การเข้ารหัสเสียงสำหรับการสื่อสาร

VOICE ENCODING FOR COMMUNICATION

นายกฤษฎา จิตราภิรมย์
นายวิศิษฐ์ ศรีเบญจรัตน์
นายสมบัติ ใบดำรงศักดิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2546

VOICE ENCODING FOR COMMUNICATION

KRITSADA JITTRAPIROM

WISIT SRIBENJARAT

SOMBAT BAIDAMRONGSAK

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KNIG MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2003

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์ การเข้ารหัสเสียงสำหรับการสื่อสาร

VOICE ENCODING FOR COMMUNICATION

นักศึกษาผู้จัดทำ	นายกฤษฎา	จิตราภิรมย์	รหัสประจำตัว	44015419
	นายวิศิษฐ์	ศรีเบญจรัตน์	รหัสประจำตัว	44015444
	นายสมบัติ	ใบดำรงศักดิ์	รหัสประจำตัว	44015448

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2546

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รศ.ดร. ฟูศักดิ์ ชีวฉุวิทย์	

วัน/เดือน/ปี ที่สอบ วันที่ 24 มีนาคม พ.ศ.2547

สถานที่สอบ ณ ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

ภาควิชารับรองแล้ว

(รศ.ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์	การเข้ารหัสเสียงสำหรับการสื่อสาร VOICE ENCODING FOR COMMUNICATION
นักศึกษาผู้จัดทำ	นายกฤษฎา จิตราภิรมย์ นายวิศิษฐ์ ศรีเบญจรัตน์ นายสมบัติ ใบดำรงศักดิ์
อาจารย์ที่ปรึกษา	รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์
ปีการศึกษา	2546

บทคัดย่อ

การสื่อสารข้อมูลเป็นที่ยอมรับกันทั่วไปว่ามีความสำคัญต่อชีวิตประจำวันมาก ซึ่งแนวโน้มของการแลกเปลี่ยนข้อมูลส่วนตัวมักจะส่งผ่านโครงข่ายของการสื่อสาร จุดนี้เองเป็นสาเหตุที่จะทำให้เกิดการรั่วไหลของข้อมูลเนื่องจากทำการจารกรรมข้อมูล ซึ่งเป็นความเสี่ยงต่อความปลอดภัยของข้อมูลและการละเมิดสิทธิส่วนบุคคล ดังนั้นในปริญญานิพนธ์นี้ได้ออกแบบทำชุดเข้ารหัสเสียงสำหรับการสื่อสาร โดยใช้ชิป FPGA โดยเทคนิคการเข้ารหัสนี้ประกอบด้วยการประยุกต์ใช้ซีนโครมของบล็อคโค้ดเชิงเส้น และการกำเนิดแบบสุ่ม ผลของการเข้ารหัสนี้ได้แสดงให้เห็นอย่างชัดเจนว่าสามารถปกปิดข้อมูลได้ และสามารถถอดรหัสออกมาฟังได้อย่างสมบูรณ์

Thesis Title	**Voice encoding for communication**
Authors	**Mr.Kritsada Jitrapirom**
	Mr.Wisit Sribenjarat
	Mr.Sombat Baidamrongsak
Thesis Advisor	**Assoc.Prof.Dr.Fusak Chevasuvit**
Year	**2003**

ABSTRACT

Data communication is accepted as an important part of our everyday life. The prend of the proprietary data exchange is through the communication networks. It will be caused the loss of data information due to data trapping , which will be remained the leading risk to data security and privacy. Therefore, This paper proposes designing a voice encoder for communication by using FPGA CHIP. The syndrome information of linear block code are applied for encoding. The result of encoding data show clearly that the original data will be hidden and can decode to original data completely.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีเพราะได้รับความเมตตาจากหลายๆท่าน จึงขอขอบพระคุณ รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์ ที่ได้ให้คำแนะนำและเป็นที่ปรึกษากับผู้จัดทำตลอดมา อีกทั้งยังเอื้อเฟื้ออุปกรณ์และเครื่องมือต่างๆ ในการทำปริญญานิพนธ์นี้ ผู้จัดทำรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ อาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่าน ที่ให้คำแนะนำอันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

และที่ลืมเสียมิได้คือ ขอกราบขอบพระคุณพระคุณคุณแม่ อันเป็นที่รักยิ่ง ที่สนับสนุนและเป็นแรงบันดาลใจในการทำปริญญานิพนธ์นี้ฉบับนี้

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์นี้ ผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากปัจจุบันการติดต่อสื่อสารทางโทรศัพท์นั้นง่ายต่อการถูกดักฟัง ในกรณีที่ผู้ใช้โทรศัพท์ต้องการที่จะติดต่อสื่อสารทางโทรศัพท์และข่าวสารนั้นสำคัญมาก ทำให้ผู้ใช้โทรศัพท์ต้องพยายามหาวิธีการในการป้องกันข้อมูลข่าวสารของตนเองให้รอดพ้นจากการถูกดักฟัง วิธีการหนึ่งที่ถูกนำมาใช้ก็คือ การเข้ารหัสข้อมูล(ENCODE) ก่อนทำการส่งข้อมูลและทำการถอดรหัส (DECODE) ข้อมูลที่ได้รับกลับคืนมา ซึ่งทำให้ข้อมูลข่าวสารรอดพ้นจากการถูกดักฟังได้ ถ้าผู้ที่ต้องการดักฟังไม่สามารถทำการถอดรหัสข้อมูลเหล่านั้นออกมาได้ ในปริญญานิพนธ์นี้ได้ใช้หลักการของบล็อกโค้ดเชิงเส้นในการเข้ารหัส และถอดรหัสโดยใช้ซีนโดรมของบล็อกโค้ดเชิงเส้นมาใช้ในการทำชุดเข้ารหัสเพื่อการสื่อสาร

1.2 วัตถุประสงค์

1. เพื่อเป็นการศึกษาหาแนวทางในการออกแบบชุดตัวเข้ารหัส และชุดตัวถอดรหัสและนำมาทำการสร้างเป็นแผงวงจร สำหรับทำงานตามโครงสร้างที่ออกแบบ
2. สามารถนำแผงวงจรชุดตัวเข้ารหัสและชุดตัวถอดรหัสที่ออกแบบไปใช้งานได้

1.3 ขอบเขตการวิจัย

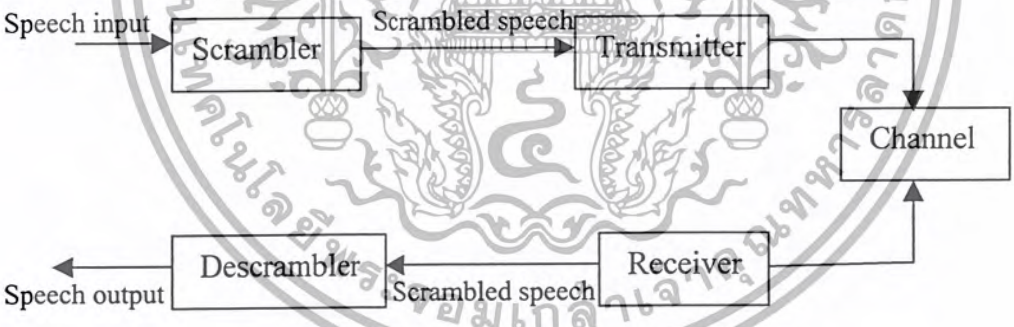
ขอบเขตการวิจัยจะเป็นการสร้างแผงวงจรชุดตัวเข้ารหัสและชุดตัวถอดรหัส โดยอาศัยหลักการของบล็อกโค้ดเชิงเส้น ซึ่งเป็นหลักวิธีการในการนำมาใช้ในการออกแบบวิธีการเข้ารหัสและวิธีการถอดรหัส และนำวิธีการที่ได้นำมาทำการออกแบบสร้างเป็นแผงวงจรชุดสร้างรหัสและแผงวงจรชุดถอดรหัส สำหรับการนำไปใช้งานได้

บทที่ 2
หลักการเข้ารหัสและบล็อกโค้ดเชิงเส้น

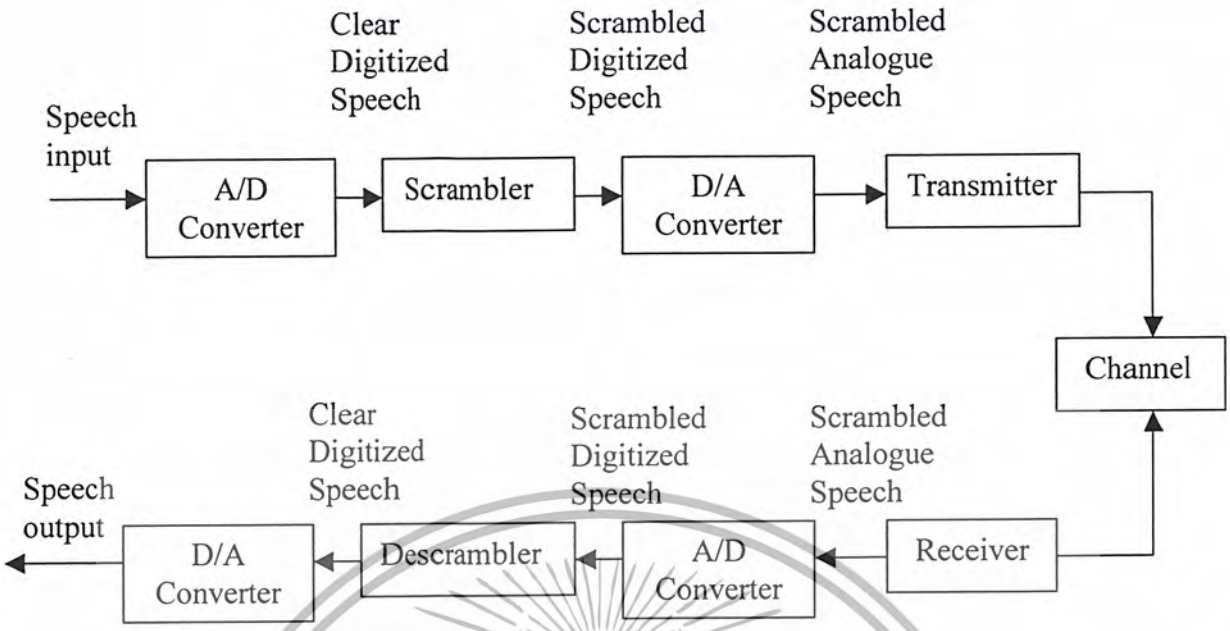
2.1 วิธีการสร้างรหัสลับ

โดยทั่วไปมักมีการกล่าวอ้างถึงวิธีการเข้ารหัสลับ (Scramble) แบบพื้นฐาน 2 วิธี คือ แบบอนาล็อก (Analog) และแบบดิจิตอล (Digital) ส่วนใหญ่เครื่องเข้ารหัสสัญญาณที่มีความซับซ้อนมากๆ มักใช้กระบวนการของการประมวลผลสัญญาณดิจิตอล (Digital Signal Processing) กระบวนการนี้ทำงานโดยการแปลงสัญญาณอนาล็อกไปเป็นสัญญาณดิจิตอลก่อน แล้วจึงทำการเข้ารหัส ส่วนในระบบที่เป็นอนาล็อกโดยส่วนใหญ่จะมีความปลอดภัยค่อนข้างน้อย

ข้อแตกต่างที่เห็นได้ชัดเจนระหว่างการเข้ารหัสแบบอนาล็อกและแบบดิจิตอลคือ รูปแบบที่เกี่ยวกับตัวส่งผ่านซึ่งเป็นอุปกรณ์ที่ใช้ในการส่งสัญญาณที่เข้ารหัสแล้ว วัตถุประสงค์ของระบบที่มีการเข้ารหัสแบบอนาล็อกคือ ส่งผ่านข่าวสารที่มีการเปลี่ยนแปลงอย่างต่อเนื่อง ในทางตรงข้ามระบบเข้ารหัสลับแบบดิจิตอลจะส่งผ่านข่าวสารด้วยสัญญาณที่สามารถนำข่าวสารไปได้เฉพาะตามจำนวนที่จำกัดไว้ ดังแสดงได้ในแผนภาพดังต่อไปนี้

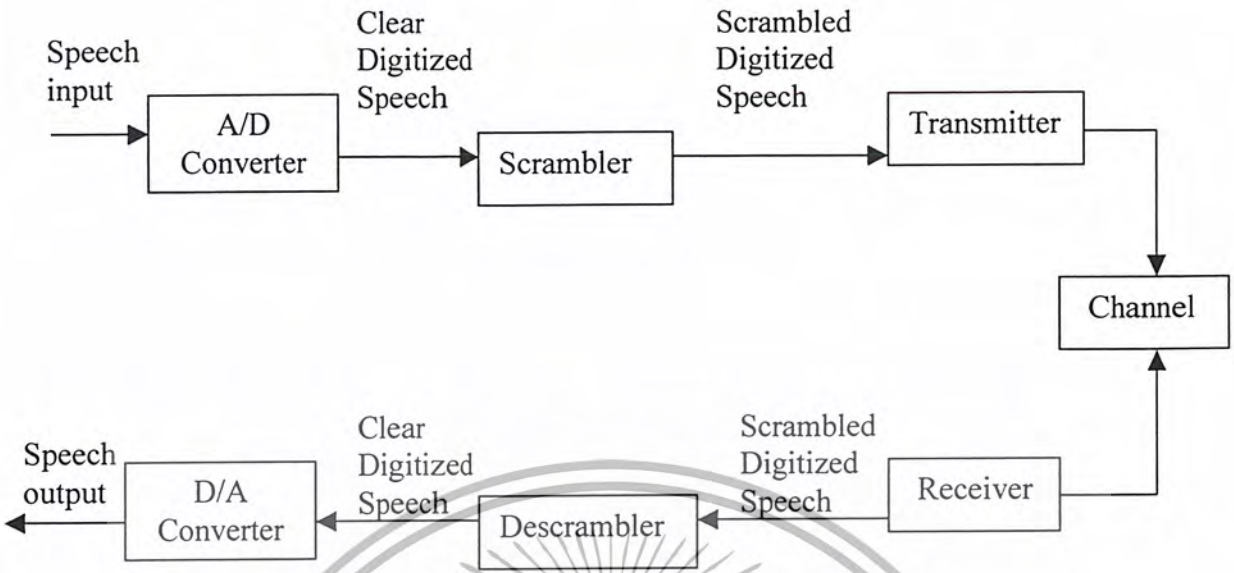


ภาพที่ 2.1 แสดงการเข้ารหัสเสียงแบบอนาล็อกชนิดไม่มีการประมวลผลสัญญาณดิจิตอล



ภาพที่ 2.2 แสดงการเข้ารหัสเสียงแบบอนาล็อกชนิดมีการประมวลผลสัญญาณแบบดิจิตอล

ในภาพที่ 2.1 และ 2.2 แสดงให้เห็นถึงการเข้ารหัสสัญญาณแบบอนาล็อก ความแตกต่างระหว่างทั้ง 2 รูปแบบคือ รูปแบบสัญญาณที่ผ่านการเข้ารหัสแล้ว เช่นในรูปภาพที่ 1 ยังคงมีสัญญาณแบบอนาล็อกทั้งกระบวนการเข้ารหัส และในระบบตามภาพที่ 2.2 สัญญาณถูกเปลี่ยนไปอยู่ในรูปของสัญญาณดิจิตอลก่อนจะทำการเข้ารหัส แล้วถูกเปลี่ยนกลับให้อยู่ในรูปสัญญาณอนาล็อกก่อนทำการส่งผ่าน และหลังการส่งผ่านและหลังการส่งผ่านจะถูกเปลี่ยนกลับให้เป็นสัญญาณดิจิตอลอีกครั้งหนึ่ง แล้วจึงจะทำการถอดรหัสลับและท้ายสุดจะถูกเปลี่ยนให้เป็นสัญญาณอนาล็อกอีกครั้งหนึ่ง ในภาพที่ 2.3 แสดงให้เห็นถึงระบบดิจิตอลอีกประเภทหนึ่ง ความแตกต่างระหว่างระบบนี้และระบบในภาพที่ 2.2 คือไม่มีตัวแปลงสัญญาณจากดิจิตอลไปเป็นอนาล็อก (D/A Converter) ในทันทีก่อนเข้าสู่เครื่องส่งผ่านและไม่มีตัวแปลงสัญญาณจากอนาล็อกไปเป็นดิจิตอลในทันทีหลังผ่านเครื่องรับ โดยจะใช้รูปแบบดังแสดงในภาพที่ 2.3



ภาพที่ 2.3 แสดงการเข้ารหัสเสียงแบบดิจิตอล

2.2 การคำนวณทางคณิตศาสตร์ของรหัสแบบไบนารี

สำหรับการเข้ารหัสในปริญญานิพนธ์ฉบับนี้ จะใช้คณิตศาสตร์ที่มีความซับซ้อนน้อย ทั้งนี้เพื่อให้สามารถนำมาสร้างวงจรฮาร์ดแวร์ได้ง่าย คณิตศาสตร์ที่ว่านี้รู้จักกันในชื่อของ Galois field (1) เนื่องจากคณิตศาสตร์ที่ใช้มีลักษณะเป็นดิจิตอล คือมีลักษณะเป็น 0 กับ 1 บางครั้งคณิตศาสตร์ดังกล่าวนี้ถูกเรียกว่า Binary field ซึ่งเขียนย่อเป็น GF(2) การทำงานของคณิตศาสตร์ดังกล่าวแสดงผลตามตารางที่ 1 โดยการบวกจะมีลักษณะเป็นการทำเอกซ์คลูซีฟออร์ (Exclusive OR)ของลอจิก ส่วนการคูณจะมีจะมีลักษณะเป็นการ AND ของลอจิก ในการเข้ารหัสนี้จะมีการนำเอาเวคเตอร์ของข้อมูลข่าวสาร (Massage) คูณกับเมทริกซ์ที่ทำการออกแบบไว้ ผลลัพธ์ที่ได้จะเกิดจากการ AND และ EX-OR ของข้อมูลที่เป็นลอจิก จากเวคเตอร์ข้อมูลและเมทริกซ์ที่กำหนด

ตารางที่ 2.1 รูปแบบการคำนวณทางคณิตศาสตร์ของรหัส

$0 + 0 = 0$	$0 \times 0 = 0$
$0 + 1 = 1$	$0 \times 1 = 0$
$1 + 0 = 1$	$1 \times 0 = 0$
$1 + 1 = 0$	$1 \times 1 = 1$

