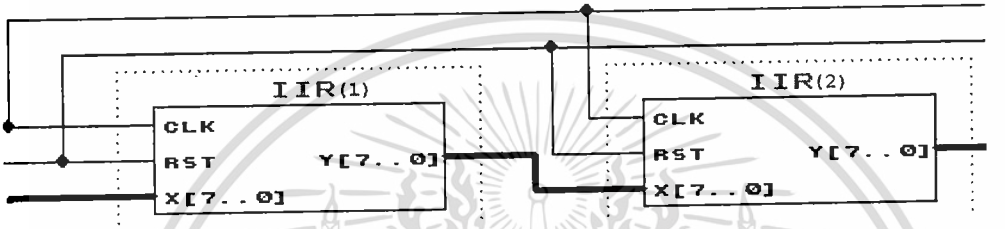
0 0 0.5 0.5 0.5 0.5 0.5

Columns 8 through 12

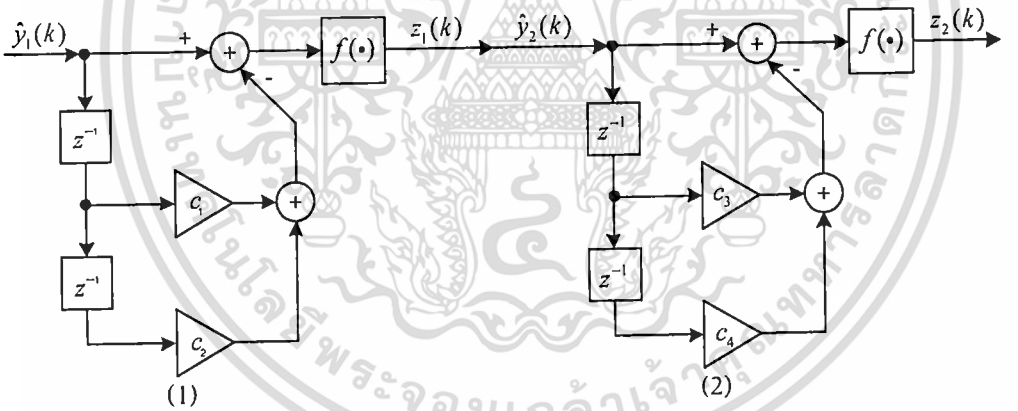
0.5 0.5 0.5 0.5 0.5

3.1.2.3 วงจรเข้ารหัสที่เป็นวงจรกรองสัญญาณดิจิตอลอันดับสี่ชนิดผลตอบสนองอิมพัลส์ไม่จำกัด (IIR 4th order filter)

วงจรเข้ารหัสแบบ IIR 4th order filter คือการนำวงจรเข้ารหัสแบบ IIR 2nd order filter สองตัวมาต่อกันแบบ Cascade ซึ่งมีโครงสร้างใน FPGA และโครงสร้างที่ใน Matlab ดังรูปที่ 3.23 และ 3.24



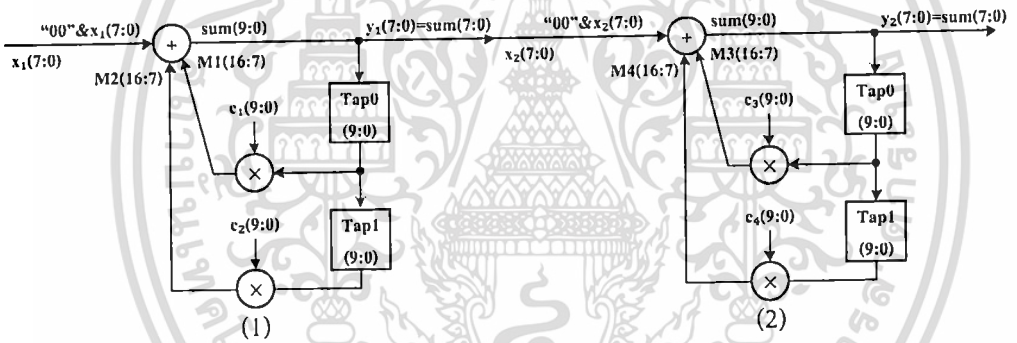
รูปที่ 3.23 วงจรเข้ารหัสแบบ IIR 4th order filter ใน FPGA



รูปที่ 3.24 โครงสร้างของวงจรเข้ารหัสแบบ IIR 4th order filter ที่ใช้ในการจำลองการทำงานโดย MATLAB

โดยลักษณะของการกำหนดเงื่อนไขในการจำลองการทำงานของวงจรเข้ารหัสแบบ IIR 4th order filter จะเหมือนกับวงจรเข้ารหัสแบบ IIR 2nd order filter แต่ทำซ้ำ 2 ครั้ง คือนำเอาต์พุตของวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (1) : $y_1(k)$ มาเป็นอินพุตของวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (2) : $x_2(k) = y_1(k)$ ดังนั้นเอาต์พุตของวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (2) : $y_2(k)$ จะเป็นข้อมูลที่ได้เข้ารหัสแบบ 4th order คือมีจำนวนของค่าสัมประสิทธิ์ถึง 4 ตัว (c_1, c_2, c_3, c_4)

โครงสร้างที่จะใช้ในการออกแบบวงจรเข้ารหัสแบบ IIR 4th order filter โดยแสดงขนาดความยาวของข้อมูลโดยละเอียด ดังรูปที่ 3.25 แสดงถึงโครงสร้างของวงจรเข้ารหัสขนาด 8 บิต



รูปที่ 3.25 โครงสร้างแบบละเอียดที่ใช้สร้างวงจรเข้ารหัส IIR 4th order filter

ซึ่งในการจำลองการทำงานจะกำหนดให้ค่าสัมประสิทธิ์ดังนี้ $c_1 = 4$, $c_2 = -1$, $c_3 = 6$ และ $c_4 = -2$ และกำหนดให้ $x_1(k) = 0.5$, $y_1(-1) = 0$ และ $y_1(-2) = 0$ โดยผลการคำนวณค่าเอาต์พุต $y_1(k)$ เปรียบเทียบระหว่างผลที่ได้จากการจำลองการทำงานด้วยโปรแกรม MATLAB และผลที่เกิดขึ้นจริงด้วยวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (1) ภายใน FPGA แสดงดังในตารางที่ 3.3

ตารางที่ 3.3 ผลการคำนวณค่า $y_1(k)$ จากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรเข้ารหัสแบบ IIR 4th order filter ในวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (1)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=0;\ I(0)=x_1(0)+c_1\ y_1(-1)+c_2\ y_1(-2)$ $=0.5+(4\times0)+(-1\times0)=0.5$ $\therefore y_1(0)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐0.5 000.000 0000+ ⇐0 000.000 0000 ⇐0 00[0.100 0000] ⇐0.5
$k=1;\ I(1)=x_1(1)+c_1\ y_1(0)+c_2\ y_1(-1)$ $=0.5+(4\times0.5)+(-1\times0)=-1.5$ $\therefore y_1(1)=[(-1.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐0.5 010.000 0000+ ⇐2 000.000 0000 ⇐0 00[0.100 0000] ⇐0.5
$k=2;\ I(2)=x_1(2)+c_1\ y_1(1)+c_2\ y_1(0)$ $=0.5+(4\times0.5)+(-1\times0.5)=2$ $\therefore y_1(2)=[(2+1)\bmod 2]-1=0$	000.100 0000+ ⇐0.5 010.000 0000+ ⇐2 111.100 0000 ⇐−0.5 11[0.000 0000] ⇐0
$k=3;\ I(3)=x_1(3)+c_1\ y_1(2)+c_2\ y_1(1)$ $=0.5+(4\times0)+(-1\times0.5)=0$ $\therefore y_1(3)=[(0+1)\bmod 2]-1=0$	000.100 0000+ ⇐0.5 000.000 0000+ ⇐0 111.100 0000 ⇐−0.5 00[0.000 0000] ⇐0
$k=4;\ I(4)=x_1(4)+c_1\ y_1(3)+c_2\ y_1(2)$ $=0.5+(4\times0)+(-1\times0)=0.5$ $\therefore y_1(4)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐0.5 000.000 0000+ ⇐0 000.000 0000 ⇐0 00[0.100 0000] ⇐0.5
$k=5;\ I(5)=x_1(5)+c_1\ y_1(4)+c_2\ y_1(3)$ $=0.5+(4\times0.5)+(-1\times0)=2.5$ $\therefore y_1(5)=[(2.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐0.5 010.000 0000+ ⇐2 000.000 0000 ⇐0 01[0.100 0000] ⇐0.5

ตารางที่ 3.3 ผลการคำนวณค่า $y_1(k)$ จากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรเข้ารหัสแบบ IIR 4th order filter ในวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (1) (ต่อ)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=6;\ I(6)=x_1(6)+c_1\ y_1(5)+c_2\ y_1(4)$ $=0.5+(4\times0.5)+(-1\times0.5)=2$ $\therefore y_1(6)=[(2+1)\bmod 2]-1=0$	000.100 0000+ $\Leftarrow 0.5$ 010.000 0000+ $\Leftarrow 2$ 111.100 0000 $\Leftarrow -0.5$ 11[0.000 0000] $\Leftarrow 0$
$k=7;\ I(7)=x_1(7)+c_1\ y_1(6)+c_2\ y_1(5)$ $=0.5+(4\times0)+(-1\times0.5)=0$ $\therefore y_1(7)=[(0+1)\bmod 2]-1=0$	000.100 0000+ $\Leftarrow 0.5$ 000.000 0000+ $\Leftarrow 0$ 111.100 0000 $\Leftarrow -0.5$ 11[0.000 0000] $\Leftarrow 0$
$k=8;\ I(8)=x_1(8)+c_1\ y_1(7)+c_2\ y_1(6)$ $=0.5+(4\times0)+(-1\times0)=0.5$ $\therefore y_1(8)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000+ $\Leftarrow 0.5$ 000.000 0000+ $\Leftarrow 0$ 000.000 0000 $\Leftarrow 0$ 00[0.100 0000] $\Leftarrow 0.5$
$k=9;\ I(9)=x_1(9)+c_1\ y_1(8)+c_2\ y_1(7)$ $=0.5+(4\times0.5)+(-1\times0)=2.5$ $\therefore y_1(9)=[(2.5+1)\bmod 2]-1=0.5$	000.100 0000+ $\Leftarrow 0.5$ 010.000 0000+ $\Leftarrow 2$ 000.000 0000 $\Leftarrow 0$ 01[0.100 0000] $\Leftarrow 0.5$

และเมื่อนำค่าเอาต์พุต $y_1(k)$ ของวงจรเข้ารหัสแบบ IIR 4th order filter จากวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (1) มาเป็นอินพุต $x_2(k)$ ของวงจรเข้ารหัสแบบ IIR 4th order filter จากวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (2) ก็ได้ค่าเอาต์พุต $y_2(k)$ ซึ่งเป็นข้อมูลที่เข้ารหัสด้วยวงจรเข้ารหัสแบบ IIR 4th order filter นั่นเอง $y_1(-2)=0$ จากนั้นนำผลการคำนวณค่าเอาต์พุต $y_2(k)$ เปรียบเทียบระหว่างผลที่ได้จากการจำลองการทำงานด้วยโปรแกรม MATLAB และผลที่เกิดขึ้นจริงด้วยวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (2) ภายใน FPGA แสดงดังในตารางที่ 3.4

ตารางที่ 3.4 ผลการคำนวณค่า $y_2(k)$ จากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรเข้ารหัสแบบ IIR 4th order filter ในวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (2)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=0;\ I(0)=x_2(0)+c_3y_2(-1)+c_4y_2(-2)$ $=0.5+(6\times0)+(-2\times0)=0.5$ $\therefore y_2(0)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000 + ⇐ 0.5 000.000 0000 + ⇐ 0 000.000 0000 ⇐ 0 00 0.100 0000 ⇐ 0.5
$k=1;\ I(1)=x_2(1)+c_3y_2(0)+c_4y_2(-1)$ $=0.5+(6\times0.5)+(-2\times0)=3.5$ $\therefore y_2(1)=[(2.5+1)\bmod 2]-1=-0.5$	000.100 0000 + ⇐ 0.5 011.000 0000 + ⇐ 3 000.000 0000 ⇐ 0 01 1.100 0000 ⇐ −0.5
$k=2;\ I(2)=x_2(2)+c_3y_2(1)+c_4y_2(0)$ $=0+(6\times0.5)+(-2\times0.5)=2$ $\therefore y_2(2)=[(2+1)\bmod 2]-1=0$	000.000 0000 + ⇐ 0 011.000 0000 + ⇐ 3 111.000 0000 ⇐ −1 01 0.000 0000 ⇐ 0
$k=3;\ I(3)=x_2(3)+c_3y_2(2)+c_4y_2(1)$ $=0+(6\times0)+(-2\times-0.5)=1$ $\therefore y_2(3)=[(1+1)\bmod 2]-1=-1$	000.000 0000 + ⇐ 0 000.000 0000 + ⇐ 0 001.000 0000 ⇐ 1 00 1.000 0000 ⇐ −1
$k=4;\ I(4)=x_2(4)+c_3y_2(3)+c_4y_2(2)$ $=0.5+(6\times-1)+(-2\times0)=-5.5$ $\therefore y_2(4)=[(-5.5+1)\bmod 2]-1=0.5$	000.100 0000 + ⇐ 0.5 010.000 0000 + ⇐ −6 000.000 0000 ⇐ 0 01 0.100 0000 ⇐ 0.5
$k=5;\ I(5)=x_2(5)+c_3y_2(4)+c_4y_2(3)$ $=0.5+(6\times0.5)+(-2\times0)=3.5$ $\therefore y_2(5)=[(3.5+1)\bmod 2]-1=-0.5$	000.100 0000 + ⇐ 0.5 011.000 0000 + ⇐ 3 000.000 0000 ⇐ 0 01 1.100 0000 ⇐ −0.5

ตารางที่ 3.4 ผลการคำนวณค่า $y_2(k)$ จากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรเข้ารหัสแบบ IIR 4th order filter ในวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (2) (ต่อ)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=6;\ I(6)=x_2(6)+c_3y_2(5)+c_4y_2(4)$ $=0+(6\times0.5)+(-2\times0.5)=2$ $\therefore y_2(6)=[(2+1)\bmod 2]-1=0$	000.000 0000+ $\Leftarrow 0$ 011.000 0000+ $\Leftarrow 3$ 111.000 0000 $\Leftarrow -1$ 01[0.000 0000] $\Leftarrow 0$
$k=7;\ I(7)=x_2(7)+c_3y_2(6)+c_4y_2(5)$ $=0+(6\times0)+(-2\times-0.5)=1$ $\therefore y_2(7)=[(1+1)\bmod 2]-1=-1$	000.000 0000+ $\Leftarrow 0$ 000.000 0000+ $\Leftarrow 0$ 001.000 0000 $\Leftarrow 1$ 00[1.000 0000] $\Leftarrow -1$
$k=8;\ I(8)=x_2(8)+c_3y_2(7)+c_4y_2(6)$ $=0.5+(6\times-1)+(-2\times0)=-5.5$ $\therefore y_2(8)=[(-5.5+1)\bmod 2]-1=0.5$	000.100 0000+ $\Leftarrow 0.5$ 010.000 0000+ $\Leftarrow -6$ 000.000 0000 $\Leftarrow 0$ 01[0.100 0000] $\Leftarrow 0.5$
$k=9;\ I(9)=x_2(9)+c_3y_2(8)+c_4y_2(7)$ $=0.5+(6\times0.5)+(-2\times-1)=3.5$ $\therefore y_2(9)=[(3.5+1)\bmod 2]-1=-0.5$	000.100 0000+ $\Leftarrow 0.5$ 011.000 0000+ $\Leftarrow 3$ 001.000 0000 $\Leftarrow 1$ 10[1.100 0000] $\Leftarrow -0.5$

ตรวจสอบผลการคำนวณโดยการใช้ค่าที่คำนวณได้เทียบกับค่าที่ได้จากโปรแกรม MATLAB ซึ่งมีโค้ดของโปรแกรมดังรูปที่ 3.26 และกำหนดให้ค่าเงื่อนไขเริ่มต้นอยู่ที่ตำแหน่ง $y(-2)\Rightarrow y(1)=0$ และ $y(-1)\Rightarrow y(2)=0$ ดังนั้นจึงเริ่มคำนวณที่ $y(0)=y(3)$ ขึ้นไป

```
clear all;
clc;
%%%%%%%%%%%%%%%%% Encoder1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% coeefficient Encoder
c1=4;
c2=-1;
% initial value
y(1)=0;
y(2)=0;

for k=3:12;
    x(k)=0.5;
    I=x(k)+c1*y(k-1)+c2*y(k-2);
    y(k)=mod(I+1,2)-1;

end;
disp('Encoder 2nd Order Value'); disp(y);


%%%%%%%%%%%%%%%%%%% Encoder2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c5=6;          % C5 of IIR filter coefficient (Encoder)
c6=-2;         % C6 of IIR filter coefficient (Encoder)
ye(1)=0;
ye(2)=0;
xe=y ;

for k=3:12;
    Ie=xe(k)+c5*ye(k-1)+c6*ye(k-2);
    ye(k)=mod(Ie+1,2)-1;
end;
disp('Encoder 4th Order Value'); disp(ye);
```

รูปที่ 3.26 โค้ดของโปรแกรม MATLAB ที่ใช้ในการจำลองการทำงานของวงจรเข้ารหัสแบบ IIR 4th order filter

ผลการคำนวณจากโปรแกรมได้ดังนี้

$y_1(k)$ [ใน code คือ $y(k)$]:

Columns 1 through 12

0 0 0.5 0.5 0 0 0.5 0.5 0 0 0.5 0.5

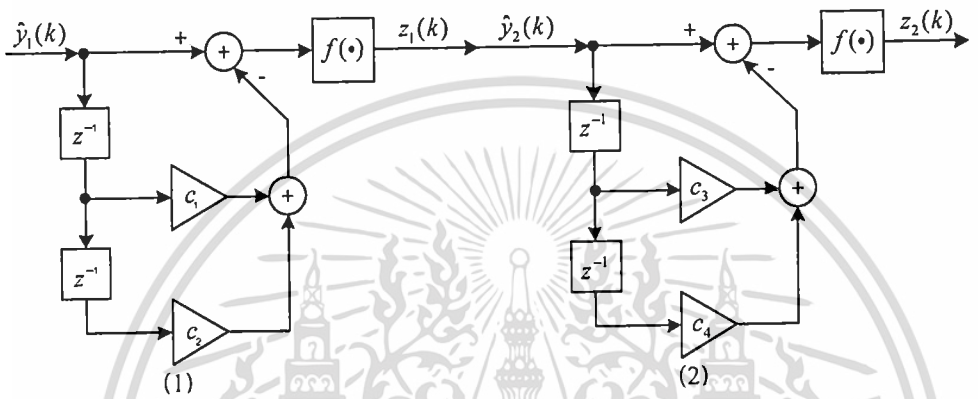
$y_2(k)$ [ใน code คือ $ye(k)$]:

Columns 1 through 12

0 0 0.5 -0.5 0 -1 0.5 -0.5 0 -1 0.5 -0.5

3.1.2.4 วงจรถอดรหัสที่เป็นวงจรกรองสัญญาณดิจิตอลอันดับสี่ชนิดผลตอบสนองอิมพัลส์จำกัด (FIR 4th order filter)

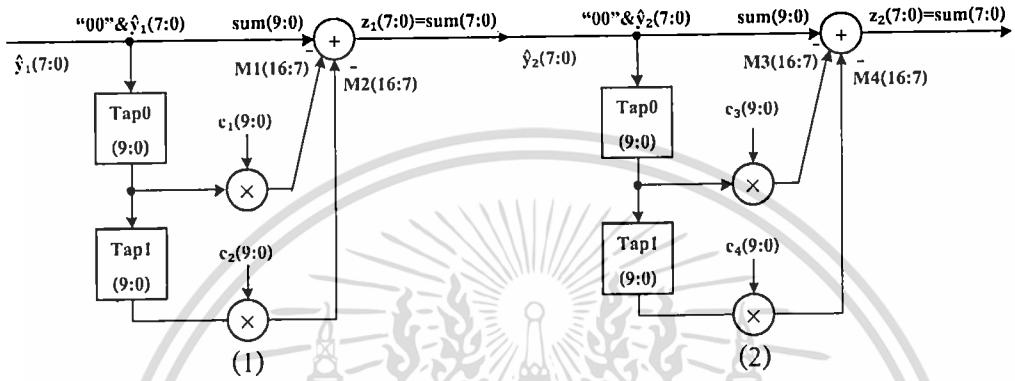
วงจรถอดรหัสแบบ FIR 4th order filter คือการนำวงจรถอดรหัสแบบ FIR 2nd order filter สองตัวมาต่อกันแบบ Cascade ดังรูปที่ 3.27



รูปที่ 3.27 โครงสร้างที่ใช้สร้างวงจรถอดรหัสแบบ FIR 4th order filter ที่ใช้ในการจำลองการทำงานจากโดย MATLAB

โดยลักษณะของการกำหนดเงื่อนไขในการจำลองการทำงานของวงจรถอดรหัสแบบ FIR 4th order filter จะเหมือนกับวงจรถอดรหัสแบบ FIR 2nd order filter แต่ทำซ้ำ 2 ครั้ง คือนำเอาต์พุตของวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (1) : $z_1(k)$ มาเป็นอินพุตของวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (2) : $x_2(k) = y_1(k)$ จะได้เอาต์พุตของวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (2) : $z_2(k)$ ออกมา

โครงสร้างที่จะใช้ในการออกแบบวงจรถอดรหัสแบบ FIR 4th order filter โดยแสดงขนาดความยาวของข้อมูลโดยละเอียด โดยในรูปที่ 3.28 ได้แสดงโครงสร้างของวงจรเข้ารหัสขนาด 8 บิต



รูปที่ 3.28 โครงสร้างแบบละเอียดที่ใช้สร้างวงจรถอดรหัสแบบ FIR 4th order filter

ซึ่งในการจำลองการทำงานจะกำหนดให้ค่าสัมประสิทธิ์ดังนี้ $c_1 = 4$, $c_2 = -1$, $c_3 = 6$ และ $c_4 = -2$ และกำหนดให้ $\hat{y}_1(-2) = 0$, $\hat{y}_1(-1) = 0$ และ อินพุตคือ $\hat{y}_1(k) = y_2(k)$ (นำเอาต์พุตของวงจรเข้ารหัสแบบ IIR 4th order filter ข้างต้นมาถอดรหัส) โดยผลการคำนวณค่าเอาต์พุต $z_1(k)$ เปรียบเทียบระหว่างผลที่ได้จากการจำลองการทำงานด้วยโปรแกรม MATLAB และผลที่เกิดขึ้นจริงด้วยวงจรเข้ารหัสแบบ FIR 2nd order filter ตัวที่ (1) ภายใน FPGA แสดงดังในตารางที่ 3.5

ตารางที่ 3.5 ผลการคำนวณค่า $z_1(k)$ จากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรถอดรหัสแบบ FIR 4th order filter ในวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (1)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=0;\ I(0) = \hat{y}_1(0) - c_1\ \hat{y}_1(-1) - c_2\ \hat{y}_1(-2)$ $= 0.5 - (6\times 0) - (-2\times 0) = 0.5$ $\therefore z_1(0) = [(0.5+1) \bmod 2] - 1 = 0.5$	000.100 0000 + $\Leftarrow 0.5$ 000.000 0000 + $\Leftarrow 0$ 000.000 0000 $\Leftarrow 0$ 00 0.100 0000 $\Leftarrow 0.5$

ตารางที่ 3.5 ผลการคำนวณค่า $z_1(k)$ จากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรถอดรหัสแบบ FIR 4th order filter ในวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (1) (ต่อ)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=1;\ I(1)=\hat{y}_1(1)-c_1\ \hat{y}_1(0)-c_2\ \hat{y}_1(-1)$ $=-0.5-(6\times0.5)-(-2\times0)=-3.5$ $\therefore z_1(1)=[(-3.5+1)\bmod 2]-1=0.5$	111.100 0000+ $\Leftarrow -0.5$ 101.000 0000+ $\Leftarrow -3$ 000.000 0000 $\Leftarrow 0$ 10 0.100 0000 $\Leftarrow 0.5$
$k=2;\ I(2)=\hat{y}_1(2)-c_1\ \hat{y}_1(1)-c_2\ \hat{y}_1(0)$ $=0-(6\times-0.5)-(-2\times0.5)=4$ $\therefore z_1(2)=[(4+1)\bmod 2]-1=0$	000.000 0000+ $\Leftarrow 0$ 011.000 0000+ $\Leftarrow 3$ 001.000 0000 $\Leftarrow 1$ 10 0.000 0000 $\Leftarrow 0$
$k=3;\ I(3)=\hat{y}_1(3)-c_1\ \hat{y}_1(2)-c_2\ \hat{y}_1(1)$ $=-1-(6\times0)-(-2\times-0.5)=-2$ $\therefore z_1(3)=[(-2+1)\bmod 2]-1=0$	111.000 0000+ $\Leftarrow -1$ 000.000 0000+ $\Leftarrow 0$ 111.000 0000 $\Leftarrow -1$ 11 0.000 0000 $\Leftarrow 0$
$k=4;\ I(4)=\hat{y}_1(4)-c_1\ \hat{y}_1(3)-c_2\ \hat{y}_1(2)$ $=0.5-(6\times-1)-(-2\times0)=6.5$ $\therefore z_1(4)=[(6.5+1)\bmod 2]-1=0.5$	000.100 0000+ $\Leftarrow 0.5$ 110.000 0000+ $\Leftarrow 6$ 000.000 0000 $\Leftarrow 0$ 11 0.100 0000 $\Leftarrow 0.5$
$k=5;\ I(5)=\hat{y}_1(5)-c_1\ \hat{y}_1(4)-c_2\ \hat{y}_1(3)$ $=-0.5-(6\times0.5)-(-2\times0)=-3.5$ $\therefore z_1(5)=[(-3.5+1)\bmod 2]-1=0.5$	111.100 0000+ $\Leftarrow -0.5$ 101.000 0000+ $\Leftarrow -3$ 000.000 0000 $\Leftarrow 0$ 10 0.100 0000 $\Leftarrow 0.5$
$k=6;\ I(6)=\hat{y}_1(6)-c_1\ \hat{y}_1(5)-c_2\ \hat{y}_1(4)$ $=0-(6\times-0.5)-(-2\times0.5)=4$ $\therefore z_1(6)=[(4+1)\bmod 2]-1=0$	000.000 0000+ $\Leftarrow 0$ 011.000 0000+ $\Leftarrow 3$ 001.000 0000 $\Leftarrow 1$ 10 0.000 0000 $\Leftarrow 0$

ตารางที่ 3.5 ผลการคำนวณค่า $z_1(k)$ จากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรถอดรหัสแบบ FIR 4th order filter ในวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (1) (ต่อ)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=7;\ I(7)=\hat{y}_1(7)-c_1\ \hat{y}_1(6)-c_2\ \hat{y}_1(5)$ $=-1-(6\times 0)-(-2\times -0.5)=-2$ $\therefore z_1(7)=[(-2+1)\bmod 2]-1=0$	111.000 0000+ $\Leftarrow -1$ 000.000 0000+ $\Leftarrow 0$ 111.000 0000 $\Leftarrow -1$ 11[0.000 0000] $\Leftarrow 0$
$k=8;\ I(8)=\hat{y}_1(8)-c_1\ \hat{y}_1(7)-c_2\ \hat{y}_1(6)$ $=0.5-(6\times -1)-(-2\times 0)=6.5$ $\therefore z_1(8)=[(6.5+1)\bmod 2]-1=0.5$	000.100 0000+ $\Leftarrow 0.5$ 110.000 0000+ $\Leftarrow 6$ 000.000 0000 $\Leftarrow 0$ 00[0.100 0000] $\Leftarrow 0.5$
$k=9;\ I(9)=\hat{y}_1(9)-c_1\ \hat{y}_1(8)-c_2\ \hat{y}_1(7)$ $=-0.5-(6\times 0.5)-(-2\times -1)=-5.5$ $\therefore z_1(9)=[(-5.5+1)\bmod 2]-1=0.5$	111.100 0000+ $\Leftarrow 0.5$ 101.000 0000+ $\Leftarrow -3$ 110.000 0000 $\Leftarrow -2$ 01[0.100 0000] $\Leftarrow 0.5$

และเมื่อนำค่าเอาต์พุต $z_1(k)$ จากวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (1) มาเป็นอินพุต $\hat{y}_2(k)$ ของวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (2) $\{z_1(k)=\hat{y}_2(k)\}$ ซึ่งจะเห็นได้ว่าค่าเอาต์พุต $z_2(k)$ ที่ได้มานั้นเท่ากับค่าอินพุต $x_1(k)$ ที่ใส่เข้าไปนั่นเอง จากนั้นคำนวณค่าเอาต์พุต $z_2(k)$ เปรียบเทียบระหว่างผลที่ได้จากการจำลองการทำงานด้วยโปรแกรม MATLAB และผลที่เกิดขึ้นจริงด้วยวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (2) ภายใน FPGA แสดงดังในตารางที่ 3.6

ตารางที่ 3.6 ผลการคำนวณค่า $z_2(k)$ จากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรถอดรหัสแบบ FIR 4th order filter ในวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (2)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=0;\ I(0)=\hat{y}_2(0)-c_3\hat{y}_2(-1)-c_4\hat{y}_2(-2)$ $=0.5-(4\times0)-(-1\times0)=0.5$ $\therefore z_2(0)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐0.5 000.000 0000+ ⇐0 000.000 0000 ⇐0 00[0.100 0000] ⇐0.5
$k=1;\ I(1)=\hat{y}_2(1)-c_3\hat{y}_2(0)-c_4\hat{y}_2(-1)$ $=0.5-(4\times0.5)-(-1\times0)=-1.5$ $\therefore z_2(1)=[(-1.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐−0.5 110.000 0000+ ⇐−2 000.000 0000 ⇐0 11[0.100 0000] ⇐0.5
$k=2;\ I(2)=\hat{y}_2(2)-c_3\hat{y}_2(1)-c_4\hat{y}_2(0)$ $=0-(4\times0.5)-(-1\times0.5)=-1.5$ $\therefore z_2(2)=[(-1.5+1)\bmod 2]-1=0.5$	000.000 0000+ ⇐0 110.000 0000+ ⇐−2 000.100 0000 ⇐1 11[0.100 0000] ⇐0.5
$k=3;\ I(3)=\hat{y}_2(3)-c_3\hat{y}_2(2)-c_4\hat{y}_2(1)$ $=0-(4\times0)-(-1\times0.5)=0.5$ $\therefore z_2(3)=[(0.5+1)\bmod 2]-1=0.5$	000.000 0000+ ⇐0 000.000 0000+ ⇐0 000.100 0000 ⇐0.5 00[0.100 0000] ⇐0.5
$k=4;\ I(4)=\hat{y}_2(4)-c_3\hat{y}_2(3)-c_4\hat{y}_2(2)$ $=0.5-(4\times0)-(-1\times0)=0.5$ $\therefore z_2(4)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐0.5 000.000 0000+ ⇐0 000.000 0000 ⇐0 00[0.100 0000] ⇐0.5
$k=5;\ I(5)=\hat{y}_2(5)-c_3\hat{y}_2(4)-c_4\hat{y}_2(3)$ $=0.5-(4\times0.5)-(-1\times0)=-1.5$ $\therefore z_2(5)=[(-1.5+1)\bmod 2]-1=0.5$	000.100 0000+ ⇐0.5 110.000 0000+ ⇐−2 000.000 0000 ⇐0 11[0.100 0000] ⇐0.5

ตารางที่ 3.6 ผลการคำนวณค่า $z_2(k)$ จากโปรแกรม MATLAB เทียบกับการคำนวณในระบบเลขฐานสองของวงจรถอดรหัสแบบ FIR 4th order filter ในวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (2) (ต่อ)

