สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ชุดทดสอบรหัสแก้ไขข้อผิดพลาด
สำหรับรหัสคอนโวลูชันและวิธีไวเทอร์บี
Error Control Coding Tester
For
Convolution / Viterbi CODEC

โดย
นายวุฒิกรณ์ ตรัยศิลานันท์
นายสันติชล ชัชวาลุรัตน์

เลขหมู่.....
เลขทะเบียน 62133
วัน,เดือน,ปี 31 ก.ค. 2549
ล.....
ส.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมสารสนเทศ
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2547

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Error Control Coding Tester

For

Convolution / Viterbi CODEC

BY

Mr. Wuthigorn Traisilanun

Mr. Suntichon Chatchawalurat

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPARTMENT OF INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2004

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	ชุดทดสอบรหัสแก้ไขข้อผิดพลาดสำหรับรหัสคอนโวลูชันและวิธีไวเทอร์บี
ชื่อนักศึกษา	นายวุฒิกรณ์ ตรัยศิลานันท์ รหัสประจำตัว 44010471
	นายสันติชล ชัชวาลุรัตน์ รหัส ประจำตัว 44010518
อาจารย์ผู้ควบคุมปริญญานิพนธ์	รศ. อรลาภ แสงอรุณ
อาจารย์ผู้ควบคุมปริญญานิพนธ์ร่วม	ผศ.ดร.อรรถสิทธิ์ หล่าสกุล
อาจารย์ผู้ควบคุมปริญญานิพนธ์ร่วม	ดร.เกียรติศักดิ์ ศรีพิมานวัฒน์ (ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ)
ระดับการศึกษา	ปริญญาตรี วิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมสารสนเทศ
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2547

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมสารสนเทศบัณฑิต

.....

(รศ. อรลาภ แสงอรุณ)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	ชุดทดสอบรหัสแก้ไขข้อผิดพลาดสำหรับรหัสคอนโวลูชันและไวเทอร์บี	
ชื่อนักศึกษา	นายวุฒิกรณ์ ตรัยศิลานันท์	รหัสประจำตัว 44010471
	นายสันติชล ชัชวาลุรัตน์	รหัส ประจำตัว 44010518
อาจารย์ผู้ควบคุมปริญญานิพนธ์	รศ. อรลาภ แสงอรุณ	
อาจารย์ผู้ควบคุมปริญญานิพนธ์ร่วม	ผศ.ดร.อรรถสิทธิ์ หล่าสกุล	
อาจารย์ผู้ควบคุมปริญญานิพนธ์ร่วม	ดร.เกียรติศักดิ์ ศรีพิมานวัฒน์ (ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ)	
ระดับการศึกษา	ปริญญาตรี วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมสารสนเทศ	
ภาควิชา	วิศวกรรมสารสนเทศ	
ปีการศึกษา	2547	

บทคัดย่อ

ปริญญานิพนธ์นี้กล่าวถึงการสร้างชุดทดสอบประสิทธิภาพรหัสแก้ไขความผิดพลาดข้อมูล โดยการวัดเปรียบเทียบค่าอัตราการผิดพลาดของบิต (BER) ต่อค่าอัตราส่วนการใช้กำลังงานในการส่งต่อสัญญาณรบกวน (Eb/No) ในช่องสัญญาณระบบสื่อสารแบบ Additive white Gaussian Noise (AWGN) การออกแบบสร้างชุดทดสอบโดยแบ่งออกเป็น ภาคกำเนิดข้อมูล ภาคมอดูเลเตอร์ ภาคกำเนิดสัญญาณรบกวนแบบเกาส์ ภาคดีมอดูเลเตอร์ ภาคหน่วงเวลา และภาคเปรียบเทียบข้อมูล การสร้างสัญญาณรบกวนแบบเกาส์ใช้วิธีบ็อกมูลเลอร์ (Box-Muller) ร่วมกับวิธีการประมาณค่าฟังก์ชันค่ารากที่สองและล็อกการิทึมแบบลิเนียร์แบบแบ่งออกสี่ส่วน ซึ่งวิธีการนี้ทำให้ชุดกำเนิดสัญญาณรบกวนมีโครงสร้างขนาดเล็กและสามารถทำงานได้ที่ความเร็วสูง ดังผลการออกแบบใช้อุปกรณ์ในเอฟพีจีเอ 171 slices 4 บล็อกแรมและ 4 วงจรคูณ และให้ค่าเอาต์พุตสูงสุดที่อัตรา 243 MSamples/sec ผลการทดสอบกับชุดเข้ารหัสคอนโวลูชัน (Convolution) และชุดถอดรหัส Viterbi ที่มีค่า K เท่ากับ 3 และ Coding Rate เท่ากับ 1/2 กราฟค่าความผิดพลาดบิตมีลักษณะตามทฤษฎีซึ่งแสดงให้เห็นว่าชุดทดสอบรหัสแก้ไขข้อผิดพลาดสามารถทดสอบรหัสแก้ไขข้อผิดพลาดนี้ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title	Error Control Coding Tester for Convolution / Viterbi CODEC
Student	Mr. Wuthigorn Traisilanun ID.44010471
	Mr. Suntichon Chatchawalurat ID.44010528
Thesis Advisor	Assoc.Prof.Ornlarp Sangaroon
Thesis Co-Advisor	Asst.Prof.Dr.Attasit Lasakul
Thesis Co-Advisor	Dr. Keattisak Sripimanwat
	(National Electronics and Computer Technology Center)
Graduate Level	Bachelor Degree of Information Engineering
Department	Information Engineering
Academic Year	2004

Abstract

This thesis presents a design and implementation of Error Control Coding Tester for testing the performance of communication prototypes in an additive white Gaussian noise (AWGN) channel. The tester is designed consisting of Data Generator, Modulator/Demodulator, Delay Module, Data Comparator and AWGN Generator. The white Gaussian noise generator based on Box-Muller method with the four segments linear quantization in logarithm and square root function approximation, are proposed. This method has been successfully implemented on Xilinx Vertex II FPGA platform. As a result, it obtains high-speed output sampling rate utilizing small resource of FPGA. Their maximum noise output sampling rates are 243 Msamples/s. However, it requires 171 slices, four block RAMs, and four multipliers on Xilinx Vertex-II FPGA. Evaluation performance has been tested with Convolution Code and Viterbi Encode (k=3 and Coding Rate=1/2). The results are matched with those of theoretical output. This tester is, therefore, a good platform for testing the given error control code. It is also flexible to be developed for testing other coding schemes as well.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงได้อย่างดี ด้วยการช่วยเหลือและสนับสนุนจากหลายๆ ท่านด้วยกันซึ่งผู้เขียนขอขอบพระคุณดังต่อไปนี้

ขอขอบคุณ รศ.อรลาภ แสงอรุณ อาจารย์ผู้ควบคุมปริญญานิพนธ์ ผศ.ดร.อรรถสิทธิ์ หล่าสกุล และดร.เกียรติศักดิ์ ศรีพิมานวัฒน์ อาจารย์ผู้ควบคุมปริญญานิพนธ์ร่วม ในการให้ความรู้ คำแนะนำและคำปรึกษาที่เป็นประโยชน์อย่างสูงในการทำปริญญานิพนธ์นี้ ผู้จัดทำรู้สึกซาบซึ้งในความอนุเคราะห์ที่ดีเยี่ยมและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณภาควิชาวิศวกรรมสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ให้สถานที่ในการทำปริญญานิพนธ์ฉบับนี้

ขอขอบคุณพี่สมาชิกห้องวิจัย Digital System Laboratory ที่ให้ความร่วมมือและให้ความช่วยเหลือในการทำปริญญานิพนธ์ฉบับนี้เป็นอย่างดี

ขอขอบคุณ โครงการวิจัยชุดต้นแบบหน่วยปรับปรุงประสิทธิภาพการสื่อสัญญาณสำหรับโครงการ 3G และการประยุกต์สำหรับอุตสาหกรรม (RDT2) ของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ที่ให้ความร่วมมือและสนับสนุนอุปกรณ์ในการทำโครงงาน

วุฒิกรณ์ ตรัยศิลานันท์
สันติชล ชัชวาลุรัตน์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เรื่อง	หน้า
บทคัดย่อภาษาไทย	ก
บทคัดย่อภาษาอังกฤษ	ข
กิตติกรรมประกาศ	ค
สารบัญ	ง
สารบัญรูป	ซ
สารบัญตาราง	ญ
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญของปัญหา	1
1.2 ความมุ่งหมายและวัตถุประสงค์ของโครงงาน	2
1.3 ขอบเขตของโครงงาน	2
1.4 โครงสร้างของชุดทดสอบรหัสแก้ไขข้อผิดพลาด	3
1.5 ขั้นตอนการดำเนินงาน	3
บทที่ 2 หลักการและทฤษฎีในการสร้างชุดทดสอบรหัสแก้ไขข้อผิดพลาด	5
2.1 การจำลองสัญญาณในระบบสื่อสาร	5
2.2 การจำลองสัญญาณรบกวนแบบ Additive White Gaussian Noise (AWGN)	7
2.2.1 ความสัมพันธ์ระหว่างค่า SNR และค่าความแปรปรวนของสัญญาณ	8
2.2.2 วิธีสร้างตัวแปรสุ่มแบบเกาส์	10
2.2.2.1. ทฤษฎีเซ็นทรัลลิมิต	10
2.2.2.2. วิธีบ็อกมูลเลอร์	10
2.2.2.3. วิธีโพลาร์	11
2.2.2.4. วิธีผสม	12
2.3 การทดสอบการกระจายตัว	12
2.3.1 Relative Error	12
2.3.2 Chi-square	13
2.4 การสร้างตัวเลขสุ่มโดยใช้ Linear Feedback Shift Register (LSFR)	13
2.4.1 Single-Bit Random Number Generator Using LFSR	13

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เรื่อง	หน้า
2.4.1.1 Single-LFSR Method	16
2.4.1.2 Single-LFSR with a Counter Method	16
2.4.1.3 Parallel-LFSR Method	17
2.4.2 Multiple-Bit Leap-Forward LFSR	17
2.5 รหัสแก้ไขข้อผิดพลาด	19
2.5.1 การเข้าและถอดรหัสรหัส Convolution	20
2.6 การสื่อสารแบบอนุกรม	21
บทที่ 3 เอฟพีจีเอพื้นฐาน	22
3.1 ความรู้พื้นฐานวงจรรวมแบบดิจิตอล	23
3.1.1 วงจรรวมแบบลอจิกมาตรฐาน	23
3.1.2 วงจรรวมแบบพีแอลดี	23
3.1.2.1 อุปกรณ์เอสพีแอลดี	24
3.1.2.2 อุปกรณ์ซีพีแอลดี	27
3.1.2.3 อุปกรณ์เอฟพีจีเอ	27
3.1.2.4 เทคโนโลยีการโปรแกรมพีแอลดี	27
3.1.3 วงจรรวมแบบเอสิก	29
3.1.3.1วงจรรวมเกตอาเรย์	29
3.1.3.2 วงจรรวมแบบเซลล์มาตรฐาน	29
3.1.3.3 วงจรรวมแบบฟูดัสตัม	30
3.2 โครงสร้างของเอฟพีจีเอ Xilinx Vertex-II	30
3.2.1 โครงสร้างภายในเอฟพีจีเอ Xilinx Vertex-II	30
3.2.2.1 บล็อกอินพุตเอาต์พุต	31
3.2.2.2 บล็อก Configurable Logic Block	31
3.2.2.3 บล็อกหน่วยความจำ 18 กิโลบิต	33
3.2.2.4 บล็อกวงจรคูณขนาด 18 x 18 บิต	33
3.2.2.5 บล็อก Digital Clock Manager	34

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เรื่อง	หน้า
3.2.2.6 บล็อกเชื่อมต่อและหาเส้นทาง	34
3.3 การออกแบบและติดตั้งวงจรดิจิตอลบนเอฟพีจีเอ	34
3.4 ภาษาเวริล็อก	37
3.4.1ประวัติความเป็นมา	37
3.4.2 โครงสร้างของภาษาเวริล็อก	38
บทที่ 4 การออกแบบโครงงาน	38
4.1 โครงสร้างชุดทดสอบรหัสแก้ไขข้อผิดพลาด	39
4.1.1 ภาคกำเนิดข้อมูล	39
4.1.2 ภาคมอดูเลเตอร์	40
4.1.3 ภาคกำเนิดสัญญาณรบกวนแบบเกาส์	41
4.1.3.1 มอดูลกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม	41
4.1.3.2 มอดูลประมาณค่าฟังก์ชัน	42
4.1.3.3 มอดูลชุดวงจรคูณค่าฟังก์ชัน	44
4.1.3.4มอดูลปรับเปลี่ยน Eb/No	44
4.1.4ภาคช่องสัญญาณ	45
4.1.5 ภาคดีมอดูเลเตอร์	45
4.1.6ภาคเปรียบเทียบข้อมูล	45
4.1.7 ภาคคอมพิวเตอร์อินเตอร์เฟส	45
4.2 โครงสร้างของชุดเข้ารหัสและถอดรหัสคอนโวลูชัน (Convolution)	45
4.2.1 ภาคเข้ารหัสคอนโวลูชัน	46
4.2.2 ภาคถอดรหัสคอนโวลูชัน	46
4.2.2.1 Branch Metric Generation	47
4.2.2.2 Survivor Path and Path Metric Update	47
4.2.2.3 Optimum Paths Trace Back	48

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เรื่อง	หน้า
บทที่ 5 ผลการทดลอง	49
5.1 ผลการสร้างสัญญาณรบกวนทางซอฟต์แวร์	49
5.1.1 วิธีของเซ็นทรัลลิมิต	49
5.1.2 วิธีบ็อกมูลเลอร์	51
5.1.3 วิธีโพลาร์	52
5.1.4 วิธีผสม	53
5.2 ผลการจำลองการทำงานแต่ละมอดูลของชุดทดสอบรหัสแก้ไขข้อผิดพลาด	54
5.2.1 มอดูลกำเนิดข้อมูล	54
5.2.2 มอดูลมอดูเลเตอร์	56
5.2.3 มอดูลชุดกำเนิดสัญญาณรบกวน	58
5.2.3.1 มอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม	58
5.2.3.2 มอดูลประมาณค่าฟังก์ชัน	59
5.2.3.3 มอดูลชุดวงจรคูณค่าฟังก์ชัน	67
5.2.3.4 มอดูลปรับเปลี่ยน Eb/No	69
5.2.4 ผลการออกแบบและจำลองการทำงานของชุดกำเนิดสัญญาณรบกวน	76
5.2.5 มอดูลดีมอดูเลเตอร์	82
5.2.6 มอดูลหน่วงเวลา	84
5.2.7 มอดูลเปรียบเทียบข้อมูล	85
5.3 ผลการจำลองการทำงานของชุดเข้ารหัสและถอดรหัสคอนโวลูชัน	89
5.3.1 มอดูลเข้ารหัสคอนโวลูชัน	89
5.3.2 มอดูลถอดรหัสคอนโวลูชัน	91
5.4 ผลการออกแบบและจำลองการทำงานรวมของชุดทดสอบรหัสแก้ไขข้อผิดพลาด	93
5.5 ผลการทดสอบชุดทดสอบรหัสคอนโวลูชัน	97

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เรื่อง		หน้า
บทที่ 6 สรุป		98
6.1 สรุปโครงงาน		98
6.2 ข้อเสนอแนะและสิ่งที่สามารถพัฒนาต่อได้		99
เอกสารอ้างอิง		100
ภาคผนวก ก	ชุดทดสอบรหัสแก้ไขข้อผิดพลาด	102
ภาคผนวก ข	ชุดเข้ารหัสและถอดรหัสคอนโวลูชัน	188
ภาคผนวก ค	โปรแกรมทดสอบและแสดงผล	193
ภาคผนวก ง	การใช้งานโปรแกรมควบคุมและแสดงผลทางคอมพิวเตอร์	201

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

รูป	หน้า
รูปที่ 1.1 โครงสร้างของชุดทดสอบแก้ไขรหัสข้อผิดพลาด	4
รูปที่ 2.1 บล็อกไดอะแกรมองค์ประกอบพื้นฐานของระบบสื่อสารดิจิตอล	6
รูปที่ 2.2 4-บิต LFSR	15
รูปที่ 2.3 4-บิต de Bruijn Counter	16
รูปที่ 2.4 4-บิต Leap-forward LFSR	19
รูปที่ 2.5 วงจรเข้ารหัส Convolution R =1/2 และ $K = 3$	20
รูปที่ 2.6 ไดอะแกรมสถานะ	21
รูปที่ 2.7 ไดอะแกรมทรีรีส	21
รูปที่ 3.1 โครงสร้างวงจรรวมแบบดิจิตอล	24
รูปที่ 3.2 โครงสร้างของพรอม	25
รูปที่ 3.3 โครงสร้างของพีแอลเอ	26
รูปที่ 3.4 โครงสร้างของพีเอแอล	26
รูปที่ 3.5 โครงสร้างของซีพีแอลดี	28
รูปที่ 3.6 โครงสร้างทั่วไปของเอฟพีจีเอ	28
รูปที่ 3.7 โครงสร้างของเอฟพีจีเอ Xilinx Virtex-II	30
รูปที่ 3.8 โครงสร้างของบล็อกอินพุตเอาต์พุต	32
รูปที่ 3.9 โครงสร้างของ Slice	32
รูปที่ 3.10 การเชื่อมต่อบล็อกหน่วยความจำขนาด 18 กิโลบิต และบล็อกวงจรคูณขนาด 18×18 บิต	34
รูปที่ 3.11 โครงสร้างของบล็อกเชื่อมต่อและการหาเส้นทาง	35
รูปที่ 3.12 ขั้นตอนการออกแบบและติดตั้งวงจรดิจิตอลบนเอฟพีจีเอ	35
รูปที่ 4.1 โครงสร้างของชุดทดสอบรหัสแก้ไขข้อผิดพลาด	40
รูปที่ 4.2 โครงสร้างทางฮาร์ดแวร์ของชุดกำเนิดสัญญาณรบกวนแบบเกาส์	42
รูปที่ 4.3 ค่าฟังก์ชัน *f(u1)* ที่ประมาณค่าโดยวิธีลิเนียร์แบบแบ่งออกสี่ส่วน	43
รูปที่ 4.4 วงจรเข้ารหัสคอนโวลูชัน R =1/2 และ $K = 3$	46

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูป	หน้า
รูปที่ 4.5 ไดอะแกรมของชุดถอดรหัส Viterbi	47
รูปที่ 5.1 ฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์โดยวิธีเซ็นทรัลลิมิต	50
รูปที่ 5.2 ฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์โดยวิธีบ็อกมูลเลอร์	51
รูปที่ 5.3 ฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์โดยวิธีโพลาร์	52
รูปที่ 5.4 ฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์โดยวิธีผสม	53
รูปที่ 5.5 ผังวงจรระดับ RTL ของมอดูลกำเนิดข้อมูล	55
รูปที่ 5.6 ผลจำลองการทำงานของมอดูลกำเนิดข้อมูล	56
รูปที่ 5.7 ผังวงจรระดับ RTL ของมอดูลมอดูเลเตอร์	57
รูปที่ 5.8 ผลจำลองการทำงานของมอดูลมอดูเลเตอร์	57
รูปที่ 5.9 ผังวงจรระดับ RTL ของมอดูล ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม	61
รูปที่ 5.10 ผลการจำลองการทำงานของมอดูล ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม	62
รูปที่ 5.11 ผังวงจรระดับ RTL ของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม	63
รูปที่ 5.12 ผลจำลองการทำงานของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม	65
รูปที่ 5.13 ผังวงจรระดับ RTL ของมอดูลประมาณค่าฟังก์ชันตรีโกณมิติ	66
รูปที่ 5.16 ผลจำลองการทำงานของมอดูลประมาณค่าฟังก์ชันตรีโกณมิติ	67
รูปที่ 5.15 ผังวงจรระดับ RTL ของมอดูลชุดวงจรคูณค่าฟังก์ชัน	69
รูปที่5.1 6 ผลจำลองการทำงานของของมอดูลชุดวงจรคูณค่าฟังก์ชัน	70
รูปที่ 5.17 ผังวงจรระดับ RTL ของมอดูลประมาณค่า SNR	71
รูปที่ 5.18 ผลจำลองการทำงานของของมอดูลประมาณค่า SNR	72
รูปที่ 5.19 ผังวงจรระดับ RTL ของมอดูลชุดวงจรคูณค่า SNR	73
รูปที่ 5.20 ผลจำลองการทำงานของของมอดูลชุดวงจรคูณค่า SNR	74
รูปที่ 5.21 ผังวงจรระดับ RTL ของมอดูลปรับเปลี่ยนค่า	75
รูปที่ 5.22 ผลจำลองการทำงานของของมอดูลปรับเปลี่ยนค่า	76
รูปที่ 5.23 ซื่อสัญญาณของชุดกำเนิดสัญญาณรบกวน	77

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูป	หน้า
รูปที่ 5.24 ผังวงจรระดับ RTL ของชุดกำเนิดสัญญาณรบกวน	77
รูปที่ 5.25 ผลจำลองการทำงานของมอดูลชุดกำเนิดสัญญาณรบกวน	79
รูปที่ 5.26 ฮิสโตรแกรมของ X จำนวน 10^6 samples	80
รูปที่ 5.27 ฮิสโตรแกรมของ Y จำนวน 10^6 samples	80
รูปที่ 5.28 ค่า PDF ของ X Y และทฤษฎี	81
รูปที่ 5.29 ค่า relative error ($\xi_x(x)$) ของ X และ Y	81
รูปที่ 5.30 ผังวงจรระดับ RTL ของมอดูลดีมอดูเลเตอร์	83
รูปที่ 5.31 ผลจำลองการทำงานของมอดูลดีมอดูเลเตอร์	83
รูปที่ 5.32 ผังวงจรระดับ RTL ของมอดูลหน่วงเวลา	86
รูปที่ 5.33 ผลจำลองการทำงานของมอดูลหน่วงเวลา	87
รูปที่ 5.34 ผังวงจรระดับ RTL ของมอดูลเปรียบเทียบข้อมูล	88
รูปที่ 5.35 ผลจำลองการทำงานของมอดูลเปรียบเทียบข้อมูล	89
รูปที่ 5.36 ผังวงจรระดับ RTL ของมอดูลเข้ารหัสคอนโวลูชัน K=3 Rate =1/2	90
รูปที่ 5.37 ผลจำลองการทำงานของมอดูลเข้ารหัสคอนโวลูชัน K = 3 Rate = 1/2	90
รูปที่ 5.38 ผังวงจรระดับ RTL ของมอดูลถอดรหัสคอนโวลูชัน	92
รูปที่ 5.39 ผลจำลองการทำงานของมอดูลถอดรหัสคอนโวลูชัน	93
รูปที่ 5.40 ผังวงจรระดับ RTL ของชุดทดสอบรหัสแก้ไขข้อผิดพลาด	96
รูปที่ 5.41 ผลจำลองการทำงานของชุดทดสอบรหัสแก้ไขข้อผิดพลาด	96
รูปที่ 5.42 ผลการทดสอบรหัสคอนโวลูชัน	97

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตาราง	หน้า
ตารางที่ 2.1 Sample Recurrence Equations	14
ตารางที่ 2.2 ตารางการสุ่มโดย Single-LFSR และ Leap-forward LFSR 4 บิต	17
ตารางที่ 3.1 ขนาดของหน่วยความจำที่สามารถโปรแกรมใช้งานได้ของบล็อกหน่วยความจำ	33
ตารางที่ 5.1 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลกำเนิดข้อมูล	55
ตารางที่ 5.2 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลกำเนิดข้อมูล	55
ตารางที่ 5.3 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลมอดูเลเตอร์	57
ตารางที่ 5.4 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลมอดูเลเตอร์	57
ตารางที่ 5.5 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม	60
ตารางที่ 5.6 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม	60
ตารางที่ 5.7 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม	63
ตารางที่ 5.8 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม	63
ตารางที่ 5.9 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลประมาณค่าฟังก์ชัน	65
ตารางที่ 5.10 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลประมาณค่าฟังก์ชัน $g(u_2)$	65
ตารางที่ 5.11 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลชุดวงจรคูณค่าฟังก์ชัน	68

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง (ต่อ)

ตาราง	หน้า
ตารางที่ 5.12 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลชุดวงจรคูณค่าฟังก์ชัน	68
ตารางที่ 5.13 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลประมาณค่า SNR	71
ตารางที่ 5.14 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลประมาณค่า SNR	71
ตารางที่ 5.15 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลชุดวงจรคูณค่า SNR	73
ตารางที่ 5.16 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลชุดวงจรคูณค่า SNR	73
ตารางที่ 5.17 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลปรับเปลี่ยนค่า	75
ตารางที่ 5.18 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลปรับเปลี่ยนค่า	75
ตารางที่ 5.19 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลชุดกำเนิดสัญญาณรบกวน	78
ตารางที่ 5.20 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลชุดกำเนิดสัญญาณรบกวน	78
ตารางที่ 5.21 การทดสอบทางสถิติของชุดกำเนิดสัญญาณรบกวน	81
ตารางที่ 5.22 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลดีมอดูเลเตอร์	82
ตารางที่ 5.23 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลดีมอดูเลเตอร์	82
ตารางที่ 5.24 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลหน่วงเวลา	85
ตารางที่ 5.25 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลหน่วงเวลา	85
ตารางที่ 5.26 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลเปรียบเทียบข้อมูล	88

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง (ต่อ)

ตาราง หน้า

ตารางที่ 5.27 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลเปรียบเทียบข้อมูล 88

ตารางที่ 5.28 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลเข้ารหัสคอนโวลูชัน 90

ตารางที่ 5.29 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลเข้ารหัสคอนโวลูชัน 90

ตารางที่ 5.30 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลถอดรหัสคอนโวลูชัน 92

ตารางที่ 5.31 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลถอดรหัสคอนโวลูชัน 92

ตารางที่ 5.32 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของชุดทดสอบรหัสแก้ไขข้อผิดพลาด 94

ตารางที่ 5.33 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของชุดทดสอบรหัสแก้ไขข้อผิดพลาด 95

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ระบบสื่อสารแบบดิจิตอลเป็นเทคโนโลยีหนึ่งที่มีความสำคัญ และถูกนำมาใช้งานเพื่ออำนวยความสะดวกในชีวิตประจำวัน เช่น การสื่อสารผ่านใยแก้วนำแสง การสื่อสารผ่านดาวเทียม หรือแม้แต่ระบบโทรศัพท์เคลื่อนที่รุ่นที่ 3 (3G) เป็นต้น การรับส่งข้อมูลข่าวสารในระบบสื่อสารดิจิตอลผ่านช่องสัญญาณการสื่อสารอาจเกิดการผิดของบิตข้อมูลขึ้นได้โดยสัญญาณรบกวนซึ่งเป็น สาเหตุหนึ่งที่ทำให้ข้อมูลข่าวสารเกิดการผิดพลาดขึ้นดังนั้นเพื่อเป็นการป้องกัน และแก้ไขความผิดพลาดของบิตข้อมูลดังกล่าวจึงนำรหัสแก้ไขความผิดพลาด (Error Control Coding) มาใช้ในการตรวจสอบและแก้ไขความผิดพลาดของข้อมูลที่เกิดขึ้น

1.1 ความเป็นมาและความสำคัญของปัญหา

การใช้รหัสแก้ไขความผิดพลาดข้อมูล เป็นแนวทางหนึ่งที่ถูกนำมาประยุกต์สำหรับการสื่อสารดิจิตอล ซึ่งสามารถทำให้อัตราการเกิดความผิดพลาดในการส่งข้อมูลอันเนื่องจากผลกระทบของสัญญาณรบกวนมีค่าลดลงได้โดยมิต้องมีการเพิ่มกำลังของเครื่องส่งต้นทาง รหัสแก้ไขข้อผิดพลาดเป็นเทคนิคที่มีความสำคัญมากอย่างหนึ่งจึงได้มีการวิจัยสร้างอุปกรณ์นี้มากมาย ซึ่งโครงการ 3G ของทางศูนย์อิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติร่วมกับห้องปฏิบัติการ DSLABs ของทางสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังได้ทำการพัฒนาชุดต้นแบบชุดถอดรหัส Viterbi ขนาดเล็กสำหรับระบบโทรศัพท์เคลื่อนที่ยุคที่ 3 [1][2][3]

ดังนั้นโครงงานนี้นำเสนอชุดทดสอบประสิทธิภาพของรหัสแก้ไขข้อผิดพลาดสำหรับรหัสคอน-โวลูชัน เพื่อใช้งานกับชุดรหัสแก้ไขข้อผิดพลาดแบบรหัสคอนโวลูชันและชุดถอดรหัส Viterbi [1] ซึ่งโครงงานนี้เป็นส่วนหนึ่งของโครงการพัฒนาต้นแบบสำหรับการสื่อสารแบบไร้สาย โครงการวิจัยระบบโทรคมนาคมสำหรับโทรศัพท์เคลื่อนที่รุ่นที่ 3 ของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ (สวทช.) ร่วมกับห้องปฏิบัติการ DSLABs ของทางสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังโดยชุดทดสอบจะสร้างสัญญาณรบกวน (Noise) เข้าไปผ่านช่องสัญญาณ Additive White Gaussian Noise (AWGN) การหาประสิทธิภาพจะทำโดยการวัดค่าความผิดพลาดของบิต (BER) ต่อค่าอัตราส่วนการใช้กำลังงานในการส่งต่อสัญญาณรบกวน (Eb/No)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 ความมุ่งหมายและวัตถุประสงค์ของโครงงาน

ปริญญานิพนธ์นี้นำเสนอการออกแบบสร้างชุดทดสอบรหัสแก้ไขข้อผิดพลาด (Error Control Coding Tester for Convolution / Viterbi CODEC) โดยการสร้างชุดกำเนิดสัญญาณรบกวนแบบ AWGN ซึ่งเป็นการจำลองสัญญาณรบกวน (Noise Signal) ที่สำคัญอย่างหนึ่งในระบบการสื่อสาร มาใช้ทดสอบรหัสแก้ไขข้อผิดพลาดข้อมูล ในการออกแบบพัฒนาสร้างชุดทดสอบนี้ใช้ภาษาอธิบายฮาร์ดแวร์ (Hardware Description Language: HDL) ในการบรรยายพฤติกรรมการทำงานของชุดทดสอบเพื่อนำไปติดตั้ง (Implementation) ในอุปกรณ์เอฟพีจีเอ (Field Programmable Gate Array) ซึ่งวัตถุประสงค์ของโครงงานมีดังนี้

- สามารถสร้างสัญญาณรบกวนแบบเกาส์ในทางซอฟต์แวร์โดย วิธีเซ็นทรัลลิมิต วิธีบ็อกมูล-เลอร์ วิธีโพลาร์ และวิธีผสม และนำหลักการมาสร้างสัญญาณรบกวนในทางฮาร์ดแวร์[4] ได้
- สามารถนำสัญญาณรบกวนที่สร้างขึ้นมาสร้างเป็น ชุดทดสอบรหัสแก้ไขข้อผิดพลาดเพื่อทดสอบประสิทธิภาพรหัสคอนโวลูชัน และชุดถอดรหัส Viterbi [1] ซึ่งเป็นส่วนหนึ่งของโครงการพัฒนาต้นแบบสำหรับการสื่อสารแบบไร้สาย โครงการวิจัยระบบโทรคมนาคมสำหรับโทรศัพท์เคลื่อนที่รุ่นที่ 3 ของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ร่วมกับ ห้องปฏิบัติการ DSLABs ของทางสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
- สามารถนำชุดทดสอบรหัสแก้ไขข้อผิดพลาดนี้เป็นต้นแบบในการพัฒนาเพื่อทดสอบ รหัสแก้ไขข้อผิดพลาดแบบอื่นๆ ได้

1.3 ขอบเขตของโครงงาน

ขอบเขตของโครงงานชุดทดสอบรหัสแก้ไขข้อผิดพลาดนี้ เริ่มจากการออกแบบสร้างสัญญาณรบกวนแบบ AWGN และนำรหัสคอนโวลูชันมาใช้การทดสอบหาประสิทธิภาพ การออกแบบชุดทดสอบนี้ได้ใช้ภาษาเวริล็อก (Verilog HDL) ในการบรรยายพฤติกรรมการทำงาน ใช้โปรแกรม Xilinx ISE และ ModelSim ในการจำลองการทำงาน การตรวจสอบข้อผิดพลาด การสังเคราะห์วงจร การวางอุปกรณ์และเชื่อมต่อสัญญาณ และการโปรแกรมลงบนอุปกรณ์เอฟพีจีเอ (FPGA) การทำงานชุดทดสอบนี้จะถูกควบคุมการทำงานได้โดยโปรแกรมคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม (RS– 232) และแสดงผลการทดสอบค่า BER ต่อ ค่า Eb/No ระหว่างค่าที่ได้จากข้อมูลที่เข้ารหัสคอนโวลูชันกับไม่ได้เข้ารหัสมาแสดงผลทางคอมพิวเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.4 โครงสร้างของชุดทดสอบรหัสแก้ไขข้อผิดพลาด

ชุดทดสอบการทำงานรหัสแก้ไขข้อผิดพลาดนี้มีโครงสร้างภายในแบ่งส่วนต่างๆได้ดังนี้ ภาคกำเนิดข้อมูล ภาคมอดูเลเตอร์ ภาคกำเนิดสัญญาณรบกวน ภาคดีมอดูเลเตอร์ ภาคหน่วงเวลา ภาคเปรียบเทียบข้อมูล และสุดท้ายเป็นภาคแสดงผล

การทำงานของชุดทดสอบเริ่มจากภาคกำเนิดสัญญาณข้อมูลส่งสัญญาณข้อมูลไปยังชุดเข้ารหัสและส่วนของตัวเปรียบเทียบ จากนั้นส่วนของตัวกำเนิดสัญญาณรบกวนทำหน้าที่กำเนิดสัญญาณรบกวนเพื่อทำให้สัญญาณที่ถูกส่งมาจากภาคเข้ารหัสเกิดผิดเพี้ยนไป สัญญาณที่ได้ถูกส่งออกไปชุดถอดรหัสเพื่อทำการถอดรหัส จากนั้นสัญญาณที่ถูกถอดรหัสจะถูกส่งกลับมายังชุดทดสอบอีกครั้งยังภาคเปรียบเทียบ เพื่อเปรียบเทียบสัญญาณข้อมูลเดิมกับสัญญาณที่ถูกถอดรหัสแล้ว เมื่อทำการเปรียบเทียบแล้วจะส่งค่าอัตราการผิดพลาดของบิต (BER) ไปแสดงผลบนหน้าจอคอมพิวเตอร์ โดยโครงสร้างของชุดทดสอบนี้แสดงดังรูปที่ 1.1

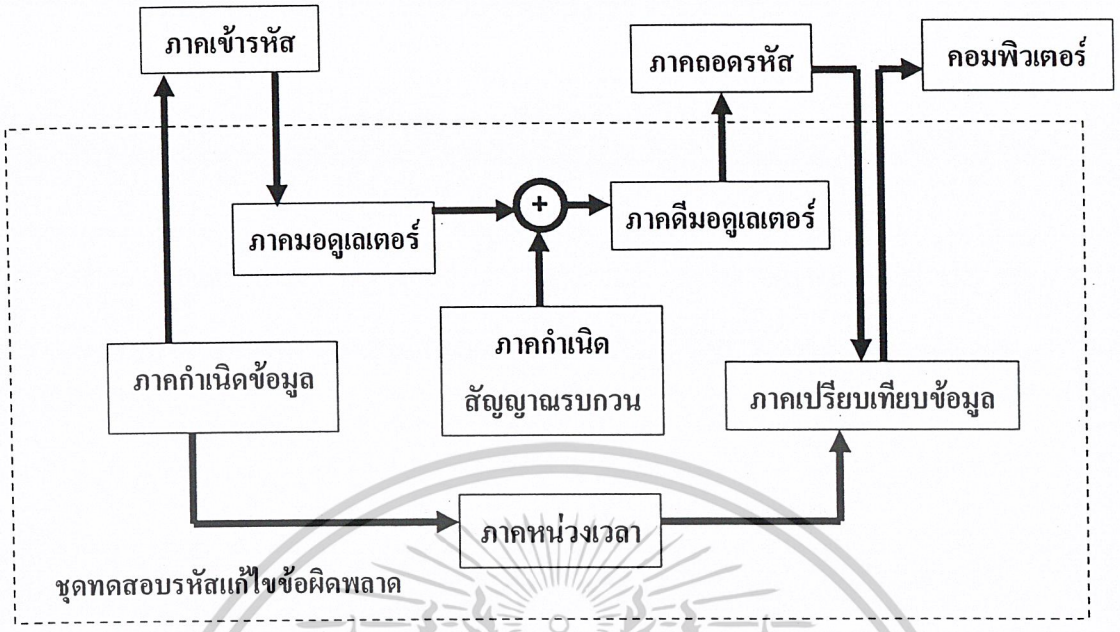
1.5 ขั้นตอนการดำเนินงาน

การสร้างชุดทดสอบรหัสแก้ไขข้อผิดพลาดนี้ ได้แบ่งการทำงานออกเป็นสองส่วน คือ ส่วนแรกเริ่มทำการศึกษาในส่วนของทฤษฎีการสร้างสัญญาณรบกวน และในส่วนที่สองได้นำความเข้าใจจากส่วนแรกมาพัฒนาให้อยู่ในรูปของฮาร์ดแวร์ เป็นชุดทดสอบรหัสแก้ไขข้อผิดพลาดที่สามารถนำมาใช้งานจริงต่อไป

การศึกษาส่วนของทฤษฎีเริ่มจากการสร้างสัญญาณรบกวนแบบเกาส์แบบต่างๆ มาเขียนโปรแกรม เพื่อดูค่าการกระจายตัวแล้วนำค่าที่ได้มาเปรียบเทียบกับ Probability Density Function (PDF) เพื่อดูว่าค่าที่ได้ในแต่ละวิธีให้ผลแตกต่างจาก PDF มากน้อยเพียงใด

จากนั้นทำการออกแบบชุดกำเนิดสัญญาณรบกวนทางฮาร์ดแวร์รวมทั้ง ชุดทดสอบรหัสแก้ไขข้อผิดพลาดแล้วติดตั้งลงบนเอฟพีจีเอ สร้างมอดูลการติดต่อระหว่างเอฟพีจีเอและคอมพิวเตอร์ซึ่งเป็นส่วนของการควบคุมและแสดงผลโดยใช้โปรแกรม Visual Basic 6 ในการเชื่อมต่อระหว่างคอมพิวเตอร์และเอฟพีจีเอผ่านทางพอร์ทอนุกรม (RS-232)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.1 โครงสร้างของชุดทดสอบรหัสแก้ไขข้อผิดพลาด

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
หลักการและทฤษฎีในการสร้างชุดทดสอบรหัสแก้ไขข้อผิดพลาด

ในการสื่อสารระบบดิจิตอลโดยทั่วไปประกอบไปด้วยภาคส่ง (Transmitter: Tx) ภาครับ (Receiver: Rx) และช่องสัญญาณการสื่อสาร (Channel) ในการทำงานของภาคส่งนั้นข้อมูลข่าวสารที่ออกมาจากแหล่งกำเนิดข่าวสาร (Information source) จะต้องถูกเข้ารหัสข้อมูลโดยภาคเข้ารหัสข่าวสาร (Source Coding) เพื่อให้ขนาดข้อมูลข่าวสารที่ถูกส่งลดลงจากนั้นข้อมูลชุดนี้จะถูกเข้ารหัสช่องสัญญาณอีกครั้งที่ภาคเข้ารหัสช่องสัญญาณ (Channel Coding) เพื่อป้องกันและลดข้อผิดพลาดที่เกิดจากสัญญาณรบกวนหรืออื่นๆที่เกิดจากการส่งข้อมูลผ่านช่องสัญญาณการสื่อสารซึ่งข้อมูลที่ถูกเข้ารหัสแก้ไขข้อผิดพลาดนี้ประกอบด้วยข้อมูลข่าวสารและส่วนที่เพิ่มขึ้นมาหรือเรียกว่า รีดันแดนซี (Redundancy) สุดท้ายนี้ข้อมูลจะถูกมอดูเลต (Modulate) เพื่อให้เหมาะสมกับช่องสัญญาณต่อไป

ส่วนในภาครับนั้นการทำงานของส่วนต่างๆจะสอดคล้องกับภาคส่งสัญญาณข้อมูล ที่รับได้จะถูกมอดูเลตกลับคืนมาที่ภาคดีมอดูเลต (Demodulate) ซึ่งข้อมูลที่รับได้นี้อาจมีความผิดพลาด รวมเข้ามาด้วยและข้อมูลนี้จะถูกถอดรหัสและแก้ไขความผิดพลาดที่เกิดขึ้นให้ถูกต้องที่ภาคถอดรหัสช่องสัญญาณ (Channel Decoder) จากนั้นข้อมูลที่ได้จากภาคถอดรหัสช่องสัญญาณจะถูกแปลงให้เป็นข้อมูลข่าวสารเดิมโดยภาคถอดรหัสแหล่งกำเนิด (Source Decoder) โดยบล็อกไดอะแกรมองค์ประกอบพื้นฐานของระบบสื่อสารดิจิตอลทั่วไปนั้นแสดงได้ดังรูปที่ 2.1 **[1]**

2.1 การจำลองสัญญาณในระบบสื่อสาร

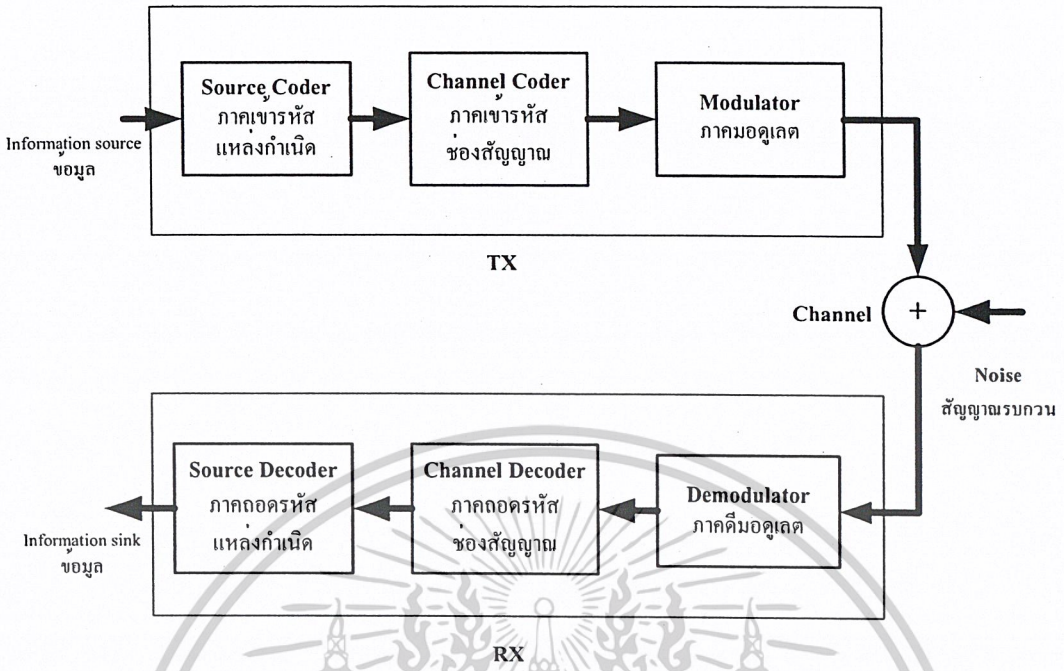
สำหรับระบบสื่อสารที่ส่งสัญญาณโดยใช้คลื่นไซน์ (Sine wave) เป็นสัญญาณพาหะจะมีลักษณะของสัญญาณดังต่อไปนี้

$$s(t) = s_c(t)\cos(2\pi f_c t) + s_s(t)\sin(2\pi f_c t) \qquad (2.1)$$

หรือเขียนให้อยู่ในรูปจำนวนเชิงซ้อนได้ดังนี้

$$s(t) = R(t)e^{j\theta(t)}e^{j2\pi f_c t} \qquad (2.2)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.1 บล็อกไดอะแกรมองค์ประกอบพื้นฐานของระบบสื่อสารดิจิตอล [1]

โดยที่ f_c เป็นความถี่ของสัญญาณพาหะ ในการจำลองลักษณะของสัญญาณในระบบสื่อสารสามารถทำได้โดยการสร้างชุดข้อมูลตามสัญญาณที่เกิดขึ้นจริงแต่เนื่องจากสัญญาณ ในระบบมีความถี่ที่สูงดังนั้นจำนวนข้อมูลที่ต้องสร้างขึ้นมาจึงมากตามไปด้วยดังนั้นการจำลองลักษณะของสัญญาณจึงนำ Envelop ของสัญญาณมาแทนสัญญาณที่ส่งจริง [7] (หรืออาจใช้การแปลงสัญญาณให้อยู่ในรูปของ Equivalent band pass signal [8]) จากสมการที่ (2) Envelop ของสัญญาณสามารถเขียนได้ดังนี้

$$z(t) = R(t)e^{j\theta(t)} \tag{2.3}$$

จากการส่งสัญญาณที่มีการมอดูเลต (Modulate) แบบ Binary Phase Shift Keying (BPSK) ซึ่งมีลักษณะของสัญญาณดังต่อไปนี้

$$s_i(t) = \begin{cases} \cos(2\pi f_c t) & d_{i=0} \\ \cos(2\pi f_c t + \pi) & d_{i=1} \end{cases} \tag{2.4}$$

ซึ่งลักษณะของ Envelop ของสัญญาณที่คาบเวลาต่างๆสามารถเขียนได้ดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$z_{bpsk,i}(t) = \begin{Bmatrix} 1 & d_{i=0} \\ -1 & d_{i=1} \end{Bmatrix} \quad (2.5)$$

ซึ่งจะเห็นว่า Envelop ของสัญญาณมีค่าความถี่ต่ำกว่าสัญญาณที่ทำการส่งจริงจึงเหมาะแก่การจำลองสัญญาณแต่เนื่องจากเป็นการจำลองสัญญาณดังนั้นเมื่อนำไปใช้ในการจำลอง ระบบสื่อสารจึงไม่สามารถวิเคราะห์สัญญาณจริงในระบบสื่อสารได้และในการจำลองสัญญาณรบกวน ในระบบสามารถทำได้โดยนำตัวแปรสุ่มไปรวมเข้ากับสัญญาณที่ทำการส่งถ้าหากใช้สัญญาณจริง ในการส่งสัญญาณรบกวนที่ใช้ในการกวนเมื่อพิจารณาที่ความถี่ f_0 จะได้สัญญาณรบกวนดังนี้

$$n(t) = n_c(t)\cos(2\pi f_0 t) - n_s(t)\sin(2\pi f_0 t) \quad (2.6)$$

เมื่อ n_c และ n_s เป็น Envelop ในแกน in-phase และ Quadrature ซึ่งได้มาจากตัวแปรสุ่มที่มีการกระจายแบบเกาส์ซึ่งมีค่าเฉลี่ยเป็นศูนย์และมีค่าความแปรปรวน (พลังงาน) เป็น σ_n^2 แต่หากใช้ Envelope ของสัญญาณสำหรับจำลองระบบสื่อสาร ลักษณะของสัญญาณรบกวนที่ใช้งานจะมีลักษณะดังสมการ

$$z_n(t) = n_c(t) + jn_s(t) \quad (2.7)$$

2.2 การจำลองสัญญาณรบกวนแบบ Additive White Gaussian Noise (AWGN)

สัญญาณรบกวนเป็นอีกหนึ่งตัวแปรที่สำคัญในระบบสื่อสาร ซึ่งเป็นตัวแปรในการจำกัดอัตราเร็วในการส่งข้อมูลเพื่อง่ายต่อการวิเคราะห์ระบบจะกำหนดสัญญาณรบกวนเป็นแบบ Additive White Gaussian Noise (AWGN) โดยคำว่า

"Additive" นั้นหมายถึงมีการนำสัญญาณหรือข้อมูลมารวมกับสัญญาณรบกวน

"White" นั้นหมายถึงสัญญาณรบกวนจะมีกำลังงานของสัญญาณ ณ ความถี่ต่างๆเท่ากันตลอดทุกความถี่

"Gaussian" ใน AWGN นั้นหมายถึงมีความผิดเพี้ยนในลักษณะของ Gaussian distribution ที่มีนิยามของสมการดังนี้

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(\frac{-x^2}{2\sigma^2}\right)} \quad (2.8)$$

เมื่อ σ^2 คือค่าความแปรปรวน (Variance)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยสัญญาณรบกวนแบบ AWGN มีลักษณะของ Spectral Density และ autocorrelation function ดังสมการต่อไปนี้

$$\Psi_n(f) = \frac{N_0}{2} \qquad ; f = [-\infty, \infty] \tag{2.9}$$

$$\psi_n(\tau) = \delta(\tau) N_0 / 2 \tag{2.10}$$

ในระบบสื่อสารสามารถนำตัวแปรสุ่มมาใช้ในการแสดงลักษณะต่างๆของระบบโดยการจำลองสัญญาณรบกวนสามารถใช้ตัวแปรสุ่มแบบเกาส์มาใช้ในการจำลองซึ่งมี Probability Density Function (PDF) ดังต่อไปนี้

$$p(x) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(x-\bar{x})^2}{2\sigma_n^2} \right) \tag{2.11}$$

เมื่อ x เป็นตัวแปรสุ่มแบบเกาส์ $\bar{x}$ เป็นค่าเฉลี่ยของการสุ่ม และ σ_n^2 เป็นค่าความแปรปรวน (Variance) โดยตัวแปรสุ่มที่แทนสัญญาณรบกวนจะเป็นตัวแปรที่มีการสุ่มแบบเกาส์ซึ่งมีค่าเฉลี่ยเป็นศูนย์และมีค่าความแปรปรวนที่เปลี่ยนแปลงไปตามขนาดของสัญญาณรบกวน และในการนำตัวแปรสุ่มไปใช้แสดงถึงสัญญาณรบกวนในระบบสื่อสาร จะนำตัวแปรสุ่มนี้ไปบวกกับสัญญาณที่ส่งมาเพื่อคำนวณลักษณะของสัญญาณที่รับได้ ดังสมการ

$$r(t) = s(t) + n(t) \tag{2.12}$$

2.2.1 ความสัมพันธ์ระหว่างค่า SNR และค่าความแปรปรวนของสัญญาณ

ผลของสัญญาณรบกวนที่เกิดขึ้นในระบบสื่อสารจะขึ้นอยู่กับค่า Signal to Noise ratio (SNR) และค่า Eb/No ดังนั้น การสร้างสัญญาณรบกวนที่มีการกระจายแบบเกาส์จึงใช้ตัวแปรทั้งสองตัวนี้เป็นตัวควบคุมความแปรปรวนของการกระจายแบบเกาส์ซึ่งโดยทั่วไปจะนิยมนำตัวแปรสุ่มที่มีค่าเฉลี่ยเป็นศูนย์และความแปรปรวนเป็นหนึ่งมาแปลงให้มีความแปรปรวนตามที่ต้องการดังสมการ

$$n(t) = \sqrt{\sigma_n^2} * gngauss(t) \tag{2.13}$$

ซึ่งค่า σ_n^2 มีความสัมพันธ์กับค่า SNR ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\frac{E_s}{N_0} = \frac{nE_b}{N_0} = \frac{1}{2\sigma^2} \tag{2.14}$$

และจากความสัมพันธ์ของ

$$\begin{aligned}\frac{E_s}{N_0} &= \frac{S.T}{(N/B)} \\ &= \left(\frac{S}{N}\right).(T.F_s) \\ \frac{E_s}{N_o} &= SNR\left[\frac{T}{T_s}\right]\end{aligned} \tag{2.15}$$

เมื่อ T คือ คาบของสัญญาณ

B คือ แบนด์วิดในการส่งข้อมูล

F_s คือ ความถี่ที่ใช้ในการสุ่มสัญญาณ (Sampling)

T_s คือ คาบของการสุ่มสัญญาณ

ดังนั้นความแปรปรวนสามารถหาค่าได้ดังนี้

$$\begin{aligned}\sqrt{\sigma_n^2} &= \frac{1}{\sqrt{2\left(\frac{E_s}{N_0}\right)}} \\ &= \frac{1}{\sqrt{2.SNR.\left(\frac{T}{T_s}\right)}}\end{aligned} \tag{2.16}$$

ในกรณีที่ระบบสื่อสารทำการมอดุเลตแบบ M-PSK ที่แต่ละสัญญาณข้อมูลถูกใช้สำหรับแสดงข้อมูลดิจิตอลจำนวน n บิต จะได้ว่า

$$\sqrt{\sigma_n^2} = \frac{1}{\sqrt{2.SNR.n}} \tag{2.17}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.2 วิธีสร้างตัวแปรสุ่มแบบเกาส์

ในการสร้างสัญญาณรบกวนที่มีการกระจายแบบเกาส์ ที่มีค่าเฉลี่ยเป็นศูนย์และค่าความแปรปรวนเป็นหนึ่งนั้นจะสามารถสร้างได้หลายวิธีดังต่อไปนี้

2.2.2.1 ทฤษฎีเซ็นทรัลลิมิต (Central Limit Theorem)

ทฤษฎีเซ็นทรัลลิมิตเป็นวิธีที่ง่ายในการสร้างสัญญาณรบกบกวนแต่มันจะให้เอาต์พุตเมื่อนำไปสร้างสัญญาณทางฮาร์ดแวร์ที่ต่ำ ดังนั้นมันจึงไม่เหมาะที่จะนำมาใช้งานแอพลิเคชัน (Application) ที่ความเร็วสูง **[10]** โดยหลักการออกแบบสัญญาณรบกวนตามทฤษฎีเซ็นทรัลลิมิตมีดังต่อไปนี้