2.3 การตรวจสอบพาริตี้ (Parity Check)

เพื่อให้เข้าใจถึงการสร้างรหัสเชิงเส้น ให้ดูตัวอย่างง่ายๆ จากการสร้างรหัสคำ (Code word) เริ่มจากข้อมูลโดยปล่อยให้ผ่านไปไปในรหัสคำโดยตรง และจะมีบิตตามหลังแบบบิตเดียว ซึ่งเป็นการคำนวณจากบิตข้อมูลทั้งหมด มีวิธีการกำหนดบิตสุดท้ายที่ตามหลังมา 2 วิธีคือ

1. กำหนดบิตสุดท้ายเพื่อให้ผลรวมแบบโมดูโล (Modulo) 2 ของบิททั้งหมดในรหัสคำ มีค่าเท่ากับหนึ่ง

2. กำหนดบิตสุดท้ายเพื่อให้ผลรวมแบบโมดูโล 2 ของบิททั้งหมดในรหัสคำ มีค่าเท่ากับศูนย์

ในกรณีแรกรหัสคำมีพาริตี้แบบคี่ (Odd parity) กล่าวคือ จำนวนบิทที่เป็นหนึ่งในรหัสคำเป็นจำนวนคี่ ส่วนในกรณีที่สองมีจำนวนบิทที่เป็นหนึ่งในรหัสคำเป็นคี่ (Even parity) บิทที่เพิ่มขึ้นนอกเหนือจากบิทข้อมูลเรียกว่าพาริตี้เช็ค (Parity check) และอาจเรียกได้ว่าเป็นการตรวจสอบแบบพาริตี้คู่หรือพาริตี้คี่

รหัสแบบพาริตี้คี่และคู่จะแสดงในตารางที่ 2 และ 3 ตามลำดับ สำหรับกรณีที่มีบิทข้อมูล 3 ตำแหน่ง จะสังเกตเห็นว่ารหัสของตารางที่ 2 ไม่มีแถวที่เป็นศูนย์ทั้งหมด ซึ่งต้องเป็นส่วนหนึ่งของรหัสที่เป็นเชิงเส้น

ตารางที่ 2.2 รหัสพาริตี้แบบคี่

ข้อมูลข่าวสาร	ข้อมูลเข้ารหัส
000	0001
001	0010
010	0100
011	0111
100	1000
101	1011
110	1101
111	1110

ดังนั้นการตรวจสอบพาริตี้แบบคี่ทำให้เกิดรหัสแบบไม่เป็นเชิงเส้น ในทางตรงกันข้ามรหัสที่ใช้พาริตี้ในตารางที่ 3 จะเป็นรหัสเชิงเส้น ระบบที่มีการตรวจสอบพาริตี้แบบคู่โดยการเพิ่มบิทที่เป็นพาริตี้นั้นได้จากการบวกแบบโมดูโล 2 ของบิทในรหัสของข้อมูลข่าวสาร ตัวอย่างเช่น

รหัสข้อมูลข่าวสาร 101 เมื่อทำการบวกแบบโมดูโล 2 ของบิทที่มีค่าเป็น 1,0 และ1 จะได้ผลลัพธ์คือ 0 ซึ่งจะเป็นค่าของบิทที่นำมาทำการต่อข้างท้ายของรหัสข้อมูลข่าวสาร กลายเป็น1010

ตารางที่ 2.3 รหัสพาริตี้แบบคู่

ข้อมูลข่าวสาร	ข้อมูลเข้ารหัส
000	0000
001	0011
010	0101
011	0110
100	1001
101	1010
110	1100
111	1111

ในการเพิ่มพาริตี้อีกหนึ่งบิทให้รหัสของข้อมูลข่าวสารนั้น ปกติมักจะนำมาตรวจสอบได้ถ้าบิทผิดไปเพียงบิทเดียวเท่านั้น แต่ไม่สามารถนำมาทำการแก้ไขบิทที่ผิดได้ ในการแก้ไขบิทที่ผิดไปจะต้องใช้วิธีการแก้เข้ารหัสที่เพิ่มพาริตี้ให้มากขึ้น อย่างเช่นการเข้ารหัสแบบบล็อกโค้ดเชิงเส้นที่จะกล่าวต่อไป

2.4 ความสามารถในการตรวจแก้บิทที่ผิดในรหัสเชิงเส้น

ในส่วนนี้จะได้กล่าวถึงคำศัพท์พื้นฐานที่ใช้ในการแก้บิทที่ผิดของรหัสเชิงเส้น

เวท (Weight) ของแฮมมิ่งสำหรับเวคเตอร์ v n-ทูเปิ้ลส์ w(v) ซึ่งหมายถึงผลรวมของจำนวนบิทของรหัสของ v ที่ไม่ได้เป็นศูนย์ ตัวอย่างเช่น ถ้า v = [1 0 1 0 1 1 0 0 0 1] จะได้ w(v) = 5

ให้ u และ v เป็นเวคเตอร์ n- ทูเปิ้ล ค่าระยะห่างระหว่าง u และ v เขียนได้เป็น d(u,v) ระยะห่างของเวคเตอร์ใดๆ คือ จำนวนบิทรหัส “1” ที่แตกต่างกันของเวคเตอร์ทั้งสองเช่น

u = [1 0 0 1 0 1 1 0 0 0 1]

v = [1 1 0 0 1 0 1 0 1 0 1]

จะได้ d(u,v)=5

ถ้านำเวคเตอร์ u และเวคเตอร์ v มาบวกกัน โดยการบวกไบนารีเป็นการทำ EX-OR ดังนั้น

u+v=[0 1 0 1 1 1 0 0 1 0 0]

เวคเตอร์รวมที่ได้ถ้านำมาหาเวทของแฮมมิ่งจะได้ว่า w(u+v)=5 ดังนั้นพอสรุปได้ว่าระยะห่างแบบแฮมมิ่งระหว่างเวคเตอร์ u และ v จะเท่ากับเวทของแฮมมิ่งจากเวคเตอร์รวม กล่าวคือ

$$d(u,v)=w(u+v) \tag{2.1}$$

สำหรับรหัสเชิงเส้นในการหาระยะห่างของแต่ละคู่ของรหัสคำ ระยะห่างที่น้อยที่สุดเขียนย่อเป็น dmin ถ้า u และ v เป็นโค้ดเวคเตอร์สองชุดของรหัสเชิงเส้น โดย u+v ก็ยังเป็นโค้ดเวคเตอร์เพราะเซทของทุกโค้ดเวคเตอร์เป็นซับสเปช (Subspace) ของทุก n- ทูเปิ้ลส์ ดังนั้นจากคำนิยามที่ว่า ระยะห่างระหว่างโค้ดเวคเตอร์ทั้งสองคือ เวทของโค้ดเวคเตอร์ที่ 3 ก็จะได้ระยะห่างน้อยสุดของรหัสเชิงเส้นเท่ากับเวทต่ำสุดของโค้ดเวคเตอร์ที่ไม่เป็นศูนย์ ค่าระยะห่างน้อยสุด และเวทต่ำสุดจะเป็นตัวกำหนดความสามารถในการแก้รหัสบิทที่ผิดของรหัสเชิงเส้น [1]

พิจารณาจากรหัสที่ใช้ส่งโดยให้ v = (v1,v2,……,vn) เป็นโค้ดเวคเตอร์ที่ใช้ส่งและให้ r = (r1,r2……,rm) เป็นเวคเตอร์ที่ได้รับจากการส่งแต่เนื่องจากเวคเตอร์ r ที่รับได้จะเป็นอะไรก็ได้ 2n เวคเตอร์ของ n-ทูเปิ้ลส์ ความแตกต่างระหว่าง r และ v คือ e

$$e = (e1,e2……,en)$$
$$= r+v$$
$$e = (r1,r2,……,rn)+(v1,v2,……,vn)$$
$$= (r1+v1,r2+v2,……rn+vn)$$

ซึ่ง e เป็นรูปแบบของรหัสที่ผิด (error pattern หรือ error vector) เมื่อ ei= ri+vi =1 นั้นก็หมายความว่าโค้ดเวคเตอร์เกิดการผิดพลาดตำแหน่งที่ ith แต่เนื่องจากในหนึ่งโค้ดเวคเตอร์มีรหัสอยู่ n บิทจึงทำให้เกิดความผิดพลาด 2n – รูปแบบที่แตกต่างกัน ไม่นับรูปแบบที่มีทุกบิทเป็นศูนย์ ทางด้านรับตัวถอดรหัสมีหน้าที่ในการตรวจหาโค้ดเวคเตอร์ที่ส่งมาจากโค้ดเวคเตอร์ r ที่รับได้ สำหรับการถอดรหัสโดยใช้วิธีแม็กซิมั่มไลค์รีฮูดนั้น ตัวถอดรหัสจะตรวจสอบว่า v เป็นเวคเตอร์ที่ใช้ในการส่ง ซึ่งจะมีค่าใกล้เคียงกับเวคเตอร์ r โดยอาศัยการดูจากระยะห่างของแฮมมิ่ง ตัวถอดรหัสสามารถทำการแก้ไขรหัสที่ผิดจำนวน t บิทในโค้ดเวคเตอร์ที่รับเข้ามาโดยที่ 2t+2 ตัวถอดรหัสสามารถทำการแก้ทุกรูปแบบที่ผิดไป t บิท จากเวคเตอร์ r ที่รับได้ดังนี้ ให้ v เป็นโค้ดเวคเตอร์ใดๆ ระยะห่างของแมมิ่งระหว่าง u,v และ r จะต้องเป็นไปตามสมการข้างล่างนี้

$$d(v,r) + d(u,r) \geq d(u,v) \tag{2.2}$$

ถ้าสมมุติว่าเกิดรหัสผิดไป t' บิต($t' \leq t$) ดังนั้นระยะห่างของแฮมมิ่งระหว่างโค้ดเวคเตอร์ที่ส่ง V กับโค้ดเวคเตอร์ที่รับ r คือ d(v,r)= t' แต่ $d(u,v) \geq d_{min} \geq 2t + 1$ สมการที่ 2.3จะให้

$$d(u,r) \geq 2t + 1 - t'$$

$$d(u,r) \geq t +$$

$$d(u,r) \geq t' \qquad (2.3)$$

จากสมการที่ 2.3 แสดงให้เห็นว่ารูปแบบของรหัสที่มีบิตผิดไป t บิตหรือน้อยกว่า เวคเตอร์ r ที่รับได้จะเข้าใกล้โค้ดเวคเตอร์ v กว่า โค้ดเวคเตอร์ u ดังนั้นตัวถอดรหัสจะสามารถแก้ไขรหัสผิดพลาดได้ถูกต้องตามความผิดพลาดของบิตที่ผิดไป ตัวถอดรหัสไม่สามารถจะแก้ทุกรูปแบบที่ผิดไป ℓ บิต เมื่อ $\ell \geq t+1$ โดยปกติแล้วการแก้รหัสผิดของโค้ดเชิงเส้นจะทำได้เมื่อ

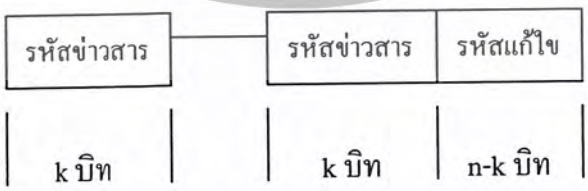
$$t = (d_{min} - 1) / 2 \qquad (2.4)$$

โดย $t = (d_{min} - 1) / 2$ เป็นค่าจำนวนเต็มไม่คิดทศนิยม และสามารถตรวจสอบรหัสผิดพลาดที่เกิดขึ้นถึง $(d_{min} - 1)$ บิตในแต่ละรหัสคำ [1]

2.5 บล็อกโค้ดเชิงเส้น (Linear block code)

จากรหัสข่าวสาร (Massage)ที่แต่ละบล็อกมีขนาด k บิต ซึ่งพบว่าจะให้ข่าวสารที่แตกต่างกันได้ถึง $2^k - 1$ ข่าวสาร (ยกเว้นบล็อกที่มีรหัสข่าวสารเป็นศูนย์หมดจะไม่มีการนำมาใช้) แต่ละบล็อกข่าวสารจะถูกนำมาเข้ารหัสเป็นบล็อกขนาด n บิต โดยจะมี n – k บิต ที่เพิ่มเข้าไปให้รหัสข่าวสาร บิตเหล่านี้ที่เพิ่มเข้าไปในแต่ละบล็อกจะเป็นพาริตี้หรือบางทีเรียกว่ารหัสแก้ไข ที่จะถูกนำมาใช้ในการตรวจสอบและตรวจสอบบิตที่ผิดไป และค่าของ n – k บิตจะขึ้นกับรหัสข่าวสารโดยตรง

บล็อกข่าวสารขนาด n บิตที่ได้จากการเข้ารหัสนี้จะเรียกว่ารหัสคำ (Code word) ถ้าหากว่ารหัสคำมีบิตของข่าวสารเดิมปรากฏอยู่ใน k บิตเริ่มต้นรหัสนั้น จะเรียกว่าซิสเต็มเมทริกซ์ (Systematic code) ยิ่งไปกว่านั้นถ้าหากแต่ละรหัสคำจากจำนวน 2^k รหัสคำที่เข้ารหัสไว้ เกิดจากการรวมกันของ k เวคเตอร์ของรหัสแบบอิสระเชิงเส้น [11] รูปแบบของบล็อกโค้ดเชิงเส้นแสดงได้ด้วยภาพที่ 2.4



ภาพที่ 2.4 แสดงรูปรหัสคำของบล็อกโค้ดเชิงเส้น

2.5.1 เมทริกซ์ตัวกำเนิด (Generator matrix)

สำหรับซับสเปซ S ของ Vn และแต่ละ n-ทูเปิ้ลส์ (Tuples) ของ S เป็นการรวมเชิงเส้นของ v1,v2,…..,vk กล่าวคือ

$$u = m_1v_1 + m_2v_2 + \ldots\ldots + m_kv_k \tag{2.5}$$

เมื่อ $m_i = 0$ สำหรับ i = 1,2,….., k ซับสเปสนี้มีขนาด k มิติของ Vn ซึ่งประกอบด้วย 2^k ของ n- ทูเปิ้ลส์ จากข้อกำหนดที่กล่าวมาซึ่งสามารถอธิบายถึงรหัสเชิงเส้นของ 2^k รหัสคำโดยเซ็ตของ k โค็ดเวคเตอร์ที่เป็นข้อกำหนดอิสระเชิงเส้น ถ้าจัด k รหัสคำเป็นอิสระต่อกันได้เมทริกซ์ k×n

$$G = \begin{bmatrix} v1 \\ v2 \\ vk \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ v_{k1} & v_{k2} & \cdots & v_{kn} \end{bmatrix} \tag{2.6}$$

เมื่อ vi = (vi1,vi2,vi3,…..vin) สำหรับ i = 1,2,…..,k ให้ m = (m1,m2,…….mk) เป็นบล็อกของข่าวสาร รหัสคำจะได้จาก

$$\begin{aligned} u &= mG \\ &= (m_1,m_2,\ldots\ldots..m_k) \begin{bmatrix} v1 \\ v2 \\ vk \end{bmatrix} \\ &= m_1v1 + m_2v_2 + \ldots\ldots\ldots + m_kv_k \end{aligned} \tag{2.7}$$

ดังนั้นรหัสที่สอดคล้องกับชุดข่าวสาร (m1,m2,……..,mk)เกิดจากการรวมแบบเชิงเส้นของแถวใน G กลุ่มแถวต่าง ๆ ของเมทริกซ์ G จะเป็นตัวผลิตรหัสเชิงเส้นและเราเรียกเมทริกซ์ G ว่าเมทริกซ์ตัวกำเนิดของรหัส รหัสเชิงเส้นที่กล่าวนี้เรียกว่ารหัส (n ,k) โดยในแต่ละบล็อกจะมีข่าวสารอยู่ k บิต ที่ถูกเข้ารหัสเป็นรหัสคำที่มีความยาวขนาด n บิต

ลักษณะของเมทริกซ์ตัวกำเนิดขนาด k × n ที่ใช้สร้างรหัสเชิงเส้น (n ,k) แสดงได้ดังสมการที่ (2.8)

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \ldots 0 & p11 & p12 \ldots p1_{,n-k} \\ 0 & 1 & 0 & 0 \ldots 0 & p21 & p22 \ldots p2_{,n-k} \\ 0 & 0 & 1 & 0 \ldots 0 & p31 & p32 \ldots p3_{,n-k} \\ 0 & 0 & 0 & 0 \ldots 1 & pk1 & pk2 \ldots pk_{,n-k} \end{bmatrix} \tag{2.8}$$

โดย $P_{ij} = 1$ หรือ 0 ให้ I_k เป็นเมทริกซ์เอกลักษณ์ (Identity matrix) ขนาด k×k และให้ P เป็นเมทริกซ์ขนาด $k \times (n-k)$ ที่มีอิลิเมนท์เป็น P_{ij} ดังนั้นเมทริกซ์ตัวกำเนิดของรหัสระบบเขียนใหม่ได้เป็น

$$G = [I_{k:}P]$$

พิจารณาถึงบล็อกของข่าวสาร m = (m1,m2,…….mk) เมื่อได้เมทริกซ์ตัวกำเนิดของสมการ (2.8)จะได้โค๊ดเวกเตอร์เป็น

$$u = (u_1,u_2,u_3,\ldots,u_n)$$

$$= (m_1,m_2,\ldots..,m_k)G$$

$$= (m1,m2,\ldots..,mk) \begin{bmatrix} 1 & 0 & 0 & 0 \ldots 0 & p11 & p12 & \ldots & p1k \\ 0 & 1 & 0 & 0 \ldots 0 & p21 & p22 & \ldots & p2k \\ 0 & 0 & 0 & 0 \ldots 1 & pk1 & pk2 & \ldots & pk,n\text{-}k \end{bmatrix} \quad (2.9)$$

จากการคูณของเมทริกซ์จะได้

$$u_i = m_i \qquad \text{สำหรับ } I = 1,2,\ldots\ldots,k \quad (2.10a)$$

และ

$$u_{k+j} = p_{1j}m_1 + p_{2j}m_2 + \ldots\ldots\ldots\ldots p_{kj}m_k \quad (2.10b)$$