ฐานสิบ	ฐานสองแบบ 2's complement
$k=6;\ I(6)=\hat{y}_2(6)-c\,\hat{y}_2(5)-c_4\hat{y}_2(4)$ $=0-(4\times0.5)-(-1\times0.5)=-1.5$ $\therefore z_2(6)=[(-1.5+1)\bmod 2]-1=0.5$	000.000 0000+ $\Leftarrow 0$ 110.000 0000+ $\Leftarrow -2$ 000.100 0000 $\Leftarrow 0.5$ 11[0.100 0000] $\Leftarrow 0.5$
$k=7;\ I(7)=\hat{y}_2(7)-c_3\hat{y}_2(6)-c_4\hat{y}_2(5)$ $=0-(4\times0)-(-1\times0.5)=0.5$ $\therefore z_2(7)=[(0.5+1)\bmod 2]-1=0.5$	000.000 0000+ $\Leftarrow 0$ 000.000 0000+ $\Leftarrow 0$ 000.100 0000 $\Leftarrow 0.5$ 00[0.100 0000] $\Leftarrow 0.5$
$k=8;\ I(8)=\hat{y}_2(8)-c_3\hat{y}_2(7)-c_4\hat{y}_2(6)$ $=0.5-(4\times0)-(-1\times0)=0.5$ $\therefore z_2(8)=[(0.5+1)\bmod 2]-1=0.5$	000.100 0000+ $\Leftarrow 0.5$ 000.000 0000+ $\Leftarrow 0$ 000.000 0000 $\Leftarrow 0$ 00[0.100 0000] $\Leftarrow 0.5$
$k=9;\ I(9)=\hat{y}_1(9)-c_1\ \hat{y}_1(8)-c_2\ \hat{y}_1(7)$ $=0.5-(4\times0.5)-(-1\times0)=-1.5$ $\therefore z_1(9)=[(-1.5+1)\bmod 2]-1=0.5$	000.100 0000+ $\Leftarrow 0.5$ 110.000 0000+ $\Leftarrow -2$ 000.000 0000 $\Leftarrow 0$ 11[0.100 0000] $\Leftarrow 0.5$

ตรวจสอบผลการคำนวณโดยการใช้ค่าที่คำนวณได้เทียบกับค่าที่ได้จากโปรแกรม MATLAB ซึ่งมีโค้ดของโปรแกรมดังรูปที่ 3.29

```
%%%%%%%%%%%%%%%% Decoder1 %%%%%%%%%%%%%%%%
c7=6;          % C1 of FIR filter
coefficient (Decoder)
c8=-2;         % C2 of FIR filter
coefficient (Decoder)
yd=ye;

for k=3:12;
    Jd=yd(k)-(c7*yd(k-1) + c8*yd(k-2));
    zd(k)=mod(Jd+1,2)-1;
end;

disp('Decoded 2nd Order Value'); disp(zd);

%%%%%%%%%%%%%%%% Decoder2 %%%%%%%%%%%%%%%%
c3=4;          % C1 of FIR filter
coefficient (Decoder)
c4=-1;         % C2 of FIR filter
coefficient (Decoder)
yr=zd;
for k=3:12;
    J=yr(k)-(c3*yr(k-1) + c4*yr(k-2));
    z(k)=mod(J+1,2)-1;
end;

disp('Decoded 4th Order Value'); disp(z);
```

รูปที่ 3.29 โค้ดของโปรแกรม MATLAB ที่ใช้ในการจำลองการทำงานของวงจรถอดรหัสแบบ FIR 4th order filter

จากโปรแกรม MATLAB จะได้ผลลัพธ์ดังต่อไปนี้

$z_1(k)$ [ใน code คือ $zd(k)$]:

Columns 1 through 12

0 0 0.5 0.5 0 0 0.5 0.5 0 0 0.5 0.5

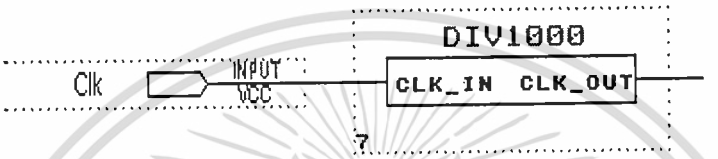
$z_2(k)$ [ใน code คือ $z(k)$]:

Columns 1 through 12

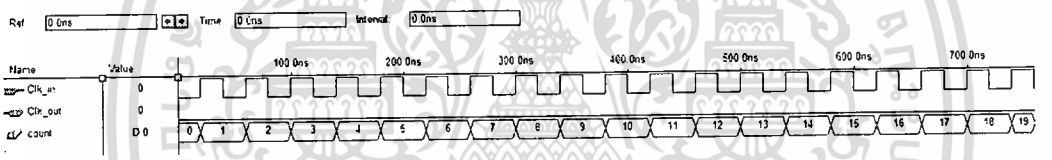
0 0 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5

3.1.2.5 วงจรหารความถี่

สัญญาณนาฬิกาในอุปกรณ์ FPGA ที่ใช้ในการทดลองสามารถสร้างความถี่ที่ 9.6 MHz อัตราการส่งของข้อมูลที่ออกจากคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมอยู่ที่ 9600 Hz ดังนั้นจะต้องสร้างวงจรหารความถี่ให้เหลืออยู่ที่ 9600 Hz เพื่อให้การรับส่งข้อมูลอนุกรมเป็นไปอย่างถูกต้อง โครงสร้างจะเป็นดังรูปที่ 3.30 และมีผลจำลองการทำงานดังรูปที่ 3.31

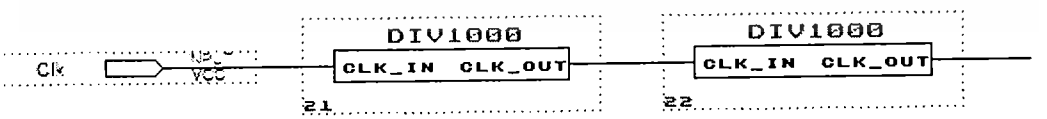


รูปที่ 3.30 วงจรหารความถี่ 9.6 kHz



รูปที่ 3.31 ผลการจำลองการทำงานของวงจรหารความถี่ 9.6 kHz

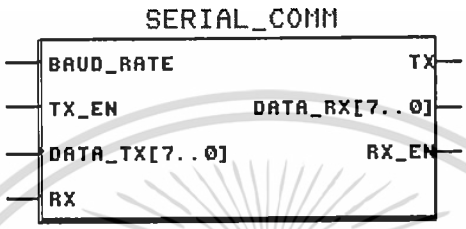
จากบล็อกไดอะแกรมการเข้ารหัส-ถอดรหัสผ่านการรับส่งข้อมูลแบบขนานด้วย FPGA รูปที่ 1.1 ซึ่งส่งข้อมูลจาก DIP switch และแสดงผลด้วย LED จำเป็นจะต้องหารความถี่ของสัญญาณให้เหลือ 9.6 Hz เพื่อให้สามารถมองเห็นแสงจาก LED ได้ทัน ซึ่งมีโครงสร้างดังรูปที่ 3.32



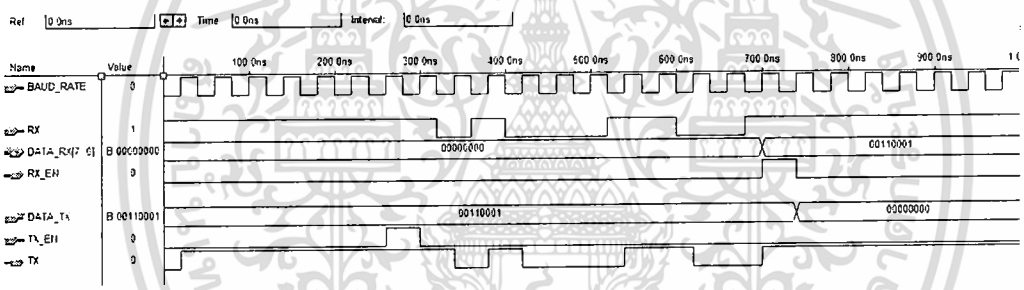
รูปที่ 3.32 วงจรหารความถี่ 9.6 Hz

3.1.2.6 วงจรสื่อสารอนุกรม

เป็นวงจรที่กำหนดคุณสมบัติและเงื่อนไขในการสื่อสารแบบอนุกรม ระหว่าง FPGA กับคอมพิวเตอร์ เช่นกำหนดอัตรา Baud Rate , ขนาดของข้อมูล , Start Bit , Stop Bit , Parity Bit เป็นต้น มีโครงสร้างดังรูปที่ 3.33 และมีผลการจำลองการทำงานดังรูปที่ 3.34



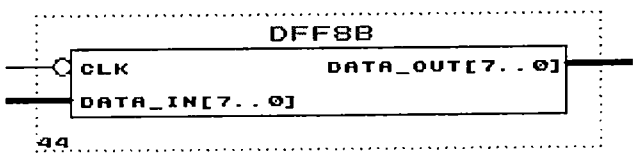
รูปที่ 3.33 วงจรสื่อสารอนุกรม



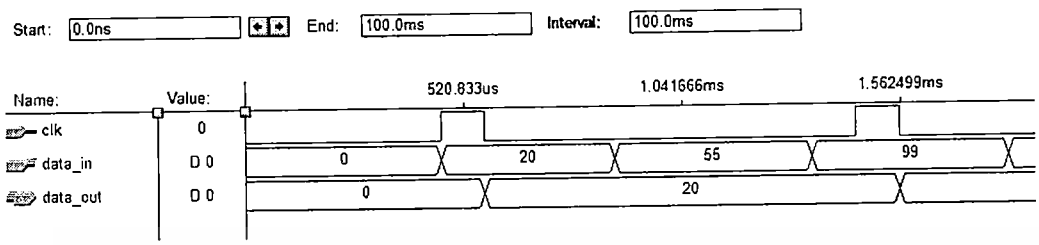
รูปที่ 3.34 ผลการจำลองการทำงานของวงจรสื่อสารอนุกรม

3.1.2.7 วงจร Latch

เป็นวงจรใช้ในการคงค่าของข้อมูลขนาด 8 บิตในฝั่งรับข้อมูลของวงจรถอดรหัสก่อนที่จะส่งข้อมูลไปยังคอมพิวเตอร์ มีโครงสร้างดังรูปที่ 3.35 และมีผลการจำลองการทำงานดังรูปที่ 3.36



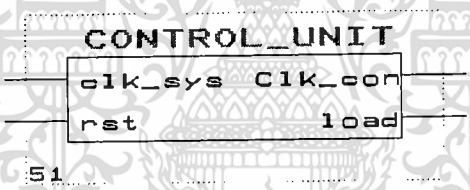
รูปที่ 3.35 วงจร Latch ข้อมูลขนาด 8 บิต



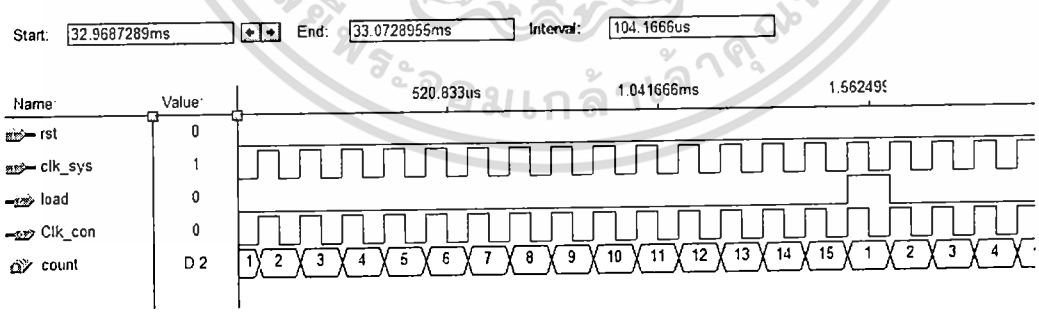
รูปที่ 3.36 ผลการจำลองการทำงานของวงจร Latch ข้อมูลขนาด 8 บิต

3.1.2.8 วงจรควบคุมสัญญาณ

เป็นวงจรที่ใช้ในการควบคุมสัญญาณในการส่งออกข้อมูลแบบอนุกรมของฝั่งเข้ารหัสโดยอาศัยการนับสัญญาณนาฬิกาจากวงจรหารความถี่ มีโครงสร้างดังรูปที่ 3.37 และมีผลการจำลองการทำงานดังรูปที่ 3.38



รูปที่ 3.37 วงจรควบคุมสัญญาณ



รูปที่ 3.38 ผลการจำลองการทำงานของวงจรควบคุมสัญญาณ

3.2 เครื่องมือที่ใช้ในการทดลอง

ในการศึกษาเรื่องการออกแบบวงจรเข้ารหัส – ถอดรหัสที่อาศัยพฤติกรรมการเกิดเคออสในวงจรกรองสัญญาณดิจิตอลและการประยุกต์ใช้งานด้านสื่อสาร จำเป็นต้องใช้เครื่องมือหลักๆ ในการทดลองได้แก่

3.2.1 อุปกรณ์ USB-RS232 converter

มีส่วนประกอบหลักคือ IC USB Controller ซึ่งในการทดลองใช้ IC FT232BM มีหน้าที่ในการติดต่อระหว่างคอมพิวเตอร์กับบอร์ด FPGA

3.2.2 FPGA

ในการทดลองจะใช้ FPGA เบอร์ EPF10K10LC84-4 ซึ่งเป็น FPGA ตระกูล FLEX10K ของบริษัท ALTERA สามารถต่อใช้งานกับวงจรภายนอกได้ 66 ขา มีความจุ 10000 เกต, Logic elements 576 Les, Embedded array blocks EABs เป็น FPGA

3.2.3 คอมพิวเตอร์

เครื่องคอมพิวเตอร์ PC หรือ คอมพิวเตอร์ Laptop ที่มีระบบปฏิบัติการ Window ซึ่งสามารถลงไดร์ฟเวอร์พอร์ตอนุกรมเสมือน และสามารถใช้งานโปรแกรมแอพพลิเคชั่นที่เชื่อมต่อกับพอร์ตอนุกรมได้

3.2.4 DIP Switch

จากบล็อกไดอะแกรมการเข้ารหัส-ถอดรหัสผ่านการรับส่งข้อมูลแบบขนานด้วย FPGA รูปที่ 1.1 จะต้องใช้ DIP switch ขนาด 8 บิต เพื่อเป็นอินพุตให้กับวงจรเข้ารหัส

3.2.5 LED

จากบล็อกไดอะแกรมการเข้ารหัส-ถอดรหัสผ่านการรับส่งข้อมูลแบบขนานด้วย FPGA รูปที่ 1.1 จะใช้หลอดไฟ LED1 ในการแสดงผลจากวงจรเข้ารหัสและจะใช้หลอดไฟ LED2 ในการแสดงผลจากวงจรถอดรหัส

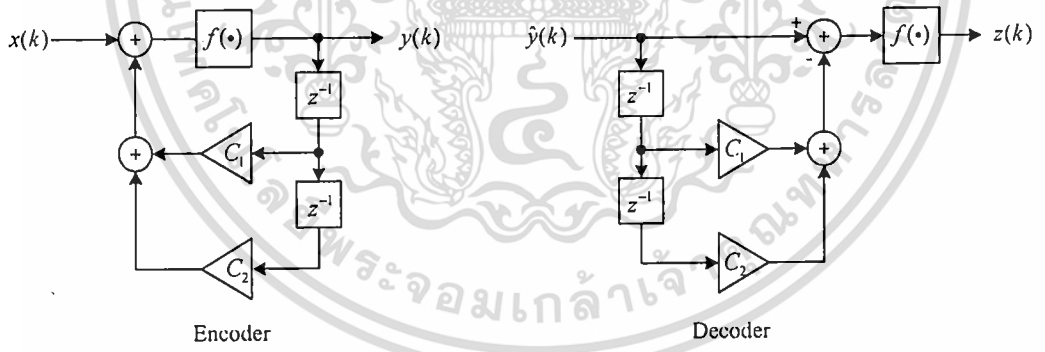
บทที่ 4

ผลการทดลอง

บทนี้แสดงผลการทดลองจากการออกแบบวงจรเข้ารหัสและถอดรหัส โดยใช้วงจรสัญญาณดิจิตอลอันดับสองและอันดับสี่ซึ่งผลการทดลองมีทั้งที่ได้จากการจำลองการทำงานด้วยโปรแกรม MATLAB และผลที่ได้จากการทดลองจริงด้วยฮาร์ดแวร์ที่สร้างขึ้นในทางปฏิบัติ

4.1 ผลการจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสอง (2^{nd} order filter) ด้วยโปรแกรม MATLAB

โดยโครงสร้างของวงจรเข้ารหัสและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสองที่ใช้ในการจำลองการทำงานแสดงได้ดังรูปที่ 4.1



รูปที่ 4.1 โครงสร้างของวงจรเข้ารหัสและถอดรหัสแบบ 2^{nd} order filter ที่ใช้ในการจำลองการทำงาน

จากโครงสร้างของวงจรเข้ารหัสและถอดรหัสแบบ 2^{nd} order filterในรูปที่ 4.1 สามารถเขียนเป็นสมการ
ผลต่างสืบเนื่องได้ดังนี้

สมการผลต่างสืบเนื่องของวงจรเข้ารหัส : $y(k)=f\{x(k)+C_1y(k-1)+C_2y(k-2)\}$

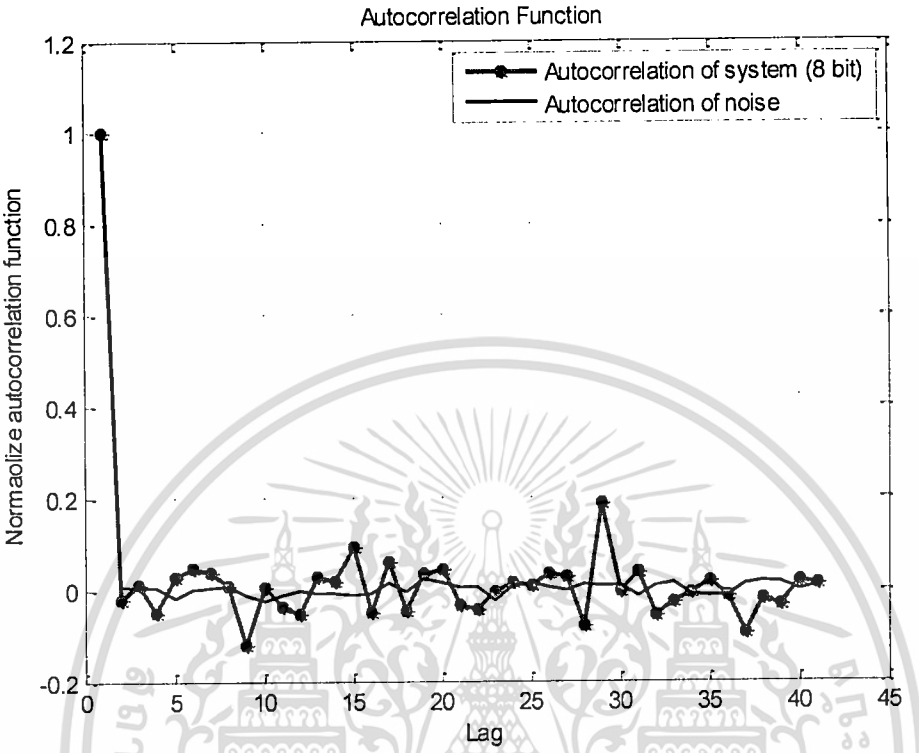
สมการผลต่างสืบเนื่องของวงจรถอดรหัส : $z(k)=f\{\hat{y}(k)-C_1\hat{y}(k-1)-C_2\hat{y}(k-2)\}$

ฟังก์ชั่น $f(\bullet)$: $f(x)=(x+1)\bmod 2-1$

เพื่อให้เข้าใจการทำงานของวงจรกรองสัญญาณดิจิตอลที่ใช้เป็นวงจรเข้ารหัสและถอดรหัส จึงต้องมีการวิเคราะห์วงจรเข้ารหัสและถอดรหัส ซึ่งในปริญญานิพนธ์นี้ได้ทำการวิเคราะห์พฤติกรรมของวงจรเข้ารหัสและถอดรหัสแบบไม่สนใจปัจจัยภายนอก โดยใช้คุณสมบัติของระบบที่ไม่เป็นเชิงเส้น (Nonlinear system) คืออินพุตเท่ากับศูนย์แต่เอาต์พุตไม่เท่ากับศูนย์ (Zero in non zero out) แต่เนื่องจากอินพุตของระบบเท่ากับศูนย์ (Zero input) จึงต้องกำหนดค่าเริ่มต้น (Initial value) ให้กับระบบด้วย

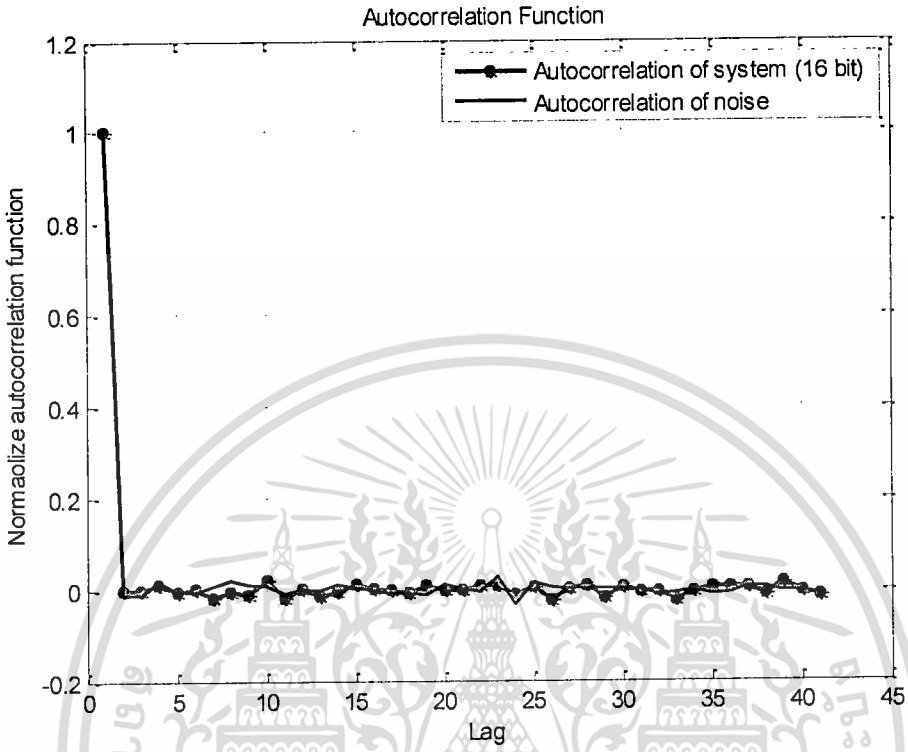
4.1.1 ผลเมื่ออินพุตเป็นศูนย์

เมื่อระบบมีอินพุตเป็นศูนย์ ลักษณะของค่าอัตสหสัมพันธ์ของผลตอบสนองของระบบจะมีลักษณะคล้ายกับค่าอัตสหสัมพันธ์ของสัญญาณรบกวน โดยการวิเคราะห์ฟังก์ชันค่าอัตสหสัมพันธ์ของเอาต์พุตของวงจรเข้ารหัส กำหนดให้ค่าเริ่มต้นให้ $y(1)=0.6135$, $y(2)=0$ โดยกำหนดให้ระบบมีขนาดของข้อมูลอินพุตเป็น 8 บิต และใช้จำนวนจุดในการทดลอง 10,000 จุดจะได้ผลการทดลองดังรูปที่ 4.2



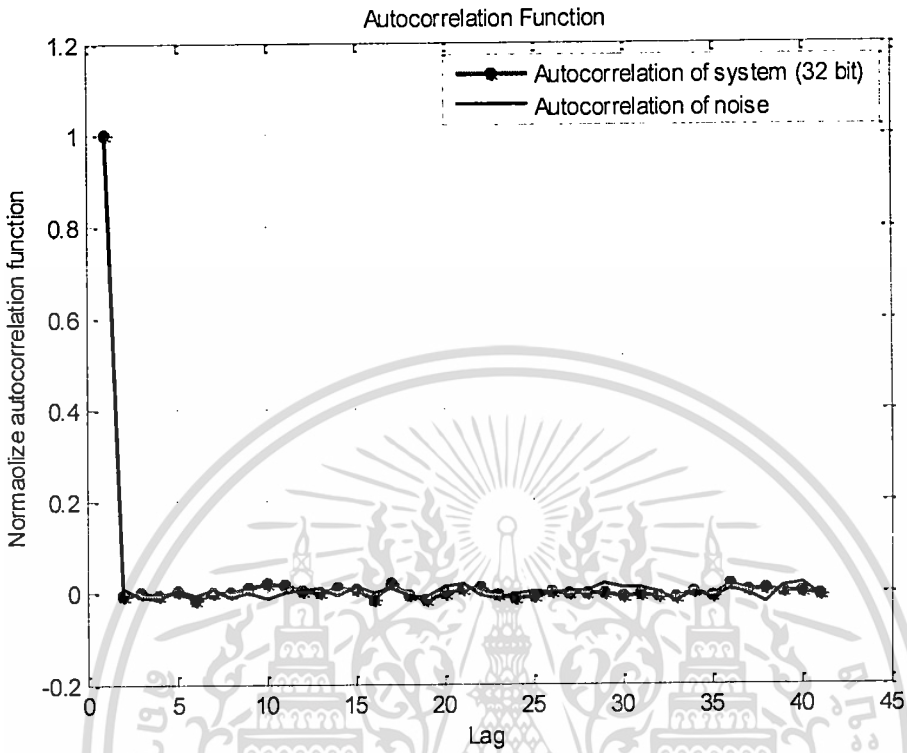
รูปที่ 4.2 กราฟค่าอัตสหสัมพันธ์ของเอาต์พุตของวงจรเข้ารหัสขนาด 8 บิต เทียบกับค่าอัตสหสัมพันธ์ของสัญญาณรบกวน

การวิเคราะห์ฟังก์ชันค่าอัตสหสัมพันธ์ของเอาต์พุตของวงจรเข้ารหัส ที่กำหนดให้ระบบมีขนาดของข้อมูลเป็น 16 บิต จะได้ผลการทดลองดังรูปที่ 4.3 โดยกำหนดให้ค่าเริ่มต้น $y(1)=0.6135$, $y(2)=0$ และใช้จำนวนจุดในการทดลอง 10,000 จุด



รูปที่ 4.3 กราฟค่าอัตสหสัมพันธ์ของเอาต์พุตของวงจรเข้ารหัสขนาด 16 บิต เทียบกับค่าอัตสหสัมพันธ์ของสัญญาณรบกวน

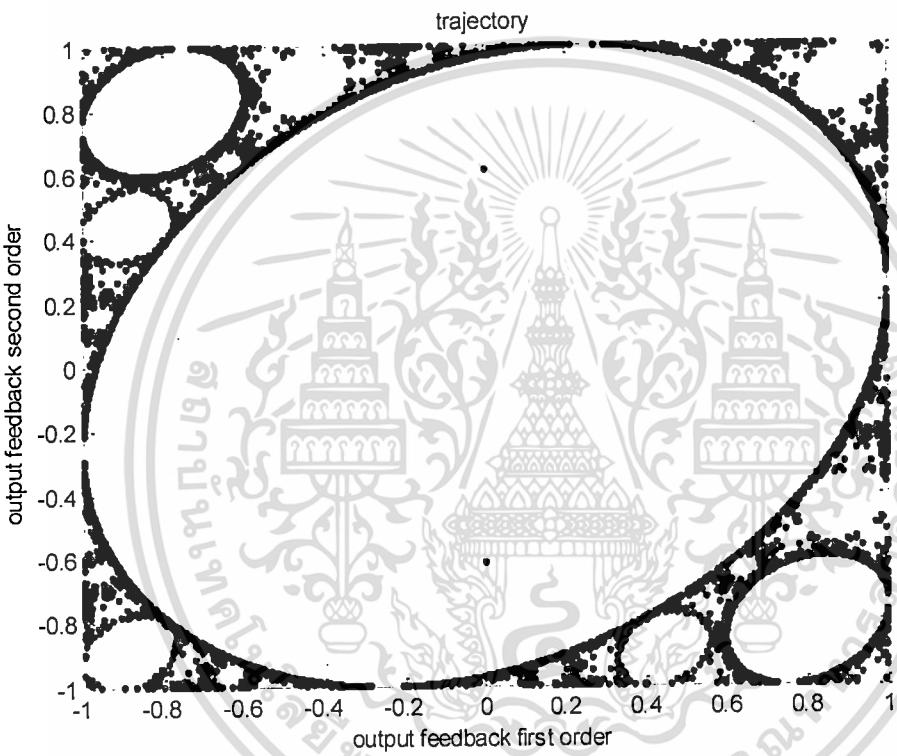
การวิเคราะห์ฟังก์ชันค่าอัตสหสัมพันธ์ของเอาต์พุตของวงจรเข้ารหัส ที่กำหนดให้ระบบมีขนาดของข้อมูลเป็น 32 บิตจะได้ผลการทดลองดังรูปที่ 4.4 โดยกำหนดให้ค่าเริ่มต้น $y(1) = 0.6135$, $y(2) = 0$ และใช้จำนวนจุดในการทดลอง 10,000 จุด



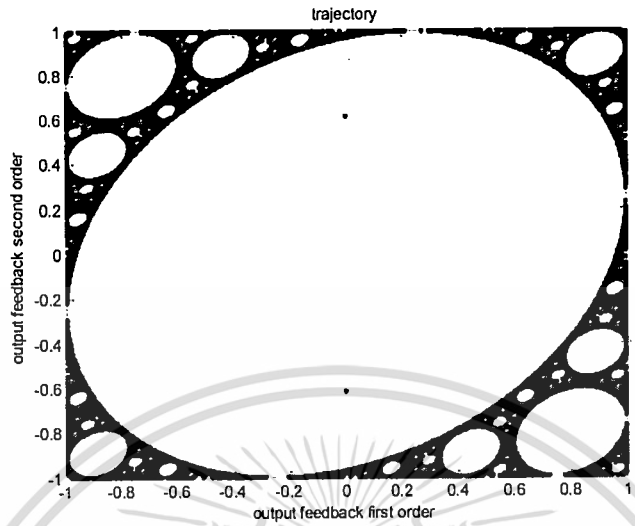
รูปที่ 4.4 กราฟค่าอัตสหสัมพันธ์ของเอาต์พุตของวงจรเข้ารหัสขนาด 32 บิต เทียบกับค่าอัตสหสัมพันธ์ของสัญญาณรบกวน

จากรูปที่ 4.2, 4.3 และ 4.4 ซึ่งแสดงการพล็อตนอร์มอลไลซ์ฟังก์ชันค่าอัตสหสัมพันธ์ของเอาต์พุตของวงจรเข้ารหัสจะเห็นได้ว่ามีลักษณะเหมือนฟังก์ชันค่าอัตสหสัมพันธ์ของสัญญาณรบกวน แสดงให้เห็นว่าเอาต์พุตของวงจรเข้ารหัสมีความสัมพันธ์กันต่ำมาก ซึ่งเป็นลักษณะของการเข้ารหัสที่ดีแสดงว่าเอาต์พุตของวงจรเข้ารหัส เป็นข้อมูลแบบสุ่มที่สามารถคาดเดาได้ยากและจะเห็นได้ว่าถ้ากำหนดให้ระบบมีจำนวนบิตมากขึ้น ฟังก์ชันค่าอัตสหสัมพันธ์ของวงเข้ารหัสจะมีความใกล้เคียงกับฟังก์ชันค่าอัตสหสัมพันธ์ของสัญญาณรบกวนมากขึ้น แต่ข้อเสียของการกำหนดให้ระบบมีจำนวนบิตมากขึ้น คือจะทำให้ระบบมีการคำนวณที่ซับซ้อนขึ้นทำให้การทำงานของระบบมีความล่าช้า