ถ้ากำหนดให้ $x_1, x_2, x_3, ..., x_N$ เป็นตัวแปรสุ่มแบบยูนิฟอร์ม (Uniform Random Variable) มีค่าเฉลี่ย m_x และส่วนเบี่ยงแบนมาตรฐาน σ_x แล้วการกระจายแบบเกาส์สามารถหาค่าได้ดังนี้

$$X_N = \frac{1}{\sigma_x \sqrt{N}} \sum_{i=0}^{N-1} (x_i - m_x) \tag{2.18}$$

จาก (2.18) ถ้าต้องการให้การกระจายตัวมีค่าใกล้เคียงกับการกระจายตัวของ PDF ที่มีสมการดังต่อไปนี้

$$P_x(x) = \frac{1}{\sqrt{2\pi}} e^{-(x-m_x)^2/2} \tag{2.19}$$

นั้นจำนวนตัวแปรสุ่มที่ต้องแรนดัม (Random) เข้ามาจะต้องมีจำนวนใกล้เคียงกับอินฟินิตี้ (Infinity)

โดยที่ X_N เป็นตัวแปรสุ่มแบบเกาส์

x_i เป็นตัวแปรเก็บค่าที่สุ่มมาจำนวน N ตัว โดยที่ตัวแปร N ไม่ขึ้นกับ X_N และมีค่าอยู่ในช่วง [0,1]

m_x เป็นค่าเฉลี่ยของตัวแปร x

N เป็นจำนวนตัวแปรสุ่มแบบยูนิฟอร์ม

2.2.2.2 วิธีบ็อกมูลเลอร์ (Box-Muller Method)

วิธีบ็อกมูลเลอร์ เป็นวิธีที่ถูกนำมาใช้มากทั้งด้านซอฟต์แวร์และฮาร์ดแวร์ **[9],[12]-[14]**ซึ่งมีวิธีการออกแบบดังนี้

ขั้นตอนที่1 ทำการสุ่มค่าตัวแปรสุ่มแบบยูนิฟอร์มสองตัวที่มี การกระจายอยู่ในช่วง [0,1]

ขั้นตอนที่2 นำตัวแปรที่ได้มาแทนค่าในสมการที่ (2.20),(2.21),(2.22)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$f(x) = \sqrt{-\ln(x_1)} \quad (2.20)$$

$$g(x_1) = \sqrt{2}\cos(2\pi x_2) \quad (2.21)$$

$$g(x_2) = \sqrt{2}\sin(2\pi x_2) \quad (2.22)$$

ขั้นตอนที่3 ตัวแปรสุ่มที่มีการกระจายตัวแบบเกาส์สามารถสร้างได้ดังนี้

$$n = f(x_1)\, g(x_2) \quad (2.23)$$

$$n_1 = f(x_1)g(x_3) \quad (2.24)$$

2.2.2.3 วิธีโพลาร์ (Polar Method)

วิธีโพลาร์เป็นวิธีที่พัฒนามาจากวิธีบ็อกมูลเลอร์ โดยยังคงต้องสร้างตัวแปรสุ่มสองตัวเหมือนกัน โดยมีหลักการดังต่อไปนี้

จากหลักการออกแบบโดยวิธีบ็อกมูลเลอร์จะได้

$$N_1 = \sqrt{-2\ln(W)}\cos(\Theta) \quad (2.25)$$

$$N_2 = \sqrt{-2\ln(W)}\sin(\Theta) \quad (2.26)$$

ในการพิจารณาจะพิจารณาจาก $\sqrt{W}\cos\Theta$ และ $\sqrt{W}\sin\Theta$ เพื่อที่จะหลีกเลี่ยงการคำนวณโดยใช้ฟังก์ชัน Sine และ Cosines

ถ้าให้ (Z_1, Z_2) ที่มีการกระจายแบบสม่ำเสมอบนวงกลมหนึ่งหน่วยจะได้ $Z_1 = R\cos\Theta$ และ $Z_2 = R\sin\Theta$ ในระบบพิกัดเชิงขั้ว R, Θ

ถ้าให้ $W = Z_1^2 + Z_2^2$ และ $\cos\Theta = \frac{Z_1}{\sqrt{W}}$ และ $\sin\Theta = \frac{Z_2}{\sqrt{W}}$

ดังนั้นสมการของบ็อกมูลเลอร์สามารถเขียนใหม่ได้ดังนี้

$$N_1 = \sqrt{\frac{-2\ln W}{W}}Z_1 \quad (2.27)$$

$$N_2 = \sqrt{\frac{-2\ln W}{W}}Z_2 \quad (2.28)$$

$$W = Z_1^2 + Z_2^2 \quad (2.29)$$

ถ้า (V_1, V_2) มีการกระจายอยู่ในช่วง $[-1,1]$ แล้ว $0 < V_1^2 + V_2^2 \le 1$ จะได้ $(Z_1, Z_2) = (V_1, V_2)$ (2.30)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.2.4 วิธีผสม (Mixed Method)

วิธีผสมเกิดจากการรวมตัวกันของสองวิธีคือ วิธีเซ็นทรัลลิมิตและวิธีบ็อกมูลเลอร์หรือวิธีโพลาร์ โดยที่ตัวแปรสุ่มแบบเกาส์ที่สร้างได้จะมีอัตราเร็วในการสร้างที่ต่ำแต่ผลที่ได้จะทำให้สัญญาณรบกวนมีคุณภาพสูง [9] แต่จะให้เอาต์พุตที่ช้าเนื่องจากมันประกอบไปด้วยวิธีบ็อกมูลเลอร์ และทฤษฎีเซ็นทรัลลิมิตซึ่งทฤษฎีเซ็นทรัลลิมิตจะให้เอาต์พุตที่ได้ช้าถ้านำไปสร้างทางฮาร์ดแวร์โดย วิธีมิกซ์ที่ใช้วิธีบ็อกมูลเลอร์มาคำนวณ มีสมการดังนี้

$$X = \frac{BM_1 + BM_2 + ... + BM_N}{\sqrt{N}} \tag{2.31}$$

โดยที่ BM เป็นค่าที่ได้จากวิธีของบ็อกมูลเลอร์

N เป็นจำนวนค่าที่นำมาคำนวณ

2.3 การทดสอบการกระจายตัว

ในการทดสอบการกระจายตัวแบบเกาส์เมื่อเปรียบเทียบกับ Probability Density Function (PDF) นั้นสามารถทดสอบได้ดังวิธีต่อไปนี้

2.3.1 Relative Error

Relative Error หรือ $\xi X(x)$ ที่สามารถหาค่าได้โดย

$$\xi X(x) = \frac{X(x) - N(0,1)(x)}{N(0,1)(x)} \tag{2.32}$$

โดย $X(x)$ คือค่า PDF ที่ต้องการทดสอบ

$N(0,1)(x)$ เป็นค่า PDF ตามทฤษฎี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.2 Chi-square

ซึ่งการทดสอบด้วยวิธีนี้จะทำการทดสอบระหว่างผลการทดลองและ PDF ฟังก์ชันซึ่งสามารถหาได้โดย

$$\chi^2 = \sum_{i=1}^{k} \frac{(Y_i - tp_i)^2}{tp_i} \tag{2.33}$$

โดย Y_i คือค่าตัวแปรสุ่มแบบเกาส์ ที่ต้องการทดสอบ

tp_i เป็นค่าที่ได้จาก PDF ตามทฤษฎี

2.4 การสร้างตัวเลขสุ่มโดยใช้ Linear Feedback Shift Register (LSFR)

ตัวเลขสุ่มมีความนิยมใช้ในงานหลายประเภท ไม่ว่าจะเป็นการเข้ารหัส การทดสอบวงจร การจำลองการทำงานของระบบ ซึ่งในที่นี้จะกล่าวถึงการสร้างตัวเลขสุ่มโดยใช้ LFSR ซึ่งมีรายละเอียดดังต่อไปนี้

2.4.1 Single-Bit Random Number Generator Using LFSR

Single-Bit Random Number Generator Using LFSR จะให้ค่าเอาท์พุตเป็น 0 หรือ 1 ซึ่งสามารถหาได้จากสมการ ดังต่อไปนี้

$$x_n = a_1 \bullet x_{n-1} \oplus a_2 \bullet x_{n-2} \oplus a_3 \bullet x_{n-3} \oplus \ldots \oplus a_m \bullet x_{n-m} \tag{2.34}$$

โดยที่ x_i เป็นลำดับที่ i ที่ถูกสร้างขึ้นมา

a_i คือ ค่าคงที่ 0 หรือ 1

$\bullet$ คือ And Operator

$\oplus$ คือ XOR Operator

m เป็นตัวกำหนดค่า period

สำหรับตัวกำเนิดตัวเลขสุ่มแบบสุ่มเทียม (Pseudo Random) รูปแบบที่ได้จะมีค่าซ้ำกันเกิดขึ้นหลังจากที่ได้ทำการสุ่มครบรอบ 1 รอบ (Cycle) แล้ว หรือเรียกว่า Period of Generator ในระบบ Linear Feedback Shift Resister (LFSR) ค่าคาบสูงสุดจะถูกกำหนดโดยค่า m โดยคำนวณได้จาก 2^m-1 เพื่อให้ได้ค่าคาบสูงสุดจะต้องกำหนดให้ a_is เป็น 0 และกำหนดค่า 1 เพียงแค่ 2-4 เท่านั้น ซึ่งวิธีนี้ง่ายต่อการเข้าใจและการสร้าง ถ้าใช้ค่า m ที่แตกต่างกันก็จะทำให้ได้สมการที่แตกต่างกันเช่นกัน ดังตารางที่ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.1 Sample Recurrence Equations **[14]**

M	Recurrence Equation
2	$x_{n-1} \oplus x_{n-2}$
3	$x_{n-1} \oplus x_{n-3}$
4	$x_{n-1} \oplus x_{n-4}$
5	$x_{n-2} \oplus x_{n-5}$
6	$x_{n-1} \oplus x_{n-6}$
7	$x_{n-1} \oplus x_{n-7}$
8	$x_{n-2} \oplus x_{n-3} \oplus x_{n-4} \oplus x_{n-8}$

ในการสร้างทางฮาร์ดแวร์นั้น ค่าใหม่ของ x_n จะขึ้นอยู่กับค่าที่เกิดก่อนหน้าค่า m คือ x_{n-1}, x_{n-1}, ... , x_{n-m} ดังนั้นจึงต้องใช้ 1บิตเรจิสเตอร์ (Register) ในการเก็บค่าของ m และหลังจากค่าใหม่เกิดขึ้น ค่าเก่าที่เก็บอยู่ในเรจิสเตอร์ก็จะถูกแทนที่ด้วยค่าใหม่ ซึ่งจะทำให้ต้องใช้ชิพเรจิสเตอร์ (Shift Register) เท่ากับ m-slot ตัว ซึ่งจะเกิดการเลื่อน ค่าเก่าออกแล้วใส่ค่าให้ใหม่ลงไปแทนทุกรอบสัญญาณนาฬิกา นอกจากจะใช้เรจิสเตอร์แล้ว ยังต้องใช้ XOR เกตด้วย เพื่อทำ การ Exclusive-OR เช่นถ้ากำหนดให้ m = 4 จะได้ว่าจะต้องใช้ 1บิตเรจิสเตอร์ทั้งหมด 4 ตัวด้วยกัน และถ้ากำหนดให้ q_3, q_2, q_1, และ q_0 เป็นค่าเอาต์พุตของเรจิสเตอร์ และกำหนดค่า q_{3_next}, q_{2_next}, q_{1_next}, และ q_{0_next} เป็นค่าอินพุต (ของเรจิสเตอร์) แล้วสมการบูลีน(Boolean)จะได้ดังต่อไปนี้

$$q_{0_next} = q_0$$

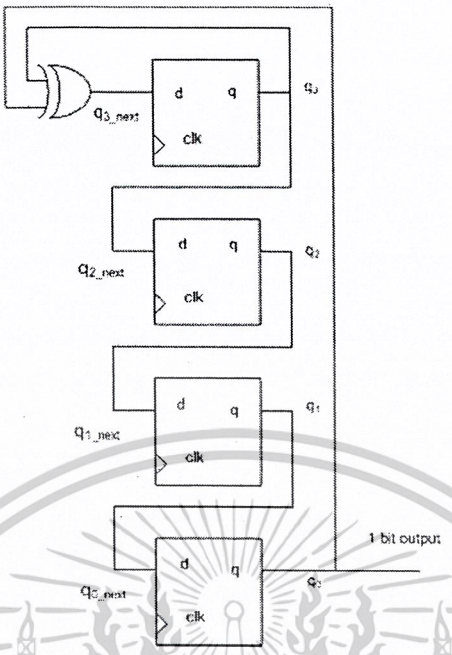
$$q_{1_next} = q_1$$

$$q_{2_next} = q_2$$

$$q_{3_next} = q_3 \oplus q_0$$

จากค่าเหล่านี้สามารถเมื่อนำมาสร้างทางฮาร์ดแวร์จะได้ดังรูปที่ 2.2 จะเห็นได้ว่าในการสร้าง Single-Bit Random Number Generator นั้นมีการใช้ เรจิสเตอร์ เพียง m ตัว และใช้ XOR gate ตั้งแต่ 1 ถึง 3 ตัว ดังนั้นจะทำให้ขนาดของวงจรที่มีขนาดเล็กมาก รวมทั้งการทำงานของวงจรมีความเร็วสูงมากตามไปด้วย ซึ่งจะทำให้มีประสิทธิภาพสูงขึ้นสำหรับค่าเริ่มต้นของวงจร LFSR จะเป็นค่าใดๆ ก็ได้ แต่ยกเว้นค่าศูนย์ทุกบิต เช่น 00…00 ซึ่งถ้าค่าอินพุตเริ่มต้นเป็นศูนย์ทั้งหมดแล้วจะทำให้ลำดับการสุ่มได้ค่าศูนย์ตลอดไป สำหรับการแก้ปัญหานี้ทำได้โดยเพิ่มวงจรเข้าไป เพื่อทำการตรวจจับ 00...00 และ 00...01 ก่อนเลื่อนลงไปในวงจร ซึ่งวงจรนี้เป็นที่รู้จักกันในชื่อว่า de Bruijn counter

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.2 4-บิต LFSR **[14]**

ซึ่งประกอบไปด้วย m-1 อินพุตแอนเกต (And gate) และ extra XOR เกต และมันยังช่วยให้ช่วงคาบเวลาเพิ่มขึ้นจาก 2^m-1 มาเป็น 2^m อีกด้วย ซึ่งสามารถนำมาเขียนเป็นสมการใหม่ได้ ดังนี้

$$q_{0_next} = q_0$$

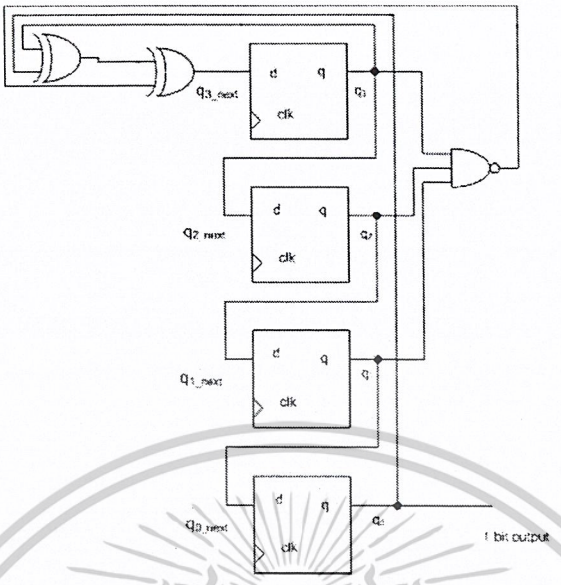
$$q_{1_next} = q_1$$

$$q_{2_next} = q_2$$

$$q_{3_next} = (q_3 \oplus q_0) \oplus (q_3' \cdot q_2' \cdot q_1')$$

และตอนนี้จะได้ค่าของลำดับการสุ่มมี 0 พิเศษเกิดขึ้น ซึ่งแสดงให้เห็นเป็น 0 ขีดเส้นใต้ไว้ดังนี้ 10000 1111 0101 100 10000 1111 0101 100 ... ซึ่งจะทำให้ได้วงจรใหม่ที่แสดงดังรูปที่ 2 และสังเกตเห็นได้ว่า De Bruijn Counter จะไปทำให้ความเป็นเชิงเส้นของระบบเสียไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 4-บิต de Bruijn Counter [14]

2.4.1.1 Single-LFSR Method

สำหรับเทคนิค Single-LFSR Method จะใช้ LFSR เพียงตัวเดียวเท่านั้น และใช้ค่าที่เก็บอยู่ในชิพเรจีสเตอร์ไปกำหนดรูปแบบของ Multiple-Bit Number ยกตัวอย่างเช่น ถ้าต้องการตัวเลขสุ่ม 4-bit เราสามารถใช้เอาต์พุตของเรจีสเตอร์ของ 4-bit LFSR ดังแสดงในรูปที่ 1 (q_0, q_1, q_2 และ q_3) สำหรับลำดับตัวเลขที่ถูกสร้างขึ้นแสดงไว้ในคอลัมน์ซ้ายในตารางที่ 2 ซึ่งสังเกตเห็นได้ว่าจะไม่มี 0000 ในลำดับนั้นโดยวิธีนี้สามารถสร้างได้โดยง่าย แต่ตัวเลขที่สุ่มขึ้นมาได้จะความสัมพันธ์กันสูง และผลการทดสอบทางสถิติก็ยังไม่ดีพอ จึงทำให้วิธีนี้ไม่เป็นที่สนใจเนื่องจากตัวเลขที่สุ่มขึ้นมาใหม่ส่วนใหญ่เป็นตัวเลขที่เกิดจากตัวเลขเดิมที่มีอยู่ และจำกัดเพียงแค่ข้อมูลใหม่ 1บิตเท่านั้น

2.4.1.2 Single-LFSR with a Counter Method

การสร้างด้วยวิธีนี้ประกอบไปด้วยวิธี Single LFSR และวงจรนับซึ่งมันจะทำการแทนที่ด้วยบิตใหม่เข้าไป ตัวเลขที่สุ่มมาได้ k บิต ต้องการ k ชิพโอเปอร์เรเตอร์ (Shift Operator) ของ LFSR เพื่อกำหนดรูปแบบการสุ่มตัวเลขใหม่ สำหรับตัว k จะมีความสัมพันธ์มากกับช่วงเวลาของ LSFR โดยข้อเสียของวิธีคือจะทำงานค่อนข้างจะช้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ตารางที่ 2.2 ตารางการสุ่มโดย Single-LFSR และ Leap-forward LFSR 4 บิต **[14]**

Single LFSR q_0,q_1,q_2,q_3		Leap-forward LFSR q_0,q_1,q_2,q_3	
1000(8)	0101(5)	1000(8)	0011(3)
0001(1)	1011(11)	1111(15)	1101(13)
0011(3)	0110(6)	0101(5)	0110(6)
0111(7)	1100(12)	1001(9)	0100(4)
1111(15)	1001(9)	0001(1)	0111(7)
1110(14)	0010(2)	1110(14)	1010(10)
1101(13)	0100(4)	1011(11)	1100(12)
1010(10)		0010(2)	

2.4.1.3 Parallel-LFSR Method

วิธีนี้ใช้ Single LFSR 1 ตัวในการสุ่มค่าตัวเลข 1 บิตเมื่อต้องการค่า k บิตเกิดขึ้นพร้อมๆกัน จะต้องใช้ Single LFSR k ตัวโดยวิธีนี้จะมีความเร็วในการทำงาน ซึ่งเหมือนกับวิธี Single-Bit Generator แต่ใช้พื้นที่ทางฮาร์ดแวร์มากและเพื่อไม่ให้เกิดความสัมพันธ์ในการสุ่ม จึงต้องเพิ่มวงจร Initialization เข้าไปแต่วงจรมีความซับซ้อนมากขึ้น

2.4.2 Multiple-Bit Leap-Forward LFSR

วิธี Multiple-Bit Leap-Forward จะใช้ LFSR เพียงตัวเดียว และให้ค่าเอาต์พุตออกมาหลายๆบิตพร้อมกันโดยการเลื่อนทั้งหมดจะกระทำพร้อมกันใน 1 สัญญาณนาฬิกา โดยวิธีมีพื้นฐานมาจากการที่ LFSR มีคุณสมบัติเป็นเชิงเส้น และสถานะของเรจิสเตอร์สามารถเขียนให้อยู่ในรูปของเวคเตอร์ (Vector) ได้ดังนี้

$$q(i+1) = A \bullet q(i) \qquad (2.35)$$

จากสมการนี้ q(i+1) และ q(i) เป็นตัวเก็บของชิพเรจิสเตอร์ลำดับที่ (i+1) และ i สเต็บ (Step) และ A เป็น Transition Matrix และถ้าสมมติให้ LSFR ทำงานที่ k สเต็บ จะได้สมการดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
62133

$$
\begin{aligned}
q(i+k) &= A \bullet q(i+k-1) \\
&= A \bullet A \bullet (q(i+k-1)) \\
&= A^{2} \bullet q(i+k-2) \\
&= A^{3} \bullet q(i+k-3) \\
&= \ldots \\
&= A^{k} \bullet q(i)
\end{aligned}
$$

การกำหนดจำนวนและโครงสร้างของ XOR ในวงจรได้จากการคำนวณหาค่า A^k และวงจรใหม่ที่สร้างขึ้นมาได้จะทำงานได้ k สเต็บใน 1 ช่วงสัญญาณนาฬิกา และก็ยังคงประกอบด้วยชิพเรจิสเตอร์ที่เหมือนกัน แม้ว่าวงจรรวมย้อมกลับ (Feedback Combinational Circuitry) มีความซับซ้อนกว่ามาก ในการสร้างวงจรขึ้นมาจริงโดยสามารถอธิบายดังตัวอย่างดังต่อไปนี้

กำหนดให้ใช้ 4-bit LFSR ดังเช่นในรูปที่ 2.3 และนำมาเขียนตามสมการได้ ดังนี้

$$q(i+1) = A \bullet q(i) \qquad (2.36)$$

โดยที่ $q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$ $A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$

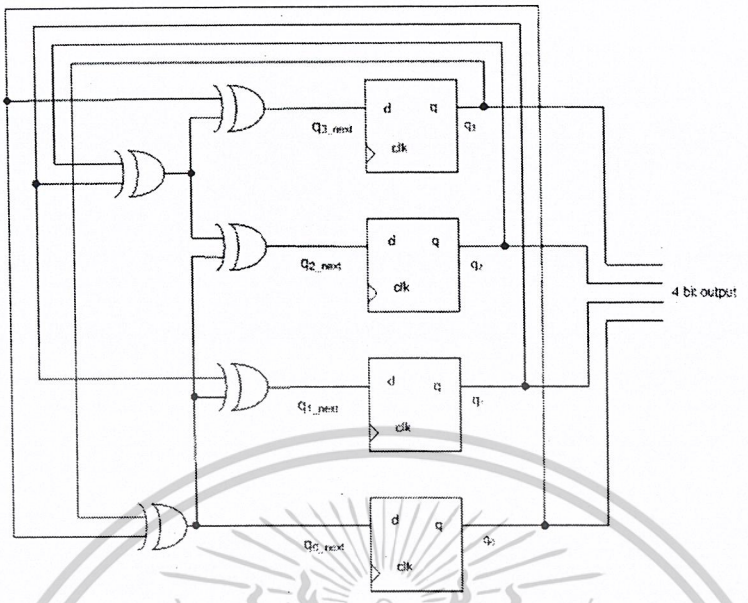
ถ้าต้องการสุ่มตัวเลขจำนวน 4 บิต ดังนั้น k = 4 แล้วจะทำให้ได้เอาท์พุตพร้อมกัน 4 ค่าในเวลาเดียวกัน ดังนี้

$$A^4 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

ดังนั้นสมการของวงจรจึงเป็น

$$\begin{bmatrix} q_{0_next} \\ q_{1_next} \\ q_{2_next} \\ q_{3_next} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.4 4-บิต Leap-forward LFSR [14]

หลังจากทำการคูณเมตริก (Matrix) เสร็จแล้ว จะได้สมการย้อนกลับของแต่ละสัญญาณ ดังนี้

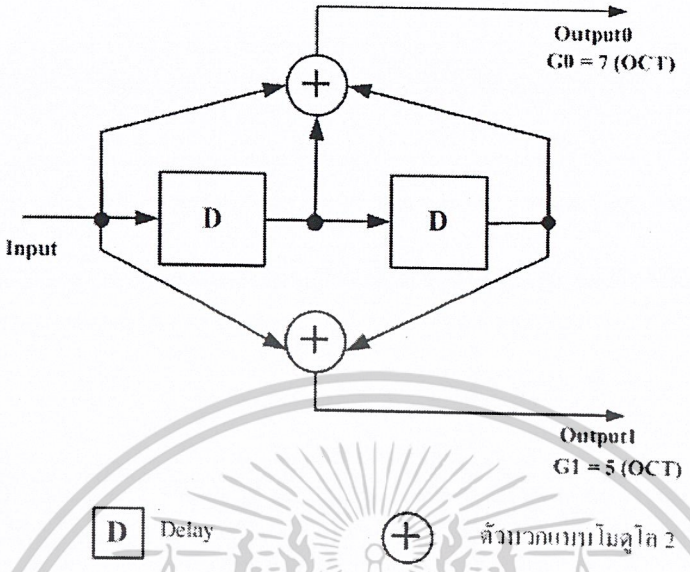
$$\begin{aligned} q_{0_next} &= q_0 \oplus q_3 \\ q_{1_next} &= q_0 \oplus q_1 \oplus q_3 \\ q_{2_next} &= q_0 \oplus q_1 \oplus q_2 \oplus q_3 \\ q_{3_next} &= q_0 \oplus q_1 \oplus q_2 \end{aligned} \tag{2.37}$$

จากสมการนี้นำมาเขียนเป็นวงจรได้ดังรูปที่ 2.4 และค่าลำดับตัวเลขที่สุ่มได้แสดงไว้ในตารางที่ 2.2 ในคอลัมน์ขวาของตาราง

2.5 รหัสแก้ไขข้อผิดพลาด

ในการส่งข้อมูลผ่านช่องสัญญาณการสื่อสารปัจจัยหลักที่ทำให้ข้อมูลเกิดผิดพลาดคือสัญญาณรบกวนดังนั้นเพื่อที่จะลดข้อผิดพลาดให้น้อยลงจึงนำรหัสแก้ไขข้อผิดพลาด มาใช้ซึ่งรหัสคอนโวลูชันเป็นหนึ่งในรหัสแก้ไขข้อผิดพลาดโดยมีรายละเอียดดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



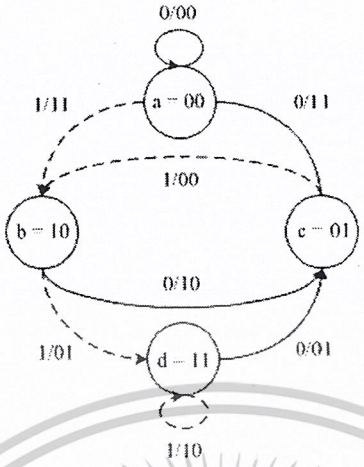
รูปที่ 2.5 วงจรเข้ารหัส Convolution R =1/2 และ $K = 3$

2.5.1 การเข้าและถอดรหัสรหัสคอนโวลูชัน

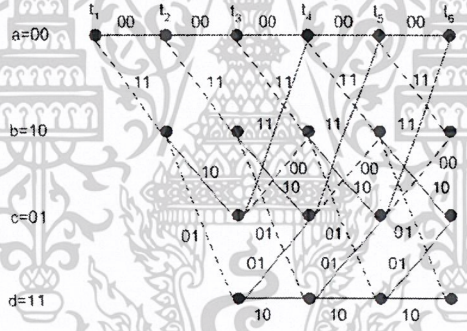
รหัสคอนโวลูชันเป็นรหัสแบบที่สามารถแก้ไขบิตผิดล่วงหน้า การเข้ารหัสจะเริ่มกระทำกับข้อมูลทีละ k บิต และเอาต์พุต ที่ได้จะมีจำนวน n บิต โดยมีอัตราการเข้ารหัส (R) เท่ากับจำนวน Input/Output หรือ (k/n) การเข้ารหัสคอนโวลูชัน ข้อมูลเอาต์พุตจะขึ้นอยู่กับข้อมูลที่เข้ามาใหม่และข้อมูลที่อยู่ในเรจิสเตอร์โดยข้อมูลเอาต์พุตขึ้นอยู่กับระยะข้อมูลที่เรียกว่า Constraint Length ซึ่งมีค่าเท่ากับจำนวนเรจิสเตอร์บวกหนึ่ง และค่า Generator Polynomial (g) ซึ่งเป็นค่าที่ใช้แสดงตำแหน่งข้อมูลอินพุต และข้อมูลในชิพเรจิสเตอร์ที่นำมาบวกแบบมอดูโล 2 ค่า Generator polynomial นี้นิยมเขียนในรูปแบบเลขฐานแปดและจากรูปที่ 2.5 แสดงวงจรการเข้ารหัสคอนโวลูชัน มีอัตราส่วนของการเข้ารหัสเท่ากับ 1/2 ค่า Constraint Length เท่ากับ 3 และมีค่า Generator polynomials G_0 เท่ากับ 7_8 และ G_1 เท่ากับ 5_8 โดยที่ข้อมูล เอาต์พุตขึ้นอยู่กับข้อมูลอินพุตรวมกับข้อมูลในเรจิสเตอร์จากวงจรเข้ารหัสรูปที่ 4.1 สามารถเขียนไดอะแกรมสถานะได้ดังนี้

จากไดอะแกรมสถานะสามารถเขียนไดอะแกรมทรีรีสได้ดังรูปที่ 2.7 ซึ่งข้อมูลเริ่มต้นที่สถานะ 00 หรือสถานะ a เมื่อรับข้อมูลอินพุต ที่มีค่าเป็น “0” เข้ามาเอาต์พุตที่เวลาถัดมาจะมีค่าเป็น “00” ตามเส้นทึบ และถ้าอินพุตเข้ามาเป็น “1” แล้วค่าเอาต์พุตจะเป็น “11”

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.6 ไดอะแกรมสถานะ



รูปที่ 2.7 ไดอะแกรมทรีรีส

2.6 การส่งข้อมูลผ่านพอร์ตอนุกรม (Serial Transmission)

การส่งข้อมูลผ่านพอร์ตอนุกรมเป็นการส่งข้อมูลแบบอะซิงโครนัส (Asynchronous Data Transmission) ซึ่งคณะกรรมการสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association) ได้กำหนดมาตรฐานการเชื่อมต่อแบบอนุกรม EIA RS-232 (x) เป็นมาตรฐานอุตสาหกรรม ซึ่งออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบ อะซิงโครนัส 2 ทิศทาง เพื่อให้มีการใช้งานที่สอดคล้องกัน ระหว่างอุปกรณ์คอมพิวเตอร์ต่างๆ การรับส่งสัญญาณจะกำหนดความยาวไม่เกิน 50 ฟุต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยมีระดับ สัญญาณตั้งแต่ 3 โวลท์ จนถึง 15 โวลท์ สำหรับลอจิก "0" และมีระดับแรงดันที่ -3 โวลท์ จนถึง -15 โวลท์ สำหรับลอจิก "1" ในการส่งข้อมูลผ่านพอร์ตอนุกรมจะใช้สายสัญญาณชนิด DB-9 โดยประกอบด้วยขาต่างๆทั้งหมด 9 ขาซึ่งแต่ละขามีรายละเอียดดังต่อไปนี้

ขาที่ 1 Carrier Detect (DCD) เป็นขาที่ใช้ตรวจสอบสัญญาณเมื่อเจอที่ปลายของสายสัญญาณจะทำให้สายสัญญาณ Active

ขาที่ 2 Receive Data (RXD) เป็นขาที่รับข้อมูลโดยที่จะรับทีละบิต

ขาที่ 3 Transmit Data (TXD) เป็นขาที่ใช้ส่งข้อมูลทีละบิต

ขาที่ 4 Data Terminal Ready (DTR) เป็นสัญญาณที่ทำให้เครื่องคอมพิวเตอร์เตรียมพร้อม

ขาที่ 5 Signal Ground (GND) เป็นสายดิน

ขาที่ 6 Data Set Ready (DSR) เป็นสัญญาณที่บอกให้ทราบอุปกรณ์ที่ติดต่อพร้อมที่จะทำงาน

ขาที่ 7 Request To Send (RTS) เป็นสัญญาณมาจากอุปกรณ์ที่ติดต่อบอกคอมพิวเตอร์พร้อมส่งข้อมูล

ขาที่ 8 Clear To Send (CTS) ตรวจจับสัญญาณจากอุปกรณ์ที่ติดต่อว่าพร้อมรับข้อมูลจากคอมพิวเตอร์

ขาที่ 9 Ring Indicator (RI) เป็นขาที่ตรวจจับสัญญาณจากอุปกรณ์ที่ติดต่อ

ในการส่งข้อมูลผ่านพอร์ตอนุกรมอัตราเร็วของการส่งข้อมูลจะต้องมีค่าเท่ากันระหว่างเครื่องส่งและเครื่องรับซึ่งความเร็วบิตในการรับและส่งข้อมูลจะนับเป็นจำนวนบิตใน 1 วินาที ซึ่งเรียกว่าอัตราบอด (Baud Rate) โดยมีหน่วยเป็นบิตต่อวินาที เช่น 300, 1,200, 2,400, 4,800 และ 9,600 บิตต่อวินาที เมื่อพิจารณาที่อัตราบอดที่ 9600 บิตต่อวินาทีทำให้แต่ละบิตใช้เวลาในการรับส่งข้อมูลเท่ากับ 1/9600 หรือ 104.1 ไมโครวินาที และเวลาในการรับส่งข้อมูลทั้ง 8 บิตจะมีค่าเท่ากับ 8 x 104.1 หรือ 832.8 ไมโครวินาที

การส่งข้อมูลผ่านพอร์ตอนุกรมแบบอะซิงโคนัส (Asynchronous) เป็นวิธีการรับและส่งข้อมูลโดยไม่ต้องอาศัยสัญญาณนาฬิกาส่งร่วมด้วย แต่จะใช้อัตราความเร็วของจำนวนข้อมูลต่อวินาที และจะทำการเพิ่มบิตข้อมูลบางอย่างร่วมไปกับการส่งข้อมูลจริง เพื่อจะได้ทำการตรวจสอบข้อมูลได้อย่างถูกต้องมากยิ่งขึ้นโดยประกอบไปด้วยบิตข้อมูล4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น (Start bit) จะมีขนาด 1 บิต

2. บิตข้อมูล (Data bit) จะเริ่มจาก บิต LSB ก่อน โดยข้อมูลที่ส่งอาจจะมีขนาด 5 ,6, 7 หรือ 8 บิต

3. บิตแสดงสภาวะเลขคู่หรือเลขคี่ (Parity bit) มีขนาด 1 บิตโดยบิตนี้จะนำไปต่อท้ายกับบิตข้อมูล

4. บิตสุดท้ายหรือบิตหยุด (Stop bit) เป็นการระบุถึงขอบเขตของการสิ้นสุดข้อมูล โดยจะทำให้ขาข้อมูลมีสถานะ ลอจิกเป็น "1" ซึ่งอาจมีจำนวนมากกว่า หนึ่งบิตก็ได้ เช่น 1 บิต 1.5 บิต หรือ 2 บิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
เอฟพีจีเอพื้นฐาน

ปัจจุบันเทคโนโลยีวงจรรวม แบบดิจิตอลได้พัฒนาอย่างรวดเร็วซึ่งในบทนี้กล่าวถึงความรู้พื้นฐานวงจรรวมแบบดิจิตอล โครงสร้างพื้นฐานของอุปกรณ์เอฟพีจีเอ Xilinx Vertex-II ตลอดจนเทคนิควิธีการการออกแบบและติดตั้งวงจรดิจิตอลบนอุปกรณ์เอฟพีจีเอ ประวัติความเป็นมาและโครงสร้างของภาษาเวริล็อก (Verilog HDL)

3.1 ความรู้พื้นฐานวงจรรวมแบบดิจิตอล

วงจรรวมแบบดิจิตอลสามารถแบ่งออกได้เป็น 3 ประเภทใหญ่ๆ คือ ลอจิกมาตรฐาน (Standard Logic) พีแอลดี (Programmable Logic Device: PLD) และเอสิก (Application Specific Integrated Circuit: ASIC) ซึ่งดังแสดงในรูปที่ 3.1

3.1.1 วงจรรวมแบบลอจิกมาตรฐาน

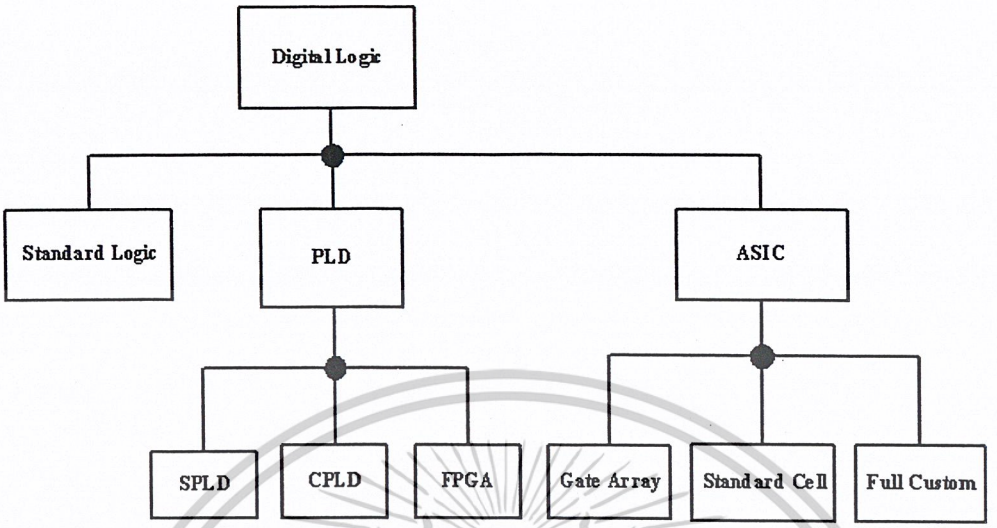
วงจรรวมประเภทลอจิกมาตรฐาน เป็นการออกแบบวงจรดิจิตอลที่ใช้วงจรรวมที่ออกแบบเป็นมาตรฐานไว้แล้ว เช่น วงจรรวมแบบทีทีแอล (TTL) หรือแบบซีมอส (CMOS) ชุดควบคุม เช่น ไมโครคอนโทรลเลอร์ หน่วยความจำต่างๆ เป็นต้น ซึ่งวงจรรวมประเภทนี้จะมีขนาดใหญ่และใช้พลังงานมาก

3.1.2 วงจรรวมแบบพีแอลดี (Programmable Logic Device: PLD)

วงจรรวมดิจิตอลประเภทพีแอลดีเป็นอุปกรณ์ที่สามารถโปรแกรม สร้างวงจรดิจิตอลภายในได้ โดยภายในพีแอลดีนี้จะประกอบด้วยกลุ่มของลอจิกพื้นฐานทั้งวงจรคอมบิเนชัน (Combination) วงจรซีเควลเชียล (Sequential) ซึ่งทำให้สามารถทำให้ผู้ใช้ออกแบบด้วยคอมพิวเตอร์ส่วนบุคคลและสามารถโปรแกรมลงบนพีเอลดีได้หลายครั้งทำให้สะดวกต่อกระใช้งานมากกว่าวงจรรวมประเภทเอสิก

พีแอลดีแบ่งได้เป็น 3 ประเภท คือ เอสพีแอลดี (Simple Programmable Logic Device: SPLD) ซีพีแอลดี (Complex Programmable Logic Device: CPLD) และเอฟพีจีเอ (Field Programmable Gate Array: FPGA) โดยมีรายละเอียดดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.1 โครงสร้างวงจรรวมแบบดิจิตอล [1]

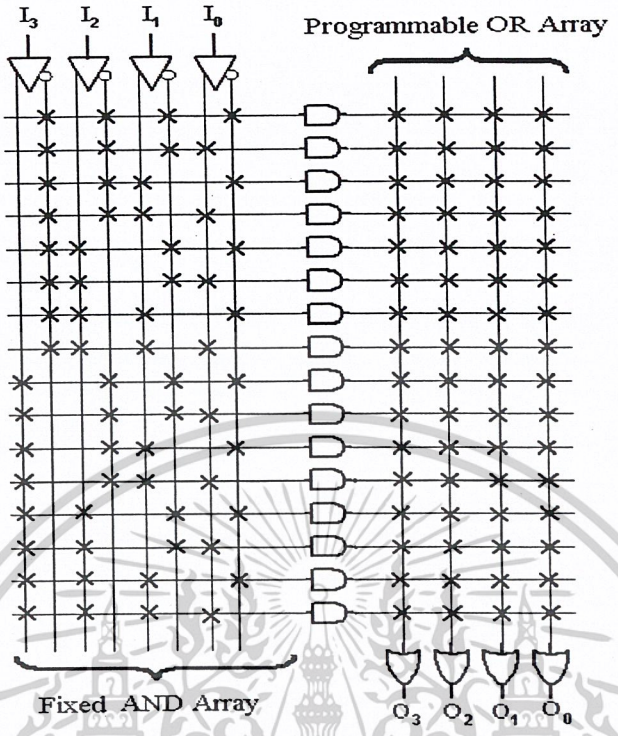
3.1.2.1 อุปกรณ์เอสพีแอลดี (Simple Programmable Logic Device: SPLD)

เอสพีแอลดีเป็นอุปกร์พีแอลดีที่สร้างขึ้นมาเป็นชนิดแรก เพื่อใช้ในการออกแบบวงจรรวมที่มีวงจรลอจิกเกตหลายๆ วงจรเข้าด้วยกัน โดยภายในอุปกรณ์เอสพีแอลดีจะประกอบไปด้วย แอนด์เกต และออร์เกต ซึ่งเรียงในรูปแบบของแถวลำดับ (Array) เมื่อทำการใช้งานผู้ใช้เพียงโปรแกรมให้เกตเหล่านี้เชื่อมต่อเข้าด้วยกันให้เป็นวงจรตามความต้องการ โดยการตัดหรือต่อฟิวส์ในตัวอุปกรณ์ อุปกรณ์เอสพีแอลดียังแบ่งออกเป็นหลายชนิด เช่น พรอม (Programmable Read Only Memory: PROM) พีแอลเอ (Programmable Logic Array: PLA) พีเอแอล (Programmable Array Logic: PAL)

พรอม (Programmable Read-Only Memory: PROM)

พรอมเป็นอุปกรณ์ที่ใช้ทำงานเป็นหน่วยความจำ ซึ่งประกอบด้วยแถวลำดับของแอนด์เกตและออร์เกต โดยเอาต์พุตของพรอมแสดงเป็นสมการฟังก์ชันของผลคูณร่วมบวก (Sum of Product) ของอินพุต แถวลำดับของแอนด์เกตและออร์เกตถูกเชื่อมต่อในลักษณะของวงจรถอดรหัส แถวลำดับของแอนด์เกตต่อกับอินพุตและแถวลำดับของออร์เกตต่อกับเอาต์พุตโดยแถวลำดับของออร์เกต สามารถโปรแกรมเชื่อมต่อได้ โครงสร้างของพรอมแสดงดังรูปที่ 3.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.2 โครงสร้างของพรอม [1]

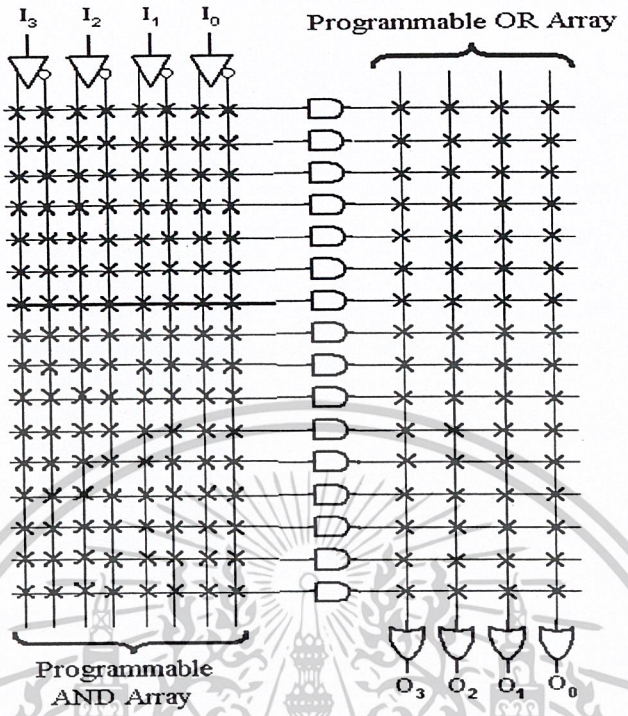
พีแอลเอ (Programmable Logic Array: PLA)

โครงสร้างของพีแอลเอประกอบด้วยแถวลำดับของแอนด์เกตและออร์เกต แถวลำดับของแอนด์เกตต่อกับอินพุตและแถวลำดับของออร์เกตต่อกับเอาต์พุต โดยการใช้งานผู้ใช้สามารถกำหนดการเชื่อมต่อได้ทั้งแถวลำดับของแอนด์เกตและออร์เกต โดยโครงสร้างของพีแอลเอแสดงดังรูปที่ 3.3

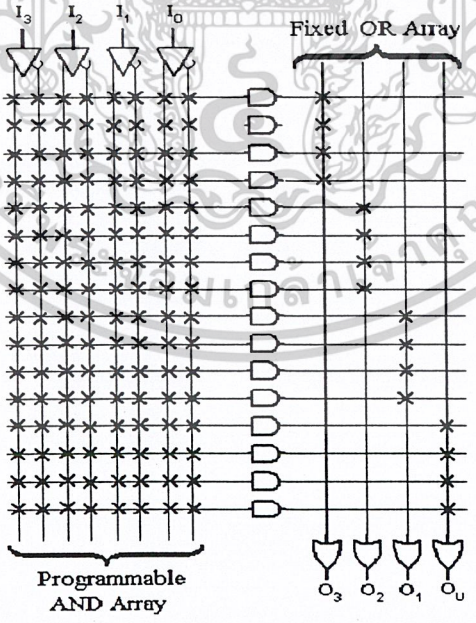
พีเอแอล (Programmable Array Logic: PAL)

โครงสร้างของพีแอลเอประกอบด้วยแถวลำดับของเอนด์เกตที่สามารถโปรแกรมการเชื่อมต่อได้ และแถวลำดับออร์เกตที่ถูกกำหนดการเชื่อมต่อไว้แล้ว โดยแถวลำดับของแอนด์เกตต่อกับอินพุตและแถวลำดับของออร์เกตต่อกับเอาต์พุต และที่เอาต์พุตของออร์เกตอาจมีฟลิปฟลอปต่อไว้เพื่อทำให้พีแอลเอสามารถโปรแกรมวงจรดิจิตอลแบบซีเควนเชียล (Sequential Circuit) ได้ ซึ่งโครงสร้างของพีเอแอลแสดงดังรูปที่ 3.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.3 โครงสร้างของพีแอลเอ [1]



รูปที่ 3.4 โครงสร้างของพีเอแอล [1]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.2.2 อุปกรณ์ซีพีแอลดี (Complex Programmable Logic Device: CPLD)

เป็นการรวมเอสพีแอลดี (SPLD) หลายตัวหรือเรียกว่า ไมโครเซลล์ (Macro Cells) เข้าด้วยกัน เป็นอุปกรณ์เพียงตัวเดียว ทำให้ซีพีแอลดีมีความจุของจำนวนเกตมากขึ้นกว่าเอสพีแอลดี โครงสร้างของซีพีแอลดีประกอบด้วย เกตต่างๆ ฟลิปฟลอป และส่วนที่ใช้เชื่อมต่อภายใน (Interconnect) ซึ่งโครงสร้างของซีพีแอลดีแสดงดังรูปที่ 3.5

3.1.2.3 อุปกรณ์เอฟพีจีเอ (Field Programmable Gate Array: FPGA)

เอฟพีจีเอ (Field Programmable Gate Array :FPGA) เป็นอุปกรณ์ที่สามารถโปรแกรมได้ เอฟพีจีเอนั้นจะมีจำนวนของเกตมากกว่าอุปกรณ์ซีพีแอลดีมาก โดยข้อดีเอฟพีจีเอนั้นผู้ใช้สามารถออกแบบโดยใช้คอมพิวเตอร์ส่วนตัวและยังสามารถโปรแกรมได้หลายครั้งซึ่งทำให้ง่ายต่อการใช้งาน และเป็นที่นิยมมากในปัจจุบันโครงสร้างภายในของเอฟพีจีเอแบ่งได้เป็น 3 ส่วนคือ บล็อกลอจิก บล็อกเชื่อมต่อภายใน และไอโอเซลล์ (I/O Cell) ดังรูปที่ 3.6 โดยลอจิกบล็อกจะอยู่ในรูปเมตริก ภายในประกอบไปด้วย วงจรเกตต่างๆ เรจิสเตอร์ หน่วยความจำ เพื่อใช้สร้างฟังก์ชันในการใช้งานโดยมีบล็อกเชื่อมต่อภายในทำหน้าที่เชื่อมต่อระหว่างบล็อกลอจิกและเชื่อมต่อกับไอโอเซลล์ ซึ่งทำหน้าที่เป็นอินพุตและเอาต์พุตเพื่อติดต่อกับวงจรภายนอก

3.1.2.4 เทคโนโลยีการโปรแกรมพีแอลดี

การโปรแกรมอุปกรณ์พีแอลดีสามารถโปรแกรมได้โดยไฟฟ้า (Electrical Programmable Switch) โดยเทคโนโลยีการโปรแกรมอุปกรณ์พีแอลดีมีดังนี้

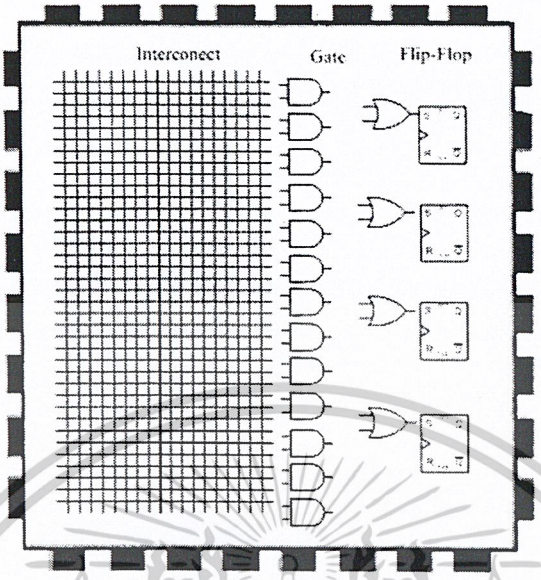
เทคโนโลยี Anti-Fuse

การโปรแกรมแบบนี้จะใช้แรงดันไฟฟ้าทำให้มีเปลี่ยนแปลงค่าของความต้านทานจากค่าสูงเป็นค่าต่ำ ทำให้ขนาดวงจรเล็ก แต่สามารถ โปรแกรมได้เพียงครั้งเดียว อุปกรณ์ที่ใช้เทคโนโลยีนี้ในการโปรแกรม เช่น พรอม พีเอแอล เป็นต้น

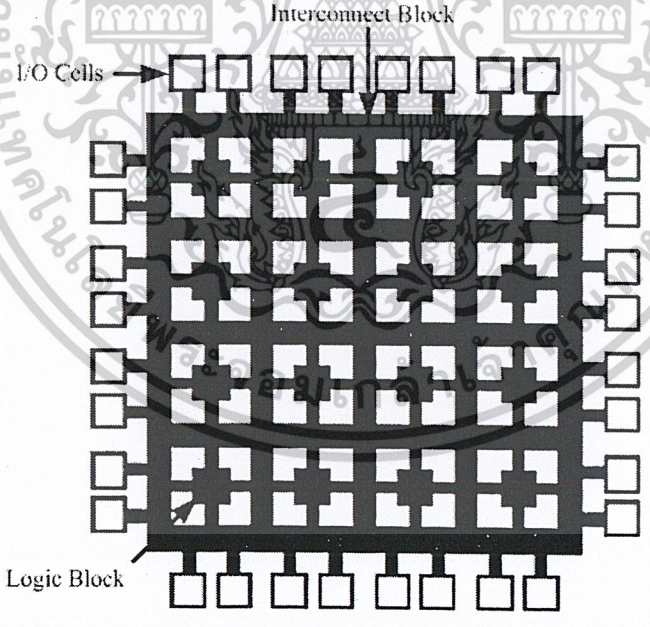
เทคโนโลยี EPROM/EEPROM

การโปรแกรมรูปแบบนี้จะทำการสร้าง EPROM หรือ EEPROM ไว้ในตัวของพีแอลดีโดยควบคุมผ่าน Pass Transistor หรือ Multiplexer ซึ่งการโปรแกรมแบบนี้ สามารถโปรแกรมซ้ำได้เหมือนเทคโนโลยี Static RAM ซึ่งข้อมูลจะยังคงอยู่แม้ไม่มีการจ่ายพลังงานให้รวมทั้งไม่ต้องใช้หน่วยความจำภายนอกในการเก็บข้อมูลของการโปรแกรมแต่การโปรแกรมจะช้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.5 โครงสร้างของซีพีแอลดี [1]



รูปที่ 3.6 โครงสร้างทั่วไปของเอฟพีจีเอ [1]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขนาดวงจรใหญ่จึงไม่สามารถสร้างอุปกรณ์ชนิดนี้ให้มีความจุเกตสูงได้ ตัวอย่าอุปกรณ์ชนิดนี้เช่น อุปกรณ์ซีพีแอลดี เป็นต้น

เทคโนโลยี Static RAM

การโปรแกรมแบบนี้จะใช้ Static RAM ในการกำหนดการเชื่อมต่อวงจร โดยจะกำหนดผ่าน Pass Transistor หรือ Multiplexer ซึ่ง Pass Transistors เป็นตัวทำให้เกิดการเชื่อมต่อระหว่างเทอร์มินอล (Terminal) ทั้งสองข้าง ส่วนตัว Multiplexer จะเป็นตัวควบคุมการเชื่อมต่อระหว่างเอาต์พุตและอินพุตตัวใดตัวหนึ่ง เทคโนโลยีแบบนี้จะสามารถโปรแกรมได้เร็วและไม่จำกัดจำนวนครั้งในการโปรแกรม รวมทั้งวงจรจะมีขนาดเล็กทำให้มีความจุของเกตสูงแต่ ต้องมีการใช้หน่วยความจำภายนอกร่วมด้วยเพื่อเก็บข้อมูลที่จะโปรแกรมและเมื่อไม่ได้จ่ายไฟเลี้ยงให้ข้อมูลจะหายไปทำให้ต้องโปรแกรมใหม่เมื่อเริ่มจ่ายพลังงานให้กับอุปกรณ์ ตัวอย่างอุปกรณ์แบบเช่น อุปกรณ์เอฟพีจีเอ เป็นต้น