สำหรับ $j = 1,2,\ldots\ldots,n-k$ จากสมการที่ 2.10aและ 2.10b จะพบว่ารหัส k บิตแรกของรหัสคำคือ รหัสของข่าวสาร ส่วน (n – k) บิตหลังเป็นฟังก์ชันเชิงเส้นของรหัสข่าวสารซึ่งเรียกว่ารหัสแก้ไข (n – k) บิตของ u หรือ รหัสตรวจสอบพาริตี้ (Parity check digits) ของรหัสคำ สมการที่ 2.10b เรียกว่าสมการพาริตี้ของรหัส

2.5.2 เมทริกซ์ในการตรวจสอบพาริตี้ (Parity check matrix)[8]

จากที่กล่าวว่าเมทริกซ์ G ขนาด k×n จะมีเมทริกซ์ H ขนาด (n – k) × n ซึ่งโรว์สเปซของ G จะตั้งฉากอยู่กับ H อินเนอร์โปรดักท์ของเวกเตอร์ในโรว์สเปซของ G กับแถวของ H จะเป็นศูนย์

$$H = \begin{bmatrix} h_1 \\ h_2 \\ h_{n-k} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & \ldots & h_{1n} \\ h_{21} & h_{22} & \ldots & h_{2n} \\ h_{n-k,1} & h_{n-k,2} & \ldots & h_{n-k,n} \end{bmatrix}$$

และให้ $u = (u_1,u_2,\ldots\ldots,u_n)$ เป็นเวคเตอร์ในโรว์สเปซของ G จะได้

$$uH^T = (0 \quad 0 \ \ldots \ 0) \tag{2.12}$$

หรือ

$$u.h_i = u_i h_{i1} + u_2 h_{i2} + \ldots\ldots\ldots + u_n h_{in} = 0 \tag{2.13}$$

สำหรับ $u = (u_1, u_2, \ldots\ldots, u_n)$ จึงสรุปได้ว่าu จะเป็นรหัสคำที่ได้จาก G ถ้าเพียงแต่ $uH^T = 0$ เมทริกซ์ H นี้เรียกว่าเมทริกซ์ในการตรวจสอบพาริตี้ หรือเรียกย่อๆ ว่าพาริตี้เมทริกซ์ ถ้าเมทริกซ์ตัวกำเนิดของรหัสได้มาจากสมการที่ 2.8 พาริตี้เมทริกซ์ ของรหัสคือ

$$H = \begin{bmatrix} p11 & p21 & \ldots & p_{k1} & 1 & 0 & 0 & \ldots & 0 \\ p12 & p22 & \ldots & p_{k2} & 0 & 1 & 0 & \ldots & 0 \\ p1_{,n-k} & p2_{,n-k} & \ldots & pk_{,n-k} & 0 & 0 & 0 & \ldots & 1 \end{bmatrix}$$

$$= [P^T I_{n-k}]$$

P^T เป็นทรานสโพส (Transpose) ของเมทริกซ์ P สมการพาริตี้ 2.10 b ได้จากเมทริกซ์ H นั่นคือ $u = (u_1, u_2, \ldots\ldots, u_n)$ เป็นรหัสคำของรหัสข้อมูล $m = (m1, m2, \ldots\ldots mk)$ เมื่อ $ui = mi$ สำหรับ $i = 1,2,\ldots.k$ แต่

$$uH^T = 0$$

จะได้ว่า

$$u_{k+j} = p_{1j}u_1 + p_{2j}u_2 + \ldots\ldots p_{kj}u_k$$

เมื่อ $j = 1,2\ldots.,n\text{-}k$ ซึ่งสมการข้างบนเป็นสมการเดียวกันกับสมการที่2.10b ในการออกแบบรหัสเชิงเส้นนั้นแมทริกซ์ P จะถูกเลือกเพื่อให้มีคุณสมบัติในการแก้บิทที่ผิด

2.6 ซีนโดรม

บล็อกของรหัสคำ n บิทเมื่อทำการส่งออกไปตัวกลางจะเกิดสัญญาณรบกวนทำให้บางบิทของข้อมูลผิดไป ซึ่งรูปแบบของบิทที่ผิดไปบางครั้งเรียกว่าโคเซ็ท (Coset) มีหลายรูปแบบถ้าหาก u เป็นรหัสคำที่ต้องการส่งโดย u มีระยะห่างต่ำสุดตามเงื่อนไขสมการ (2.4) และe_i เป็นรูปแบบของบิทที่ผิดไปในระหว่างการติดต่อสื่อสาร ทางด้านรับจะได้รับรหัสคำเป็น r กล่าวคือ

$$r = e_i \oplus u \tag{2.16}$$

การคำนวณหาซีนโดรมของรหัสคำที่รับได้ทำได้โดย

$$S = rH^T$$

$$= (e_i \oplus u)H^T + uH^T \tag{2.17}$$

ถ้าหาก u เป็นรหัสคำที่ถูกต้องจะพบว่า

$$uH^T = 0 \tag{2.18}$$

ดังนั้นซีนโดรมคือ $S = H^T_1 H^T$

โดยปกติแล้วแต่ละรูปแบบของบิทที่ผิดไปถ้าไม่ซ้ำจะมีค่าซีนโดรมที่ไม่เท่ากัน ในการพิสูจน์ทำได้โดยถ้าสมมติ ว่าเป็นรูปแบบของบิทที่ผิดไปคนละรูปแบบแต่ให้ค่าซีนโดรมเท่ากัน อย่างเช่น รูปแบบของบิทที่ผิดไป e_i กับ e_t เมื่อซีนโดรมคือ

$$S_1 = e_i H^T$$

$$S_2 = e_t H^T$$

แต่ $S_1 = S_2$ จะได้

$$e_i H^T = e_t H^T \tag{2.19}$$

หรือ

$$(e_i - e_t) H^T = 0$$

เนื่องจาก H^T ไม่เป็นศูนย์ ดังนั้น $e_i - e_t$ จะมีค่าเป็นศูนย์

$$e_i - e_t = 0$$

ผลต่างของ e_i กับ e_t จะเป็นศูนย์ก็ต่อเมื่อทุกบิทใน e_i กับ e_t จะเหมือนกับแบบบิทต่อบิท จึงสรุปได้ว่า

$$e_i = e_t$$

ซึ่งขัดกับสมมติฐานที่ว่า e_i กับ e_t เป็นคนละรูปแบบ

ดังนั้นพอสรุปได้ว่าถ้าหารูปแบบบิทที่ผิดไปคนละรูปแบบจะให้ค่าซีนโดรมที่แตกต่างกันออกไป กล่าวคือ

$$e_i H^T \neq e_t H^T \tag{2.20}$$

ถ้าหากรหัสคำ (n , k) ถูกนำมาคำนวณหาซีนโดรม จะได้ซีนโดรมขนาด n- k ดังนั้นซีนโดรมที่แตกต่างกันจะมีจำนวน 2^{n-k} ค่าจะพบว่ารูปแบบของบิทที่ผิดไปหนึ่งรูปแบบกับซีนโดรมที่สอดคล้องและเป็นแบบหนึ่งต่อหนึ่ง ปกติแล้วทางด้านรับจะมีตารางของซีนโดรมและรูปแบบของบิทที่สอดคล้องกันกับซีนโดรมเก็บเอาไว้ ดังนั้นทางด้านรับจะมีขั้นตอนการทำงาน 4 ขั้นตอน กล่าวคือ

1. คำนวณซีนโดรมของรหัสคำ r ที่ได้รับจาก $S = rH^T$
2. เปิดตารางของซีนโดรมเพื่อดึงเอารูปแบบที่ผิดให้ซีนโดรมเหมือนกับที่คำนวณได้ ถ้าเป็นรูปแบบของ e_i
3. รหัสที่ถูกต้องคำนวณได้จาก $u = r + e_i$
4. ดึง k บิทแรกจากรหัสคำของ u ซึ่งจะเป็นรหัสข่าวสาร m ที่ส่งมา

ตัวอย่างเช่น ถ้าหากมีแมทริกซ์ ตัวกำเนิด G เป็น

$$G=\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

สามารถแปลงเป็นแมตริกซ์ตรวจสอบพาริตี้ H เป็น

$$H=\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

ถ้าสมมติว่ารูปแบบรหัสที่ผิดไปเพียงหนึ่งบิท เป็น $e_i = [0, 0, 0, 0, 0, 1]$

$$S = e_i H^T = [0\ 0\ 0\ 0\ 0\ 0\ 1] \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [0\ 0\ 1]$$

สำหรับรูปแบบต่างๆของรหัสคำที่ผิดไปเพียงบิทเดียวกับซินโดรมที่สอดคล้องพอสรุปได้ดังตารางข้างล่าง

ตารางที่ 2.4 แสดงซินโดรมที่ได้จากรูปแบบรหัสที่ผิดไปหนึ่งบิท

ซินโดรม	รูปแบบรหัสที่ผิดไปเพียงบิทเดียว
001	000001
010	000010
100	000100
110	001000
101	010000
011	100000

ถ้าสมมติว่าทางด้านส่งต้องการส่งรหัสข่าวสาร m = [101] จะได้รหัสคำ u

$$u = mG = [1\ 0\ 1] \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$= [1\ 0\ 1\ 1\ 0\ 1]$$

เมื่อทำการส่งรหัสคำ u ผ่านช่องส่งสัญญาณที่มีการรบกวน จะพบว่าทางด้านรับรหัสค่า r เป็น [1 0 0 1 0 1] ทางด้านรับจะทำการคำนวณหาค่าซีนโดรมจากรหัสค่า r ถ้าซีนโดรมเป็นศูนย์หมดทุกบิทแสดงว่ารหัสคำ r ที่รับกับรหัส u ที่ส่งเป็นรหัสเดียวกัน แต่ถ้าหากค่าซีนโดรมไม่เป็นศูนย์ก็ต้องทำการเปิดตารางที่ 4 เพื่อหารูปแบบที่ผิดไปที่สอดคล้องการคำนวณซีนโดรมทำได้โดย

$$S = rH^T = [1\ 0\ 0\ 1\ 0\ 1] \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [1\ 1\ 0]$$

ซีนโดรมที่ได้จะนำไปตรวจสอบกับตารางพบว่ารูปแบบของรหัสที่ผิดไปคือ [0,0,1,0,0,0,] ดังนั้นการแก้ไขรหัส r เพื่อให้ได้รหัสคำ u ที่ส่งมาทำได้โดย

$$u = r + e = [1\ 0\ 0\ 1\ 0\ 1] + [0\ 0\ 1\ 0\ 0\ 0]$$

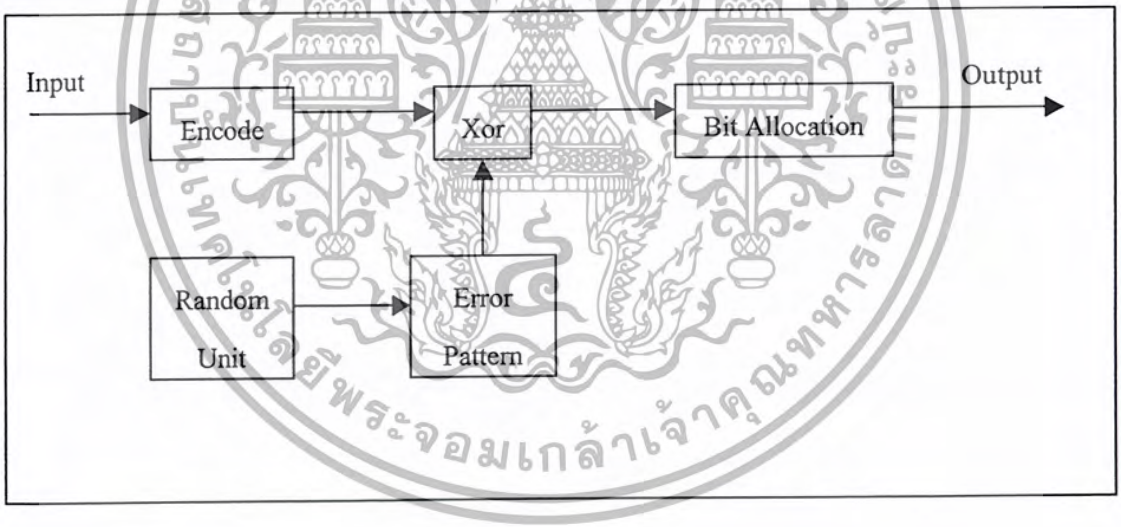
$$= [1\ 0\ 1\ 1\ 0\ 1]$$

จากวิธีการดึงรหัสคำที่ถูกต้องกลับคืนจากรหัสคำที่ผิด จึงได้ถูกนำมาใช้เป็นหลักการเข้ารหัสลับให้กับข้อมูลในวิทยานิพนธ์ฉบับนี้ หลักการทำงานของการแก้รหัสลับคือ จากรหัสคำที่มีอยู่จะถูกนำมาบวกกับรูปแบบรหัสที่ผิดต่างๆ ที่กำหนดไว้ โดยแต่ละแบบของรหัสที่ผิดจะมีซีนโดรมที่สอดคล้องเก็บเป็นตารางข้อมูลเอาไว้ ซึ่งรหัสที่ผิดมีหลายรูปแบบ ดังนั้นการบวกรหัสคำกับรูปแบบรหัสที่ผิด จะทำการเลือกรูปแบบรหัสที่ผิดแบบสุ่มเทียม (Psrudo random) วิธีการนี้ทางด้านรับจะมีรหัสคำที่ผิดอยู่ตลอดเวลา แต่จะสามารถนำเอารหัสคำที่ถูกต้องกลับคืนมาได้ถ้าหากรู้ว่าแมทริกซ์ G คืออะไร ในการเพิ่มความซับซ้อนการเข้ารหัสลับ จะมีการเรียงลำดับตำแหน่งบิทในแต่รหัสคำที่ผ่านการบวกรูปแบบบิทที่ผิดไปเรียบร้อยแล้ว รูปแบบของการเรียงสลับบิทที่ผิดจะมีอยู่หลายรูปแบบเช่นกัน การเลือกรูปแบบการสลับบิทจะใช้ค่าของซีนโดรมเป็นตัวเลือก สำหรับรายละเอียดในการเข้ารหัสจะกล่าวในบทต่อไป

บทที่ 3
หลักการออกแบบตัวเข้ารหัสและตัวถอดรหัสลับ

ในการออกแบบตัวเข้ารหัสและถอดรหัสแบบดิจิตอลนี้ สัญญาณที่จะนำมามาป้อนทางอินพุทสำหรับทำการเข้ารหัสจะต้องเป็นสัญญาณที่อยู่ในรูปของสัญญาณดิจิตอล แต่ถ้าอินพุทที่ถูกป้อนเข้ามาอยู่ในรูปของสัญญาณอนาล็อก จะต้องทำการแปลงสัญญาณเหล่านั้นให้อยู่ในรูปของสัญญาณดิจิตอล ก่อนที่จะส่งเข้าทางด้านอินพุทของตัวเข้ารหัส ส่วนทางด้านตัวถอดรหัสลับก็จะได้รับข้อมูลที่ถูกส่งผ่านเข้ามาทางอินพุทที่อยู่ในรูปสัญญาณดิจิตอลเช่นกัน และทำการถอดรหัสกลับ แต่ถ้ารูปสัญญาณเดิมเป็นแบบอนาล็อก ก็จะต้องใช้ตัวแปลงสัญญาณดิจิตอลกลับเป็นสัญญาณอนาล็อกกลับคืนมาเช่นเดิม ในส่วนนี้จะกล่าวถึงวิธีการเฉพาะส่วนของตัวเข้ารหัสและตัวถอดรหัสเท่านั้น

3.1 การออกแบบตัวเข้ารหัสลับ



ภาพที่ 3.1 แสดงแผนภาพการเข้ารหัสลับ

ในการออกแบบตัวสร้างรหัสลับจะเป็นการนำข่าวสารข้อมูลที่เข้ามาทางด้านอินพุท มาทำการแปลงให้มีรูปแบบของข้อมูลแตกต่างไปจากเดิม โดยทำการแปลงข้อมูลข่าวสารที่เข้ามาทางด้านอินพุทที่มีขนาด k บิท (ในที่นี้ใช้ 8 บิท) ให้เป็นรหัสคำ ที่มีขนาด n บิท (12 บิท) แล้วทำการบวกรหัสคำที่ได้กับรูปแบบผิดพลาดที่ทำการสร้างขึ้นจำนวน 15 รูปแบบหลังจากนั้นจะทำการสลับตำแหน่งของรหัสคำซึ่งมีรูปแบบการสลับตำแหน่ง 16 รูปแบบที่ทำการบวกรูปแบบผิดพลาดแล้ว ก็จะทำให้ได้รหัสข้อมูลลับที่ทำการเข้ารหัสแล้ว ซึ่งสามารถแบ่งเป็นส่วนๆของการสร้างรหัสลับดังแสดงในภาพที่ 3.1 ได้ดังนี้

3.1.1 ส่วนของการเข้ารหัส (Encoder)

เป็นส่วนที่ทำหน้าที่ในการแปลงรหัสข้อมูลข่าวสารที่เข้ามาทางอินพุท ที่มีขนาดของข้อมูล k บิท มาทำการแปลงให้เป็นรหัสคำ ในการแปลงนี้จะใช้วิธีการของรหัสบล็อกโค้ดเชิงเส้น โดยจะเป็นการนำรหัสข่าวสาร m ที่มีขนาด k บิทไปทำการคูณกับแมทริกซ์ตัวกำเนิด G (Generator Matrix) จะได้ผลลัพธ์เป็นรหัสคำที่มีขนาด k บิท จะมีจำนวนของบิทที่เพิ่มขึ้นมาจากรหัสของข้อมูลข่าวสาร (n- k) บิท (4 บิท) โดยรหัสที่เพิ่มขึ้นนี้จะถูกนำมาใช้ในการแก้รหัสผิดซึ่งจะกล่าวในหัวข้อต่อไป

ภายในแมทริกซ์ตัวกำเนิด G จะประกอบไปด้วยแมทริกซ์เอกลักษณ์ I_k ที่มีขนาด k x k และแมทริกซ์พาริตี้ P ที่มีขนาด k x (n-k) จะแสดงได้ในสมการที่ 3.1

$$G = [I_k : P] \tag{3.1}$$

แมทริกซ์ตัวกำเนิด G ที่ใช้แสดงในภาพที่ 3.2

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & : & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & : & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & : & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & : & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & : & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & : & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & : & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & : & 1 & 1 & 0 & 1 \end{bmatrix}$$

