การออกแบบวงจรรวมขนาดใหญ่มากสำหรับการประมวลผลรหัสลับไอดีอีเอ

A Very Large Scale Integrated Circuit Implementation of International Data Encryption Algorithm

สาวิตร์ ตัณฑนุช

Sawit Tanthanuch

วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

มหาวิทยาลัยสงขลานครินทร์

Master of Engineering Thesis in Electrical Engineering

Prince of Songkla University

2544

ชื่อวิทยานิพนธ์ การออกแบบวงจรรวมขนาดใหญ่มากสำหรับการประมวลผลรหัสลับไอดีอีเอ

ผู้เขียน นายสาวิตร์ ตัณฑนุช

สาขาวิชา วิศวกรรมไฟฟ้า

คณะกรรมการที่ปรึกษา

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.เกริกชัย ทองหนู)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ชัชวาล ยนต์หงส์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ สมพัฒน์ รุ่งตะวันเรืองศรี)

คณะกรรมการสอบ

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.เกริกชัย ทองหนู)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ชัชวาล ยนต์หงส์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ สมพัฒน์ รุ่งตะวันเรืองศรี)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ บุญเจริญ วงศ์กิตติศึกษา)

.....กรรมการ
(อาจารย์ไพบูลย์ นวลนิล)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

.....
(รองศาสตราจารย์ ดร.ปิติ ทฤษฎิคุณ)
คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์	การออกแบบวงจรรวมขนาดใหญ่มากสำหรับการประมวลผลรหัสลับไอดีอีเอ
ผู้เขียน	นายสาวิตร์ ตัณฑนุช
สาขาวิชา	วิศวกรรมไฟฟ้า
ปีการศึกษา	2544

บทคัดย่อ

งานวิจัยนี้นำเสนอการพัฒนาระบบรหัสลับขนาด 64 บิตแบบบล็อกชนิดไอดีอีเอด้วยวงจรรวมขนาดใหญ่มากแบบโปรแกรมซ้ำได้ชนิดเอฟพีจีเอเบอร์เอ็กซ์ซี 4062 เอ็กซ์แอลเอ 09 รุ่นเอชคิว 240 โดยใช้ภาษาบรรยายฮาร์ดแวร์ร่วมกับใช้แผนผังไฟฟ้าโดยมีเงื่อนไขบังคับด้วยขนาดของพื้นที่น้อยที่สุด ผลการออกแบบทำให้ระบบรหัสลับทำงานได้ที่ 9.1 เมกกะเฮิรตซ์และทำงานครบวงรอบสมบูรณ์ 9 รอบ ด้วยคาบเวลา 5.8232 ไมโครวินาทีทำให้ได้อัตราการส่งผ่านข้อมูลที่เข้ารหัสแล้ว 13.3 เมกกะบิตต่อวินาทีและใช้ทรัพยากรไป 1381 ซีแอลบีหรือร้อยละ 59.95 ของวงจรรวมทั้งหมด

คำสำคัญ : รหัสลับ, ไอดีอีเอ, เอฟพีจีเอ, วงจรรวม

Thesis Title	A Very Large Scale Integrated Circuit implementation of International Data Encryption Algorithm
Author	Mr. Sawit Tanthanuch
Major Program	Electrical Engineering
Academic Year	2001

Abstract

This research presents the implementation of 64-bit block cipher international data encryption algorithm (IDEA) using reconfigurable logic device XC4062XLA09-HQ240 field programmable gate array (FPGA). Hardware Description Language (HDL) and schematic capture were used for the design entry and synthesis with emphasis on area usage optimization. The designed system operated at 9.1 MHz when the process took 5.8232 μs for 9 rounds full block crypt with the 13.3 Mbit/s throughput. The design used 1381 configurable logic blocks(CLBs) or 59.95% of the available silicon resources.

Keyword : Cryptography, IDEA, FPGA, VLSI

กิตติกรรมประกาศ

ผู้เขียนขอแสดงคำขอบพระคุณต่อ ผู้ช่วยศาสตราจารย์ ดร.เกริกชัย ทองหนู ประธานกรรมการที่ปรึกษา ผู้ช่วยศาสตราจารย์ ชัชวาล ยนต์หงส์ และ ผู้ช่วยศาสตราจารย์ สมพัฒน์ รุ่งตะวันเรืองศรี กรรมการที่ปรึกษา ที่ได้ให้คำแนะนำ และการสนับสนุนจนทำให้วิทยานิพนธ์นี้สำเร็จลุล่วงไปด้วยดี

ขอขอบคุณผู้ช่วยศาสตราจารย์ บุญเจริญ วงศ์กิตติศึกษาและ ผู้ช่วยศาสตราจารย์ เลียง คูบุรัตถ์ ที่ได้ช่วยเหลือในการจัดหาอุปกรณ์ต่างๆสำหรับการทำวิจัยตลอด จนช่วยตรวจแก้ไขวิทยานิพนธ์ให้ดำเนินไปอย่างสมบูรณ์