3.1.3 วงจรรวมแบบเอสิก (Application Specific Integrated Circuit: ASIC)

วงจรรวมประเภทเอสิก (Application Specific Integrated Circuit: ASIC) จะนิยมใช้ในเชิงพาณิชย์ เมื่อทำการจำลองการทำงานเสร็จก็ส่งไฟล์ที่ได้จากการออกแบบนี้ให้บริษัททำการเจือสารเพื่อสร้างวงจรรวม ซึ่งจะใช้ระยะเวลานานตั้งแต่การออกแบบจนได้รับวงจรรวมรวมทั้งเสียค่าใช้จ่ายในการเจือสารสูงโดยวงจรรวมประเภทนี้แบ่งได้เป็น 3 ประเภท คือ

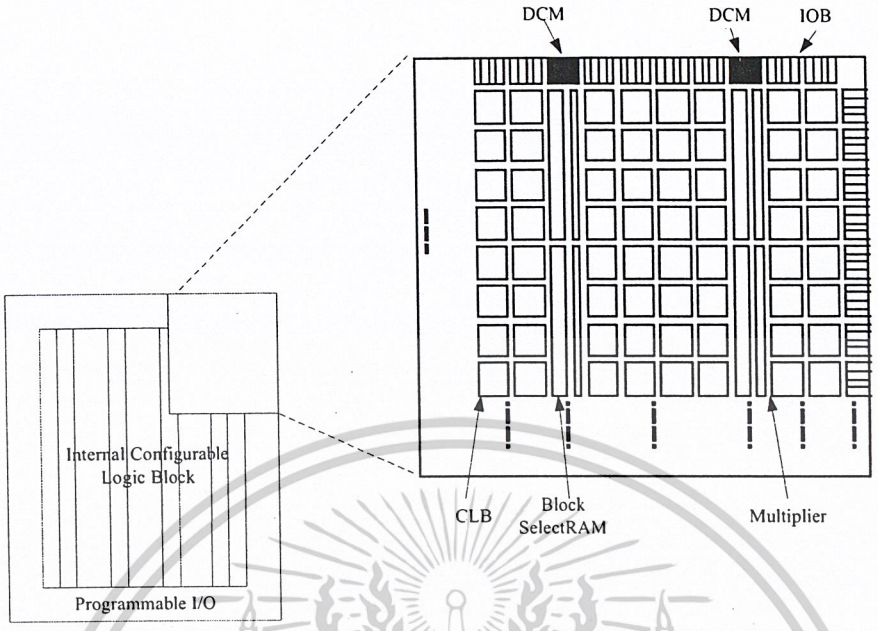
3.1.3.1 วงจรรวมเกตอาเรย์

วงจรรวมประเภทเกตอาเรย์ (Gate Array) นี้ประกอบไปด้วยแถวลำดับของเกตกับขั้วต่อสายไฟ (Pad) สำหรับเชื่อมต่อกับวงจรภายนอกเมื่อผู้ใช้ทำการออกแบบการเชื่อมต่อระหว่างเกตและขั้วต่อสายไฟเสร็จแล้วจึงนำไฟล์ที่ได้ออกแบบไปให้บริษัทผู้ผลิตทำการเจือสาร

3.1.3.2 วงจรรวมแบบเซลล์มาตรฐาน

วงจรรวมประเภทนี้ผู้ใช้เลือกกลุ่มวงจรที่ทำหน้าที่ต่างๆ เช่น เกต ฟลิบฟลอป ตัวนับ หน่วยความจำ หรือไมโครโปรเซสเซอร์ จากแฟ้มข้อมูลของบริษัทผู้ผลิตมาทำการออกแบบวงจรรวม ซึ่งการผลิตด้วยวิธีนี้จะต้นทุนที่สูงรวมทั้งใช้ระยะเวลาการเจือสารที่นาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่3.7 โครงสร้างของเอฟพีจีเอ Xilinx Virtex-II [1]

3.1.3.3 วงจรรวมแบบฟูลัสตัม

วงจรรวมประเภทนี้ผู้ใช้เป็นผู้ออกแบบเองทั้งหมดเมื่อได้ทำการออกแบบเสร็จ แล้วผู้ใช้จะส่งไฟล์ที่ได้ทำการออกแบบนี้ไปให้บริษัทผู้ผลิตเจือสารเพื่อทำให้ประหยัด การเจือสารรวมทั้งการออกแบบลายวงจรลงบนแผ่นซิลิกอนแผ่นเดียวกัน

3.2 โครงสร้างของเอฟพีจีเอ Xilinx Vertex-II

ในที่นี้จะกล่าวถึงเอฟพีจีเอตระกูล Vertex-II ของบริษัท Xilinx ซึ่งเอฟพีจีเอตระกูลนี้ถูกออกแบบให้มีประสิทธิภาพและความจุของเกตสูง ตั้งแต่ 40K เกตจนถึง 8M เกต ในกระบวนการผลิตเอฟพีจีเอตระกูล Vertex-II นี้ใช้เทคโนโลยีแบบ CMOS 0.15 um/0.12 um 8-Layer Metal ทำให้เอฟพีจีเอตระกูลนี้มีความเร็วในการทำงานสูงและมีการใช้พลังงานน้อย [17] ซึ่งมีรายละเอียดดังต่อไปนี้

3.2.1 โครงสร้างภายในเอฟพีจีเอ Xilinx Vertex-II

โครงสร้างของเอฟพีจีเอตระกูล Vertex-II ประกอบด้วยสองส่วนโดยส่วนแรกจะเป็นส่วนของบล็อกอินพุตเอาต์พุต และส่วนที่สองจะเป็นส่วนของ Internal Configurable Logic Block (ICLB) ซึ่งแสดงดังรูปที่ 3.7 [17] ซึ่งส่วน ICLB นั้นประกอบด้วย 4 อุปกรณ์หลัก คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- บล็อก Configurable Logic Block (CLB) ประกอบด้วยวงจรลอจิกแบบคอมบิเนชัน (Combinatorial) และแบบซิงโครนัส (Synchronous) เรจิสเตอร์และบัฟเฟอร์
- บล็อก SelectRAM เป็นหน่วยความจำแบบ Dual-port RAM ที่สามารถเลือกขนาดได้ โดยมีขนาดสูงสุด 18 กิโลบิต
- บล็อก Multiplier เป็นบล็อกวงจรคูณขนาด 18 x 18 บิต
- บล็อก DCM (Digital Clock Manager) เป็นบล็อกที่จัดการกับสัญญาณนาฬิกา ซึ่งบล็อกนี้สามารถคูณหรือหารความถี่สัญญาณนาฬิกา และนอกจากนั้นยังสามารถเลื่อนเฟส (Phase Shift) ของสัญญาณนาฬิกาได้

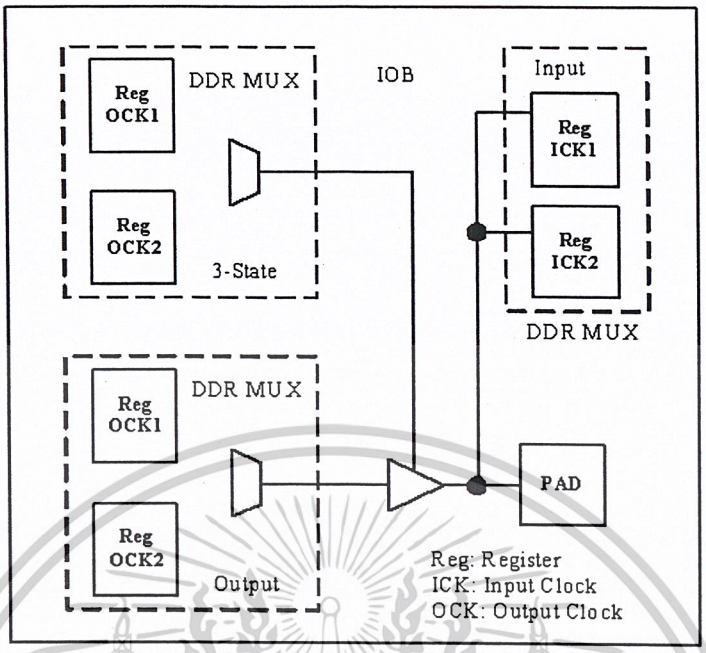
3.2.2.1 บล็อกอินพุตเอาต์พุต

บล็อกอินพุตเอาต์พุต (IOB) เป็นส่วนที่ใช้ติดต่อกับวงจรภายนอก ซึ่งบล็อกอินพุตและเอาต์พุตสามารถเปลี่ยนให้เป็นอินพุตหรือเอาต์พุตแบบ Single Data Rate หรือแบบ Double Data Rate (DDR) ได้โดยการโปรแกรมภายในของบล็อกอินพุตเอาต์พุตนี้ประกอบด้วย 6 รีจิสเตอร์ (Register) และ 3 DDR Multiplexers ดังรูปที่ 3.8 และ 3.9 บล็อกอินพุตเอาต์พุตนี้ต่ออยู่กับส่วน Switch Matrix เพื่อเชื่อมต่อกับอุปกรณ์ภายในอื่นๆ ส่วน DDR Multiplexer ใช้ในการส่งข้อมูลแบบ DDR

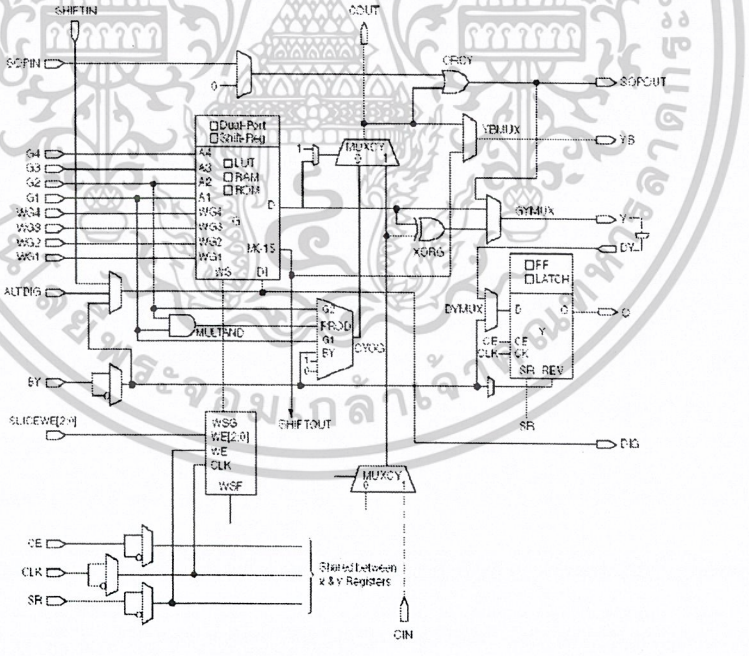
3.2.2.2 บล็อก Configurable Logic Block (CLB)

บล็อกนี้จะประกอบด้วย วงจรลอจิกแบบคอมบิเนชัน และแบบซิงโครนัส ซึ่งต่ออยู่กับ Switch Matrix เพื่อเชื่อมต่อกับอุปกรณ์ภายในอื่นๆ โดยภายใน CLB จะประกอบด้วย Slices จำนวน 4 Slices ซึ่ง Slice จะประกอบด้วยอุปกรณ์ดังนี้ คือ ตัวสร้างฟังก์ชันสี่อินพุต จำนวน 2 ตัว วงจรลอจิกตัวทด วงจรบวก วงจรมัลติเพล็กขนาดใหญ่ และหน่วยความจำจำนวน 2 ตัวโดยตัวสร้างฟังก์ชันสี่อินพุตสามารถโปรแกรมให้เป็นสิ่งต่างๆไม่ว่าจะเป็นตัวเปิดตารางสี่อินพุต (4-input Look-upTable) หน่วยความจำขนาด 16 บิต ชิพเรจิสเตอร์ 16 บิตที่สามารถปรับเปลี่ยนได้ โดยเอาต์พุตของส่วนนี้จะต่อกับหน่วยความจำ ซึ่งเชื่อต่อกับอุปกรณ์อื่นๆ โดยหน่วยความจำนี้ยังสามารถโปรแกรมให้เป็นดีฟลิปฟลอป (D Flip-Flop) หรือแลตช์ (Latch) ดังรูปที่ 3.9

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.8 โครงสร้างของบล็อกอินพุตเอาต์พุต [1]



รูปที่3.9 โครงสร้างของ Slice [1]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.1 ขนาดของหน่วยความจำที่สามารถโปรแกรมใช้งานได้ของบล็อกหน่วยความจำ **[7]**

16K x 1 บิต	2K x 9 บิต
8K x 2 บิต	1K x 18 บิต
4K x 4 บิต	512 x 36 บิต

3.2.2.3 บล็อกหน่วยความจำ 18 กิโลบิต

ในส่วนของบล็อกนี้สามารถโปรแกรมให้เป็นหน่วยความจำแบบ Single-port RAM หรือหน่วยความจำแบบ Dual-port RAM และยังเลือกขนาดจาก 16 K x 1 บิตจนถึง 512 x 32 บิต รวมทั้งสามารถเลือกการทำงานเป็นซิงโครนัสหรืออะซิงโครนัสตารางที่ 3.1 แสดงขนาดหน่วยความจำที่สามารถเลือกโปรแกรมให้ใช้งานได้ของบล็อกหน่วยความจำนี้

3.2.2.4 บล็อกวงจรคูณขนาด 18 x 18 บิต

บล็อกวงจรคูณสามารถปรับเปลี่ยนขนาดได้ ซึ่งขนาดสูงสุดคือ 18 x 18 บิต และยังสามารถคูณแบบ 2's Complement บล็อกวงจรคูณนี้สามารถทำงานอิสระหรือทำงานร่วมกับบล็อกหน่วยความจำ 18 กิโลบิตซึ่ง "Multiply-Accumulator (MAC)" เป็นฟังก์ชันการทำงานในการประมวลผลสัญญาณดิจิตอล (Digital Signal Processing) โดยรูปที่ 3.10 แสดงการเชื่อมต่อของบล็อกหน่วยความจำ 18 กิโลบิตและบล็อกวงจรคูณขนาด 18 x 18 บิต ซึ่งบล็อกทั้งสองจะถูกเชื่อมต่อกับ Switch Matrix

3.2.2.5 บล็อก Digital Clock Manager (DCM)

บล็อก DCM (Digital Clock Manager) เป็นบล็อกที่จัดการสัญญาณนาฬิกาของเอฟพีจีเอ โดยการทำงานใช่หลักการ Digital Delay Line และการป้อนกลับ ซึ่งลักษณะการทำงาน 3 แบบ ดังนี้

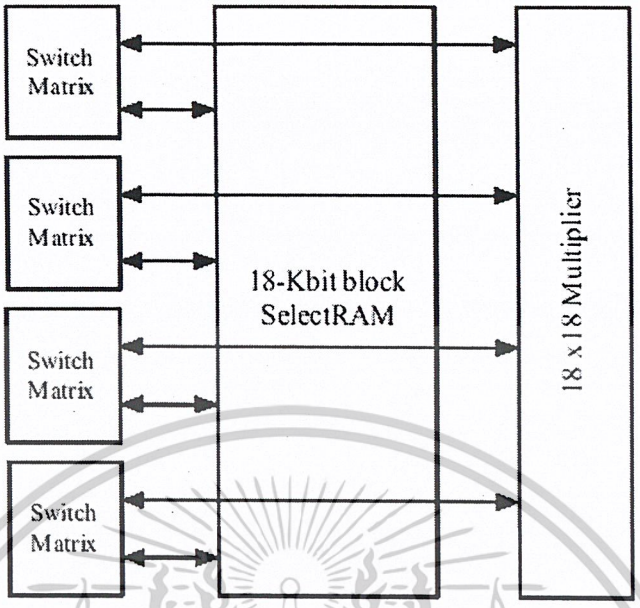
- **Clock De-skew**

การทำงานนี้จะ เป็นบล็อกที่คอยจัดการสัญญาณนาฬิกาของเอฟพีจีเอใหม่ ซึ่งมีเฟสตรงกับอินพุตของสัญญาณนาฬิกา

- **Frequency Synthesis**

การทำงานนี้ DCM สามารถกำเนิดความถี่สัญญาณนาฬิกาใหม่ โดยใช้การคูณหรือการหารสัญญาณนาฬิกาอินพุต โดยในเอฟพีจีเอตระกูลนี้จะมีตัวหารคือ 1.5, 2, 2.5, 3, 3.5, 4, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15 และ 16 ส่วนการคูณนั้นคือ M/D โดย M และ N เป็นจำนวนเต็ม แต่ค่าความถี่ที่ได้จากการคูณต้องไม่เกิน DCM Timing Parameter ของเอฟพีจีเอแต่ละรุ่น (สูงสุดไม่เกิน 350 MHz ในเอฟพีจีเอตระกูล Vertex II) **[17]**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.10 การเชื่อมต่อบล็อกหน่วยความจำขนาด 18 กิโลบิตและบล็อกวงจรคูณขนาด 18×18 บิต [1]

- **Phase shifting**

DCM สามารถเลื่อนเฟสของสัญญาณนาฬิกาได้ ซึ่งการเลื่อนเฟสของสัญญาณนาฬิกา สามารถทำได้ 90, 180 และ 270 องศา

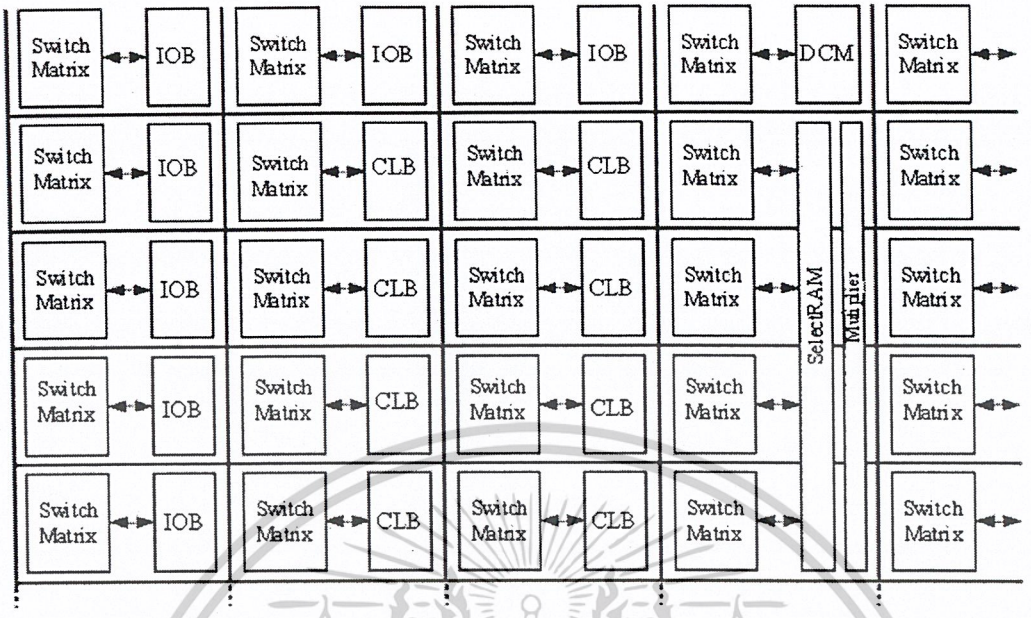
3.2.2.6 บล็อกเชื่อมต่อและหาเส้นทาง

การเชื่อมต่อและหาเส้นทางจะใช้อุปกรณ์ Switch Matrix ในการเชื่อมต่ออุปกรณ์ภายในเข้าด้วยกันเรียกวิธีนี้ว่า "Active Interconnect Technology" โดยตัวอย่างของอุปกรณ์ภายในเช่น CLB, IOB, Block RAM. Multiplier และ DCM ซึ่งทำให้การเข้าถึงอุปกรณ์ต่างๆ ทำได้อย่างรวดเร็ว ซึ่งรูปที่ 3.11 แสดงโครงสร้างการเชื่อมต่อและหาเส้นทางของเอฟพีจีเอตระกูล Vertex-II

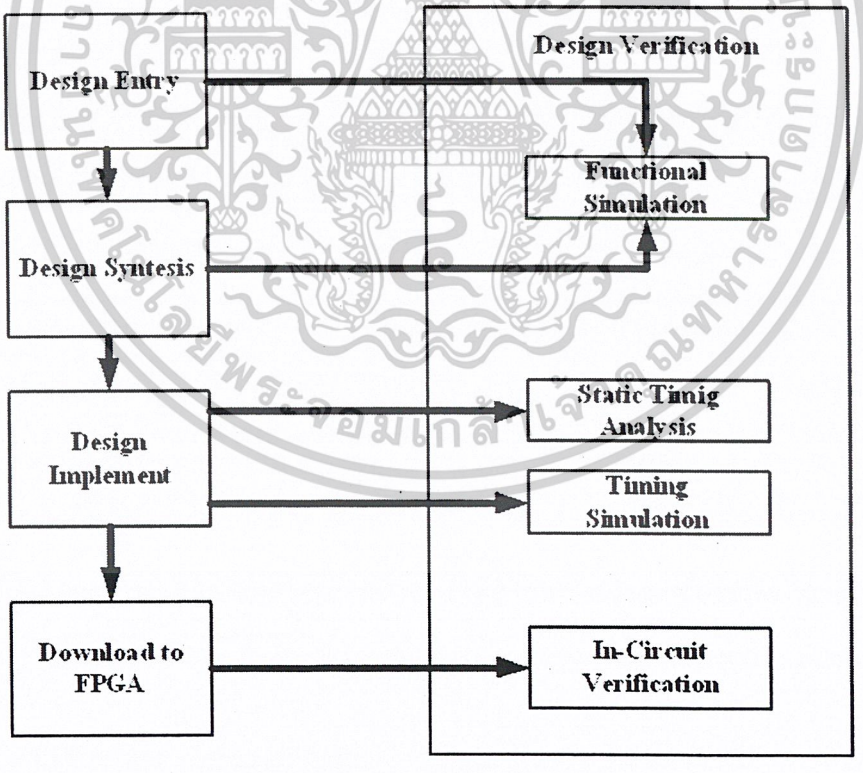
3.3 การออกแบบและติดตั้งวงจรดิจิตอลบนเอฟพีจีเอ

การออกแบบวงจรดิจิตอลเพื่อติดตั้งกับเอฟพีจีเอนั้นใช้ภาษาอธิบายฮาร์ดแวร์ (Hardware Description Language: HDL) เช่น ภาษาวีเอชดีแอล (VHDL) หรือภาษาเวริล็อก (Verilog HDL) ในการออกแบบวงจร จากนั้นนำโค้ดที่ได้นี้ไปสังเคราะห์ (Synthesis) เป็นวงจร ผลการสังเคราะห์ที่ได้คือ เน็ตลิสต์ในระดับเกต (Gate-level Netlist Format) ซึ่งใช้สำหรับเอฟพีจีของบริษัท Xilinx **[5]** ขั้นตอนการออกแบบและติดตั้งวงจรดิจิตอลบนเอฟพีจีเอ **[18]** แสดงดังรูปที่ 3.12 โดยมีรายละเอียดดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.11 โครงสร้างของบล็อกเชื่อมต่อและการหาเส้นทาง [1]



รูปที่ 3.12 ขั้นตอนการออกแบบและติดตั้งวงจรดิจิตอลบนเอฟพีจีเอ [1]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **การออกแบบ**

การออกแบบวงจรเริ่มต้นจาก การเขียนภาษาอธิบายฮาร์ดแวร์หรือเอชดีแอล (HDL) ในการอธิบายพฤติกรรม (Behavioral) การทำงานของวงจรซึ่งการเขียนภาษาเอชดีแอลสามารถใช้โปรแกรม Text Editor ทั่วไปหรือใช้ซอฟต์แวร์ที่สนับสนุนการเขียนภาษาเอชดีแอล ซึ่งทำให้ง่ายในการเขียนและยังช่วยในการตรวจสอบความถูกต้องตามหลักไวยากรณ์เช่น ModelSim ของบริษัท Model Technology เป็นต้น หลังจากการออกแบบวงจรด้วยภาษาเอชดีแอลแล้วจะต้องทำการ ตรวจสอบวงจรที่ออกแบบมานั้นสามารถทำงานได้ตามที่ออกแบบหรือไม่ ซึ่งทำได้โดยการจำลองการทำงานระดับพฤติกรรม (Behavioral Simulation) ของวงจร โดยการเขียนภาษาเอชดีแอลเพื่อทดสอบวงจรหรือเรียกว่า "Testbench" ซึ่งมีการกำหนดรูปแบบของสัญญาณอินพุต และเมื่อจำลองการทำงานด้วยโปรแกรมซิมมูเลเตอร์ (Simulator) เช่น ModelSim ก็จะได้สัญญาณเอาต์พุตทำให้สามารถตรวจสอบว่าการทำงานของวงจรที่ถูกตรวจสอบ (Design Under Test: DUT) ทำงานถูกต้องหรือไม่ ซึ่งถ้าไม่ถูกต้องก็สามารถกลับไปแก้ไขใหม่ได้

- **การติดตั้ง**

จากผลที่ได้จากการสังเคราะห์ที่เป็นเน็ตลิสต์ในระดับเกตนั้นจะถูกแปลงให้เป็นกลุ่มของลอจิก จากนั้นจะทำการวางและเชื่อมเส้นทาง (Place & Route) ในเอฟพีจีเอโดยซอฟต์แวร์ของผู้ผลิตเอฟพีจีเอ เช่น Xilinx ISE เป็นต้น ซึ่งโปรแกรมจะรายงานเวลาการทำงานของวงจร (Static Timing) และสามารถตรวจสอบความถูกต้องโดยทำการจำลองการทำงานระดับฐานเวลา (Timing Simulation) ด้วยโปรแกรมซิมูเลเตอร์โดยใช้ Testbench เดิม ซึ่งการจำลองการทำงานระดับฐานเวลาถือว่าเป็นการจำลองการทำงานขั้นตอนสุดท้าย โดยวิธีนี้จะให้รายละเอียดที่แน่นอนเกี่ยวกับเรื่องของเวลา เช่น ความล่าช้าของสัญญาณไฟฟ้าตามเส้นทางต่างๆ ภายในวงจร คาบของสัญญาณนาฬิกา น้อยที่สุดที่สามารถใช้กับวงจรได้ เป็นต้น

การโปรแกรมลงเอฟพีจีเอ

เมื่อตรวจสอบด้วยการจำลองการทำงานระดับฐานเวลาแล้ว สามารถแปลงเน็ตลิสต์ในระดับเกตเป็นข้อมูลวงจรซึ่งอยู่ในรูปของบิตสตรีม (Bit-Stream) เพื่อนำบิตสตรีมนี้โปรแกรมลงเอฟพีจีเอต่อไป ขั้นตอนนี้ใช้ซอฟต์แวร์ของผู้ผลิตเอฟพีจีเอ ซึ่งการตรวจสอบความถูกต้องในการทำงานลำดับสุดท้ายคือการตรวจสอบในวงจรใช้งานจริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 ภาษาเวริล็อก (Verilog HDL)

ภาษาเวริล็อก (Verilog HDL) เป็นภาษาอธิบายฮาร์ดแวร์ (Hardware Description Language: HDL) ที่ใช้ในการออกแบบวงจรระบบดิจิตอล ภาษาเวริล็อกนั้นเป็นหนึ่งในสองภาษาอธิบายฮาร์ดแวร์ที่นิยมใช้กันมากในการออกแบบวงจรระบบดิจิตอลในปัจจุบัน ซึ่งได้แก่ ภาษาวีเอชดีแอล (Very High speed integrated circuit Hardware Description Language: VHDL) ซึ่งโครงสร้างของภาษาเวริล็อกนั้นจะมีความซับซ้อนน้อยกว่าภาษาวีเอชดีแอล

3.4.1 ประวัติความเป็นมา

ภาษาเวริล็อกถูกคิดค้น โดย บริษัท Gateway Design Automation ในปี 1981 ซึ่งเหตุการณ์ที่สำคัญและความเป็นมากล่าวเรียงตามลำดับได้ดังนี้ [5]

ปี 1983 บริษัท Gateway Design Automation ได้เปิดตัวภาษาอธิบายฮาร์ดแวร์ ซึ่งเรียกว่า "Verilog HDL" หรือเรียกง่ายๆ ว่า "Verilog" พร้อมทั้งส่วนการจำลองการทำงานาของภาษาเวริล็อก (Verilog Simulator)

ปี 1985 ภาษาเวริล็อกได้มีการปรับปรุงเพิ่มเพิ่มและได้ออกเวอร์ชันใหม่ของส่วนการจำลองการทำงาน ซึ่งเรียกว่า "Verilog XL"

ปี 1987 ภาษาล็อกเป็นที่นิยมใช้ ขนาดของวงจรที่ออกแบบนั้นเริ่มเกินความสามารถของชุดจำลองการทำงานของ Gateway Design System และบริษัทอื่นๆ

ปี 1989 บริษัท Cadence ซึ่งเป็นบริษัทผู้ผลิตซอฟต์แวร์ออกแบบวงจรรวมชั้นนำในปัจจุบันได้พัฒนาภาษาเวริล็อกต่อจากบริษัท Gateway Design System

ปี 1990 บริษัท Cadence ได้แยกภาษาเวริล็อกและส่วนการจำลองการทำงานออกจากกันโดยได้นำภาษาเวริล็อกเปิดเผยต่อสาธารณะ และในปีนี้ผู้ใช้ภาษาเวริล็อกและบริษัทต่างๆ ได้จัดตั้งสมาคม Open Verilog International (OVI) เป็นผู้กำหนดควบคุมรายละเอียด (Specification) ของภาษาริล็อก และได้จัดทำคู่มืออ้างอิงสำหรับภาษาเวริล็อกเวอร์ชัน 1.0 (Verilog 1.0 Reference Manual) ขึ้นมา

ปี 1993 สมาคม OVI ได้ปรับปรุงเพิ่มเติมความสามารถของภาษาเวริล็อก เช่น สามารถของอาร์เรย์ และได้ออกคู่มืออ้างอิงสำหรับภาษาเวริล็อกใหม่ เป็นเวอร์ชัน 2.0 (Verilog 2.0 Reference Manual)

ปี 1995 สมาคม OVI ได้ส่งภาษาเวริล็อกเวอร์ชัน 2.0 ให้สมาคม IEEE ซึ่งสมาคม IEEE ได้ปรับปรุงและได้กำหนดเป็นมาตรฐานใหม่ที่เรียกว่า มาตรฐาน IEEE 1394-1995

ปี 2001 มีปรับปรุงให้เพิ่มความสามารถของภาษาเวริล็อกมาตรฐาน IEEE 1394 ทั้งด้านการสังเคราะห์วงจรและการจำลองการทำงาน อีกทั้งยังปรับปรุงให้ง่ายต่อการใช้งาน ซึ่งสมาคม IEEE กำหนดเป็นมาตรฐานใหม่ที่เรียกว่า มาตรฐาน IEEE 1394-2001

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.2 โครงสร้างของภาษาเวริล็อก

ภาษาเวริล็อกนั้นมีไวยากรณ์คล้าย ภาษาซี เช่น จบประโยคด้วย ; หรือใช้ // เพื่อไม่ให้นำไปใช้ในการคอมไพล์ เป็นต้น โครงสร้างของภาษาเวริล็อกนั้นประกอบด้วยมอดูล (Module) ซึ่งแต่ละมอดูลนั้นสามารถรวมเข้าด้วยกันเป็นมอดูลขนาดใหญ่หรือเรียกว่า ท็อปมอดูล (Top Module) โดยแต่ละมอดูลย่อยนั้นสามารถทำงานพร้อมกันซึ่งโครงสร้างของภาษาเวริล็อกประกอบด้วยส่วนต่างๆ ดังแสดงในตัวอย่างการอธิบายพฤติกรรมของวงจรดีฟลิปฟลอป (D Flip-Flop) ได้ ดังนี้

```
Module dff(clk,rst,din,dout);              // <ชื่อมอดูล>(<ชื่อพอร์ต>)
Input din, clk, rst                        //<ชื่อสัญญาณอินพุต>
Output dout;                               //<ชื่อสัญญาณเอาต์พุต>
Always @(posedge clk or posedge rst)       //<รายละเอียดโปรแกรม>
      If (rst)
            Dout <= l'b0;
Else   dout<= din;
end module       //<จบมอดูล>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การออกแบบโครงงาน

บทนี้กล่าวถึงการออกแบบชุดทดสอบรหัสแก้ไขข้อผิดพลาดโดยภายในชุดทดสอบรหัสแก้ไขข้อผิดพลาด ประกอบด้วยสองส่วนด้วยกันคือ ส่วนชุดทดสอบ และส่วนภาคเข้ารหัสและถอดรหัสคอนโวลูชัน โดยในส่วนของชุดทดสอบนั้นประกอบไปด้วยส่วนต่างๆ เช่น ภาคกำเนิดข้อมูล ภาคมอดูเลเตอร์ ภาคกำเนิดสัญญาณรบกวนแบบเกาส์ ภาคดีมอดูเลเตอร์ ภาคหน่วงเวลา และภาคเปรียบเทียบข้อมูล โดยรูปที่ 4.1 แสดงส่วนประกอบของชุดทดสอบรหัสแก้ไขข้อผิดพลาด การออกแบบในแต่ละส่วนของชุดทดสอบรหัสแก้ไขข้อผิดพลาดใช้ภาษาเวริล็อก ซึ่งจะเรียกส่วนต่างๆว่า “มอดูล” เมื่อทำการออกแบบในแต่ละมอดูลเสร็จแล้วจึงนำมารวมกันเป็นชุดทดสอบรหัสแก้ไขข้อผิดพลาด

4.1 โครงสร้างชุดทดสอบรหัสแก้ไขข้อผิดพลาด

ภายในชุดทดสอบรหัสแก้ไขข้อผิดพลาดประกอบไปส่วนต่างๆ ทั้งภาคกำเนิดข้อมูล ภาคมอดูเลเตอร์ ภาคกำเนิดสัญญาณรบกวน ภาคดีมอดูเลเตอร์ ภาคหน่วงเวลา และสุดท้ายคือภาคเปรียบเทียบข้อมูลซึ่งมีรายละเอียดและหลักการออกแบบของแต่ละภาคดังต่อไปนี้

4.1.1 ภาคกำเนิดข้อมูล

ภาคกำเนิดข้อมูลเป็นภาคที่สร้างสัญญาณข้อมูลขนาด 1 บิต และมีลักษณะการกระจายตัวแบบยูนิฟอร์มโดยอาศัย Linear Feedback Shift Register (LFSR) ซึ่งสัญญาณข้อมูลที่สร้างได้จะถูกส่งเข้าไปที่ภาคมอดูเลตและภาคหน่วงเวลาของชุดทดสอบรหัสแก้ไขข้อผิดพลาด โดยหลักการสร้างสัญญาณข้อมูลใช้ LFSR ดังสมการต่อไปนี้

$$x_n = a_1 \bullet x_{n-1} \oplus a_2 \bullet x_{n-2} \oplus a_3 \bullet x_{n-3} \oplus \dots \oplus a_m \bullet x_{n-m} \qquad (4.1)$$

โดยที่ x_i เป็นข้อมูลลำดับที่ i ที่ถูกสร้างขึ้นมา

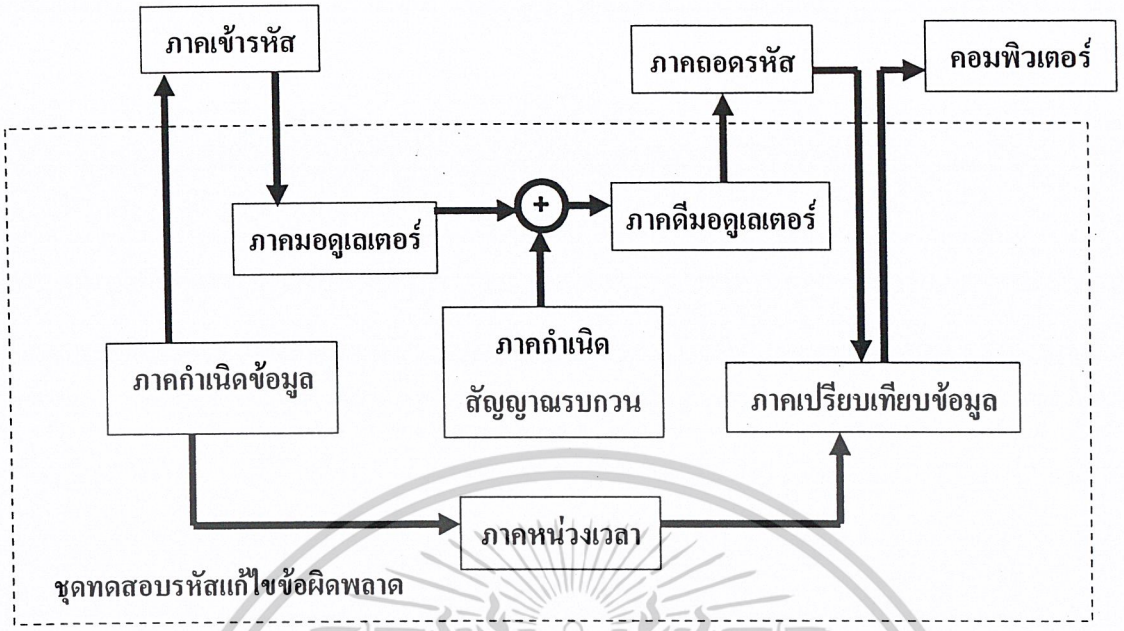
a_i คือ ค่าคงที่ 0 หรือ 1

$\bullet$ คือ And Operator

$\oplus$ คือ XOR Operator

m เป็นค่าที่เก็บไว้ในเรจิสเตอร์ของ LFSR

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.1 โครงสร้างของชุดทดสอบรหัสแก้ไขข้อผิดพลาด

โดยค่าของตัวเลขสุ่มที่เกิดขึ้นใหม่คือค่าของ x_n ซึ่งการเกิดขึ้นอยู่กับค่าที่เกิดก่อนหน้า รวมทั้งค่าที่อยู่ภายใน LFSR คือค่าของ m (x_{n-1}, x_{n-1}, ... , x_{n-m}) ดังนั้นจำนวนของเรจิสเตอร์ทั้งหมดที่ต้องใช้ในการสร้างข้อมูลใหม่จะขึ้นอยู่กับสมการโพลิโนเมียลดังสมการที่ (4.1) หลังจากค่าของตัวเลขสุ่มที่เกิดขึ้นใหม่ ค่าตัวเลขสุ่มที่เกิดก่อนจะถูกแทนที่ด้วยค่าตัวเลขสุ่มที่เกิดขึ้นใหม่ ซึ่งจะทำให้ต้องใช้ชิพเรจิสเตอร์ (Shift Register) เท่ากับ m-slot ตัว ในการเลื่อน ค่าตัวเลขสุ่มเก่าออกแล้วใส่ค่าตัวเลขสุ่มใหม่ลงไปแทนในทุกรอบของสัญญาณนาฬิกา นอกจากจะใช้ชิพเรจิสเตอร์แล้ว ยังต้องใช้ XOR เกตด้วย เพื่อทำการ Exclusive-OR ระหว่างค่าที่อยู่ภายใน LFSR เพื่อให้กำเนิดค่าตัวเลขสุ่มใหม่ โดยค่าเริ่มต้นของ m จะเป็นอะไรก็ได้ยกเว้นค่าที่เป็นศูนย์ทั้งหมดเนื่องจากจะทำให้ค่าของข้อมูลที่ถูกสร้างขึ้นเป็นศูนย์ตามไปด้วย

4.1.2 ภาคมอดูเลเตอร์

ในส่วนของภาคมอดูเลเตอร์ของชุดทดสอบรหัสแก้ไขข้อผิดพลาดนั้น อาศัยการมอดูเลต (Modulate) แบบ Binary Phase Shift Keying (BPSK) ซึ่งเป็นการเปลี่ยนบิต “0” ให้เป็น 1 และ บิต “1” ให้เป็น -1 โดยแสดงดังสมการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$Y = 1\text{-}2X \qquad (4.2)$$

โดยที่ Y คือบิตข้อมูลที่ผ่านการมอดูเลต
X คือบิตข้อมูลที่เข้ามามอดูเลต

4.1.3 ภาคกำเนิดสัญญาณรบกวนแบบเกาส์

ภาคกำเนิดสัญญาณรบกวนแบบเกาส์ในโครงงานนี้ใช้วิธีบ็อกมูลเลอร์ในการออกแบบซึ่งจะให้ค่าของสัญญาณรบกวนที่สร้างขึ้นมีค่าเฉลี่ยที่ศูนย์และค่าความแปรปรวนที่หนึ่งโดยวิธีของบ็อกมูลเลอร์แสดงดังสมการต่อไปนี้

$$f(u_1) = \sqrt{-\ln(u_1)} \qquad (4.1)$$

$$g1(u_2) = \sqrt{2}\cos(2\pi u_2) \qquad (4.2)$$

$$g2(u_2) = \sqrt{2}\sin(2\pi u_2) \qquad (4.3)$$

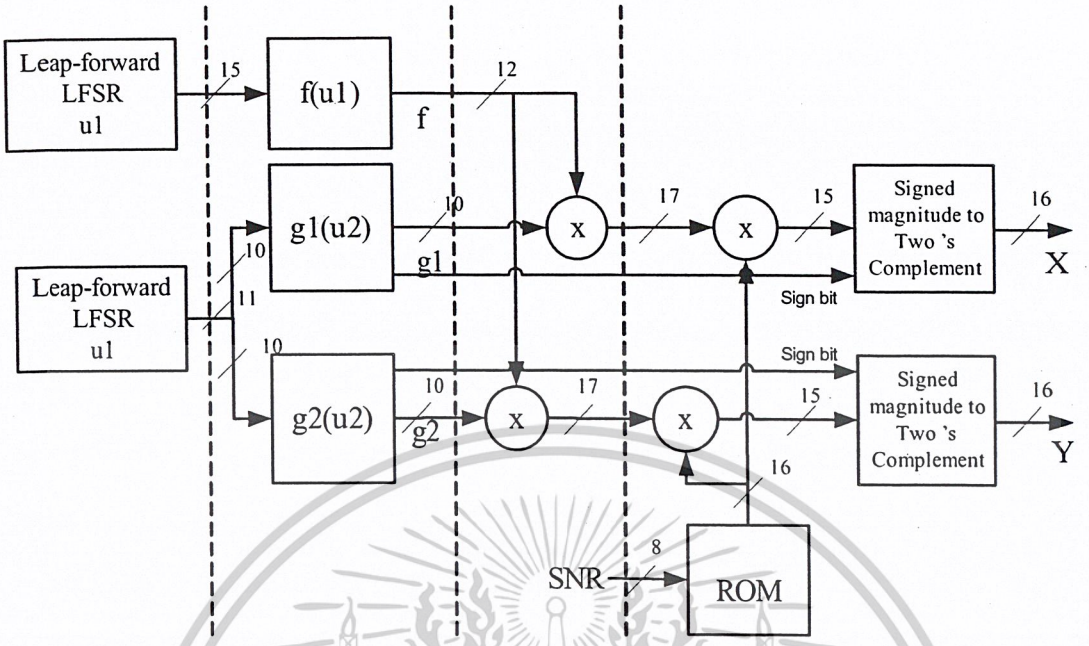
$$n = f(u_1)g(u_2) \qquad (4.4)$$

โดย u_1 และ u_2 คือตัวแปรสุ่มแบบยูนิฟอร์ม (Uniform Random Variable) ที่มีค่าการกระจายอยู่ในช่วง [0,1] ในการคำนวณโดยใช้ตัวดำเนินการ เช่น รากที่สองของล็อกการิทึม และฟังก์ชันตรีโกณมิตินั้นทำให้วงจรทางด้านฮาร์ดแวร์มีความซับซ้อนมาก และใช้พื้นที่ของเอฟพีจีเอในการสร้างจำนวนมาก ดังนั้นเพื่อที่จะลดความซับซ้อน และลดการใช้พื้นที่ในการสร้างดังนั้นจึงต้องทำการคำนวณค่าฟังก์ชัน $f(u_1)$ และ $g(u_2)$ ก่อนแล้วนำมาเก็บในหน่วยความจำ (Look-up Table) **[4]** ซึ่งวิธีนี้จะทำให้ใช้หน่วยความจำที่มากขึ้นแต่มีความเร็วในการทำงานที่สูง จากรูปที่ 4.2 เป็นโครงสร้างของชุดกำเนิดสัญญาณรบกวนโดยจะแบ่งการทำงานออกเป็นสี่ส่วนดังต่อไปนี้

4.1.3.1 มอดูลกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม

ส่วนนี้จะเป็นการสร้างตัวแปรสุ่มแบบยูนิฟอร์ม (Uniform Random) ขึ้นมาสองตัวซึ่งจะมีค่าการกระจายอยู่ระหว่าง [0,1] คือค่าของ u1 และ u2 ซึ่งการสุ่มนั้นจะอาศัย Leap-forward LFSR (Leap-forward Linear Feedback Shift Register) ซึ่ง Leap-forward LFSR มีโครงสร้างขนาดเล็กและให้ความเร็วเอาต์พุตสูงรวมทั้งให้ค่าความสัมพันธ์ของบิตที่สร้างน้อยกว่า LFSR (Linear Feedback Shift Register) **[14]**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



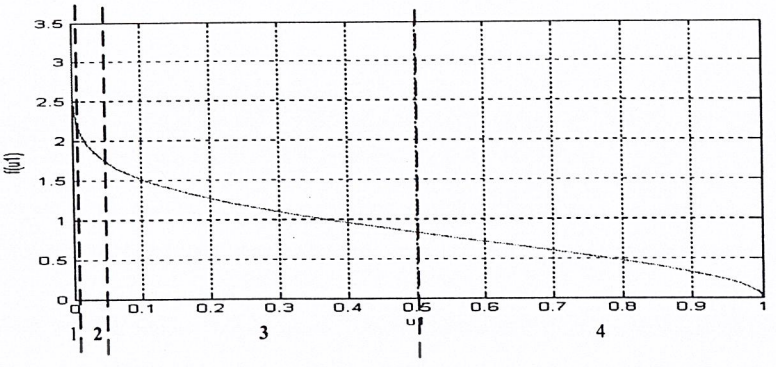
รูปที่ 4.2 โครงสร้างทางฮาร์ดแวร์ของชุดกำเนิดสัญญาณรบกวนแบบเกาส์ [4]

ในโครงงานนี้กำหนดค่าให้ u1 มีขนาด 15 บิต และ u2 มีขนาด 11 บิต ดังนั้นสัญญาณรบกวนที่สร้างอาศัยสมการที่ (4.4) ซึ่งเป็นผลคูณระหว่างค่าของฟังก์ชัน $f(u_1)$ และ $g(u_2)$ จากการที่ฟังก์ชัน $g(u_2)$ ที่ประกอบด้วยฟังก์ชัน sine และ cosine มีค่า โดยทั่วไปอยู่ระหว่าง [-1,1] ทำให้ค่าของฟังก์ชัน $f(u_1)$ เป็นค่าที่เป็นตัวกำหนดค่าการกระจายสูงสุดของตัวแปรสุ่มแบบเกาส์ ซึ่งในการสร้างนั้นจะใช้ Leap-forward LFSR 15 บิตในการสุ่มค่าของฟังก์ชัน $f(u_1)$ ทำให้ค่าตัวแปรสุ่มแบบเกาส์มีการกระจายสูงสุดเป็น 4.5σ และคาบของการสุ่มอยู่ที่ 2^{26} samples

4.1.3.2 มอดูลประมาณค่าฟังก์ชัน

ในส่วนนี้จะทำการประมาณค่าฟังก์ชัน $f(u_1), g1(u_2)$ และ $g2(u_2)$ ซึ่งได้มาจากสมการที่ (4.1) (4.2) และ (4.3) โดย u1 และ u2 คือตัวเลขสุ่มแบบยูนิฟอร์ม โดยภายในมอดูลประมาณค่าฟังก์ชันจะแบ่งออกเป็นสองมอดูลย่อยด้วยกันคือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.3 ค่าฟังก์ชัน *f(u1)* ที่ประมาณค่าโดยวิธีลิเนียร์แบบแบ่งออกสี่ส่วน **[4]**

มอดูลประมาณค่าฟังก์ชันลอกการิทึม

การประมาณค่าลอกการิทึมจะอาศัยการแบ่งค่าของฟังก์ชัน $f(u_1)$ แบบ Linear quantization สี่ส่วน **[4]** โดยการประมาณค่านั้นจะอาศัยการแทนค่าตัวแปรสุ่ม u_1 โดยทำการแบ่งค่า u_1 ออกเป็น 1024 ค่า ดังนั้นแต่ละค่าที่ทำการแทนค่าของ u_1 จะมีช่วงห่างกัน 1/1024 เมื่อได้ค่าของฟังก์ชัน $f(u_1)$ ก็จะนำค่าที่ได้เปลี่ยนเป็นเลขฐานสองซึ่งโครงงานนี้จะนำค่าที่ได้จะนำมาเก็บไว้ใน log_rom จากรูปที่ 4.3 แสดงค่าของฟังก์ชัน $f(u_1)$ โดยค่าสูงสุดของฟังก์ชัน $f(u_1)$ มีค่าประมาณ 3 ดังนั้นการเก็บค่าของฟังก์ชัน $f(u_1)$ ใน log_rom จึงใช้เพียงแค่ 2 บิตในการเก็บค่าจำนวนเต็มและ 10 บิต ในการเก็บค่าทศนิยมและเพื่อที่ให้สัญญาณรบกวนที่สร้างมีค่าการกระจายตัวที่ดีดังนั้นค่า u_1 จึงควรมีค่าเข้าใกล้ศูนย์

มอดูลประมาณค่าฟังก์ชัน $g(u_2)$

การประมาณค่าฟังก์ชัน $g(u_2)$ จะเก็บค่าที่ได้ลงใน sin_rom และ cos_rom เนื่องจากฟังก์ชัน $g(u_2)$ ประกอบไปด้วยฟังก์ชัน $g1(u_2)$ ซึ่งเป็นฟังก์ชันที่ประกอบด้วยฟังก์ชัน Cosine ดังสมการที่ (4.6) และฟังก์ชัน $g2(u_2)$ ที่ประกอบด้วยฟังก์ชัน Sine ดังสมการที่ (4.7) ในการแบ่งในช่วงของฟังก์ชัน $g(u_2)$ เพื่อนำมาเก็บไว้ในหน่วยความจำ sin_rom และ cos_rom จะทำการประมาณค่าเพียง 1 ควอเตอร์ (Quarter) คือควอเตอร์ที่ 1 เนื่องจากคุณสมบัติการสมมาตรของฟังก์ชันตรีโกณมิติ โดยเหตุผลที่ใช้เพียงครอเตอร์เดียวนั้น เพื่อเป็นการลดขนาดของหน่วยความจำที่ใช้ในการสร้างชุดกำเนิดสัญญาณรบกวน **[4]** และ ต้องการเก็บค่าของฟังก์ชัน $g(u_2)$ เป็นบวกทั้งหมดเพื่อง่ายต่อการออกแบบวงจรคูณ โดยเครื่องหมายของฟังก์ชันจะถูกเลือกอีกที่ที่ มอดูปรับเปลี่ยน Eb/No จากสมการที่ (4.6) และ (4.7) ค่าของ $g(u_2)$ จะมีค่าสูงสุดเท่ากับ 1 ทำให้การออกแบบใช้ 1 บิตในการเก็บค่าจำนวนเต็มและ 9 บิตในการเก็บค่าทศนิยม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.3.3 มอดูลชุดวงจรคูณค่าฟังก์ชัน

ตัวเลขจากการประมาณค่าฟังก์ชันในมอดูลที่สองเป็นแบบ Signed Magnitude Fractional และค่าฟังก์ชัน *f(u1)* เป็นค่าบวกเท่านั้นการออกแบบวงจรคูณในมอดูลที่สามจึงทำได้ง่าย การออกแบบจึงสามารถแยกบิตเครื่องหมายออกจากวงจรคูณ โดยใช้บล็อกวงจรคูณ 18x18 บิตในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 ในการหาค่าสัญญาณรบกวนที่ค่าความแปรปรวนเท่ากับหนึ่งและมีค่าเฉลี่ยเป็นศูนย์

4.1.3.4 มอดูลปรับเปลี่ยน Eb/No

การทำงานมอดูลนี้จะประกอบด้วยหน่วยความจำเก็บค่าความแปรปรวนที่ถูกเลือกโดย Eb/No ตามสมการที่ (4.5) และ (4.6) ซึ่งค่า Eb/No มีผลต่อการกระจายตัวของตัวแปรสุ่มแบบเกาส์ดังสมการ

$$\frac{E_s}{N_0} = \frac{1}{2\sigma^2} \tag{4.5}$$

$$\text{หรือ } \sigma = \sqrt{\frac{1}{2(Es/No)}} \tag{4.6}$$

ค่าพลังงานที่ใช้ส่งข้อมูลสัญลักษณ์ต่อความเข้มของสัญญาณรบกวน (Es/No) มีความสัมพันธ์กับค่าพลังงานที่ใช้ส่งข้อมูลบิตต่อความเข้มของสัญญาณรบกวน (Eb/*No*) ดังสมการ

$$\frac{E_s}{N_0} = \frac{nEb}{No} \tag{4.7}$$

โดยจะถูกนำมาคูณกับค่าสัญญาณรบกวนที่ค่าความแปรปรวนเท่ากับหนึ่งที่วงจรคูณ หลังจากนั้นจึงแปลงตัวเลขแบบ Signed Magnitude Fractional ให้เป็นแบบ 2's Complement แบบคิดเครื่องหมายขนาด 16 บิต ที่มีจำนวนเต็ม 4 บิตและทศนิยม 12 บิต เพื่อใช้ในการทดสอบรหัสแก้ไขความผิดพลาดข้อมูลต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.4 ภาคช่องสัญญาณ

ภาคช่องสัญญาณเป็นการรวมกันระหว่างข้อมูลที่ส่งกับสัญญาณรบกวน ซึ่งจะทำให้ข้อมูลที่ส่งผ่านช่องสัญญาณการสื่อสารเกิดการผิดพลาด

$$r(t) = s(t) + n(t) \tag{4.8}$$

จากสมการที่ 4.8 แสดงสัญญาณที่ผ่านช่องสัญญาณการสื่อสาร $r(t)$ ซึ่งเป็นการรวมกันระหว่างสัญญาณข้อมูล $s(t)$ และสัญญาณรบกวน $n(t)$

4.1.5 ภาคดีมอดูเลเตอร์

ภาคดีมอดีดูเลเตอร์เป็นการแปลงสัญญาณที่รับจากช่องสัญญาณการสื่อสารที่ผ่าน การมอดูเลตแบบ BPSK ให้กลับมาเป็นข้อมูลดังเดิมซึ่งข้อมูลที่ได้รับกลับมานั้นอาจเกิดการผิดพลาดขึ้นเนื่องมาจากปัจจัยหลายอย่างซึ่งสัญญาณรบกวนเป็นหนึ่งในสาเหตุที่ทำให้บิตของข้อมูลที่ส่งเกิดการผิดพลาดขึ้น

4.1.6 ภาคเปรียบเทียบข้อมูล

ภายในภาคเปรียบเทียบข้อมูลนี้จะประกอบด้วยวงจรในการเปรียบเทียบข้อมูลที่อาศัย XOR ในการเปรียบเทียบข้อมูลที่มาจากภาคกำเนิดข้อมูลและภาคดีมอดูเลเตอร์และสร้างเป็น สัญญาณอินพุตของวงจรนับจำนวนบิตที่ผิดพลาด

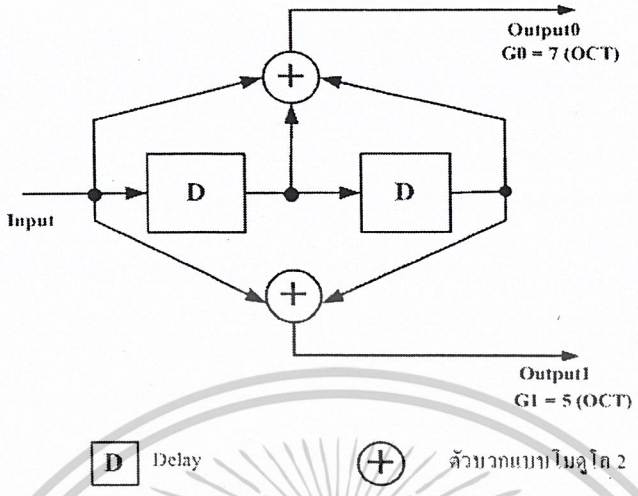
4.1.7 ภาคคอมพิวเตอร์อินเตอร์เฟส

ภาคคอมพิวเตอร์อินเตอร์เฟสเป็นภาคที่ติดต่อระหว่างเอฟพีจีเอ และคอมพิวเตอร์เพื่อใช้ในการแสดงผลการผิดพลาดของบิตข้อมูลระหว่างข้อมูลที่เข้ารหัสและข้อมูลที่ไม่ได้เข้ารหัสแก้ไขข้อผิดพลาด โดยติดต่อผ่านพอร์ตอนุกรม (RS-232) โดยใช้อัตราบอด (Baud Rate) ที่ 9600 บิตต่อวินาทีเป็นตัวกำหนดความเร็วในการส่งข้อมูลระหว่างเอฟพีจีเอและคอมพิวเตอร์

4.2 โครงสร้างของชุดเข้ารหัสและถอดรหัสคอนโวลูชัน (Convolution)

การส่งข้อมูลผ่านระบบสื่อสารสิ่งหนึ่งที่ทำให้เกิดการผิดพลาดของข้อมูลคือ สัญญาณรบกวน ในการป้องกันการผิดพลาดของข้อมูลที่ส่งผู้ส่งอาจจะใช้การเข้ารหัสแก้ไขข้อผิดพลาด ก่อนที่จะส่งข้อมูลผ่านระบบสื่อสารซึ่งรหัสคอนโวลูชันเป็นหนึ่งในรหัสแก้ไขข้อผิดพลาดล่วงหน้า สามารถช่วยลดจำนวนบิตข้อมูลที่ผิดลงได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.4 วงจรเข้ารหัสคอนโวลูชัน R =1/2 และ $K = 3$

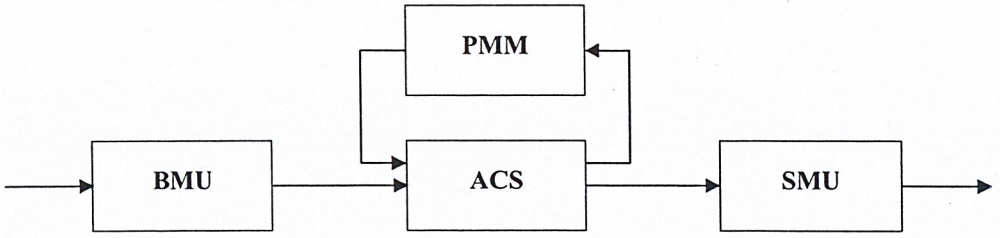
4.2.1 ภาคเข้ารหัสคอนโวลูชัน

การออกแบบวงจรเข้ารหัสคอนโวลูชันจะอาศัยเรจิสเตอร์ ในการเก็บค่าข้อมูลที่เข้ามาภายในวงจรก่อนแล้วนำมารวมกับค่าข้อมูลที่เข้ามาใหม่ภายใต้การมอดูโลด้วย 2 โดยกำหนดสถานะเริ่มต้นให้กับเรจิสเตอร์เป็น 0 โดยโครงงานนี้ใช้การเข้ารหัสคอนโวลูชันที่มีค่า Code Rate เท่ากับ 1/2 และมีค่า K เท่ากับ 3 ดังรูปที่ 4.4

4.2.2 ภาคถอดรหัสคอนโวลูชัน

การถอดรหัสคอนโวลูชันด้วยวิธี Viterbi ซึ่งลักษณะการทำงานเป็นแบบ Maximum Likelihood Decoding โดยผลลัพธ์ที่ได้จากการถอดรหัสจะเป็นเส้นทางเพียงเส้นทางเดียวที่มีความน่าจะเป็นสูงสุดจากเส้นทางทั้งหมดในไดอะแกรมทรีรีส (Trellis Diagram) ซึ่งมีลักษณะคล้ายกับข้อมูลที่ถูกส่งมากที่สุด รูปที่ 4.6 แสดงวงจรถอดรหัสคอนโวลูชันซึ่งแบ่งขั้นตอนออกเป็น 3 ขั้นตอนดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.5 ไดอะแกรมของชุดถอดรหัส Viterbi [1]

4.2.2.1 Branch Metric Generation

ขั้นตอนนี้คำนวณหาค่า Branch Metric จากข้อมูลอินพุตที่รับเข้ามา r กับค่าเอาต์พุตของการเข้ารหัส C การคำนวณหาค่า Branch Metric ต้องคำนวณทุกๆ สาขาหรือ Branch ตามสเตท (State) การทำงานโดย Branch เท่ากับ 2^K การคำนวณหาค่า Branch Metric แสดงดังสมการ 4.9

$$BM_{i,j,n} = (r_n - C_{i,j})^2 \tag{4.9}$$

โดย ค่า BM แทนค่า Branch Metric ระหว่างสเตท i ไปยังสเตท j ณ เวลา n

ค่า r แทนค่า ข้อมูลอินพุตที่รับเข้ามา ณ เวลา n

ค่า C แทนค่าเอาต์พุตของการเข้ารหัสระหว่างสเตท i ไปยังสเตท j ณ เวลา n

4.2.2.2 Survivor Path and Path Metric Update

ขั้นตอนนี้คำนวณหาค่า Survivor Path และ Path Metric จากจำนวน สเตทการทำงานทั้งหมด ค่า Path Metric ที่เลือกไว้เพื่อใช้ในการหาค่า Path Metric ครั้งต่อไป (Update) ส่วนค่า Survivor Path เป็นค่าที่ใช้ในการตัดสินใจหาค่าเอาต์พุต โดยการคำนวณหาค่า Survivor Path และ Path Metric นั้นค่าของ Branch Metric และ Path Metric จะถูกเข้าด้วยกัน ซึ่งผลการบวกนั้นมีสองค่าที่เข้ามาในแต่ละจุดเชื่อมต่อ (Trellis Node) ของแผนภาพ Trellis โดยค่า Path Metric เป็นค่าที่เลือกจากค่าผลบวกที่น้อยกว่า ส่วนค่า Survivor Path เป็นสเตทการทำงานที่น้อยกว่าจากการเลือก Path Metric ซึ่งแสดงดังสมการ4.10

$$PM_{j,n} = min(PM_{i,n-1} + BM_{i,j,n}, PM_{i+1,n-1} + BM_{i+1,j,n}) \tag{4.10}$$

โดย ค่า PM แทน Path metric ระหว่างสเตท i ไปยังสเตท j ณ เวลา n

ค่า BM แทนค่า Branch metric ระหว่างสเตท i ไปยังสเตท j ณ เวลา n

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.2.3 Optimum Paths Trace Back

ขั้นตอนนี้เป็นขั้นตอนการตัดสินใจหาค่าเอาต์พุต โดยใช้ค่า Survivor Path ในแต่ละ สเตทที่บันทึกไว้มาตัดสินใจเลือกเส้นทางของข้อมูล โดยการตัดสินใจหาเส้นทางของข้อมูลจะเริ่มจาก Survivor Path ในอดีต (Trace Back) โดยในการเริ่มต้นจากเวลาผ่านไป L (Latency) โดยในทางปฏิบัติการ L ต้องมีค่ามากกว่าห้าเท่าของค่า K (Constrain Length) จึงทำให้ข้อมูลที่ถูกถอดรหัสมีความถูกต้องสูง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
ผลการทดลอง

บทนี้กล่าวถึงผลการออกแบบและทดสอบการสร้างสัญญาณรบกวนแบบเกาส์ทางซอฟต์แวร์ โดยใช้โปรแกรม MATLAB เพื่อทดสอบคุณสมบัติข้อมูลทางสถิติของวิธีการสร้างสัญญาณรบกวนแบบเกาส์ และผลการออกแบบชุดทดสอบรหัสแก้ไขข้อผิดพลาดซึ่งประกอบไปด้วยส่วนต่างๆ เช่น ภาคกำเนิดข้อมูล ภาคมอดูเลเตอร์ ภาคเข้ารหัส ภาคถอดรหัส ภาคกำเนิดสัญญาณรบกวน ภาคดีมอดูเลเตอร์ และสุดท้ายคือภาคเปรียบเทียบข้อมูล การออกแบบชุดทดสอบรหัสแก้ไขข้อผิดพลาดนี้ใช้ภาษาเวริล็อก (Verilog HDL) ในการอธิบายพฤติกรรมการทำงานวงจรชุดทดสอบและทำการติดตั้งลงบนเอฟพีจีเอ Xilinx Vertex II รุ่น xc2V1000-6 โดยใช้โปรแกรม Xilinx ISE ในการออกแบบ การสังเคราะห์วงจร (Synthesis) และการติดตั้ง และโปรแกรม ModelSim ในการจำลองการทำงานและตรวจสอบผลการทำงาน

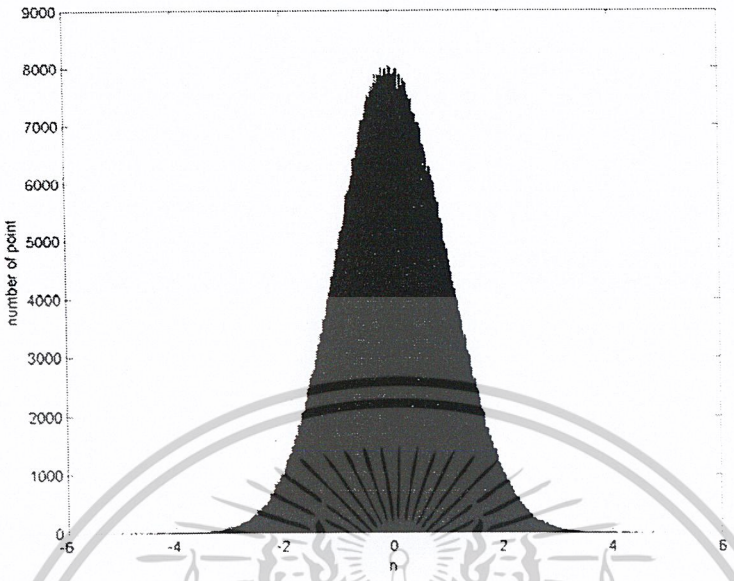
5.1 ผลการสร้างสัญญาณรบกวนแบบเกาส์ทางซอฟต์แวร์

การสร้างสัญญาณรบกวนที่มีการกระจายตัวแบบเกาส์นั้น โดยใช้โปรแกรม MATLAB ในการออกแบบใช้วิธีต่างๆเช่น วิธีบ็อกมูลเลอร์ วิธีเซ็นทรัลลิมิต วิธีโพลาร์ และวิธีผสม ซึ่งได้ทำการสุ่มตัวเลขสุ่มแบบยูนิฟอร์ม (Uniform Random Variable) ซึ่งจะมีค่าการกระจายอยู่ระหว่าง [0,1] มาทั้งหมด 1,000,000 เพื่อใช้ในการทดสอบการกระจายตัวโดยตัวเลขสุ่มที่มีการกระจายตัวแบบยูนิฟอร์มนี้มีค่าเฉลี่ยอยู่ที่ 0.5 และส่วนเบี่ยงเบนมาตรฐานที่ 0.28 โดยผลการสร้างได้แสดงดังนี้

5.1.1 วิธีของเซ็นทรัลลิมิต

จากสมการ (2.18) เป็นวิธีในการสร้างสัญญาณรบกวนที่มีการกระจายแบบเกาส์มีค่าเฉลี่ยเป็นศูนย์ และค่าความแปรปรวนเป็นหนึ่ง ตัวอย่างโปรแกรม MATLAB แสดงขั้นตอนการสร้างสัญญาณรบกวนแบบเกาส์ และในรูปที่ 5.1 แสดงฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์ที่สร้างขึ้นโดยแกน “x” แสดงถึงค่าสัญญาณรบกวนที่สร้างขึ้น (n) และแกน “y” แสดงจำนวนครั้งในการสร้าง (number of point)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

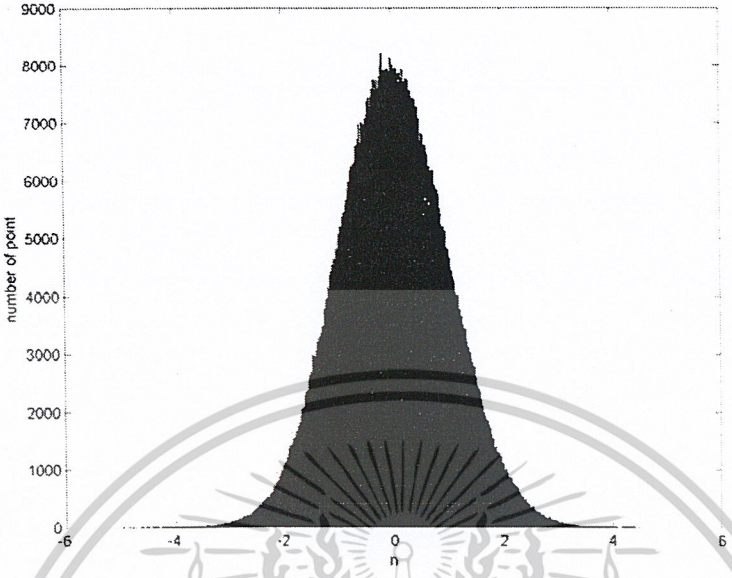


รูปที่ 5.1 ฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์โดยวิธีเซ็นทรัลลิมิต

ตัวอย่างโปรแกรม MATLAB