ภาพที่ 3.2 แสดงแมทริกซ์ตัวกำเนิด

จากรหัสข้อมูลข่าวสาร m ที่มีขนาด 8 บิท นำมาทำการคูณกับแมทริกซ์ตัวกำเนิด G จะได้รหัสคำที่มีขนาด 12 บิท ดังแสดงในสมการที่ 3.2

$$u = mG \tag{3.2}$$

ทำให้สามารถแสดงวิธีการสร้างรหัสคำ u ในภาพที่ 7

$$u = [m_0 \ m_1 \ m_2 \ m_3 \ m_4 \ m_5 \ m_6 \ m_7] \begin{bmatrix} 1&0&0&0&0&0&0&0&:&0&0&1&1 \\ 0&1&0&0&0&0&0&0&:&1&0&0&1 \\ 0&0&1&0&0&0&0&0&:&1&1&0&0 \\ 0&0&0&1&0&0&0&0&:&0&1&1&0 \\ 0&0&0&0&1&0&0&0&:&1&1&1&0 \\ 0&0&0&0&0&1&0&0&:&0&1&1&1 \\ 0&0&0&0&0&0&1&0&:&1&0&1&1 \\ 0&0&0&0&0&0&0&1&:&1&1&0&1 \end{bmatrix}$$

ภาพที่ 3.3 แสดงการสร้างรหัสคำ

3.1.2 ส่วนของรูปแบบผิดพลาด (Error Pattern)

สำหรับแมทริกซ์ตัวกำเนิดในภาพที่ 3.2 นั้นจะแก้รหัสผิดได้เพียงบิทเดียวแต่ในการเข้ารหัสลับนี้จะใช้รูปแบบของบิทที่ผิดไปสองบิท ใน 8 บิทแรกของรหัสคำซึ่งเป็นบิทของข่าวสาร ดังนั้นจึงไม่เป็นไปตามกฎของซินโดรม กล่าวคืออาจมีบิทที่ผิดไปมากกว่าหนึ่งรูปแบบที่ให้ซินโดรมเหมือนกัน ดังแสดงในตารางที่ 3.1เนื่องจากการเข้ารหัสบล็อกโค้ดเชิงเส้น (12,8) จะมีซินโดรมได้ 4 บิท หรือมีซินโดรมได้ 15 รูปแบบที่แตกต่างกัน (ไม่นับที่ทุกบิทเป็นศูนย์หมด) เนื่องจากรหัสที่ผิดไปสองบิทใน 8 บิทแรก จะมีรูปแบบที่ไม่เหมือนกันได้เป็น 28 รูปแบบ ดังแสดงในคอลัมน์ที่สองของตารางที่ 3.1 ในกรณีที่รูปแบบที่ผิดมีหลายรูปแบบจะเลือกเอารูปแบบที่ผิดที่ให้ค่าสูงที่สุด อย่างเช่นรูปแบบ (3,5) ซึ่งมีบิทที่ผิดคือบิทที่ 3 กับ 5 จาก 8 บิทแรกของรหัสคำ กับรูปแบบ (2,7) ซึ่งมีบิทที่ผิดคือบิทที่ 2 กับ 7 จาก 8 บิทแรกของรหัสคำ จะพบว่ารูปแบบ (2,7) มีค่าสูงสุดกว่า (3,5) จึงเลือกรูปแบบ (2,7) เนื่องจากการเข้ารหัสลับสัญญาณเสียง ถ้าเลือกบิทที่ผิดมีค่าสูงจะทำให้แอมปลิจูด (Amplitude) ของสัญญาณเสียงที่รวมกับรหัสที่ผิดแล้วเกิดการเปลี่ยนแปลงสูงดังนั้น รูปแบบบิตที่ผิดที่ได้เลือกไว้ และซินโดรมที่สอดคล้องพอสรุปได้ดังตารางที่ 3.1

$$r = u \oplus e \tag{3.3}$$

ตารางที่ 3.1 รูปแบบผิดพลาด

ค่าซีนโดรม	ตำแหน่งบิทที่ผิด	เลือกรูปแบบ	รูปแบบบิทที่ผิด
0001	(3,5) (2,7)	(2,7)	001000010000
0010	(1,6) (2,4)	(1,6)	010000100000
0011	(4,7)	(4,7)	000010010000
0100	(0,5) (1,7)	(0,5)	100001000000
0101	(0,3) (1,2) (4,6)	(4,6)	000010100000
0110	(6,7)	(6,7)	000000110000
0111	(1,4) (2,6)	(1,4)	010010000000
1000	(0,6) (3,4)	(3,4)	000110000000
1001	(4,5)	(4,5)	000011000000
1010	(0,1) (2,3) (5,7)	(5,7)	000001010000
1011	(2,5) (3,7)	(2,5)	001001000000
1100	(5,6)	(5,6)	000001100000
1101	(0,4) (3,6)	(0,4)	100010000000
1110	(0,7) (1,5)	(0,7)	100000010000
1111	(0,2) (1,3)	(1,3)	010100000000

3.1.3 ส่วนของการกำเนิดสัญญาณสุ่มเทียม (Pseudo random sequence)

เป็นส่วนของการเกิดสัญญาณสุ่มเทียม มาทำหน้าที่เป็นตัวเลือกรูปแบบผิดพลาดที่จะมาทำการบวกแบบเอกคลูซีฟออร์กับรหัสคำที่ได้ เป็นดังตารางที่ 3.2

ตารางที่ 3.2 รูปแบบการกำเนิดแบบสุ่ม

ลำดับ	รูปแบบ
0	0001
1	0010
2	0100
3	1000
4	0011
5	0110
6	1100
7	1011

ลำดับ	รูปแบบ
8	0101
9	1010
10	0111
11	1110
12	1111
13	1101
14	1001

3.1.4 ส่วนของการสลับตำแหน่งบิท (Bit allocation)

เป็นการสลับตำแหน่งของข้อมูลที่เป็นรหัสคำที่ทำการบวกกับรูปแบบผิดพลาดแล้ว โดยจะทำการสลับเฉพาะ 8 บิทแรกเท่านั้น เพื่อเพิ่มความซับซ้อนในการป้องกันข้อมูลให้มีความปลอดภัยยิ่งขึ้น โดยการนำ 4 บิทหลังที่ได้จากรหัสคำมาเป็นตัวเลือกรูปแบบของการสลับตำแหน่งบิทข้อมูลซึ่งมีรูปแบบการสลับบิท 16 รูปแบบดังแสดงในตารางที่ 3.3

3.2 การออกแบบตัวถอดรหัสลับ

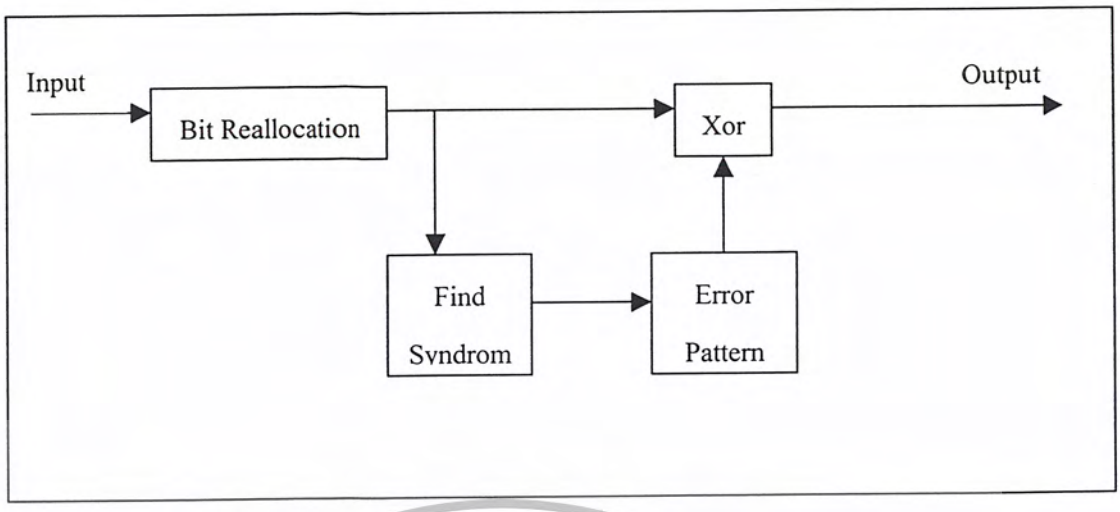
ในการออกแบบตัวถอดรหัสลับเป็นการนำข้อมูลข่าวสารที่ผ่านการเข้ารหัสลับแล้วกลับคืนมาให้ได้ข้อมูลเดิม โดยทำการถอดรหัสข้อมูลข่าวสาร จากข่าวสารที่ทำการเข้ารหัสลับแล้วมีขนาด n บิทเข้ามาทางอินพุท โดยเริ่มจากการสลับตำแหน่งของข้อมูลกลับ ให้ข้อมูลที่ได้มีตำแหน่งเหมือนเดิม โดยใช้ข้อมูล 4 บิทที่เพิ่มเข้ามาเป็นตัวเลือกรูปแบบที่จะทำการสลับตำแหน่งบิทกลับ แล้วทำการคำนวณหาค่าซีนโดรมที่จะมาเป็นตัวเลือกรูปแบบที่ผิดพลาดดังแสดงในตารางที่ 4 มาทำการบวกแบบเอกคลูซีฟออร์กับข้อมูลข่าวสารนั้น ทำให้ได้ข้อมูลข่าวสารเดิมกลับคืนมาแบ่งเป็นแต่ละขั้นตอนดังแสดงในภาพที่ 3.4

ตารางที่ 3.3 การสลับตำแหน่งบิท

ลำดับรูปแบบ	ตำแหน่งบิทที่ทำการสลับ							
	LSB							MSB
0000	0	2	3	4	7	6	1	5
0001	2	4	6	1	0	3	7	5
0010	5	7	0	4	2	6	1	3
0011	6	4	0	3	7	5	1	2
0100	4	0	6	3	7	1	2	5
0101	5	1	6	2	4	3	7	0
0110	7	0	1	2	6	4	5	3
0111	6	2	1	0	7	3	4	5
1000	3	1	4	0	6	2	5	7
1001	2	5	4	1	6	3	0	7
1010	1	7	3	4	5	2	0	6
1011	5	1	6	2	7	0	3	4
1100	6	4	0	3	1	2	7	5
1101	7	0	1	2	3	4	5	6
1110	4	0	7	3	2	1	5	6
1111	5	0	4	1	2	3	7	6

3.2.1 ส่วนของการสลับตำแหน่งบิทกลับ (Bit reaiiocation)

เป็นส่วนที่ทำหน้าที่ ในการสลับตำแหน่งของบิทข้อมูลที่เป็นรหัสคำที่ผิดกลับคืนมาให้ได้ตำแหน่งที่ถูกต้องตรงกับตำแหน่งเดิม โดยจะทำการสลับตำแหน่งข้อมูลกลับเฉพาะ 8 บิทแรกที่ถูกสลับไว้เท่านั้น และ 4 บิทที่เหลือจะเป็นตัวเลือกรูปแบบของการสลับตำแหน่งข้อมูลกลับ ซึ่งจะมีรูปแบบการสลับตำแหน่งบิทข้อมูลดังแสดงในตารางที่ 3.3



ภาพที่ 3.4 แสดงแผนภาพการถอดรหัสลับ

3.2.2 ส่วนของการคำนวณหาซีนโดรม

ในส่วนของการคำนวณหาซีนโดรมประกอบไปด้วย แมทริกซ์ตรวจสอบพาริตี้ H ที่ถูกนำมาใช้ในการคำนวณหาซีนโดรมสำหรับนำมาใช้ในการถอดรหัสที่ผิดไปโดยแมทริกซ์ตรวจสอบพาริตี้ H ประกอบขึ้นจากแมทริกซ์ทรานสโพสของแมทริกซ์พาริตี้ P ที่มีขนาด (n-k)x k และแมทริกซ์เอกลักษณ์ I_{n-k} คือ

$$H = [\, P^{T} I_{n-k} \,] \tag{3.4}$$

แมทริกซ์ตรวจสอบพาริตี้ H แสดงในภาพที่ 3.5

$$H = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

ภาพที่ 3.5 แสดงแมทริกซ์พาริตี้

ตารางที่ 3.4 การสลับตำแหน่งบิทกลับ

ลำดับรูปแบบ	ตำแหน่งบิทที่ทำการสลับ							
	LSB							MSB
0000	0	6	1	2	3	7	5	4
0001	4	3	0	5	1	7	2	6
0010	2	6	4	7	3	0	5	1
0011	2	6	7	3	1	5	0	4
0100	1	5	6	3	0	7	2	4
0101	7	1	3	5	4	0	2	6
0110	1	2	3	7	5	6	4	0
0111	3	2	1	5	6	7	0	4
1000	3	1	5	0	2	6	4	7
1001	6	3	0	5	2	1	4	7
1010	6	0	5	2	3	4	7	1
1011	5	1	3	6	7	0	2	4
1100	2	4	5	3	1	7	0	6
1101	1	2	3	4	5	6	7	0
1110	1	5	4	3	0	6	7	2
1111	1	3	4	5	2	0	7	6

ในการคำนวณหาซินโดรม S ได้จากการนำรหัสคำที่ได้รับมาทำการคูณกับแมทริกซ์ทรานสโพสของ H ซึ่ง H^T เป็นแมทริกซ์ โดย 4 แถวสุดท้ายของ H^T จะเป็นแมทริกซ์เอกลักษณ์ โดยใน 8 แถวแรกได้จากการทำทรานสโพสแมทริกซ์ P ของแมทริกซ์ตัวกำเนิด โดยทั้ง 8 แถวจะแตกต่างกัน และ r เป็นรหัสที่ได้รับเข้ามาซึ่งสามารถทำการคำนวณได้ดังแสดงในสมการที่ 3.5

$$S = r H^T \tag{3.5}$$

ซีนโดรมสามารถทำการหาได้ดังภาพที่3.6

$$S = [r_0 \quad r_1 \quad \ldots\ldots \quad r_{11} \quad r_{12}] \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ภาพที่ 3.6 แสดงการหาค่าซีนโดรม

แต่เนื่องจากแต่ละแถว (ROW) ของแมทริกซ์ G และแมทริกซ์ H จะตั้งฉากกันจะให้อินเนอร์โปรดัก (Inner product) เป็นศูนย์ ก็หมายความว่า $u H^T = 0$ นั่นเอง ทำการแทนค่า r ด้วย สมการที่3.5 ได้ว่า

$$\begin{aligned} S &= (u \oplus e) H^T \\ S &= u H^T \oplus e H^T \\ S &= e H^T \end{aligned} \tag{3.6}$$

จากสมการที่3.5 และ3.6 เป็นตัวบอกให้ทราบว่าค่าซีนโดรมที่คำนวณได้จากรหัสคำที่ผิดไปเนื่องจากรูปแบบผิดพลาด e ที่คูณกับ H^T นั้นให้ค่าซีนโดรมเหมือนกันกับการนำรูปแบบผิดพลาด e ไปคูณโดยตรงกับ H^T ด้วยเหตุนี้ค่าของซีนโดรมจึงเป็นตัวบ่งบอกว่ารหัส r ที่ได้รับมีรูปแบบผิดพลาด e รูปแบบใดปรากฏอยู่ ดังนั้นรูปแบบการดึงรหัสคำที่ถูกส่ง u กลับคืนมาทำได้โดยการนำ e ไปทำการบวกเอกคลูซีฟออร์กับรหัส r อีกครั้งหนึ่ง ดังแสดงในสมการที่3.7 ซึ่งทำให้สามารถทำการถอดรหัสได้ข้อมูลข่าวสารที่ถูกต้องกลับคืนมา

$$\begin{aligned} r \oplus e &= (u \oplus e) \oplus e \\ &= u \oplus (e \oplus e) \\ &= u \end{aligned} \tag{3.7}$$

บทที่ 4
การออกแบบตัวเข้ารหัสและถอดรหัสโดยใช้ FPGA

FPGA ย่อมาจาก Field Programmable Gate Array เป็นวงจรรวมทางดิจิตอลที่คุณสามารถโปรแกรมวงจรหรือฟังค์ชันการทำงานลงไปภายในตัวชิฟได้เอง เหมาะสำหรับการออกแบบวงจรและการออกแบบชิฟต้นแบบของวงจรทางดิจิตอล ข้อดีคือ เปรียบเทียบกับการออกแบบวงจรดิจิตอลโดยการใช้ IC Gates หรือใช้ IC TTL หลายๆตัวบนแผ่น PCB เราสามารถออกแบบวงจร การเชื่อมต่อและคุณสมบัติต่างๆด้วย Software ได้ จากนั้นเมื่อทดลอง Simulate ได้ผลน่าพอใจแล้วจึงโปรแกรมลงบนชิฟ FPGA จะเห็นว่าการแก้ไขทำได้ง่าย เพียงแก้บน Software (เสมือนอุปกรณ์ดิจิตอลของคุณอยู่ในรูปของ Software แก้ไขง่ายและแลกเปลี่ยนกันใช้ได้) และทำการโปรแกรมใหม่ (โปรแกรมซ้ำได้) ลดความยุ่งยากจากการเปลี่ยนอุปกรณ์ใหม่ การนำ IC จำนวนมากมาต่อกัน การออกแบบ PCB ใหม่ และความผิดพลาดที่อาจเกิดขึ้นได้จากลายวงจร ลดการเกิดสัญญาณรบกวนจากการออกแบบ PCB และการใช้อุปกรณ์มากๆได้ เปรียบเทียบกับการออกแบบโดย ASIC การออกแบบวงจรรวม (IC) ต้นแบบโดย ASIC พัฒนาได้ยาก เนื่องจากการแก้ไขวงจรแต่ละครั้งหมายถึงการเริ่มต้นขบวนการใหม่ทั้งหมด เช่น การออกแบบ Layout และการทำบน Silicon wafer เป็นต้น รวมถึงทรัพยากรทั้ง Hardware และ Software ในการออกแบบมีราคาแพง ดังนั้น การนำ FPGA ไปช่วยในการออกแบบทำให้การพัฒนาและการแก้ไขทำได้สะดวกและประหยัดขึ้น ซึ่งการออกแบบทำได้โดยเขียนวงจร Schematics ประกอบกับการเขียนภาษาอธิบายลักษณะพฤติกรรมหรือ Hardware Description Language (ฟังเข้าใจยากครับ ที่จริงก็คือเขียนวงจรกับเขียนโปรแกรมประกอบครับ) จากนั้นทำการสังเคราะห์และโปรแกรมลงบนชิพ FPGA ด้วย Software เช่น MAX+PLUS II ผู้ใช้สามารถออกแบบและแก้ไขวงจรได้ง่าย จะเห็นว่า เทคโนโลยี FPGA จะช่วยให้นักศึกษาและผู้ที่สนใจสามารถออกแบบ IC ของตนเองได้ นอกจากนี้เมื่อนักออกแบบสร้าง IC ของตนเองขึ้นมาแล้วยังสามารถป้องกันการลอกเลียนแบบได้อีกด้วย