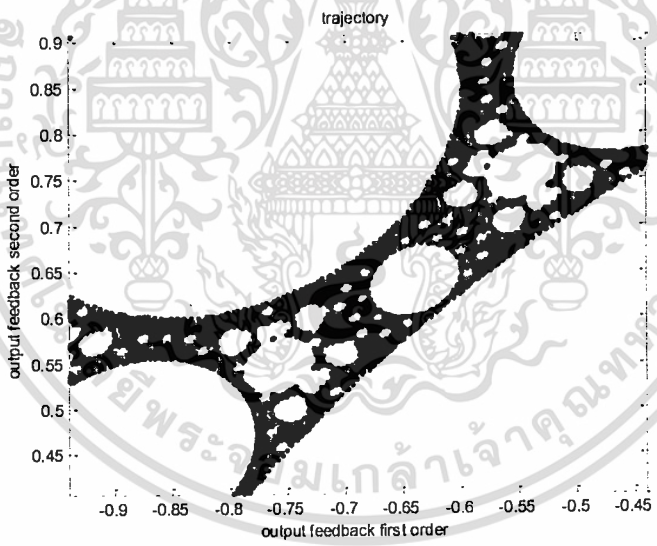
ในการทดลองเมื่ออินพุตเป็นศูนย์และกำหนดให้เงื่อนไขเริ่มต้น $y(1) = -0.6135$ และ $y(2) = 0.6135$ ค่าสัมประสิทธิ์ $c_1 = 0.5$ และ $c_2 = -1$ ใช้จำนวนจุดในการทดลอง 10,000 จุด และ100,000 จุด แล้วทดลองพล็อตเส้นทางการเคลื่อนที่ (Trajectory) โดยการพล็อตค่า Output feedback first order ของวงจรกรองสัญญาณ เทียบกับค่า Output feedback second order ของวงจรกรองสัญญาณเช่นเดียวกันจะแสดงได้ดังรูปที่ 4.5



รูปที่ 4.5 Trajectory จำนวนจุดในการทดลอง 10,000 จุด



รูปที่ 4.6 Trajectory จำนวนจุดในการทดลอง 100,000 จุด

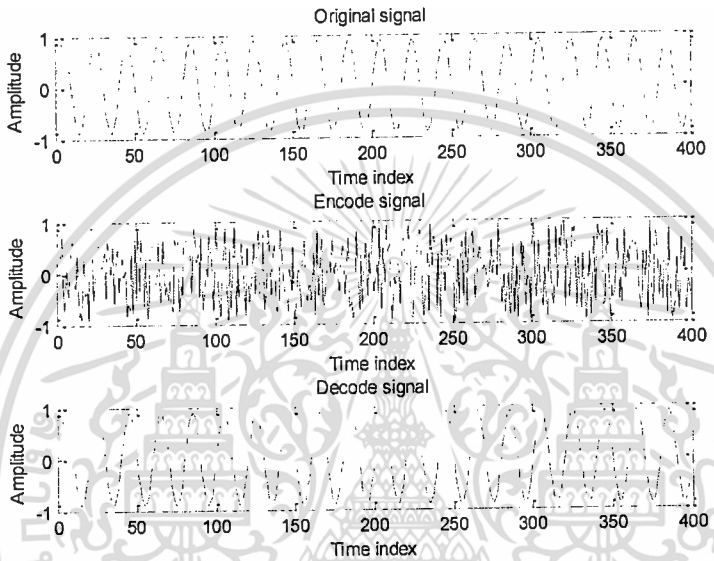


รูปที่ 4.7 ภาพขยาย Trajectory จำนวนจุดในการทดลอง 100,000 จุด

จากรูปที่ 4.6 และ 4.7 จะบ่งบอกถึงความเป็นแฟรกตัล หรือความคล้ายกับตัวเอง (Self-similarity) (จะเห็นชัดเจนในรูปที่ 4.7) ซึ่งเป็นคุณสมบัติหนึ่งที่บ่งบอกถึงความเป็นเคออสได้เช่นกัน

4.1.2 ผลการจำลองการทำงานด้วยสัญญาณไซน์

กำหนดให้สัญญาณไซน์มีความถี่ $\omega = 0.1\pi$ rad, ความยาว $k = 1:400$, Amplitude $= 0.8$ และกำหนดค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัสให้เหมือนกัน คือ $c_1 = 4$ และ $c_2 = -1$ ซึ่งผลของการจำลองการทำงานที่ได้แสดงดังรูปที่ 4.8

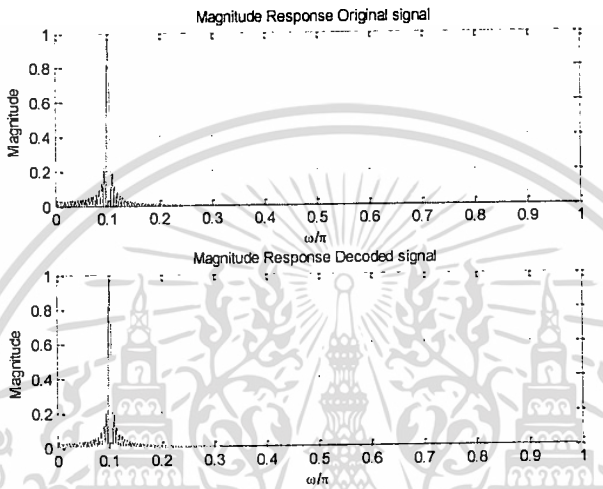


รูปที่ 4.8 ผลการจำลองการทำงานกับสัญญาณไซน์ด้วยวงจรเข้ารหัสและถอดรหัส โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน

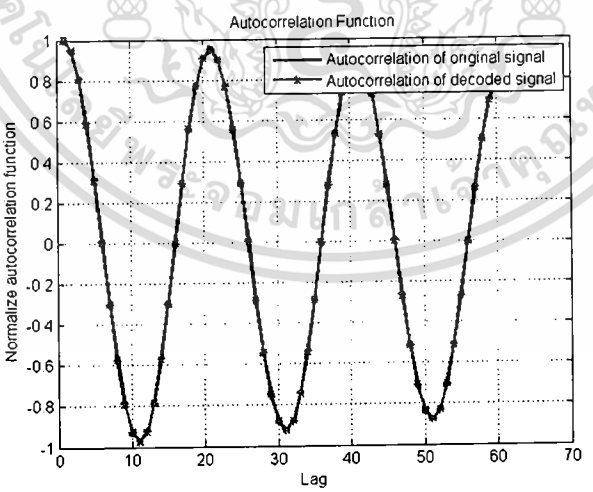
จากรูปที่ 4.8 จะเห็นว่าสามารถทำการเข้ารหัสและถอดรหัสได้ เมื่อกำหนดค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส ให้เหมือนกันคือ $c_1 = 4$ และ $c_2 = -1$ ซึ่งหาค่าระยะห่างยูคลีดีน (Euclidean distance) ระหว่างสัญญาณ Original signal และ Decoded signal จะได้ดังสมการที่ (4.1)

$$D = \sqrt{\sum_{k=1}^{400} \left\{ \left(x(k) - z(k) \right)^2 \right\}} = 4.3841\text{e-}015 \approx 0 \tag{4.1}$$

จากสมการที่ (4.1) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าประมาณ 0 ซึ่งแสดงว่าสัญญาณทั้งสองนั้นมีความใกล้เคียงกันมาก และยังสามารถตรวจเช็คได้จากการเปรียบเทียบสเปกตรัมและกราฟค่าอัตสหสัมพันธ์ของสัญญาณไซน์ระหว่าง Original signal และ Decoded signal ที่มีความเหมือนกันดังรูปที่ 4.9 และ 4.10

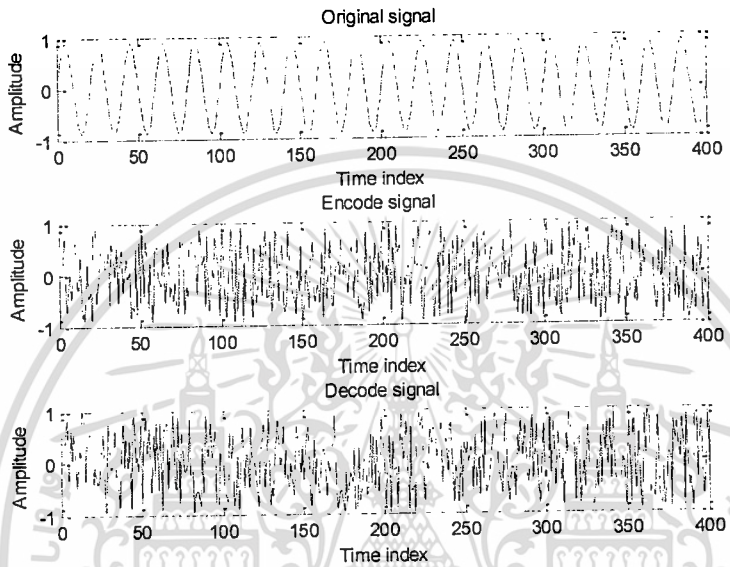


รูปที่ 4.9 สเปกตรัมของสัญญาณไซน์ (Original signal) เปรียบเทียบกับสัญญาณที่ผ่านการถอดรหัส (Decoded signal) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน



รูปที่ 4.10 กราฟค่าอัตสหสัมพันธ์ของสัญญาณไซน์ Original signal และ Decoded signal โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน

จากนั้นทำการทดลองโดยเปลี่ยนค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัส ให้มีค่าต่างกัน โดยวงจรเข้ารหัสกำหนดให้ $c_1 = 4$ และ $c_2 = -1$ ส่วนวงจรถอดรหัสกำหนดให้ $c_1 = 4$ และ $c_2 = -2$ จะได้ผลการทดลองดังรูปที่ 4.11

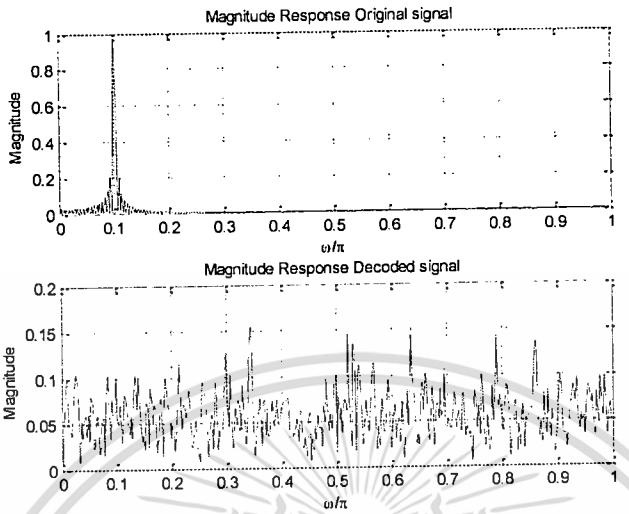


รูปที่ 4.11 ผลการจำลองการทำงานกับสัญญาณไซน์ด้วยวงจรเข้ารหัสและถอดรหัสโดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน

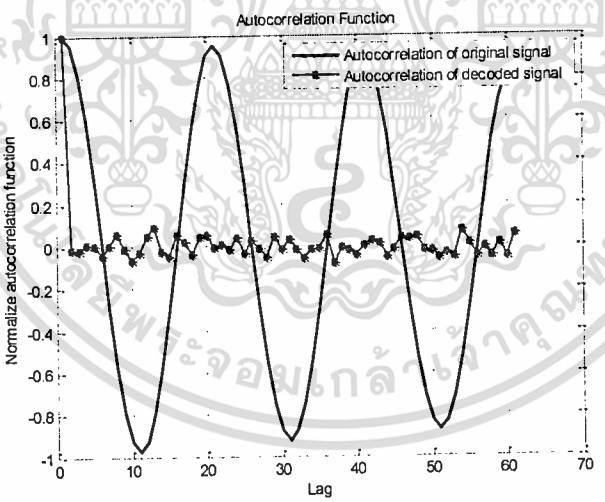
ซึ่งหาค่าระยะห่างยูคลีดีนได้ดังนี้

$$D = \sqrt{\sum_{k=1}^{400} \left\{ \left(x(k) - z(k) \right)^2 \right\}} = 17.6097 \tag{4.2}$$

จากสมการที่ (4.2) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่ามากกว่า 0 ซึ่งแสดงว่าสัญญาณทั้งสองนั้นมีความแตกต่าง และเมื่อตรวจเช็คจากการเปรียบเทียบสเปกตรัมและกราฟค่าอัตสหสัมพันธ์ของสัญญาณไซน์ระหว่าง Original signal และ Decoded signal ที่มีความแตกต่างกันดังรูปที่ 4.12 และ 4.13



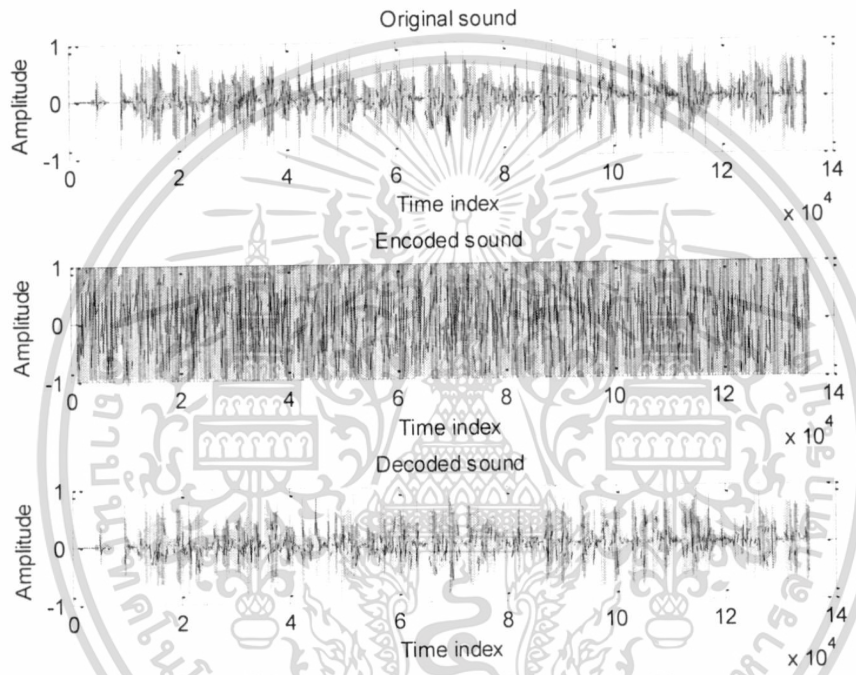
รูปที่ 4.12 สเปกตรัมของสัญญาณไซน์ (Original signal) เปรียบเทียบกับสัญญาณที่ผ่านการถอดรหัส (Decoded signal) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน



รูปที่ 4.13 กราฟค่าอัตสหสัมพันธ์ของสัญญาณไซน์ Original signal และ Decoded signal(2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน

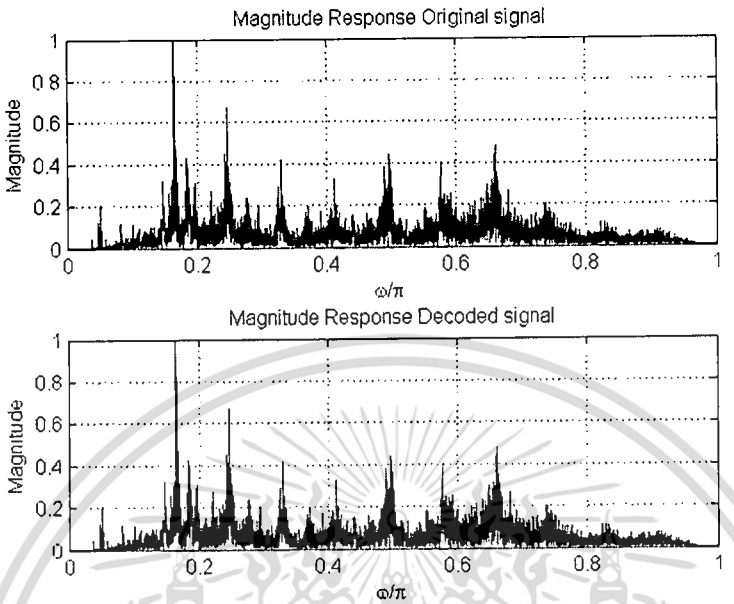
4.1.3 ผลการจำลองการทำงานด้วยข้อมูลเสียง

ในการจำลองการทำงานของวงจรเข้ารหัสและถอดรหัส กับสัญญาณเสียง โดยใช้โปรแกรม MATLAB อ่านข้อมูลเสียงแล้วนำข้อมูลที่ได้ไปทำการเข้ารหัสและถอดรหัสกำหนดค่าสัมประสิทธิ์เหมือนกันคือกำหนดให้ $c_1 = 4$ และ $c_2 = -1$ ซึ่งผลการทดลองแสดงได้ดังรูปที่ 4.14

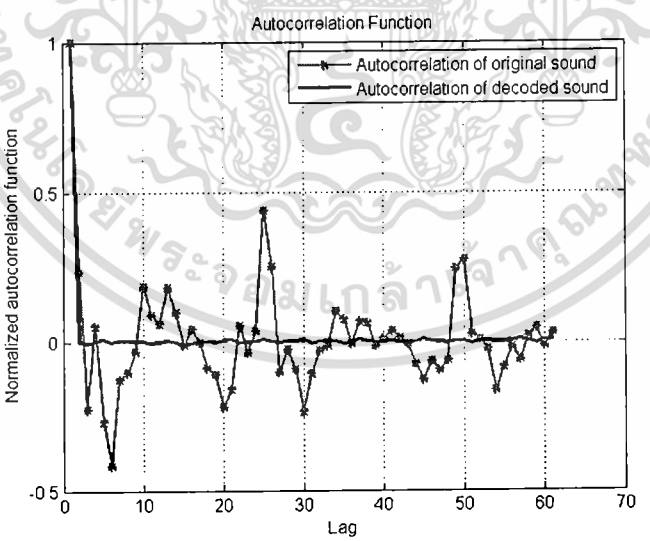


รูปที่ 4.14 ผลการจำลองการทำงานที่มีอินพุตคือข้อมูลเสียงโดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน

จากรูปที่ 4.14 เห็นได้ว่าสามารถถอดรหัสสัญญาณเสียงออกมาได้ เมื่อกำหนดค่าสัมประสิทธิ์ให้เหมือนกันทั้งฝั่งเข้ารหัสและถอดรหัส ซึ่งแสดงว่าวงจรเข้ารหัสและถอดรหัสสามารถนำไปใช้กับการเข้ารหัสและถอดรหัสเสียงได้ และสามารถตรวจเช็คได้จากการเปรียบเทียบสเปกตรัมและกราฟค่าอัตสหสัมพันธ์ของสัญญาณเสียงระหว่าง Original signal และ Decoded signal ที่มีความเหมือนกันดังรูปที่ 4.15 และ 4.16

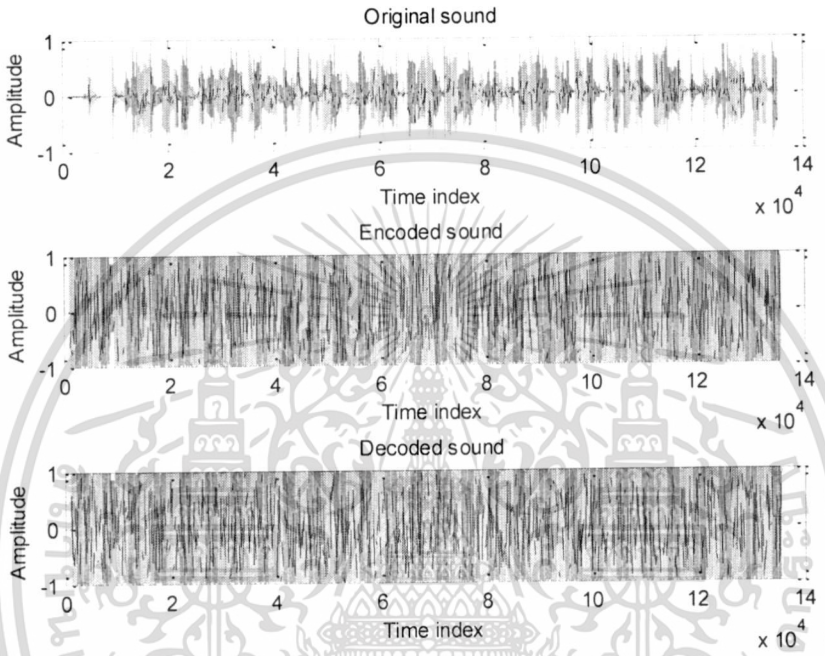


รูปที่ 4.15 สเปกตรัมของข้อมูลเสียง (Original signal) เปรียบเทียบกับสัญญาณที่ผ่านการถอดรหัส (Decoded signal) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน



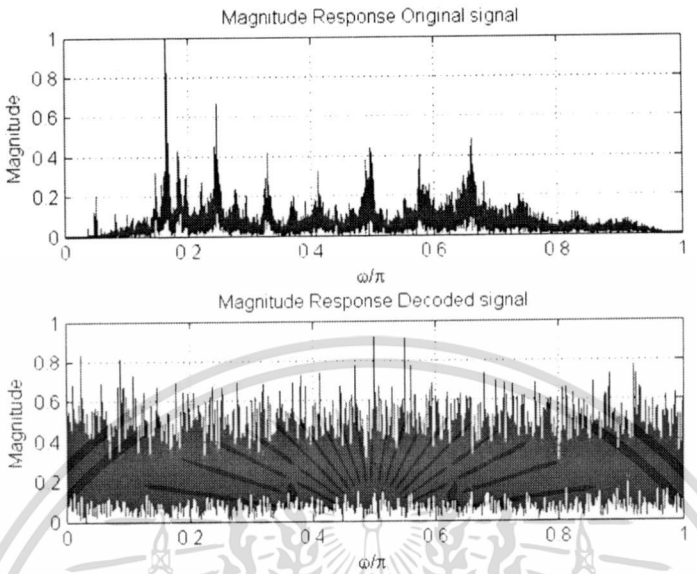
รูปที่ 4.16 กราฟค่าอัตสหสัมพันธ์ของข้อมูลเสียง Original signal และ Decoded signal โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน

จากนั้นทดลองเปลี่ยนค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัสให้มีค่าต่างกัน โดยวงจรเข้ารหัสกำหนดให้ $c_1 = 4$ และ $c_2 = -1$ ส่วนวงจรถอดรหัสกำหนดให้ $c_1 = 4$ และ $c_2 = -2$ จะได้ผลการทดลองดังรูปที่ 4.17

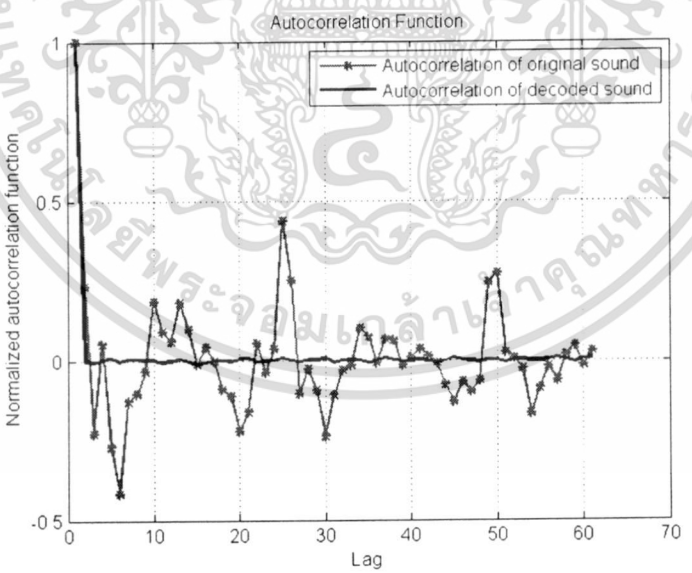


รูปที่ 4.17 ผลการจำลองการทำงานที่มีอินพุตคือข้อมูลเสียงโดยให้ค่าสัมประสิทธิ์ของวงจรมีไม่ค่าเท่ากัน

จากรูปที่ 4.17 เห็นได้ว่าไม่สามารถถอดรหัสสัญญาณเสียงออกมาได้ เมื่อกำหนดค่าสัมประสิทธิ์ของวงจรเข้าและถอดรหัสให้ต่างกัน และเมื่อตรวจเช็คจากการเปรียบเทียบสเปกตรัมและกราฟค่าอัตสหสัมพันธ์ของสัญญาณไซน์ระหว่าง Original signal และ Decoded signal ที่มีความแตกต่างกันดังรูปที่ 4.18 และ 4.19



รูปที่ 4.18 สเปกตรัมของข้อมูลเสียง (Original signal) เปรียบเทียบกับสัญญาณที่ผ่านการถอดรหัส (Decoded signal) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน

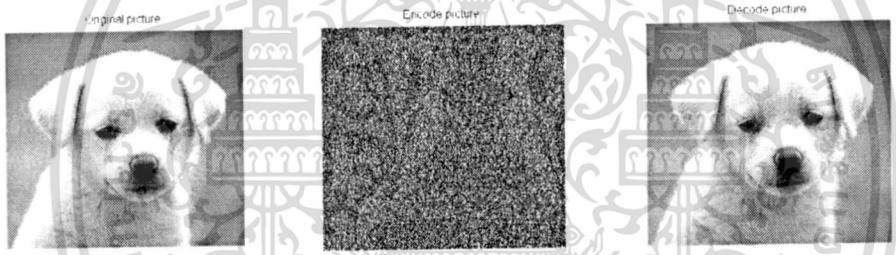


รูปที่ 4.19 กราฟค่าอัตสหสัมพันธ์ของข้อมูลเสียง Original signal และ Decoded signal โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน

4.1.4 ผลการจำลองการทำงานด้วยข้อมูลภาพ

สำหรับการจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสกับข้อมูลภาพ โดยให้โปรแกรม MATLAB อ่านภาพระดับสีเทา (gray scale) ซึ่งค่าที่อ่านได้จะอยู่ในรูปของเมทริกซ์ 2 มิติคือขนาดกว้างคูณยาวของรูปหรือค่าพิกเซล (pixel) แต่ในการเข้ารหัสและถอดรหัสของโครงงานวิจัยนี้นี้ทำงานแบบ 1 มิติ ดังนั้นจึงต้องเรียงข้อมูลใหม่โดยใช้คำสั่ง reshape ของโปรแกรม MATLAB แล้วจึงทำการเข้าและถอดรหัสโดยเลือกใช้ค่าสัมประสิทธิ์ตามที่กำหนด

โดยรูปที่ 4.20 เป็นการจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสกับข้อมูลภาพ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัสเหมือนกันคือ $c_1 = 4$ และ $c_2 = -1$



รูปที่ 4.20 ผลการจำลองการทำงานกับข้อมูลภาพโดยกำหนดค่าสัมประสิทธิ์เหมือนกัน

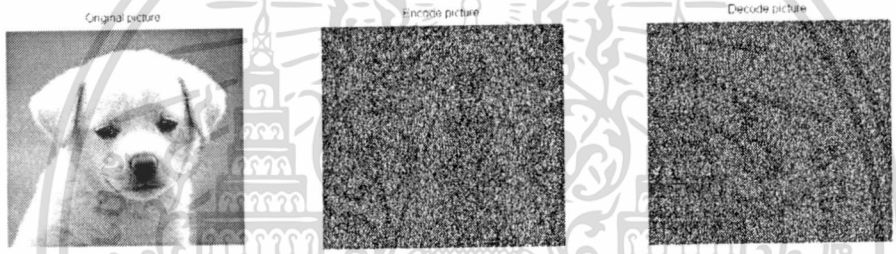
จากรูปที่ 4.20 จะเห็นว่าสามารถทำการถอดรหัสได้ เมื่อกำหนดค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัสให้เหมือนกันคือ $c_1 = 4$ และ $c_2 = -1$ ซึ่งจะหาค่าระยะห่างยูคลีดีนและค่า PSNR ระหว่างรูปภาพ Original picture และ Decoded picture จะได้ดังสมการที่ (4.2) และ (4.3) ตามลำดับ

$$D = \sqrt{\sum_{k=1}^{400} \left\{ \left(x(k) - z(k) \right)^2 \right\}} \approx 0 \tag{4.2}$$

$$\text{PSNR} = 20\log_{10}\left(\frac{b}{\text{RMSE}}\right) = 71.04 \text{ dB} \tag{4.3}$$

จากสมการที่ (4.2) และ (4.3) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าประมาณเท่ากับ 0 บ่งบอกถึงความเหมือนของรูปภาพและค่า PSNR ที่มีค่าสูงซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงกันมาก

รูปที่ 4.21 เป็นการจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสกับข้อมูลภาพ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัส ($c_1 = 4$, $c_2 = -1$) และถอดรหัส ($c_1 = 2$, $c_2 = 1$) ต่างกัน



รูปที่ 4.21 ผลการจำลองการทำงานกับข้อมูลภาพโดยกำหนดค่าสัมประสิทธิ์ต่างกัน

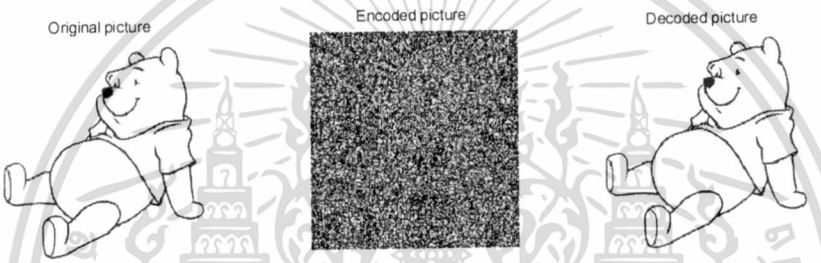
จากรูปที่ 4.21 จะเห็นว่าไม่สามารถทำการถอดรหัสได้ เมื่อกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัส ($c_1 = 4$, $c_2 = -1$) และถอดรหัสต่างกัน ($c_1 = 2$, $c_2 = 1$) ซึ่งจะหาค่าระยะห่างยูคลีดีนและค่า PSNR ระหว่างรูปภาพ Original picture และ Decoded picture จะได้ดังสมการที่ (4.4) และ (4.5) ตามลำดับ

$$D = \sqrt{\sum_{k=1}^{400}\left\{\left(x(k) - z(k)\right)^2\right\}} = 271.61 \tag{4.4}$$

$$\text{PSNR} = 20\log_{10}\left(\frac{b}{\text{RMSE}}\right) = 27.10 \text{ dB} \tag{4.5}$$

จากสมการที่ (4.4) และ (4.5) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าสูง บ่งบอกถึงความแตกต่างของรูปภาพและค่า PSNR ที่มีค่าต่ำซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงน้อยมาก

จำลองการทำงานของวงจรเข้ารหัสและถอดรหัสกับภาพการ์ตูนโดยให้โปรแกรมอ่านภาพแล้วทำการเข้าและถอดรหัส โดยกำหนดค่าสัมประสิทธิ์เหมือนกันคือ $c_1 = 4, c_2 = -1$ ได้ผลการทดลองดังรูปที่ 4.22



รูปที่ 4.22 ผลการจำลองการทำงานกับภาพการ์ตูนโดยกำหนดค่าสัมประสิทธิ์เหมือนกัน

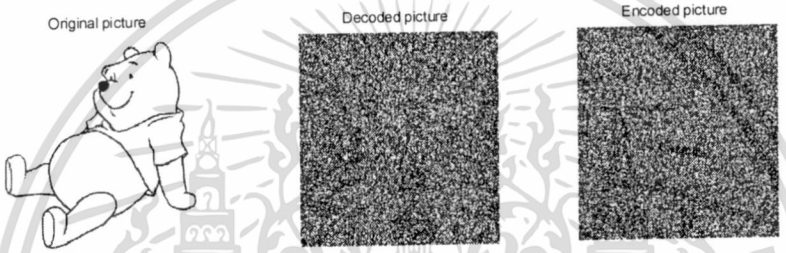
จากรูปที่ 4.22 จะเห็นว่าสามารถทำการถอดรหัสได้ เมื่อกำหนดค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัสให้เหมือนกันคือ $c_1 = 4$ และ $c_2 = -1$ ซึ่งจะหาค่าระยะห่างยูคลีดีนและค่า PSNR ระหว่างรูปภาพ Original picture และ Decoded picture จะได้ดังสมการที่ (4.6) และ (4.7) ตามลำดับ

$$D = \sqrt{\sum_{k=1}^{400} \left\{ \left(x(k) - z(k) \right)^2 \right\}} \approx 0 \tag{4.6}$$

$$\text{PSNR} = 20 \log_{10} \left(\frac{b}{\text{RMSE}} \right) = 67.31 \text{ dB} \tag{4.7}$$

จากสมการที่ (4.6) และ (4.7) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าประมาณเท่ากับ 0 บ่งบอกถึงความเหมือนของรูปภาพและค่า PSNR ที่มีค่าสูงซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงกันมาก

รูปที่ 4.23 เป็นการจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสกับข้อมูลภาพ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัส ($c_1 = 4$, $c_2 = -1$) และถอดรหัส ($c_1 = 2$, $c_2 = 1$) ต่างกัน



รูปที่ 4.23 ผลการจำลองการทำงานกับภาพการ์ตูนโดยกำหนดค่าสัมประสิทธิ์ต่างกัน

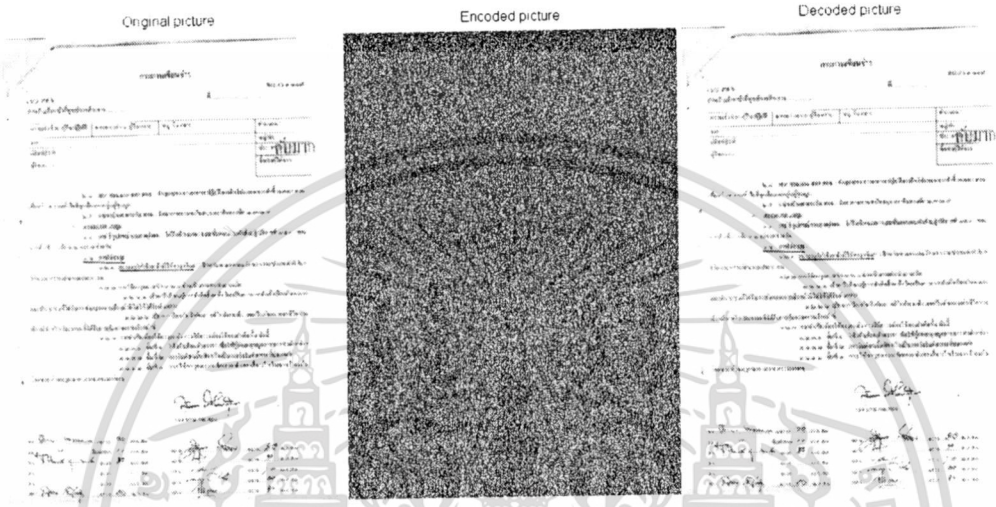
จากรูปที่ 4.23 จะเห็นว่าไม่สามารถทำการถอดรหัสได้ เมื่อกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัส ($c_1 = 4$, $c_2 = -1$) และถอดรหัสต่างกัน ($c_1 = 2$, $c_2 = 1$) ซึ่งจะหาค่าระยะห่างยูคลีดีนและค่า PSNR ระหว่างรูปภาพ Original picture และ Decoded picture จะได้ดังสมการที่ (4.8) และ (4.9) ตามลำดับ

$$D = \sqrt{\sum_{k=1}^{400} \left\{ \left(x(k) - z(k) \right)^2 \right\}} = 228.13 \tag{4.8}$$

$$\text{PSNR} = 20 \log_{10} \left(\frac{b}{\text{RMSE}} \right) = 27.11 \text{ dB} \tag{4.9}$$

จากสมการที่ (4.8) และ (4.9) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าสูง บ่งบอกถึงความแตกต่างของรูปภาพและค่า PSNR ที่มีค่าต่ำซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงน้อยมาก

จำลองการทำงานของวงจรเข้ารหัสและถอดรหัสด้วยภาพเอกสารลับโดยให้โปรแกรมอ่านภาพแล้วทำการเข้าและถอดรหัส โดยกำหนดค่าสัมประสิทธิ์เหมือนกันคือ $c_1 = 4$, $c_2 = -1$ ได้ผลการทดลองดังรูปที่ 4.24



รูปที่ 4.24 ผลการจำลองการทำงานด้วยภาพเอกสารลับโดยกำหนดค่าสัมประสิทธิ์เหมือนกัน

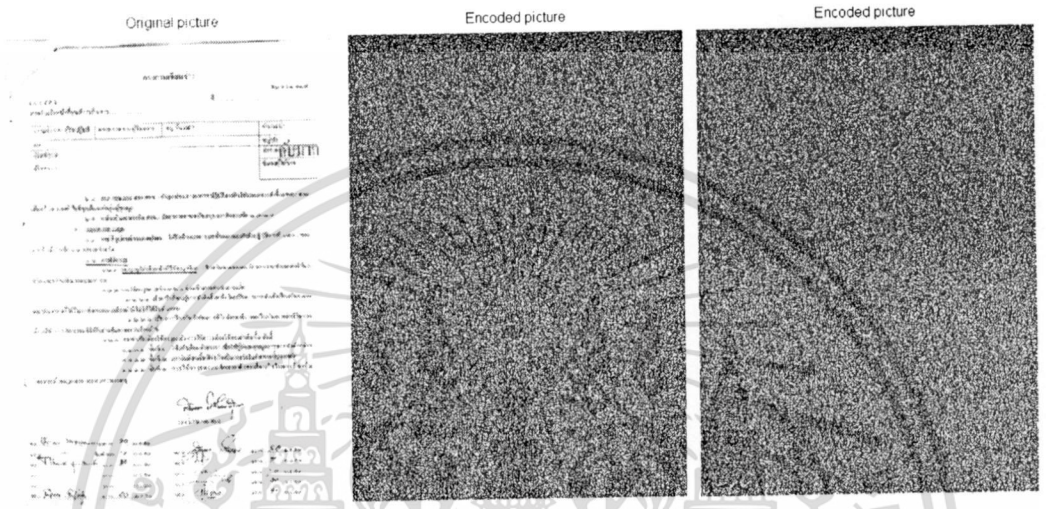
จากรูปที่ 4.24 จะเห็นว่าสามารถทำการถอดรหัสได้ เมื่อกำหนดค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัสให้เหมือนกันคือ $c_1 = 4$ และ $c_2 = -1$ ซึ่งจะหาค่าระยะห่างยูคลีดีนและค่า PSNR ระหว่างรูปภาพ Original picture และ Decoded picture จะได้ดังสมการที่ (4.10) และ (4.11) ตามลำดับ

$$D = \sqrt{\sum_{k=1}^{400} \left\{ \left(x(k) - z(k) \right)^2 \right\}} \approx 0 \tag{4.10}$$

$$\text{PSNR} = 20 \log_{10} \left(\frac{b}{\text{RMSE}} \right) = 76.42 \text{ dB} \tag{4.11}$$

จากสมการที่ (4.10) และ (4.11) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าประมาณเท่ากับ 0 บ่งบอกถึงความเหมือนของรูปภาพและค่า PSNR ที่มีค่าสูงซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงกันมาก

รูปที่ 4.25 เป็นการจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสด้วยภาพเอกสารลับ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัส ($c_1 = 4$, $c_2 = -1$) และถอดรหัสต่างกัน ($c_1 = 2$, $c_2 = 1$)



รูปที่ 4.25 ผลการจำลองการทำงานด้วยภาพเอกสารลับโดยกำหนดค่าสัมประสิทธิ์ต่างกัน

จากรูปที่ 4.25 จะเห็นว่าไม่สามารถทำการถอดรหัสได้ เมื่อกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัส ($c_1 = 4$, $c_2 = -1$) และถอดรหัสต่างกัน ($c_1 = 2$, $c_2 = 1$) ซึ่งจะหาค่าระยะห่างยูคลีดีนและค่า PSNR ระหว่างรูปภาพ Original picture และ Decoded picture จะได้ดังสมการที่ (4.12) และ (4.13) ตามลำดับ

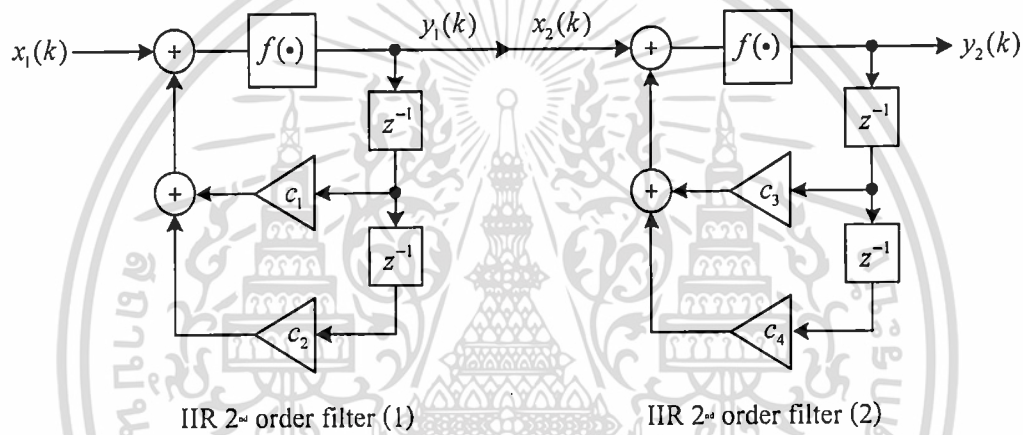
$$D = \sqrt{\sum_{k=1}^{400} \left\{ \left(x(k) - z(k) \right)^2 \right\}} = 285.41 \tag{4.12}$$

$$\text{PSNR} = 20 \log_{10} \left(\frac{b}{\text{RMSE}} \right) = 27.31 \text{ dB} \tag{4.13}$$

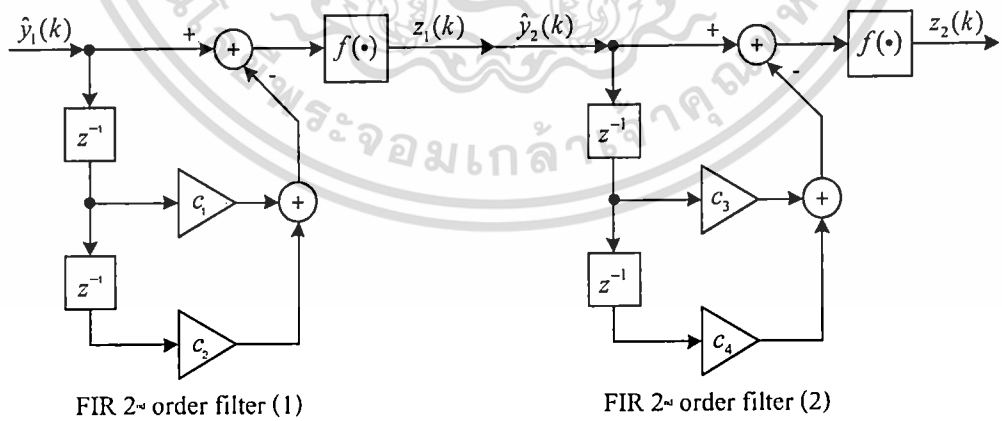
จากสมการที่ (4.12) และ (4.13) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าสูง บ่งบอกถึงความแตกต่างของรูปภาพและค่า PSNR ที่มีค่าต่ำซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงน้อยมาก

4.2 ผลการจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสี่ (4thorder filter) ด้วยโปรแกรม MATLAB

โดยโครงสร้างของวงจรเข้ารหัสและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสี่ใช้ในการจำลองการทำงานจะเกิดจากการนำวงจรกรองสัญญาณดิจิตอลอันดับสอง (2nd order filter) 2 ตัวมาต่อกันแบบ Cascade จะได้วงจรเข้ารหัสและถอดรหัสแบบ 4th order filter แสดงดังรูปที่ 4.26 และ 4.27 ตามลำดับ



รูปที่ 4.26 โครงสร้างของวงจรเข้ารหัสแบบ IIR 4th order filter

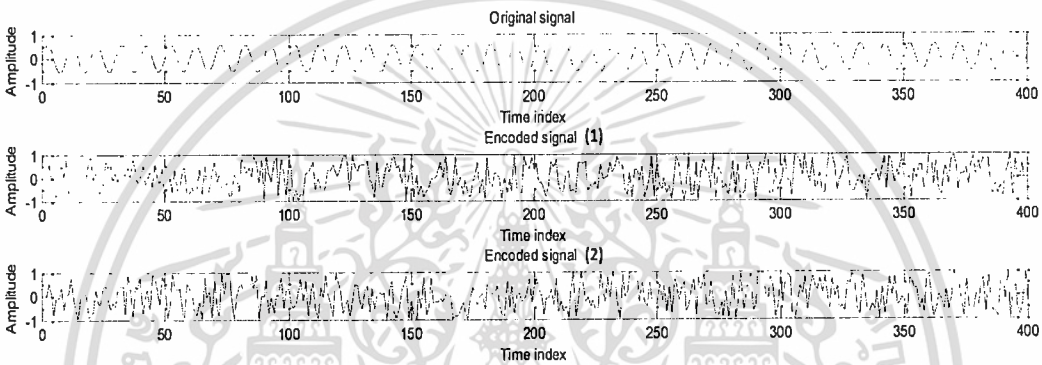


รูปที่ 4.27 โครงสร้างของวงจรถอดรหัสแบบ FIR 4th order filter

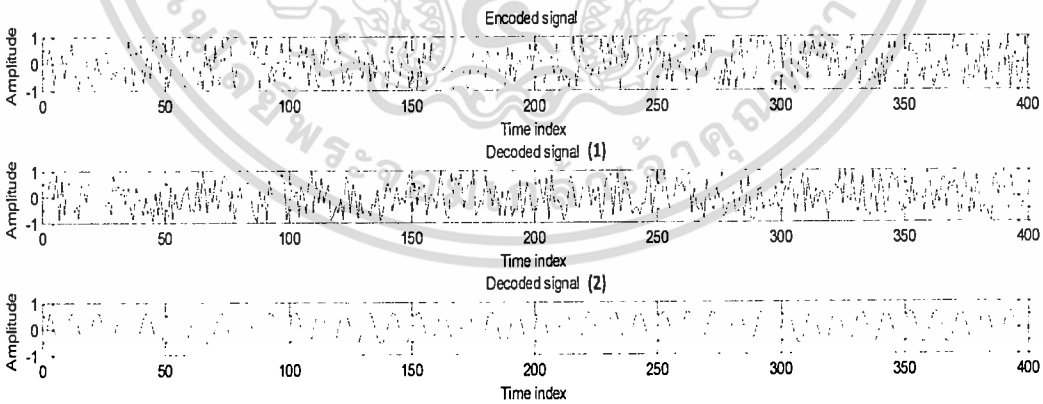
4.2.1 ผลการจำลองการทำงานด้วยสัญญาณไซน์

โดยกำหนดให้สัญญามีความถี่ $\omega = 0.1\pi$ rad กำหนดให้มีความยาว $k = 1:400$ Amplitude = 0.8

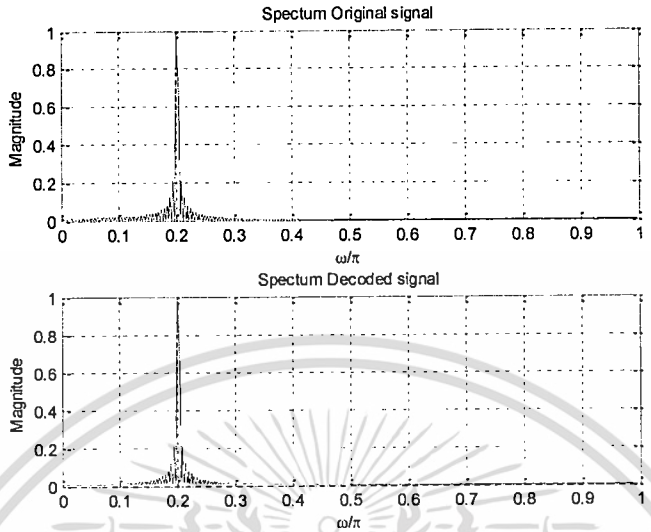
4.2.1.1 กำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัสมีค่าเท่ากันคือ $c_1 = 4, c_2 = -1$ $c_3 = 6$ และ $c_4 = -2$ ซึ่งผลของการจำลองการทำงานที่ได้แสดงดังรูปที่ 4.28 และ 4.29



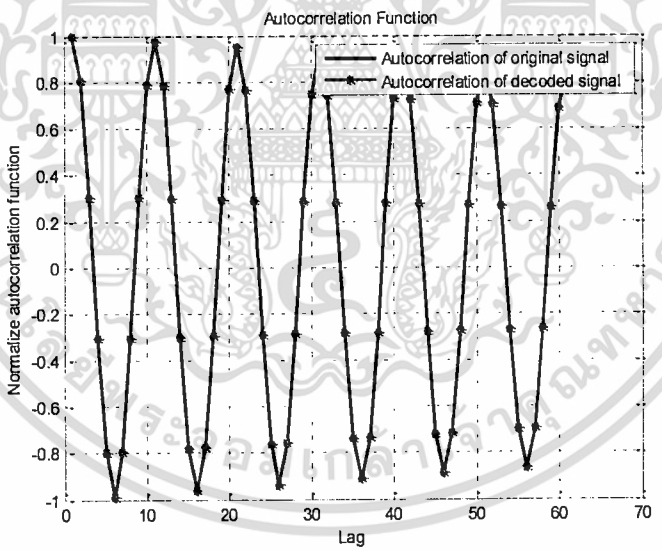
รูปที่ 4.28 ผลการจำลองการทำงานที่มีอินพุตคือสัญญาณไซน์ (Original signal) แล้วทำการเข้ารหัสด้วยวงจรเข้ารหัสแบบ IIR 2^{nd} order filter ตัวที่ (1) และ (2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน



รูปที่ 4.29 ผลการจำลองการทำงานที่มีอินพุตคือสัญญาณไซน์ที่ผ่านการเข้ารหัส (Encoded signal) แล้วทำการถอดรหัสด้วยวงจรถอดรหัสแบบ FIR 2^{nd} order filter ตัวที่ (1) และ (2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน



รูปที่ 4.30 สเปกตรัมของสัญญาณไซน์ (Original signal) เปรียบเทียบกับสัญญาณที่ผ่านการถอดรหัส (Decoded signal(2)) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน



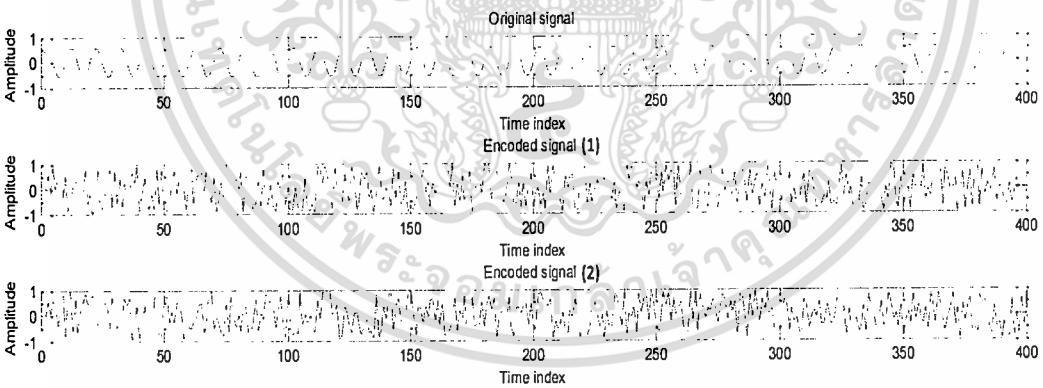
รูปที่ 4.31 กราฟค่าอัตสหสัมพันธ์ของสัญญาณ Original signal และ Decoded signal(2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน

จากรูปที่ 4.28 และ 4.29 จะเห็นว่าสามารถทำการเข้ารหัสและถอดรหัสได้ เนื่องจากกำหนดค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัสให้มีค่าเท่ากัน จะหาค่าระยะห่างยูคลีดีนระหว่างสัญญาณ Original signal และ Decoded signal(2) จะได้ดังสมการที่ (4.14)

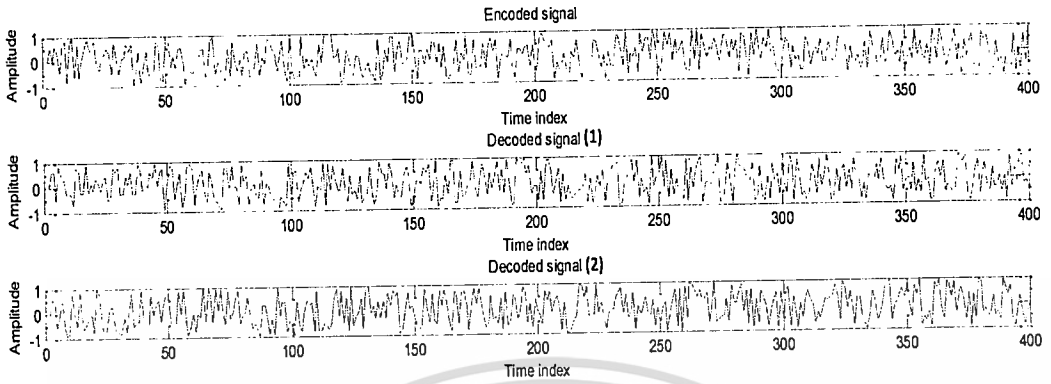
$$D=\sqrt{\sum_{k=1}^{400}\left\{\left(x(k)-z(k)\right)^2\right\}}=2.3195\text{e-}014\approx 0 \tag{4.14}$$

จากสมการที่ (4.14) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าประมาณ 0 อีกทั้งสัญญาณ Original signal และ Decoded signal(2) นั้นมีขนาดสเปกตรัมเท่ากัน (ในรูปที่ 4.30) และกราฟค่าอัตสหสัมพันธ์ของสัญญาณทั้ง 2 นั้นทับกัน (ในรูปที่ 4.31) ซึ่งแสดงว่าสัญญาณทั้งสองนั้นมีความใกล้เคียงกันมาก

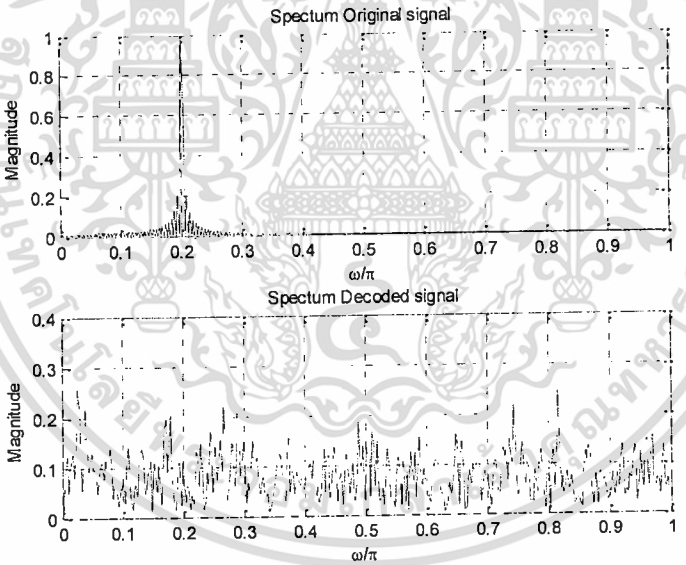
4.2.1.2 กำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัสมีค่าไม่เท่ากันคือ ($c_1=4\ c_2=-1\ c_3=6\ c_4=-2$) และ ($c_1=4\ c_2=-1\ c_3=5\ c_4=-2$) ตามลำดับ ซึ่งผลของการจำลองการทำงานที่ได้แสดงดังรูปที่ 4.32 และ 4.33



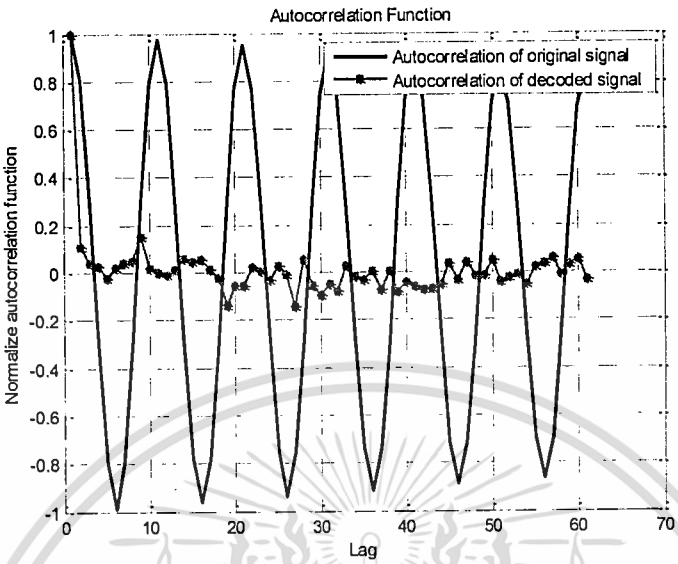
รูปที่ 4.32 ผลการจำลองการทำงานที่มีอินพุตคือสัญญาณไซน์ (Original signal) แล้วทำการเข้ารหัสด้วยวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (1) และ (2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน



รูปที่ 4.33 ผลการจำลองการทำงานที่มีอินพุตคือสัญญาณไซน์ที่ผ่านการเข้ารหัส (Encoded signal) แล้วทำการถอดรหัสด้วยวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (1) และ (2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน



รูปที่ 4.34 สเปกตรัมของสัญญาณไซน์ Original signal เปรียบเทียบกับสัญญาณที่ผ่านการถอดรหัส Decoded signal (2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน



รูปที่ 4.35 กราฟค่าอัตสหสัมพันธ์ของสัญญาณ Original signal และ Decoded signal(2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน

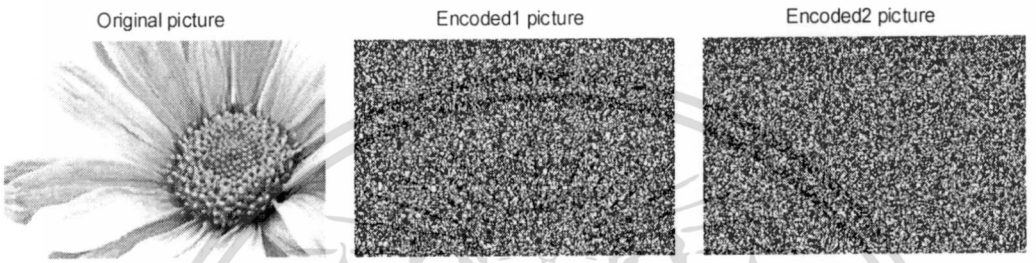
จากรูปที่ 4.32 และ 4.33 จะเห็นว่าวงจรถอดรหัสจะไม่สามารถทำการถอดรหัสได้ เมื่อกำหนดค่าสัมประสิทธิ์ของวงจรต่างกัน ซึ่งหาค่าระยะห่างยูคลีดีนได้ดังนี้

$$D = \sqrt{\sum_{k=1}^{400}\left\{\left(x(k)-z(k)\right)^2\right\}} = 14.3531 \tag{4.15}$$

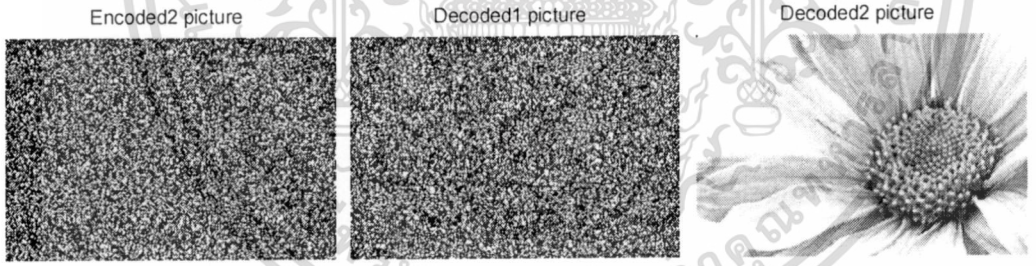
จากสมการที่ (4.15) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่ามากกว่า 0 รวมทั้งสัญญาณ Original signal และ Decoded signal(2) มีขนาดสเปกตรัมที่ไม่เท่ากัน (ในรูปที่ 4.34) และกราฟค่าอัตสหสัมพันธ์ของสัญญาณทั้ง 2 นั้นมีความแตกต่างกัน (ในรูปที่ 4.35) ซึ่งแสดงว่าสัญญาณทั้งสองนั้นมีความไม่เหมือนของสัญญาณ

4.2.2 ผลการจำลองการทำงานด้วยข้อมูลภาพ

4.2.2.1 กำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัสมีค่าเท่ากันคือ $c_1 = 4, c_2 = -1$ $c_3 = 6$ และ $c_4 = -2$ ซึ่งผลของการจำลองการทำงานที่ได้แสดงดังรูปที่ 4.36 และ 4.37



รูปที่ 4.36 ผลการจำลองการทำงานที่มีอินพุตคือข้อมูลภาพ (Original picture) แล้วทำการเข้ารหัสด้วยวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (1) และ (2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน



รูปที่ 4.37 ผลการจำลองการทำงานที่มีอินพุตคือข้อมูลภาพที่ผ่านการเข้ารหัส (Encoded signal) แล้วทำการถอดรหัสด้วยวงจรถอดรหัสแบบ FIR 2nd order filter ตัวที่ (1) และ (2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน

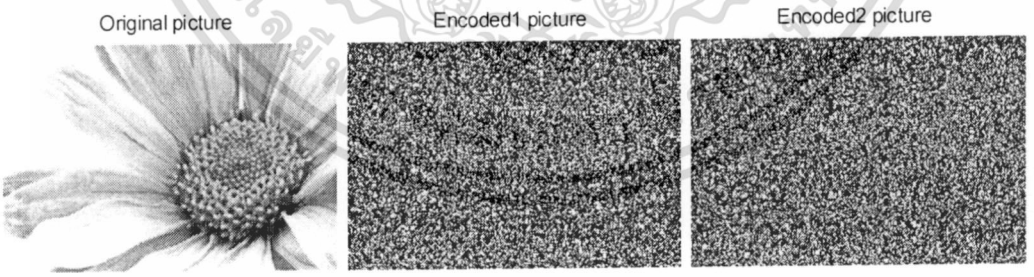
จากรูปที่ 4.36 และ 4.37 จะเห็นว่าไม่สามารถทำการถอดรหัสได้ เมื่อกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัส ($c_1 = 4$, $c_2 = -1$) และถอดรหัสต่างกัน ($c_1 = 2$, $c_2 = 1$) ซึ่งจะหาค่าระยะห่างยูคลีดีนและค่า PSNR ระหว่างรูปภาพ Original picture และ Decoded picture จะได้ดังสมการที่ (4.16) และ (4.17) ตามลำดับ

$$D = \sqrt{\sum_{k=1}^{400} \left\{ \left(x(k) - z(k) \right)^2 \right\}} = 271.61 \tag{4.16}$$

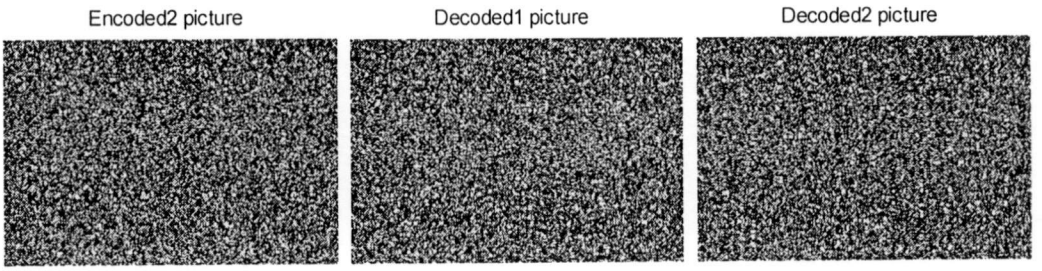
$$\text{PSNR} = 20 \log_{10} \left(\frac{b}{\text{RMSE}} \right) = 27.10 \text{ dB} \tag{4.17}$$

จากสมการที่ (4.16) และ (4.17) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าสูง บ่งบอกถึงความแตกต่างของรูปภาพและค่า PSNR ที่มีค่าต่ำซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงน้อยมาก

4.2.2.2 กำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัสมีค่าไม่เท่ากันคือ ($c_1 = 4$ $c_2 = -1$ $c_3 = 6$ $c_4 = -2$) และ ($c_1 = 4$ $c_2 = -1$ $c_3 = 5$ $c_4 = -2$) ซึ่งผลของการจำลองการทำงานที่ได้แสดงดังรูปที่ 4.38 และ 4.39



รูปที่ 4.38 ผลการจำลองการทำงานที่มีอินพุตคือข้อมูลภาพ (Original picture) แล้วทำการเข้ารหัสด้วยวงจรเข้ารหัสแบบ IIR 2nd order filter ตัวที่ (1) และ (2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน



รูปที่ 4.39 ผลการจำลองการทำงานที่มีอินพุตคือข้อมูลภาพที่ผ่านการเข้ารหัส (Encoded signal) แล้วทำการถอดรหัสด้วยวงจรถอดรหัสแบบ FIR 2^{nd} order filter ตัวที่ (1) และ (2) โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน

จากรูปที่ 4.38 และ 4.39 จะเห็นว่าไม่สามารถทำการถอดรหัสได้ เมื่อกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัส ($c_1 = 4$ $c_2 = -1$ $c_3 = 6$ $c_4 = -2$) และถอดรหัส ($c_1 = 4$ $c_2 = -1$ $c_3 = 5$ $c_4 = -2$) ต่างกัน ซึ่งจะหาค่าระยะห่างยูคลีดีนและค่า PSNR ระหว่างรูปภาพ Original picture และ Decoded picture จะได้ดังสมการที่ (4.18) และ (4.19) ตามลำดับ

$$D = \sqrt{\sum_{k=1}^{400} \left\{ \left(x(k) - z(k) \right)^2 \right\}} = 186.6484 \tag{4.18}$$

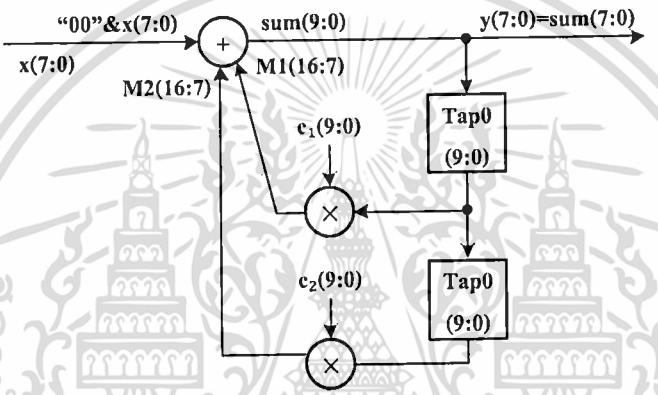
$$\text{PSNR} = 20 \log_{10} \left(\frac{b}{\text{RMSE}} \right) = 27.08 \text{ dB} \tag{4.19}$$

จากสมการที่ (4.18) และ (4.19) จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าสูง บ่งบอกถึงความแตกต่างของรูปภาพและค่า PSNR ที่มีค่าต่ำซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงน้อยมาก

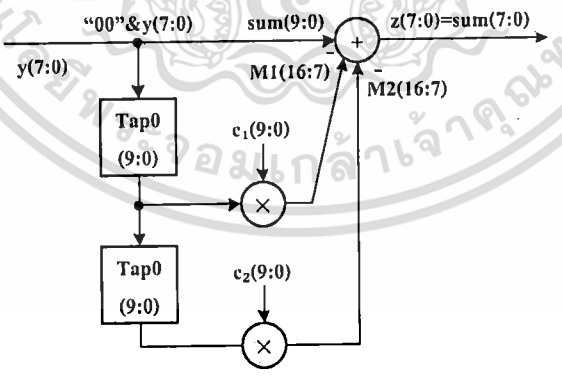
4.3 ผลการทดลองการทำงานของวงจรเข้ารหัสและถอดรหัสโดยอุปกรณ์ FPGA

4.3.1 วงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสองที่มีอินพุตและเอาต์พุตขนาด 8 บิต

โดยโครงสร้างวงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสองที่มีอินพุตและเอาต์พุตขนาด 8 บิต แสดงได้ดังรูปที่ 4.40 และ 4.41 ตามลำดับ

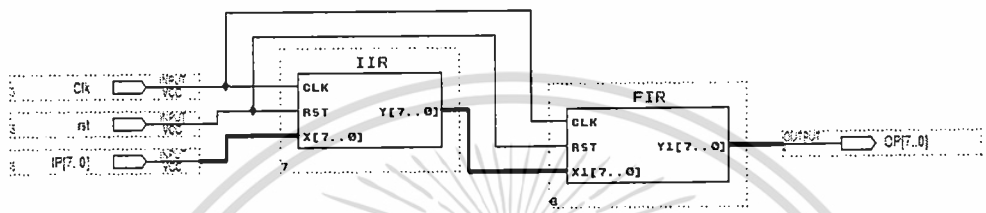


รูปที่ 4.40 โครงสร้างของวงจรเข้ารหัส แบบ IIR 2nd order filter ขนาด 8 บิต

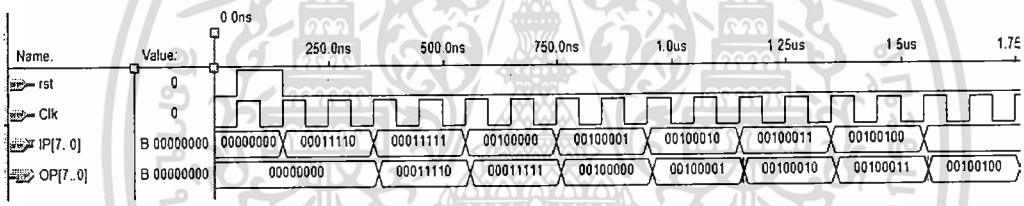


รูปที่ 4.41 โครงสร้างของวงจรถอดรหัสแบบ FIR 2nd order filter ขนาด 8 บิต

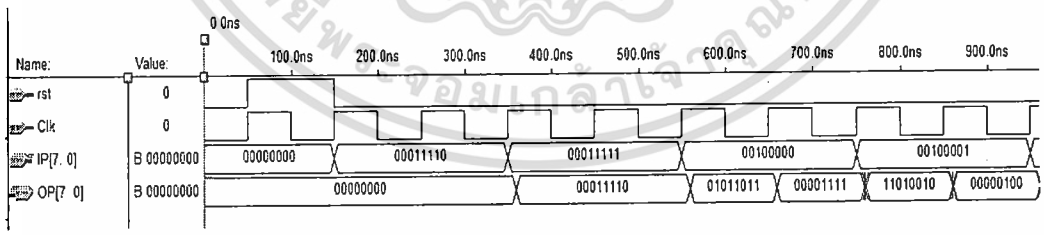
โครงสร้าง RTL Schematic แสดงการเชื่อมต่อวงจรเข้ารหัสแบบ IIR 2nd order filter กับวงจรถอดรหัสแบบ FIR 2nd order filter ได้ดังรูปที่ 4.42 สามารถจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสขนาด 8 บิต ในรูปแบบเลขฐานสอง โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากันและไม่เท่ากัน ดังรูปที่ 4.43 และ 4.44 ตามลำดับ



รูปที่ 4.42 โครงสร้าง RTL Schematic ของวงจรเข้าและถอดรหัสขนาด 8 บิต



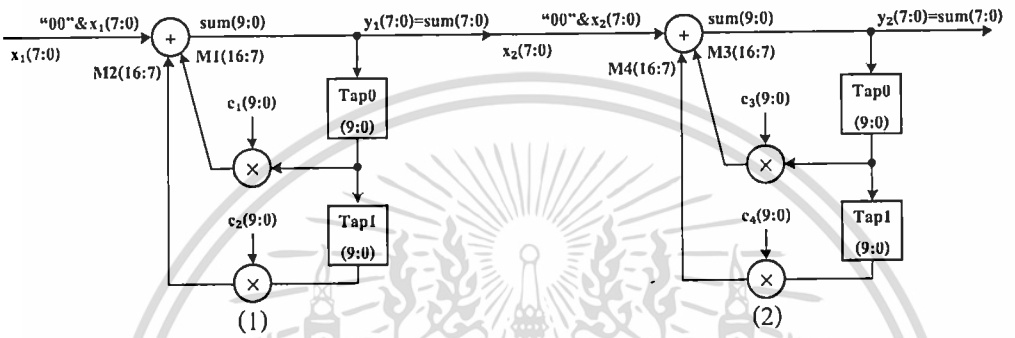
รูปที่ 4.43 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสในรูปเลขฐานสอง
โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากัน



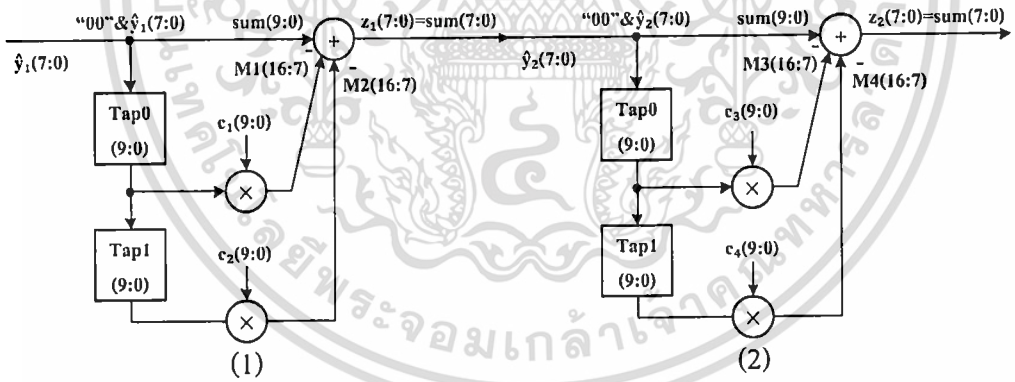
รูปที่ 4.44 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสในรูปเลขฐานสอง
โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส ไม่เท่ากัน

4.3.2 วงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสี่ที่มีอินพุตและเอาต์พุตขนาด 8 บิต

โดยโครงสร้างวงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับ สี่ที่มีอินพุตและเอาต์พุตขนาด 8 บิต แสดงได้ดังรูปที่ 4.45 และ 4.46 ตามลำดับ

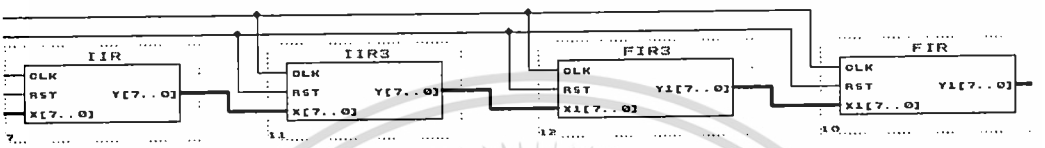


รูปที่ 4.45 โครงสร้างของวงจรเข้ารหัส แบบ IIR 4th order filter ขนาด 8 บิต

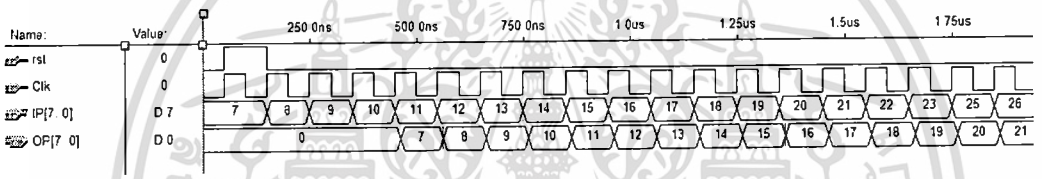


รูปที่ 4.46 โครงสร้างของวงจรถอดรหัส แบบ FIR 4th order filter ขนาด 8 บิต

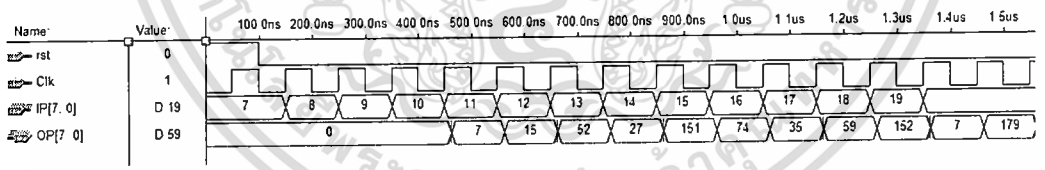
โครงสร้าง RTL Schematic แสดงการเชื่อมต่อวงจรเข้ารหัสแบบ IIR 4th order filter กับวงจรถอดรหัสแบบ FIR 4th order filter ได้ดังรูปที่ 4.47 สามารถจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสขนาด 8 บิต ในรูปแบบเลขฐานสอง โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากันและไม่เท่ากัน ดังรูปที่ 4.48 และ 4.49 ตามลำดับ



รูปที่ 4.47 โครงสร้าง RTL Schematic ของวงจรเข้าและถอดรหัสแบบ 4th order filter ขนาด 8 บิต



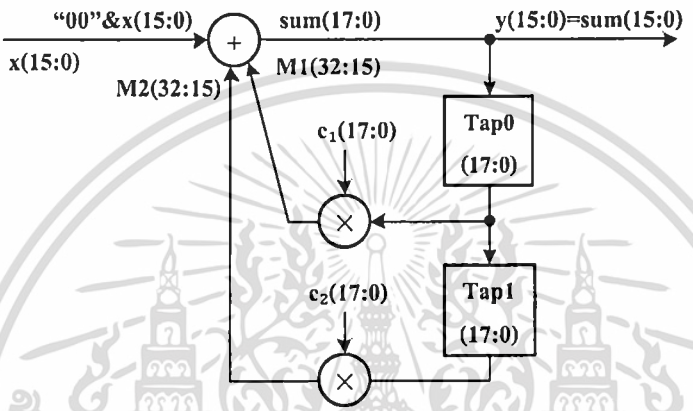
รูปที่ 4.48 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสในรูปเลขฐานสิบ
โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากัน



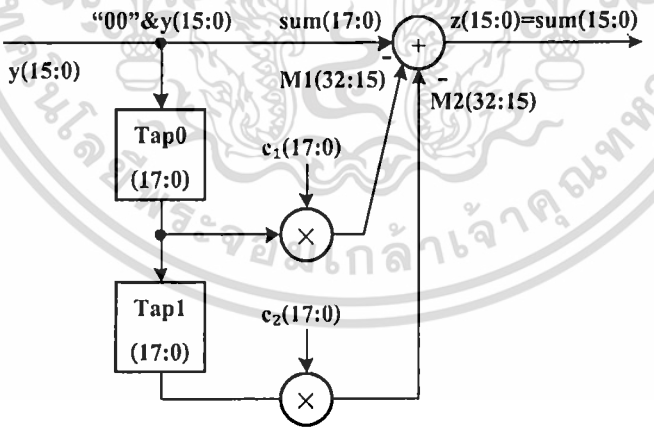
รูปที่ 4.49 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสในรูปเลขฐานสิบ
โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส ไม่เท่ากัน

4.3.3 วงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสองที่มีอินพุตและเอาต์พุตขนาด 16 บิต

โดยโครงสร้างวงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสองที่มีอินพุตและเอาต์พุตขนาด 16 บิต แสดงได้ดังรูปที่ 4.50 และ 4.51 ตามลำดับ

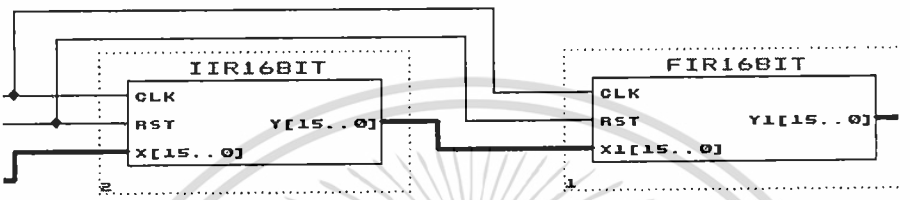


รูปที่ 4.50 โครงสร้างของวงจรเข้ารหัสแบบ IIR 2nd order filter ขนาด 16 บิต

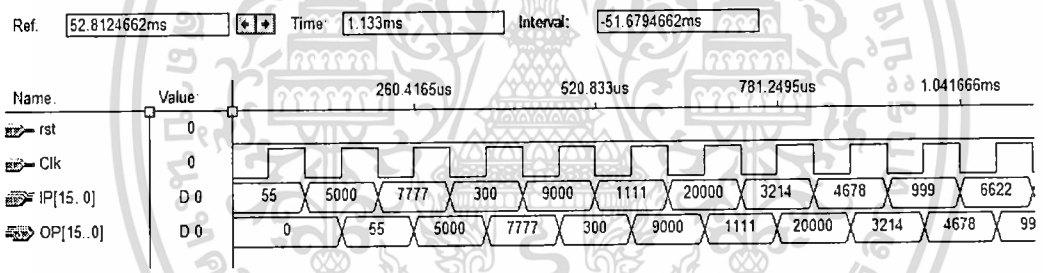


รูปที่ 4.51 โครงสร้างของวงจรถอดรหัสแบบ FIR 2nd order filter ขนาด 16 บิต

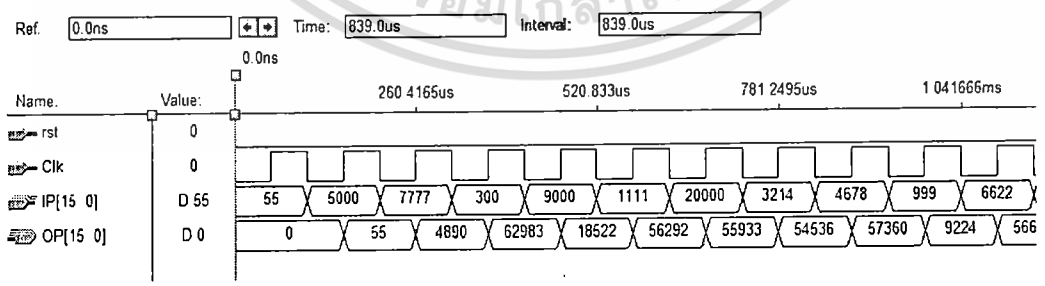
โครงสร้าง RTL Schematic แสดงการเชื่อมต่อวงจรเข้ารหัสแบบ IIR 2nd order filter กับวงจรถอดรหัสแบบ FIR 2nd order filter ได้ดังรูปที่ 4.52 สามารถจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสขนาด 16 บิต ในรูปแบบเลขฐานสอง โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากันและไม่เท่ากัน ดังรูปที่ 4.53 และ 4.54 ตามลำดับ



รูปที่ 4.52 โครงสร้าง RTL Schematic ของวงจรเข้าและถอดรหัส แบบ 2nd order filter ขนาด 16 บิต



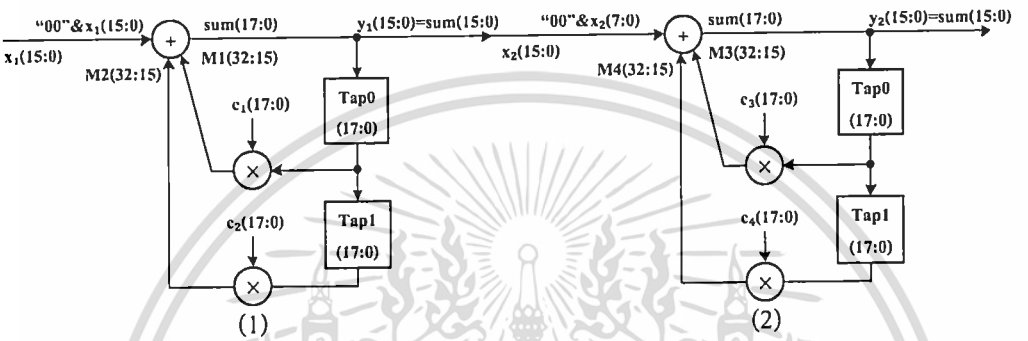
รูปที่ 4.53 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 2nd order filter ขนาด 16 บิตในรูปเลขฐานสิบ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากัน



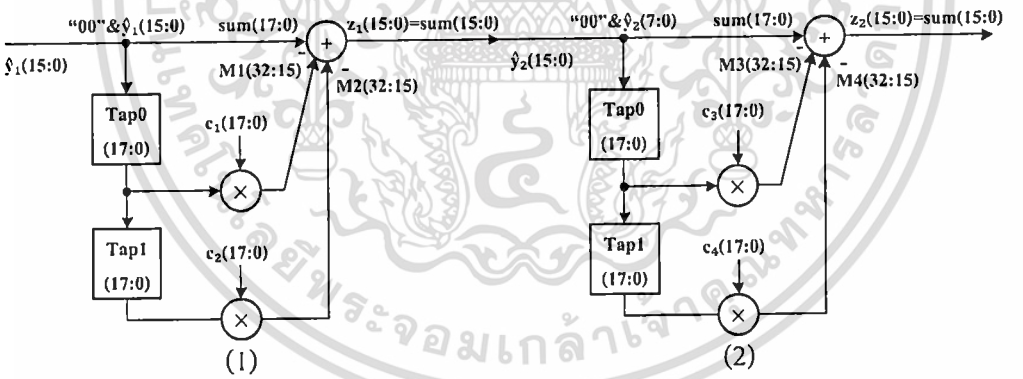
รูปที่ 4.54 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 2nd order filter ขนาด 16 บิตในรูปเลขฐานสิบ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส ไม่เท่ากัน

4.3.4 วงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสี่ที่มีอินพุตและเอาต์พุตขนาด 16 บิต

โดยโครงสร้างวงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับ สี่ที่มีอินพุตและเอาต์พุตขนาด 16 บิต แสดงได้ดังรูปที่ 4.55 และ 4.56 ตามลำดับ

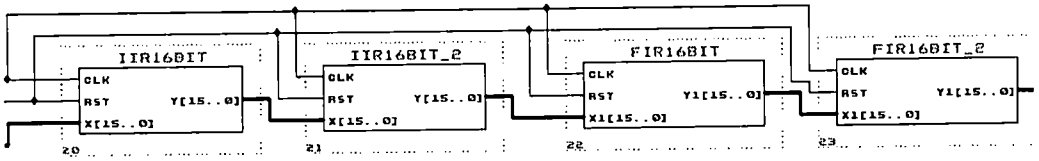


รูปที่ 4.55 โครงสร้างของวงจรเข้ารหัสแบบ 4th order filter ขนาด 16 บิต

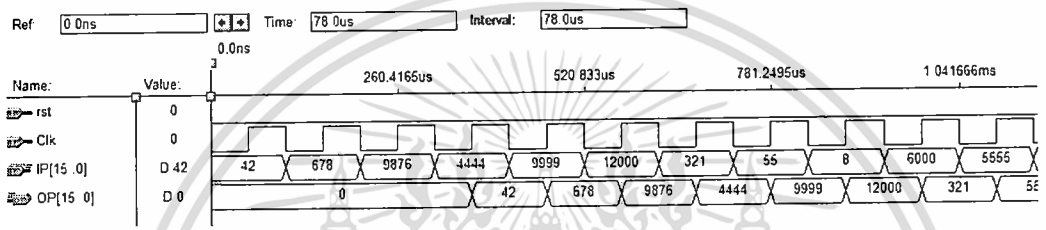


รูปที่ 4.56 โครงสร้างของวงจรถอดรหัสแบบ 4th order filter ขนาด 16 บิต

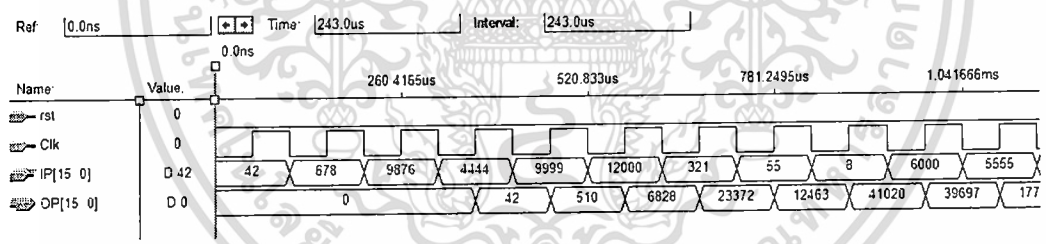
โครงสร้าง RTL Schematic แสดงการเชื่อมต่อวงจรเข้ารหัสแบบ IIR 4th order filter กับวงจรถอดรหัสแบบ FIR 4th order filter ได้ดังรูปที่ 4.57 สามารถจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสขนาด 16 บิต ในรูปแบบเลขฐานสอง โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากันและไม่เท่ากัน ดังรูปที่ 4.58 และ 4.59 ตามลำดับ



รูปที่ 4.57 โครงสร้าง RTL Schematic ของวงจรเข้าและถอดรหัส
แบบ 4th order filter ขนาด 16 บิต



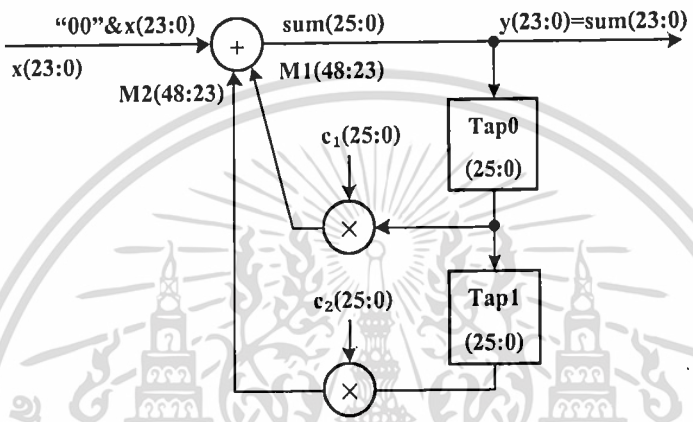
รูปที่ 4.58 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 4th order filter ขนาด 16 บิต
ในรูปเลขฐานสิบโดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากัน



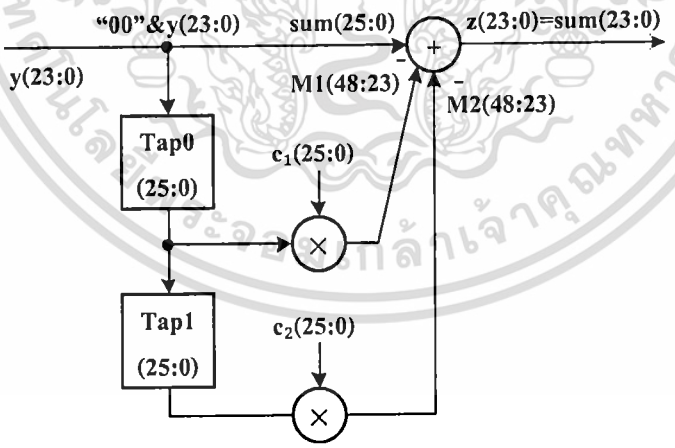
รูปที่ 4.59 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 4th order filter ขนาด 16 บิต
ในรูปเลขฐานสิบโดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส ไม่เท่ากัน

4.3.5 วงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสองที่มีอินพุตและเอาต์พุตขนาด 24 บิต

โดยโครงสร้างวงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสองที่มีอินพุตและเอาต์พุตขนาด 24 บิต แสดงได้ดังรูปที่ 4.60 และ 4.61 ตามลำดับ

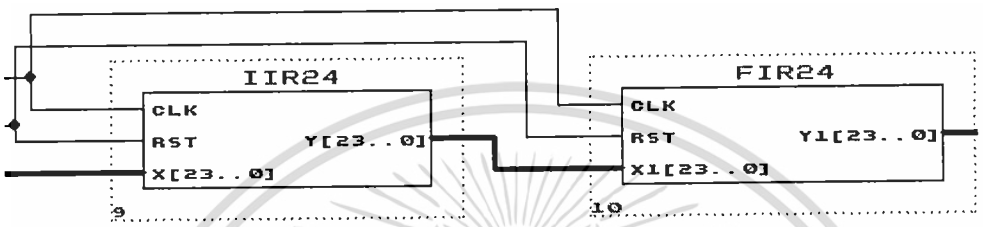


รูปที่ 4.60 โครงสร้างของวงจรเข้ารหัสแบบ IIR 2nd order filter ขนาด 24 บิต

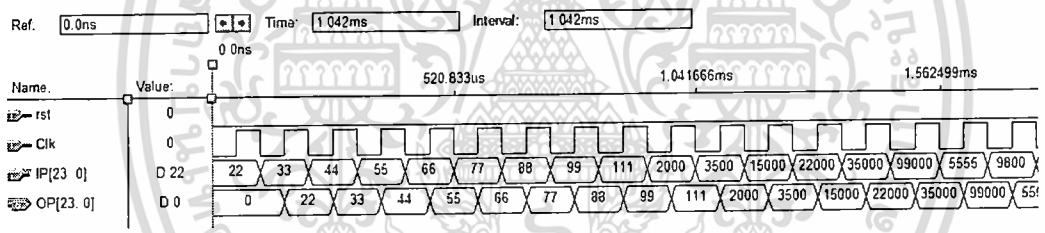


รูปที่ 4.61 โครงสร้างของวงจรถอดรหัสแบบ FIR 2nd order filter ขนาด 24 บิต

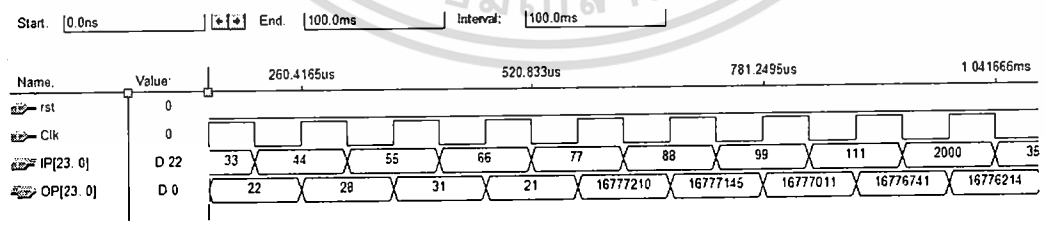
โครงสร้าง RTL Schematic แสดงการเชื่อมต่อวงจรเข้ารหัสแบบ IIR 2^{nd} order filter กับวงจรถอดรหัสแบบ FIR 2^{nd} order filter ได้ดังรูปที่ 4.62 สามารจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสขนาด 24 บิต ในรูปแบบเลขฐานสอง โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากันและไม่เท่ากัน ดังรูปที่ 4.63 และ 4.64 ตามลำดับ



รูปที่ 4.62 โครงสร้าง RTL Schematic ของวงจรเข้าและถอดรหัส แบบ 2^{nd} order filter ขนาด 24 บิต



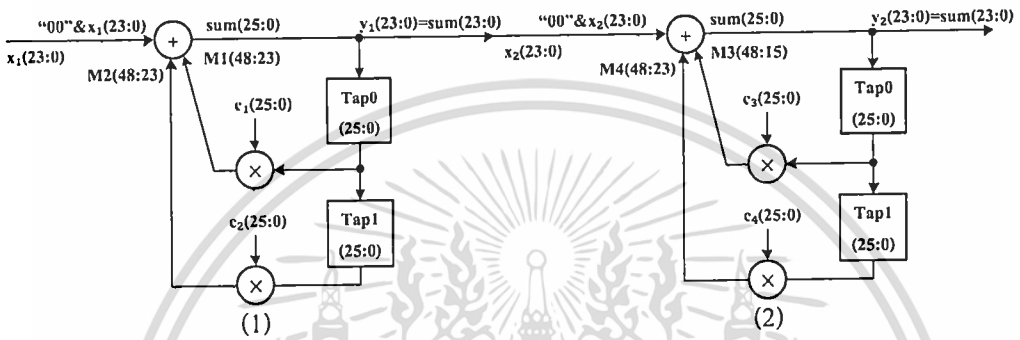
รูปที่ 4.63 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 2^{nd} order filter ขนาด 24 บิต ในรูปเลขฐานสิบ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากัน



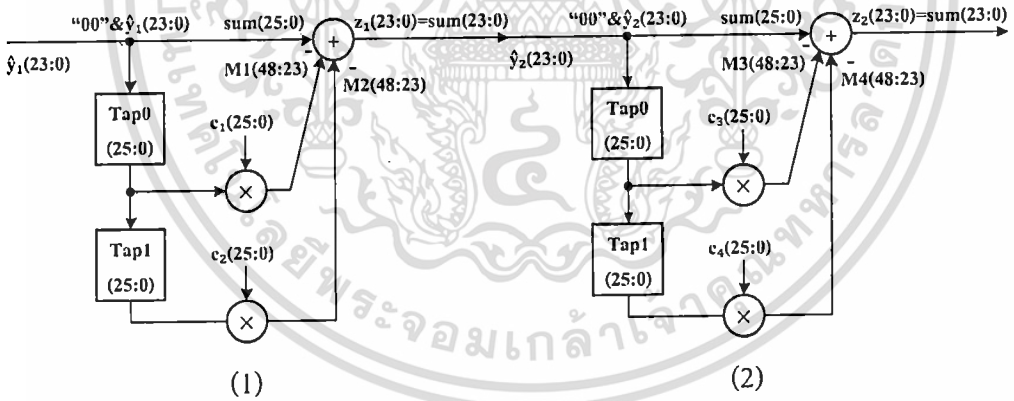
รูปที่ 4.64 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 2^{nd} order filter ขนาด 24 บิต ในรูปเลขฐานสิบ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส ไม่เท่ากัน