```
for j =1:1000000//Loop for(j) เพื่อสร้างตัวแปรสุ่มแบบเกาส์ 1000000 ค่า
m = 10000;      //กำหนดให้ m มีค่าเท่ากับ 10000โดยค่า m ควรมีจำนวนเข้าใกล้
                infinity
x = rand(1,m);  //สุ่มตัวแปรที่มีการกระจายแบบ uniform random ทั้งหมด m ค่า
sum = 0;
 for i = 1:m                          //Loop for(i)
    sum = sum + x(i);     //หาผลรวมของตัวแปร x
 end                                  //End Loop for(i)
 xn(j) = sqrt(12/m).*(sum-0.5*m);     //สร้างสัญญาณรบกวน
end                                   //End Loop for(j)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.2 ฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์โดยวิธีบ็อกมูลเลอร์

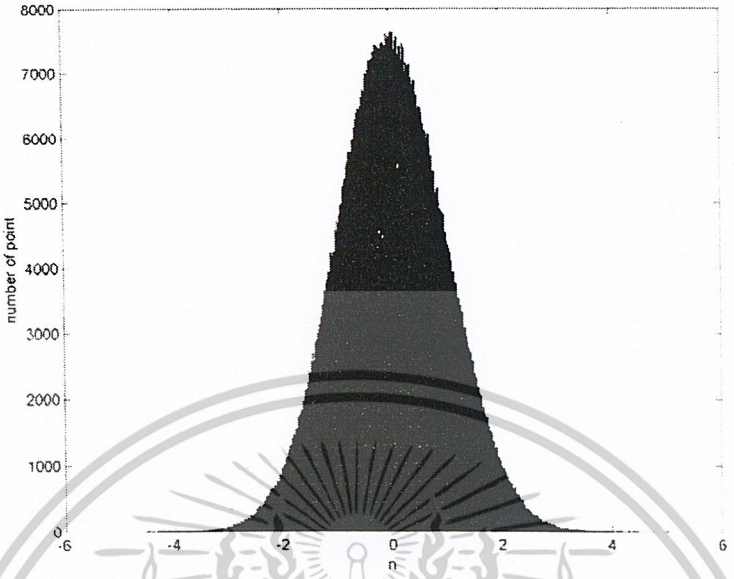
5.1.2 วิธีบ็อกมูลเลอร์

จากสมการที่ (2.20) ถึง สมการที่ (2.24) เป็นสมการในการสร้างสัญญาณรบกวนที่มีค่าเฉลี่ยเป็นศูนย์ และค่าความแปรปรวนเป็นหนึ่ง ดังตัวอย่างโปรแกรม MATLAB แสดงขั้นตอนการสร้างสัญญาณรบกวนแบบเกาส์ และในรูปที่ 5.2 แสดงฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์ที่สร้างขึ้นโดยแกน “x” แสดงถึงค่าสัญญาณรบกวนที่สร้างขึ้น (n) และแกน “y” แสดงจำนวนครั้งในการสร้าง (number of point)

ตัวอย่างโปรแกรม MATLAB

```
x1 = rand(1,1000000);    //สุ่มตัวแปรแบบ uniform random ทั้งหมด 1000000 ค่า
x2 = rand(1,1000000);
f1 = sqrt(-1*log(x1));    //หาค่าฟังก์ชัน f(x)ตามสมการที่ (2.20)
f2 = (sqrt(2))*cos(2*pi*x2);        //หาค่าฟังก์ชัน g(x)ตามสมการที่ (2.21)
n  = f1.*f2;        //สร้างสัญญาณรบกวนดังสมการที่(2.23)
save boxmuller n;        //เก็บค่าที่ได้ในไฟล์ชื่อ boxmuller
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.3 ฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์โดยวิธีโพลาร์

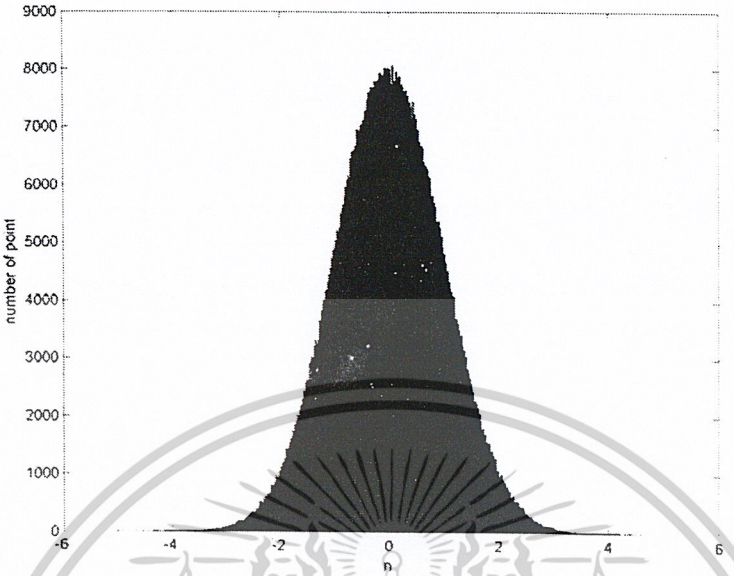
5.1.3 วิธีโพลาร์ (Polar Method)

วิธีโพลาร์เป็นวิธีที่พัฒนามาจากวิธีบ็อกมูลเลอร์ โดยสุ่มตัวแปรสองตัวที่มีการกระจายแบบยูนิ-ฟอร์ม โดยหลักการสร้างจะอาศัยสมการที่ (2.25) ถึงสมการที่ (2.30) ดังแสดงโดยตัวอย่างโปรแกรม MATLAB และในรูปที่ 5.3 แสดงฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์ที่สร้างขึ้นโดยแกน "x" แสดงถึงค่าสัญญาณรบกวนที่สร้างขึ้น (n) และแกน "y" แสดงจำนวนครั้งในการสร้าง (number of point)

ตัวอย่างโปรแกรม MATLAB

```
a1 = rand(1,1000000);    //สุ่มตัวแปรแบบ uniform random ทั้งหมด 100 ค่า
a2 = rand(1,100000);
v1 = 2.*a1 - 1;          //สร้างค่าตัวแปรสุ่มให้มีทั้งค่าบวกและค่าลบ
v2 = 2.*a2 - 1;
for i = 1:m                              //Loop for
    r(i) = v1(i).^2 + v2(i) .^2;         //หาค่า r ตามสมการที่ (28)
    if (r(i) <= 1)                       //ตรวจสอบเงื่อนไขตามสมการ(29)
        fac(i) = sqrt((-2 .* log(r(i)))/r(i));
        an(j) = v2(i) .* fac(i);         //สร้างสัญญาณรบกวนตามสมการ(27)
    end                                  // End if
end                                      //End for
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.4 ฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์โดยวิธีผสม

5.1.4 วิธีผสม

วิธีผสมเกิดจากการรวมตัวกันของสองวิธีคือ วิธีเซ็นทรัลลิมิตและวิธีบ็อกมูลเลอร์หรือวิธีโพลาร์ หลักการสร้างสัญญาณรบกวนแบบเกาส์ตามสมการที่ (2.31) โดยตัวอย่างโปรแกรม MATLAB แสดงวิธีการสร้างสัญญาณรบกวน และในรูปที่ 5.4 แสดงฮิสโตรแกรมของตัวแปรสุ่มแบบเกาส์ที่สร้างขึ้นโดยแกน "x" แสดงถึงค่าสัญญาณรบกวนที่สร้างขึ้น (n) และแกน "y" แสดงจำนวนครั้งในการสร้าง (number of point)