4.1 การออกแบบฟังก์ชันลอจิกโดยใช้ FPGA

ผู้เขียนได้เลือกใช้ บอร์ด FPGA รุ่น POWER ACEX1K-30 ซึ่งเป็นบอร์ด FPGA ที่สามารถนำไปใช้งานเป็นบอร์ดทดลองหรือบอร์ดพัฒนางานต่างๆทางด้านระบบดิจิตอล ด้วยความจุเกตุประมาณ 30,000 เกตุ ซึ่งสามารถนำไปใช้ออกแบบระบบดิจิตอลที่มีขนาดใหญ่หรือสลับซับซ้อนได้ เนื่องจากชิฟ FPGA ตระกูล ACEX สามารถรองรับ Digital I/O ได้ 3 ระดับคือ

2.5V 3.3V และ 5V ดังนั้นบนบอร์ด POWER ACEX1K จึงมีแหล่งจ่ายไฟ 3 แหล่งคือ 2.5V 3.3V และ 5V เพื่อความสะดวกในการใช้งานอุปกรณ์ดิจิตอลหลายๆแบบและ ในแต่ละรุ่นจะมี Socket สำหรับใส่ Configuration Device เพื่อเก็บวงจรที่ได้ออกแบบไว้ สำหรับโหลดวงจรโดยอัตโนมัติลงใน FPGA เมื่อเริ่มจ่ายไฟเลี้ยงเข้าบอร์ด บอร์ด FPGA ในกลุ่ม POWER ACEX1K SERIESซึ่งจะทำการออกแบบสำหรับใช้งานในส่วนต่างๆ ดังนี้

4.1.1 ส่วนของการเข้ารหัส (Encoder)

เป็นการสร้างรหัสคำซึ่งจะเป็นไปตามสมการที่ 3.2 โดยที่ m คือรหัสข้อมูลข่าวสารที่เข้ามาทางอินพุทมีขนาด 8 บิท ที่นำมาทำการคูณกับแมทริกซ์ตัวกำเนิด G ส่วน u คือรหัสคำที่ปรากฏทางเอท์พุทซึ่งจะมีขนาด 12 บิท ทำให้สามารถเขียนเป็นสมการคูณกันของข้อมูลข่าวสารที่เข้ามากับแมทริกซ์ตัวกำเนิดดังแสดงในสมการที่ 4.1

$$
\begin{aligned}
u_0 &= m_0 & u_1 &= m_1 \\
u_2 &= m_2 & u_3 &= m_3 \\
u_4 &= m_4 & u_5 &= m_5 \\
u_6 &= m_6 & u_7 &= m_7 \\
u_8 &= m_1 \oplus m_2 \oplus m_4 \oplus m_6 \oplus m_7 \\
u_9 &= m_2 \oplus m_3 \oplus m_4 \oplus m_5 \oplus m_7 \\
u_{10} &= m_0 \oplus m_3 \oplus m_4 \oplus m_5 \oplus m_6 \\
u_{11} &= m_0 \oplus m_1 \oplus m_5 \oplus m_6 \oplus m_7
\end{aligned}
\qquad (4.1)
$$

จากสมการที่ 4.1 จะเห็นได้ว่า u_0 - u_7 มีค่าเท่ากับ m_0 - m_7 และในส่วนของ u_8 - u_{11} จะถูกนำมาใช้ในการสร้างสมการสมการลอจิก ซึ่งหมายความว่า output ที่ได้นั้นเป็น 12 บิท ทำการสร้างวงจรที่มี 8 อินพุทและมี 12 เอาท์พุท และเขียนภาษาอธิบายลักษณะพฤติกรรม (VHDL) โดยเอาท์พุทแยกออกเป็น u_0 - u_7 ซึ่งสามารถ LATCH ออกไปได้เลย ส่วน u_8 - u_{11} ก็ให้กระทำตามสมการ 4.1

4.1.2 ส่วนของรูปแบบผิดพลาด (Error Pattern)

เป็นการออกแบบโดยการใช้ Clk สร้างอินพุท 4 บิทโดยใช้การกำเนิดสัญญาณแบบสุ่มเทียมมาทำการเลือกรูปแบบผิดพลาดดังแสดงในตารางที่ 5 ซึ่งจะได้เป็นเอาท์พุท 8 บิทที่จะไปทำการ XOR กับ u_0 - u_7

4.1.3 ส่วนของการสลับบิท (Bit allocation)

การสลับบิทของรหัสคำที่จะปรากฏทางเอาท์พุท ตามรูปแบบที่ทำการกำหนดไว้ในตารางที่7 โดยจะใช้เอาท์พุท 4 บิท u_8 - u_{11}ที่ได้จากส่วนของการเข้ารหัสเป็นตัวเลือก

4.1.4 ส่วนของการสลับบิทกลับ (Bit reallocation)

เป็นส่วนที่ทำหน้าที่ในการสลับบิทกลับ ทำงานในลักษณะตรงกันข้ามกับการสลับบิทข้อมูลเพื่อให้ได้ข้อมูลที่มีตำแหน่งถูกต้องกลับคืนมา มีรูปแบบตามตารางที่ 8 โดยจะใช้เอาท์พุท 4 บิทที่ส่วนของเข้ารหัสส่งมา (u_8 - u_{11}) มาทำการเลือกรูปแบบ

4.1.5 ส่วนของการหาค่าซีนโดรม (Syndrome)

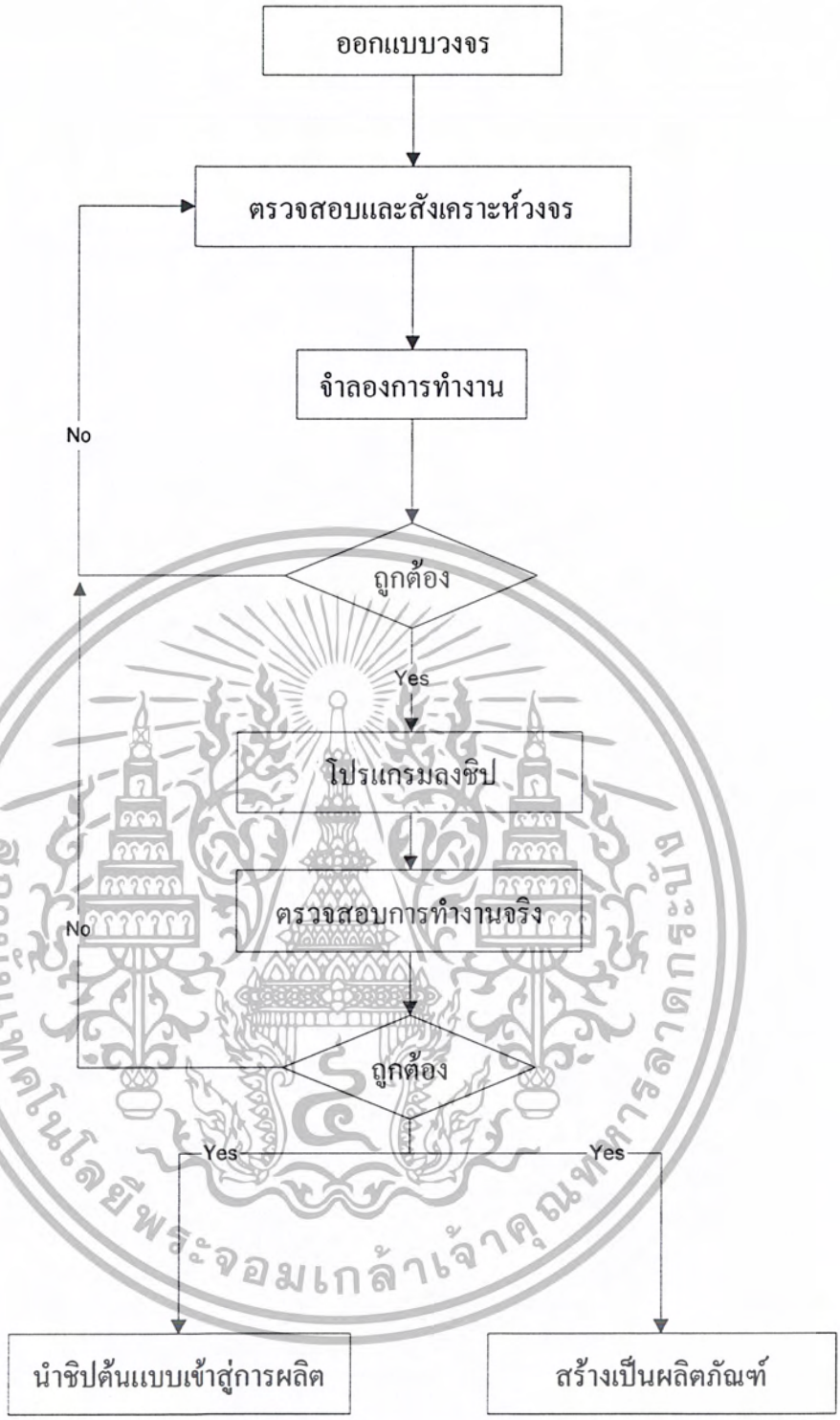
ส่วนการหาค่าที่ใช้ในการเลือกรูปแบบที่ผิดพลาดมาทำการ XOR กับข้อมูลข่าวสารจากการเข้ารหัส เพื่อให้ได้ข้อมูลที่ถูกต้องกลับคืนมา จากสมการที่ 3.5 สำหรับการคำนวณหาซีนโดรมสามารถนำมาเขียนเป็นสมการรูปของลอจิกได้ดังนี้

$$\begin{aligned} S_0 &= r_1 \oplus r_2 \oplus r_4 \oplus r_6 \oplus r_7 \oplus r_8 \\ S_1 &= r_2 \oplus r_3 \oplus r_4 \oplus r_5 \oplus r_7 \oplus r_9 \\ S_2 &= r_0 \oplus r_3 \oplus r_4 \oplus r_5 \oplus r_6 \oplus r_{10} \\ S_3 &= r_0 \oplus r_1 \oplus r_5 \oplus r_6 \oplus r_7 \oplus r_{11} \end{aligned} \qquad (4.2)$$

จากสมการที่ 4.2 เป็นการคำนวณหาค่าซีนโดรม $S_0 - S_3$ สามารถนำมาเขียนเป็นสมการลอจิกได้ โดยจะมี อินพุท 12 บิท โดย $r_0 - r_8$ จะได้มาจากการสลับบิทกลับ ส่วน $r_8 - r_{11}$ ก็คือ u_8 - u_{11} ที่ส่วนของการเข้ารหัสส่งมานั่นเอง ค่าซีนโดรมที่ได้จะไปทำการเลือกรูปแบบผิดพลาดที่จะมาทำการ XOR กับข้อมูลเข้ารหัสที่มีตำแหน่งถูกต้อง

จากส่วนต่างๆนำมาทำการออกแบบวงจรโดยใช้โปรแกรม MAX+PLUS II ซึ่งเป็นโปรแกรมที่ถูกสร้างและพัฒนาขึ้นโดยบริษัท ALTERA เพื่อใช้สำหรับออกแบบ – สังเคราะห์วงจรทางดิจิตอลและโปรแกรมวงจรลอจิกที่สร้างขึ้นลงในชิป FPGA ซึ่งในการออกแบบด้วยโปรแกรม MAX+PLUS II สามารถทำได้หลายลักษณะ เช่น

- ใช้การกำหนดรูปแบบ Waveform ทางอินพุทและเอาท์พุทของวงจรที่ต้องการ
- วาดวงจรโดยการนำเกทหรือสัญลักษณ์ต่างๆ มาเชื่อมต่อกัน
- ใช้ภาษาอธิบายพฤติกรรมของวงจรที่ต้องการ เช่น VHDL



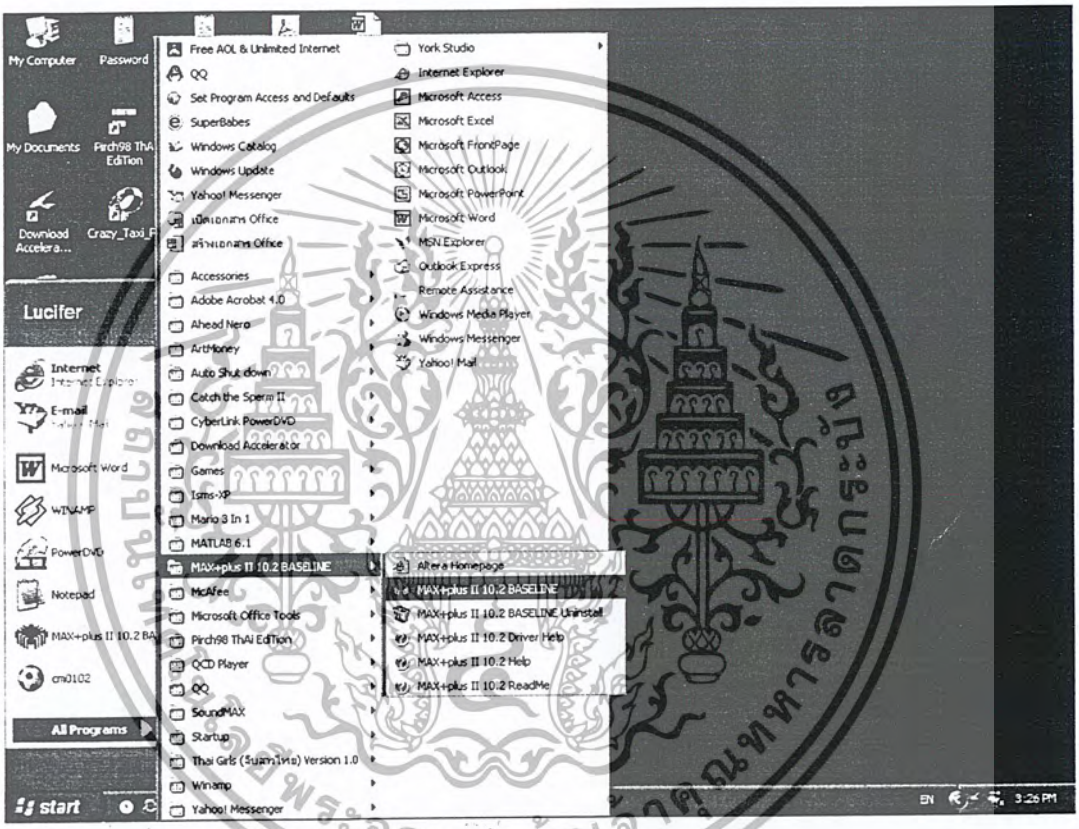
ภาพที่4.1 ขั้นตอนการออกแบบ

4.2 ขั้นตอนการออกแบบโดยใช้โปรแกรม MAX+PLUS II

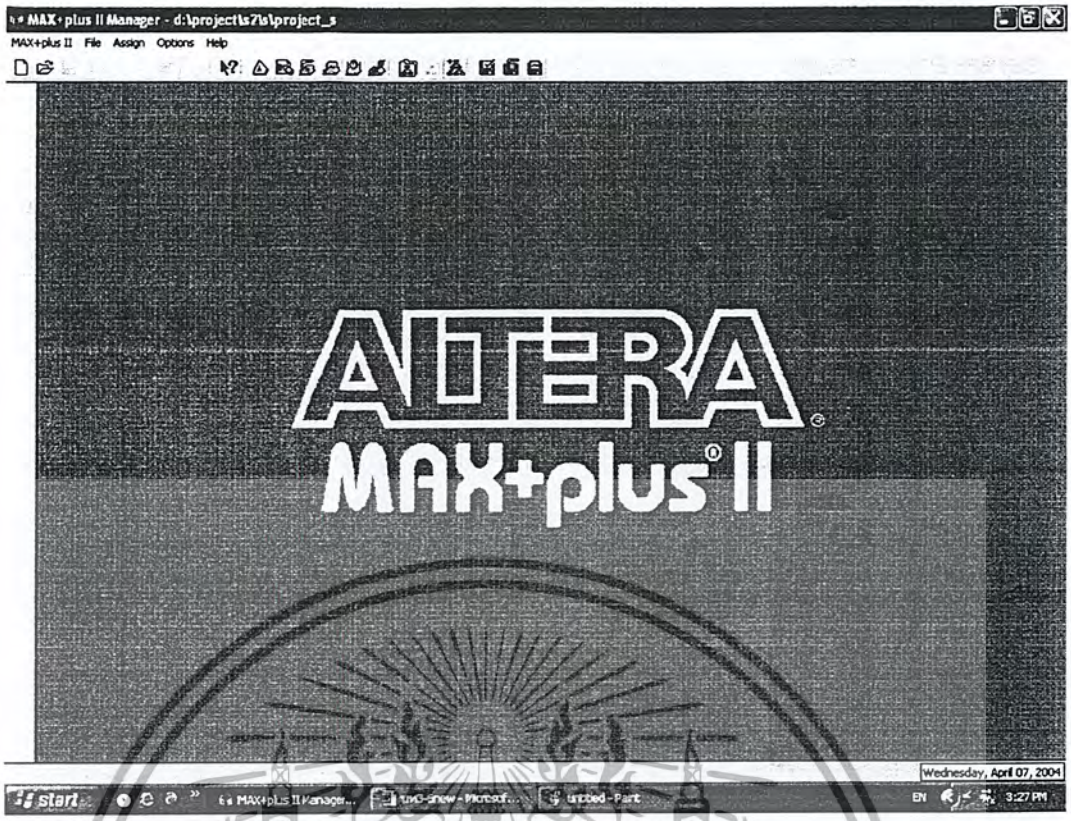
ในการออกแบบวงจรด้วยโปรแกรม MAX+PLUS II สามารถทำได้ดังภาพที่ 4.1 ซึ่งต่อจากนี้จะอธิบายตั้งแต่ขั้นตอนการออกแบบวงจร, การจำลองการทำงานของวงจรที่ออกแบบมา, การโปรแกรมลงบนชิป และตรวจสอบทำงานจริง