4.3.6 วงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสี่ที่มีอินพุตและเอาต์พุตขนาด 24 บิต

โดยโครงสร้างวงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสี่ที่มีอินพุตและเอาต์พุตขนาด 24 บิต แสดงได้ดังรูปที่ 4.65 และ 4.66 ตามลำดับ

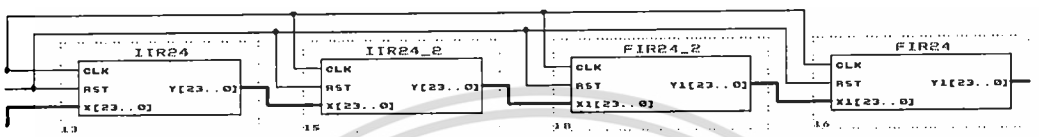


รูปที่ 4.65 โครงสร้างของวงจรเข้ารหัสแบบ IIR 4th order filter ขนาด 24 บิต

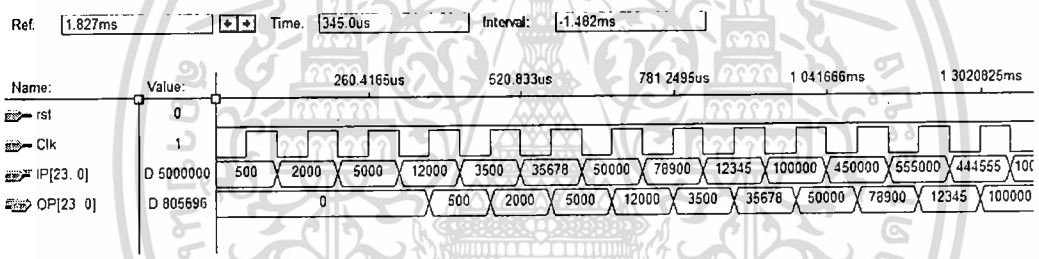


รูปที่ 4.66 โครงสร้างของวงจรถอดรหัสแบบ IIR 4th order filter ขนาด 24 บิต

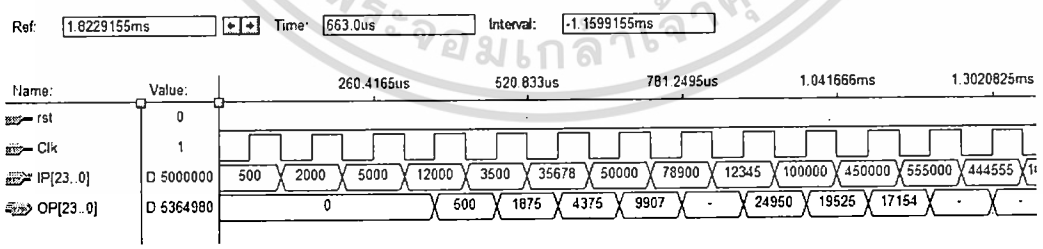
โครงสร้าง RTL Schematic แสดงการเชื่อมต่อวงจรเข้ารหัสแบบ IIR 4th order filter กับวงจรถอดรหัสแบบ FIR 4th order filter ได้ดังรูปที่ 4.67 สามารถจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสขนาด 24 บิต ในรูปแบบเลขฐานสอง โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากันและไม่เท่ากัน ดังรูปที่ 4.68 และ 4.69 ตามลำดับ



รูปที่ 4.67 โครงสร้าง RTL Schematic ของวงจรเข้าและถอดรหัส แบบ 4th order filter ขนาด 24 บิต



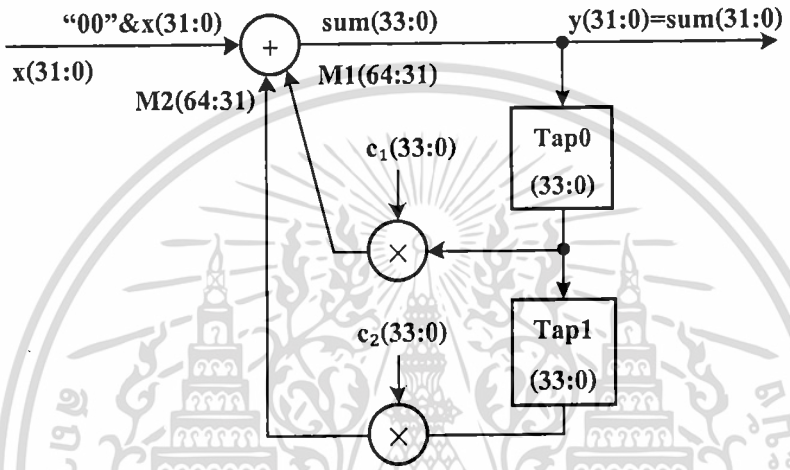
รูปที่ 4.68 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 4th order filter ขนาด 24 บิต ในรูปเลขฐานสิบ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากัน



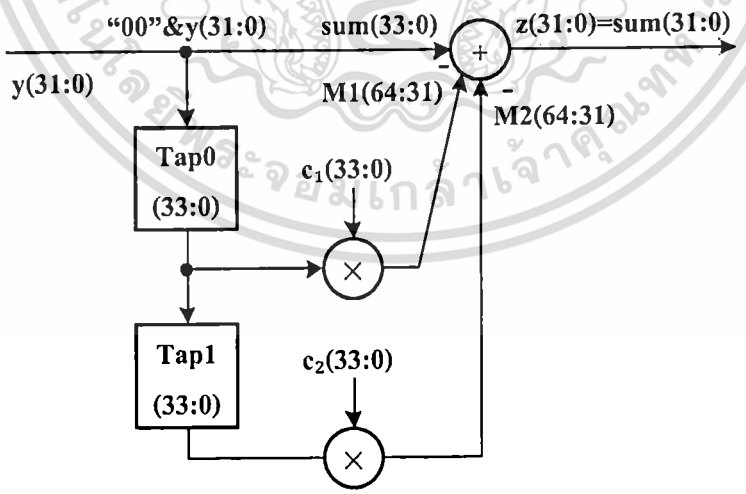
รูปที่ 4.69 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 4th order filter ขนาด 24 บิต ในรูปเลขฐานสิบ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส ไม่เท่ากัน

4.3.7 วงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสองที่มีอินพุตและเอาต์พุตขนาด 32 บิต

โดยโครงสร้างวงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสองที่มีอินพุตและเอาต์พุตขนาด 32 บิต แสดงได้ดังรูปที่ 4.70 และ 4.71 ตามลำดับ

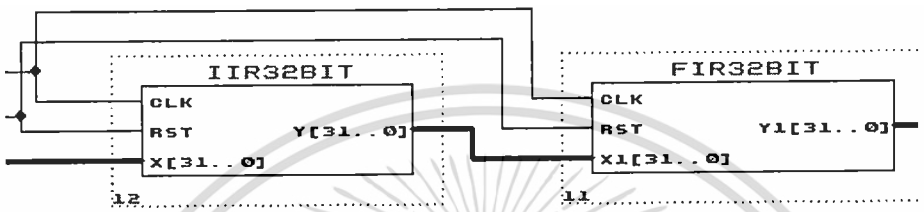


รูปที่ 4.70 โครงสร้างของวงจรเข้ารหัสแบบ IIR 2nd order filter ขนาด 32 บิต

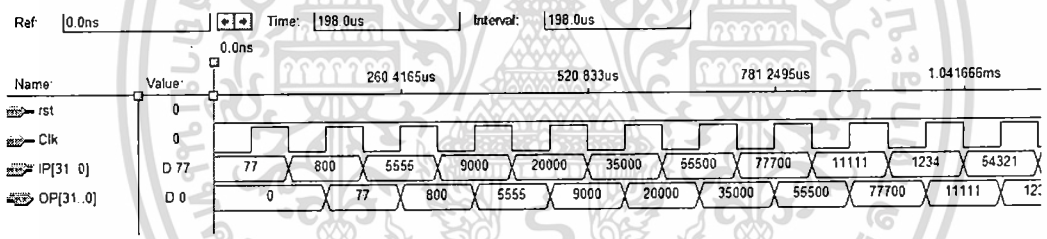


รูปที่ 4.71 โครงสร้างของวงจรถอดรหัสแบบ FIR 2nd order filter ขนาด 32 บิต

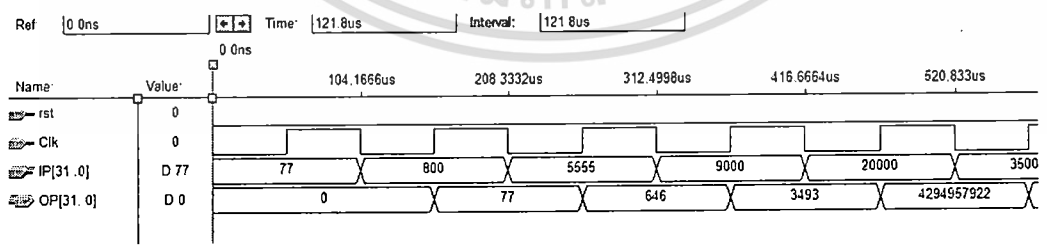
โครงสร้าง RTL Schematic แสดงการเชื่อมต่อวงจรเข้ารหัสแบบ IIR 2^{nd} order filter กับวงจรถอดรหัสแบบ FIR 2^{nd} order filter ได้ดังรูปที่ 4.72 สามารถจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสขนาด 32 บิต ในรูปแบบเลขฐานสอง โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากันและไม่เท่ากัน ดังรูปที่ 4.73 และ 4.74 ตามลำดับ



รูปที่ 4.72 โครงสร้าง RTL Schematic ของวงจรเข้าและถอดรหัสแบบ 2^{nd} order filter ขนาด 32 บิต



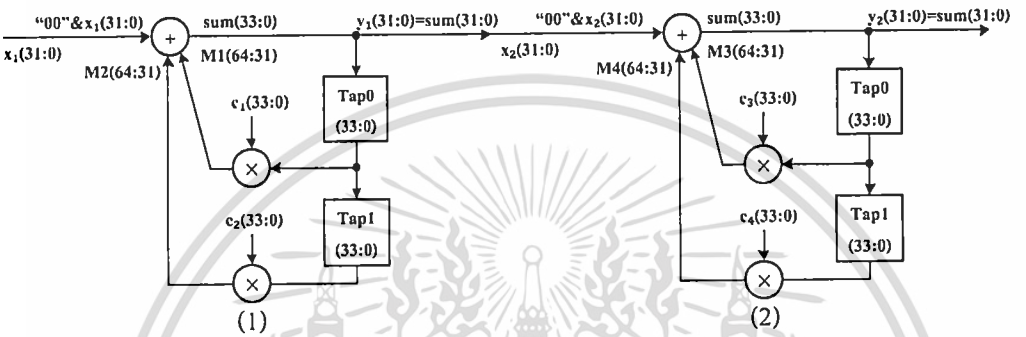
รูปที่ 4.73 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 2^{nd} order filter ขนาด 32 บิตในรูปเลขฐานสิบ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากัน



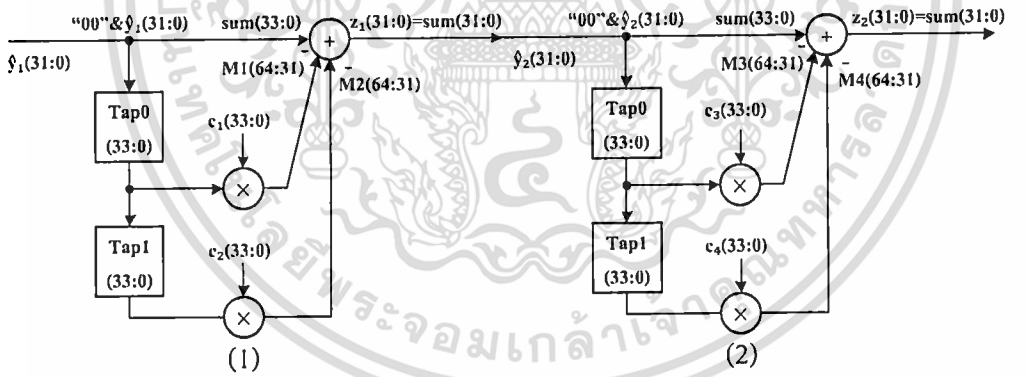
รูปที่ 4.74 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 2^{nd} order filter ขนาด 32 บิตในรูปเลขฐานสิบ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส ไม่เท่ากัน

4.3.8 วงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสี่ที่มีอินพุตและเอาต์พุตขนาด 32 บิต

โดยโครงสร้างวงจรเข้าและถอดรหัสแบบวงจรกรองสัญญาณดิจิตอลอันดับสี่ที่มีอินพุตและเอาต์พุตขนาด 32 บิต แสดงได้ดังรูปที่ 4.75 และ 4.76 ตามลำดับ

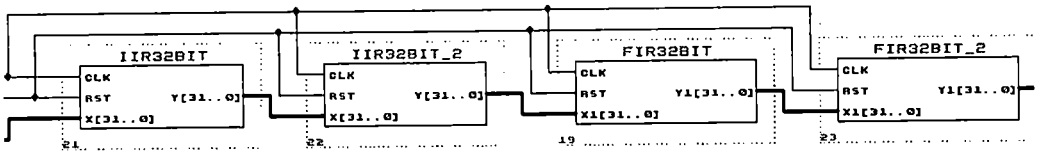


รูปที่ 4.75 โครงสร้างของวงจรเข้ารหัสแบบ IIR 4th order filter ขนาด 32 บิต

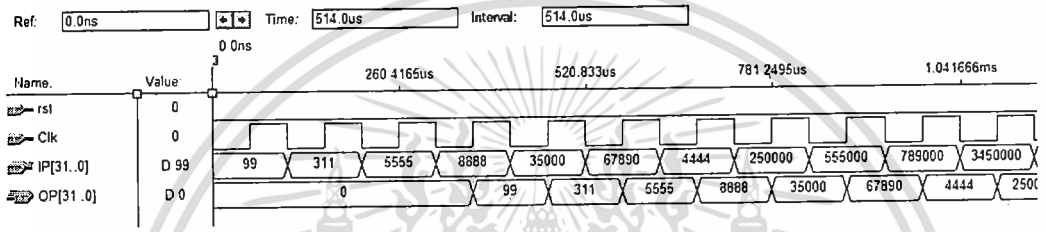


รูปที่ 4.76 โครงสร้างของวงจรถอดรหัสแบบ IIR 4th order filter ขนาด 32 บิต

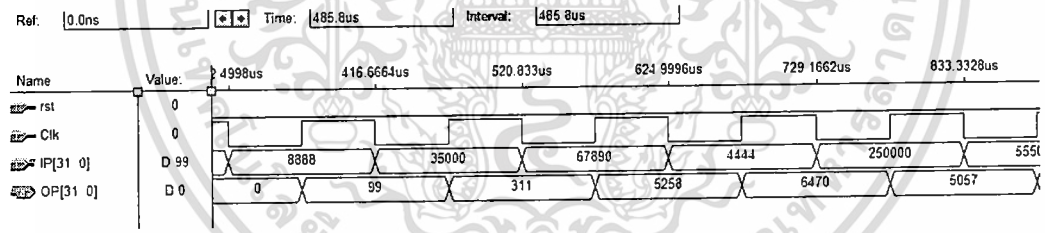
โครงสร้าง RTL Schematic แสดงการเชื่อมต่อวงจรเข้ารหัสแบบ IIR 4th order filter กับวงจรถอดรหัสแบบ FIR 4th order filter ได้ดังรูปที่ 4.77 สามารถจำลองการทำงานของวงจรเข้ารหัสและถอดรหัสขนาด 32 บิต ในรูปแบบเลขฐานสอง โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากันและไม่เท่ากัน ดังรูปที่ 4.78 และ 4.79 ตามลำดับ



รูปที่ 4.77 โครงสร้าง RTL Schematic ของวงจรเข้าและถอดรหัส
แบบ 4th order filter ขนาด 32 บิต



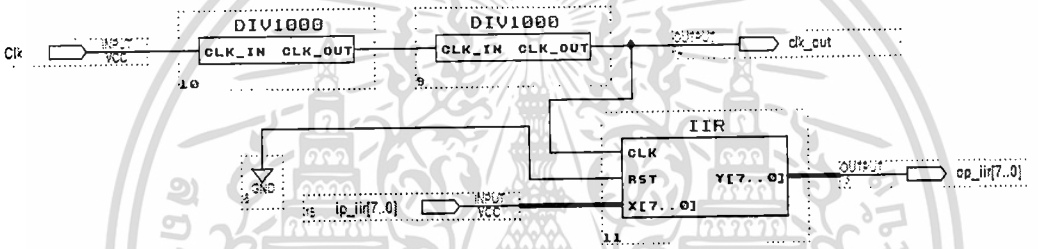
รูปที่ 4.78 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 4th order filter ขนาด 32 บิต
ในรูปเลขฐานสิบโดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส เท่ากัน



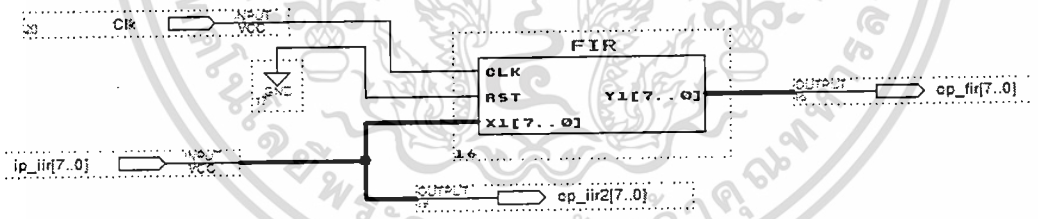
รูปที่ 4.79 ผลการจำลองการทำงานของวงจรเข้าและถอดรหัสแบบ 4th order filter ขนาด 32 บิต
ในรูปเลขฐานสิบโดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัส ไม่เท่ากัน

4.4 ผลการทดลองการเข้ารหัส-ถอดรหัสผ่านการรับส่งข้อมูลแบบขนานด้วย FPGA

ในการทดลองการเข้ารหัส-ถอดรหัสผ่านการรับส่งแบบขนาน กำหนดให้อินพุตขนาด 8 บิต ป้อนข้อมูลโดยผ่าน DIP switch ไปยังวงจรเข้ารหัสของ FPGA1 ซึ่งอินพุตที่ผ่านการเข้ารหัสแล้ว (cipher) จะแสดงผลผ่าน LED1 จากนั้นจะส่งข้อมูลแบบขนานไปยังวงจรถอดรหัสของ FPGA2 และแสดงผลของเอาท์พุตที่ถูกถอดรหัสแล้วออกมาผ่าน LED2 ดังรูปที่ 1.1 โดยโครงสร้าง RTL Schematic ที่ใช้ในการเข้ารหัส-ถอดรหัสผ่านการรับส่งข้อมูลแบบขนานด้วย FPGA แสดงดังรูปที่ 4.80 และ 4.81



รูปที่ 4.80 โครงสร้าง RTL Schematic ของวงจรเข้ารหัสผ่านการรับส่งแบบขนาน

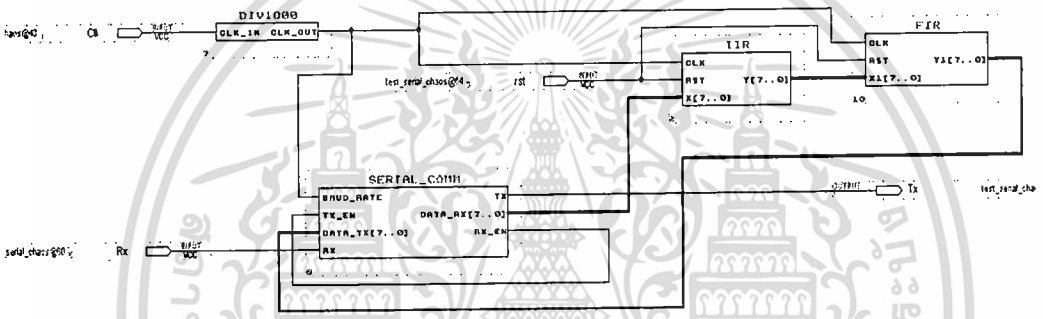


รูปที่ 4.81 โครงสร้าง RTL Schematic ของวงจรถอดรหัสผ่านการรับส่งแบบขนาน

ในการส่งข้อมูลขนาด 8 บิตด้วย FPGA1 จาก DIP switch (ip_iir[7..0]) ซึ่งจะผ่านวงจรเข้ารหัส จากนั้นจะส่งไปยังหลอดไฟ LED1 ที่เป็น cipher (op_iir[7..0]) ใช้ตรวจดูความเป็นไดนามิคของสัญญาณจากวงจรเข้ารหัส และจะส่งไปยัง FPGA2 ให้กับวงจรถอดรหัส LED2 เพื่อถอดรหัสสัญญาณและนำไปแสดงผลหลอดไฟ LED2 เพื่อที่จะให้ตาคนสามารถมองเห็นการแสดงผลของหลอดไฟได้ทันจะต้องหารความถี่ให้ลดลงเหลือ 9.6 Hz

4.5 ผลการทดลองในการรับส่งข้อมูลอนุกรมเพื่อเข้ารหัส-ถอดรหัสโดยใช้คอมพิวเตอร์เครื่องเดียว

ในการทดลองการรับส่งข้อมูลอนุกรมเพื่อเข้ารหัส-ถอดรหัสโดยใช้คอมพิวเตอร์เครื่องเดียว จะทดลองส่งข้อมูลตัวอักษรจากเครื่องคอมพิวเตอร์ส่งผ่านการรับส่งแบบอนุกรมไปยัง FPGA เพื่อทำการเข้ารหัส-ถอดรหัส จากนั้นข้อมูลจะถูกส่งกลับมาที่คอมพิวเตอร์เครื่องเดิม ดังรูปที่ 1.2 โดยมีโครงสร้าง RTL Schematic ภายใน FPGA แสดงดังรูปที่ 4.82

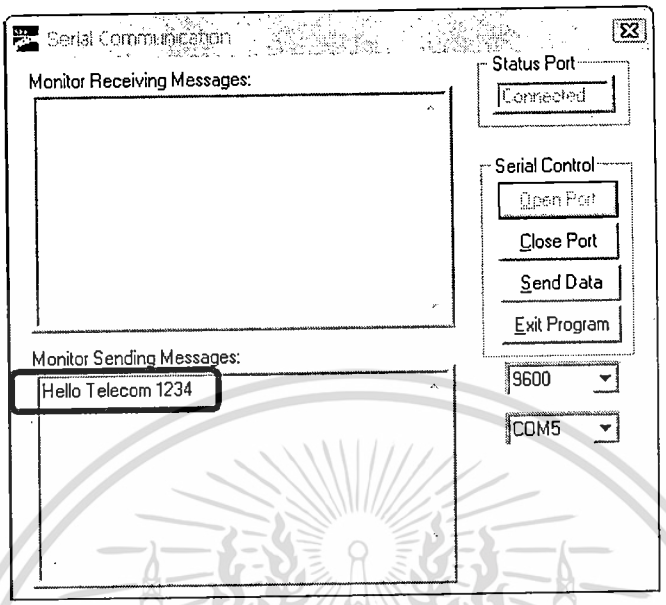


รูปที่ 4.82 โครงสร้าง RTL Schematic ของวงจรเข้ารหัสถอดรหัสเชื่อมต่อกับวงจรสื่อสารอนุกรม

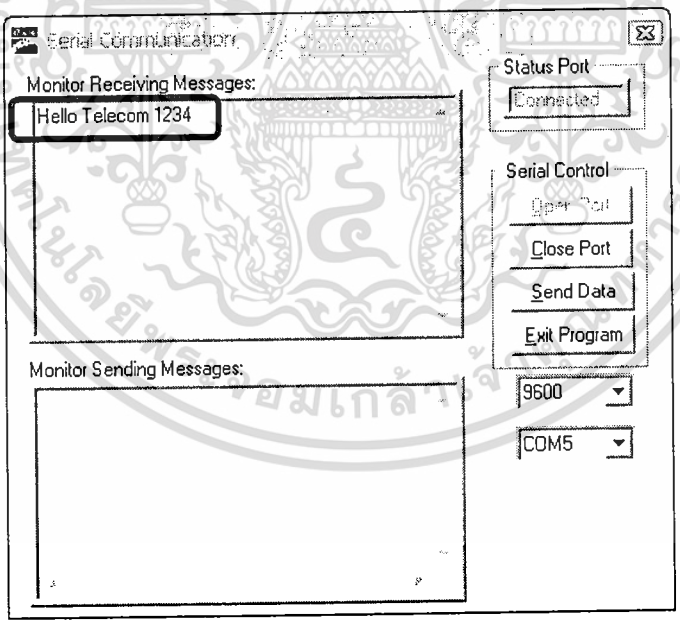
4.5.1 ผลการทดสอบการส่งข้อมูลตัวอักษรผ่านโปรแกรมรับส่งข้อมูลอนุกรม

4.5.1.1 กำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัสเท่ากัน

จากรูปที่ 4.83 และ 4.84 จะเห็นได้ว่าเมื่อค่าสัมประสิทธิ์มีค่าเท่ากันทั้งวงจรเข้าและถอดรหัส เมื่อทำการส่งข้อมูลก็จะสามารถรับข้อมูลได้ถูกต้อง

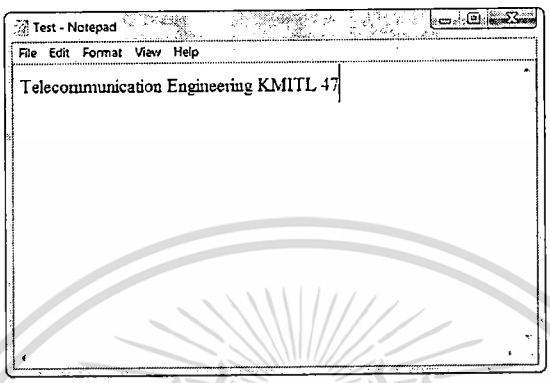


รูปที่ 4.83 หน้าต่างโปรแกรมส่งข้อมูลตัวอักษรที่กำหนดให้ค่าสัมประสิทธิ์ของวงจรเท่ากัน

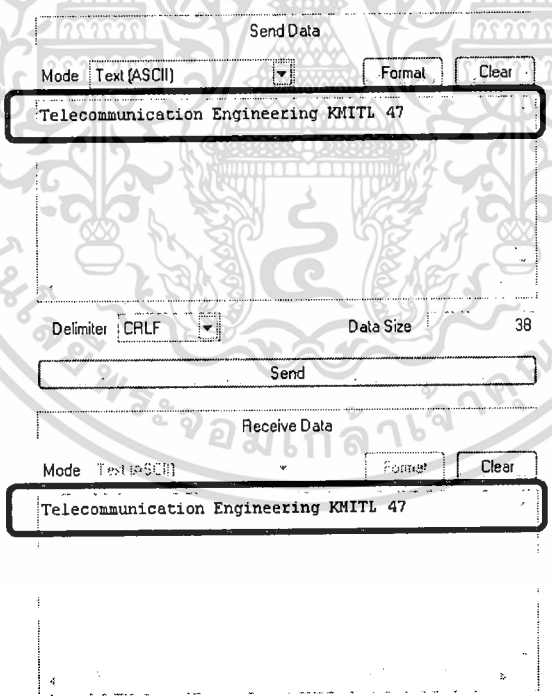


รูปที่ 4.84 หน้าต่างโปรแกรมรับข้อมูลตัวอักษรที่กำหนดให้ค่าสัมประสิทธิ์ของวงจรเท่ากัน

และเมื่อทำการทดลองส่งข้อมูลที่อยู่ในรูปของไฟล์ .text ดังรูปที่ 4.85 ผ่านโปรแกรมก็จะได้รับข้อมูลที่ถูกต้อง ดังรูปที่ 4.86



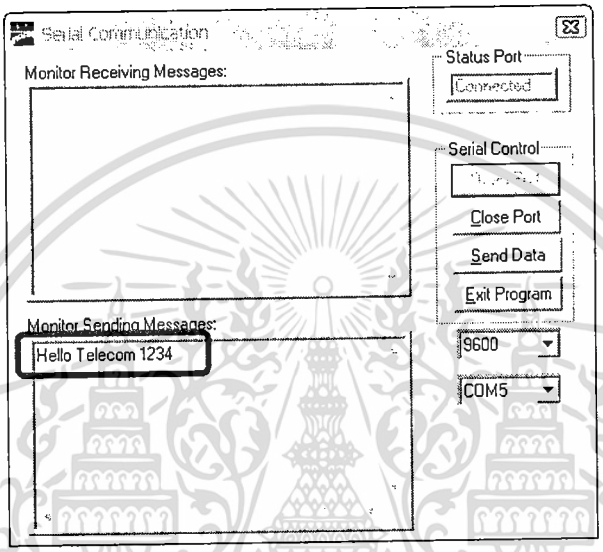
รูปที่ 4.85 ข้อมูลที่อยู่ในรูปของไฟล์ .text ที่ใช้ในการส่งผ่านโปรแกรมเมื่อกำหนดให้ค่าสัมประสิทธิ์ของวงจรเท่ากัน



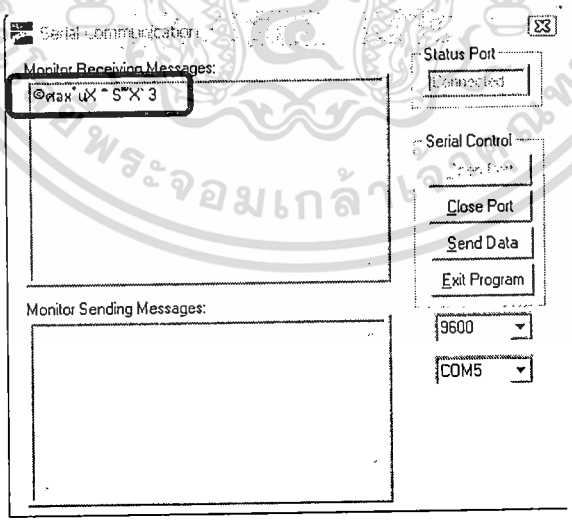
รูปที่ 4.86 หน้าต่างโปรแกรมที่ใช้ในการส่งและรับข้อมูลที่อยู่ในรูปของไฟล์ .text เมื่อกำหนดให้ค่าสัมประสิทธิ์ของวงจรเท่ากัน

4.5.1.2 กำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและถอดรหัสไม่เท่ากัน

จากรูปที่ 4.87 และ 4.88 จะเห็นได้ว่าเมื่อค่าสัมประสิทธิ์มีค่าไม่เท่ากันของวงจรเข้าและถอดรหัส เมื่อทำการส่งข้อมูลจะไม่สามารถรับข้อมูลได้ถูกต้อง

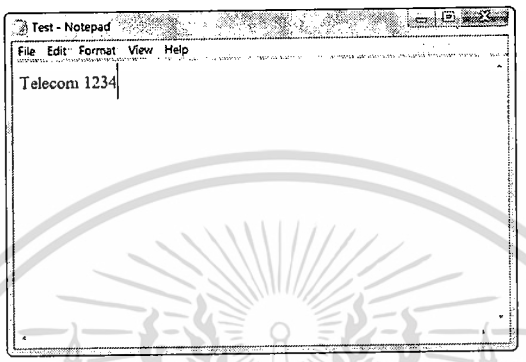


รูปที่ 4.87 หน้าต่างโปรแกรมส่งข้อมูลตัวอักษรที่กำหนดให้ค่าสัมประสิทธิ์ของวงจรไม่เท่ากัน