ตัวอย่างโปรแกรม MATLAB ในการสร้างตัวแปรสุ่มแบบเกาส์โดยวิธีผสมซึ่งอาศัยวิธีบ็อกมูลเลอร์

```
x1 = rand(1,1000000);    //สุ่มตัวแปรแบบ uniform random ทั้งหมด 1000000 ค่า
x2 = rand(1,1000000);
f1 = sqrt(-1*log(x1));           //แทนค่าตามสมการที่(19)
f2 = (sqrt(2))*cos(2*pi*x2);     //แทนค่าตามสมการที่(20)
n  = f1.*f2;
a = [1 1 1 1];
an = conv(n,a)./2;               //หาผลรวมตามสมการที่(30)โดยใช้การ
                                 convolution
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สรุปผลการสร้างสัญญาณรบกวนแบบเกาส์

การสร้างสัญญาณรบกวนแบบเกาส์โดยใช้โปรแกรม MATLAB ซึ่งดยในวิธีเซ็นทรัลลิมิตจะให้ผลการสร้างที่ช้าเนื่องจากต้องมีการสุ่มตัวเลขที่มี การกระจายแบบยูนิฟอร์มจำนวนมากเพื่อสร้างสัญญาณรบกวนแบบเกาส์หนึ่งตัวแปรสุ่ม การสร้างสัญญาณรบกวนแบบเกาส์โดยวิธีบ็อกมูลเลอร์เป็นวิธีที่ง่ายแต่เนื่องจากวิธีบ็อกมูลเลอร์มีฟังก์ชันล็อกกาลิทึมและ ฟังก์ชันตรีโกณมิติทำให้ใช้พื้นที่ในการสร้างทางฮาร์ดแวร์จำนวนมาก วิธีโพลาร์เป็นวิธีที่แปลงมาจากวิธีบ็อกมูลเลอร์โดยทำการกำจัดฟังก์ชันตรีโกรณมิติออกไป และวิธีผสมเป็นวิธีที่ให้ผลการสร้างสัญญาณรบกวนที่ดีที่สุดแต่ในการทำงานของวิธีผสมนี้จะช้าเนื่องจากประกอบด้วยวิธีเซ็นทรัลลิมิต และวิธีบ็อกมูลเลอร์ ซึ่งวิธีเซ็นทรัลลิมิตจะทำให้ผลของวิธีผสมนี้ช้า

5.2 ผลการจำลองการทำงานของแต่ละมอดูลชุดทดสอบรหัสแก้ไขข้อผิดพลาด

มอดูลชุดทดสอบรหัสแก้ไขข้อผิดพลาดประกอบไปด้วยส่วนต่างๆ เช่น ภาคกำเนิดข้อมูล ภาคมอดูเลเตอร์ ภาคกำเนิดสัญญาณรบกวน ภาคดีมอดูเลเตอร์ และภาคเปรียบเทียบข้อมูล การออกแบบในแต่ละส่วนจะใช้ภาษาเวริล็อกโดยภาษาเวริล็อกจะเรียกส่วนต่างๆว่า "มอดูล" ซึ่งผลการออกแบบและจำลองการทำงานของแต่ละมอดูลของชุดทดสอบรหัสแก้ไขข้อผิดพลาดมีดังนี้

5.2.1 มอดูลกำเนิดข้อมูล

มอดูลกำเนิดข้อมูลเป็นมอดูลที่กำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 1 บิต ซึ่งเป็นส่วนหนึ่งของชุดทดสอบรหัสแก้ไขข้อพลาด เพื่อสร้างบิตข้อมูลเพื่อเป็นอินพุตให้แก่ภาคเข้ารหัสคอนโวลูชันและภาคเปรียบเทียบข้อมูลเพื่อหาค่าอัตราการผิดพลาดของบิต (Bit Error Rate: BER) ผลการออกแบบและจำลองการทำงานของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มมีดังนี้

ผลการสังเคราะห์มอดูลกำเนิดข้อมูล

ผลการสังเคราะห์มอดูลกำเนิดข้อมูลโดยใช้โปรแกรม Xilinx ISE หรือเขียนโปรแกรมทดสอบ (Testbench) โดยใช้ภาษาเวริล็อก จะได้กลุ่มของเน็ตลิสต์ ซึ่งประกอบด้วยชิพเรจิสเตอร์ และ Multiplexer ดังแสดงในตารางที่ 5.1 และ ในตารางที่ 5.2 แสดงจำนวนอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลกำเนิดข้อมูลโดยในตารางจะประกอบด้วย จำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop) จำนวนตัวเปิดตารางสี่อินพุต (4-Input LUT) ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ซึ่งหมายถึงการป้อนสัญญาณอินพุตต้องป้อนเข้าก่อนเวลานี้นับจากเวลาเกิดการเปลี่ยนแปลงของสัญญาณนาฬิกา

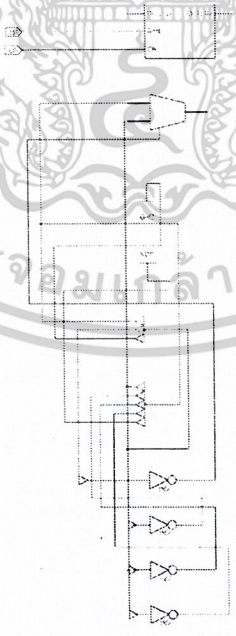
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.1 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลกำเนิดข้อมูล

อุปกรณ์	จำนวน
Register	1
13-bit register	1
Multiplexer	1
2-to-1 multiplexer	1

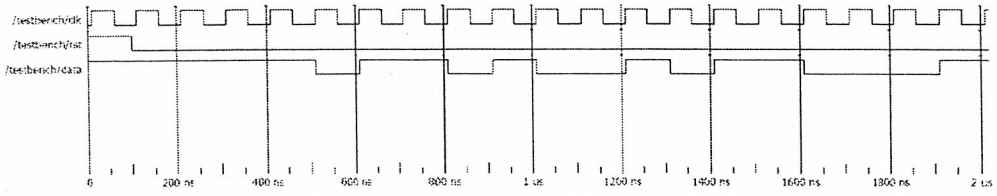
ตารางที่ 5.2 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลกำเนิดข้อมูล

อุปกรณ์	Slices	Slices Flip Flop	4 input LUT
จำนวน	7	13	3
Maximum Frequency			514.403 MHz
Minimum input arrival time before clock			2.038 ns
Maximum output required time after clock			6.423 ns



รูปที่ 5.5 ผังวงจรระดับ RTL ของมอดูลกำเนิดข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.6 ผลจำลองการทำงานของมอดูลกำเนิดข้อมูล

ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) หมายถึงสัญญาณเอาต์พุตจะเกิดหลังจากเวลานี้นับจากนับจากเวลาที่เกิดการเปลี่ยนแปลงของสัญญาณนาฬิกา และในส่วนของรูปที่ 5.5 แสดงผังวงจรระดับ RTL ของมอดูลกำเนิดข้อมูล

ผลการจำลองการทำงานของมอดูลกำเนิดข้อมูล

ผลการจำลองการทำงานของมอดูลกำเนิดข้อมูลโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณอินพุตทดสอบของมอดูลกำเนิดข้อมูลประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) และสัญญาณข้อมูล (data) ที่สร้างขึ้นขนาด 1 บิตเพื่อเป็นบิตข้อมูลที่ถูกใช้ในชุดทดสอบรหัสแก้ไขข้อผิดพลาดโดยส่งต่อไปยังภาคเข้ารหัสคอนโวลูชันและเปรียบเทียบข้อมูลเพื่อนับจำนวนบิตที่ผิดพลาด โดยก่อนการจำลองการทำงานต้องเซ็ตสัญญาณรีเซ็ตเป็น 1 ก่อนเพื่อให้วงจรทำงานได้อย่างถูกต้อง ดังรูปที่ 5.6

5.2.2 มอดูลมอดูเลเตอร์

มอดูลมอดูเลเตอร์เป็นมอดูลที่ทำการมอดูเลตข้อมูลที่รับมาจากภาคเข้ารหัสคอนโวลูชันก่อนส่งต่อไปยังช่องสัญญาณแบบ AWGN (Additive white Gaussian Noise)โดยชุดทดสอบรหัสแก้ไขข้อผิดพลาดนี้ใช้การมอดูเลตแบบ Binary Phase Shift Keying (BPSK) โดยผลการออกแบบและจำลองการทำงานของมอดูลมอดูเลเตอร์มีดังนี้

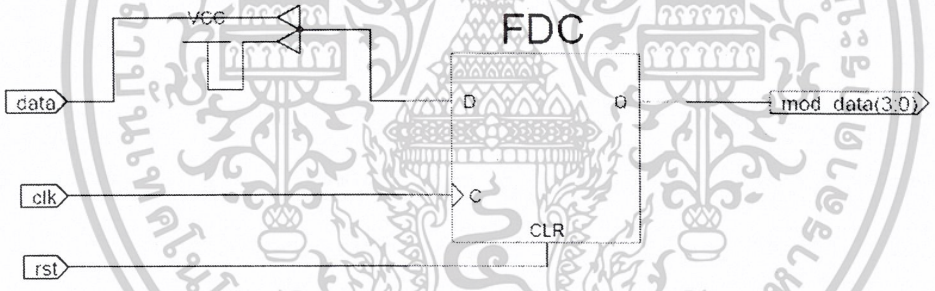
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.3 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลมอดูเลเตอร์

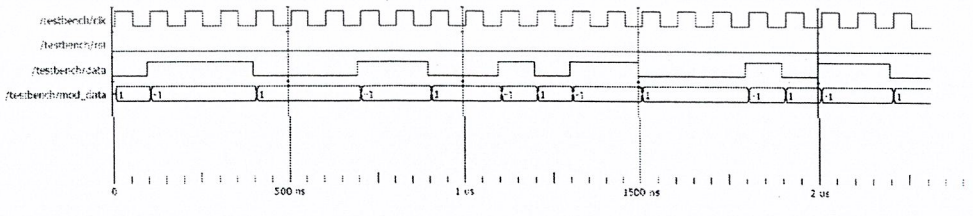
อุปกรณ์	จำนวน
Register	1
8-bit register	1

ตารางที่ 5.4 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลมอดูเลเตอร์

อุปกรณ์	Slices	Slices Flip Flop
จำนวน	5	8
Maximum Frequency		543.478 MHz
Minimum input arrival time before clock		1.84 ns
Maximum output required time after clock		5.880 ns



รูปที่ 5.7 ผังวงจรระดับ RTL ของมอดูลมอดูเลเตอร์



รูปที่ 5.8 ผลจำลองการทำงานของมอดูลมอดูเลเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลการสังเคราะห์มอดูลมอดูเลเตอร์

ผลการสังเคราะห์มอดูลมอดูเลเตอร์โดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งภายในมอดูลมอดูเลเตอร์ประกอบไปด้วยชิพเรจิสเตอร์ขนาด 8 บิต ดังแสดงในตารางที่ 5.3 และ ในตารางที่ 5.4 แสดงถึงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลมอดูเลเตอร์โดยในตารางจะประกอบด้วยจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop) ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) และในส่วนของรูปที่ 5.7 แสดงผังวงจรระดับ RTL ของมอดูลมอดูเลเตอร์

ผลการจำลองการทำงานของมอดูลมอดูเลเตอร์

ผลการจำลองการทำงานของมอดูลมอดูเลเตอร์ทดสอบโดยใช้โปรแกรม Xilinx ISE ซึ่งสัญญาณทดสอบของมอดูลมอดูเลเตอร์ประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) สัญญาณข้อมูล (data) ที่รับมาจากภาคเข้ารหัสคอนโวลูชันและสัญญาณข้อมูลที่ถูกมอดูเลต (mod_data) ซึ่งมอดูลดีมอดูเลเตอร์นี้ใช้การมอดูเลตแบบ BPSK โดยจะแทนข้อมูล 1 ด้วย -1 หรือ 1111 แบบ 2'Complement ขนาด 4 บิต และแทนข้อมูล 0 ด้วย 1 หรือ 0001 แบบ 2'Complement ขนาด 4 บิต เพื่อที่จะสามารถนำมารวมกับสัญญาณรบกวนที่สร้างขึ้นในมอดูลชุดกำเนิดสัญญาณรบกวนโดยผลการทำงานของมอดูลมอดูเลเตอร์แสดงดังรูปที่ 5.8

5.2.3 มอดูลชุดกำเนิดสัญญาณรบกวน

การออกแบบชุดกำเนิดสัญญาณรบกวนจะถูกแบ่งออกทั้งหมดสี่ส่วนด้วนกัน ประกอบไปด้วยมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม มอดูลประมาณค่าฟังก์ชัน มอดูลคูณค่าฟังก์ชัน และมอดูลปรับเปลี่ยน Eb/No ซึ่งผลการออกแบบและจำลองการทำงานของชุดกำเนิดสัญญาณรบกวนในแต่ละมอดูลย่อยมีดังนี้

5.2.3.1 มอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม

มอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มเป็น ส่วนประกอบแรกของชุดกำเนิดสัญญาณรบกวนแบบเกาส์โดยในชุดกำเนิดสัญญาณรบกวนนี้มีส่วนของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม นี้สองส่วนคือมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม ขนาด 11 บิตและ 15 บิต เพื่อเป็นสัญญาณในการอ้างตำแหน่งข้อมูลที่เก็บในหน่วยความจำ cos_rom sin_rom และ log_rom ผลการออกแบบและจำลองการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำงานของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มมีดังนี้

ผลการสังเคราะห์มอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม

ผลการสังเคราะห์มอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มด้วยโปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ในระดับเกต ซึ่งประกอบด้วย ชิพเรจิสเตอร์ และ XOR เกตดังตารางที่ 5.5 ในตารางที่ 5.6 แสดงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มและในตารางที่ 5.6 ยังแสดงจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop) จำนวนตัวเปิดตารางสี่อินพุต (4-Input LUT) ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) โดยรูปที่ 5.9(ก) แสดงผังวงจรระดับ RTL ของมอดูล ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 11 บิต และรูปที่ 5.9(ข) แสดงผังวงจรระดับ RTL ของมอดูล ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 15 บิต

ผลการจำลองการจำลองการทำงานของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม

การทำงานของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม ซึ่งทดสอบโดยใช้โปรแกรม Xilinx ISE ซึ่งสัญญาณทดสอบของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มนี้ประกอบด้วยสัญญาณนาฬิกา (clock) สัญญาณรีเซ็ต (reset) และผลที่ได้จะเป็นสัญญาณข้อมูล (data) ที่มีการกระจายแบบยูนิฟอร์มขนาด 15 บิต ใช้อ้างตำแหน่งหน่วยความจำ log_rom และสัญญาณข้อมูลขนาด 11 บิต ใช้อ้างตำแหน่งหน่วยความจำ cos_rom และ sin_rom โดยผลการจำลองการทำงานของมอดูล ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มนี้แสดงดังรูป 5.10

5.2.3.2 มอดูลประมาณค่าฟังก์ชัน

มอดูลประมาณค่าฟังก์ชันเป็นมอดูลที่ใช้เก็บค่าที่ได้จากการประมาณค่าตามสมการที่ (4.5) แบบลิเนียร์ (Linear) โดยแบ่งเป็นสี่ส่วนเก็บไว้ในหน่วยความจำ log_rom และเก็บค่าที่ได้จากการประมาณค่าฟังก์ชันตรีโกณมิติ (Sine และ Cosine) ตามสมการที่ (4.6) และ (4.7) ไว้ในหน่วยความจำ sin_rom และ cos_rom ตามลำดับโดยใช้ตัวเลขสุ่มแบบยูนิฟอร์มที่สร้างมาจากมอดูลกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มในการอ้างตำแหน่งของหน่วยความจำซึ่งผลการออกแบบและจำลองการทำงานของมอดูลประมาณค่าฟังก์ชันมีดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

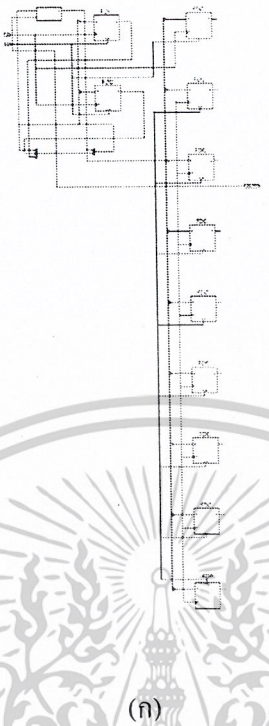
ตารางที่ 5.5 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม

อุปกรณ์	**จำนวน**
ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 11 บิต	
Register	11
1-bit register	11
XOR	11
1-bit xor	11
ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 15 บิต	
Register	15
1-bit register	15
XOR	15
1-bit xor	15

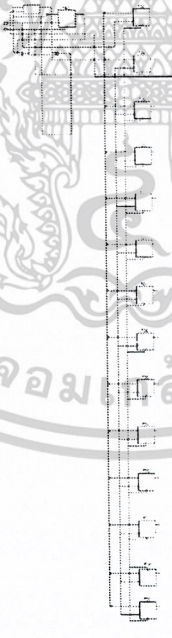
ตารางที่ 5.6 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม

อุปกรณ์	Slices	Slices Flip Flop	4 input LUT	Block RAMs
จำนวน(LFSR11 บิต)	6	11	11	-
จำนวน(LFSR15 บิต)	9	15	15	-
LFSR			11 บิต	15 บิต
Maximum Frequency			524.384 MHz	524.384 MHz
Minimum input arrival time before clock			1.962 ns	2.123 ns
Maximum output required time after clock			6.386 ns	6.386 ns

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



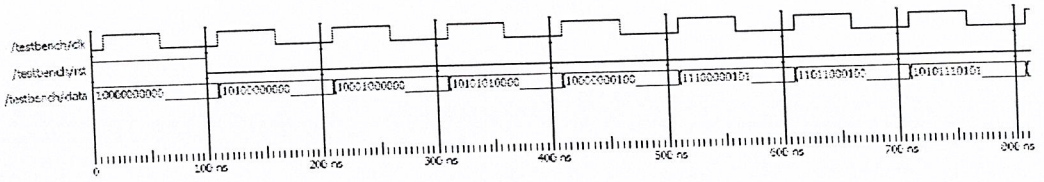
(ก)



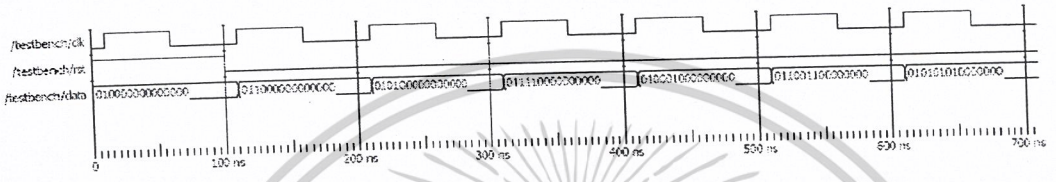
(ข)

รูปที่ 5.9 ผังวงจรระดับ RTL ของมอดูล ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม
(ก) ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 11 บิต (ข)ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 15 บิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(ก)



(ข)

รูปที่ 5.10 ผลการจำลองการทำงานของมอดูล ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม
(ก) ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 11 บิต (ข)ชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 15 บิต

มอดูลประมาณค่าฟังก์ชันลอกกาลิทึม

มอดูลประมาณค่าฟังก์ชันลอกกาลิทึมเป็นการประมาณค่าตามสมการ (4.5) ซึ่งประกอบไปด้วยฟังก์ชันล็อกกาลิทึมซึ่งเป็นฟังก์ชันพื้นที่จำนวนมากในการสร้างทางฮาร์ดแวร์ดังนั้นจึงต้อง ประมาณค่าเก็บไว้ในหน่วยความจำเพื่อง่ายต่อการออกแบบและความเร็วในการทำงานเพื่อที่จะได้ค่า $f(u_1)$ ทุกรอบของสัญญาณนาฬิกา โดยผลการออกแบบและผลการจำลองการทำงานของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึมมีดังต่อไปนี้

ผลการสังเคราะห์มอดูลประมาณค่าฟังก์ชันลอกกาลิทึม

ผลการสังเคราะห์มอดูลประมาณค่าฟังก์ชันลอกกาลิทึมโดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งประกอบด้วย เรจิสเตอร์และหน่วยความจำขนาด 1024×12 ดังแสดงในตารางที่ 5.7

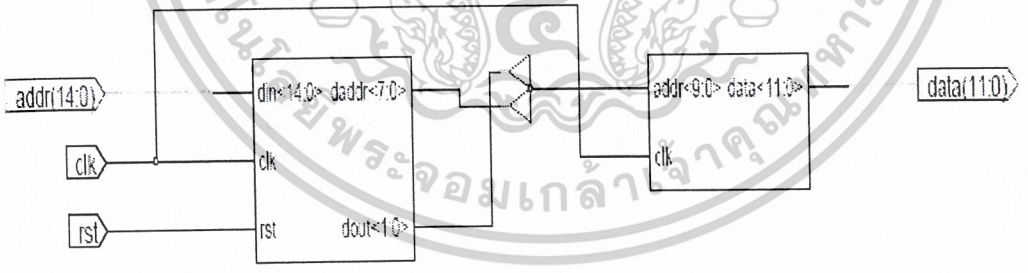
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.7 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม

อุปกรณ์	จำนวน
Rom	1
1024×12 -bit Rom	1
Register	11
1-bit register	10
12-bit register	1

ตารางที่ 5.8 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม

อุปกรณ์	Slices	Slices Flip Flop	4 input LUT	BRAM
จำนวน	15	10	26	1
Maximum Frequency				956.938 MHz
Minimum input arrival time before clock				4.78 ns
Maximum output required time after clock				7.485 ns



รูปที่ 5.11 ผังวงจรระดับ RTL ของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม

จากตารางที่ 5.8 ซึ่งแสดงถึงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม โดยในตารางจะประกอบด้วยจำนวน Slices จำนวน ฟลิปฟลอป (Slice Flip Flop) จำนวนตัวเปิดตารางสี่อินพุต (4-Input LUT) หน่วยความจำ (BRAM) ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) และในส่วนของรูปที่ 5.11 แสดงผังวงจรระดับ RTL ของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม

ผลการจำลองการทำงานของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม

ผลการจำลองการทำงานของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึมโดยใช้โปรแกรม Xilinx ISE ซึ่งสัญญาณอินพุตทดสอบของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึมประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) สัญญาณอ้างอิงตำแหน่ง (addr) ซึ่งเป็นค่าที่ได้มาจากมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 15 บิต ซึ่งจะใช้เป็นตัวอ้างตำแหน่งข้อมูลที่เก็บในหน่วยความจำและค่าเอาต์พุตที่ได้เป็นข้อมูล (data) ขนาด 12 บิต ซึ่งจะเห็นได้ว่าค่าการประมาณของฟังก์ชัน $f(u_1)$ ที่เก็บไว้ในหน่วยความจำจะออกมาทุกรอบของสัญญาณนาฬิกา โดยผลการจำลองการทำงานของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึมจะแสดงในรูปของข้อมูลไบนารีดังรูปที่ 5.12

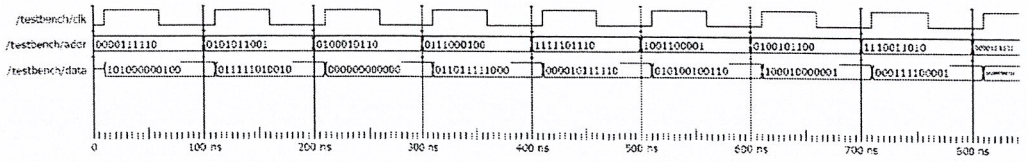
มอดูลประมาณค่าฟังก์ชัน $g(u_2)$

มอดูลประมาณค่าฟังก์ชัน $g(u_2)$ โดยเป็นการประมาณค่า ตามสมการที่ (4.6) ซึ่งเป็นการประมาณค่าฟังก์ชัน $g1(u_2)$ และ สมการที่ (4.7) ที่เป็นการประมาณค่า $g2(u_2)$ แล้วนำค่าที่ได้จากการประมาณมาเก็บไว้ในหน่วยความจำเพื่อที่ต้องการความเร็วในการทำงานเพราะจะให้ค่าการประมาณค่าฟังก์ชัน $g(u_2)$ ทุกรอบของสัญญาณนาฬิกา โดยผลการออกแบบและผลการจำลองการทำงานของมอดูลประมาณค่าฟังก์ชัน $g(u_2)$ มีดังต่อไปนี้

ผลการสังเคราะห์มอดูลประมาณค่าฟังก์ชัน $g(u_2)$

ผลการสังเคราะห์มอดูลประมาณค่าฟังก์ชัน $g(u_2)$ โดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งประกอบด้วยชิพเรจิสเตอร์และหน่วยความจำขนาด 1024×10 บิตดังแสดงในตารางที่ 5.9 ตารางที่ 5.10 ซึ่งแสดงถึงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลประมาณค่าฟังก์ชัน $g(u_2)$ โดยในตารางประกอบด้วยจำนวน Slices

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.12 ผลจำลองการทำงานของมอดูลประมาณค่าฟังก์ชันลอกกาลิทึม

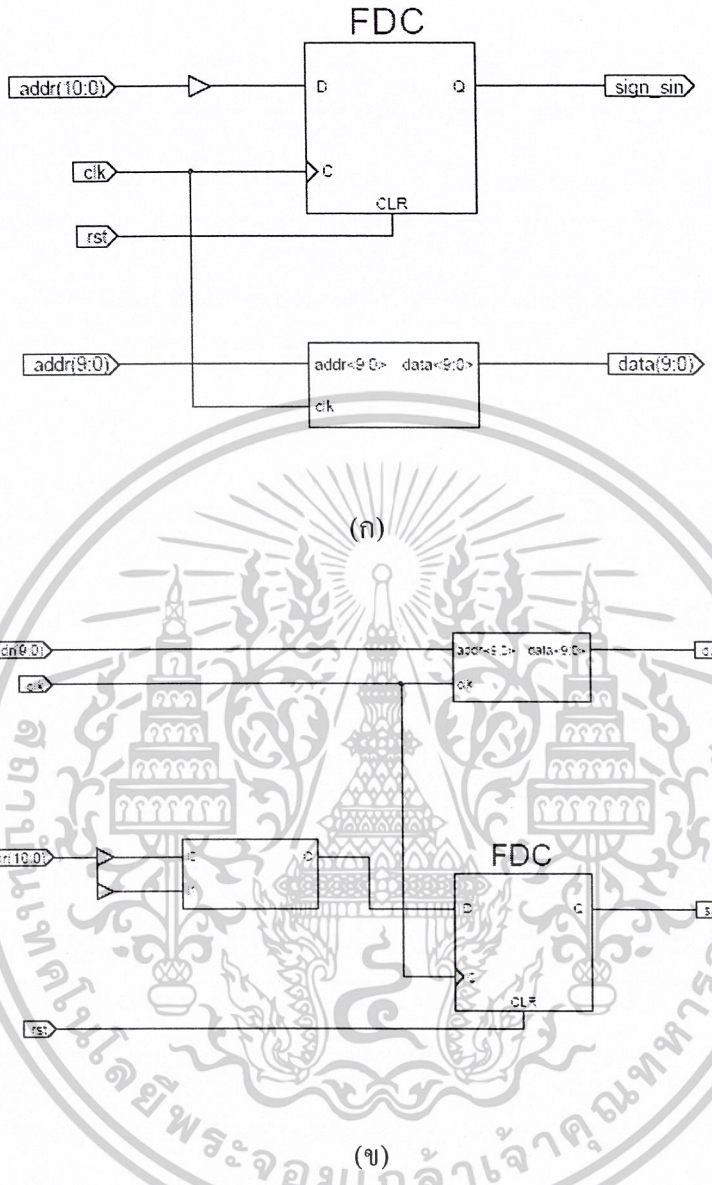
ตารางที่ 5.9 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลประมาณค่าฟังก์ชัน $g(u_2)$

อุปกรณ์	จำนวน
หน่วยความจำฟังก์ชัน g1(u_2)	
ROM	1
1024×10-bit ROM	1
Register	2
10-bit register	1
1-bit register	1
หน่วยความจำฟังก์ชัน g2(u_2)	
ROM	1
1024×10-bit ROM	1
Register	2
10-bit register	1
1-bit register	1

ตารางที่ 5.10 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลประมาณค่าฟังก์ชัน $g(u_2)$

อุปกรณ์	Slices	Slices Flip Flop	4 input LUT	BRAM
จำนวน(ฟังก์ชัน g1(u_2))	1	1	-	1
จำนวน(ฟังก์ชัน g2(u_2))	1	1	-	1
ฟังก์ชัน			g1(u_2)	g2(u_2)
Maximum Frequency			719.424 MHz	471.031MHz
Minimum input arrival time before clock			1.390 ns	2.123 ns
Maximum output required time after clock			1.898 ns	7.485 ns

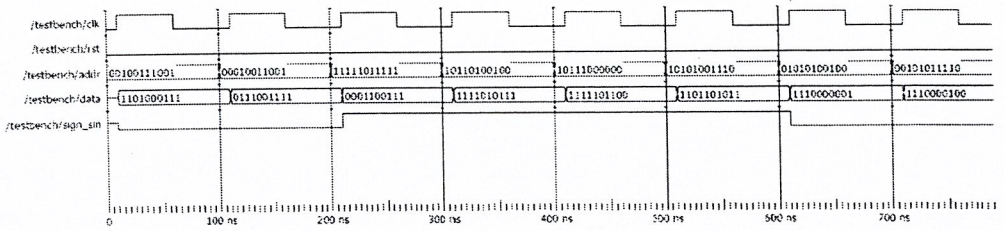
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



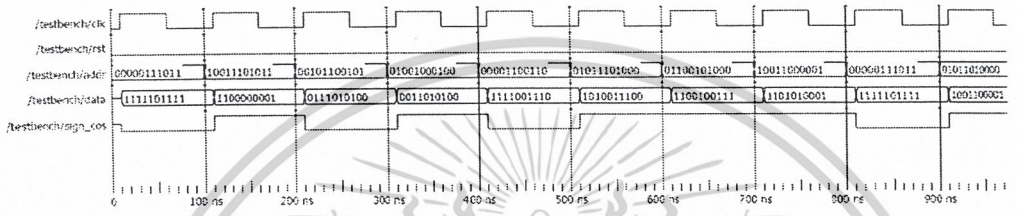
รูปที่ 5.13 ผังวงจรระดับ RTL ของมอดูลประมาณค่าฟังก์ชันตรีโกณมิติ
ก) ฟังก์ชัน Sine (ข) ฟังก์ชันCosine

จำนวนฟลิปฟลอป (Slice Flip Flop) จำนวนหน่วยความจำ (BRAM) ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) และในส่วนของรูปที่ 5.13 แสดงผังวงจรระดับ RTL ของมอดูลประมาณค่าฟังก์ชัน $g(u_2)$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(ก)



(ข)

รูปที่ 5.14 ผลจำลองการทำงานของมอดูลประมาณค่าฟังก์ชันตรีโกณมิติ
ก) ฟังก์ชัน Sine (ข) ฟังก์ชันCosine

ผลการจำลองการทำงานมอดูลประมาณค่าฟังก์ชัน $g(u_2)$

ผลการจำลองการทำงานของมอดูลประมาณค่าฟังก์ชัน $g(u_2)$ โดยใช้โปรแกรม Xilinx ISE โดยซึ่งสัญญาณอินพุตทดสอบของมอดูลประมาณค่าฟังก์ชันตรีโกณมิติประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) สัญญาณอ้างอิงตำแหน่ง (addr) ซึ่งเป็นค่าที่ได้มาจากมอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์มขนาด 11 บิต โดย 10 บิตจะใช้อ้างตำแหน่งข้อมูลที่เก็บในหน่วยความจำและอีก 1 บิต จะใช้เป็นตัวกำหนดเครื่องหมายของสัญญาณรบกวนที่สร้างขึ้นและค่าข้อมูล (data) ที่ออกมาจากหน่วยความจำมีขนาด 10 บิต โดยจะเห็นได้ว่าค่าการประมาณของฟังก์ชัน $g(u_2)$ ออกมาทุกรอบของสัญญาณนาฬิกา ซึ่งผลการจำลองการทำงานของมอดูลประมาณค่าฟังก์ชัน $g(u_2)$ แสดงดังรูปที่ 5.14

5.2.3.3 มอดูลชุดวงจรคูณค่าฟังก์ชัน

มอดูลชุดวงจรคูณค่าฟังก์ชันเป็นวงจรคูณโดยอาศัยค่าอินพุต จากมอดูลประมาณค่าฟังก์ชันโดยการออกแบบวงจรคูณในมอดูลนี้จะใช้บล็อกวงจรคูณ 18x18 บิตในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000ซึ่งค่าที่ได้จะเป็นค่าของตัวแปรสุ่มแบบเกาส์มีค่าฉลี่ยเท่ากับศูนย์ และค่าความแปรปรวนเท่ากับหนึ่งโดยผลการออกแบบและผลการจำลองการทำงานของ มอดูลชุดวงจรคูณค่าฟังก์ชันมีดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.11 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลชุดวงจรคูณค่าฟังก์ชัน

อุปกรณ์	จำนวน
Register	3
10-bit register	1
12-bit register	1
17-bit register	1

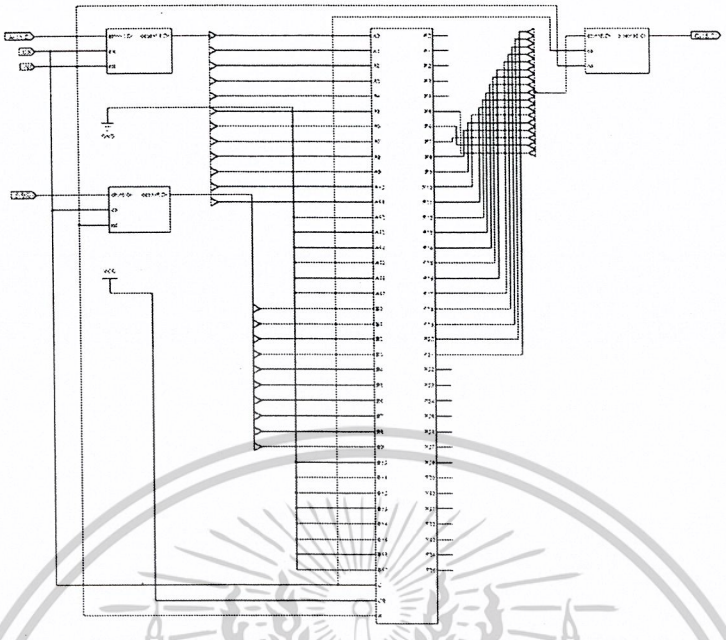
ตารางที่ 5.12 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลชุดวงจรคูณค่าฟังก์ชัน

อุปกรณ์	Slices	Slices Flip Flop	MULT18×18
จำนวน	22	39	1
Maximum Frequency			388.95MHz
Minimum input arrival time before clock			2.59ns
Maximum output required time after clock			5.880 ns

ผลการสังเคราะห์มอดูลชุดวงจรคูณค่าฟังก์ชัน

ผลการสังเคราะห์มอดูลชุดวงจรคูณค่าฟังก์ชัน โดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งประกอบไปด้วยเรจิสเตอร์ดังแสดงในตารางที่ 5.11 และ ตารางที่ 5.12 แสดงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลชุดวงจรคูณค่าฟังก์ชัน โดยในตารางจะประกอบด้วยจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop) บล็อกวงจรคูณขนาด 18×18 ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) และรูปที่ 5.15 แสดงผังวงจรระดับ RTL ของมอดูลชุดวงจรคูณค่าฟังก์ชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.15 ผังวงจรระดับ RTL ของมอดูลชุดวงจรคูณค่าฟังก์ชัน

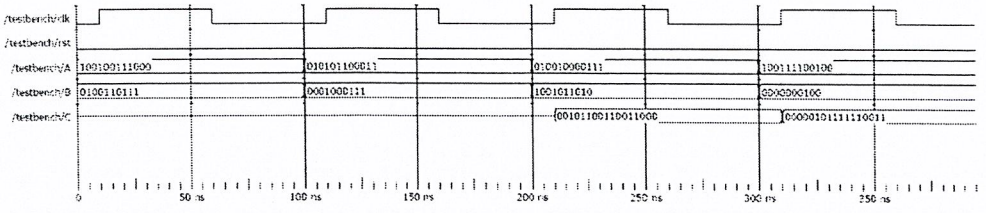
ผลการจำลองการทำงานของมอดูลชุดวงจรคูณค่าฟังก์ชัน

ผลการจำลองการทำงานของมอดูลชุดวงจรคูณค่าฟังก์ชันทดสอบโดยใช้โปรแกรม Xilinx ISE มีสัญญาณอินพุตทดสอบประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) สัญญาณข้อมูลจากมอดูลประมาณค่าฟังก์ชันตรีโกณมิติขนาด 10 บิตและสัญญาณข้อมูลมอดูลประมาณค่าฟังก์ชันลอกกาลิทึมขนาด 12 บิต และผลที่ได้สัญญาณเอาต์พุตขนาด 17 บิต ซึ่งการทำงานของมอดูลชุดวงจรคูณค่าฟังก์ชันนี้จะทำงานล่าช้าไปประมาณ 2 รอบของสัญญาณนาฬิกาดังแสดงในรูปที่ 5.16

5.2.3.4 มอดูลปรับเปลี่ยน Eb/No

มอดูลปรับเปลี่ยน Eb/No เป็นมอดูลปรับเปลี่ยนค่าอัตราส่วนการใช้กำลังงานในการส่งต่อสัญญาณรบกวน Eb/No เพื่อนำเอาต์พุตมาใช้ปรับเปลี่ยนค่าการกระจายตัวของตัวแปรสุ่มแบบเกาส์ ตามสมการที่ (4.8) และ (4.9) โดยมอดูลนี้จะประกอบด้วยหน่วยความจำในการเก็บค่าความแปรปรวนและหลังจากนั้นจะถูกนำมาคูณกับค่าของสัญญาณรบกวนที่สร้างขึ้นที่เป็นผลมาจาก มอดูลคูณค่าฟังก์ชัน ค่าที่ได้จะถูกเปลี่ยนเป็น 2's Complement แบบคิดเครื่องหมายขนาด 16 บิต ที่มีบิตจำนวนเต็ม 4 บิต และบิตทศนิยมขนาด 12 บิต โดยมอดูลปรับเปลี่ยนค่า Eb/No ประกอบด้วยมอดูลย่อยซึ่งมีรายละเอียดดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.16 ผลจำลองการทำงานของของมอดูลชุดวงจรคูณค่าฟังก์ชัน

มอดูลประมาณค่า Signal to Noise Ratio (SNR)

มอดูลประมาณค่า SNR ประกอบด้วยหน่วยความจำที่ใช้เก็บค่า Eb/No เพื่อใช้คูณกับสัญญาณรบกวนที่สร้างขึ้นจากมอดูลชุดวงจรคูณค่าฟังก์ชันซึ่งเป็นสัญญาณรบกวนที่มีค่าเฉลี่ยที่ศูนย์และความแปรปรวนเป็นหนึ่งเพื่อเป็นการปรับค่าการกระจายตัวของสัญญาณรบกวนที่สร้างขึ้น โดยผลการออกแบบและผลการจำลองการทำงานของมอดูลประมาณค่า SNR มีดังต่อไปนี้

ผลการสังเคราะห์มอดูลประมาณค่า SNR

ผลการสังเคราะห์มอดูลประมาณค่า SNR โดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ซึ่งประกอบด้วยหน่วยความจำขนาด 256×16 บิต และเรจิสเตอร์ ดังแสดงในตารางที่ 5.13 และ ในตารางที่ 5.14 แสดงถึงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลประมาณค่าSNRโดยในตารางจะประกอบด้วยหน่วยความจำ (BRAM) ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After clock) และในส่วนของรูปที่ 5.17 แสดงผังวงจรระดับ RTL ของมอดูลประมาณค่าSNR

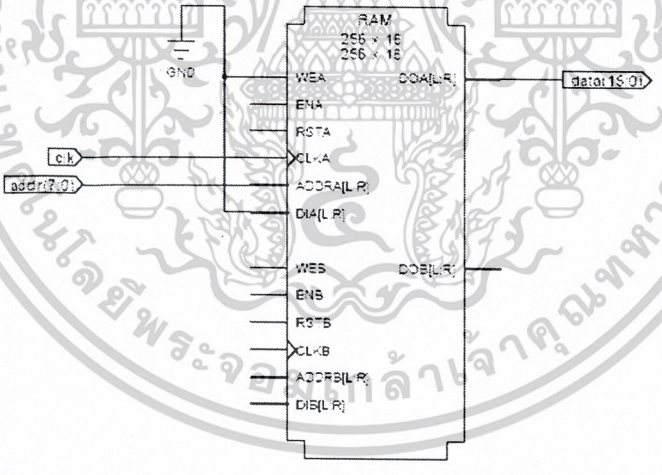
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.13 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลประมาณค่า SNR

อุปกรณ์	จำนวน
ROM	1
256×16 -bit ROM	1
Register	1
16-bit register	1

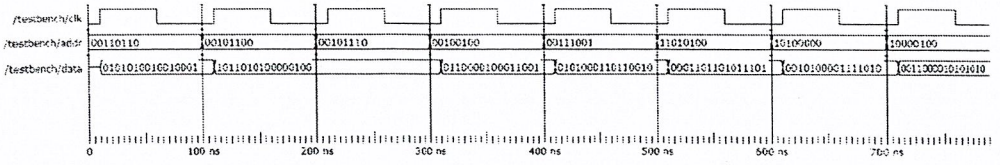
ตารางที่ 5.14 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลประมาณค่า SNR

อุปกรณ์	BRAM
จำนวน	1
Maximum Frequency	800.640 MHz
Minimum input arrival time before clock	1.249ns
Maximum output required time after clock	7.485 ns



รูปที่ 5.17 ผังวงจรระดับ RTL ของมอดูลประมาณค่า SNR

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.18 ผลจำลองการทำงานของของมอดูลประมาณค่า SNR

ผลการจำลองการทำงานของมอดูลประมาณค่า SNR

ผลการจำลองการทำงานของมอดูลประมาณค่า SNR โดยทดสอบโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณอินพุตทดสอบของมอดูลมอดูลประมาณค่า SNR ประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณอ้างอิงตำแหน่ง (addr) มีขนาด 8 บิตโดยผลที่ได้จะได้ข้อมูล (data) ขนาด 16 บิต โดยจะเห็นได้ว่าค่าของ Eb/No ที่มาจากการปรับเปลี่ยนค่า SNR จะออกมาในทุกรอบของสัญญาณนาฬิกาซึ่งสอดคล้องกับค่าการประมาณฟังก์ชัน $f(u_1)$ และ $g(u_2)$ ทำให้สามารถสร้างสัญญาณรบกวนที่มีค่าต่างๆได้ในทุกรอบของสัญญาณนาฬิกา ซึ่งผลการจำลองการทำงานของมอดูลประมาณค่า SNR แสดงดังรูปที่ 5.18

มอดูลวงจรคูณค่า SNR

มอดูลชุดวงจรคูณค่า SNR เป็นวงจรคูณโดยอาศัยตัวเลขจากการประมาณค่าในมอดูลประมาณค่าSNR และ ค่าที่ได้มอดูลวงจรคูณค่าฟังก์ชันโดยการออกแบบวงจรคูณในมอดูลนี้จะใช้บล็อกวงจรคูณ 18x18 บิตในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 มอดูลชุดวงจรคูณค่า SNR ประกอบไปด้วยชิพ-เรจิสเตอร์โดยผลการออกแบบและผลการจำลองการทำงานของมอดูลวงจรคูณค่า SNR มีดังต่อไปนี้

ผลการสังเคราะห์มอดูลชุดวงจรคูณค่าSNR

ผลการสังเคราะห์มอดูลชุดวงจรคูณค่า SNR โดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ต-ลิสต์ ประกอบไปด้วยเรจิสเตอร์ดังแสดงในตารางที่ 5.15 และ ตารางที่ 5.16 แสดงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลชุดวงจรคูณค่า SNR โดยในตารางจะประกอบด้วยจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop) วงจรคูณขนาด 18×18 ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญ์าณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock)

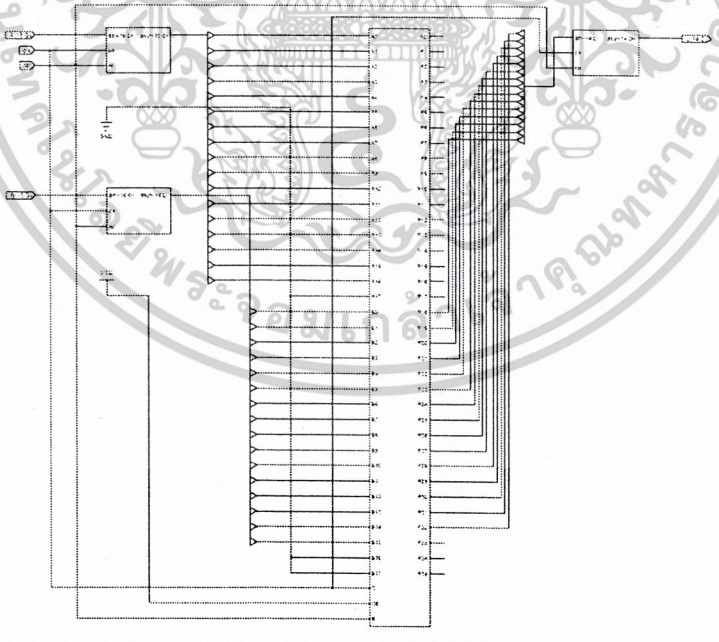
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.15 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลชุดวงจรคูณค่า SNR

อุปกรณ์	จำนวน
Register	3
17-bit register	1
16-bit register	1
15-bit register	1

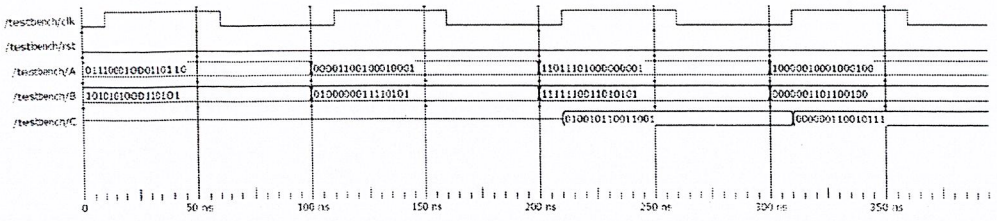
ตารางที่ 5.16 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลชุดวงจรคูณค่า SNR

อุปกรณ์	Slices	Slices Flip Flop	MULT18×18
จำนวน	28	48	1
Maximum Frequency			388.954 MHz
Minimum input arrival time before clock			2.607 ns
Maximum output required time after clock			5.880 ns



รูปที่ 5.19 ผังวงจรระดับ RTL ของมอดูลชุดวงจรคูณค่า SNR

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.20 ผลจำลองการทำงานของของมอดูลชุดวงจรคูณค่า SNR

รูปที่ 5.19 แสดงผังวงจรระดับ RTL ของมอดูลชุดวงจรคูณค่า SNR

ผลการจำลองการทำงานของมอดูลชุดวงจรคูณค่า SNR

ผลการจำลองการทำงานของมอดูลชุดวงจรคูณค่า SNR ทดสอบโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณอินพุตทดสอบของมอดูลมอดูลชุดวงจรคูณค่า SNR ประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) สัญญาณข้อมูลจากมอดูลวงจรคูณค่าฟังก์ชันขนาด 17 บิตซึ่งเป็นสัญญาณรบกวนที่มีค่าเฉลี่ยที่ศูนย์และค่าความแปรปรวนเป็นหนึ่งและสัญญาณข้อมูลจากมอดูลประมาณค่า SNR ขนาด 16 บิต ผลการคูณจะมีการล่าช้าไปสองรอบของสัญญาณนาฬิกาและจะมีขนาด 15 บิตซึ่งเป็นสัญญาณรบกวนที่มีค่าการกระจายตามค่าของ SNR โดยเก็บอยู่ในรูปแบบ Signed Magnitude Fractional ดังแสดงในรูปที่ 5.20

มอดูลปรับเปลี่ยนค่า

มอดูลปรับเปลี่ยนค่าเป็นมอดูลในการเปลี่ยนค่าสัญญาณรบกวนที่สร้างขึ้นจาก มอดูลชุดวงจรคูณค่า SNR จาก Signed Magnitude Fractional ให้เป็นแบบ 2's Complement โดยผลการออกแบบและผลการจำลองการทำงานของมอดูลปรับเปลี่ยนค่ามีดังต่อไปนี้

ผลการสังเคราะห์มอดูลปรับเปลี่ยนค่า

ผลการสังเคราะห์มอดูลปรับเปลี่ยนค่าโดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งจะประกอบไปด้วยเรจิสเตอร์ขนาดต่างๆดังแสดงในตารางที่ 5.17

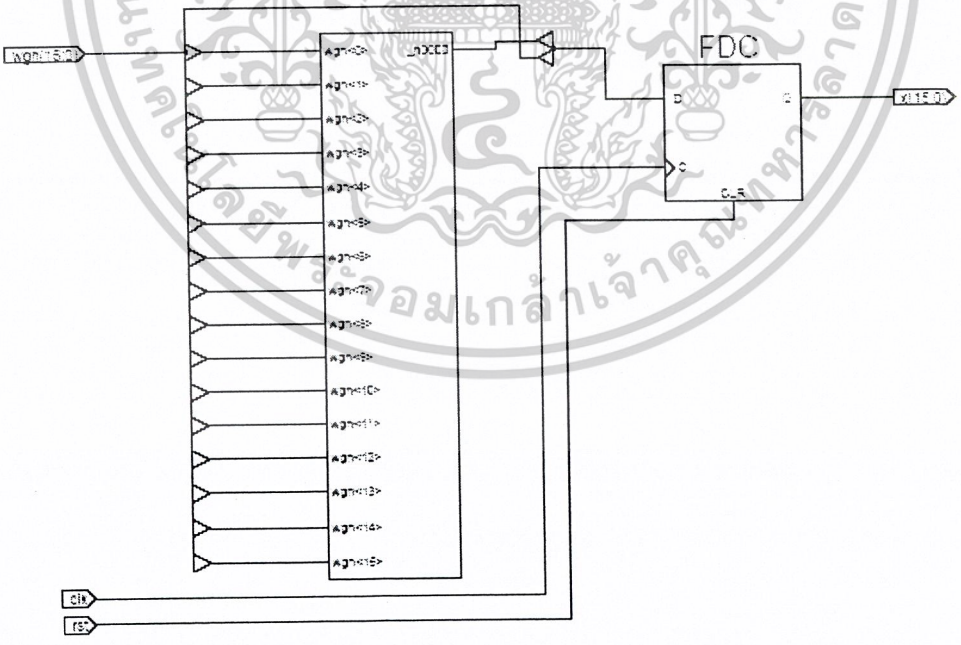
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.17 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลปรับเปลี่ยนค่า

อุปกรณ์	จำนวน
Register	2
15-bit register	1
1-bit register	1

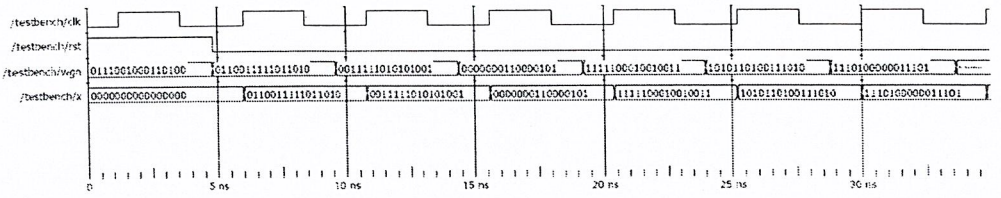
ตารางที่ 5.18 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลปรับเปลี่ยนค่า

อุปกรณ์	Slices	Slices Flip Flop	4 input LUT
จำนวน	9	16	5
Maximum Frequency			310.945MHz
Minimum input arrival time before clock			3.216 ns
Maximum output required time after clock			5.880 ns



รูปที่ 5.21 ผังวงจรระดับ RTL ของมอดูลปรับเปลี่ยนค่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.22 ผลจำลองการทำงานของของมอดูลปรับเปลี่ยนค่า

ตารางที่ 5.18 ซึ่งแสดงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลปรับเปลี่ยนค่า และประกอบด้วยจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop) จำนวนตัวเปิดตารางสี่อินพุต (4-Input LUT) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) และในส่วนของรูปที่ 5.21 แสดงผังวงจรระดับ RTL ของมอดูลปรับเปลี่ยนค่า

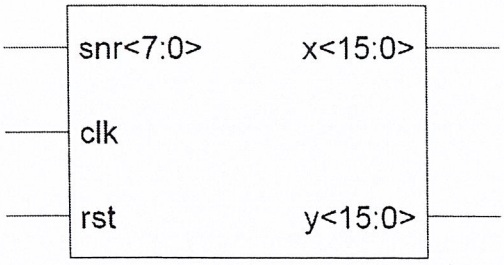
ผลการจำลองการทำงานของมอดูลปรับเปลี่ยนค่า

ผลการจำลองการทำงานของมอดูลปรับเปลี่ยนค่าโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณอินพุตทดสอบของมอดูลปรับเปลี่ยนค่าประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) สัญญาณของสัญญาณรบกวนที่สร้างขึ้นจากมอดูลประมาณค่า SNR ขนาด 16 บิต ซึ่งผลการจำลองแสดงว่าสัญญาณรบกวนที่สร้างขึ้นจะสามารถสร้างขึ้นได้ในทุกรอบของสัญญาณนาฬิกา โดยอยู่ในรูปแบบข้อมูลไบนารีขนาด 16 บิตแบบ 2'Complement ดังรูปที่ 5.22

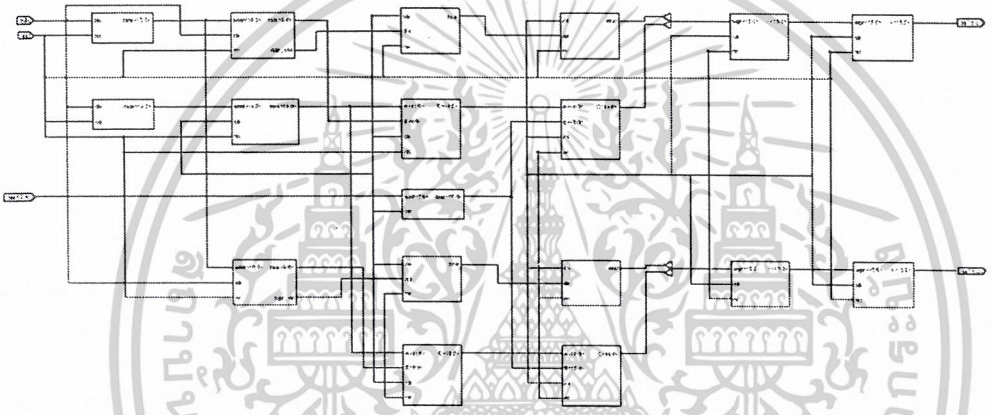
5.2.4 ผลการออกแบบและจำลองการทำงานของชุดกำเนิดสัญญาณรบกวน

มอดูลที่ออกแบบไว้ 4 มอดูล คือ มอดูลชุดกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม มอดูลประมาณค่าฟังก์ชัน มอดูลชุดวงจรคูณค่าฟังก์ชัน และมอดูลปรับเปลี่ยน Eb/No มาประกอบกันเป็นมอดูลชุดกำเนิดสัญญาณรบกวน ซึ่งซื่อสัญญาณอินพุตของมอดูลชุดกำเนิดสัญญาณรบกวนประกอบไปด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) สัญญาณ SNR และสัญญาณรบกวนเอาต์พุตซึ่งแสดงดังรูป 5.23

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.23 ชื่อสัญญาณของชุดกำเนิดสัญญาณรบกวน



รูปที่ 5.24 ผังวงจรระดับ RTL ของชุดกำเนิดสัญญาณรบกวน

ผลการสังเคราะห์มอดูลชุดกำเนิดสัญญาณรบกวน

ผลการสังเคราะห์มอดูลชุดกำเนิดสัญญาณรบกวนโดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งจะประกอบไปด้วยอุปกรณ์ดิจิตอลประเภทต่างๆดังแสดงในตารางที่ 5.19 และ ในตารางที่ 5.20 แสดงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลชุดกำเนิดสัญญาณรบกวน โดยในตารางที่ 5.20 ยังประกอบด้วยจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop) จำนวนตัวเปิดตารางสี่อินพุต (4-Input LUT) หน่วยความจำ (BRAM) วงจรคูณขนาด 18×18 ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) และในส่วนของรูปที่ 5.24 แสดงผังวงจรระดับ RTL ของมอดูลชุดกำเนิดสัญญาณรบกวน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

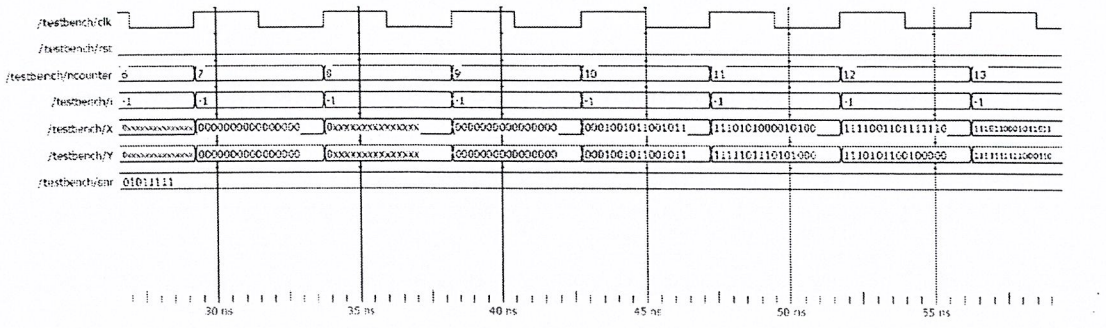
ตารางที่ 5.19 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลชุดกำเนิดสัญญาณรบกวน

อุปกรณ์	จำนวน
ROM	4
1024×12-bit ROM	1
1024×10-bit ROM	2
256×16-bit ROM	1
Register	78
1-bit register	58
15-bit register	6
12-bit register	3
10-bit register	4
17-bit register	4
16-bit register	3
Multiplexer	2
2-to-1 Multiplexer	2
Adder/Subtractor	2
15-bit adder	2
XOR	27
1-bit xor	27

ตารางที่ 5.20 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลชุดกำเนิดสัญญาณรบกวน

อุปกรณ์	Slices	Slices Flip Flop	4 input LUT	BRAM	MULT18×18
จำนวน	171	264	121	4	4
Maximum Frequency				243.962 MHz	
Minimum input arrival time before clock				2.678 ns	
Maximum output required time after clock				5.880 ns	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.25 ผลจำลองการทำงานของมอดูลชุดกำเนิดสัญญาณรบกวน

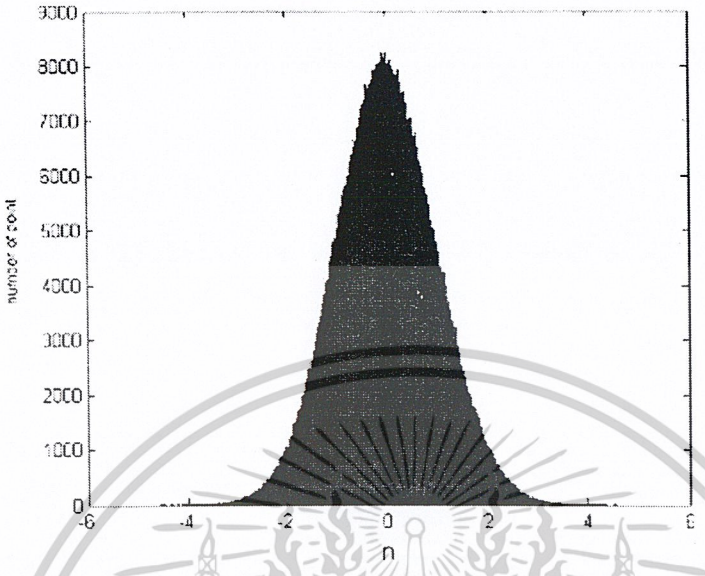
ผลการจำลองการทำงานของมอดูลชุดกำเนิดสัญญาณรบกวน

ผลการจำลองการทำงานของมอดูลชุดกำเนิดสัญญาณรบกวน โดยทดสอบโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณอินพุตทดสอบของมอดูลชุดกำเนิดสัญญาณรบกวนประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) สัญญาณ SNR ขนาด 7 บิตซึ่งผลการจำลองการทำงานของมอดูลชุดกำเนิดสัญญาณรบกวนแสดง ให้เห็นว่าสัญญาณรบกวนที่สร้างขึ้นสัญญาณแรกเมื่อสัญญาณนาฬิกาผ่านไป 10 รอบและจะสร้างต่อเนื่องกันไปทุกรอบของสัญญาณนาฬิกาโดยผลการจำลองการทำงานนี้แสดงสัญญาณ X และ Y ซึ่งเป็นสัญญาณรบกวนขนาด 16 บิต แบบ 2'Complement ดังรูปที่ 5.25

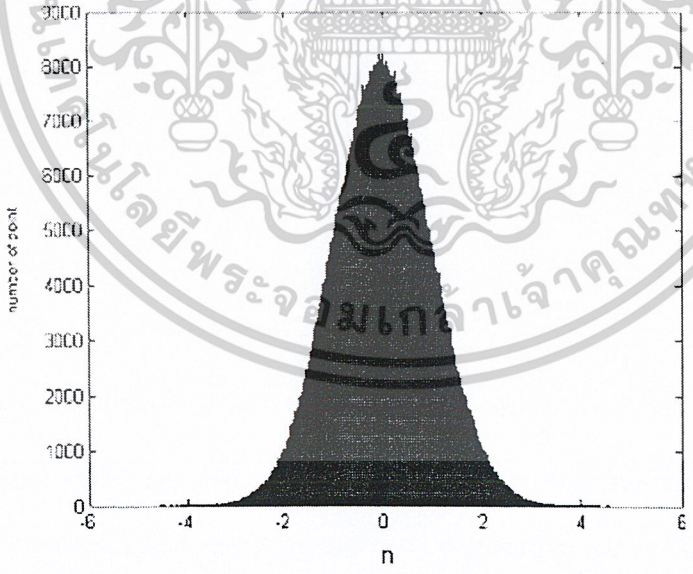
ผลการออกแบบและทดสอบชุดกำเนิดสัญญาณรบกวน

ชุดกำเนิดสัญญาณรบกวนที่มีการกระจายแบบเกาส์ที่ออกแบบใช้ภาษาเวริล็อก และติดตั้งลงบนอุปกรณ์เอฟพีจีเอ Xilinx Vertex-II รุ่น xc2v1000 ผลการออกแบบใช้อุปกรณ์ในเอฟพีจีเอ 171slices 4 บล็อกแรมและ 4 วงจรคูณ ความถี่สูงสุดที่ทำงานได้ 243 MHz ดังนั้นค่าเอาต์พุตสูงสุด 243 MSamples/s การทดสอบประสิทธิภาพทางสถิติของสัญญาณรบกวนที่สร้างขึ้นโดยชุดกำเนิดสัญญาณรบกวนนี้ (สัญญาณ X และ สัญญาณ Y) ได้แก่ การหาค่าฮิสโตรแกรม ค่าฟังก์ชันความหนาแน่นความน่าจะเป็น (PDF) ค่าเฉลี่ย (Mean) ค่าเบี่ยงเบนมาตรฐาน (SD) ค่า Chi-Square (χ^2) และ ค่า Relative Error (ξ_x) ซึ่งทดสอบจากการจำลองการทำงานจำนวน 10^6 samples รูปที่ 5.26 และ 5.27 แสดงฮิสโตรแกรมของเอาต์พุต X และ Y ตามลำดับโดยแกน "x" แสดงถึงค่าสัญญาณรบกวนที่สร้างขึ้น (n) และแกน "y" แสดงจำนวนครั้งในการสร้าง (number of point)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

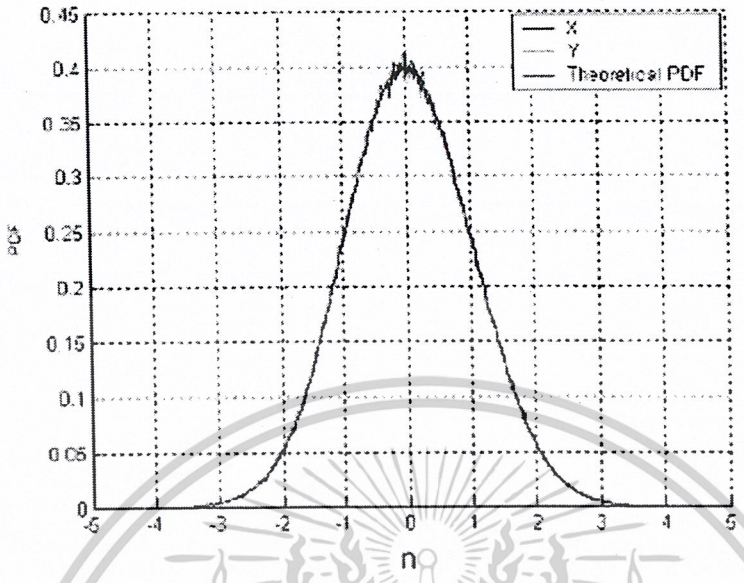


รูปที่ 5.26 ฮิสโตรแกรมของสัญญาณรบกวน X จำนวน 10^6 samples

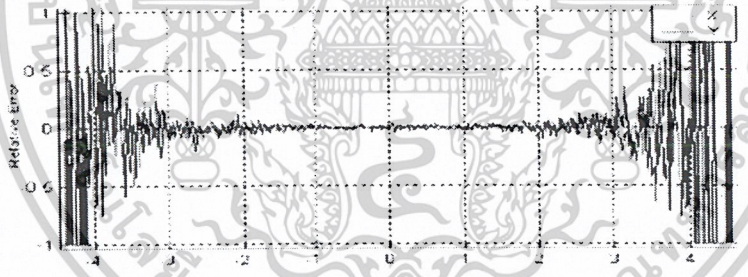


รูปที่ 5.27 ฮิสโตรแกรมของสัญญาณรบกวน Y จำนวน 10^6 samples

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.28 ค่า PDF ของสัญญาณรบกวน X Y และทฤษฎี



รูปที่ 5.29 ค่า relative error ($\xi_x(x)$) ของสัญญาณรบกวน X และ Y

ตาราง ที่ 5.21 การทดสอบทางสถิติของชุดกำเนิดสัญญาณรบกวน

รายละเอียด	งานวิจัยนี้ $(10^6)^*$	
	X	Y
Mean	0.0001	0.0004
SD	0.9988	0.9985
χ^2	0.0274	0.0314

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.22จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลดีมอดูเลเตอร์

อุปกรณ์	จำนวน
Registers	1
16-bit register	1
# Adders/Subtractors	1
16-bit adder	1

ตารางที่ 5.23 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลดีมอดูเลเตอร์

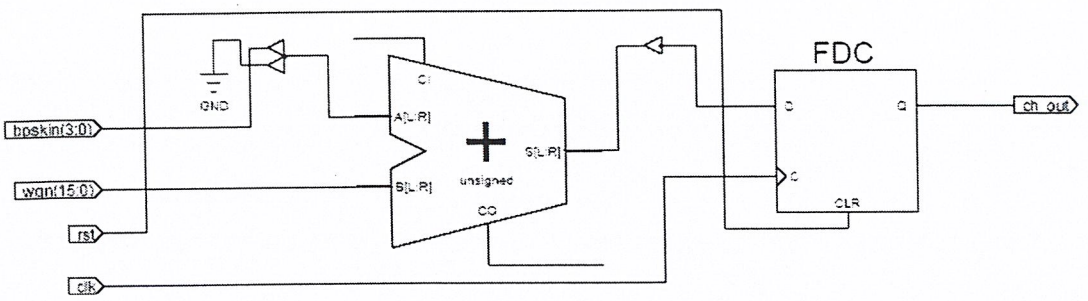
อุปกรณ์	Slices	Slices Flip Flop	4 input LUT
มอดูลเปรียบเทียบข้อมูล	2	1	4
Maximum Frequency			341.180 MHz
Minimum input arrival time before clock			2.931 ns
Maximum output required time after clock			5.880 ns

ในรูปที่ 5.28 แสดงค่า PDF ของเอาต์พุต X และ Y จากการคำนวณทางทฤษฎี ค่า relative error ของค่าฟังก์ชันความหนาแน่นความน่าจะเป็นห่างจากค่าทางทฤษฎีน้อยกว่า 1% ที่ $|x| < 4\sigma$ ดังแสดงในรูปที่ 5.29 และตารางที่ 5.21 แสดงการทดสอบทางสถิติของชุดกำเนิดสัญญาณรบกวนที่ได้สร้างขึ้นโดยประกอบด้วยค่าเฉลี่ย ค่าความแปรปรวน และค่า Relative Error

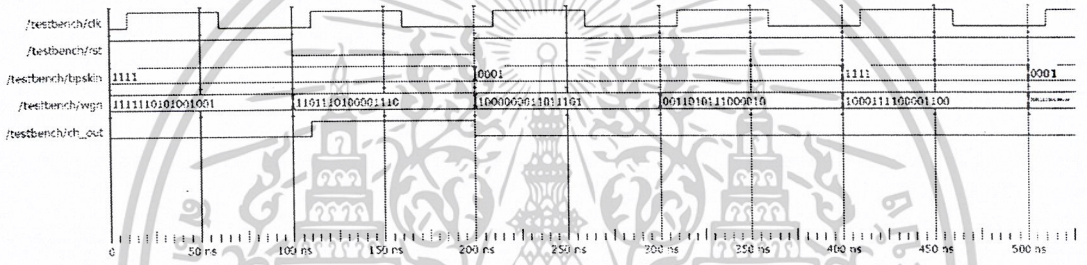
5.2.5 มอดูลดีมอดูเลเตอร์

มอดูลดีมอดูเลเตอร์เป็นมอดูลที่ทำหน้าที่ดีมอดูเลตข้อมูลซึ่งภายในมอดูลดีมอดูเลเตอร์ ถูกออกแบบให้รวมช่องสัญญาณเข้าไปด้วยก่อนทำการดีมอดูเลตข้อมูลกลับมา โดยช่องสัญญาณเป็นการรวมกันของข้อมูลที่ผ่านการมอดูเลตมาจากภาคมอดูเลเตอร์ กับสัญญาณรบกวนที่มอดูลชุดกำเนิดสัญญาณรบกวนสร้างขึ้นก่อนนำข้อมูลที่ได้มาทำการดีมอดูเลต ผลการออกแบบและจำลองการทำงานมีดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.30 ผังวงจรระดับ RTL ของมอดูลดีมอดูเลเตอร์



รูปที่ 5.31 ผลจำลองการทำงานของมอดูลดีมอดูเลเตอร์

ผลการสังเคราะห์มอดูลดีมอดูเลเตอร์

ผลการสังเคราะห์มอดูลดีมอดูเลตโดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งจะประกอบไปด้วยเรจิสเตอร์ และ วงจรบวก ดังแสดงในตารางที่ 5.22 และ ในตารางที่ 5.23 แสดงถึงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลดีมอดูเลเตอร์โดยในตารางจะประกอบด้วยจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop) จำนวนตัวเปิดตารางสี่อินพุต (4-Input LUT) ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) ในส่วนของรูปที่ 5.30 แสดงผังวงจรระดับ RTL ของมอดูลดีมอดูเลเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลการจำลองการทำงานของมอดูลดีมอดูเลเตอร์

ผลการจำลองการทำงานของมอดูลดีมอดูเลเตอร์ทดสอบโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณอินพุตทดสอบของมอดูลดีมอดูเลเตอร์ประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) และสัญญาณข้อมูลที่ผ่านการมอดูเลต (bpskin) ซึ่งแทนด้วยสัญญาณข้อมูล 4 บิต แบบ 2' Complement สัญญาณรบกวน (wgn) เมื่อนำสัญญาณรบกวนที่สร้างขึ้นมารวมกับสัญญาณที่ผ่านการมอดูเลตมาจากมอดูลมอดูเลเตอร์แล้วทำการดีมอดูเลตจะได้ข้อมูลไบนารีขนาด 1บิต (ch_out) ดังรูปที่ 5.31

5.2.6 มอดูลหน่วงเวลา

มอดูลหน่วงเวลาเป็นมอดูลที่ทำการหน่วงเวลาข้อมูลจากมอดูลกำเนิดข้อมูลโดย Delay1 เป็นการหน่วงเวลาข้อมูลที่มาจากมอดูลกำเนิดข้อมูล เพื่อที่ทำให้ข้อมูลเดียวกันที่ส่งผ่านมอดูลมอดูเลเตอร์และมอดูดีมอดูเลเตอร์สามารถเปรียบเทียบกันได้เพื่อหาจำนวนบิตข้อมูลที่ผิดพลาด และ Delay2 เป็นการหน่วงเวลาข้อมูลที่ออกจากมอดูลกำเนิดข้อมูลเพื่อที่ นำมาเปรียบเทียบกับข้อมูลชุดเดียวกันที่ส่งผ่านมอดูลมอดูเลเตอร์ มอดูเข้ารหัส มอดูดีมอดูเลเตอร์ และมอดูลถอดรหัสเพื่อหาจำนวนบิตข้อมูลที่ผิดพลาด ซึ่งผลการออกแบบและจำลองการทำงานของมอดูลหน่วงเวลามีดังนี้

ผลการสังเคราะห์มอดูลหน่วงเวลา

ผลการสังเคราะห์มอดูลหน่วงเวลา โดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งประกอบไปด้วยชิพเรจิสเตอร์เพื่อใช้เก็บข้อมูลดังแสดงในตารางที่ 5.24 และ ในตารางที่ 5.25 แสดงถึงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลหน่วงเวลาโดยในตารางจะประกอบด้วยจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop) ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) และในส่วนของรูปที่ 5.32 แสดงผังวงจรระดับ RTL ของมอดูลหน่วงเวลา

ผลการจำลองการทำงานของมอดูลหน่วงเวลา

ผลการจำลองการทำงานของมอดูลหน่วงเวลา ซึ่งทดสอบโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณอินพุตทดสอบของมอดูลหน่วงเวลาประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) และสัญญาณข้อมูล (din) ที่มาจากมอดูลกำเนิดข้อมูลซึ่งผลการจำลองการทำงานของมอดูลหน่วงเวลาจะแสดงการหน่วงเวลาของข้อมูลจากแหล่งกำเนิดตามสัญญาณนาฬิกา ดังรูปที่ 5.33

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.24 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลหน่วงเวลา

อุปกรณ์	จำนวน
Register	
Delay 1	2
1-bit register	2
Delay 2	24
1-bit register	24

ตารางที่ 5.25 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลหน่วงเวลา

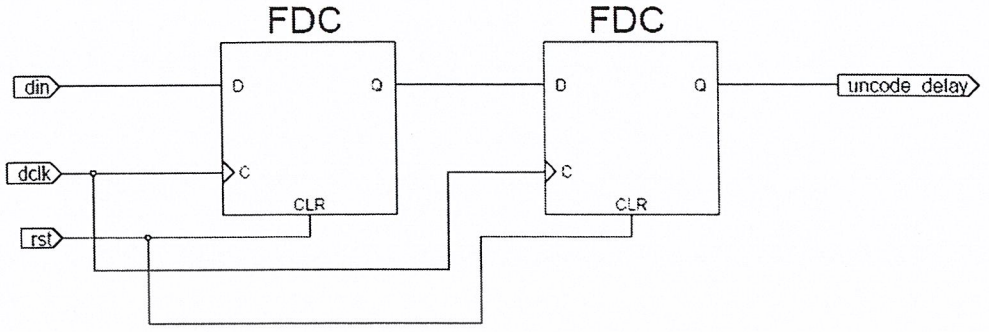
อุปกรณ์	Slices	Slices Flip Flop
Delay 1	1	2
Delay 2	14	24
Delay	Delay1	Delay2
Maximum Frequency	948.767 MHz	948.767MHz
Minimum input arrival time before clock	1.480 ns	2.536 ns
Maximum output required time after clock	5.880 ns	5.880 ns

โดยในรูปที่ 5.33 (ก) แสดงการหน่วงเวลา 2 รอบของสัญญาณนาฬิกาเพื่อนำข้อมูลที่ได้มาเปรียบเทียบกับข้อมูลที่ผ่านการมอดูเลตและดีมอดูเลตโดยที่ไม่ต้องผ่านมอดูเข้ารหัส และถอดรหัสเพื่อหาจำนวนบิตข้อมูลที่ผิดพลาด ส่วนในรูป 5.33 (ข) แสดงการหน่วงเวลาข้อมูล 23 รอบของสัญญาณนาฬิกาเพื่อนำมาเปรียบเทียบกับข้อมูลที่ผ่านมอดูลเข้ารหัส มอดูลมอดูเลเตอร์ มอดูลดีมอดูเลเตอร์ และมอดูลถอดรหัสแล้วหาจำนวนบิตข้อมูลที่ผิดพลาด

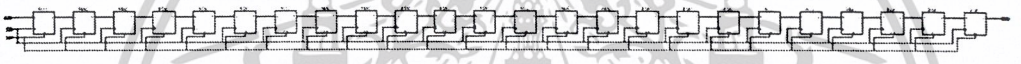
5.2.7 มอดูลเปรียบเทียบข้อมูล (Comparator)

มอดูเปรียบเทียบข้อมูลที่เป็นส่วนหนึ่งของชุดทดสอบรหัสแก้ไขข้อผิดพลาด ซึ่งมอดูลเปรียบเทียบข้อมูลนี้ประกอบด้วยวงจรนับการผิดพลาดของบิต และวงจรเปรียบเทียบการผิดของบิตผลการออกแบบและจำลองการทำงานของมอดูเปรียบเทียบข้อมูลมีดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(ก)

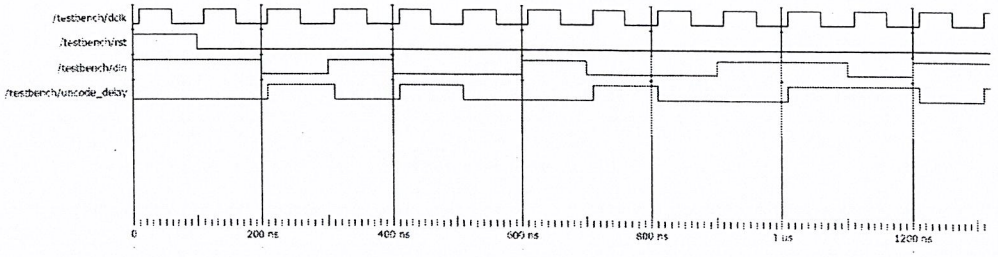


(ข)

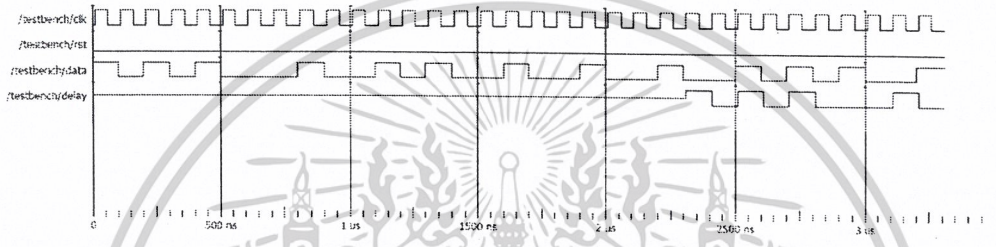
รูปที่ 5.32 ผังวงจรระดับ RTL ของมอดูลหน่วงเวลา

ก) Delay1 (ข) Delay 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(ก)



(ข)

รูปที่ 5.33 ผลจำลองการทำงานของมอดูลหน่วงเวลา

ก) Delay 1 ข) Delay 2

ผลการสังเคราะห์มอดูลเปรียบเทียบข้อมูล

ผลการสังเคราะห์มอดูลเปรียบเทียบข้อมูลโดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งจะประกอบไปด้วยเรจิสเตอร์ วงจรนับบิตผิดขนาด 24 บิต และวงจรเปรียบเทียบ (Comparator) ดังแสดงในตารางที่ 5.26 และ ในตารางที่ 5.27 ซึ่งแสดงถึงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลเปรียบเทียบข้อมูลโดยในตารางจะประกอบด้วยจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop)จำนวนตัวเปิดตารางสี่อินพุต (4-Input LUT) ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อย ที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) และในส่วนของรูปที่ 5.34 แสดงผังวงจรระดับ RTL ของมอดูลเปรียบเทียบข้อมูล

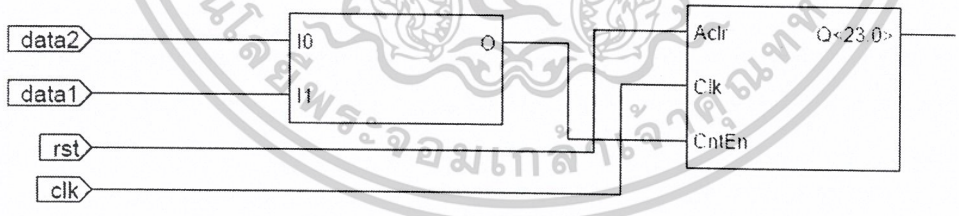
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.26 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลเปรียบเทียบข้อมูล

อุปกรณ์	จำนวน
Register	1
1-bit register	1
Counter	1
24-bit up counter	1
Comparators	1
21-bit comparator greatequal	1

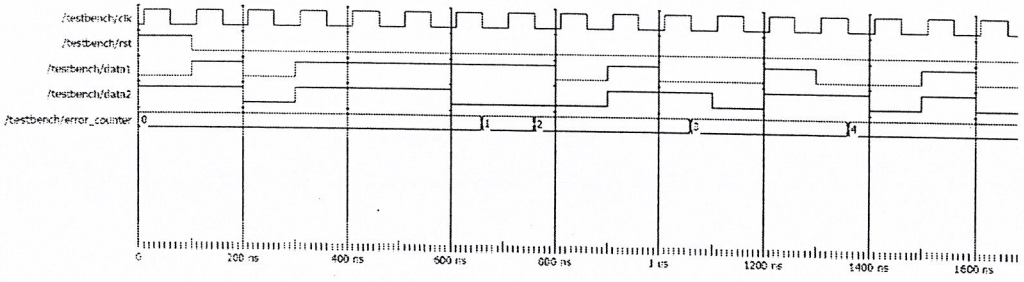
ตารางที่ 5.27 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลเปรียบเทียบข้อมูล

อุปกรณ์	Slices	Slices Flip Flop	4 input LUT
มอดูลเปรียบเทียบข้อมูล	21	22	38
Maximum Frequency			200.562 MHz
Minimum input arrival time before clock			2.444 ns
Maximum output required time after clock			5.880 ns



รูปที่ 5.34 ผังวงจรระดับ RTL ของมอดูลเปรียบเทียบข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.35 ผลจำลองการทำงานของมอดูลเปรียบเทียบข้อมูล

ผลการจำลองการทำงานของมอดูลเปรียบเทียบข้อมูล

ผลการจำลองการทำงานของมอดูลเปรียบเทียบข้อมูลซึ่งทดสอบโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณอินพุตทดสอบของมอดูลเปรียบเทียบข้อมูลประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) และสัญญาณข้อมูล (data) ที่ต้องการเปรียบเทียบสองสัญญาณซึ่งมาจากมอดูลหน่วงเวลาและมอดูลดีมอดูเลเตอร์ โดยก่อนการเปรียบเทียบต้องเซ็ตสัญญาณรีเซ็ตเป็น 1 ก่อน และผลที่ได้จะนำมาผ่านวงจรนับเพื่อนับบิตผิดดังแสดงในสัญญาณ error_counter ที่บอกจำนวนบิตผิด ดังรูปที่ 5.35

5.3 ผลการจำลองการทำงานของชุดเข้ารหัสและถอดรหัสคอนโวลูชัน

การออกแบบชุดเข้ารหัสและถอดรหัสคอนโวลูชันเพื่อใช้ในการทดสอบชุดทดสอบรหัสแก้ไขข้อผิดพลาดโดยการออกแบบใช้ K เท่ากับ 3 Code Rate เท่ากับ 1/2 ซึ่งผลการออกแบบและจำลองการทำงานมีดังต่อไปนี้

5.3.1 มอดูลเข้ารหัสคอนโวลูชัน

มอดูลดีมอดูเลตเป็นส่วนหนึ่งของ ชุดทดสอบรหัสแก้ไขข้อผิดพลาดภายในมอดูลนี้ประกอบด้วยชิพเรจิสเตอร์ขนาด และ XOR เกตซึ่ง เป็นอุปกรณ์พื้นฐานที่ใช้เข้ารหัสคอนโวลูชัน ซึ่งในชุดทดสอบรหัสแก้ไขข้อผิดพลาดใช้ Code Rate เท่ากับ 1/2 ดังนั้นบิตข้อมูล 1 บิต จะให้ข้อมูลที่ถูกเข้ารหัส 2 บิต โดยผลการออกแบบและจำลองการทำงานของมอดูลดีมอดูเลตมีดังนี้

ผลการสังเคราะห์มอดูลเข้ารหัสคอนโวลูชัน

ผลการสังเคราะห์มอดูลเข้ารหัสคอนโวลูชันโดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งประกอบไปด้วยอุปกรณ์ดิจิตอลประเภทต่างๆดังแสดงในตารางที่ 5.28

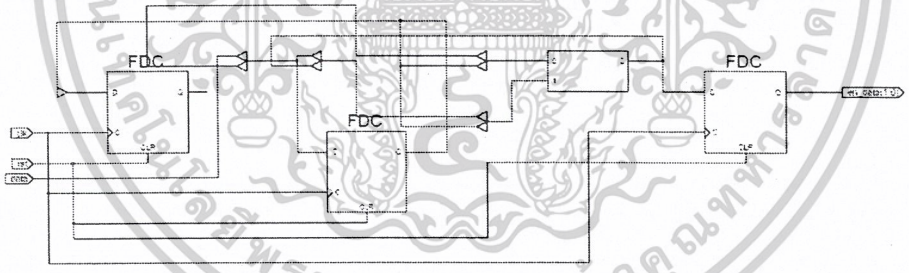
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.28 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลเข้ารหัสคอนโวลูชัน

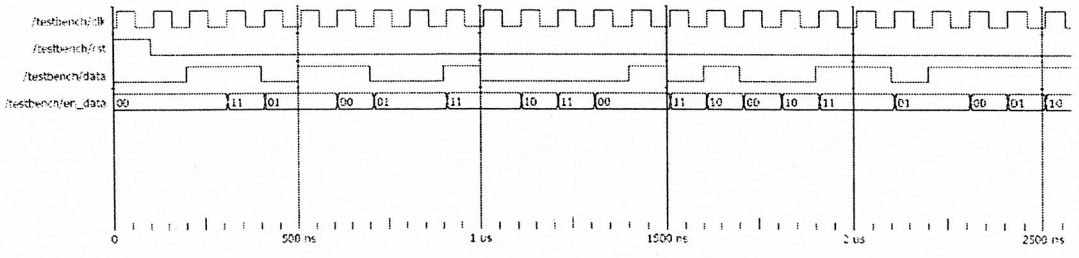
อุปกรณ์	จำนวน
Register	5
1-bit register	5
Xors	2
1 –bit xor2	2

ตารางที่ 5.29 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลเข้ารหัสคอนโวลูชัน

อุปกรณ์	Slices	Slices Flip Flop	4 input LUT
จำนวน	3	5	2
Maximum Frequency			543.774 MHz
Minimum input arrival time before clock			1.730 ns
Maximum output required time after clock			5.880 ns



รูปที่ 5.36 ผังวงจรระดับ RTL ของมอดูลเข้ารหัสคอนโวลูชัน K=3 Rate =1/2



รูปที่ 5.37 ผลจำลองการทำงานของมอดูลเข้ารหัสคอนโวลูชัน K = 3 Rate = 1/2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.29 ซึ่งแสดงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของเข้ารหัสคอนโวลูชันโดยในตารางจะประกอบด้วยจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop)จำนวนตัวเปิดตารางสี่อินพุต (4-Input LUT) ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) และในส่วนของรูปที่ 5.36 แสดงผังวงจรระดับ RTL ของมอดูลเข้ารหัสคอนโวลูชัน

ผลการจำลองการทำงานของมอดูลเข้ารหัสคอนโวลูชัน

ผลการจำลองการทำงานของมอดูลเข้ารหัสคอนโวลูชัน ทดสอบโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณทดสอบของมอดูลเข้ารหัสคอนโวลูชัน ประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) และสัญญาณข้อมูลขนาด 1 บิต (data) ที่มาจากภาคกำเนิดข้อมูล ซึ่งผลการจำลองการทำงานของมอดูลเข้ารหัสคอนโวลูชันจะแสดงในรูปของข้อมูล (en_data) ไบนารีขนาด 2 บิตดังรูปที่ 5.37

5.3.2 มอดูลถอดรหัสคอนโวลูชัน

เมื่อข้อมูลที่ผ่านการเข้ารหัสที่มอดูลเข้ารหัสคอนโวลูชันถูกส่งมายังมอดูลมอดูเลเตอร์ และส่งต่อมายังมอดูดีมอดูเลเตอร์โดยภายในจะประกอบด้วยช่องสัญญาณที่เป็นการรวมกัน ระหว่างบิตข้อมูลที่ผ่านการมอดูเลตกับสัญญาณรบกวนก่อนที่จะทำการดีมอดูเลตกลับมา ซึ่งอาจเกิดการผิดพลาดของบิตข้อมูลแล้วจึงนำข้อมูลเหล่านี้จะถูกส่งต่อมายังมอดูลถอดรหัสคอนโวลูชัน เพื่อทำการถอดรหัสข้อมูลที่ถูกส่งมาและทำการแก้ไขข้อมูลที่ผิดพลาดโดยผลการออกแบบและ จำลองการทำงานของมอดูลถอดรหัสคอนโวลูชันมีดังต่อไปนี้

ผลการสังเคราะห์มอดูลถอดรหัสคอนโวลูชัน

ผลการสังเคราะห์มอดูลถอดรหัสคอนโวลูชันโดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ต-ลิสต์ ซึ่งจะประกอบไปด้วยอุปกรณ์ดิจิตอลประเภทต่างๆดังแสดงในตารางที่ 5.30 และ ในตารางที่ 5.31 ซึ่งแสดงถึงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของมอดูลถอดรหัสคอนโวลูชันโดยในตารางจะประกอบด้วยจำนวน Slices จำนวน ฟลิปฟลอป (Slice Flip Flop)จำนวนตัวเปิดตารางสี่อินพุต (4-Input LUT) ค่าความถี่สูงสุด (Maximum Frequency)

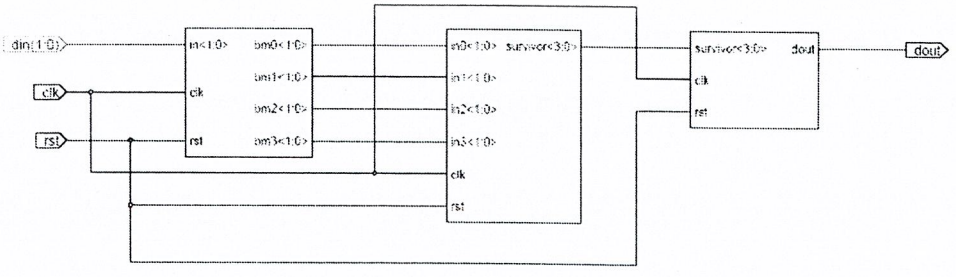
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.30 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของมอดูลถอดรหัสคอนโวลูชัน

อุปกรณ์	จำนวน
Registers	68
2-bit register	4
4-bit register	5
1-bit register	59
Multiplexers	63
2-to-1 multiplexer	63
Adders/Subtractors	8
4-bit adder	8
Comparators	4
4-bit comparator greater	4
Xors	4
1-bit xor3	4

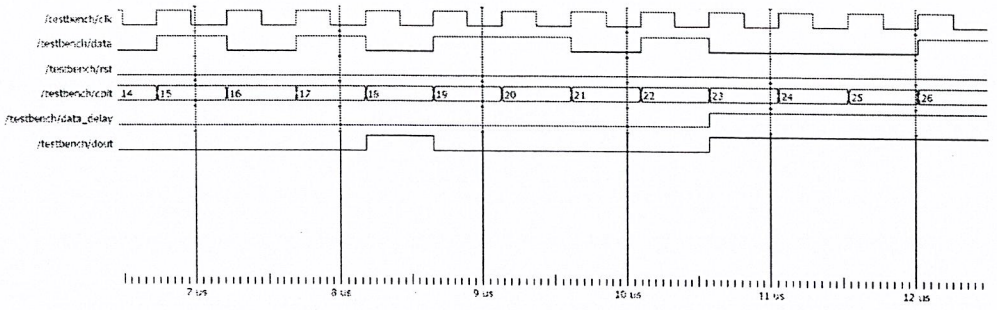
ตารางที่ 5.31 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของมอดูลถอดรหัสคอนโวลูชัน

อุปกรณ์	Slices	Slices Flip Flop	4 input LUT
จำนวน	74	85	131
Maximum Frequency			160.953 MHz
Minimum input arrival time before clock			3.100 ns
Maximum output required time after clock			5.880 ns



รูปที่ 5.38 ผังวงจรระดับ RTL ของมอดูลถอดรหัสคอนโวลูชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.39 ผลจำลองการทำงานของมอดูลถอดรหัสคอนโวลูชัน

ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock)ในส่วนของรูปที่ 5.38 แสดงผังวงจรระดับ RTL ของมอดูลถอดรหัสคอนโวลูชัน

ผลการจำลองการทำงานของมอดูลถอดรหัสคอนโวลูชัน

ผลการจำลองการทำงานของมอดูลถอดรหัสคอนโวลูชันทดสอบโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณอินพุตทดสอบของมอดูลถอดรหัสคอนโวลูชันประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) และสัญญาณข้อมูลที่ผ่านการดีมอดูเลต (din) ข้อมูลที่ถูกถอดรหัส (dout) และข้อมูลที่ผ่านมอดูลหน่วงเวลา (data_delay) เพื่อเป็นการเปรียบเทียบการทำงานของมอดูลถอดรหัสคอนโวลูชัน ซึ่งผลการจำลองการทำงานของมอดูลถอดรหัสคอนโวลูชันจะแสดงดังรูปที่ 5.39 โดยจะเห็นได้ว่าการถอดรหัสจะเริ่มถูกต้องเมื่อสัญญาณนาฬิกาผ่านไปครบ 23 รอบ

5.4 ผลการออกแบบและจำลองการทำงานรวมของชุดทดสอบรหัสแก้ไขข้อผิดพลาด

ชุดทดสอบรหัสแก้ไขข้อผิดพลาดนั้นประกอบไปด้วยส่วนต่างๆซึ่งประกอบไปด้วย ส่วนกำเนิดข้อมูล ส่วนมอดูเลเตอร์ ส่วนเข้ารหัส ส่วนกำเนิดสัญญาณรบกวน ส่วนถอดรหัส ส่วนดีมอดูเลเตอร์ และส่วนเปรียบเทียบข้อมูลโดยในการออกแบบชุดทดสอบรหัสแก้ไขข้อผิดพลาดจะแบ่งการทำงานเป็นสองเส้นทางคือเส้นทางแรกเป็นการส่งข้อมูลจากกมอดูลกำเนิดข้อมูลผ่าน มอดูลมอดูเลเตอร์ มอดูลดีมอดูเลเตอร์ และมอดูลเปรียบเทียบข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.32 จำนวนอุปกรณ์ที่ได้จากการสังเคราะห์ของชุดทดสอบรหัสแก้ไขข้อผิดพลาด

อุปกรณ์	จำนวน
RAM	7
1024x10-bit single-port block RAM	3
1024x12-bit single-port block RAM	2
256x16-bit single-port block RAM	2
Registers	281
1-bit register	229
10-bit register	3
12-bit register	3
15-bit register	9
16-bit register	6
17-bit register	6
2- bit register	4
3-bit register	1
4-bit register	8
6- bit register	7
8-bit register	5
Multiplexers	2
1-bit 8-to-1 multiplexer	1
8-bit 8-to-1 multiplexer	1
Adders/ Subtractors	22
15-bit adder	3
16-bit adder	3
4-bit adder	9
6- bit adder	7
Comparators	5
21-bit comparator greatequal	1
4 -bit comparator greater	4
Xor	4
1 –bit xor3	4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.33 อุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วการทำงานของชุดทดสอบรหัสแก้ไขข้อผิดพลาด

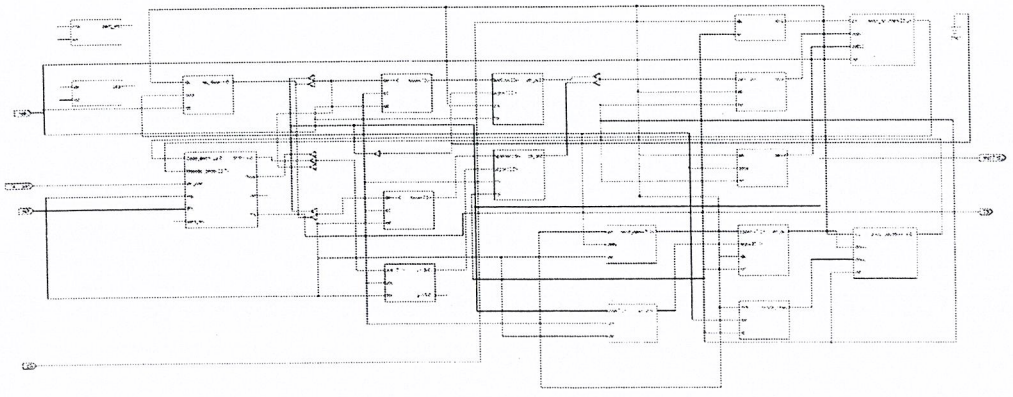
อุปกรณ์	Slices	Slices Flip Flop	4 input LUT	BRAM	MULT18×18
ชุดทดสอบรหัสแก้ไขข้อผิดพลาด	421	625	509	7	6
Maximum Frequency					160.953 MHz
Minimum input arrival time before clock					4.828 ns
Maximum output required time after clock					6.318 ns

โดยไม่ผ่านมอดูลเข้ารหัสคอนโวลูชันและมอดูลถอดรหัสคอนโวลูชัน ส่วนอีกเส้นทางหนึ่งเป็นเส้นทางที่ผ่านมอดูลเข้ารหัสคอนโวลูชันและมอดูลถอดรหัสคอนโวลูชัน ซึ่งมีผลการออกแบบและจำลองการทำงานของชุดทดสอบรหัสแก้ไขข้อผิดพลาดมีดังนี้

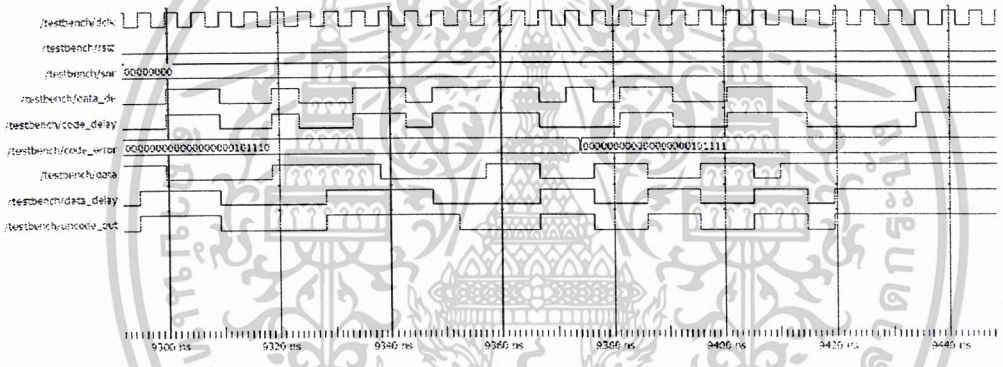
ผลการสังเคราะห์ชุดทดสอบรหัสแก้ไขข้อผิดพลาด

ผลการสังเคราะห์ชุดทดสอบรหัสแก้ไขข้อผิดพลาด โดยใช้โปรแกรม Xilinx ISE จะได้กลุ่มของเน็ตลิสต์ ซึ่งจะประกอบไปด้วยอุปกรณ์ดิจิตอลประเภทต่างๆดังแสดงในตารางที่ 5.32 และ ในตารางที่ 5.33 ซึ่งแสดงถึงจำนวนของอุปกรณ์ที่ใช้ในเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 และความเร็วในการทำงานของชุดทดสอบรหัสแก้ไขข้อผิดพลาด โดยในตารางประกอบด้วยจำนวน Slices จำนวนฟลิปฟลอป (Slice Flip Flop) จำนวนตัวเปิดตารางสี่อินพุต (4-Input LUT) จำนวนบล็อกหน่วยความจำ (BRAM) จำนวนบล็อกวงจรคูณขนาด 18×18 ค่าความถี่สูงสุด (Maximum Frequency) ค่าเวลาน้อยที่สุดที่สัญญาณอินพุตต้องมาก่อนสัญญาณนาฬิกา (Minimum Input Arrival Time Before Clock) ค่าเวลามากสุดที่สัญญาณเอาต์พุตเกิดเกิดหลังสัญญาณนาฬิกา (Maximum Output Required Time After Clock) และในส่วนของรูปที่ 5.40 แสดงผังวงจรระดับ RTL ของชุดทดสอบรหัสแก้ไขข้อผิดพลาด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.40 ผังวงจรระดับ RTL ของชุดทดสอบรหัสแก้ไขข้อผิดพลาด

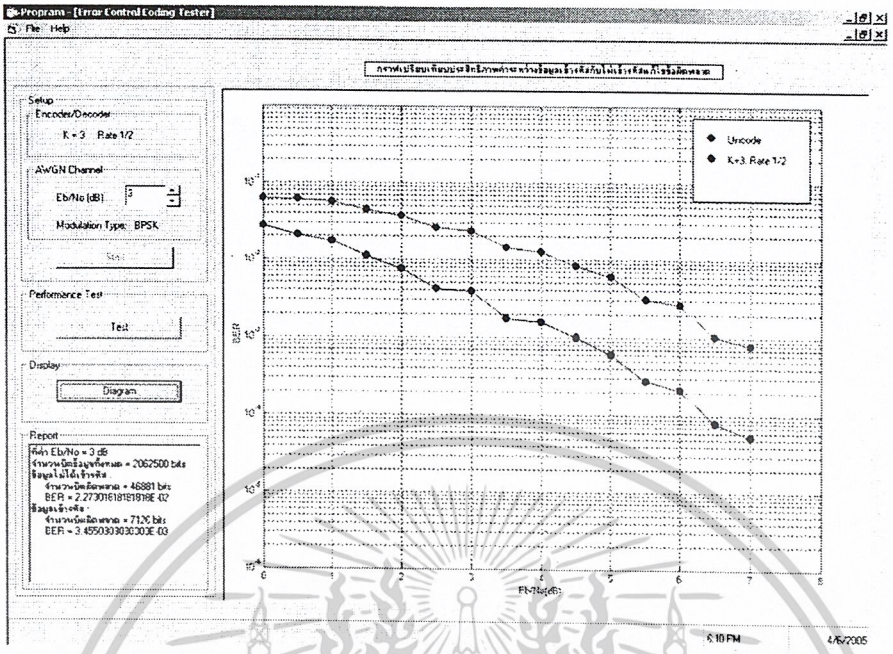


รูปที่ 5.41 ผลจำลองการทำงานของชุดทดสอบรหัสแก้ไขข้อผิดพลาด

ผลการจำลองการทำงานของชุดทดสอบรหัสแก้ไขข้อผิดพลาด

ผลการจำลองการทำงานของชุดทดสอบรหัสแก้ไขข้อผิดพลาดซึ่งทดสอบโดยใช้โปรแกรม Xilinx ISE โดยสัญญาณอินพุตทดสอบของชุดทดสอบรหัสแก้ไขข้อผิดพลาดประกอบด้วยสัญญาณนาฬิกา (clk) สัญญาณรีเซ็ต (rst) และสัญญาณข้อมูล (data) ที่มาจากมอดูลกำเนิดข้อมูล สัญญาณข้อมูลที่เข้ารหัส (data_de) สัญญาณข้อมูลที่ไม่ได้เข้ารหัส (uncode_data) และสัญญาณที่มาจากมอดูลหน่วงเวลาคือหน่วงเวลาข้อมูลที่เข้ารหัส (code_delay) และสัญญาณหน่วงเวลาข้อมูลที่ไม่ได้เข้ารหัส (uncode_delay) และนำสัญญาณข้อมูลเข้ารหัส และข้อมูลหน่วงเวลามาทำการเปรียบเทียบเพื่อหาจำนวนบิตข้อมูลที่ผิดพลาด (error_counter) ขนาด 24 บิต ดังรูปที่ 5.41

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.42 ผลการทดสอบรหัสคอนโวลูชัน

5.5 ผลการทดสอบชุดทดสอบรหัสคอนโวลูชัน

ในการทดสอบการทำงานของชุดทดสอบรหัสแก้ไขข้อผิดพลาดโดยใช้รหัสคอนโวลูชัน ที่มีค่า K = 3 และ Code Rate = 1/2 ซึ่งทำการออกแบบโดยใช้ภาษาเวริล็อกและติดตั้งชุดทดสอบในรูปแบบของ IP Core ลงเอฟพีจีเอ Xilinx Vertex II รุ่น xc2v1000 โดยโปรแกรมในการจำลองการทำงานของชุดทดสอบรหัสแก้ไขข้อผิดพลาดนี้ [1][2][3][4] เป็นส่วนหนึ่งของโครงการพัฒนาต้นแบบสำหรับการสื่อสารแบบไร้สาย โครงการวิจัยระบบโทรคมนาคมสำหรับโทรศัพท์เคลื่อนที่รุ่นที่ 3 ของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ (สวทช.) ร่วมกับ ห้องปฏิบัติการ DSLABs ของทางสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง โดยได้พัฒนาต่อให้รองรับการทดสอบรหัสคอนโวลูชันโดยผลการทดสอบจะแสดงผ่านทางหน้าจอคอมพิวเตอร์ในรูปแบบ Bit Error Rate (BER) ซึ่งเป็นการเปรียบเทียบระหว่างข้อมูลที่เข้ารหัสคอนโวลูชันและไม่ได้เข้ารหัสคอนโวลูชันที่ค่า Eb/No ต่างๆดังรูปที่ 5.42

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6
สรุป

รหัสแก้ไขข้อผิดพลาดเป็นเทคนิคสำคัญอย่างหนึ่งในระบบสื่อสารเพื่อช่วยลดความผิดพลาดที่อาจเกิดขึ้นในการรับส่งข้อมูลได้ ซึ่งปัจจัยหนึ่งที่ทำให้เกิดความผิดพลาดนั้นเรียกว่า สัญญาณรบกวน การเข้ารหัสแบบคอนโวลูชันและการถอดรหัสแบบไวเทอร์บีเป็นรูปแบบหนึ่งที่มีประสิทธิภาพสูงนิยมนำมาใช้แก้ปัญหานี้ โดยเฉพาะในระบบโทรศัพท์เคลื่อนที่ยุคที่ 3 ของ 3GPP กำหนดให้รหัสนี้เป็นมาตรฐานการเข้ารหัสในช่องสัญญาณเสียงด้วย และมีงานวิจัยสร้างอุปกรณ์นี้มากมาย หนึ่งในนั้นคือ ชุดถอดรหัสไวเทอร์บีขนาดเล็กของ NECTEC ร่วมมือกับ KMITL [1][2][3] โครงงานนี้จึงสร้างชุดทดสอบรหัสแก้ไขข้อผิดพลาดสำหรับการเข้ารหัสแบบคอนโวลูชัน (Code Rate เท่ากับ 1/2 และ K เท่ากับ 3) และมีการถอดรหัสไวเทอร์บีขึ้นเพื่อทดสอบประสิทธิภาพของรหัสนั้น โดยการวัดค่าความผิดพลาดบิตในช่องสัญญาณแบบ Additive White Gaussian Noise (AWGN) และนำมาแสดงผลทางคอมพิวเตอร์

6.1 สรุปโครงงาน

โครงงานนี้เป็นการสร้างชุดทดสอบรหัสแก้ไขข้อผิดพลาดโดย วัดอัตราการผิดพลาดของบิตข้อมูล (BER) ต่อค่าอัตราส่วนการใช้กำลังงานต่อสัญญาณรบกวน (Eb/No) ในช่องสัญญาณแบบ AWGN ซึ่งภายในชุดทดสอบรหัสแก้ไขข้อผิดพลาดประกอบด้วยส่วนต่างๆ เช่น ภาคกำเนิดข้อมูล ภาคมอดูเลเตอร์ ภาคกำเนิดสัญญาณรบกวน ภาคดีมอดูเลเตอร์ ภาคหน่วงเวลา และภาคเปรียบเทียบข้อมูล การออกแบบชุดกำเนิดสัญญาณรบกวนใช้วิธีบ็อกมูลเลอร์ (Box-Muller) ร่วมกับวิธีการประมาณค่าฟังก์ชั่นค่ารากที่สองและ ล็อกการิทึมแบบลิเนียร์แบบแบ่งออกสี่ส่วน ทำให้ชุดกำเนิดสัญญาณรบกวนมีโครงสร้างขนาดเล็กและสามารถทำงานได้ที่ความเร็วสูง [4] ผลการออกแบบชุดกำเนิดสัญญาณรบกวนใช้อุปกรณ์ในเอฟพีจีเอ 171 slices 4 บล็อกแรมและ 4 วงจรคูณ และให้ค่าเอาต์พุตสูงสุดที่อัตรา 243 MSamples/s มีค่า relative error ของค่าฟังก์ชันความหนาแน่นความน่าจะเป็นห่างจากค่าทางทฤษฎีน้อยกว่า 1% และผลการออกแบบมอดูลย่อยต่างๆแล้วนำมาประกอบกันเป็นมอดูลชุดทดสอบรหัสแก้ไขข้อผิดพลาดใช้อุปกรณ์ในเอฟพีจีเอ จำนวน 421 slices 7 บล็อกแรมและ 4 วงจรคูณและทำงานด้วยความเร็วสูงสุด 160 MHz การทดสอบรหัสแก้ไขข้อผิดพลาดนั้นใช้รหัสคอนโวลูชันที่ Code Rate เท่ากับ 1/2 และ K เท่ากับ 3 เป็นรหัสทดสอบ ผลการทดสอบแสดงในรูปแบบของกราฟความสัมพันธ์ระหว่างอัตราการผิดพลาดของบิต (BER) กับอัตราส่วนการใช้กำลังงานในการส่งต่อสัญญาณรบกวน (Eb/No) โดยเปรียบเทียบระหว่างข้อมูลที่เข้ารหัสและข้อมูลที่ไม่ได้เข้ารหัส ซึ่งกราฟค่าความผิดพลาดของบิตที่ได้มีลักษณะตามทฤษฎีซึ่งแสดงให้เห็นว่าชุดทดสอบรหัสแก้ไขข้อผิดพลาดนี้สามารถทดสอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รหัสแก้ไขข้อผิดพลาดนี้ได้ และสามารถพัฒนาให้ชุดทดสอบนี้ใช้กับอุปกรณ์รหัสแก้ไขข้อผิดพลาดอื่นๆ ได้

6.2 ข้อเสนอแนะและสิ่งที่สามารถพัฒนาต่อไปได้

ชุดทดสอบในโครงงานนี้พัฒนาอยู่ในรูป IP Core การทดสอบกับรหัสแก้ไขข้อผิดพลาดต้องติดตั้งทั้งสองส่วนในชิพ FPGA แล้วส่งผลการทดสอบไปแสดงผลที่คอมพิวเตอร์ ซึ่งข้อเสนอแนะในการพัฒนามีดังนี้

- การสร้างส่วนแสดงผลเพิ่ม เช่น แสดงผลโดยใช้ LCD จะทำให้เพิ่มความสะดวกในการทดสอบและเคลื่อนย้าย
- พัฒนาให้ชุดทดสอบนี้สามารถทดสอบกับอุปกรณ์รหัสแก้ไขข้อผิดพลาดในเพล็ตฟอร์มอื่นๆ เช่น อุปกรณ์ ASIC หรือ DSP เป็นต้น โดยการพัฒนาส่วนอินเตอร์เฟสของ FPGA ให้สามารถเชื่อมต่ออุปกรณ์เหล่านี้ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

[1] นายชัยวัฒน์ แก้วสาย. "การออกแบบสร้างชุดถอดรหัส viterbi ขนาดเล็กสำหรับระบบโทรศัพท์เคลื่อนที่ยุคที่3." วิทยานิพนธ์วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมสารสนเทศ บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.2546

[2] Chaiwat Keawsai, Keattisak Sripimanwat, and Attasit Lasakul "Modified Register Exchange Method of Viterbi Decoder for 3GPP Mobile System" The 1st ECTI Annual Conference (ECTI-CON 2004), 2004. Thailand

[3] ชัยวัฒน์ แก้วสาย, เกียรติศักดิ์ ศรีพิมานวัฒน์, อรรถสิทธิ์ หล่าสกุล "การออกแบบสร้างชุดถอดรหัส Viterbi ขนาดเล็กสำหรับระบบโทรศัพท์เคลื่อนที่ยุคที่ 3" การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 26 (EECON-26) 2546.

[4] นายชัยวัฒน์ แก้วสาย, นายเกียรติศักดิ์ ศรีพิมานวัฒน์, นายอรรถสิทธิ์ หล่าสกุล. "การสร้างสัญญาณรบกวนแบบเกาส์สำหรับทดสอบประสิทธิภาพระบบสื่อสาร". การประชุมประจำปี สวทช.2548

[5] วัชรากร หนูทอง. "FPGA Design Workshop". ศูนย์พัฒนาธุรกิจออกแบบวงจรรวม (TIDI) ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC)

[6] Douglas J Smith. "HDL Chip Design. Madison" , AL, USA: Doone Publication.

[7] Floyd M. Gardner, John D. Baker . "*Simulation Techniques, Models of Communication* Signals". John Wiley & Son Inc., 1997

[8] John G. Proakis. " Digital Communications" . 4th Edition, McGRAW-HULL, 2000.

[9] J.L. Danger et al. "Efficient FPGA implementation of Gaussian noise generator for communication channel emulation", Proc. 7th IEEE Int. Conf. on Elect., Circ. and Syst. (ICECS ' 2K), 2000.

[10] Yongquan Fan, Zilic, Z., "A novel scheme of implementing high speed AWGN communication channel emulators in FPGAs" Circuits and Systems, 2004. ISCAS '04., 2004

[11] A. Ghazel et al., "Design and performance analysis of a high speed AWGN communication channel emulator", Proc. IEEE Paci.c Rim Conf.on Commun. Comput. and Sig. Proc., Vol. 2, pp. 374–377, 2001.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

[12] John G. Proakis . "Masoud Salehi Contemporary communication systems using MATLAB" Thomson-Engineering, 1998.

[13] Dong-U Lee, Wayne Luk, John Villasenor, Peter Y.K. Cheung "A Hardware Gaussian Noise Generator for Channel Code Evaluation" 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'03), 2003

[14] P.P. Chu and R.E. Jones, "Design techniques of FPGA based random number generator", Proc.Military and Aerospace Applications of Prog. Devices and Tech. Conf., 1999.

[15] การสื่อสารแบบอนุกรม. Available URL:http://campus.en.kku.ac.th/coe2004-19/background.html

[16] พอร์ตอนุกรม . Available URL : http://202.8.85.164/~adisak51/page06_1.html

[17] Xilinx Inc. "Vertex™-II Platform FPGAs : Detailed Description." . Available URL : http://www.xilinx.com .

[18] Xilinx Inc. "Xilinx ISE5.1 Software Manuals" . Available URL : http://www.xilinx.com

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก

```
//ชุดทดสอบรหัสแก้ไขข้อผิดพลาด
//มอดูลหลัก
module top3(clk,clk_uart,rstc,RX,TX,snr);
        input clk;
        input clk_uart;
  input rstc;
  input RX;
  output TX;
  output [7:0] snr;
        wire clk,rst,Rate,data_de,code_delay,send_en;
        wire code_out,code_out1,data_delay,uncode_out;
        wire data;
        wire [7:0]  bpsk;
        wire [15:0] noise,noise2,noise3;
        wire [23:0] code_error,uncode_error;
        wire [3:0] snrc;
        wire [1:0] en_data,din;
        wire [3:0] bpsk_data;
assign snr ={Rate,snrc,3'b0};
assign din ={code_out1,code_out};
clkdiv64 clkdiv64(clk,rstc,dclk);
data_generator Data_generator(dclk,rstc,data);
encode1 encode1(dclk,rstc,data,en_data);
bpsk_mod bpsk_mod(dclk,rstc,en_data,bpsk);
bpsk  bpsk_moddata(dclk,rstc,data,bpsk_data);
wgn wgnoise(dclk,rstc,snr,noise,noise2);
wgn2 wgn2(dclk,rstc,snr,noise3);
bqpsk_chan bqpsk_chan10(dclk,rstc,bpsk[3:0],noise,code_out);
bqpsk_chan bqpsk_chan1(dclk,rstc,bpsk[7:4],noise2,code_out1);
bpsk_chan  bpsk_chan(dclk,rstc,bpsk_data,noise3,uncode_out);
decode1 viterbi(dclk,rstc,din,data_de);
Delay Delay(dclk,rstc,data,data_delay);
Delay_dataENCODE Delay_dataENCODE(dclk,rstc,data,code_delay);
error_counter code_Counter(dclk,rst,data_de,code_delay,code_error);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
error_counter uncode_Counter(dclk,rst,uncode_out,data_delay,uncode_error);
data_counter Data_Counter(dclk,rst,sent_en);
Com_interface Com_interface(clk_uart,rstc,sent_en,uncode_error,code_error,RX,TX,Rate,snrc,rst);
Endmodule
```

//มอดูลหารสัญญาณนาฬิกา

```
module clkdiv64(clk,rst,dclk);
  input clk,rst;
  output dclk;
        reg [5:0] count;// 5   2
        assign    dclk = count[5];
always @(negedge clk or posedge rst)
begin
        if (rst)
        count = 0;
        else
        count = count + 1;
end
endmodule
```

//มอดูลกำเนิดข้อมูล

```
module data_generator(clk,rst,data);
        parameter       pol_gen = 13'b0000000011011;
        input clk,rst;  output data;
        reg[12:0]       lfsr_ln;
        assign data   = lfsr_ln[12];
always @(posedge clk or posedge rst)
begin
  if (rst)
                lfsr_ln   = 13'b1111101100000;
        else
    if (lfsr_ln[12] == 0)
                        lfsr_ln= {lfsr_ln[11:0],1'b0};
                else
                        lfsr_ln = {lfsr_ln[11:0],1'b0} ^ pol_gen;
end
endmodule
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

// มอดูลมอดูเลเตอร์แบบ BPSK

```
module bpsk_mod(clk,rst,din,bpsk);
   input clk;
   input rst;
   input [1:0] din;
   output [7:0] bpsk;
          reg [7:0] bpsk;
always @(posedge clk or posedge rst)
begin
  if (rst)
                bpsk <= 0;
        else
        begin
                if (din== 0)
                        bpsk <= 8'b00010001;
         else
                 if (din == 1)
                bpsk <= 8'b00011111;
                else
                if (din == 2)
                        bpsk <= 8'b11110001;
                else
                        bpsk <= 8'b11111111;
                end
        end
endmodule
```

// มอดูลชุดกำเนิดสัญญาณรบกวน (มอดูลหลักของชุดกำเนิดสัญญาณรบกวน)

```
module wgn(clk,rst,snr,x,y);
   input clk,rst;
          input [7:0] snr;
          output [15:0] x,y;
          wire [14:0] u1;
          wire [10:0] u2;
          wire [8:0] snrbp;
          wire [11:0] f;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        wire [9:0] g1,g2;
        wire [15:0] sigma;
        wire [15:0] wx,wy,x1,y1;
        wire [16:0] wgnx,wgny;
        wire [14:0] nx,ny;
        wire sign_cos,sign_wgnc,sign_n;
        wire sx,sy,signx,signy      ;
        assign wx = {signx,nx};
assign wy = {signy,ny};
leapLFSR15 urg1(clk,rst,u1);
leapLFSR11 urg2(clk,rst,u2);
log_rom log_rom(clk,rst,u1,f);
cos_rom2 cos_rom(clk,rst,u2,g1,sign_cos);
sin_rom2 sin_rom(clk,rst,u2,g2,sign_sin);
mul12x10 mul1(clk,rst,f,g1,wgnx);
mul12x10 mul2(clk,rst,f,g2,wgny);
sign_cos_delay delay1(clk,rst,sign_cos,sx);
sign_cos_delay delay2(clk,rst,sign_sin,sy);
snr_romc2 snr_rom(clk,snr,sigma);
mul17x16 mulsnr1(clk,rst,wgnx,sigma,nx);
mul17x16 mulsnr2(clk,rst,wgny,sigma,ny);
comp compx1(clk,rst,wx,x1);
comp2 compx2(clk,rst,x1,x);
comp compy1(clk,rst,wy,y1);
comp2 compy2(clk,rst,y1,y);
sign_cos_delay delay3(clk,rst,sx,signx);
sign_cos_delay delay4(clk,rst,sy,signy);
endmodule
```

//มอดูลกำเนิดตัวเลขสุ่มแบบยูนิฟอร์ม

```
module lfsr(clock,reset,data);
   input clock;
   input reset;
   output [10:0] data;
reg [10:0] data ;
always @(posedge clock or posedge reset)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
begin
  if (reset)
        data <= 11'b100_0000_0000;
        else
  begin
        data[0] <= data[0] ^ data[2];
        data[1] <= data[1] ^ data[3];
        data[2] <= data[2] ^ data[4];
        data[3] <= data[3] ^ data[5];
        data[4] <= data[4] ^ data[6];
        data[5] <= data[5] ^ data[7];
        data[6] <= data[6] ^ data[8];
        data[7] <= data[7] ^ data[9];
        data[8] <= data[8] ^ data[10];
        data[9] <= data[9] ^ data[0] ^ data[2];
        data[10] <= data[10] ^ data[1] ^ data[3];
        end
end
endmodule
module  leapLFSR15(clock,reset,data);
        input           clock;
        input           reset;
        output [14:0]   data;
 reg [14:0] data;
always @(posedge clock or posedge reset)
begin
  if (reset)
        data <= 15'b100000_00000000;
        else
  begin
        data[0] <= data[0] ^ data[1];
        data[1] <= data[1] ^ data[2];
        data[2] <= data[2] ^ data[3];
        data[3] <= data[3] ^ data[4];
        data[4] <= data[4] ^ data[5];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            data[5] <= data[5] ^ data[6];
            data[6] <= data[6] ^ data[7];
            data[7] <= data[7] ^ data[8];
            data[8] <= data[8] ^ data[9];
            data[9] <= data[9] ^ data[10];
            data[10] <= data[10] ^ data[11];
            data[11] <= data[11] ^ data[12];
            data[12] <= data[12] ^ data[13];
            data[13] <= data[13] ^ data[14];
            data[14] <= data[0] ^ data[1] ^ data[14];
            end
end
endmodule
```