4.2.1 การออกแบบวงจร (Design Entry)

เรียกโปรแกรม MAX+PLUS II จาก Start Menu



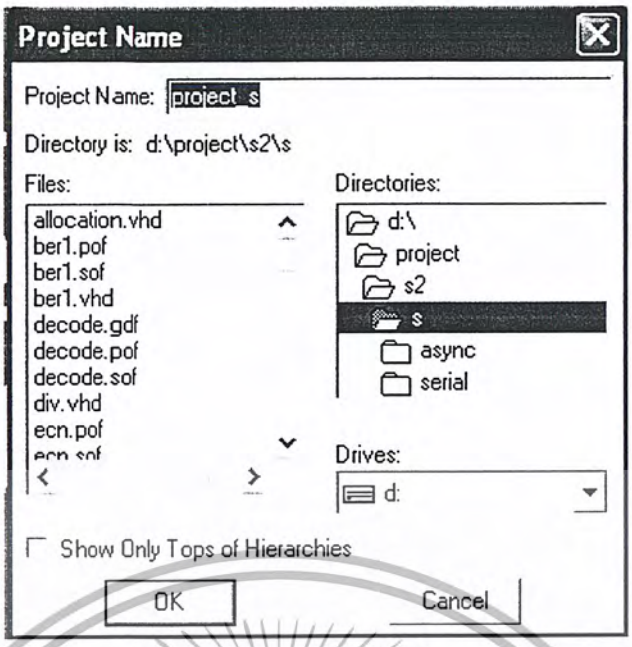
ภาพที่ 4.2 การเรียกโปรแกรม MAX+PLUS II



ภาพที่ 4.3 โปรแกรม MAX+PLUS II

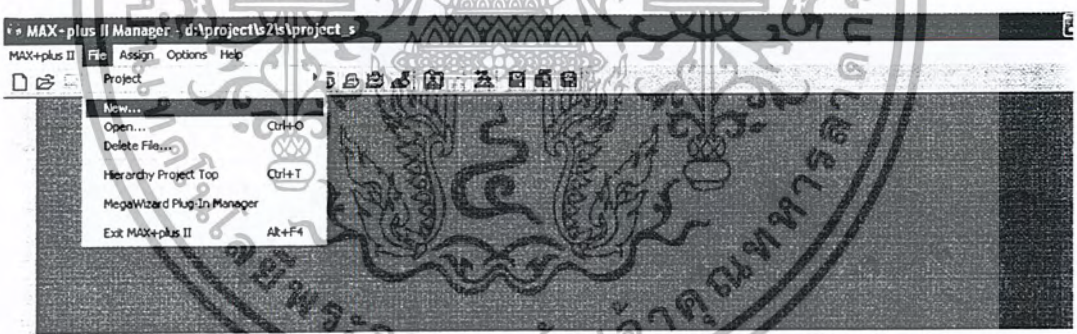
ทำการสร้างโปรเจ็คสำหรับวงจรที่ต้องการออกแบบโดยเริ่มจากการตั้งชื่อของโปรเจ็คที่เมนู File/Project/Name โดยผู้เขียนใช้เป็น project_s

ภาพที่ 4.4 การเลือกเมนูเพื่อสร้างโปรเจ็ค

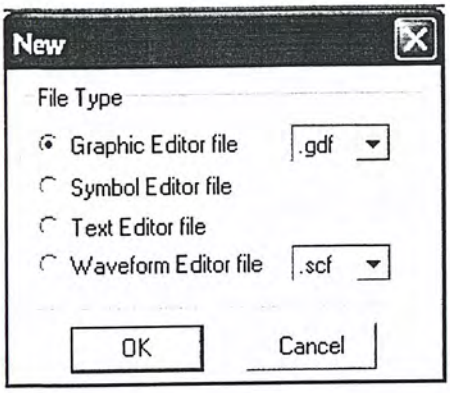


ภาพที่ 4.5 การตั้งชื่อโปรเจ็ค

เลือกประเภทของไฟล์ที่จะสร้างจากเมนู File / New / Graphic Editor File จากนั้นจะมีไดอะล็อกให้เลือกประเภทของไฟล์ที่ต้องการสร้างขึ้นมา



ภาพที่ 4.6 การเลือกเมนูเพื่อสร้างไฟล์

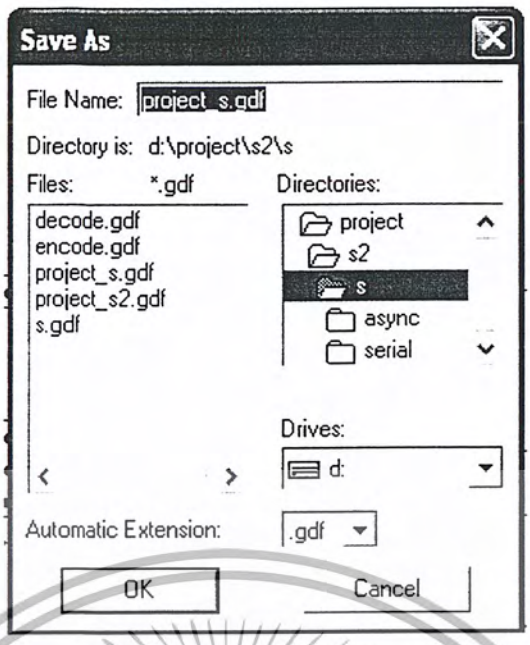


ภาพที่ 4.7 การเลือกชนิดของไฟล์ที่จะสร้าง

ไฟล์แต่ละชนิดจะมีความหมายดังต่อไปนี้

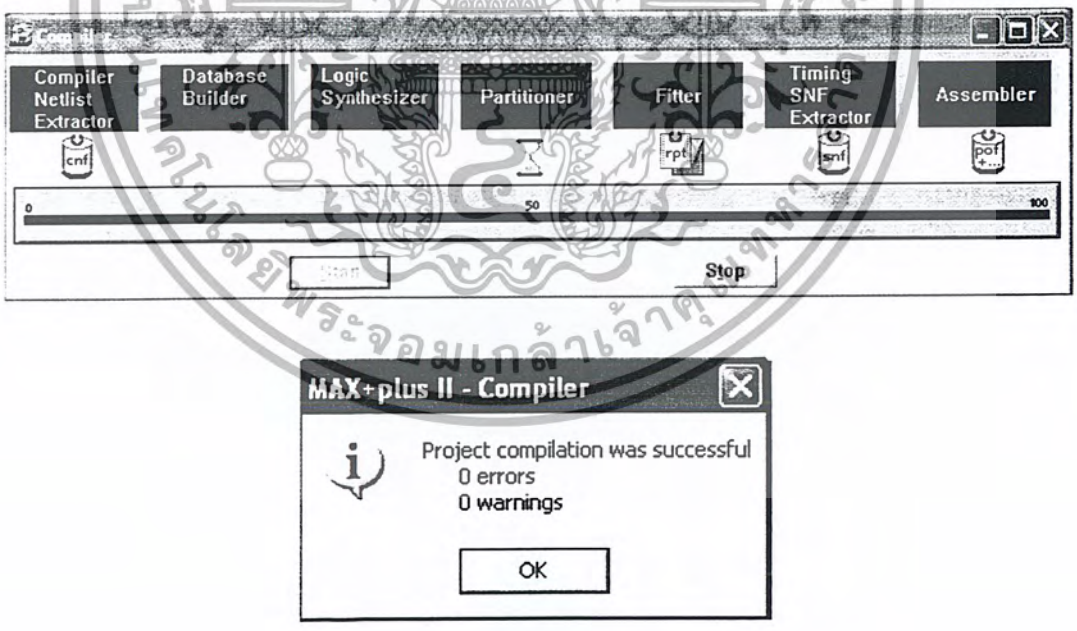
- Graphic Editor File : เป็นไฟล์กราฟฟิกที่เราสามารถนำเอาอุปกรณ์ต่างๆใน Library มาต่อกันเป็นวงจรได้เลย
- Symbol Editor File : เป็นไฟล์ที่ใช้เก็บสัญลักษณ์เพื่อสื่อให้ทราบว่าโมเดลที่เราได้สร้างขึ้นมีอินพุทและเอาท์พุทเป็นอย่างไร
- Text Editor File : เป็น Text File ใช้สำหรับเขียน Source Code เพื่ออธิบายพฤติกรรมของวงจรหรือโมเดลต่างๆที่เราจะสร้างหรือเพื่อใช้สำหรับเก็บข้อความทั่วๆไป
- Waveform Editor File : เป็นไฟล์ Waveform ที่มีไว้สำหรับอธิบายลักษณะ Waveform ของวงจรที่เราต้องการ ซึ่งถ้ามีการกำหนด Waveform ทั้งทางอินพุทและเอาท์พุทก็จะใช้นามสกุลของไฟล์เป็น wdf (Waveform Design File) แต่ถ้าใช้ในการกำหนดรูปแบบของสัญญาณอินพุท เพื่อให้โปรแกรม MAX+PLUS II จำลองการทำงานและบันทึกผลที่ได้จะใช้ไฟล์ที่มีนามสกุลเป็น scf (Simulate Channel File)

เมื่อออกแบบวงจรเสร็จเรียบร้อยแล้วให้ทำการบันทึกวงจรที่สร้างมาจากเมนู File / Save as จากนั้น Save ชื่อไฟล์เป็น project_s.gdf แล้วคลิ้กที่ปุ่ม OK



ภาพที่ 4.8 การบันทึกเป็นไฟล์ชื่อ project_s.gdf

ตรวจสอบความถูกต้องในการต่อวงจร File/Project/Save&Check หากต่อวงจรถูกต้องตามเงื่อนไขของโปรแกรม MAX+PLUSE II จะมีหน้าต่างปรากฏขึ้นมาดังภาพที่ 4.9



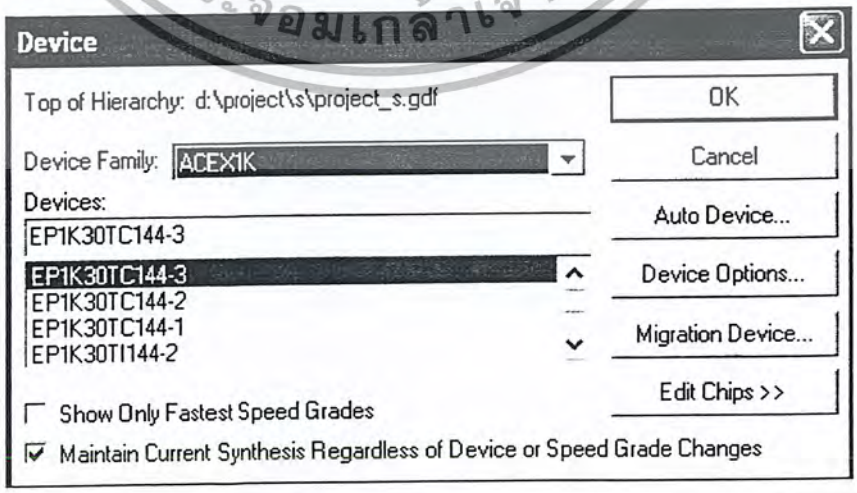
ภาพที่ 4.9 การตรวจสอบความถูกต้อง

4.2.2 การตรวจสอบและสังเคราะห์วงจร

ขั้นตอนนี้เป็นการตรวจสอบและสังเคราะห์วงจร (Compiler and Synthesis) เริ่มจากทำการระบุเบอร์ของชิปที่จะใช้จากเมนู Assign / Device ซึ่งจะมีไดอะล็อก Device ปรากฏขึ้นมาเพื่อให้เลือก Device ที่ต้องการใช้งาน เบอร์ของ CPLD/FPGA ที่จะทำการเลือกให้สังเกตจากด้านบนของตัวชิปที่ใช้งานเช่น

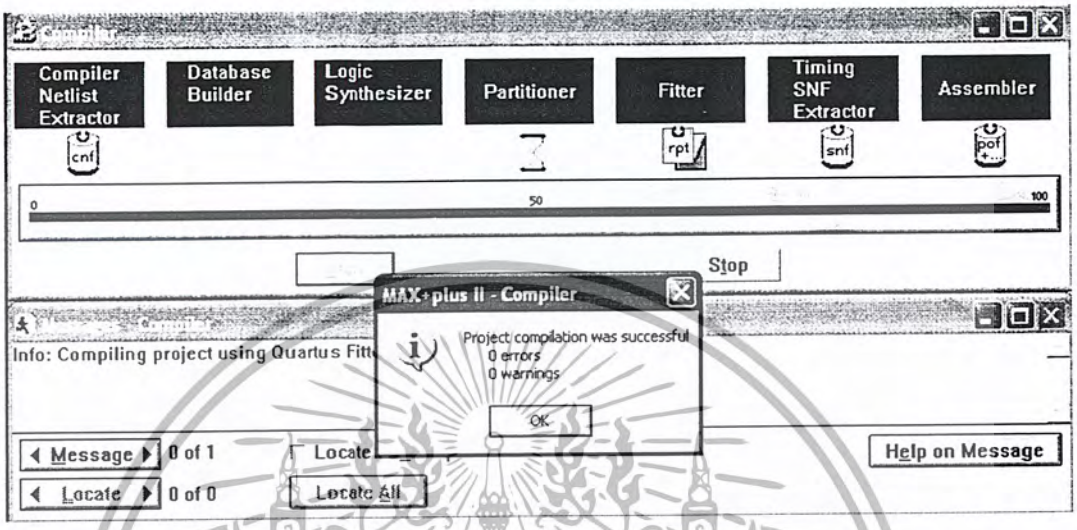
- หากเป็นผลิตภัณฑ์ในรุ่น Ezy PLD-DEV01 เบอร์ของ CPLD จะเป็น EMP3032ALC44-10 ก็ให้ทำการเลือก Device Family เป็น MAX3000Aและระบุเบอร์ชิปในช่อง Device เป็น EMP3032ALC44-10 และหากเป็นผลิตภัณฑ์ในรุ่น Ezy PLD-DEV02 เบอร์ของ CPLD จะเป็น EPM3064ALC44-10 ก็ให้เลือก Device Family เป็น MAX3000Aและระบุเบอร์ชิปในช่อง Devices เป็น EMP3064ALC44-10
- กรณีที่เป็นผลิตภัณฑ์ในกลุ่ม Wizard FLEX Series ให้เลือก Device Family เป็น FLEX10K และเลือก Device เป็น EPF10K10LC84-4
- กรณีที่เป็นผลิตภัณฑ์ในกลุ่ม Wizard PLD Series ให้เลือก Device Family เป็น MAX7000S และเลือก Device เป็น EMP7128SLC84-15
- หากเป็นผลิตภัณฑ์ในรุ่น POWER ACEX1K-10 ให้เลือก Device Family เป็น ACEX1K และระบุเบอร์ชิปในช่อง Devices เป็น EP1K10TC144-3
- หากเป็นผลิตภัณฑ์ในรุ่น POWER ACEX1K-30 ให้เลือก Device Family เป็น ACEX1K และระบุเบอร์ชิปในช่อง Devices เป็น EP1K30TC144-3 และหากเป็นผลิตภัณฑ์ในรุ่น POWER ACEX1K-50 ให้เลือก Device Family เป็น ACEX1K และระบุเบอร์ชิปในช่อง Devices เป็นEP1K50TC144-3

ซึ่งต้อง Uncheck ช่อง Show Only Fastest Speed Grades ก่อน เพื่อให้ไดอะล็อกแสดง Device ที่มีอยู่ทั้งหมดขึ้นมาดังภาพที่4.10



ภาพที่ 4.10 ระบุเบอร์ของชิป FPGA

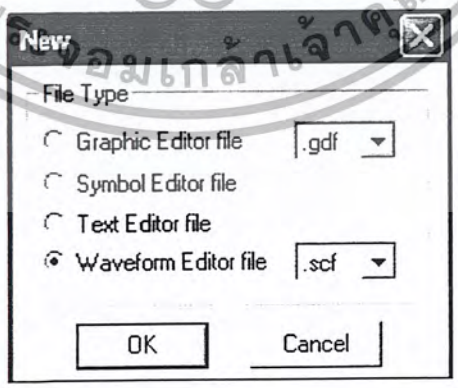
ทำการคอมไพล์วงจรที่ได้สร้างขึ้นจากเมนู MAX+PLUSII/Compiler หลังจากนั้นจะมีไดอะล็อกคอมไพล์ ปรากฏขึ้นมา ให้กดปุ่ม Start เพื่อเริ่มทำการคอมไพล์ หลังจากการคอมไพล์เสร็จสิ้นลงก็จะมีหน้าต่างรายงานผลการคอมไพล์ error และ warning หากมีข้อผิดพลาดเกิดขึ้นก็จะมีข้อความสีแดงบอกว่าผิดพลาดพร้อมทั้งสาเหตุของข้อผิดพลาด



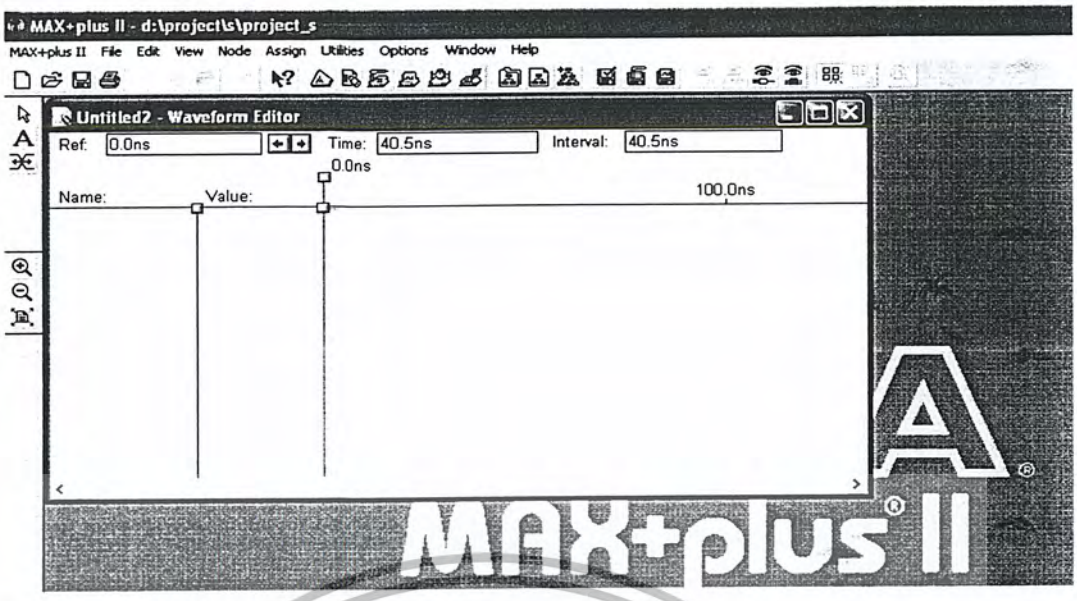
ภาพที่ 4.11 การคอมไพล์

4.2.3 จำลองการทำงาน (Simulation)