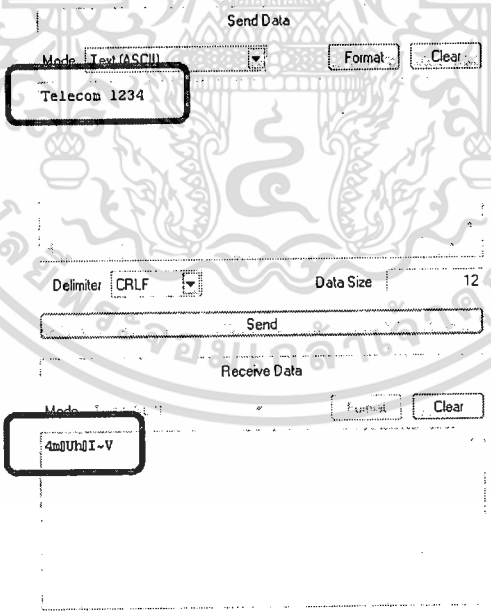


รูปที่ 4.88 หน้าต่างโปรแกรมรับข้อมูลตัวอักษรที่กำหนดให้ค่าสัมประสิทธิ์ของวงจรไม่เท่ากัน

เมื่อทดลองส่งข้อมูลที่อยู่ในรูปของไฟล์ .text ดังรูปที่ 4.89 ผ่านโปรแกรม โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากันก็จะไม่สามารถได้รับข้อมูลที่ถูกต้อง ดังรูปที่ 4.90



รูปที่ 4.89 ข้อมูลที่อยู่ในรูปของไฟล์ .text ที่ใช้ในการส่งผ่านโปรแกรม
เมื่อกำหนดค่าสัมประสิทธิ์ของวงจรไม่เท่ากัน



รูปที่ 4.90 หน้าต่างโปรแกรมที่ใช้ในการส่งและข้อมูลที่อยู่ในรูปของไฟล์ .text
เมื่อกำหนดค่าสัมประสิทธิ์ของวงจรไม่เท่ากัน

4.3.1.2 กำหนดให้ค่าสัมประสิทธิ์ของวงจรถอดรหัสมีความใกล้เคียงกับวงจรเข้ารหัสในระดับทศนิยม

ทดลองส่งข้อมูลตัวอักษรขนาด 8 บิต ด้วยโปรแกรมผ่านการรับส่งข้อมูลแบบอนุกรมไปยังวงจรเข้าและถอดรหัส โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรถอดรหัสมีความใกล้เคียงต่างๆกับวงจรเข้ารหัสในระดับทศนิยม แล้วนำผลของอินพุตและเอาต์พุตที่ได้มาเปรียบเทียบกันในระบบเลขฐานสอง, ฐานสิบ และ ฐานสิบหก ดังรูปที่ 4.91 ถึงรูปที่ 4.97

	Character	Hex	Dec
Input	abcd	61 62 63 64	97 98 99 100
output	ÍÖÐ	CD D6 17 D0	205 214 23 208

	Bin
Input	0110 0001 , 0110 0010 , 0110 0011 , 0110 0100
output	1100 1101 , 1101 0110 , 0001 0111 , 1101 0000

รูปที่ 4.91 ข้อมูลอินพุตและเอาต์พุตเมื่อกำหนดให้ค่าสัมประสิทธิ์วงจรเข้ารหัสคือ $c_1 = 4$ และ $c_2 = -1$ และค่าสัมประสิทธิ์ของวงจรถอดรหัสคือ $c_1 = 3.5$ และ $c_2 = -1$

	Character	Hex	Dec
Input	abcd	61 62 63 64	97 98 99 100
output	D¥Æ◆?	44 a5 c6 8f	68 165 198 143

	Bin
Input	0110 0001 , 0110 0010 , 0110 0011 , 0110 0100
output	0100 0100 , 1010 0101 , 1100 0110 , 1000 1111

รูปที่ 4.92 ข้อมูลอินพุตและเอาต์พุตเมื่อกำหนดให้ค่าสัมประสิทธิ์วงจรเข้ารหัสคือ $c_1 = 4$ และ $c_2 = -1$ และค่าสัมประสิทธิ์ของวงจรถอดรหัสคือ $c_1 = 3.75$ และ $c_2 = -1$

	Character	Hex	Dec
Input	abcd	61 62 63 64	97 98 99 100
output	sëuc	73 eb 75 63	115 235 117 099

	Bin
Input	0110 0001 , 0110 0010 , 0110 0011 , 0110 0100
output	0111 0011 , 1110 1011 , 0111 0101 , 0110 0011

รูปที่ 4.93 ข้อมูลอินพุตและเอาต์พุตเมื่อกำหนดให้ค่าสัมประสิทธิ์วงจรเข้ารหัสคือ $c_1 = 4$ และ $c_2 = -1$ และค่าสัมประสิทธิ์ของวงจรถอดรหัสคือ $c_1 = 3.875$ และ $c_2 = -1$

	Character	Hex	Dec
Input	abcd	61 62 63 64	97 98 99 100
output	ibde	69 62 64 65	105 98 100 101

	Bin
Input	0110 0001 , 0110 0010 , 0110 0011 , 0110 0100
output	0110 1001 , 0110 0010 , 0110 0100 , 0110 0101

รูปที่ 4.94 ข้อมูลอินพุตและเอาต์พุตเมื่อกำหนดให้ค่าสัมประสิทธิ์วงจรเข้ารหัสคือ $c_1 = 4$ และ $c_2 = -1$ และค่าสัมประสิทธิ์ของวงจรถอดรหัสคือ $c_1 = 3.9375$ และ $c_2 = -1$

	Character	Hex	Dec
Input	abcd	61 62 63 64	97 98 99 100
output	cded	63 64 65 64	99 100 101 100

	Bin
Input	0110 0001 , 0110 0010 , 0110 0011 , 0110 0100
output	0110 0011 , 0110 0100 , 0110 0101 , 0110 0100

รูปที่ 4.95 ข้อมูลอินพุตและเอาต์พุตเมื่อกำหนดให้ค่าสัมประสิทธิ์วงจรเข้ารหัสคือ $c_1 = 4$ และ $c_2 = -1$ และค่าสัมประสิทธิ์ของวงจรถอดรหัสคือ $c_1 = 3.96875$ และ $c_2 = -1$

	Character	Hex	Dec
Input	abcd	61 62 63 64	97 98 99 100
output	gfeb	67 66 65 62	103 102 101 98

	Bin
Input	0110 0001 , 0110 0010 , 0110 0011 , 0110 0100
output	0110 0111 , 0110 0110 , 0110 0101 , 0110 0010

รูปที่ 4.96 ข้อมูลอินพุตและเอาต์พุตเมื่อกำหนดให้ค่าสัมประสิทธิ์วงจรเข้ารหัสคือ $c_1 = 4$ และ $c_2 = -1$ และค่าสัมประสิทธิ์ของวงจรถอดรหัสคือ $c_1 = 3.984375$ และ $c_2 = -1$

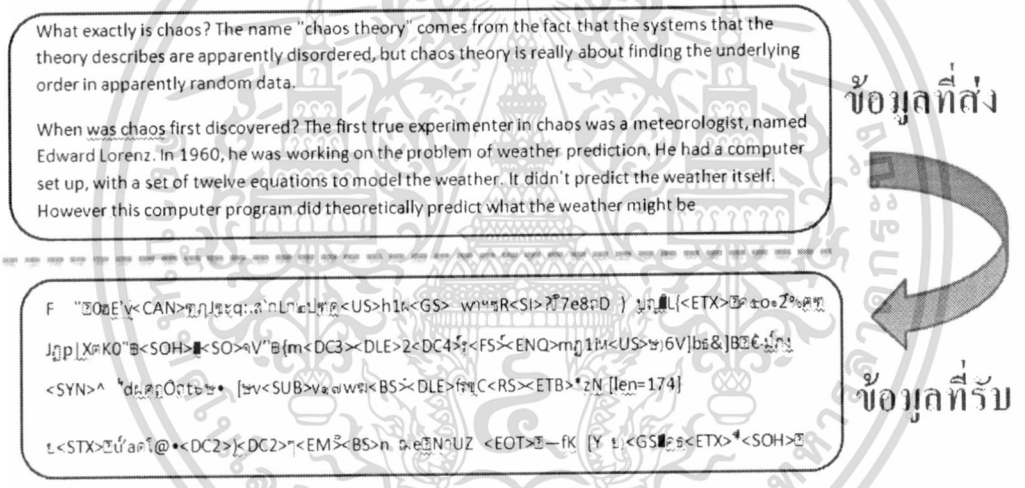
	Character	Hex	Dec
Input	abcd	61 62 63 64	97 98 99 100
output	cedg	63 65 64 67	99 101 100 103

	Bin
Input	0110 0001 , 0110 0010 , 0110 0011 , 0110 0100
output	0110 0011 , 0110 0101 , 0110 0100 , 0110 0111

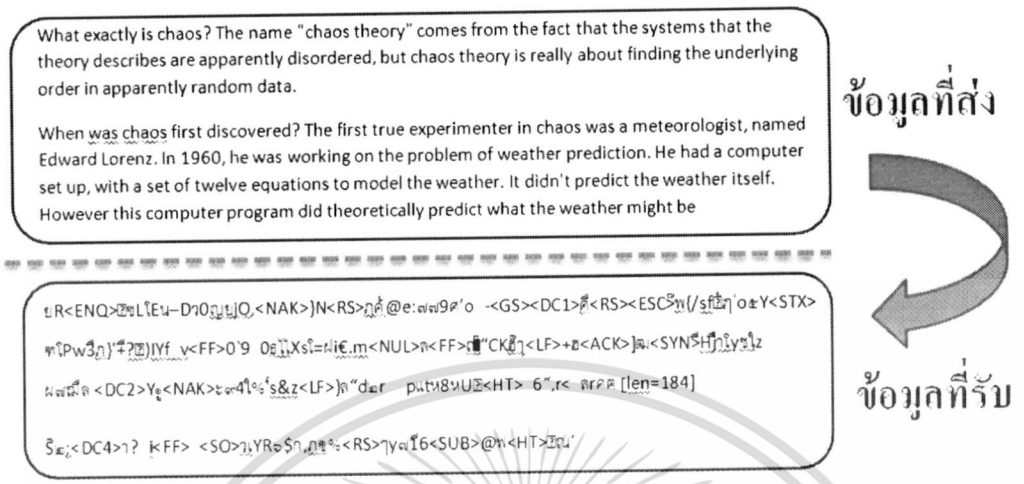
รูปที่ 4.97 ข้อมูลอินพุตและเอาต์พุตเมื่อกำหนดให้ค่าสัมประสิทธิ์วงจรเข้ารหัสคือ $c_1 = 4$ และ $c_2 = -1$ และค่าสัมประสิทธิ์ของวงจรถอดรหัสคือ $c_1 = 3.9921875$ และ $c_2 = -1$

จากการสังเกตการทดลองเพิ่มค่าความละเอียดของสัมประสิทธิ์วงจรถอดรหัส มีผลต่อข้อมูลที่ถูกถอดรหัสออกมาเมื่อเพิ่มความละเอียดจุดทศนิยมของค่าสัมประสิทธิ์ตัวถอดรหัส จะทำให้ระยะห่างของข้อมูลถอดรหัส ใกล้เคียงข้อมูลต้นฉบับมากขึ้น และเมื่อเพิ่มค่าสัมประสิทธิ์ให้ละเอียดเพิ่มขึ้นให้มีค่าตั้งแต่ 3.9 ขึ้นไป จะทำให้ได้ข้อมูลที่ถอดรหัสออกมาได้ผิดเพี้ยนไปจากข้อมูลต้นฉบับ 0 – 8 ตำแหน่งของข้อมูลตัวอักษรในตารางรหัส ASCII Code

ทดลองส่งข้อมูลตัวอักษรแบบเป็นประโยคยาว 2 ครั้ง โดยกำหนดค่าสัมประสิทธิ์ของวงจรให้มีค่าใกล้เคียงกันมากๆ ในระดับจุดทศนิยม แล้วนำผลที่ถอดรหัสทั้ง 2 ครั้งมาเปรียบเทียบกัน ดังรูปที่ 4.98 และ 4.99



รูปที่ 4.98 ทดลองส่งข้อมูลตัวอักษรเป็นประโยคยาว ครั้งที่ 1 เมื่อกำหนดให้ค่าสัมประสิทธิ์วงจรเข้ารหัสคือ $c_1 = 4$ และ $c_2 = -1$ และค่าสัมประสิทธิ์ของวงจรถอดรหัส
คือ $c_1 = 3.9921875$ และ $c_2 = -1$



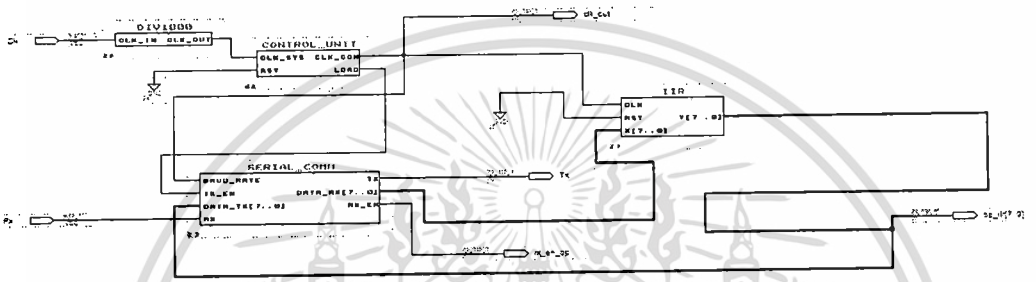
รูปที่ 4.99 ทดลองส่งข้อมูลตัวอักษรเป็นประโยคยาว ครั้งที่ 2 เมื่อกำหนดให้ค่าสัมประสิทธิ์วงจรเข้ารหัสคือ $c_1 = 4$ และ $c_2 = -1$ และค่าสัมประสิทธิ์ของวงจรถอดรหัสคือ $c_1 = 3.9921875$ และ $c_2 = -1$

จากการสังเกตการทดลองส่งข้อมูลตัวอักษรเป็นประโยคข้อความยาวๆ จะได้ข้อมูลหลังผ่านวงจรถอดรหัสแล้วเป็นข้อมูลที่ผิดเพี้ยนไป ไม่สามารถคาดเดารูปแบบ หรือทำนายล่วงหน้าได้ แม้จะใช้ค่าสัมประสิทธิ์วงจรถอดรหัสที่ใกล้เคียงกับวงจรเข้ารหัสมากที่สุดก็ตาม และการส่งข้อมูลซ้ำเดิมอีกครั้ง ก็เปลี่ยนแปลงรูปแบบผลลัพธ์ไปจากเดิมไปทุกครั้งที่ส่ง

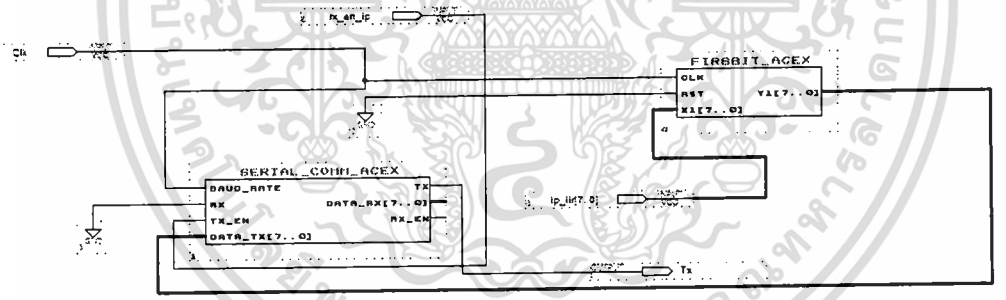
4.6 ผลการทดลองในการรับส่งข้อมูลอนุกรมเพื่อเข้ารหัส-ถอดรหัสโดยใช้คอมพิวเตอร์ 2 เครื่อง

จากรูปที่ 4.4 เมื่อคอมพิวเตอร์ Alice ส่งข้อมูลตัวอักษรผ่านทางพอร์ตสื่อสารอนุกรมไปยังอุปกรณ์ FPGA จะผ่านวงจรสื่อสารอนุกรม (Serial Communication) ไปยังวงจรเข้ารหัส (Encoder) จากนั้นจะได้ข้อมูลที่ถูกเข้ารหัสแล้วส่งยังวงจรถอดรหัส (Decoder) ของฝั่งรับ วงจรนี้จะทำการถอดรหัสลับของข้อมูลและส่งข้อมูลผ่านไปยังวงจรสื่อสารอนุกรม (Serial Communication) เพื่อส่งข้อมูลผ่านไปยังพอร์ตสื่อสารอนุกรมของคอมพิวเตอร์ Bob ที่เป็นฝั่งรับ

ในทางกลับกันเครื่องคอมพิวเตอร์ Bob ต้องการส่งข้อมูลกลับไปยังเครื่องคอมพิวเตอร์ Alice ก็จะผ่านขั้นตอนการทำงานข้างต้นเหมือนเดิมเช่นเดียวกัน โดยระหว่างคอมพิวเตอร์ Alice และคอมพิวเตอร์ Bob กำลังรับส่งข้อมูล จะมีคอมพิวเตอร์ Charlie เข้ามาดักจับข้อมูล เพื่อทดสอบความแข็งแกร่งของข้อมูลเมื่อถูกเข้ารหัสโดยมีโครงสร้าง RTL Schematic ของวงจรเข้ารหัส (IIR) และวงจรถอดรหัส (FIR) ดังรูปที่ 4.100 และ 4.101



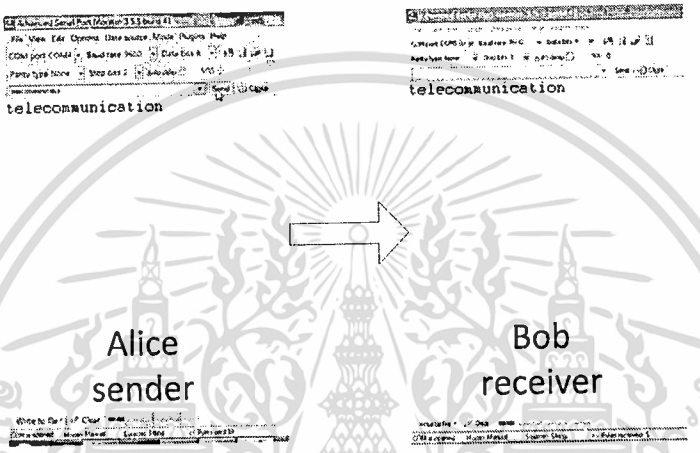
รูปที่ 4.100 โครงสร้าง RTL Schematic ของวงจรเข้ารหัสผ่านการรับส่งข้อมูลแบบอนุกรม



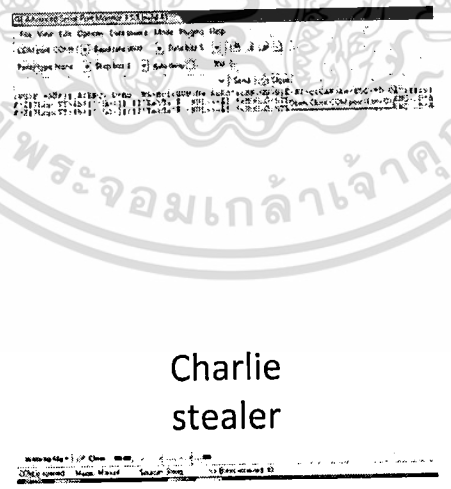
รูปที่ 4.101 โครงสร้าง RTL Schematic ของวงจรถอดรหัสผ่านรับส่งข้อมูลแบบอนุกรม

4.6.1 ผลการทดลองการส่งข้อมูลตัวอักษรโดยกำหนดให้ค่าสัมประสิทธิ์เท่ากัน

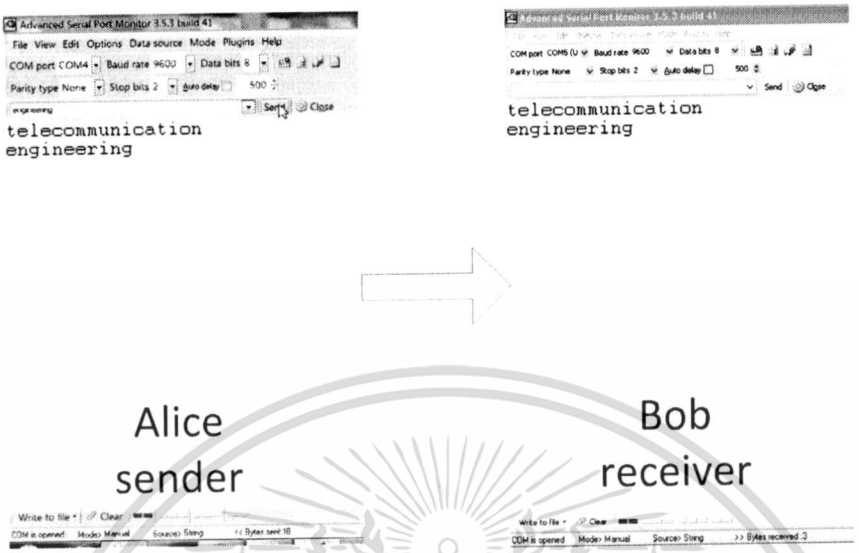
จากรูปที่ 4.102 ถึงรูปที่ 4.105 เป็นผลการทดลองการส่งข้อมูลตัวอักษรทั้ง 2 ครั้ง โดย Alice ทำการส่งข้อมูลไปยัง Bob และมี Charlie ดักจับข้อมูล ซึ่งเมื่อกำหนดให้ค่าสัมประสิทธิ์มีค่าเท่ากัน Bob ก็จะสามารถรับข้อมูลได้ถูกต้อง และข้อมูลที่ Charlie ดักจับได้ก็มีความผิดเพี้ยน



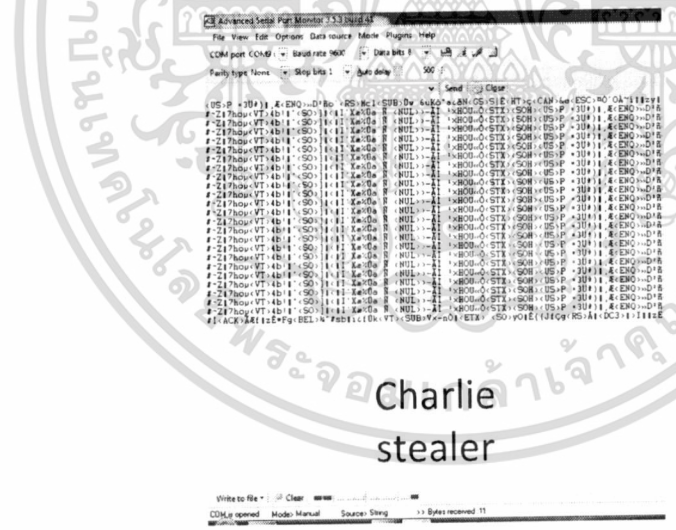
รูปที่ 4.102 ผลการทดลองการส่งข้อมูลตัวอักษรครั้งที่ 1
โดยกำหนดให้ค่าสัมประสิทธิ์เท่ากัน



รูปที่ 4.103 ผลการทดลองดักจับการส่งข้อมูลตัวอักษรครั้งที่ 1
โดยกำหนดให้ค่าสัมประสิทธิ์เท่ากัน



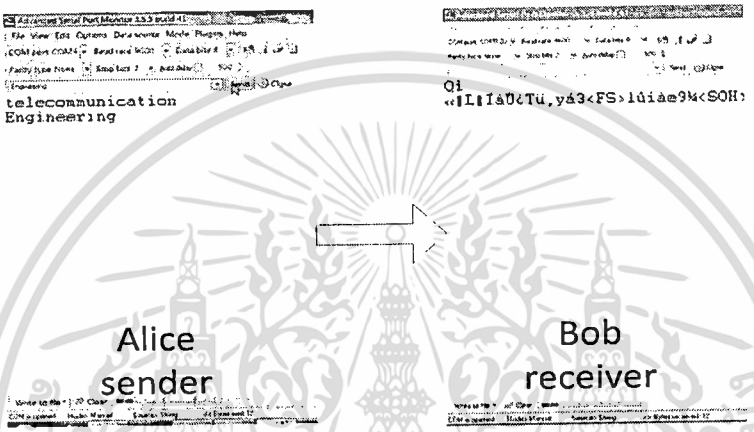
รูปที่ 4.104 ผลการทดลองการส่งข้อมูลตัวอักษรครั้งที่ 2
โดยกำหนดให้ค่าสัมประสิทธิ์เท่ากัน



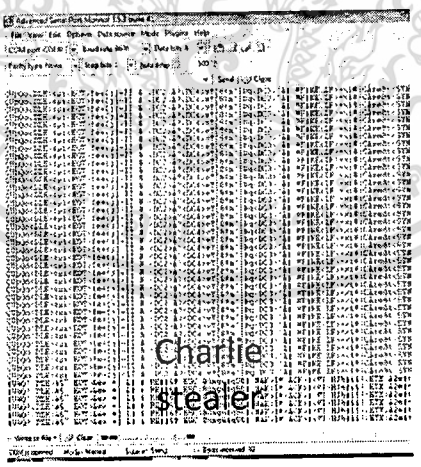
รูปที่ 4.105 ผลการทดลองดักจับการส่งข้อมูลตัวอักษรครั้งที่ 2
โดยกำหนดให้ค่าสัมประสิทธิ์เท่ากัน

4.6.2 ผลการทดลองการส่งข้อมูลตัวอักษรโดยกำหนดให้ค่าสัมประสิทธิ์ไม่เท่ากัน

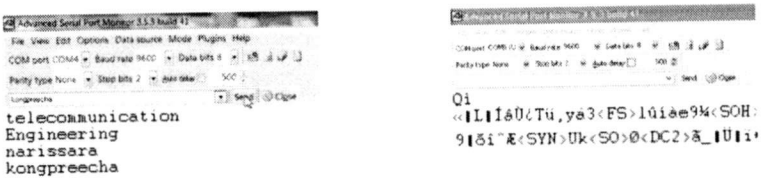
จากรูปที่ 4.106 ถึงรูปที่ 4.109 เป็นผลการทดลองการส่งข้อมูลตัวอักษรทั้ง 2 ครั้ง โดย Alice ทำการส่งข้อมูลไปยัง Bob และมี Charlie ดักจับข้อมูล ซึ่งเมื่อกำหนดให้ค่าสัมประสิทธิ์มีค่าไม่เท่ากัน Bob ก็จะไม่สามารถรับข้อมูลได้ถูกต้อง และข้อมูลที่ Charlie ดักจับได้ก็มีความผิดเพี้ยน



รูปที่ 4.106 ผลการทดลองการส่งข้อมูลตัวอักษรครั้งที่ 1
โดยกำหนดให้ค่าสัมประสิทธิ์ไม่เท่ากัน



รูปที่ 4.107 ผลการทดลองดักจับการส่งข้อมูลตัวอักษรครั้งที่ 1
โดยกำหนดให้ค่าสัมประสิทธิ์ไม่เท่ากัน



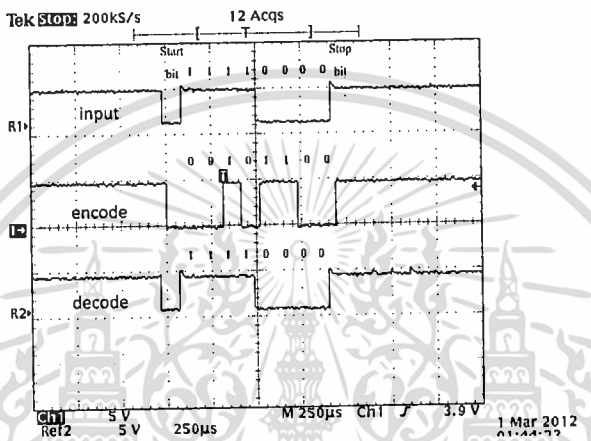
รูปที่ 4.108 ผลการทดลองการส่งข้อมูลตัวอักษรครั้งที่ 2
โดยกำหนดให้ค่าสัมประสิทธิ์ไม่เท่ากัน

รูปที่ 4.109 ผลการทดลองดักจับการส่งข้อมูลตัวอักษรครั้งที่ 2
โดยกำหนดให้ค่าสัมประสิทธิ์ไม่เท่ากัน

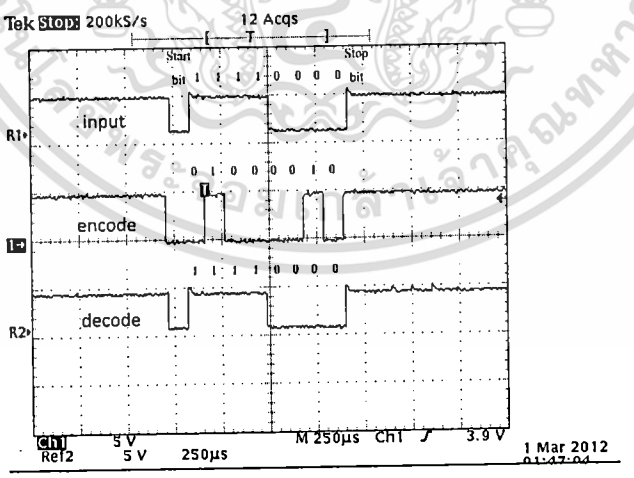
และจากการสังเกตที่เครื่องคอมพิวเตอร์ Charlie ซึ่งได้ดักจับข้อมูลที่ผ่านการเข้ารหัสมาแล้วนั้น จะเป็นข้อมูลลับที่ไม่ซ้ำค่า (Dynamic) คือจะเปลี่ยนค่าไปตามเวลาและอินพุตที่ได้รับเข้ามา ซึ่งแสดงถึงความเป็นเคออส

4.7 ผลการวัดสัญญาณทางไฟฟ้าเมื่อส่งข้อมูลแบบอนุกรมระหว่างวงจรเข้ารหัสและถอดรหัสบนอุปกรณ์ FPGA

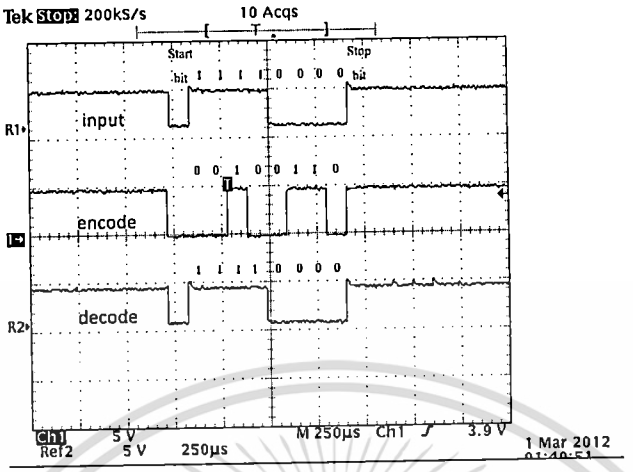
4.7.1 เมื่อกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัสมีค่าเท่ากัน



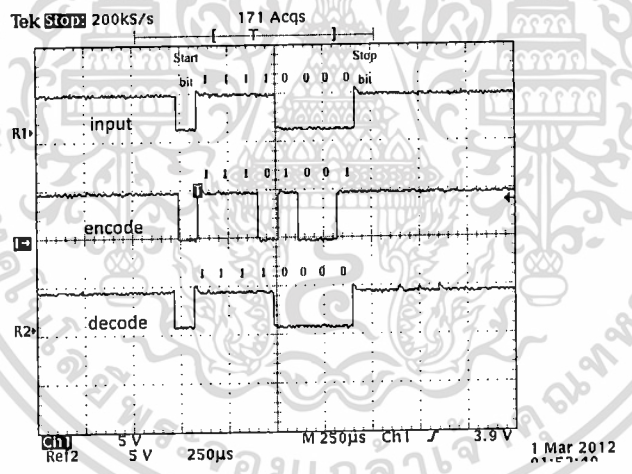
รูปที่ 4.110 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส โดยมีอินพุตคือ [0 0 0 0 1 1 1 1] ครั้งที่ 1



รูปที่ 4.111 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส โดยมีอินพุตคือ [0 0 0 0 1 1 1 1] ครั้งที่ 2