// มอดูลประมาณค่าฟังก์ชัน $g2(u_2)$

```
module sin_rom2(clock,reset,addr,data,sign_sin);
   input clock,reset;
   input [10:0] addr;
   output sign_sin;
   output [9:0] data;
            reg sign_sin;
s_rom2 s_rom(clock,addr[9:0],data);
always @(posedge clock or posedge reset)
begin
            if (reset)
            sign_sin <= 0;
                        else
            sign_sin <= addr[10];
end
endmodule
module s_rom2(clk,addr,data);
   input clk;
   input [9:0] addr;
   output [9:0] data;
 reg [9:0] data;
always @(posedge clk)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
begin
        case (addr)
0  : data = 10'b0000000000;
1  : data = 10'b0000000011;
2  : data = 10'b0000000110;
3  : data = 10'b0000001001;
4  : data = 10'b0000001100;
5  : data = 10'b0000001111;
6  : data = 10'b0000010010;
7  : data = 10'b0000010101;
8  : data = 10'b0000011001;
9  : data = 10'b0000011100;
10  : data = 10'b0000011111;
11  : data = 10'b0000100010;
12  : data = 10'b0000100101;
13  : data = 10'b0000101000;
14  : data = 10'b0000101011;
15  : data = 10'b0000101111;
16  : data = 10'b0000110010;
17  : data = 10'b0000110101;
18  : data = 10'b0000111000;
19  : data = 10'b0000111011;
20  : data = 10'b0000111110;
21  : data = 10'b0001000001;
22  : data = 10'b0001000101;
23  : data = 10'b0001001000;
24  : data = 10'b0001001011;
25  : data = 10'b0001001110;
26  : data = 10'b0001010001;
27  : data = 10'b0001010100;
28  : data = 10'b0001010111;
29  : data = 10'b0001011010;
30  : data = 10'b0001011110;
31  : data = 10'b0001100001;
32  : data = 10'b0001100100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
33 : data = 10'b0001100111;
34 : data = 10'b0001101010;
35 : data = 10'b0001101101;
36 : data = 10'b0001110000;
37 : data = 10'b0001110011;
38 : data = 10'b0001110111;
39 : data = 10'b0001111010;
40 : data = 10'b0001111101;
41 : data = 10'b0010000000;
42 : data = 10'b0010000011;
43 : data = 10'b0010000110;
44 : data = 10'b0010001001;
45 : data = 10'b0010001100;
46 : data = 10'b0010010000;
47 : data = 10'b0010010011;
48 : data = 10'b0010010110;
49 : data = 10'b0010011001;
50 : data = 10'b0010011100;
51 : data = 10'b0010011111;
52 : data = 10'b0010100010;
53 : data = 10'b0010100101;
54 : data = 10'b0010101000;
55 : data = 10'b0010101011;
56 : data = 10'b0010101111;
57 : data = 10'b0010110010;
58 : data = 10'b0010110101;
59 : data = 10'b0010111000;
60 : data = 10'b0010111011;
61 : data = 10'b0010111110;
62 : data = 10'b0011000001;
63 : data = 10'b0011000100;
64 : data = 10'b0011000111;
65 : data = 10'b0011001010;
66 : data = 10'b0011001101;
67 : data = 10'b0011010001;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
68 : data = 10'b0011010100;
69 : data = 10'b0011010111;
70 : data = 10'b0011011010;
71 : data = 10'b0011011101;
72 : data = 10'b0011100000;
73 : data = 10'b0011100011;
74 : data = 10'b0011100110;
75 : data = 10'b0011101001;
76 : data = 10'b0011101100;
77 : data = 10'b0011101111;
78 : data = 10'b0011110010;
79 : data = 10'b0011110101;
80 : data = 10'b0011111000;
81 : data = 10'b0011111011;
82 : data = 10'b0011111110;
83 : data = 10'b0100000001;
84 : data = 10'b0100000100;
85 : data = 10'b0100001000;
86 : data = 10'b0100001011;
87 : data = 10'b0100001110;
88 : data = 10'b0100010001;
89 : data = 10'b0100010100;
90 : data = 10'b0100010111;
91 : data = 10'b0100011010;
92 : data = 10'b0100011101;
93 : data = 10'b0100100000;
94 : data = 10'b0100100011;
95 : data = 10'b0100100110;
96 : data = 10'b0100101001;
97 : data = 10'b0100101100;
98 : data = 10'b0100101111;
99 : data = 10'b0100110010;
100 : data = 10'b0100110101;
101 : data = 10'b0100111000;
102 : data = 10'b0100111011;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
103 : data = 10'b0100111110;
104 : data = 10'b0101000001;
105 : data = 10'b0101000100;
106 : data = 10'b0101000111;
107 : data = 10'b0101001010;
108 : data = 10'b0101001101;
109 : data = 10'b0101010000;
110 : data = 10'b0101010011;
111 : data = 10'b0101010110;
112 : data = 10'b0101011000;
113 : data = 10'b0101011011;
114 : data = 10'b0101011110;
115 : data = 10'b0101100001;
116 : data = 10'b0101100100;
117 : data = 10'b0101100111;
118 : data = 10'b0101101010;
119 : data = 10'b0101101101;
120 : data = 10'b0101110000;
121 : data = 10'b0101110011;
122 : data = 10'b0101110110;
123 : data = 10'b0101111001;
124 : data = 10'b0101111100;
125 : data = 10'b0101111111;
126 : data = 10'b0110000010;
127 : data = 10'b0110000100;
128 : data = 10'b0110000111;
129 : data = 10'b0110001010;
130 : data = 10'b0110001101;
131 : data = 10'b0110010000;
132 : data = 10'b0110010011;
133 : data = 10'b0110010110;
134 : data = 10'b0110011001;
135 : data = 10'b0110011100;
136 : data = 10'b0110011110;
137 : data = 10'b0110100001;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
138 : data = 10'b0110100100;
139 : data = 10'b0110100111;
140 : data = 10'b0110101010;
141 : data = 10'b0110101101;
142 : data = 10'b0110110000;
143 : data = 10'b0110110010;
144 : data = 10'b0110110101;
145 : data = 10'b0110111000;
146 : data = 10'b0110111011;
147 : data = 10'b0110111110;
148 : data = 10'b0111000001;
149 : data = 10'b0111000011;
150 : data = 10'b0111000110;
151 : data = 10'b0111001001;
152 : data = 10'b0111001100;
153 : data = 10'b0111001111;
154 : data = 10'b0111010010;
155 : data = 10'b0111010100;
156 : data = 10'b0111010111;
157 : data = 10'b0111011010;
158 : data = 10'b0111011101;
159 : data = 10'b0111011111;
160 : data = 10'b0111100010;
161 : data = 10'b0111100101;
162 : data = 10'b0111101000;
163 : data = 10'b0111101011;
164 : data = 10'b0111101101;
165 : data = 10'b0111110000;
166 : data = 10'b0111110011;
167 : data = 10'b0111110101;
168 : data = 10'b0111111000;
169 : data = 10'b0111111011;
170 : data = 10'b0111111110;
171 : data = 10'b1000000000;
172 : data = 10'b1000000011;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
173 : data = 10'b1000000110;
174 : data = 10'b1000001001;
175 : data = 10'b1000001011;
176 : data = 10'b1000001110;
177 : data = 10'b1000010001;
178 : data = 10'b1000010011;
179 : data = 10'b1000010110;
180 : data = 10'b1000011001;
181 : data = 10'b1000011011;
182 : data = 10'b1000011110;
183 : data = 10'b1000100001;
184 : data = 10'b1000100011;
185 : data = 10'b1000100110;
186 : data = 10'b1000101001;
187 : data = 10'b1000101011;
188 : data = 10'b1000101110;
189 : data = 10'b1000110001;
190 : data = 10'b1000110011;
191 : data = 10'b1000110110;
192 : data = 10'b1000111000;
193 : data = 10'b1000111011;
194 : data = 10'b1000111110;
195 : data = 10'b1001000000;
196 : data = 10'b1001000011;
197 : data = 10'b1001000101;
198 : data = 10'b1001001000;
199 : data = 10'b1001001011;
200 : data = 10'b1001001101;
201 : data = 10'b1001010000;
202 : data = 10'b1001010010;
203 : data = 10'b1001010101;
204 : data = 10'b1001010111;
205 : data = 10'b1001011010;
206 : data = 10'b1001011100;
207 : data = 10'b1001011111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
208  : data = 10'b1001100001;
209  : data = 10'b1001100100;
210  : data = 10'b1001100111;
211  : data = 10'b1001101001;
212  : data = 10'b1001101100;
213  : data = 10'b1001101110;
214  : data = 10'b1001110001;
215  : data = 10'b1001110011;
216  : data = 10'b1001110101;
217  : data = 10'b1001111000;
218  : data = 10'b1001111010;
219  : data = 10'b1001111101;
220  : data = 10'b1001111111;
221  : data = 10'b1010000010;
222  : data = 10'b1010000100;
223  : data = 10'b1010000111;
224  : data = 10'b1010001001;
225  : data = 10'b1010001100;
226  : data = 10'b1010001110;
227  : data = 10'b1010010000;
228  : data = 10'b1010010011;
229  : data = 10'b1010010101;
230  : data = 10'b1010011000;
231  : data = 10'b1010011010;
232  : data = 10'b1010011100;
233  : data = 10'b1010011111;
234  : data = 10'b1010100001;
235  : data = 10'b1010100011;
236  : data = 10'b1010100110;
237  : data = 10'b1010101000;
238  : data = 10'b1010101011;
239  : data = 10'b1010101101;
240  : data = 10'b1010101111;
241  : data = 10'b1010110010;
242  : data = 10'b1010110100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
243 : data = 10'b1010110110;
244 : data = 10'b1010111000;
245 : data = 10'b1010111011;
246 : data = 10'b1010111101;
247 : data = 10'b1010111111;
248 : data = 10'b1011000010;
249 : data = 10'b1011000100;
250 : data = 10'b1011000110;
251 : data = 10'b1011001000;
252 : data = 10'b1011001011;
253 : data = 10'b1011001101;
254 : data = 10'b1011001111;
255 : data = 10'b1011010001;
256 : data = 10'b1011010100;
257 : data = 10'b1011010110;
258 : data = 10'b1011011000;
259 : data = 10'b1011011010;
260 : data = 10'b1011011100;
261 : data = 10'b1011011111;
262 : data = 10'b1011100001;
263 : data = 10'b1011100011;
264 : data = 10'b1011100101;
265 : data = 10'b1011100111;
266 : data = 10'b1011101001;
267 : data = 10'b1011101100;
268 : data = 10'b1011101110;
269 : data = 10'b1011110000;
270 : data = 10'b1011110010;
271 : data = 10'b1011110100;
272 : data = 10'b1011110110;
273 : data = 10'b1011111000;
274 : data = 10'b1011111010;
275 : data = 10'b1011111101;
276 : data = 10'b1011111111;
277 : data = 10'b1100000001;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
278 : data = 10'b1100000011;
279 : data = 10'b1100000101;
280 : data = 10'b1100000111;
281 : data = 10'b1100001001;
282 : data = 10'b1100001011;
283 : data = 10'b1100001101;
284 : data = 10'b1100001111;
285 : data = 10'b1100010001;
286 : data = 10'b1100010011;
287 : data = 10'b1100010101;
288 : data = 10'b1100010111;
289 : data = 10'b1100011001;
290 : data = 10'b1100011011;
291 : data = 10'b1100011101;
292 : data = 10'b1100011111;
293 : data = 10'b1100100001;
294 : data = 10'b1100100011;
295 : data = 10'b1100100101;
296 : data = 10'b1100100111;
297 : data = 10'b1100101001;
298 : data = 10'b1100101011;
299 : data = 10'b1100101101;
300 : data = 10'b1100101110;
301 : data = 10'b1100110000;
302 : data = 10'b1100110010;
303 : data = 10'b1100110100;
304 : data = 10'b1100110110;
305 : data = 10'b1100111000;
306 : data = 10'b1100111010;
307 : data = 10'b1100111100;
308 : data = 10'b1100111101;
309 : data = 10'b1100111111;
310 : data = 10'b1101000001;
311 : data = 10'b1101000011;
312 : data = 10'b1101000101;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
313 : data = 10'b1101000111;
314 : data = 10'b1101001000;
315 : data = 10'b1101001010;
316 : data = 10'b1101001100;
317 : data = 10'b1101001110;
318 : data = 10'b1101001111;
319 : data = 10'b1101010001;
320 : data = 10'b1101010011;
321 : data = 10'b1101010101;
322 : data = 10'b1101010110;
323 : data = 10'b1101011000;
324 : data = 10'b1101011010;
325 : data = 10'b1101011100;
326 : data = 10'b1101011101;
327 : data = 10'b1101011111;
328 : data = 10'b1101100001;
329 : data = 10'b1101100010;
330 : data = 10'b1101100100;
331 : data = 10'b1101100110;
332 : data = 10'b1101100111;
333 : data = 10'b1101101001;
334 : data = 10'b1101101011;
335 : data = 10'b1101101100;
336 : data = 10'b1101101110;
337 : data = 10'b1101101111;
338 : data = 10'b1101110001;
339 : data = 10'b1101110011;
340 : data = 10'b1101110100;
341 : data = 10'b1101110110;
342 : data = 10'b1101110111;
343 : data = 10'b1101111001;
344 : data = 10'b1101111010;
345 : data = 10'b1101111100;
346 : data = 10'b1101111110;
347 : data = 10'b1101111111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
348 : data = 10'b1110000001;
349 : data = 10'b1110000010;
350 : data = 10'b1110000100;
351 : data = 10'b1110000101;
352 : data = 10'b1110000111;
353 : data = 10'b1110001000;
354 : data = 10'b1110001010;
355 : data = 10'b1110001011;
356 : data = 10'b1110001100;
357 : data = 10'b1110001110;
358 : data = 10'b1110001111;
359 : data = 10'b1110010001;
360 : data = 10'b1110010010;
361 : data = 10'b1110010100;
362 : data = 10'b1110010101;
363 : data = 10'b1110010110;
364 : data = 10'b1110011000;
365 : data = 10'b1110011001;
366 : data = 10'b1110011010;
367 : data = 10'b1110011100;
368 : data = 10'b1110011101;
369 : data = 10'b1110011111;
370 : data = 10'b1110100000;
371 : data = 10'b1110100001;
372 : data = 10'b1110100010;
373 : data = 10'b1110100100;
374 : data = 10'b1110100101;
375 : data = 10'b1110100110;
376 : data = 10'b1110101000;
377 : data = 10'b1110101001;
378 : data = 10'b1110101010;
379 : data = 10'b1110101011;
380 : data = 10'b1110101101;
381 : data = 10'b1110101110;
382 : data = 10'b1110101111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
383 : data = 10'b1110110000;
384 : data = 10'b1110110010;
385 : data = 10'b1110110011;
386 : data = 10'b1110110100;
387 : data = 10'b1110110101;
388 : data = 10'b1110110110;
389 : data = 10'b1110110111;
390 : data = 10'b1110111001;
391 : data = 10'b1110111010;
392 : data = 10'b1110111011;
393 : data = 10'b1110111100;
394 : data = 10'b1110111101;
395 : data = 10'b1110111110;
396 : data = 10'b1110111111;
397 : data = 10'b1111000000;
398 : data = 10'b1111000010;
399 : data = 10'b1111000011;
400 : data = 10'b1111000100;
401 : data = 10'b1111000101;
402 : data = 10'b1111000110;
403 : data = 10'b1111000111;
404 : data = 10'b1111001000;
405 : data = 10'b1111001001;
406 : data = 10'b1111001010;
407 : data = 10'b1111001011;
408 : data = 10'b1111001100;
409 : data = 10'b1111001101;
410 : data = 10'b1111001110;
411 : data = 10'b1111001111;
412 : data = 10'b1111010000;
413 : data = 10'b1111010001;
414 : data = 10'b1111010010;
415 : data = 10'b1111010010;
416 : data = 10'b1111010011;
417 : data = 10'b1111010100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
418  : data = 10'b1111010101;
419  : data = 10'b1111010110;
420  : data = 10'b1111010111;
421  : data = 10'b1111011000;
422  : data = 10'b1111011001;
423  : data = 10'b1111011010;
424  : data = 10'b1111011010;
425  : data = 10'b1111011011;
426  : data = 10'b1111011100;
427  : data = 10'b1111011101;
428  : data = 10'b1111011110;
429  : data = 10'b1111011110;
430  : data = 10'b1111011111;
431  : data = 10'b1111100000;
432  : data = 10'b1111100001;
433  : data = 10'b1111100010;
434  : data = 10'b1111100010;
435  : data = 10'b1111100011;
436  : data = 10'b1111100100;
437  : data = 10'b1111100101;
438  : data = 10'b1111100101;
439  : data = 10'b1111100110;
440  : data = 10'b1111100111;
441  : data = 10'b1111100111;
442  : data = 10'b1111101000;
443  : data = 10'b1111101001;
444  : data = 10'b1111101001;
445  : data = 10'b1111101010;
446  : data = 10'b1111101011;
447  : data = 10'b1111101011;
448  : data = 10'b1111101100;
449  : data = 10'b1111101100;
450  : data = 10'b1111101101;
451  : data = 10'b1111101110;
452  : data = 10'b1111101110;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
453 : data = 10'b1111101111;
454 : data = 10'b1111101111;
455 : data = 10'b1111110000;
456 : data = 10'b1111110000;
457 : data = 10'b1111110001;
458 : data = 10'b1111110001;
459 : data = 10'b1111110010;
460 : data = 10'b1111110010;
461 : data = 10'b1111110011;
462 : data = 10'b1111110011;
463 : data = 10'b1111110100;
464 : data = 10'b1111110100;
465 : data = 10'b1111110101;
466 : data = 10'b1111110101;
467 : data = 10'b1111110110;
468 : data = 10'b1111110110;
469 : data = 10'b1111110111;
470 : data = 10'b1111110111;
471 : data = 10'b1111110111;
472 : data = 10'b1111111000;
473 : data = 10'b1111111000;
474 : data = 10'b1111111001;
475 : data = 10'b1111111001;
476 : data = 10'b1111111001;
477 : data = 10'b1111111010;
478 : data = 10'b1111111010;
479 : data = 10'b1111111010;
480 : data = 10'b1111111011;
481 : data = 10'b1111111011;
482 : data = 10'b1111111011;
483 : data = 10'b1111111011;
484 : data = 10'b1111111100;
485 : data = 10'b1111111100;
486 : data = 10'b1111111100;
487 : data = 10'b1111111100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
488 : data = 10'b1111111101;
489 : data = 10'b1111111101;
490 : data = 10'b1111111101;
491 : data = 10'b1111111101;
492 : data = 10'b1111111110;
493 : data = 10'b1111111110;
494 : data = 10'b1111111110;
495 : data = 10'b1111111110;
496 : data = 10'b1111111110;
497 : data = 10'b1111111110;
498 : data = 10'b1111111111;
499 : data = 10'b1111111111;
500 : data = 10'b1111111111;
501 : data = 10'b1111111111;
502 : data = 10'b1111111111;
503 : data = 10'b1111111111;
504 : data = 10'b1111111111;
505 : data = 10'b1111111111;
506 : data = 10'b1111111111;
507 : data = 10'b1111111111;
508 : data = 10'b1111111111;
509 : data = 10'b1111111111;
510 : data = 10'b1111111111;
511 : data = 10'b1111111111;
512 : data = 10'b1111111111;
513 : data = 10'b1111111111;
514 : data = 10'b1111111111;
515 : data = 10'b1111111111;
516 : data = 10'b1111111111;
517 : data = 10'b1111111111;
518 : data = 10'b1111111111;
519 : data = 10'b1111111111;
520 : data = 10'b1111111111;
521 : data = 10'b1111111111;
522 : data = 10'b1111111111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
523 : data = 10'b1111111111;
524 : data = 10'b1111111111;
525 : data = 10'b1111111111;
526 : data = 10'b1111111111;
527 : data = 10'b1111111110;
528 : data = 10'b1111111110;
529 : data = 10'b1111111110;
530 : data = 10'b1111111110;
531 : data = 10'b1111111110;
532 : data = 10'b1111111110;
533 : data = 10'b1111111101;
534 : data = 10'b1111111101;
535 : data = 10'b1111111101;
536 : data = 10'b1111111101;
537 : data = 10'b1111111100;
538 : data = 10'b1111111100;
539 : data = 10'b1111111100;
540 : data = 10'b1111111100;
541 : data = 10'b1111111011;
542 : data = 10'b1111111011;
543 : data = 10'b1111111011;
544 : data = 10'b1111111011;
545 : data = 10'b1111111010;
546 : data = 10'b1111111010;
547 : data = 10'b1111111010;
548 : data = 10'b1111111001;
549 : data = 10'b1111111001;
550 : data = 10'b1111111001;
551 : data = 10'b1111111000;
552 : data = 10'b1111111000;
553 : data = 10'b1111110111;
554 : data = 10'b1111110111;
555 : data = 10'b1111110111;
556 : data = 10'b1111110110;
557 : data = 10'b1111110110;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
558 : data = 10'b1111110101;
559 : data = 10'b1111110101;
560 : data = 10'b1111110100;
561 : data = 10'b1111110100;
562 : data = 10'b1111110011;
563 : data = 10'b1111110011;
564 : data = 10'b1111110010;
565 : data = 10'b1111110010;
566 : data = 10'b1111110001;
567 : data = 10'b1111110001;
568 : data = 10'b1111110000;
569 : data = 10'b1111110000;
570 : data = 10'b1111101111;
571 : data = 10'b1111101111;
572 : data = 10'b1111101110;
573 : data = 10'b1111101110;
574 : data = 10'b1111101101;
575 : data = 10'b1111101100;
576 : data = 10'b1111101100;
577 : data = 10'b1111101011;
578 : data = 10'b1111101011;
579 : data = 10'b1111101010;
580 : data = 10'b1111101001;
581 : data = 10'b1111101001;
582 : data = 10'b1111101000;
583 : data = 10'b1111100111;
584 : data = 10'b1111100111;
585 : data = 10'b1111100110;
586 : data = 10'b1111100101;
587 : data = 10'b1111100101;
588 : data = 10'b1111100100;
589 : data = 10'b1111100011;
590 : data = 10'b1111100010;
591 : data = 10'b1111100010;
592 : data = 10'b1111100001;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
593 : data = 10'b1111100000;
594 : data = 10'b1111011111;
595 : data = 10'b1111011110;
596 : data = 10'b1111011110;
597 : data = 10'b1111011101;
598 : data = 10'b1111011100;
599 : data = 10'b1111011011;
600 : data = 10'b1111011010;
601 : data = 10'b1111011010;
602 : data = 10'b1111011001;
603 : data = 10'b1111011000;
604 : data = 10'b1111010111;
605 : data = 10'b1111010110;
606 : data = 10'b1111010101;
607 : data = 10'b1111010100;
608 : data = 10'b1111010011;
609 : data = 10'b1111010010;
610 : data = 10'b1111010010;
611 : data = 10'b1111010001;
612 : data = 10'b1111010000;
613 : data = 10'b1111001111;
614 : data = 10'b1111001110;
615 : data = 10'b1111001101;
616 : data = 10'b1111001100;
617 : data = 10'b1111001011;
618 : data = 10'b1111001010;
619 : data = 10'b1111001001;
620 : data = 10'b1111001000;
621 : data = 10'b1111000111;
622 : data = 10'b1111000110;
623 : data = 10'b1111000101;
624 : data = 10'b1111000100;
625 : data = 10'b1111000011;
626 : data = 10'b1111000010;
627 : data = 10'b1111000000;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
628 : data = 10'b1110111111;
629 : data = 10'b1110111110;
630 : data = 10'b1110111101;
631 : data = 10'b1110111100;
632 : data = 10'b1110111011;
633 : data = 10'b1110111010;
634 : data = 10'b1110111001;
635 : data = 10'b1110110111;
636 : data = 10'b1110110110;
637 : data = 10'b1110110101;
638 : data = 10'b1110110100;
639 : data = 10'b1110110011;
640 : data = 10'b1110110010;
641 : data = 10'b1110110000;
642 : data = 10'b1110101111;
643 : data = 10'b1110101110;
644 : data = 10'b1110101101;
645 : data = 10'b1110101011;
646 : data = 10'b1110101010;
647 : data = 10'b1110101001;
648 : data = 10'b1110101000;
649 : data = 10'b1110100110;
650 : data = 10'b1110100101;
651 : data = 10'b1110100100;
652 : data = 10'b1110100010;
653 : data = 10'b1110100001;
654 : data = 10'b1110100000;
655 : data = 10'b1110011111;
656 : data = 10'b1110011101;
657 : data = 10'b1110011100;
658 : data = 10'b1110011010;
659 : data = 10'b1110011001;
660 : data = 10'b1110011000;
661 : data = 10'b1110010110;
662 : data = 10'b1110010101;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
663 : data = 10'b1110010100;
664 : data = 10'b1110010010;
665 : data = 10'b1110010001;
666 : data = 10'b1110001111;
667 : data = 10'b1110001110;
668 : data = 10'b1110001100;
669 : data = 10'b1110001011;
670 : data = 10'b1110001010;
671 : data = 10'b1110001000;
672 : data = 10'b1110000111;
673 : data = 10'b1110000101;
674 : data = 10'b1110000100;
675 : data = 10'b1110000010;
676 : data = 10'b1110000001;
677 : data = 10'b1101111111;
678 : data = 10'b1101111110;
679 : data = 10'b1101111100;
680 : data = 10'b1101111010;
681 : data = 10'b1101111001;
682 : data = 10'b1101110111;
683 : data = 10'b1101110110;
684 : data = 10'b1101110100;
685 : data = 10'b1101110011;
686 : data = 10'b1101110001;
687 : data = 10'b1101101111;
688 : data = 10'b1101101110;
689 : data = 10'b1101101100;
690 : data = 10'b1101101011;
691 : data = 10'b1101101001;
692 : data = 10'b1101100111;
693 : data = 10'b1101100110;
694 : data = 10'b1101100100;
695 : data = 10'b1101100010;
696 : data = 10'b1101100001;
697 : data = 10'b1101011111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
698  : data = 10'b1101011101;
699  : data = 10'b1101011100;
700  : data = 10'b1101011010;
701  : data = 10'b1101011000;
702  : data = 10'b1101010110;
703  : data = 10'b1101010101;
704  : data = 10'b1101010011;
705  : data = 10'b1101010001;
706  : data = 10'b1101001111;
707  : data = 10'b1101001110;
708  : data = 10'b1101001100;
709  : data = 10'b1101001010;
710  : data = 10'b1101001000;
711  : data = 10'b1101000111;
712  : data = 10'b1101000101;
713  : data = 10'b1101000011;
714  : data = 10'b1101000001;
715  : data = 10'b1100111111;
716  : data = 10'b1100111101;
717  : data = 10'b1100111100;
718  : data = 10'b1100111010;
719  : data = 10'b1100111000;
720  : data = 10'b1100110110;
721  : data = 10'b1100110100;
722  : data = 10'b1100110010;
723  : data = 10'b1100110000;
724  : data = 10'b1100101110;
725  : data = 10'b1100101101;
726  : data = 10'b1100101011;
727  : data = 10'b1100101001;
728  : data = 10'b1100100111;
729  : data = 10'b1100100101;
730  : data = 10'b1100100011;
731  : data = 10'b1100100001;
732  : data = 10'b1100011111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
733 : data = 10'b1100011101;
734 : data = 10'b1100011011;
735 : data = 10'b1100011001;
736 : data = 10'b1100010111;
737 : data = 10'b1100010101;
738 : data = 10'b1100010011;
739 : data = 10'b1100010001;
740 : data = 10'b1100001111;
741 : data = 10'b1100001101;
742 : data = 10'b1100001011;
743 : data = 10'b1100001001;
744 : data = 10'b1100000111;
745 : data = 10'b1100000101;
746 : data = 10'b1100000011;
747 : data = 10'b1100000001;
748 : data = 10'b1011111111;
749 : data = 10'b1011111101;
750 : data = 10'b1011111010;
751 : data = 10'b1011111000;
752 : data = 10'b1011110110;
753 : data = 10'b1011110100;
754 : data = 10'b1011110010;
755 : data = 10'b1011110000;
756 : data = 10'b1011101110;
757 : data = 10'b1011101100;
758 : data = 10'b1011101001;
759 : data = 10'b1011100111;
760 : data = 10'b1011100101;
761 : data = 10'b1011100011;
762 : data = 10'b1011100001;
763 : data = 10'b1011011111;
764 : data = 10'b1011011100;
765 : data = 10'b1011011010;
766 : data = 10'b1011011000;
767 : data = 10'b1011010110;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
768 : data = 10'b1011010100;
769 : data = 10'b1011010001;
770 : data = 10'b1011001111;
771 : data = 10'b1011001101;
772 : data = 10'b1011001011;
773 : data = 10'b1011001000;
774 : data = 10'b1011000110;
775 : data = 10'b1011000100;
776 : data = 10'b1011000010;
777 : data = 10'b1010111111;
778 : data = 10'b1010111101;
779 : data = 10'b1010111011;
780 : data = 10'b1010111000;
781 : data = 10'b1010110110;
782 : data = 10'b1010110100;
783 : data = 10'b1010110010;
784 : data = 10'b1010101111;
785 : data = 10'b1010101101;
786 : data = 10'b1010101011;
787 : data = 10'b1010101000;
788 : data = 10'b1010100110;
789 : data = 10'b1010100011;
790 : data = 10'b1010100001;
791 : data = 10'b1010011111;
792 : data = 10'b1010011100;
793 : data = 10'b1010011010;
794 : data = 10'b1010011000;
795 : data = 10'b1010010101;
796 : data = 10'b1010010011;
797 : data = 10'b1010010000;
798 : data = 10'b1010001110;
799 : data = 10'b1010001100;
800 : data = 10'b1010001001;
801 : data = 10'b1010000111;
802 : data = 10'b1010000100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
803 : data = 10'b1010000010;
804 : data = 10'b1001111111;
805 : data = 10'b1001111101;
806 : data = 10'b1001111010;
807 : data = 10'b1001111000;
808 : data = 10'b1001110101;
809 : data = 10'b1001110011;
810 : data = 10'b1001110001;
811 : data = 10'b1001101110;
812 : data = 10'b1001101100;
813 : data = 10'b1001101001;
814 : data = 10'b1001100111;
815 : data = 10'b1001100100;
816 : data = 10'b1001100001;
817 : data = 10'b1001011111;
818 : data = 10'b1001011100;
819 : data = 10'b1001011010;
820 : data = 10'b1001010111;
821 : data = 10'b1001010101;
822 : data = 10'b1001010010;
823 : data = 10'b1001010000;
824 : data = 10'b1001001101;
825 : data = 10'b1001001011;
826 : data = 10'b1001001000;
827 : data = 10'b1001000101;
828 : data = 10'b1001000011;
829 : data = 10'b1001000000;
830 : data = 10'b1000111110;
831 : data = 10'b1000111011;
832 : data = 10'b1000111000;
833 : data = 10'b1000110110;
834 : data = 10'b1000110011;
835 : data = 10'b1000110001;
836 : data = 10'b1000101110;
837 : data = 10'b1000101011;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
838 : data = 10'b1000101001;
839 : data = 10'b1000100110;
840 : data = 10'b1000100011;
841 : data = 10'b1000100001;
842 : data = 10'b1000011110;
843 : data = 10'b1000011011;
844 : data = 10'b1000011001;
845 : data = 10'b1000010110;
846 : data = 10'b1000010011;
847 : data = 10'b1000010001;
848 : data = 10'b1000001110;
849 : data = 10'b1000001011;
850 : data = 10'b1000001001;
851 : data = 10'b1000000110;
852 : data = 10'b1000000011;
853 : data = 10'b1000000000;
854 : data = 10'b0111111110;
855 : data = 10'b0111111011;
856 : data = 10'b0111111000;
857 : data = 10'b0111110101;
858 : data = 10'b0111110011;
859 : data = 10'b0111110000;
860 : data = 10'b0111101101;
861 : data = 10'b0111101011;
862 : data = 10'b0111101000;
863 : data = 10'b0111100101;
864 : data = 10'b0111100010;
865 : data = 10'b0111011111;
866 : data = 10'b0111011101;
867 : data = 10'b0111011010;
868 : data = 10'b0111010111;
869 : data = 10'b0111010100;
870 : data = 10'b0111010010;
871 : data = 10'b0111001111;
872 : data = 10'b0111001100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
873  : data = 10'b0111001001;
874  : data = 10'b0111000110;
875  : data = 10'b0111000011;
876  : data = 10'b0111000001;
877  : data = 10'b0110111110;
878  : data = 10'b0110111011;
879  : data = 10'b0110111000;
880  : data = 10'b0110110101;
881  : data = 10'b0110110010;
882  : data = 10'b0110110000;
883  : data = 10'b0110101101;
884  : data = 10'b0110101010;
885  : data = 10'b0110100111;
886  : data = 10'b0110100100;
887  : data = 10'b0110100001;
888  : data = 10'b0110011110;
889  : data = 10'b0110011100;
890  : data = 10'b0110011001;
891  : data = 10'b0110010110;
892  : data = 10'b0110010011;
893  : data = 10'b0110010000;
894  : data = 10'b0110001101;
895  : data = 10'b0110001010;
896  : data = 10'b0110000111;
897  : data = 10'b0110000100;
898  : data = 10'b0110000010;
899  : data = 10'b0101111111;
900  : data = 10'b0101111100;
901  : data = 10'b0101111001;
902  : data = 10'b0101110110;
903  : data = 10'b0101110011;
904  : data = 10'b0101110000;
905  : data = 10'b0101101101;
906  : data = 10'b0101101010;
907  : data = 10'b0101100111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
908 : data = 10'b0101100100;
909 : data = 10'b0101100001;
910 : data = 10'b0101011110;
911 : data = 10'b0101011011;
912 : data = 10'b0101011000;
913 : data = 10'b0101010110;
914 : data = 10'b0101010011;
915 : data = 10'b0101010000;
916 : data = 10'b0101001101;
917 : data = 10'b0101001010;
918 : data = 10'b0101000111;
919 : data = 10'b0101000100;
920 : data = 10'b0101000001;
921 : data = 10'b0100111110;
922 : data = 10'b0100111011;
923 : data = 10'b0100111000;
924 : data = 10'b0100110101;
925 : data = 10'b0100110010;
926 : data = 10'b0100101111;
927 : data = 10'b0100101100;
928 : data = 10'b0100101001;
929 : data = 10'b0100100110;
930 : data = 10'b0100100011;
931 : data = 10'b0100100000;
932 : data = 10'b0100011101;
933 : data = 10'b0100011010;
934 : data = 10'b0100010111;
935 : data = 10'b0100010100;
936 : data = 10'b0100010001;
937 : data = 10'b0100001110;
938 : data = 10'b0100001011;
939 : data = 10'b0100001000;
940 : data = 10'b0100000100;
941 : data = 10'b0100000001;
942 : data = 10'b0011111110;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
943 : data = 10'b0011111011;
944 : data = 10'b0011111000;
945 : data = 10'b0011110101;
946 : data = 10'b0011110010;
947 : data = 10'b0011101111;
948 : data = 10'b0011101100;
949 : data = 10'b0011101001;
950 : data = 10'b0011100110;
951 : data = 10'b0011100011;
952 : data = 10'b0011100000;
953 : data = 10'b0011011101;
954 : data = 10'b0011011010;
955 : data = 10'b0011010111;
956 : data = 10'b0011010100;
957 : data = 10'b0011010001;
958 : data = 10'b0011001101;
959 : data = 10'b0011001010;
960 : data = 10'b0011000111;
961 : data = 10'b0011000100;
962 : data = 10'b0011000001;
963 : data = 10'b0010111110;
964 : data = 10'b0010111011;
965 : data = 10'b0010111000;
966 : data = 10'b0010110101;
967 : data = 10'b0010110010;
968 : data = 10'b0010101111;
969 : data = 10'b0010101011;
970 : data = 10'b0010101000;
971 : data = 10'b0010100101;
972 : data = 10'b0010100010;
973 : data = 10'b0010011111;
974 : data = 10'b0010011100;
975 : data = 10'b0010011001;
976 : data = 10'b0010010110;
977 : data = 10'b0010010011;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
978  : data = 10'b0010010000;
979  : data = 10'b0010001100;
980  : data = 10'b0010001001;
981  : data = 10'b0010000110;
982  : data = 10'b0010000011;
983  : data = 10'b0010000000;
984  : data = 10'b0001111101;
985  : data = 10'b0001111010;
986  : data = 10'b0001110111;
987  : data = 10'b0001110011;
988  : data = 10'b0001110000;
989  : data = 10'b0001101101;
990  : data = 10'b0001101010;
991  : data = 10'b0001100111;
992  : data = 10'b0001100100;
993  : data = 10'b0001100001;
994  : data = 10'b0001011110;
995  : data = 10'b0001011010;
996  : data = 10'b0001010111;
997  : data = 10'b0001010100;
998  : data = 10'b0001010001;
999  : data = 10'b0001001110;
1000  : data = 10'b0001001011;
1001  : data = 10'b0001001000;
1002  : data = 10'b0001000101;
1003  : data = 10'b0001000001;
1004  : data = 10'b0000111110;
1005  : data = 10'b0000111011;
1006  : data = 10'b0000111000;
1007  : data = 10'b0000110101;
1008  : data = 10'b0000110010;
1009  : data = 10'b0000101111;
1010  : data = 10'b0000101011;
1011  : data = 10'b0000101000;
1012  : data = 10'b0000100101;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
1013  : data = 10'b0000100010;
1014  : data = 10'b0000011111;
1015  : data = 10'b0000011100;
1016  : data = 10'b0000011001;
1017  : data = 10'b0000010101;
1018  : data = 10'b0000010010;
1019  : data = 10'b0000001111;
1020  : data = 10'b0000001100;
1021  : data = 10'b0000001001;
1022  : data = 10'b0000000110;
1023  : data = 10'b0000000011;
default : data = 10'bx;
  endcase
end
endmodule
// มอดูลประมาณค่าฟังก์ชัน g1(u2)
module cos_rom2(clock,reset,addr,data,sign_cos);
   input clock,reset;
   input [10:0] addr;
   output sign_cos;
   output [9:0] data;
          reg sign_cos;
c_rom2   c_rom (clock,addr[9:0],data);

always @(posedge clock or posedge reset)
begin
          if (reset)
          sign_cos <= 0;
 else
          sign_cos <= addr[10] ^ addr[9];
end
endmodule
module c_rom2(clock,addr,data);
   input clock;
   input [9:0] addr;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  output [9:0] data;
        //1 bit integer and 15f
reg [9:0] data;
always @(posedge clock)
begin
        case (addr)
0  : data = 10'b1111111111;
1  : data =  10'b1111111111;
2  : data = 10'b1111111111;
3  : data = 10'b1111111111;
4  : data = 10'b1111111111;
5  : data = 10'b1111111111;
6  : data = 10'b1111111111;
7  : data = 10'b1111111111;
8  : data = 10'b1111111111;
9  : data = 10'b1111111111;
10  : data = 10'b1111111111;
11  : data = 10'b1111111111;
12  : data = 10'b1111111111;
13  : data = 10'b1111111111;
14  : data = 10'b1111111111;
15  : data = 10'b1111111110;
16  : data = 10'b1111111110;
17  : data = 10'b1111111110;
18  : data = 10'b1111111110;
19  : data = 10'b1111111110;
20  : data = 10'b1111111110;
21  : data = 10'b1111111101;
22  : data = 10'b1111111101;
23  : data = 10'b1111111101;
24  : data = 10'b1111111101;
25  : data = 10'b1111111100;
26  : data = 10'b1111111100;
27  : data = 10'b1111111100;
28  : data = 10'b1111111100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
29 : data = 10'b1111111011;
30 : data = 10'b1111111011;
31 : data = 10'b1111111011;
32 : data = 10'b1111111011;
33 : data = 10'b1111111010;
34 : data = 10'b1111111010;
35 : data = 10'b1111111010;
36 : data = 10'b1111111001;
37 : data = 10'b1111111001;
38 : data = 10'b1111111001;
39 : data = 10'b1111111000;
40 : data = 10'b1111111000;
41 : data = 10'b1111110111;
42 : data = 10'b1111110111;
43 : data = 10'b1111110111;
44 : data = 10'b1111110110;
45 : data = 10'b1111110110;
46 : data = 10'b1111110101;
47 : data = 10'b1111110101;
48 : data = 10'b1111110100;
49 : data = 10'b1111110100;
50 : data = 10'b1111110011;
51 : data = 10'b1111110011;
52 : data = 10'b1111110010;
53 : data = 10'b1111110010;
54 : data = 10'b1111110001;
55 : data = 10'b1111110001;
56 : data = 10'b1111110000;
57 : data = 10'b1111110000;
58 : data = 10'b1111101111;
59 : data = 10'b1111101111;
60 : data = 10'b1111101110;
61 : data = 10'b1111101110;
62 : data = 10'b1111101101;
63 : data = 10'b1111101100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
64 : data = 10'b1111101100;
65 : data = 10'b1111101011;
66 : data = 10'b1111101011;
67 : data = 10'b1111101010;
68 : data = 10'b1111101001;
69 : data = 10'b1111101001;
70 : data = 10'b1111101000;
71 : data = 10'b1111100111;
72 : data = 10'b1111100111;
73 : data = 10'b1111100110;
74 : data = 10'b1111100101;
75 : data = 10'b1111100101;
76 : data = 10'b1111100100;
77 : data = 10'b1111100011;
78 : data = 10'b1111100010;
79 : data = 10'b1111100010;
80 : data = 10'b1111100001;
81 : data = 10'b1111100000;
82 : data = 10'b1111011111;
83 : data = 10'b1111011110;
84 : data = 10'b1111011110;
85 : data = 10'b1111011101;
86 : data = 10'b1111011100;
87 : data = 10'b1111011011;
88 : data = 10'b1111011010;
89 : data = 10'b1111011010;
90 : data = 10'b1111011001;
91 : data = 10'b1111011000;
92 : data = 10'b1111010111;
93 : data = 10'b1111010110;
94 : data = 10'b1111010101;
95 : data = 10'b1111010100;
96 : data = 10'b1111010011;
97 : data = 10'b1111010010;
98 : data = 10'b1111010010;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
99  : data = 10'b1111010001;
100 : data = 10'b1111010000;
101 : data = 10'b1111001111;
102 : data = 10'b1111001110;
103 : data = 10'b1111001101;
104 : data = 10'b1111001100;
105 : data = 10'b1111001011;
106 : data = 10'b1111001010;
107 : data = 10'b1111001001;
108 : data = 10'b1111001000;
109 : data = 10'b1111000111;
110 : data = 10'b1111000110;
111 : data = 10'b1111000101;
112 : data = 10'b1111000100;
113 : data = 10'b1111000011;
114 : data = 10'b1111000010;
115 : data = 10'b1111000000;
116 : data = 10'b1110111111;
117 : data = 10'b1110111110;
118 : data = 10'b1110111101;
119 : data = 10'b1110111100;
120 : data = 10'b1110111011;
121 : data = 10'b1110111010;
122 : data = 10'b1110111001;
123 : data = 10'b1110110111;
124 : data = 10'b1110110110;
125 : data = 10'b1110110101;
126 : data = 10'b1110110100;
127 : data = 10'b1110110011;
128 : data = 10'b1110110010;
129 : data = 10'b1110110000;
130 : data = 10'b1110101111;
131 : data = 10'b1110101110;
132 : data = 10'b1110101101;
133 : data = 10'b1110101011;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
134 : data = 10'b1110101010;
135 : data = 10'b1110101001;
136 : data = 10'b1110101000;
137 : data = 10'b1110100110;
138 : data = 10'b1110100101;
139 : data = 10'b1110100100;
140 : data = 10'b1110100010;
141 : data = 10'b1110100001;
142 : data = 10'b1110100000;
143 : data = 10'b1110011111;
144 : data = 10'b1110011101;
145 : data = 10'b1110011100;
146 : data = 10'b1110011010;
147 : data = 10'b1110011001;
148 : data = 10'b1110011000;
149 : data = 10'b1110010110;
150 : data = 10'b1110010101;
151 : data = 10'b1110010100;
152 : data = 10'b1110010010;
153 : data = 10'b1110010001;
154 : data = 10'b1110001111;
155 : data = 10'b1110001110;
156 : data = 10'b1110001100;
157 : data = 10'b1110001011;
158 : data = 10'b1110001010;
159 : data = 10'b1110001000;
160 : data = 10'b1110000111;
161 : data = 10'b1110000101;
162 : data = 10'b1110000100;
163 : data = 10'b1110000010;
164 : data = 10'b1110000001;
165 : data = 10'b1101111111;
166 : data = 10'b1101111110;
167 : data = 10'b1101111100;
168 : data = 10'b1101111010;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
169 : data = 10'b1101111001;
170 : data = 10'b1101110111;
171 : data = 10'b1101110110;
172 : data = 10'b1101110100;
173 : data = 10'b1101110011;
174 : data = 10'b1101110001;
175 : data = 10'b1101101111;
176 : data = 10'b1101101110;
177 : data = 10'b1101101100;
178 : data = 10'b1101101011;
179 : data = 10'b1101101001;
180 : data = 10'b1101100111;
181 : data = 10'b1101100110;
182 : data = 10'b1101100100;
183 : data = 10'b1101100010;
184 : data = 10'b1101100001;
185 : data = 10'b1101011111;
186 : data = 10'b1101011101;
187 : data = 10'b1101011100;
188 : data = 10'b1101011010;
189 : data = 10'b1101011000;
190 : data = 10'b1101010110;
191 : data = 10'b1101010101;
192 : data = 10'b1101010011;
193 : data = 10'b1101010001;
194 : data = 10'b1101001111;
195 : data = 10'b1101001110;
196 : data = 10'b1101001100;
197 : data = 10'b1101001010;
198 : data = 10'b1101001000;
199 : data = 10'b1101000111;
200 : data = 10'b1101000101;
201 : data = 10'b1101000011;
202 : data = 10'b1101000001;
203 : data = 10'b1100111111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
204 : data = 10'b1100111101;
205 : data = 10'b1100111100;
206 : data = 10'b1100111010;
207 : data = 10'b1100111000;
208 : data = 10'b1100110110;
209 : data = 10'b1100110100;
210 : data = 10'b1100110010;
211 : data = 10'b1100110000;
212 : data = 10'b1100101110;
213 : data = 10'b1100101101;
214 : data = 10'b1100101011;
215 : data = 10'b1100101001;
216 : data = 10'b1100100111;
217 : data = 10'b1100100101;
218 : data = 10'b1100100011;
219 : data = 10'b1100100001;
220 : data = 10'b1100011111;
221 : data = 10'b1100011101;
222 : data = 10'b1100011011;
223 : data = 10'b1100011001;
224 : data = 10'b1100010111;
225 : data = 10'b1100010101;
226 : data = 10'b1100010011;
227 : data = 10'b1100010001;
228 : data = 10'b1100001111;
229 : data = 10'b1100001101;
230 : data = 10'b1100001011;
231 : data = 10'b1100001001;
232 : data = 10'b1100000111;
233 : data = 10'b1100000101;
234 : data = 10'b1100000011;
235 : data = 10'b1100000001;
236 : data = 10'b1011111111;
237 : data = 10'b1011111101;
238 : data = 10'b1011111010;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
239 : data = 10'b1011111000;
240 : data = 10'b1011110110;
241 : data = 10'b1011110100;
242 : data = 10'b1011110010;
243 : data = 10'b1011110000;
244 : data = 10'b1011101110;
245 : data = 10'b1011101100;
246 : data = 10'b1011101001;
247 : data = 10'b1011100111;
248 : data = 10'b1011100101;
249 : data = 10'b1011100011;
250 : data = 10'b1011100001;
251 : data = 10'b1011011111;
252 : data = 10'b1011011100;
253 : data = 10'b1011011010;
254 : data = 10'b1011011000;
255 : data = 10'b1011010110;
256 : data = 10'b1011010100;
257 : data = 10'b1011010001;
258 : data = 10'b1011001111;
259 : data = 10'b1011001101;
260 : data = 10'b1011001011;
261 : data = 10'b1011001000;
262 : data = 10'b1011000110;
263 : data = 10'b1011000100;
264 : data = 10'b1011000010;
265 : data = 10'b1010111111;
266 : data = 10'b1010111101;
267 : data = 10'b1010111011;
268 : data = 10'b1010111000;
269 : data = 10'b1010110110;
270 : data = 10'b1010110100;
271 : data = 10'b1010110010;
272 : data = 10'b1010101111;
273 : data = 10'b1010101101;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
274 : data = 10'b1010101011;
275 : data = 10'b1010101000;
276 : data = 10'b1010100110;
277 : data = 10'b1010100011;
278 : data = 10'b1010100001;
279 : data = 10'b1010011111;
280 : data = 10'b1010011100;
281 : data = 10'b1010011010;
282 : data = 10'b1010011000;
283 : data = 10'b1010010101;
284 : data = 10'b1010010011;
285 : data = 10'b1010010000;
286 : data = 10'b1010001110;
287 : data = 10'b1010001100;
288 : data = 10'b1010001001;
289 : data = 10'b1010000111;
290 : data = 10'b1010000100;
291 : data = 10'b1010000010;
292 : data = 10'b1001111111;
293 : data = 10'b1001111101;
294 : data = 10'b1001111010;
295 : data = 10'b1001111000;
296 : data = 10'b1001110101;
297 : data = 10'b1001110011;
298 : data = 10'b1001110001;
299 : data = 10'b1001101110;
300 : data = 10'b1001101100;
301 : data = 10'b1001101001;
302 : data = 10'b1001100111;
303 : data = 10'b1001100100;
304 : data = 10'b1001100001;
305 : data = 10'b1001011111;
306 : data = 10'b1001011100;
307 : data = 10'b1001011010;
308 : data = 10'b1001010111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
309 : data = 10'b1001010101;
310 : data = 10'b1001010010;
311 : data = 10'b1001010000;
312 : data = 10'b1001001101;
313 : data = 10'b1001001011;
314 : data = 10'b1001001000;
315 : data = 10'b1001000101;
316 : data = 10'b1001000011;
317 : data = 10'b1001000000;
318 : data = 10'b1000111110;
319 : data = 10'b1000111011;
320 : data = 10'b1000111000;
321 : data = 10'b1000110110;
322 : data = 10'b1000110011;
323 : data = 10'b1000110001;
324 : data = 10'b1000101110;
325 : data = 10'b1000101011;
326 : data = 10'b1000101001;
327 : data = 10'b1000100110;
328 : data = 10'b1000100011;
329 : data = 10'b1000100001;
330 : data = 10'b1000011110;
331 : data = 10'b1000011011;
332 : data = 10'b1000011001;
333 : data = 10'b1000010110;
334 : data = 10'b1000010011;
335 : data = 10'b1000010001;
336 : data = 10'b1000001110;
337 : data = 10'b1000001011;
338 : data = 10'b1000001001;
339 : data = 10'b1000000110;
340 : data = 10'b1000000011;
341 : data = 10'b1000000000;
342 : data = 10'b0111111110;
343 : data = 10'b0111111011;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
344 : data = 10'b0111111000;
345 : data = 10'b0111110101;
346 : data = 10'b0111110011;
347 : data = 10'b0111110000;
348 : data = 10'b0111101101;
349 : data = 10'b0111101011;
350 : data = 10'b0111101000;
351 : data = 10'b0111100101;
352 : data = 10'b0111100010;
353 : data = 10'b0111011111;
354 : data = 10'b0111011101;
355 : data = 10'b0111011010;
356 : data = 10'b0111010111;
357 : data = 10'b0111010100;
358 : data = 10'b0111010010;
359 : data = 10'b0111001111;
360 : data = 10'b0111001100;
361 : data = 10'b0111001001;
362 : data = 10'b0111000110;
363 : data = 10'b0111000011;
364 : data = 10'b0111000001;
365 : data = 10'b0110111110;
366 : data = 10'b0110111011;
367 : data = 10'b0110111000;
368 : data = 10'b0110110101;
369 : data = 10'b0110110010;
370 : data = 10'b0110110000;
371 : data = 10'b0110101101;
372 : data = 10'b0110101010;
373 : data = 10'b0110100111;
374 : data = 10'b0110100100;
375 : data = 10'b0110100001;
376 : data = 10'b0110011110;
377 : data = 10'b0110011100;
378 : data = 10'b0110011001;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
379 : data = 10'b0110010110;
380 : data = 10'b0110010011;
381 : data = 10'b0110010000;
382 : data = 10'b0110001101;
383 : data = 10'b0110001010;
384 : data = 10'b0110000111;
385 : data = 10'b0110000100;
386 : data = 10'b0110000010;
387 : data = 10'b0101111111;
388 : data = 10'b0101111100;
389 : data = 10'b0101111001;
390 : data = 10'b0101110110;
391 : data = 10'b0101110011;
392 : data = 10'b0101110000;
393 : data = 10'b0101101101;
394 : data = 10'b0101101010;
395 : data = 10'b0101100111;
396 : data = 10'b0101100100;
397 : data = 10'b0101100001;
398 : data = 10'b0101011110;
399 : data = 10'b0101011011;
400 : data = 10'b0101011000;
401 : data = 10'b0101010110;
402 : data = 10'b0101010011;
403 : data = 10'b0101010000;
404 : data = 10'b0101001101;
405 : data = 10'b0101001010;
406 : data = 10'b0101000111;
407 : data = 10'b0101000100;
408 : data = 10'b0101000001;
409 : data = 10'b0100111110;
410 : data = 10'b0100111011;
411 : data = 10'b0100111000;
412 : data = 10'b0100110101;
413 : data = 10'b0100110010;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
414 : data = 10'b0100101111;
415 : data = 10'b0100101100;
416 : data = 10'b0100101001;
417 : data = 10'b0100100110;
418 : data = 10'b0100100011;
419 : data = 10'b0100100000;
420 : data = 10'b0100011101;
421 : data = 10'b0100011010;
422 : data = 10'b0100010111;
423 : data = 10'b0100010100;
424 : data = 10'b0100010001;
425 : data = 10'b0100001110;
426 : data = 10'b0100001011;
427 : data = 10'b0100001000;
428 : data = 10'b0100000100;
429 : data = 10'b0100000001;
430 : data = 10'b0011111110;
431 : data = 10'b0011111011;
432 : data = 10'b0011111000;
433 : data = 10'b0011110101;
434 : data = 10'b0011110010;
435 : data = 10'b0011101111;
436 : data = 10'b0011101100;
437 : data = 10'b0011101001;
438 : data = 10'b0011100110;
439 : data = 10'b0011100011;
440 : data = 10'b0011100000;
441 : data = 10'b0011011101;
442 : data = 10'b0011011010;
443 : data = 10'b0011010111;
444 : data = 10'b0011010100;
445 : data = 10'b0011010001;
446 : data = 10'b0011001101;
447 : data = 10'b0011001010;
448 : data = 10'b0011000111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
449 : data = 10'b0011000100;
450 : data = 10'b0011000001;
451 : data = 10'b0010111110;
452 : data = 10'b0010111011;
453 : data = 10'b0010111000;
454 : data = 10'b0010110101;
455 : data = 10'b0010110010;
456 : data = 10'b0010101111;
457 : data = 10'b0010101011;
458 : data = 10'b0010101000;
459 : data = 10'b0010100101;
460 : data = 10'b0010100010;
461 : data = 10'b0010011111;
462 : data = 10'b0010011100;
463 : data = 10'b0010011001;
464 : data = 10'b0010010110;
465 : data = 10'b0010010011;
466 : data = 10'b0010010000;
467 : data = 10'b0010001100;
468 : data = 10'b0010001001;
469 : data = 10'b0010000110;
470 : data = 10'b0010000011;
471 : data = 10'b0010000000;
472 : data = 10'b0001111101;
473 : data = 10'b0001111010;
474 : data = 10'b0001110111;
475 : data = 10'b0001110011;
476 : data = 10'b0001110000;
477 : data = 10'b0001101101;
478 : data = 10'b0001101010;
479 : data = 10'b0001100111;
480 : data = 10'b0001100100;
481 : data = 10'b0001100001;
482 : data = 10'b0001011110;
483 : data = 10'b0001011010;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
484 : data = 10'b0001010111;
485 : data = 10'b0001010100;
486 : data = 10'b0001010001;
487 : data = 10'b0001001110;
488 : data = 10'b0001001011;
489 : data = 10'b0001001000;
490 : data = 10'b0001000101;
491 : data = 10'b0001000001;
492 : data = 10'b0000111110;
493 : data = 10'b0000111011;
494 : data = 10'b0000111000;
495 : data = 10'b0000110101;
496 : data = 10'b0000110010;
497 : data = 10'b0000101111;
498 : data = 10'b0000101011;
499 : data = 10'b0000101000;
500 : data = 10'b0000100101;
501 : data = 10'b0000100010;
502 : data = 10'b0000011111;
503 : data = 10'b0000011100;
504 : data = 10'b0000011001;
505 : data = 10'b0000010101;
506 : data = 10'b0000010010;
507 : data = 10'b0000001111;
508 : data = 10'b0000001100;
509 : data = 10'b0000001001;
510 : data = 10'b0000000110;
511 : data = 10'b0000000011;
512 : data = 10'b0000000000;
513 : data = 10'b0000000011;
514 : data = 10'b0000000110;
515 : data = 10'b0000001001;
516 : data = 10'b0000001100;
517 : data = 10'b0000001111;
518 : data = 10'b0000010010;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
519 : data = 10'b0000010101;
520 : data = 10'b0000011001;
521 : data = 10'b0000011100;
522 : data = 10'b0000011111;
523 : data = 10'b0000100010;
524 : data = 10'b0000100101;
525 : data = 10'b0000101000;
526 : data = 10'b0000101011;
527 : data = 10'b0000101111;
528 : data = 10'b0000110010;
529 : data = 10'b0000110101;
530 : data = 10'b0000111000;
531 : data = 10'b0000111011;
532 : data = 10'b0000111110;
533 : data = 10'b0001000001;
534 : data = 10'b0001000101;
535 : data = 10'b0001001000;
536 : data = 10'b0001001011;
537 : data = 10'b0001001110;
538 : data = 10'b0001010001;
539 : data = 10'b0001010100;
540 : data = 10'b0001010111;
541 : data = 10'b0001011010;
542 : data = 10'b0001011110;
543 : data = 10'b0001100001;
544 : data = 10'b0001100100;
545 : data = 10'b0001100111;
546 : data = 10'b0001101010;
547 : data = 10'b0001101101;
548 : data = 10'b0001110000;
549 : data = 10'b0001110011;
550 : data = 10'b0001110111;
551 : data = 10'b0001111010;
552 : data = 10'b0001111101;
553 : data = 10'b0010000000;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
554 : data = 10'b0010000011;
555 : data = 10'b0010000110;
556 : data = 10'b0010001001;
557 : data = 10'b0010001100;
558 : data = 10'b0010010000;
559 : data = 10'b0010010011;
560 : data = 10'b0010010110;
561 : data = 10'b0010011001;
562 : data = 10'b0010011100;
563 : data = 10'b0010011111;
564 : data = 10'b0010100010;
565 : data = 10'b0010100101;
566 : data = 10'b0010101000;
567 : data = 10'b0010101011;
568 : data = 10'b0010101111;
569 : data = 10'b0010110010;
570 : data = 10'b0010110101;
571 : data = 10'b0010111000;
572 : data = 10'b0010111011;
573 : data = 10'b0010111110;
574 : data = 10'b0011000001;
575 : data = 10'b0011000100;
576 : data = 10'b0011000111;
577 : data = 10'b0011001010;
578 : data = 10'b0011001101;
579 : data = 10'b0011010001;
580 : data = 10'b0011010100;
581 : data = 10'b0011010111;
582 : data = 10'b0011011010;
583 : data = 10'b0011011101;
584 : data = 10'b0011100000;
585 : data = 10'b0011100011;
586 : data = 10'b0011100110;
587 : data = 10'b0011101001;
588 : data = 10'b0011101100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
589 : data = 10'b0011101111;
590 : data = 10'b0011110010;
591 : data = 10'b0011110101;
592 : data = 10'b0011111000;
593 : data = 10'b0011111011;
594 : data = 10'b0011111110;
595 : data = 10'b0100000001;
596 : data = 10'b0100000100;
597 : data = 10'b0100001000;
598 : data = 10'b0100001011;
599 : data = 10'b0100001110;
600 : data = 10'b0100010001;
601 : data = 10'b0100010100;
602 : data = 10'b0100010111;
603 : data = 10'b0100011010;
604 : data = 10'b0100011101;
605 : data = 10'b0100100000;
606 : data = 10'b0100100011;
607 : data = 10'b0100100110;
608 : data = 10'b0100101001;
609 : data = 10'b0100101100;
610 : data = 10'b0100101111;
611 : data = 10'b0100110010;
612 : data = 10'b0100110101;
613 : data = 10'b0100111000;
614 : data = 10'b0100111011;
615 : data = 10'b0100111110;
616 : data = 10'b0101000001;
617 : data = 10'b0101000100;
618 : data = 10'b0101000111;
619 : data = 10'b0101001010;
620 : data = 10'b0101001101;
621 : data = 10'b0101010000;
622 : data = 10'b0101010011;
623 : data = 10'b0101010110;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
624 : data = 10'b0101011000;
625 : data = 10'b0101011011;
626 : data = 10'b0101011110;
627 : data = 10'b0101100001;
628 : data = 10'b0101100100;
629 : data = 10'b0101100111;
630 : data = 10'b0101101010;
631 : data = 10'b0101101101;
632 : data = 10'b0101110000;
633 : data = 10'b0101110011;
634 : data = 10'b0101110110;
635 : data = 10'b0101111001;
636 : data = 10'b0101111100;
637 : data = 10'b0101111111;
638 : data = 10'b0110000010;
639 : data = 10'b0110000100;
640 : data = 10'b0110000111;
641 : data = 10'b0110001010;
642 : data = 10'b0110001101;
643 : data = 10'b0110010000;
644 : data = 10'b0110010011;
645 : data = 10'b0110010110;
646 : data = 10'b0110011001;
647 : data = 10'b0110011100;
648 : data = 10'b0110011110;
649 : data = 10'b0110100001;
650 : data = 10'b0110100100;
651 : data = 10'b0110100111;
652 : data = 10'b0110101010;
653 : data = 10'b0110101101;
654 : data = 10'b0110110000;
655 : data = 10'b0110110010;
656 : data = 10'b0110110101;
657 : data = 10'b0110111000;
658 : data = 10'b0110111011;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
659 : data = 10'b0110111110;
660 : data = 10'b0111000001;
661 : data = 10'b0111000011;
662 : data = 10'b0111000110;
663 : data = 10'b0111001001;
664 : data = 10'b0111001100;
665 : data = 10'b0111001111;
666 : data = 10'b0111010010;
667 : data = 10'b0111010100;
668 : data = 10'b0111010111;
669 : data = 10'b0111011010;
670 : data = 10'b0111011101;
671 : data = 10'b0111011111;
672 : data = 10'b0111100010;
673 : data = 10'b0111100101;
674 : data = 10'b0111101000;
675 : data = 10'b0111101011;
676 : data = 10'b0111101101;
677 : data = 10'b0111110000;
678 : data = 10'b0111110011;
679 : data = 10'b0111110101;
680 : data = 10'b0111111000;
681 : data = 10'b0111111011;
682 : data = 10'b0111111110;
683 : data = 10'b1000000000;
684 : data = 10'b1000000011;
685 : data = 10'b1000000110;
686 : data = 10'b1000001001;
687 : data = 10'b1000001011;
688 : data = 10'b1000001110;
689 : data = 10'b1000010001;
690 : data = 10'b1000010011;
691 : data = 10'b1000010110;
692 : data = 10'b1000011001;
693 : data = 10'b1000011011;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
694 : data = 10'b1000011110;
695 : data = 10'b1000100001;
696 : data = 10'b1000100011;
697 : data = 10'b1000100110;
698 : data = 10'b1000101001;
699 : data = 10'b1000101011;
700 : data = 10'b1000101110;
701 : data = 10'b1000110001;
702 : data = 10'b1000110011;
703 : data = 10'b1000110110;
704 : data = 10'b1000111000;
705 : data = 10'b1000111011;
706 : data = 10'b1000111110;
707 : data = 10'b1001000000;
708 : data = 10'b1001000011;
709 : data = 10'b1001000101;
710 : data = 10'b1001001000;
711 : data = 10'b1001001011;
712 : data = 10'b1001001101;
713 : data = 10'b1001010000;
714 : data = 10'b1001010010;
715 : data = 10'b1001010101;
716 : data = 10'b1001010111;
717 : data = 10'b1001011010;
718 : data = 10'b1001011100;
719 : data = 10'b1001011111;
720 : data = 10'b1001100001;
721 : data = 10'b1001100100;
722 : data = 10'b1001100111;
723 : data = 10'b1001101001;
724 : data = 10'b1001101100;
725 : data = 10'b1001101110;
726 : data = 10'b1001110001;
727 : data = 10'b1001110011;
728 : data = 10'b1001110101;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
729 : data = 10'b1001111000;
730 : data = 10'b1001111010;
731 : data = 10'b1001111101;
732 : data = 10'b1001111111;
733 : data = 10'b1010000010;
734 : data = 10'b1010000100;
735 : data = 10'b1010000111;
736 : data = 10'b1010001001;
737 : data = 10'b1010001100;
738 : data = 10'b1010001110;
739 : data = 10'b1010010000;
740 : data = 10'b1010010011;
741 : data = 10'b1010010101;
742 : data = 10'b1010011000;
743 : data = 10'b1010011010;
744 : data = 10'b1010011100;
745 : data = 10'b1010011111;
746 : data = 10'b1010100001;
747 : data = 10'b1010100011;
748 : data = 10'b1010100110;
749 : data = 10'b1010101000;
750 : data = 10'b1010101011;
751 : data = 10'b1010101101;
752 : data = 10'b1010101111;
753 : data = 10'b1010110001;
754 : data = 10'b1010110100;
755 : data = 10'b1010110110;
756 : data = 10'b1010111000;
757 : data = 10'b1010111011;
758 : data = 10'b1010111101;
759 : data = 10'b1010111111;
760 : data = 10'b1011000010;
761 : data = 10'b1011000100;
762 : data = 10'b1011000110;
763 : data = 10'b1011001000;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
764 : data = 10'b1011001011;
765 : data = 10'b1011001101;
766 : data = 10'b1011001111;
767 : data = 10'b1011010001;
768 : data = 10'b1011010100;
769 : data = 10'b1011010110;
770 : data = 10'b1011011000;
771 : data = 10'b1011011010;
772 : data = 10'b1011011100;
773 : data = 10'b1011011111;
774 : data = 10'b1011100001;
775 : data = 10'b1011100011;
776 : data = 10'b1011100101;
777 : data = 10'b1011100111;
778 : data = 10'b1011101001;
779 : data = 10'b1011101100;
780 : data = 10'b1011101110;
781 : data = 10'b1011110000;
782 : data = 10'b1011110010;
783 : data = 10'b1011110100;
784 : data = 10'b1011110110;
785 : data = 10'b1011111000;
786 : data = 10'b1011111010;
787 : data = 10'b1011111101;
788 : data = 10'b1011111111;
789 : data = 10'b1100000001;
790 : data = 10'b1100000011;
791 : data = 10'b1100000101;
792 : data = 10'b1100000111;
793 : data = 10'b1100001001;
794 : data = 10'b1100001011;
795 : data = 10'b1100001101;
796 : data = 10'b1100001111;
797 : data = 10'b1100010001;
798 : data = 10'b1100010011;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
799  : data = 10'b1100010101;
800  : data = 10'b1100010111;
801  : data = 10'b1100011001;
802  : data = 10'b1100011011;
803  : data = 10'b1100011101;
804  : data = 10'b1100011111;
805  : data = 10'b1100100001;
806  : data = 10'b1100100011;
807  : data = 10'b1100100101;
808  : data = 10'b1100100111;
809  : data = 10'b1100101001;
810  : data = 10'b1100101011;
811  : data = 10'b1100101101;
812  : data = 10'b1100101110;
813  : data = 10'b1100110000;
814  : data = 10'b1100110010;
815  : data = 10'b1100110100;
816  : data = 10'b1100110110;
817  : data = 10'b1100111000;
818  : data = 10'b1100111010;
819  : data = 10'b1100111100;
820  : data = 10'b1100111101;
821  : data = 10'b1100111111;
822  : data = 10'b1101000001;
823  : data = 10'b1101000011;
824  : data = 10'b1101000101;
825  : data = 10'b1101000111;
826  : data = 10'b1101001000;
827  : data = 10'b1101001010;
828  : data = 10'b1101001100;
829  : data = 10'b1101001110;
830  : data = 10'b1101001111;
831  : data = 10'b1101010001;
832  : data = 10'b1101010011;
833  : data = 10'b1101010101;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
834 : data = 10'b1101010110;
835 : data = 10'b1101011000;
836 : data = 10'b1101011010;
837 : data = 10'b1101011100;
838 : data = 10'b1101011101;
839 : data = 10'b1101011111;
840 : data = 10'b1101100001;
841 : data = 10'b1101100010;
842 : data = 10'b1101100100;
843 : data = 10'b1101100110;
844 : data = 10'b1101100111;
845 : data = 10'b1101101001;
846 : data = 10'b1101101011;
847 : data = 10'b1101101100;
848 : data = 10'b1101101110;
849 : data = 10'b1101101111;
850 : data = 10'b1101110001;
851 : data = 10'b1101110011;
852 : data = 10'b1101110100;
853 : data = 10'b1101110110;
854 : data = 10'b1101110111;
855 : data = 10'b1101111001;
856 : data = 10'b1101111010;
857 : data = 10'b1101111100;
858 : data = 10'b1101111110;
859 : data = 10'b1101111111;
860 : data = 10'b1110000001;
861 : data = 10'b1110000010;
862 : data = 10'b1110000100;
863 : data = 10'b1110000101;
864 : data = 10'b1110000111;
865 : data = 10'b1110001000;
866 : data = 10'b1110001010;
867 : data = 10'b1110001011;
868 : data = 10'b1110001100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
869 : data = 10'b1110001110;
870 : data = 10'b1110001111;
871 : data = 10'b1110010001;
872 : data = 10'b1110010010;
873 : data = 10'b1110010100;
874 : data = 10'b1110010101;
875 : data = 10'b1110010110;
876 : data = 10'b1110011000;
877 : data = 10'b1110011001;
878 : data = 10'b1110011010;
879 : data = 10'b1110011100;
880 : data = 10'b1110011101;
881 : data = 10'b1110011111;
882 : data = 10'b1110100000;
883 : data = 10'b1110100001;
884 : data = 10'b1110100010;
885 : data = 10'b1110100100;
886 : data = 10'b1110100101;
887 : data = 10'b1110100110;
888 : data = 10'b1110101000;
889 : data = 10'b1110101001;
890 : data = 10'b1110101010;
891 : data = 10'b1110101011;
892 : data = 10'b1110101101;
893 : data = 10'b1110101110;
894 : data = 10'b1110101111;
895 : data = 10'b1110110000;
896 : data = 10'b1110110010;
897 : data = 10'b1110110011;
898 : data = 10'b1110110100;
899 : data = 10'b1110110101;
900 : data = 10'b1110110110;
901 : data = 10'b1110110111;
902 : data = 10'b1110111001;
903 : data = 10'b1110111010;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
904 : data = 10'b1110111011;
905 : data = 10'b1110111100;
906 : data = 10'b1110111101;
907 : data = 10'b1110111110;
908 : data = 10'b1110111111;
909 : data = 10'b1111000000;
910 : data = 10'b1111000010;
911 : data = 10'b1111000011;
912 : data = 10'b1111000100;
913 : data = 10'b1111000101;
914 : data = 10'b1111000110;
915 : data = 10'b1111000111;
916 : data = 10'b1111001000;
917 : data = 10'b1111001001;
918 : data = 10'b1111001010;
919 : data = 10'b1111001011;
920 : data = 10'b1111001100;
921 : data = 10'b1111001101;
922 : data = 10'b1111001110;
923 : data = 10'b1111001111;
924 : data = 10'b1111010000;
925 : data = 10'b1111010001;
926 : data = 10'b1111010010;
927 : data = 10'b1111010010;
928 : data = 10'b1111010011;
929 : data = 10'b1111010100;
930 : data = 10'b1111010101;
931 : data = 10'b1111010110;
932 : data = 10'b1111010111;
933 : data = 10'b1111011000;
934 : data = 10'b1111011001;
935 : data = 10'b1111011010;
936 : data = 10'b1111011010;
937 : data = 10'b1111011011;
938 : data = 10'b1111011100;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
939  : data = 10'b1111011101;
940  : data = 10'b1111011110;
941  : data = 10'b1111011110;
942  : data = 10'b1111011111;
943  : data = 10'b1111100000;
944  : data = 10'b1111100001;
945  : data = 10'b1111100010;
946  : data = 10'b1111100010;
947  : data = 10'b1111100011;
948  : data = 10'b1111100100;
949  : data = 10'b1111100101;
950  : data = 10'b1111100101;
951  : data = 10'b1111100110;
952  : data = 10'b1111100111;
953  : data = 10'b1111100111;
954  : data = 10'b1111101000;
955  : data = 10'b1111101001;
956  : data = 10'b1111101001;
957  : data = 10'b1111101010;
958  : data = 10'b1111101011;
959  : data = 10'b1111101011;
960  : data = 10'b1111101100;
961  : data = 10'b1111101100;
962  : data = 10'b1111101101;
963  : data = 10'b1111101110;
964  : data = 10'b1111101110;
965  : data = 10'b1111101111;
966  : data = 10'b1111101111;
967  : data = 10'b1111110000;
968  : data = 10'b1111110000;
969  : data = 10'b1111110001;
970  : data = 10'b1111110001;
971  : data = 10'b1111110010;
972  : data = 10'b1111110010;
973  : data = 10'b1111110011;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
974 : data = 10'b1111110011;
975 : data = 10'b1111110100;
976 : data = 10'b1111110100;
977 : data = 10'b1111110101;
978 : data = 10'b1111110101;
979 : data = 10'b1111110110;
980 : data = 10'b1111110110;
981 : data = 10'b1111110111;
982 : data = 10'b1111110111;
983 : data = 10'b1111110111;
984 : data = 10'b1111111000;
985 : data = 10'b1111111000;
986 : data = 10'b1111111001;
987 : data = 10'b1111111001;
988 : data = 10'b1111111001;
989 : data = 10'b1111111010;
990 : data = 10'b1111111010;
991 : data = 10'b1111111010;
992 : data = 10'b1111111011;
993 : data = 10'b1111111011;
994 : data = 10'b1111111011;
995 : data = 10'b1111111011;
996 : data = 10'b1111111100;
997 : data = 10'b1111111100;
998 : data = 10'b1111111100;
999 : data = 10'b1111111100;
1000 : data = 10'b1111111101;
1001 : data = 10'b1111111101;
1002 : data = 10'b1111111101;
1003 : data = 10'b1111111101;
1004 : data = 10'b1111111110;
1005 : data = 10'b1111111110;
1006 : data = 10'b1111111110;
1007 : data = 10'b1111111110;
1008 : data = 10'b1111111110;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
1009  : data = 10'b1111111110;
1010  : data = 10'b1111111111;
1011  : data = 10'b1111111111;
1012  : data = 10'b1111111111;
1013  : data = 10'b1111111111;
1014  : data = 10'b1111111111;
1015  : data = 10'b1111111111;
1016  : data = 10'b1111111111;
1017  : data = 10'b1111111111;
1018  : data = 10'b1111111111;
1019  : data = 10'b1111111111;
1020  : data = 10'b1111111111;
1021  : data = 10'b1111111111;
1022  : data = 10'b1111111111;
1023  : data = 10'b1111111111;
default : data = 10'bx;
  endcase
end
endmodule
// มอดูลชุดวงจรคูณค่าฟังก์ชัน
 module mul12x10(clk,rst,A,B,C);
   input clk;
   input rst;
   input [11:0] A;
   input [9:0] B;
   output [16:0] C;
          wire [11:0] A1;
          wire [9:0] B1;
          wire [16:0] C1;
         wire [35:0] p;
  supply1 ce;
          assign C1 = p[21:5];
 dff12 dff0(clk,rst,A,A1);
 dff10 dff1(clk,rst,B,B1);
 MULT18X18S MUL(.P(p), .A({6'h0,A1}), .B({8'h0,B1}), .C(clk), .CE(ce), .R(rst));
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
dff17 dff2(clk,rst,C1,C);
endmodule
module dff17(clk,rst,din,dout);
        input [16:0]    din;
        input clk,rst;
        output [16:0]   dout;
        reg [16:0]      dout;
always @(posedge clk or posedge rst)
begin
        if (rst)
                dout <= 0; //asynchronous RESET active High
  else
                dout <= din;
end
endmodule
 module dff12(clk,rst,din,dout);
        input [11:0]    din;
        input clk,rst;
        output [11:0]   dout;
        reg [11:0]      dout;
always @(posedge clk or posedge rst)
begin
        if (rst)
                dout <= 0;
  else
                dout <= din;
end
endmodule

 module dff10(clk,rst,din,dout);
        input [9:0]     din;
        input clk,rst;
        output [9:0]    dout;
        reg [9:0]  dout;
always @(posedge clk or posedge rst)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
begin
        if (rst)
                dout <= 0;
  else
                dout <= din;
end
endmodule
```