จำลองการทำงานของวงจร (Simulation) โดยการสร้าง Waveform จากเมนู File/New/Waveform Editor file ซึ่งจะมีไดอะล็อกปรากฏขึ้นดังรูปที่ 15 โดยในขณะนี้ Wave Editor จะปรากฏขึ้นมาดังภาพที่ 4.12

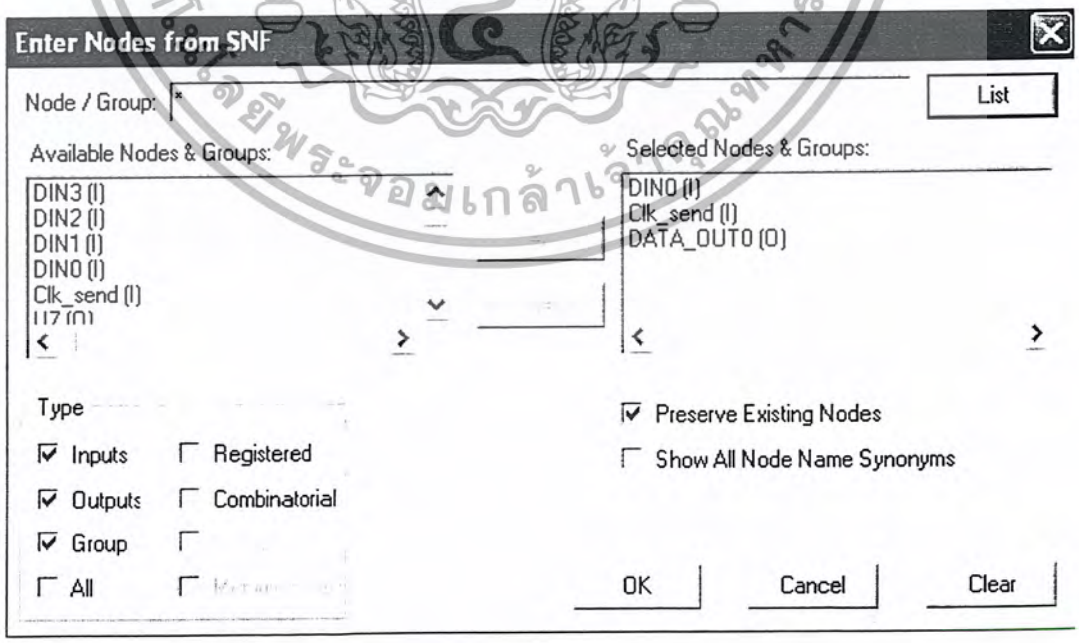


ภาพที่ 4.12 การเลือกสร้างไฟล์ Waveform



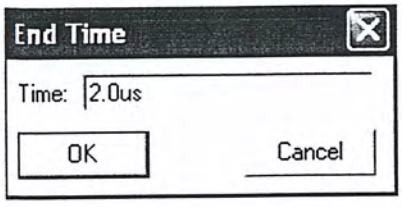
ภาพที่ 4.13 Waveform Editor

ก่อนทำการSimilate เราจะต้องกำหนดลักษณะของสัญญาณอินพุทให้แก่วงจรก่อน โดยการโหลดโหนดต่างๆ เข้ามาโดยใช้เมนู Noad/Enter Nodes form SNF...ซึ่งจะมีไดอะล็อก Enter Nodes form SNF ปรากฏขึ้นมา ต่อไปให้คลิกที่ปุ่ม List เพื่อแสดงโหนดต่างๆที่อยู่ในวงจรขึ้นมา จากนั้นเลือกโหนดที่เป็น input และ output ทั้งหมดที่อยู่ใน Listbox ทางด้านซ้ายโดยการดับเบิ้ลคลิกทุกๆรายการที่อยู่ในช่องนี้ คุณจะพบว่าในรายการต่างๆ จะเข้าไปอยู่ใน Listbox ทางด้านขวาหลังจากการดับเบิ้ลคลิกดังภาพที่ 4.14



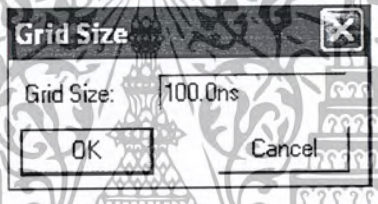
ภาพที่ 4.14 การเลือกโหนดของ input กับ output

กำหนดเวลาสิ้นสุดการ Similate(End Time) ให้กับโปรแกรมจากเมนู File/End Time... จากนั้นจะมีไดอะล็อก End Time ปรากฏขึ้นมา ให้ใส่ค่า End Time เท่ากับ 1.0 us ซึ่งจะเป็นการบอกให้โปรแกรมทำการSimilate ตั้งแต่ 0.0 us จนถึง 2.0 us ดังภาพที่4.15



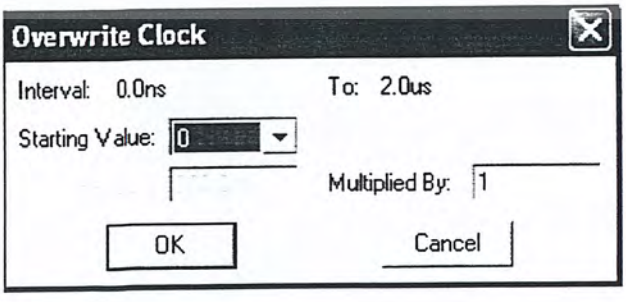
ภาพที่ 4.15 การกำหนดค่า End Time

กำหนดขนาดของกริดจากเมนู Option/Grid Size... ซึ่งจะมีไดอะล็อก Grid Size ปรากฏขึ้นมา จากนั้นกำหนดให้กริดมีขนาดเท่ากับ 100 ns ดังภาพที่ 4.16



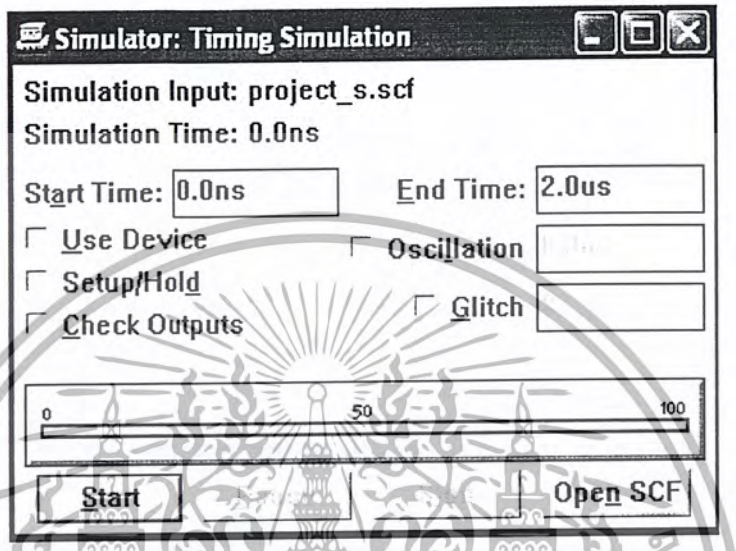
ภาพที่ 4.16 การกำหนดขนาดของกริด

ทำการกำหนดรูปแบบสัญญาณให้กับ Node Input โดยการคลิกที่CLK_Send ที่ Waveform Editor ให้มีแถบดำปรากฏขึ้นมาหลังจากนั้นทำการกำหนดรูปแบบสัญญาณให้มีลักษณะเป็นพัลซ์โดยใช้เมนู Edit / Overwrite / Clock โดยจะมีไดอะล็อก Overwrite Clock ปรากฏขึ้น ในช่องMultiply By ให้ใส่ 1 แล้วกดที่ปุ่ม OK ซึ่งจะเป็นการกำหนดให้สัญญาณในช่วงลอจิก “1” และลอจิก”0” มีค่าเวลาเป็น 1 เท่าของ Grid Size

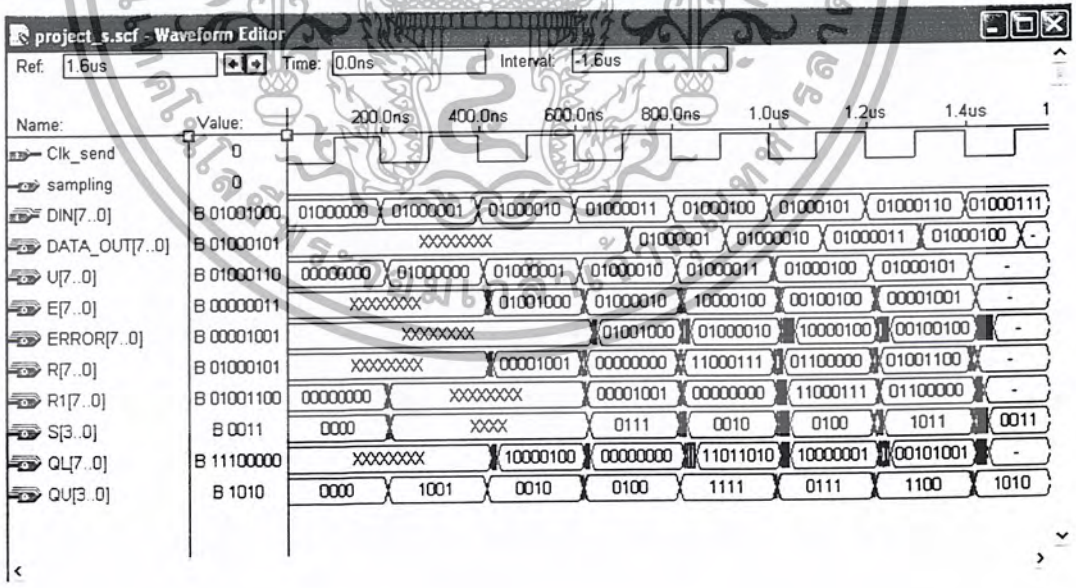


ภาพที่ 4.17 การกำหนดสัญญาณ Clock

จำลองการทำงานของวงจรจากเมนู MAX+PLUSII / Simulator ซึ่งจะมีหน้าต่าง Timing Simulation ปรากฏขึ้นดังภาพที่ 4.18 มาจากนั้นให้กดปุ่ม Start เพื่อทำการ Simulate วงจร และเมื่อการคำนวณเสร็จสิ้นลงก็จะมีไดอะล็อกขึ้นมารายงานผลการ Simulate ว่ามีerror หรือ warning หรือไม่ ส่วนผลของการ Simulate จะปรากฏที่ Waveform Editor ดังภาพที่ 4.19

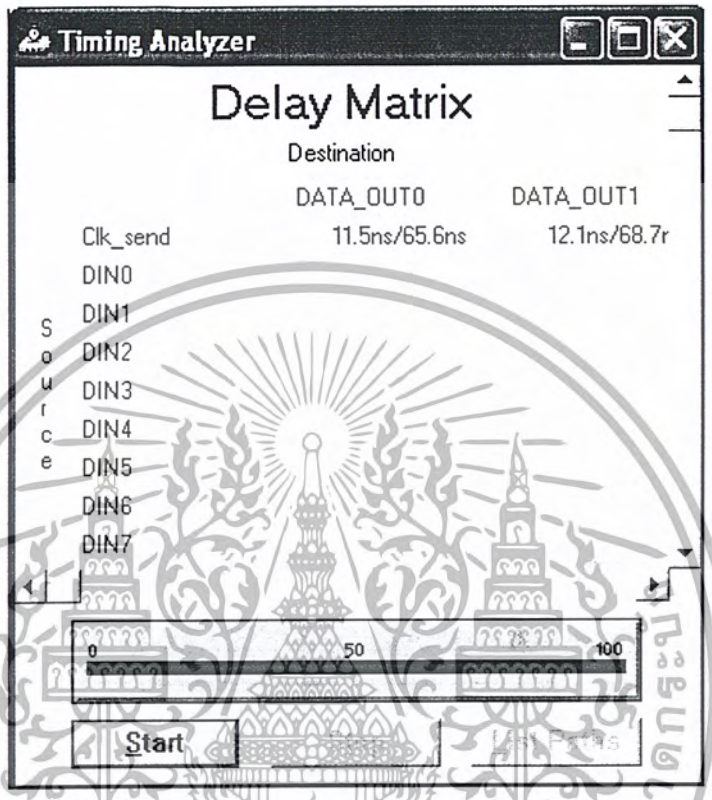


ภาพที่ 4.18 หน้าต่างของ Simulator



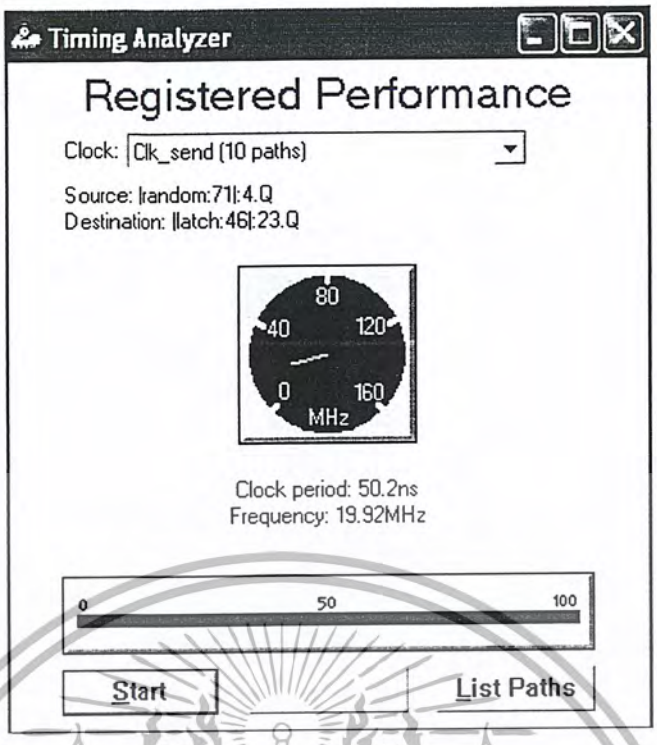
ภาพที่ 4.19 ผลการ Simulate

จากผลการจำลองการทำงานสามารถนำมาวิเคราะห์ Timing เพื่อใช้สำหรับหาค่า Delay Time ระหว่างโหนดต่างๆ โดยใช้เมนู MAX+PLUSII/Timing Analyzer ซึ่งจะมีไดอะล็อก Timing Analyzer ปรากฏขึ้นมา จากนั้นทำการวิเคราะห์ค่าเวลาหน่วงจากเมนู Analysis/Delay Time ดังภาพที่4.20



ภาพที่ 4.20 วิเคราะห์ค่าเวลาหน่วง

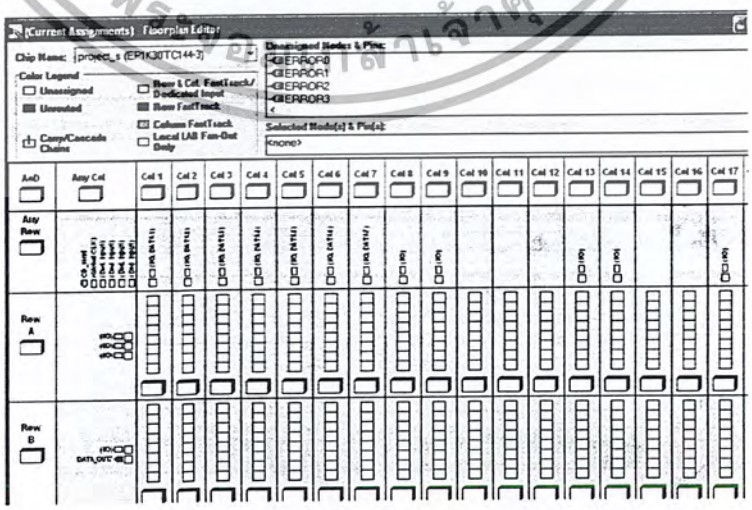
นอกจากนี้ ยังสามารถวิเคราะห์คำนวณหาค่าความถี่สูงสุดที่วงจรจะสามารถทำงานได้ โดยการเลือกที่เมนู Analysis/Registered Performance ดังภาพที่ 4.21



ภาพที่ 4.21 วิเคราะห์หาความถี่สูงสุดที่วงจรสามารถทำงานได้

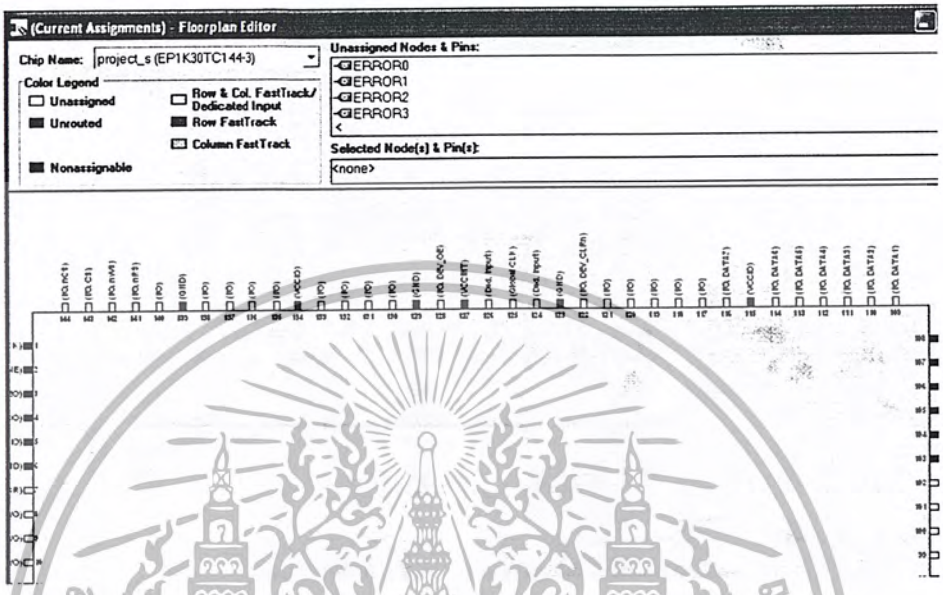
4.2.4 การโปรแกรมลงชิป FPGA

ขั้นตอนนี้จะเป็นการโปรแกรมวงจรที่ออกแบบไว้ลงในชิป FPGA ซึ่งก่อนที่จะทำการโปรแกรมนั้นจำเป็นต้องมีการกำหนดตำแหน่งขาของอุปกรณ์ต่างๆ ที่อยู่ในวงจรกับขาของ FPGA ก่อน โดยเริ่มจากการเรียกเมนู Max+Plus/FloorPlan Editor หลังจากนั้นหน้าต่าง Floorplan Editor จะปรากฏขึ้นมาดังภาพที่ 4.22ซึ่งเป็น Layout แบบ LAB View (Logic Array Block View)