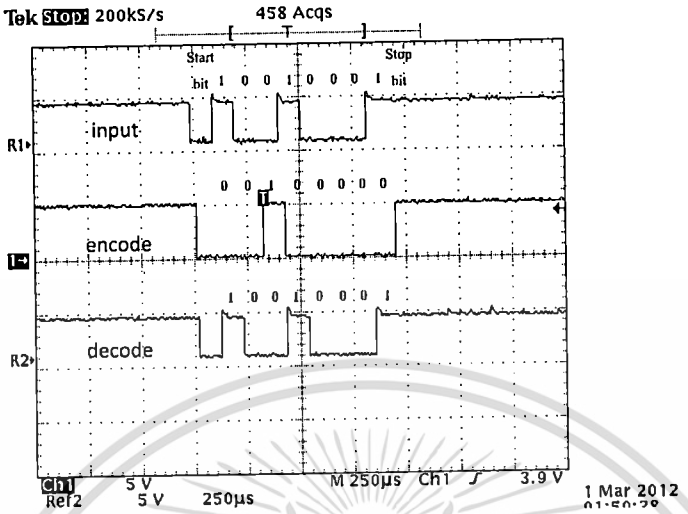


รูปที่ 4.112 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส
โดยมีอินพุตคือ [0 0 0 0 1 1 1 1] ครั้งที่ 3

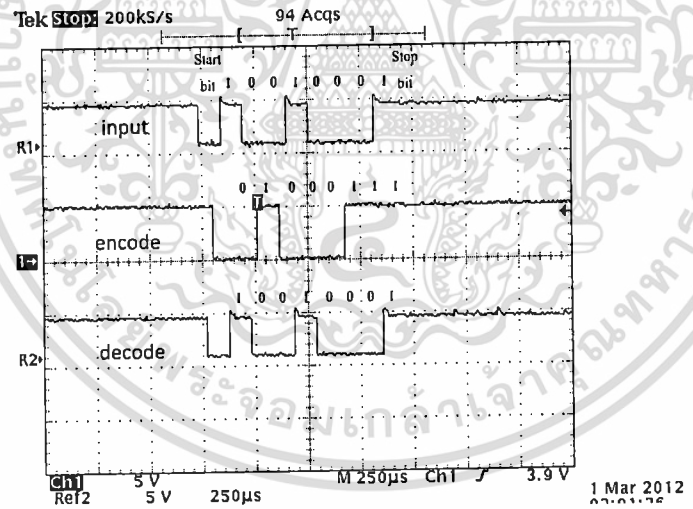


รูปที่ 4.113 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส
โดยมีอินพุตคือ [0 0 0 0 1 1 1 1] ครั้งที่ 4

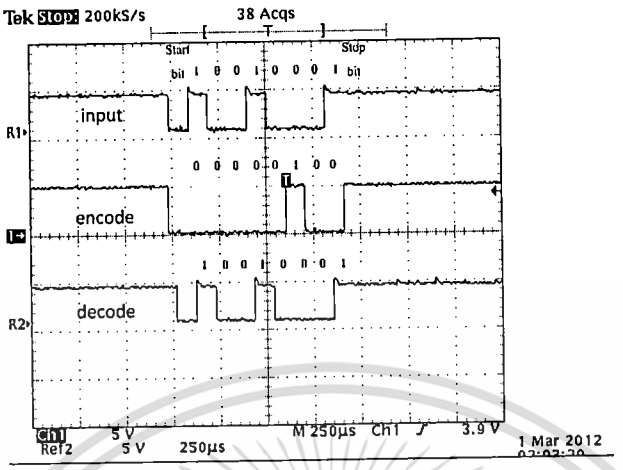
เมื่อสังเกตค่าของอินพุตที่ผ่านการเข้ารหัส (encode) จากการป้อนด้วยอินพุตเท่ากับ [0 0 0 0 1 1 1 1] จากรูปที่ 4.110, 4.111, 4.112 และ 4.113 จะเห็นว่าถึงแม้จะทำการเข้ารหัสด้วยอินพุตค่าเดิม 4 ครั้ง แต่ค่าของอินพุตที่ผ่านการเข้ารหัส (encode) จะมีค่าไม่ซ้ำเดิม ซึ่งแสดงถึงความเป็นไดนามิคของระบบ (Dynamic system)



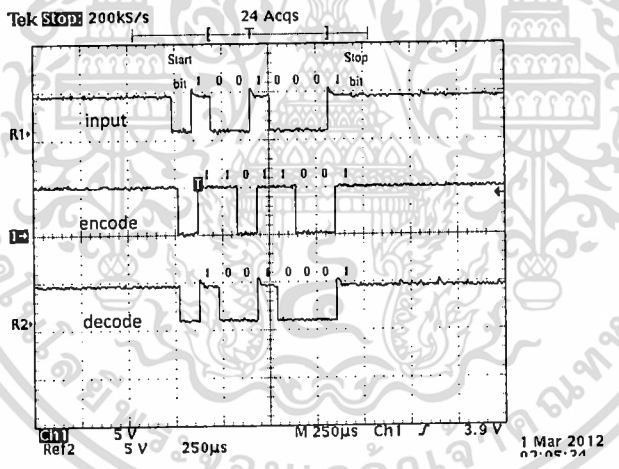
รูปที่ 4.114 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส
โดยมีอินพุตคือ [1 0 0 0 1 0 0 1] ครั้งที่ 1



รูปที่ 4.115 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส
โดยมีอินพุตคือ [1 0 0 0 1 0 0 1] ครั้งที่ 2



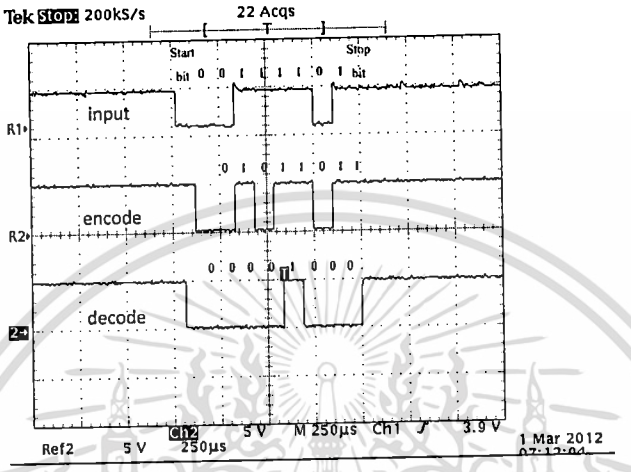
รูปที่ 4.116 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส
โดยมีอินพุตคือ [1 0 0 0 1 0 0 1] ครั้งที่ 3



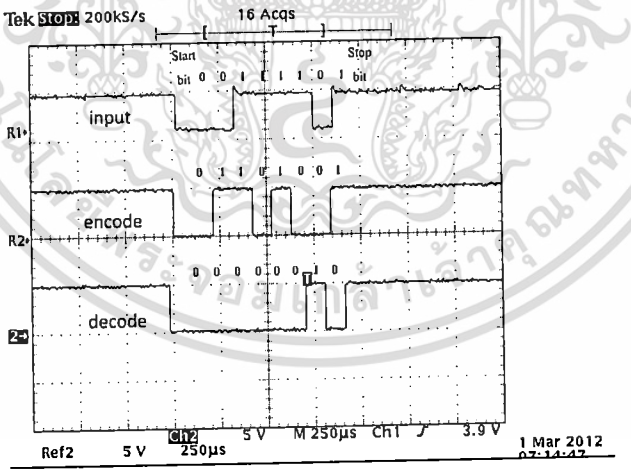
รูปที่ 4.117 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส
โดยมีอินพุตคือ [1 0 0 0 1 0 0 1] ครั้งที่ 4

และอีกหนึ่งตัวอย่างของการป้อนอินพุตด้วย [1 0 0 0 1 0 0 1] ทั้ง 4 ครั้ง จากรูปที่ 4.114, 4.115, 4.116 และ 4.117 จะแสดงให้เห็นว่าค่าของอินพุตที่ผ่านการเข้ารหัส (encode) จะไม่ซ้ำเดิมทั้ง 4 ครั้ง

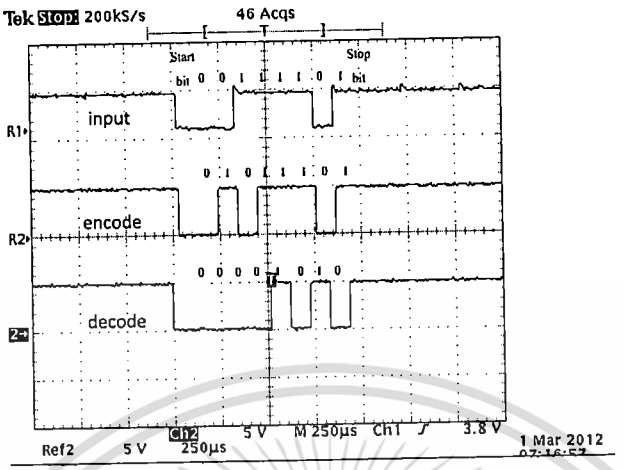
4.7.2 เมื่อกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัสมีค่าไม่เท่ากัน



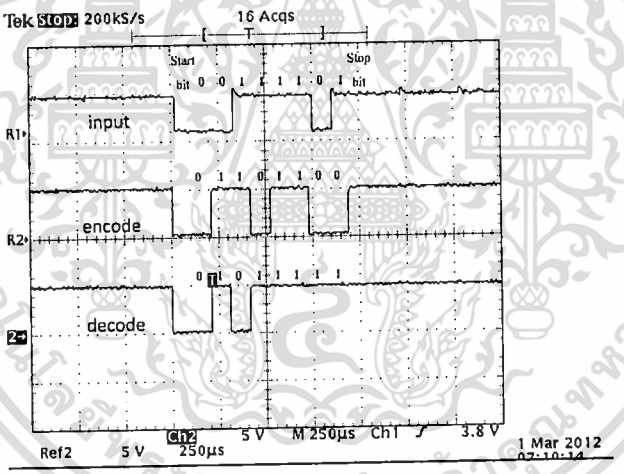
รูปที่ 4.118 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส โดยมีอินพุตคือ [1 0 1 1 1 1 0 0] ครั้งที่ 1



รูปที่ 4.119 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส โดยมีอินพุตคือ [1 0 1 1 1 1 0 0] ครั้งที่ 2

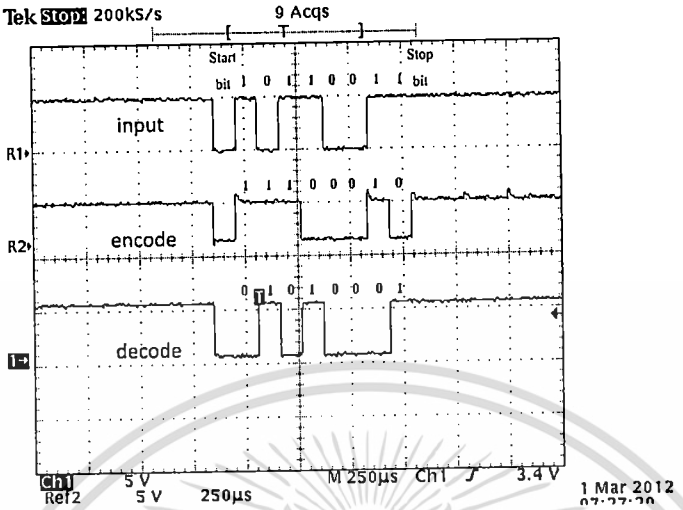


รูปที่ 4.120 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส
โดยมีอินพุตคือ [1 0 1 1 1 1 0 0] ครั้งที่ 3

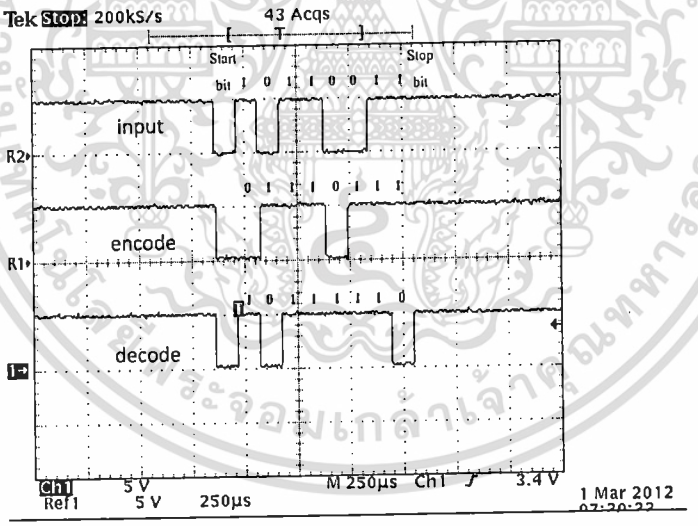


รูปที่ 4.121 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส
โดยมีอินพุตคือ [1 0 1 1 1 1 0 0] ครั้งที่ 4

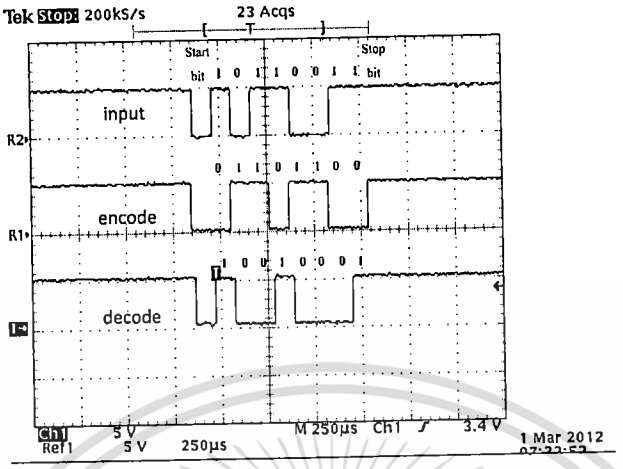
เมื่อสังเกตค่าของอินพุตที่ผ่านการเข้ารหัส (encode) จากการป้อนอินพุตด้วย [1 0 1 1 1 1 0 0] จากรูปที่ 4.118, 4.119, 4.120 และ 4.121 จะเห็นว่าถึงแม้จะทำการเข้ารหัสด้วยอินพุตค่าเดิม 4 ครั้ง แต่ค่าของอินพุตที่ผ่านการเข้ารหัส (encode) จะมีค่าไม่ซ้ำเดิม ซึ่งแสดงถึงความเป็นไดนามิคของระบบ (Dynamic system) แต่เนื่องจากค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัสมีค่าไม่เท่ากันจึงทำให้ไม่สามารถถอดรหัสได้



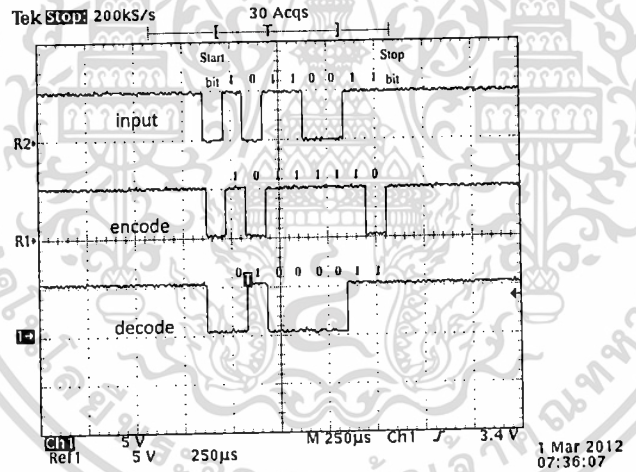
รูปที่ 4.122 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส โดยมีอินพุตคือ [1 1 0 0 1 1 0 1] ครั้งที่ 1



รูปที่ 4.123 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส โดยมีอินพุตคือ [1 1 0 0 1 1 0 1] ครั้งที่ 2



รูปที่ 4.124 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส
โดยมีอินพุตคือ [1 1 0 0 1 1 0 1] ครั้งที่ 3



รูปที่ 4.125 ค่าวัดสัญญาณทางไฟฟ้าเมื่อผ่านวงจรเข้ารหัสและถอดรหัส
โดยมีอินพุตคือ [1 1 0 0 1 1 0 1] ครั้งที่ 4

และอีกหนึ่งตัวอย่างของการป้อนอินพุตด้วย [1 1 0 0 1 1 0 1] ทั้ง 4 ครั้ง จากรูปที่ 4.122, 4.123, 4.124 และ 4.125 จะแสดงให้เห็นว่าค่าของอินพุตที่ผ่านการเข้ารหัส (encode) จะไม่ซ้ำเดิมทั้ง 4 ครั้ง แต่เนื่องจากค่าสัมประสิทธิ์ของวงจรเข้ารหัสและวงจรถอดรหัสมีค่าไม่เท่ากันจึงทำให้ไม่สามารถถอดรหัสได้

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ในโครงงานวิจัยนี้นี้ได้นำเสนอการใช้วงจรกรองสัญญาณดิจิตอลอันดับที่สอง ในการสร้างวงจรเข้ารหัสและถอดรหัสแบบเคออติก ซึ่งวงจรเข้ารหัสเป็นวงจรกรองสัญญาณดิจิตอลชนิดผลตอบสนองอิมพัลส์ไม่จำกัด (IIR filter) โดยถ้าเลือกค่าสัมประสิทธิ์ของวงจรกรองอย่างน้อยหนึ่งตัวให้อยู่ภายนอกขอบเขตพื้นที่สามเหลี่ยมเสถียรภาพ จะทำให้ตำแหน่งของโพลอยู่ภายนอกวงกลมหนึ่งหน่วย ซึ่งทำให้วงจรเข้ารหัสเกิดความไม่มีเสถียรภาพและแสดงพฤติกรรมความเป็นเคออติก ส่วนในการจำลองการทำงานบนโปรแกรม MATLAB จะต้องอาศัยฟังก์ชัน $f(\bullet)$ เพื่อจำลองให้เกิดการล้นแบบส่วนเติมเต็มสองที่ไม่เป็นเชิงเส้นเป็นผลให้เกิดพฤติกรรมเคออสขึ้น แต่การออกแบบในฮาร์ดแวร์จริงไม่จำเป็นต้องใช้ฟังก์ชั่น $f(\bullet)$ เพื่อให้เกิดการล้น แต่พฤติกรรมการล้นแบบส่วนเติมเต็มสองจะสามารถเกิดขึ้นได้เองบนฮาร์ดแวร์จากการบวกเลขแบบส่วนเติมเต็มสอง สำหรับวงจรถอดรหัสนั้นใช้วงจรกรองสัญญาณแบบผลตอบสนองอิมพัลส์จำกัด (FIR filter) ซึ่งเป็นส่วนกลับ (inverse) ของวงจรเข้ารหัส โดยเมื่อเลือกค่าสัมประสิทธิ์ให้มีมีค่าตรงกับวงจรเข้ารหัสจะทำให้ตำแหน่งซีโร่ของวงจรถอดรหัสตรงกับตำแหน่งโพลของวงจรเข้ารหัสทำให้เกิดการหักล้าง (cancel) กัน ดังนั้นจึงต้องเลือกค่าสัมประสิทธิ์ของวงจรเข้าและถอดรหัสให้ตรงกันจึงจะสามารถเข้ารหัสและถอดรหัสได้

วงจรเข้ารหัสและถอดรหัสที่ได้กล่าวมาข้างต้น จะสามารถเพิ่มความซับซ้อนให้มากขึ้นได้ โดยการเพิ่ม order ให้กับวงจรกรองสัญญาณดิจิตอลที่ใช้เป็นวงจรเข้ารหัสและถอดรหัสคล้ายกับวิธีการเพิ่มรอบการเข้ารหัสของ Double DES หรือ Triple DES วิธีที่ใช้ในการเพิ่ม order ให้กับวงจรเข้ารหัสและถอดรหัส จะใช้วิธีการนำวงจรกรองสัญญาณดิจิตอลอันดับที่สองชนิด IIR มาต่อ Cascade กันสำหรับฝั่งเข้ารหัส ส่วนฝั่งถอดรหัสก็จะใช้วิธีเดียวกันคือนำวงจรกรองสัญญาณดิจิตอลอันดับที่สองชนิด FIR มาต่อ Cascade กัน จะทำให้ได้วงจรเข้ารหัสถอดรหัสที่มี

ความซับซ้อนเพิ่มมากขึ้นตามจำนวน order ที่เพิ่มเข้าไป จำนวน order ก็จะสามารถเพิ่มขึ้นได้เป็นเลขคู่ตั้งแต่ 2, 4, 6, 8, ... , n order ตามความต้องการ ซึ่งเมื่อ order เพิ่มขึ้นจำนวนกุญแจ (ค่าสัมประสิทธ์) ที่ใช้ในการเข้ารหัสก็จะเพิ่มมากขึ้นด้วยตามจำนวน order ความแข็งแกร่งในการเข้ารหัสลับข้อมูลก็จะมีมากขึ้น

ในการจำลองการทำงานของวงจรเข้ารหัสโดยกำหนดให้มีอินพุตเป็นศูนย์แล้วได้เอาต์พุตที่ไม่เป็นศูนย์ (zero in non-zero out) นั้นเป็นคุณสมบัติของระบบที่ไม่เป็นเชิงเส้น ซึ่งพบว่าพฤติกรรมของวงจรเข้ารหัสจะสร้างสัญญาณเอาต์พุตที่มีลักษณะคล้ายสัญญาณรบกวนและเมื่อนำวงจรเข้ารหัสและถอดรหัสแบบ 2^{nd} order และ 4^{th} order ไปจำลองการทำงานบนโปรแกรม MATLAB โดยใช้ข้อมูลชนิดต่างๆ เช่น สัญญาณคลื่นรูปไซน์, ข้อมูลเสียง และข้อมูลภาพ พบว่าสัญญาณที่ถูกเข้ารหัสแล้วจะมีลักษณะคล้ายสัญญาณรบกวน ซึ่งแสดงพฤติกรรมของความเป็นเคออส ไม่สามารถคาดเดาได้

ในส่วนของการออกแบบวงจรในอุปกรณ์ FPGA ทำการบรรยายพฤติกรรมการทำงานด้วยภาษา VHDL จะออกแบบวงจรเข้ารหัสและถอดรหัสโดยอาศัยหลักการออกแบบวงจรกรองสัญญาณดิจิตอลชนิด IIR และ FIR ทั้งแบบ 2^{nd} order และ 4^{th} order โดยได้ทำการออกแบบวงจรให้มีการเข้ารหัสและถอดรหัสข้อมูลเป็น 8 , 16 , 24 และ 32 บิต ซึ่งเมื่อจำนวนบิตของค่าสัมประสิทธ์ที่ใช้เป็นกุญแจเพิ่มมากขึ้นก็จะทำให้จำนวนกุญแจที่เป็นไปได้ทั้งหมดในการถอดรหัสนั้นเพิ่มมากขึ้น ความปลอดภัยก็จะเพิ่มมากขึ้นด้วย

จำนวนกุญแจที่เป็นไปได้ทั้งหมด = 2^n (n คือจำนวนบิตของค่าสัมประสิทธ์)

ในวงจรเข้ารหัสหนึ่งตัวจะมีกุญแจในการเข้ารหัส (ค่าสัมประสิทธ์) สองตัว เมื่อเพิ่มบิตในการเข้ารหัสให้มากขึ้นและเพิ่ม order ให้กับวงจรเข้ารหัสถอดรหัส จำนวนกุญแจเพิ่มมากขึ้น จำนวนครั้งในการคาดเดากุญแจก็จะเพิ่มมากขึ้นด้วย สามารถแสดงตารางเปรียบเทียบได้ดังนี้

ตารางที่ 5.1 เปรียบเทียบจำนวนกุญแจที่เป็นไปได้

ขนาดกุญแจ (บิต)	จำนวนกุญแจที่เป็นไปได้ (ต่อค่า C หนึ่งตัว)	จำนวนกุญแจที่เป็นไปได้ (ต่อค่า C สองตัว)	จำนวนครั้งในการคาดเดากุญแจ (2 Order)
8	$2^{8} = 256$	512	65536
16	$2^{16} = 65536$	131072	4.29×10^{9}
24	$2^{24} = 16777216$	33.55×10^{6}	4.29×10^{9}
32	$2^{32} = 4.29 \times 10^{9}$	8.58×10^{9}	2.81×10^{14}
64	$2^{64} = 1.84 \times 10^{19}$	3.68×10^{19}	3.4×10^{38}
128	$2^{128} = 3.4 \times 10^{38}$	6.8×10^{38}	1.15×10^{77}
256	$2^{256} = 1.15 \times 10^{77}$	2.31×10^{77}	1.32×10^{154}

จากการออกแบบวงจรเข้ารหัสและถอดรหัส ด้วยภาษา VHDL จะจำลองการทำงานผ่านโปรแกรม MAX+ plus II และโปรแกรมวงจรลงบนอุปกรณ์ FPGA ซึ่งผลข้อมูลที่ถูกเข้ารหัสบนฮาร์ดแวร์จริงจะเป็นแบบกึ่งเคออส (Quasin-Chaotic) ที่มีความเป็นไดนามิคเปลี่ยนแปลงตลอดเวลา แต่จากผลการทดลองในหัวข้อย่อย 4.6 จะเห็นว่าข้อมูลที่ถูกเข้ารหัสนั้นมีความเป็นไดนามิคที่มีความซ้ำคาบอยู่ ซึ่งถ้าเพิ่มบิตข้อมูลในการเข้ารหัสที่เพิ่มมากขึ้นก็จะทำให้เพิ่มความเข้าใกล้ความเป็นเคออสที่แท้จริงเพิ่มมากขึ้นด้วย

ในส่วนของนำไปประยุกต์ใช้งานด้านสื่อสาร ได้ทำการออกแบบวงจรสื่อสารข้อมูลอนุกรมเพิ่มเติมในอุปกรณ์ FPGA เพื่อนำไปใช้สื่อสารรับส่งข้อมูลตัวอักษรผ่านทางพอร์ตอนุกรมในเครื่องคอมพิวเตอร์ จากการทดสอบโดยใช้โปรแกรมสื่อสารอนุกรมพื้นฐานบนเครื่องคอมพิวเตอร์ทั่วไป (Serial Communication or Hyper-terminal) พบว่าสามารถรับส่งข้อมูลในรูปของตัวอักษรได้ เมื่อกำหนดให้วงจรเข้ารหัสและวงจรถอดรหัสมีค่าสัมประสิทธิ์ที่เหมือนกัน จะได้ข้อมูลตัวอักษรที่ได้รับมาเป็นข้อมูลที่ถูกต้องตรงกับฝั่งส่ง แต่ถ้ากำหนดให้วงจรเข้ารหัสและวงจรถอดรหัสมีค่าสัมประสิทธิ์ที่ไม่เหมือนกัน เมื่อทดสอบการส่งข้อมูลตัวอักษรจะได้ข้อมูลตัวอักษรที่ได้รับมาจากฝั่งส่งเป็นข้อมูลตัวอักษรที่ผิดเพี้ยนไปจากเดิม และถ้าทดสอบโดยการใช้

เครื่องคอมพิวเตอร์เครื่องที่สามทำหน้าที่ดักจับแอบดูข้อมูลที่ถูกเข้ารหัสในระหว่างทางที่ข้อมูลกำลังถูกส่งไปยังฝั่งรับข้อมูล ก็จะพบว่าเครื่องคอมพิวเตอร์ที่ทำการดักจับแอบดูข้อมูลจะไม่สามารถมองเห็นข้อมูลที่ถูกต้องแท้จริงได้

5.2 ข้อเสนอแนะ

สำหรับโครงงานวิจัยนี้เป็นการศึกษาและจำลองพฤติกรรมของการเกิดเคออสที่เกิดขึ้นภายในวงจรกรองสัญญาณดิจิตอล แล้วนำไปออกแบบสร้างวงจรเข้ารหัสและถอดรหัส ส่วนของวงจรเข้ารหัสและถอดรหัสสร้างโดยการออกแบบวงจรกรองสัญญาณดิจิตอลชนิด IIR และ FIR และได้นำไปประยุกต์ใช้งานกับการส่งข้อมูลทางการสื่อสารแบบอนุกรม และในการพัฒนาชิ้นงานให้นำไปใช้ประโยชน์ได้หลากหลาย อาจจะมีการไปประยุกต์ใช้งานกับการสื่อสารในรูปแบบอื่นๆ

ในส่วนของการกำหนดค่าสัมประสิทธิ์ที่เป็นกุญแจในการเข้ารหัส-ถอดรหัส เมื่อเพิ่ม order ให้กับวงจรเข้ารหัส จะทำให้จำนวนกุญแจมากขึ้นซึ่งเป็นการเพิ่มความซับซ้อนให้กับการเข้ารหัสข้อมูลที่ดี แต่เมื่อนำไปใช้จริงนั้นจะพบปัญหาจำนวนกุญแจนั้นมากเกินไป ไม่สะดวกต่อผู้ใช้งาน ดังนั้นแนวทางในการพัฒนาต่อไปอาจจะทำการหาวิธีลดจำนวนการใช้กุญแจลงโดยไม่ลดความซับซ้อนระบบ และหาวิธีการป้องกันการคาดเดากุญแจเพื่อป้องกันการโจมตีระบบแบบ Brute-Force ได้

บรรณานุกรม

[1] L. O. Chua and T. Lin, "Chaos in digital filters," IEEE Trans. Circuits and Systems, pp. 648-658, vol. 35, no. 6, June 1988.

[2] T. Lin and L. O. Chua, "On chaos of digital filters in the real world," IEEE Trans. Circuits and Systems, pp. 557-558, vol. 38, no. 5, May 1991.

[3] Y. Umezawa, M. Dobashi, H. Kamata and T. Endo, "Chaos signal generator by IIR digital filters including nonlinear functions and its application," Proc. 1998 Second International Conference on Knowledge-Based Intelligent Electronic Systems, pp. 169-175, 21-23 April 1998, Adelaide, Australia.

[4] H. Kamata, Y. Umezawa, M. Dobashi and T. Endo, "Private communications with chaos based on the fixed-point computation," IEICE Trans. Fundamentals, pp. 1238-1246, vol. E83-A, no. 6, June 2000.

[5] X. Yu and Z. Galias, "Periodic behaviors in a digital filter with two's complement arithmetic," IEEE Trans. Circuits and Systems-I: Fundamental theory and applications, pp. 1177-1190, vol. 48, no. 10, October 2001.

[6] B. W. K. Ling, A. Y. P. Chan, T. P. L. Wong and P. K. S. Tam, "Autonomous response of a third-order digital filter with two's complement arithmetic realized in parallel form," Communication in Information and Systems, pp. 435-454, vol. 2 no. 4, December 2002.

[7] D. R. Frey, "Chaotic digital encoding: An approach to secure communication," IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Process., pp. 660-666, vol. 40, no. 10, October 1993.

[8] K. Kutzer, W. Schwarz and A. C. Davies, "Chaotic signals generated by digital filter overflow," Proc. IEEE International Symposium on Circuits and Systems 1994 (ISCAS' 94), pp. 17-20, vol. 6, 30 May-2 June 1994.

[9] K. Kelber and T. Kilias, "Analysis of an encoder-decoder-system based on digital filter structures with two's complement overflow characteristic," Proc. IEEE International Symposium on Circuits and Systems 1996 (ISCAS' 96), pp. 166-169, vol. 3, 12-15 May 1996.

[10]A. Leuciuc, "Information transmission using chaotic discrete-time filter," IEEE Trans. Circuits and Systems, Part 1, pp. 82-88, vol. 47, no. 1, January 2000.

[11]สมเกียรติ ตั้งกิจวานิชย์. "ทฤษฎีความโกลาหล." http://th.wikipedia.org/wiki/ทฤษฎีความอลวน.

[12]สิทธิพงษ์ คิวหา, สุนทรินทร์ ศิลป์ท้าว, สุนัย เนสะและ. "*อุปกรณ์เข้ารหัส – ถอดรหัสแบบ CHAOTIC บนพื้นฐานความไม่เป็นเชิงเส้นจากการล้นในวงจรกรองสัญญาณดิจิตอลสำหรับความปลอดภัยในการสื่อสาร*." ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สาขาวิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2553.

[13]วรเทพ ไพบูลย์รัตนากร ,*เปิดโลก FPGA กับ Wizard PLD-A01*. กรุงเทพฯ : บริษัทแอสทรอนลอจิก รีเสิร์ชแอนด์ดิวิลอปเมนต์ จำกัด.

[14]วรเทพ ไพบูลย์รัตนากร, บุญอนันต์ เกียงเอีย. *สัมผัสโลก USB ด้วย Ezy USB Module*. กรุงเทพฯ : บริษัทแอสทรอน ลอจิก รีเสิร์ชแอนด์ดิวิลอปเมนต์ จำกัด., 2545.