// มอดูลประมาณค่า SNR

```
module snr_romc2(clk,addr,data);
   input clk;
   input [7:0] addr;
   output [15:0] data;
 reg [15:0] data;
always @(posedge clk)
begin
          case (addr)
8'b00000000 : data = 16'b1000000000000000;
8'b00000001 : data = 16'b0111111010001000;
8'b00000010 : data = 16'b0111110100010110;
8'b00000011 : data = 16'b0111101110100111;
8'b00000100 : data = 16'b0111101000111101;
8'b00000101 : data = 16'b0111100011010110;
8'b00000110 : data = 16'b0111011101110100;
8'b00000111 : data = 16'b0111011000010110;
8'b00001000 : data = 16'b0111010010111100;
8'b00001001 : data = 16'b0111001101100110;
8'b00010000 : data = 16'b0111001000010100;
8'b00010001 : data = 16'b0111000011000110;
8'b00010010 : data = 16'b0110111101111011;
8'b00010011 : data = 16'b0110111000110101;
8'b00010100 : data = 16'b0110110011110010;
8'b00010101 : data = 16'b0110101110110010;
8'b00010110 : data = 16'b0110101001110111;
8'b00010111 : data = 16'b0110100100111111;
8'b00011000 : data = 16'b0110100000001010;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
8'b00011001 : data = 16'b0110011011011001;
8'b00100000 : data = 16'b0110010110101100;
8'b00100001 : data = 16'b0110010010000010;
8'b00100010 : data = 16'b0110001101011100;
8'b00100011 : data = 16'b0110001000111000;
8'b00100100 : data = 16'b0110000100011001;
8'b00100101 : data = 16'b0101111111111100;
8'b00100110 : data = 16'b0101111011100011;
8'b00100111 : data = 16'b0101110111001101;
8'b00101000 : data = 16'b0101110010111010;
8'b00101001 : data = 16'b0101101110101010;
8'b00110000 : data = 16'b0101101010011101;
8'b00110001 : data = 16'b0101100110010100;
8'b00110010 : data = 16'b0101100010001101;
8'b00110011 : data = 16'b0101011110001010;
8'b00110100 : data = 16'b0101011010001001;
8'b00110101 : data = 16'b0101010110001100;
8'b00110110 : data = 16'b0101010010010001;
8'b00110111 : data = 16'b0101001110011001;
8'b00111000 : data = 16'b0101001010100100;
8'b00111001 : data = 16'b0101000110110010;
8'b01000000 : data = 16'b0101000011000011;
8'b01000001 : data = 16'b0100111111010110;
8'b01000010 : data = 16'b0100111011101100;
8'b01000011 : data = 16'b0100111000000101;
8'b01000100 : data = 16'b0100110100100000;
8'b01000101 : data = 16'b0100110000111110;
8'b01000110 : data = 16'b0100101101011111;
8'b01000111 : data = 16'b0100101010000010;
8'b01001000 : data = 16'b0100100110101000;
8'b01001001 : data = 16'b0100100011010000;
8'b01010000 : data = 16'b0100001111111010;
8'b01010001 : data = 16'b0100001100100111;
8'b01010010 : data = 16'b0100001001010111;
8'b01010011 : data = 16'b0100010110001001;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
8'b01010100 : data = 16'b0100010010111101;
8'b01010101 : data = 16'b0100001111110100;
8'b01010110 : data = 16'b0100001100101100;
8'b01010111 : data = 16'b0100001001101000;
8'b01011000 : data = 16'b0100000110100101;
8'b01011001 : data = 16'b0100000011100101;
8'b01100000 : data = 16'b0100000000100110;
8'b01100001 : data = 16'b0011111101101010;
8'b01100010 : data = 16'b0011111010110001;
8'b01100011 : data = 16'b0011110111111001;
8'b01100100 : data = 16'b0011110101000011;
8'b01100101 : data = 16'b0011110010010000;
8'b01100110 : data = 16'b0011101111011110;
8'b01100111 : data = 16'b0011101100101111;
8'b01101000 : data = 16'b0011101010000001;
8'b01101001 : data = 16'b0011100111010110;
8'b01110000 : data = 16'b0011100100101100;
8'b01110001 : data = 16'b0011100010000101;
8'b01110010 : data = 16'b0011011111011111;
8'b01110011 : data = 16'b0011011100111100;
8'b01110100 : data = 16'b0011011010011010;
8'b01110101 : data = 16'b0011010111111010;
8'b01110110 : data = 16'b0011010101011011;
8'b01110111 : data = 16'b0011010010111111;
8'b01111000 : data = 16'b0011010000100101;
8'b01111001 : data = 16'b0011001110001100;
8'b10000000 : data = 16'b0011001011110101;
8'b10000001 : data = 16'b0011001001011111;
8'b10000010 : data = 16'b0011000111001100;
8'b10000011 : data = 16'b0011000100111010;
8'b10000100 : data = 16'b0011000010101010;
8'b10000101 : data = 16'b0011000000011011;
8'b10000110 : data = 16'b0010111110001110;
8'b10000111 : data = 16'b0010111100000011;
8'b10001000 : data = 16'b0010111001111001;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
8'b10001001 : data = 16'b0010110111110001;
8'b10010000 : data = 16'b0010110101101010;
8'b10010001 : data = 16'b0010110011100101;
8'b10010010 : data = 16'b0010110001100001;
8'b10010011 : data = 16'b0010101111011111;
8'b10010100 : data = 16'b0010101101011111;
8'b10010101 : data = 16'b0010101011100000;
8'b10010110 : data = 16'b0010101001100010;
8'b10010111 : data = 16'b0010100111100110;
8'b10011000 : data = 16'b0010100101101011;
8'b10011001 : data = 16'b0010100011110010;
8'b10100000 : data = 16'b0010100001111010;
8'b10100001 : data = 16'b0010100000000011;
8'b10100010 : data = 16'b0010011110001110;
8'b10100011 : data = 16'b0010011100011010;
8'b10100100 : data = 16'b0010011010100111;
8'b10100101 : data = 16'b0010011000110110;
8'b10100110 : data = 16'b0010010111000110;
8'b10100111 : data = 16'b0010010101010111;
8'b10101000 : data = 16'b0010010011101010;
8'b10101001 : data = 16'b0010010001111110;
8'b10110000 : data = 16'b0010010000010011;
8'b10110001 : data = 16'b0010001110101001;
8'b10110010 : data = 16'b0010001101000001;
8'b10110011 : data = 16'b0010001011011001;
8'b10110100 : data = 16'b0010001001110011;
8'b10110101 : data = 16'b0010001000001110;
8'b10110110 : data = 16'b0010000110101010;
8'b10110111 : data = 16'b0010000101001000;
8'b10111000 : data = 16'b0010000011100110;
8'b10111001 : data = 16'b0010000010000110;
8'b11000000 : data = 16'b0010000000100110;
8'b11000001 : data = 16'b0001111111001000;
8'b11000010 : data = 16'b0001111101101011;
8'b11000011 : data = 16'b0001111100001111;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
8'b11000100 : data = 16'b0001111010110100;
8'b11000101 : data = 16'b0001111001011010;
8'b11000110 : data = 16'b0001111000000001;
8'b11000111 : data = 16'b0001110110101001;
8'b11001000 : data = 16'b0001110101010010;
8'b11001001 : data = 16'b0001110011111100;
8'b11010000 : data = 16'b0001110010100111;
8'b11010001 : data = 16'b0001110001010011;
8'b11010010 : data = 16'b0001110000000000;
8'b11010011 : data = 16'b0001101110101110;
8'b11010100 : data = 16'b0001101101011101;
8'b11010101 : data = 16'b0001101100001101;
8'b11010110 : data = 16'b0001101010111110;
8'b11010111 : data = 16'b0001101001101111;
8'b11011000 : data = 16'b0001101000100010;
8'b11011001 : data = 16'b0001100111010101;
8'b11100000 : data = 16'b0001100110001010;
8'b11100001 : data = 16'b0001100100111111;
8'b11100010 : data = 16'b0001100011110101;
8'b11100011 : data = 16'b0001100010101100;
8'b11100100 : data = 16'b0001100001100011;
8'b11100101 : data = 16'b0001100000011100;
8'b11100110 : data = 16'b0001011111010101;
8'b11100111 : data = 16'b0001011110001111;
8'b11101000 : data = 16'b0001011101001010;
8'b11101001 : data = 16'b0001011100000110;
8'b11110000 : data = 16'b0001011011000011;
8'b11110001 : data = 16'b0001011010000000;
8'b11110010 : data = 16'b0001011000111110;
8'b11110011 : data = 16'b0001010111111101;
8'b11110100 : data = 16'b0001010110111100;
8'b11110101 : data = 16'b0001010101111101;
8'b11110110 : data = 16'b0001010100111110;
8'b11110111 : data = 16'b0001010011111111;
8'b11111000 : data = 16'b0001010011000010;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
8'b11111001  : data = 16'b0001010010000101;
default : data = 16'b1011010100000100;
  endcase
end
endmodule
```

//มอดูลคูณค่า SNR

```
module mul17x16(clk,rst,A,B,C);
   input clk;
   input rst;
   input [16:0] A;
   input [15:0] B;
   output [14:0] C;
          wire [16:0] A1;
          wire [15:0] B1;
          wire [14:0] C1;
          wire [35:0] p;
   supply1 ce;
          assign C1 = p[32:18];
  dff17 dff0(clk,rst,A,A1);
  dff16 dff1(clk,rst,B,B1);
  MULT18X18S MUL(.P(p), .A({1'b0,A1}), .B({2'b0,B1}), .C(clk), .CE(ce), .R(rst));
  dff15 dff2(clk,rst,C1,C);
  endmodule
  module dff17(clk,rst,din,dout);
          input [16:0]      din;
          input clk,rst;
          output [16:0]     dout;
          reg [16:0]        dout;
   always @(posedge clk or posedge rst)
   begin
          if (rst)
                  dout <= 0;
    else
                  dout <= din;
   end
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
endmodule
module dff16(clk,rst,din,dout);
        input [15:0]        din;
        input clk,rst;
        output [15:0]       dout;
        reg [15:0]          dout;
always @(posedge clk or posedge rst)
begin
        if (rst)
                dout <= 0; //asynchronous RESET active High
  else
                dout <= din;
end
endmodule
module dff15(clk,rst,din,dout);
        input [14:0]        din;
        input clk,rst;
        output [14:0]       dout;
        reg [14:0]          dout;
always @(posedge clk or posedge rst)
begin
        if (rst)
                dout <= 0; //asynchronous RESET active High
  else
                dout <= din;
end
endmodule
//มอดูลหน่วงเวลา
module sign_cos_delay(clk,rst,din,dout);
        input     din,clk,rst;
        output    dout;
        reg [3:0] rout;
        assign dout = rout[0];
always @(posedge clk or posedge rst)
begin
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        if (rst)
                rout <= 0; //asynchronous RESET active High
 else
                rout <= {din,rout[3:1]};
end
endmodule
```

// มอดูลปรับเปลี่ยนค่า

```
module comp(clock,reset,wgn,x);
   input clock;
   input reset;
   input [15:0] wgn;
   output [15:0] x;
 wire [14:0] div;
 reg s;
 reg [14:0] wgnd2;
 assign div = wgn[14:0];
 assign x = {s,wgnd2};
 always @(posedge clock or posedge reset)
 begin
          if (reset)
          begin
          wgnd2 <= 0;
          s <= 0;
          end
          else
          begin
          s <= wgn[15];
          if (wgn[15])
          wgnd2  <= (~div) + 1;
          else
          wgnd2 <= div;
          end
          end
   endmodule
   module comp2(clock,reset,wgn,x);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  input clock;
  input reset;
  input [15:0] wgn;
  output [15:0] x;
wire [14:0] div;
reg s;
reg [14:0] wgnd2;
assign div = wgn[14:0];
assign x = {s,wgnd2};
always @(posedge clock or posedge reset)
begin
        if (reset)
        begin
        wgnd2 <= 0;
        s <= 0;
        end
        else
        begin
        wgnd2 <= div;
        if (div == 0)
        s <= 0;
        else
        s <= wgn[15];
        end
  end
  endmodule
  //มอดูลดีมอดูเลเตอร์
  module bpsk_chan(clk,rst,bpskin,wgn,ch_out);
    input clk;
    input rst;
    input [3:0] bpskin;
    input [15:0] wgn;
    output ch_out;
        wire [15:0] A,B;
        reg [15:0] C;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
assign A = {bpskin,12'b0};
assign B = wgn;
assign ch_out = C[15];
always @(posedge clk or posedge rst)
begin
 if (rst)
 C <= 0;
 else
 C <= A + B;
end
endmodule
```

//มอดูลวงจรนับ

```
module data_counter(dclk,rst,sent_en);
  input dclk;
  input rst;
  output sent_en;
        reg sent_en;
        reg [20:0] data_counter;
always @(posedge dclk or posedge rst)
begin
  if (rst)
        begin
              sent_en = 0;
    data_counter = 0;
        end
  else
        begin
        if (data_counter >= 2062500)
                    sent_en = 1;
              else
              data_counter = data_counter + 1;
  end
end
endmodule
```

//มอดูลหน่วงเวลาของข้อมูลไม่ได้เข้ารหัส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
module Delay(dclk,rst,din,uncode_delay);
  input dclk;
  input rst;
  input din;
  output uncode_delay;
        reg [1:0] SR;
assign uncode_delay        = SR[0];
always @(posedge dclk or posedge rst)
begin
if (rst)
        SR <=0;
else
        SR <= {din,SR[1]};
end
endmodule
//มอดูลหน่วงเวลาของข้อมูลเข้ารหัส
module Delay_dataENCODE(clk,rst,data,delay);
  input clk;
  input rst;
  input data;
  output delay;
                reg [23:0] SR;
assign delay        = SR[0];
always @(posedge clk or posedge rst)
begin
if (rst)
        SR <=0;
else
        SR <= {data,SR[23:1]};
end
endmodule
// มอดูลนับบิตผิด
module error_counter(clk,rst,data1,data2,error_counter);
  input clk;
  input rst,data1,data2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  output  [23:0] error_counter;
          wire  error;
          reg  [23:0] error_counter;
          assign error =data1 ^ data2;
always @(negedge clk or posedge rst)
begin
          if (rst)
          error_counter = 24'b0;
   else
                    if (error)
                              error_counter = error_counter + 1;
end
endmodule
```

//มอดูลเชื่อมต่อกับคอมพิวเตอร์

```
module Com_interface(clk_uart,rstc,sent_en,Uncode_error,Code_error,RX,TX,Rate,SNR,rst);
          input clk_uart,rstc,sent_en;
          input [23:0]  Uncode_error,Code_error;
          input RX;
          output TX;
          output  Rate,rst;
          output [3:0] SNR;
          wire tclk,rclk,rx_en;
          wire [2:0] send_select;
          wire [7:0] data_tx,data_rx,ebno_out;
                    reg Rate,rst;
                    reg [3:0] SNR;
   clkdiv clkdiv(clk_uart,rstc,tclk,rclk);
   uart_rx RX1(rclk,rstc,RX,data_rx,rx_en);
   uart_tx TX1(tclk,rstc,sent_en,data_tx,TX,tx_rdy);
   mux6_8
   MUX(tx_rdy,rstc,Uncode_error[7:0],Uncode_error[15:8],Uncode_error[23:16],Code_error[7:0],Code_error[15:8],C
   ode_error[23:16],ebno_out,send_select,data_tx);
   conter3 COUNTER0(tx_rdy,rst,send_select);
   assign ebno_out = data_rx;// {3'b0,Rate,SNR};
   always @ (negedge rclk or posedge rstc)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
begin
 if (rstc)
 begin
 Rate = 0;
 SNR = 0;
 end
 else if  (rx_en == 1)
 begin
 SNR = data_rx[3:0];
 Rate = data_rx[4];
 end
 end
 always @ (posedge tclk or posedge rstc)
 begin
 if (rstc)
            rst = 1;
            else
                  if (send_select == 3'b111)
                                    rst = 1;
                              else
                                    rst = 0;
 end
 endmodule
 module clkdiv(clk,rst,tclk,rclk);
    input clk,rst;
    output tclk,rclk;
            reg [5:0] count;// 5   2
            assign     tclk = count[5];
            assign     rclk = count[2];
  always @(posedge clk or posedge rst)
  begin
            if (rst)
            count = 0;
            else
            count = count + 1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
end
endmodule
module conter3(clk,rst,COUNT);
   input clk;
   input rst;
   output [2:0] COUNT;
            reg [2:0] COUNT;
always @(posedge clk or posedge rst)
begin
  if (rst)
     COUNT <= 3'b0;
  else
     COUNT <= COUNT + 1;
end
endmodule
module mux6_8(clk,rst,A,B,C,D,E,F,G,SEL,MUX_OUT);
              input clk,rst;
    input [2:0] SEL;
    input [7:0]  A, B, C, D, E, F,G;
    output [7:0] MUX_OUT;
    reg [7:0] MUX_OUT;
 always @ (posedge clk or posedge rst)
 begin
 if (rst)
            MUX_OUT = 0;
 else
 begin
   case (SEL)
     3'b000 : MUX_OUT = A;
     3'b001 : MUX_OUT = B;
     3'b010 : MUX_OUT = C;
     3'b011 : MUX_OUT = D;
                    3'b100 : MUX_OUT = E;
     3'b101 : MUX_OUT = F;
                    3'b110 : MUX_OUT = G;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                default : MUX_OUT = 0;
  endcase
end
end
endmodule
module uart_rx(clk,rst,rx,data_rx,rx_en);
        input clk,rst,rx;
        output rx_en;
        output [7:0] data_rx;
        wire i_reg,r_reg;
        reg rx_en;
        reg [1:0] State_RX;
        reg [3:0] RX_Data_Count;
        reg [2:0] scount;
        reg [7:0] Buffer_RX,data_rx;
        reg [6:0] rx_reg ;
        parameter [1:0]
         Idel = 2'b00,
    ReceiveData = 2'b01,
    Stop = 2'b10;
 assign i_reg = rx_reg[6] | rx_reg[3] | rx_reg[0];
 assign r_reg = (rx_reg[3] & rx_reg[4]) | (rx_reg[4] & rx_reg[5]) | (rx_reg[3] & rx_reg[5]);
 always @(posedge clk or posedge rst)
 begin
 if (rst)// == 0)
          begin
          data_rx = 8'b11111111;
          rx_en = 0;
          State_RX = Idel;
          RX_Data_Count = 0;
          Buffer_RX = 0;
          rx_reg = 7'b1111111;
          scount = 0;
          end
  else
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
begin
        rx_reg = {rx,rx_reg[6:1]};
        case (State_RX)
    Idel:
                begin
                rx_en = 0;
                if (i_reg == 0)
                        begin
                        RX_Data_Count = 0;
                        State_RX = ReceiveData;
                        end
                else
                        begin
                        RX_Data_Count = 0;
                        State_RX = Idel;
                        end
                end
                ReceiveData: begin
                rx_en = 0;
                if (RX_Data_Count == 8)
                        begin
                        if (i_reg == 0)
                        State_RX = Idel;
                        else
                        State_RX = Stop;
                        end
                else
                        begin
                        if (scount == 7)
                        begin
                        Buffer_RX[RX_Data_Count] = r_reg;
                        RX_Data_Count = RX_Data_Count + 1;
                        scount = 0;
                        end
                        else
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    begin
                    scount = scount +1;
                    State_RX = ReceiveData;
                    end
                    end
                end
Stop : begin
                rx_en = 1;
                RX_Data_Count = 0;
                data_rx = Buffer_RX;
                State_RX = Idel;
                end
 default : begin
                data_rx = 0;
                rx_en = 0;
                RX_Data_Count = 0;
                State_RX = Idel;
                end
 endcase
        end
end
endmodule
module uart_tx(clk,rst,tx_en,data_tx,tx,tx_rdy);
        input clk,rst,tx_en;
        input [7:0] data_tx;
        output tx,tx_rdy;
        reg tx,tx_rdy;
        reg [1:0] State_tx;
        reg [2:0] tx_count;
        parameter [1:0]
         Idel = 2'b00,
   Start = 2'b01,
         Transdata = 2'b10,
   Stop = 2'b11;
 always @(posedge clk or posedge rst)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
begin
if (rst)// == 0)
        begin
        State_tx = Idel;
        tx_count = 0;
        tx = 1;
        tx_rdy = 0;
        end
else
        case (State_tx)
   Idel: begin
                tx = 1;
                tx_rdy = 0;
                if  (tx_en == 1)
                        begin
                        tx_count = 0;
                        State_tx= Start;
                        end
                else
                        begin
                        tx_count = 0;
                        State_tx = Idel;
                        end
                end
                Start: begin
                tx = 0;
                tx_rdy = 1;
                tx_count = 0;
                State_tx = Transdata;
                end
                Transdata : begin
                tx_rdy = 1;
                if  (tx_count == 7)
                        begin
                        tx = data_tx[tx_count];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                State_tx = Stop;
                end
            else
                begin
                tx = data_tx[tx_count];
                tx_count = tx_count + 1;
                State_tx = Transdata;
                end
        end
    Stop : begin
                tx = 1;
                tx_count = 0;
                tx_rdy = 0;
                State_tx = Idel;
                end
    endcase
end
endmodule
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข

ชุดเข้ารหัสและถอดรหัสคอนโวลูชัน

//มอดูลเข้ารหัสคอนโวลูชัน K =3 Rate =1/2

```
module encodeK3R12(clk,rst,data_in,enc_out,data_delay);
        input clk;
        input rst;
        input data_in;
        output data_delay;
        output[1:0] enc_out;
        reg[1:0] enc_out;
        reg[2:0] D;
        reg [36:0] SR;
        assign data_delay =SR[0];
always @(posedge clk or posedge rst)
begin
        if (rst)
                begin
                        D <=0;
                        enc_out <=0;
                        SR <= 0;
                end
        else
                begin
                        SR <= {data_in,SR[36:1]};
                        D<= {data_in,D[2:1]};
                        enc_out[1] <= D[2] ^ D[1] ^ D[0]; //7
                        enc_out[0] <= D[2] ^ D[0]; //5
                end
end
endmodule
```

//มอดูลถอดรหัสคอนโวลูชัน

// มอดูลหลัก

```
module top(clk,rst,din,dout);
   input clk;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  input rst;
  input [1:0] din;
  output dout;
  wire [1:0] bm0,bm1,bm2,bm3;
  wire [3:0] survivor;
bmg bmg1(clk,rst,din,bm0,bm1,bm2,bm3);
acs acs1(clk,rst,bm0,bm1,bm2,bm3,survivor);
smu smu(clk,rst,survivor,dout);
endmodule
```

//มอดูล BMU

```
module bmg(clk,rst,in,bm0,bm1,bm2,bm3);
   input  clk;        input rst;
   input [1:0] in;
   output [1:0] bm0;
   output [1:0] bm1;
   output [1:0] bm2;
   output [1:0] bm3;
   reg [1:0] bm0,bm1,bm2,bm3;
 always @(posedge clk or posedge rst)
 if (rst)
            begin
            bm0 <= 0;
            bm1 <= 0;
            bm2 <= 0;
            bm3 <= 0;
            end
  else
            begin
            case (in)
      2'b00 : {bm0,bm1,bm2,bm3} <= 8'b00010110;
      2'b01 : {bm0,bm1,bm2,bm3} <= 8'b01001001;
      2'b10 : {bm0,bm1,bm2,bm3} <= 8'b01100001;
      2'b11 : {bm0,bm1,bm2,bm3} <= 8'b10010100;
      default :{bm0,bm1,bm2,bm3} <= 8'b0;
            endcase
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
end
endmodule
// มอดูลACS
module acs(clk,rst,in0,in1,in2,in3,survivor);
   input [1:0] in0;
   input [1:0] in1;
   input [1:0] in2;
   input [1:0] in3;
   output [3:0] survivor;
   input clk;
   input rst;
wire[3:0] p0,p1,p2,p3,m0,m1,m2,m3;
wire s0,s1,s2,s3;
bacs bacs0(in0,in3,in3,in0,p1,p0,s2,s0,m2,m0);
bacs bacs1(in2,in1,in1,in2,p3,p2,s3,s1,m3,m1);
dff4 dff0(clk,rst,m0,p0);
dff4 dff1(clk,rst,m1,p1);
dff4 dff2(clk,rst,m2,p2);
dff4 dff3(clk,rst,m3,p3);
assign survivor = {s3,s2,s1,s0};
endmodule
//มอดูลSMU
module smu(clk,rst,survivor,dout);
    input clk;
    input rst;
    input [3:0] survivor;
    output dout;
            wire [3:0] s1,s2,s3,s4,s5,s6,s7,s8,s9;
            wire [3:0] s10,s11,s12,s13,s14,s15,s16,s17,s18,s19;
            wire [3:0] s20,s21,s22,s23,s24,s25,s26,s27,s28,s29,s30,s31;
            wire [1:0] se1,se2,se3,se4,se5,se6,se7,se8,se9,se10;
            wire [1:0] se11,se12,se13,se14,se15;
   dff4 dff00(clk,rst,survivor,s1);
   dff4 dff01(clk,rst,s1,s2);
   dff4 dff0(clk,rst,s2,s3);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
dff4 dff1(clk,rst,s3,s4);
dff4 dff2(clk,rst,s4,s5);
dff4 dff3(clk,rst,s5,s6);
dff4 dff4(clk,rst,s6,s7);
dff4 dff5(clk,rst,s7,s8);
dff4 dff6(clk,rst,s8,s9);
dff4 dff7(clk,rst,s9,s10);
dff4 dff8(clk,rst,s10,s11);
dff4 dff9(clk,rst,s11,s12);
dff4 dff10(clk,rst,s12,s13);
dff4 dff11(clk,rst,s13,s14);
dff4 dff12(clk,rst,s14,s15);
dff4 dff13(clk,rst,s15,s16);
dff4 dff14(clk,rst,s16,s17);
dff4 dff15(clk,rst,s17,s18);
dff4 dff16(clk,rst,s18,s19);
dff4 dff17(clk,rst,s19,s20);
dff4 dff18(clk,rst,s20,s21);
dff4 dff19(clk,rst,s21,s22);
dff4 dff20(clk,rst,s22,s23);
dff4 dff21(clk,rst,s23,s24);
dff4 dff22(clk,rst,s24,s25);
dff4 dff23(clk,rst,s25,s26);
dff4 dff24(clk,rst,s26,s27);
dff4 dff25(clk,rst,s27,s28);
dff4 dff26(clk,rst,s28,s29);
dff4 dff27(clk,rst,s29,s30);
dff4 dff28(clk,rst,s30,s31);
mux4_1 m0(clk,rst,s1,2'b0,se1);
mux4_1 m1(clk,rst,s3,se1,se2);
mux4_1 m2(clk,rst,s5,se2,se3);
mux4_1 m3(clk,rst,s7,se3,se4);
mux4_1 m4(clk,rst,s9,se4,se5);
mux4_1 m5(clk,rst,s11,se5,se6);
mux4_1 m6(clk,rst,s13,se6,se7);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
mux4_1 m7(clk,rst,s15,se7,se8);
mux4_1 m8(clk,rst,s17,se8,se9);
mux4_1 m9(clk,rst,s19,se9,se10);
mux4_1 m10(clk,rst,s21,se10,se11);
mux4_1 m11(clk,rst,s23,se11,se12);
mux4_1 m12(clk,rst,s25,se12,se13);
mux4_1 m13(clk,rst,s27,se13,se14);
mux4_1 m14(clk,rst,s29,se14,se15);
mux4 m15(clk,rst,s31,se15,dout);
endmodule
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ค

โปรแกรมทดสอบและแสดงผล (Visual Basic)

// ประกาศตัวแปร

```
Option Explicit
Dim SNR As Integer
Private Const Rate As Boolean = False
Dim conDisplay As Boolean
Dim EbNo, EbNo3 As Integer
Dim UncodeBER(15) As Double
Dim CodeBER(15) As Double
Dim CodeBER3(15) As Double
Dim no As Integer
Dim StopEbNo As Integer
Dim X, Y, Xc, Yc As Double
Dim UncodeError As Long
Dim CodeError As Long
Dim CodeBERtmp, UnCodeBERtmp As Double
Dim EbNoRx As Single
Private Const K3 As Boolean = False
```

// โปรแกรมย่อยสำหรับกำหนดค่าเริ่มต้น Eb/No

```
Private Sub cmdEnable_Click()
PictureGraph.Cls
For no = 0 To 15
shrUncode(no).Visible = False
shrCode2(no).Visible = False
shrCode3(no).Visible = False
Next
If Option1.Value = True Then
MSComm.Output = Chr(SNR)
shrCode3(0).Visible = False
Else
MSComm.Output = Chr(SNR + 16)
shrCode2(0).Visible = False
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
End If
cmdEnable.Enabled = False
Timer1.Enabled = True
End Sub
// โปรแกรมย่อยสำหรับกำหนดค่าเริ่มต้นการสร้างกราฟเปรียบเทียบระหว่างข้อเข้ารหัสและไม่เข้ารหัส
Private Sub cmdTest_Click()
PictureGraph.Visible = True
PictureDiagram.Visible = False
For no = 0 To 15
shrUncode(no).Visible = False
shrCode2(no).Visible = False
shrCode3(no).Visible = False
Next
// กำหนดค่า Eb/No และ SNR เท่ากับ 0
EbNo = 0
SNR = 0
PictureGraph.Cls
If cmdTest.Caption = "Test" Then
cmdTest.Caption = "Stop"
Timer1.Enabled = False
Timer2.Enabled = True
Else
cmdTest.Caption = "Test"
Timer2.Enabled = False
Timer1.Enabled = True
MSComm.Output = Chr(0)
Text1.Text = "0"
SNR = 0
End If
End Sub
// โปรแกรมย่อยสำหรับเปลี่ยนรูปการแสดงผลระหว่างกราฟและไดอะแกรม
Private Sub Command3_Click()
If conDisplay = True Then
Command3.Caption = "Graph"
PictureGraph.Visible = False
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PictureDiagram.Visible = True
conDisplay = False
Else
Command3.Caption = "Diagram"
PictureDiagram.Visible = False
PictureGraph.Visible = True
conDisplay = True
End If
End Sub
```

// โปรแกรมย่อยสำหรับกำหนดค่าเริ่มต้นเมื่อเข้าสู่โปรแกรม

```
Public Sub Form_Load()
Label4.Visible = False
Shape3.Visible = False
cmdEnable.Caption = "Start"
Option1.Enabled = False
Option1.Visible = False
Option2.Enabled = False
Option2.Visible = False
```

// กำหนดให้มีการแสดงไดอะแกรมเมื่อเข้าสู่โปรแกรม

```
PictureGraph.Visible = False
PictureDiagram.Visible = True
PictureGraph.Top = 56
PictureGraph.Width = 753
```

// เปิดการเชื่อมต่อ RS-232

```
MSComm.PortOpen = True
MSComm.Output = Chr(0)
For no = 1 To 15
Load shrUncode(no)
Load shrCode2(no)
Load shrCode3(no)
Next
End Sub
```

// โปรแกรมย่อยสำหรับเรียกดูรายละเอียดเกี่ยวกับโปรแกรมนี้

```
Private Sub mnuAbout_Click()
   Form1.WindowState = 1          Load frmAbout
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
End Sub
// โปรแกรมย่อยในการจบการทำงาน
Private Sub mnuExit_Click()
End
End Sub
// โปรแกรมย่อยสำหรับเชื่อมต่อคอมพิวเตอร์กับ FPGA ผ่าน RS-232 port
Private Sub MSComm_OnComm()
// กำหนดค่าตัวแปร Local
Dim tmp As String
Dim data As String
Dim error As String
Dim C1, C2, C3, U1, U2, U3, S As String
Dim Ci1, Ci2, Ci3, Ui3, Ui1, Ui2 As Long
Dim tmp_tx As Byte
// รับข้อมูลมาจากเอฟพีจีเอ
If MSComm.CommEvent = 2 Then
tmp = MSComm.Input
U1 = Left(tmp, 1)
U2 = Mid(tmp, 2, 1)
U3 = Mid(tmp, 3, 1)
C1 = Mid(tmp, 4, 1)
C2 = Mid(tmp, 5, 1)
C3 = Mid(tmp, 6, 1)
S = Mid(tmp, 7, 1)
Ui1 = Asc(U1)
Ui2 = Asc(U2)
Ui3 = Asc(U3)
Ci1 = Asc(C1)
Ci2 = Asc(C2)
Ci3 = Asc(C3)
EbNoRx = Asc(S)
UncodeError = Ui1 + (Ui2 * 256) + (Ui3 * 65536)
CodeError = Ci1 + (Ci2 * 256) + (Ci3 * 65536)
If CodeError > 1 Then 'Potect Over flow
CodeBERtmp = CodeError / 2062500
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Else
CodeBERtmp = 0.000001
End If
UnCodeBERtmp = UncodeError / 2062500
If Rate = False Then
EbNoRx = Asc(S) / 2
Text4.Text = "K = 3, Rate 1/2 "
Else
EbNoRx = (Asc(S) - 16) / 2
Text4.Text = "K = 3, Rate 1/3 "
End If
```

// กำหนดตำแหน่งพิกัดกราฟ BER ต่อ Eb/No

```
Y = -(Log(UnCodeBERtmp) / Log(10)) * 88
X = (EbNoRx * 82)
Yc = -(Log(CodeBERtmp) / Log(10)) * 88
Xc = (EbNoRx * 82)
```

// รายงานผลคำนวณหา BER

```
Text2.Text = "At Eb/No: " & EbNoRx & "  dB" & vbCrLf & _
"No. of Data: 2062500 bits" & vbCrLf & _
"No. of Uncode Error: " & UncodeError & " bits" & vbCrLf & "Uncode BER: " & UnCodeBERtmp & vbCrLf & _
"No. of Code Error: " & CodeError & " bits" & vbCrLf & "Code BER: " & CodeBERtmp
Text3.Text = "At Eb/No: " & EbNoRx & "  dB" & vbCrLf & _
"No. of Data: 2062500 bits" & vbCrLf & _
"No. of Uncode Error: " & UncodeError & " bits" & vbCrLf & "Uncode BER: " & UnCodeBERtmp & vbCrLf & _
"No. of Code Error: " & CodeError & " bits" & vbCrLf & "Code BER: " & CodeBERtmp
If Timer2.Enabled = False Then
shrUncode(0).Top = 8 + Y
shrUncode(0).Left = 40 + X
End If
End If
End Sub
```

// กำหนด Rate ของรหัสคอนโวลูชัน

```
Private Sub Option1_Click()
cmdEnable.Enabled = True
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Private Sub Option2_Click()
cmdEnable.Enabled = True
End Sub
// โปรแกรมย่อยสำหรับแสดงจุดกระพริบของกราฟเริ่มต้น ณ ค่า Eb/No เริ่มต้น
Private Sub Timer1_Timer()
If shrUncode(0).Visible = True Then
shrUncode(0).Visible = False
shrCode2(0).Visible = False
shrCode3(0).Visible = False
Else
shrUncode(0).Visible = True
If Option1.Value = True Then
shrCode2(0).Top = 8 + Yc
shrCode2(0).Left = 40 + Xc
shrCode2(0).Visible = True
Else
shrCode3(0).Top = 8 + Yc
shrCode3(0).Left = 40 + Xc
shrCode3(0).Visible = True
End If
End If
End Sub
// โปรแกรมย่อยlesiy[สร้างกราฟเมื่อมีการคลิกปุ่ม Test
Private Sub Timer2_Timer()
// ส่งค่าเริ่มต้น Eb/No ออกไปเอฟพีจีเอ
MSComm.Output = Chr(EbNo)
// กำหนดตำแหน่งกราฟ
If EbNo < 16 Then
shrUncode(EbNo).Top = 8 + Y
shrUncode(EbNo).Left = 40 + X
shrCode2(EbNo).Top = 8 + Yc
shrCode2(EbNo).Left = 40 + Xc
// โปรแกรมวาดกราฟ
If EbNo > 0 Then
shrUncode(EbNo).Visible = True
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
shrCode2(EbNo).Visible = True
If EbNo > 1 Then
PictureGraph.Line (shrUncode(EbNo - 1).Left + 5, shrUncode(EbNo - 1).Top + 5)-(shrUncode(EbNo).Left + 5,
shrUncode(EbNo).Top + 5), RGB(255, 0, 0)
PictureGraph.Line (shrCode2(EbNo - 1).Left + 5, shrCode2(EbNo - 1).Top + 5)-(shrCode2(EbNo).Left + 5,
shrCode2(EbNo).Top + 5), RGB(0, 0, 255)
End If
End If
ElseIf K3 = True Then
EbNo3 = EbNo - 16
shrCode3(EbNo3).Top = 8 + Yc
shrCode3(EbNo3).Left = 40 + Xc
If EbNo > 16 Then
shrCode3(EbNo3).Visible = True
If EbNo > 17 Then
PictureGraph.Line (shrCode3(EbNo3 - 1).Left + 5, shrCode3(EbNo3 - 1).Top + 5)-(shrCode3(EbNo3).Left + 5,
shrCode3(EbNo3).Top + 5), RGB(0, 255, 0)
End If
End If
End If
EbNo = EbNo + 1
If EbNo > (31 - 16) Then
Timer2.Enabled = False
cmdEnable.Enabled = True
cmdTest.Caption = "Test"
Timer1.Enabled = False
MSComm.Output = Chr(0)
SNR = 0
Text1.Text = "0"
End If
End Sub
```

// โปรแกรมรับค่า Eb/No เป็นอินพุต

// ลดค่า Eb/No

```
Private Sub UpDown1_DownClick()
If SNR > 0 Then
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SNR = SNR - 1
Text1.Text = Val(Text1.Text) - 0.5
cmdEnable.Enabled = True
End If
End Sub
```

// เพิ่มค่า Eb/No

```
Private Sub UpDown1_UpClick()
If SNR < 15 Then
SNR = SNR + 1
Text1.Text = Val(Text1.Text) + 0.5
cmdEnable.Enabled = True
End If
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ง

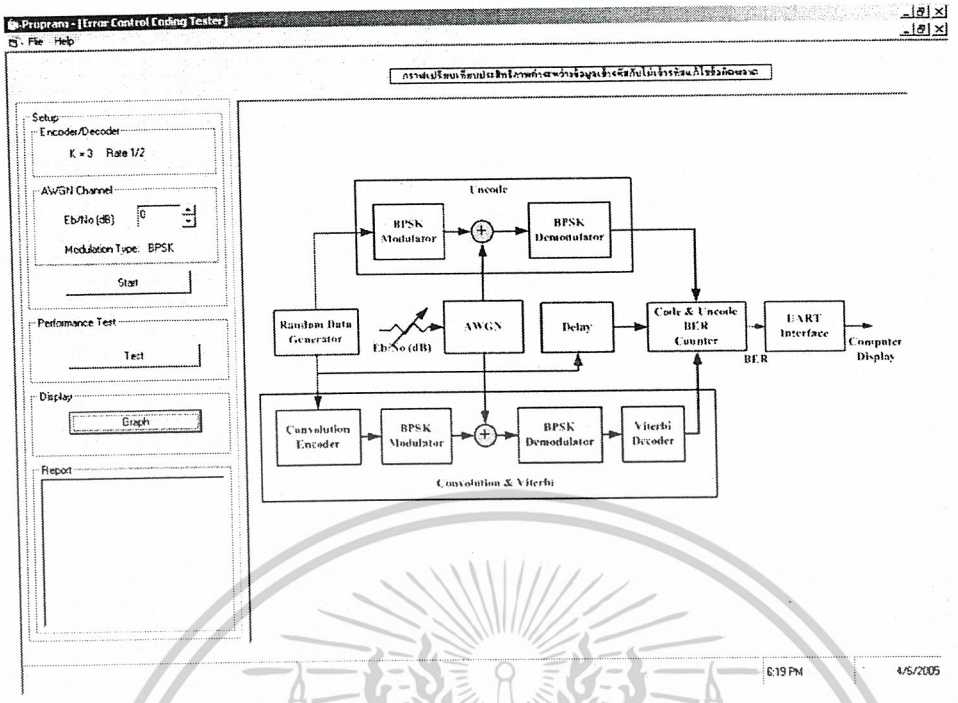
การใช้งานโปรแกรมควบคุมและแสดงผลทางคอมพิวเตอร์

โปรแกรมส่วนควบคุมและแสดงผลจะทำหน้าที่ควบคุมและแสดงผล โดยเชื่อมต่อระหว่างคอมพิวเตอร์กับบอร์ด FPGA ผ่านพอร์ต RS-232 เมื่อเข้าสู่โปรแกรมจะปรากฏดังรูปที่ 1 การใช้งานสามารถทำเรียกดูผลการทดสอบได้ 2 ลักษณะคือ ที่ค่า Eb/No ตั้งแต่ 0-7 dB และเฉพาะค่าที่ผู้ใช้กำหนด

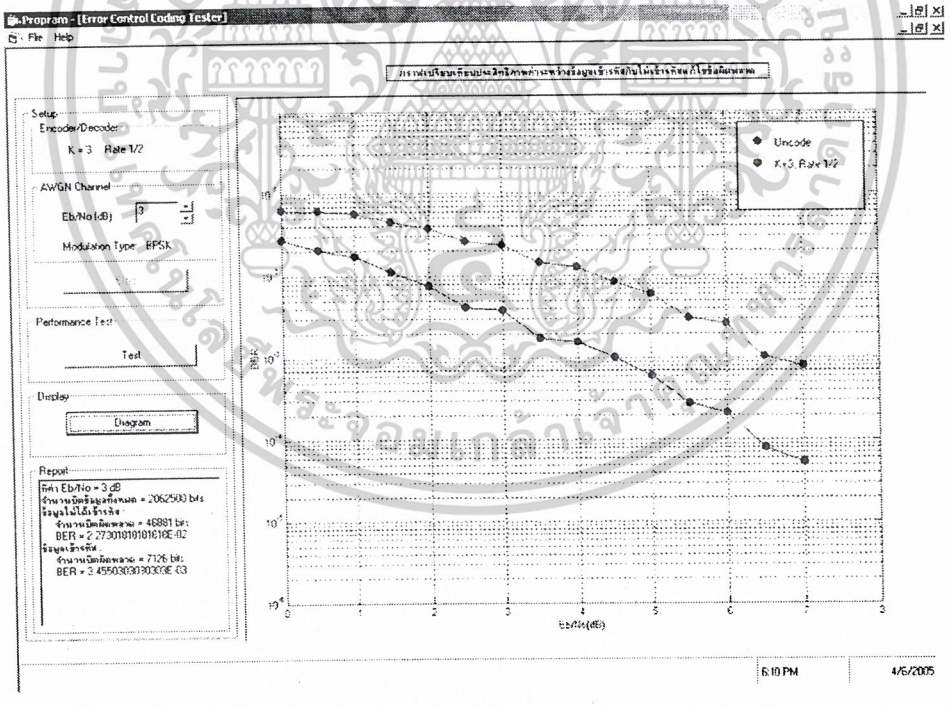
- ผลการทดสอบที่ค่า Eb/No ตั้งแต่ 0-7 dB ดังรูปที่ 2 มีขั้นตอนดังนี้
    1. กำหนดค่า Eb/No
    2. คลิกปุ่ม Start
    3. คลิกปุ่ม Test
- ผลการทดสอบเฉพาะค่า Eb/No ที่ผู้ใช้กำหนด ดังรูปที่ 3 มีขั้นตอนดังนี้
    1. กำหนดค่า Eb/No
    2. คลิกปุ่ม Start
    3. คลิกปุ่ม Diagram
- ปุ่ม Diagram ไว้สลับการแสดงผลระหว่าง diagram ของชุดทดสอบและกราฟผลการทดสอบ

นอกจากแสดงผลเป็นกราฟแล้วยังมีหน้าต่าง Report แสดงผลการการทดสอบอีกด้วย โดยมีรายละเอียด ดังรูปที่ 4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

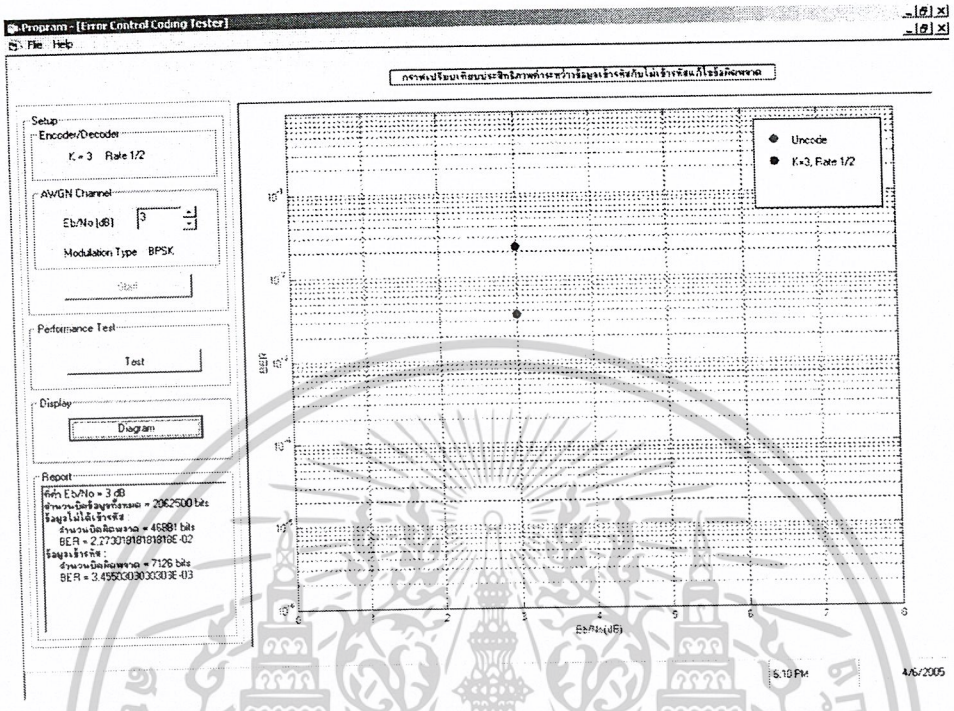


รูปที่ 1 แสดงหน้าต่างแรกเมื่อเรียกโปรแกรมใช้งาน

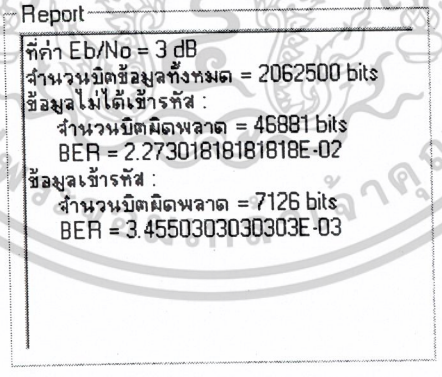


รูปที่ 2 แสดงกราฟเปรียบเทียบค่า BER ต่อ Eb/No ตั้งแต่ 0 – 7 dB ระหว่างข้อมูลเข้ารหัสคอนโวลูชันและไม่มีการเข้ารหัส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3 แสดงกราฟเปรียบเทียบค่า BER ต่อ Eb/No ที่ค่า Eb/No เท่ากับ 3 dB ระหว่างข้อมูลเข้ารหัสคอนโวลูชันและไม่มีการเข้ารหัส



รูปที่ 4 แสดงหน้าต่างรายงานผลเปรียบเทียบระหว่างข้อมูลเข้ารหัสคอนโวลูชันและไม่มีการเข้ารหัส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้