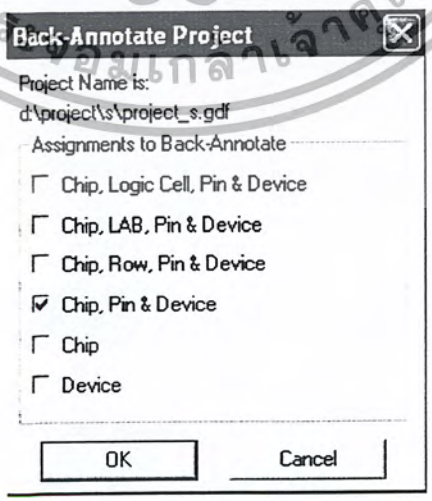
ภาพที่ 4.22 Floorplan Editor ใน Layout แบบ LAB View

เนื่องจาก Layout แบบ LAB View นี้ อาจพิจารณาได้ลำบากในแง่ของการต่อวงจร ดังนั้นจึงควรเปลี่ยน Layout ให้เป็นแบบ Device View ดังภาพที่ 4.23 ซึ่งLayout แบบนี้จะเป็นการมองจากตำแหน่งขาที่แท้จริงของ FPGA เบอร์ที่เราระบุไว้โดยการเรียกเมนู Layout/ Device View



ภาพที่ 4.23 Floorplan Editor ใน Layout แบบ Device view

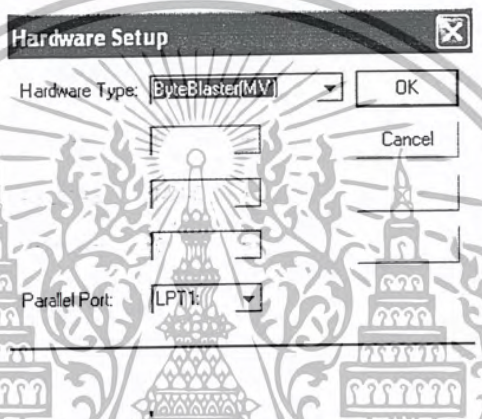
หลังจากนั้นทำการกำหนดอินพุทและเอาท์พุทของวงจรลงไป โดยเรียกเมนู Assign/Back-Annotate Project... ซึ่งจะมีไดอะล็อก Back-Annotate Project ปรากฏขึ้นมา ให้คลิกที่ Chip,Pin&Device/OK ดังภาพที่ 4.24



ภาพที่ 4.24 เลือก Chip,Pin&Device

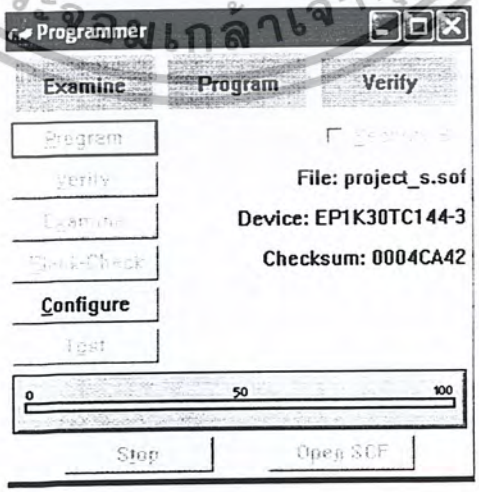
สังเกตได้ว่าเมื่อผ่านขั้นตอนนี้ไปแล้วโปรแกรม MAX+PLUSII จะทำการกำหนดตำแหน่งขาต่างๆ มาให้เรียบร้อยแล้วแต่อาจยังไม่เหมาะสมกับการเชื่อมต่อของบอร์ดทดลอง ดังนั้นจึงต้องทำการเปลี่ยนแปลงตำแหน่งขาต่างๆ ของชิป FPGA ก่อน โดยเลือกเมนู Layout/Current Assignment Floorplan

ทำการโปรแกรมวงจรลงในชิป FPGA โดยเลือกที่เมนู MAX+PLUSII / Programmer ซึ่งหากยังไม่เคยติดตั้งอุปกรณ์สำหรับโปรแกรมชิป FPGA (สาย Byte Blaster) ก็จะมีไดอะล็อก Hardware Setup ปรากฏขึ้นมา ให้นำสาย Byte Blaster มาต่อที่พอร์ตขนาน (พอร์ต Printer) แล้วเลือก Hardware Type เป็น Byte Blaster (MV) และกดปุ่ม OK ดังภาพที่ 4.25



ภาพที่ 4.25 Hardware Setup

เมื่อโปรแกรม MAX+PLUSII ตรวจพบว่ามีสาย Byte Blaster ต่ออยู่ที่พอร์ตขนานแล้ว หน้าต่าง Programmer จะเป็นดังภาพที่ 4.26 จากนั้นให้กดปุ่ม Program เพื่อโหลดวงจรลงชิป FPGA



ภาพที่ 4.26 Programmer

ข้อสังเกตอย่างหนึ่งในการโปรแกรมวงจรคือหาก Device ของ FPGA ที่เลือกเป็นอุปกรณ์ประเภท SRAM-Base FPGA เบอร์ IC จะขึ้นต้นด้วย EPF ไฟล์ที่โปรแกรมจะมีนามสกุลเป็น pof ซึ่งก่อนที่จะโปรแกรมวงจรลงในชิป FPGA ให้ทำการตรวจสอบก่อนว่าได้ทำการต่อสาย Byte Blaster กับบอร์ดทดลองเรียบร้อยแล้ว และได้ทำการจ่ายไฟให้กับบอร์ดทดลองแล้ว มิฉะนั้นการโปรแกรมข้อมูลลงในชิป FPGA จะไม่สามารถทำได้

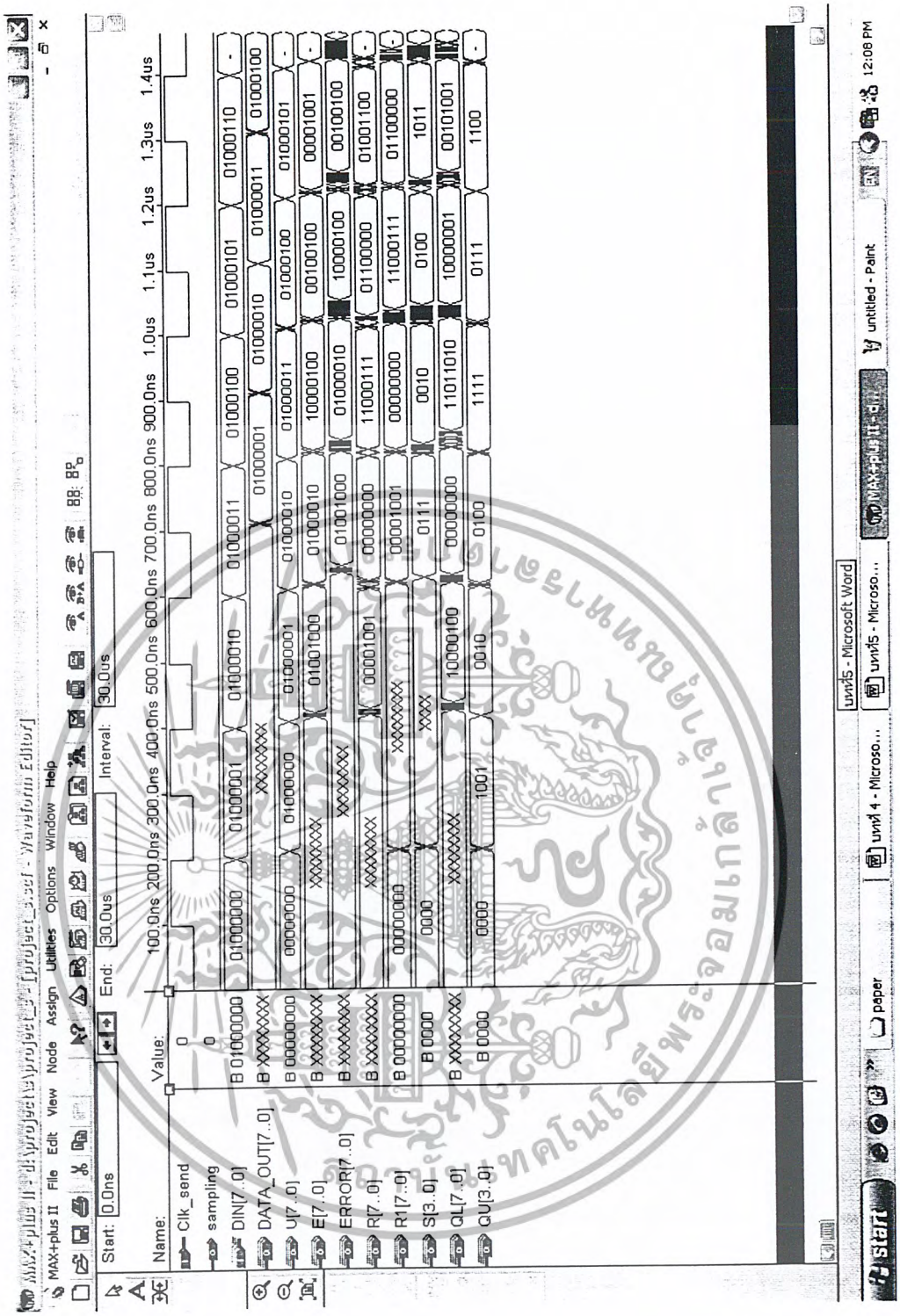
บทที่ 5

การทดลองและผลการทดลอง

5.1 การทดลอง

การทดลองเป็นการ Simulate โดยการป้อนสัญญาณดิจิตอล 8 บิท เข้าไปโดยเอาท์พุทที่ได้จากการเข้ารหัส (12 บิท) นำมาต่อกับอินพุทของการถอดรหัสเลย เพื่อที่จะดูว่าเอาท์พุทที่ได้จะเหมือนกับอินพุทหรือไม่

DIN[7..0]	คือ สัญญาณดิจิตอล 8 บิท ที่ป้อนเข้าไป
DATA_OUT [7..0]	คือ สัญญาณเอาท์พุท 8บิทที่ทำการถอดรหัสแล้ว
U [7..0]	คือ สัญญาณที่มีค่าท่ากับสัญญาณดิจิตอล 8 บิท ที่ป้อนเข้าไป ตามสมการที่ 4.1
E [7..0]	คือ รูปแบบผิดพลาดที่นำไป XOR กับ U
R [7..0]	คือ ผลที่ได้จาก U XOR กับ E
QL [7..0]	คือ เอาท์พุท 8 บิทที่ได้จากการเข้ารหัส และส่งไปยังตัวถอดรหัส
QU [3..0]	คือ เอาท์พุท 4 บิทที่ได้จากกระทำตามสมการ 4.1 และเป็นตัวเลือกรูปแบบในการสลับบิท และการสลับบิทกลับ
R1 [7..0]	คือ ผลที่ได้จากการสลับบิทกลับจากข้อมูลที่ได้มาจากการเข้ารหัส
S [3..0]	คือ ค่าซีนโดรมซึ่งกระทำตามสมการ 4.2
ERROR [7..0]	คือ รูปแบบผิดพลาดที่ได้จากค่าซีนโดรม



ภาพที่ 5.1 ผลจากการ Simulate

5.2 สรุปผลการทดลอง

จากการทดลองจะเห็นได้ว่าสัญญาณเอาท์พุทที่ได้เหมือนกับสัญญาณอินพุท แต่จะ DELAY เมื่อผลการSimulate ออกมาตรงแล้วก็ทำการกำหนดขาอินพุทเอาท์พุทต่างๆที่ต้องการใช้งาน แล้วทำการ Program ลงบนชิป FPGA โดยใช้โปรแกรม MAX+ PLUS II โดยวิธีการใช้งานได้อธิบายไปแล้วในบทที่ 4

บทที่ 6
สรุปผล

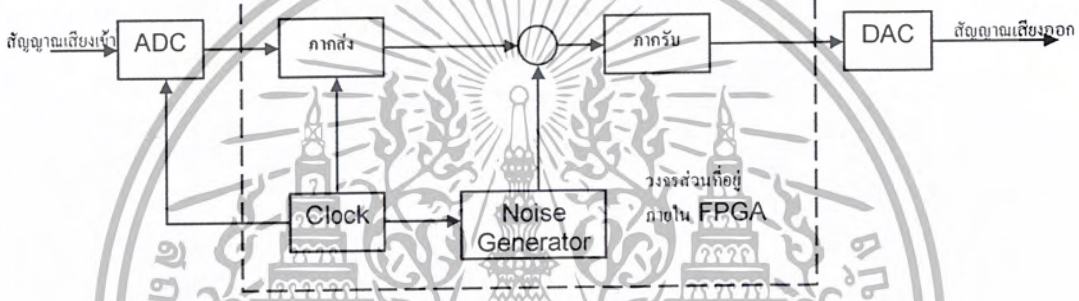
6.1 บทสรุป

จากการออกแบบ การสร้างและการทดลอง พบว่าระบบปกปิดข้อมูลนี้ได้รับการพัฒนาขึ้นจากอุปกรณ์แบบแยกส่วนเป็นอุปกรณ์ FPGA มีตัวนับลำดับแบบสุ่มที่ยาวขึ้น และเพิ่มเติมวงจรจัดเฟรมข้อมูลและทดลองจัดเปลี่ยนวงจร โดยการกลับคำของบิทต่างๆเพื่อให้มีการเข้ารหัสสัญญาณเสียงและข้อมูลภาพที่มีประสิทธิภาพมากกว่างานวิจัยเดิม รวมไปถึงความปลอดภัยของตัววงจรและความสะดวกในการพัฒนาอันเนื่องมาจากข้อดีของ FPGA

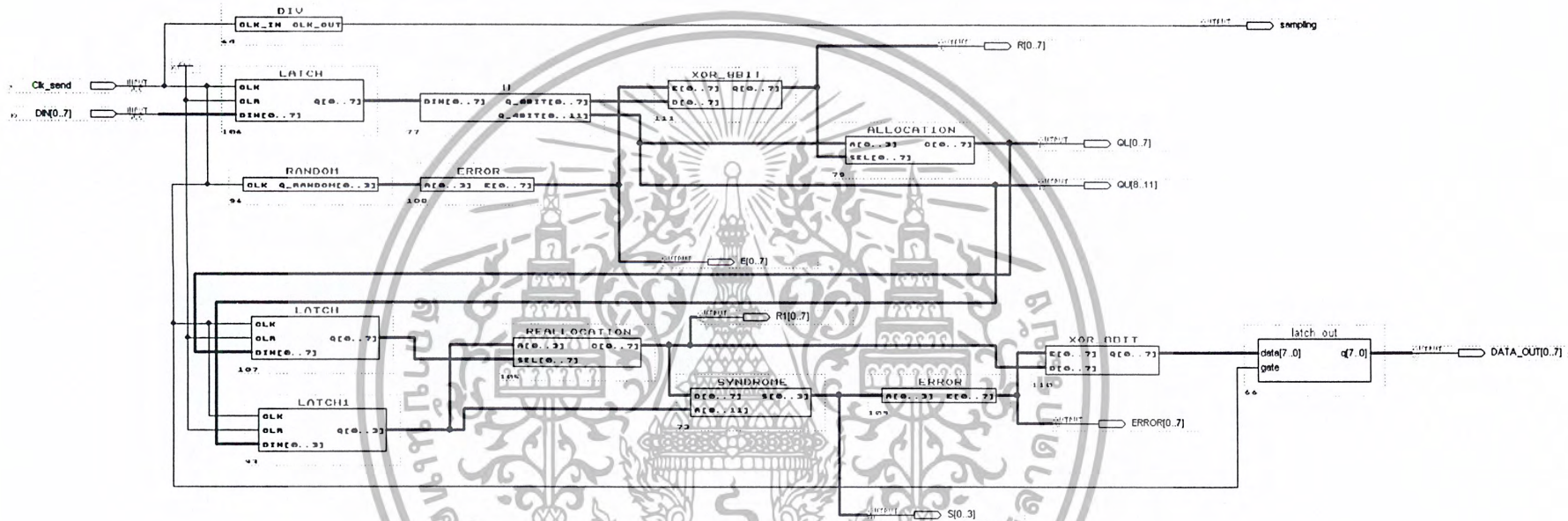
บรรณานุกรม

1. ฟูศักดิ์ ชีวสุวิทย์ "การแก้รหัสผิด" คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง พ.ศ.2538
2. ฟูศักดิ์ ชีวสุวิทย์ และคเชนทร์ แจ่มกมล. "การเข้ารหัสลับโดยใช้ซีนโครมของลิเนียร์บล็อกโค้ดและการสลับบิตโดยอาศัยเพอซีโดแรนดอม." วารสารทางวิชาการของสมาคมคอมพิวเตอร์แห่งประเทศไทย
3. ฟูศักดิ์ ชีวสุวิทย์ และคเชนทร์ แจ่มกมล. "การออกแบบตัวเข้ารหัสลับแบบเวลาจริงโดยใช้ซีนโครมของลิเนียร์บล็อกโค้ดเชิงเส้นและการสลับบิตโดยอาศัยการกำเนิดแบบสุ่ม" การประชุมวิชาการทางไฟฟ้าครั้งที่ 18 มหาวิทยาลัยเทคโนโลยีมหานคร,หน้า 1034-1038, พฤศจิกายน 2538

ภาคผนวก



บล็อกผังวงจรรวมของระบบปกปิดข้อมูล



วงจรที่ออกแบบโดยใช้โปรแกรม MAX PLUS II