ขอขอบคุณอาจารย์ปราโมทย์ จูฑาพร และ อาจารย์ไพบูลย์ นวลนิล ที่กรุณาช่วยตรวจแก้ไขวิทยานิพนธ์ จนบรรลุวัตถุประสงค์

ขอขอบคุณผู้ช่วยศาสตราจารย์ ดร.อำนวย สิทธิชัยเจริญ อาจารย์ วีระพันธุ์ มุสิกสาร อาจารย์ฉัตรชัย จันทรพริ้ม อาจารย์วรรณรัช สันติอมรทัต และ อาจารย์ปัญญายศ ไชยการ ภาควิชาวิศวกรรมคอมพิวเตอร์ ที่ให้ความอนุเคราะห์ในเรื่องฮาร์ดแวร์และซอฟต์แวร์ที่ใช้สำหรับงานวิจัยนี้

ขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร. ชูศักดิ์ ลิ่มสกุล ดร.ปราการ คุรุหงษา และ ดร.นิตยา ซีการ์ ที่สนับสนุนให้ผู้เขียนได้เข้าศึกษาในสถาบันแห่งนี้ รวมไปถึงคณาจารย์และบุคลากรในภาควิชาวิศวกรรมไฟฟ้าทุกท่านที่ได้ให้คำปรึกษาและให้ความช่วยเหลือในด้านต่างๆ จนงานวิจัยสามารถสำเร็จลุล่วงได้เป็นอย่างดี

ขอขอบพระคุณ บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ ที่ให้การสนับสนุนทุนในการทำวิจัย

สุดท้ายนี้ข้าพเจ้าขอน้อมรำลึกถึงพระคุณของ บิดา มารดา และกำลังใจจากบุคคลในครอบครัวที่ส่งเสริมและสนับสนุนในทุกๆเรื่องจนสำเร็จการศึกษา

สาวิตร์ ตัณฑนุช

สารบาญ

	หน้า
บทคัดย่อ	(3)
Abstract	(4)
กิตติกรรมประกาศ	(5)
สารบาญ	(6)
รายการตาราง	(9)
รายการภาพประกอบ	(10)
ตัวย่อและสัญลักษณ์	(12)
บทที่	
1. บทนำ	1
1.1 ความสำคัญและที่มาของหัวข้อวิจัย	1
1.2 การตรวจเอกสาร	3
1.3 วัตถุประสงค์	5
1.4 ประโยชน์ที่คาดว่าจะได้รับ	6
1.5 ขอบเขตของการวิจัย	6
1.6 ขั้นตอนและวิธีดำเนินงานวิจัย	6
2. ขั้นตอนและวิธีการของรหัสลับไอดีอีเอ	7
2.1 บทนำ	7
2.2 หลักการของรหัสลับไอดีอีเอ	8
2.3 ความรู้เบื้องต้นเกี่ยวกับทฤษฎีจำนวนและพีชคณิตนามธรรม	10
2.4 กลไกการเข้ารหัส	13
2.5 กลไกการถอดรหัส	14
2.6 ความปลอดภัยของระบบรหัสลับไอดีอีเอ	18
2.7 รหัสกุญแจที่อ่อนแอ	21
2.8 สรุป	21
3.โปรแกรมคอมพิวเตอร์สำหรับรหัสลับไอดีอีเอ	22
3.1 บทนำ	22

3.2 การสร้างรหัสกุญแจย่อย 52 ชุด 23
3.3 ตัวดำเนินการมอดุโลของผลบวกและมอดุโลวิธีปรับปรุงของผลคูณ 24
3.4 ตัวดำเนินการค่าผกผันมอดุโลวิธีปรับปรุงของผลคูณและค่าผกผันมอดุโลของผลบวก 25
3.5 การสร้างกุญแจสำหรับการถอดรหัส 27
3.6 ส่วนประมวลผลรหัส 29
3.7 สรุป 29
4. สถาปัตยกรรมฮาร์ดแวร์ของวงจรรวมต้นแบบ 30
4.1 บทนำ 30
4.2 โครงสร้างและสถาปัตยกรรมภายในเอฟพีจีเอ 30
4.3 การเลือกวงจรฮาร์ดแวร์เพื่อสร้างต้นแบบ 41
4.4 สรุป 41
5. การออกแบบวงจรดิจิตอล 42
5.1 บทนำ 42
5.2 การออกแบบด้วยภาษาบรรยาย 42
5.3 การออกแบบด้วยแผนผังไฟฟ้า 48
5.4 การสังเคราะห์วงจร 49
5.5 สรุป 49
6. ผลการทดลอง 51
6.1 บทนำ 51
6.2 การทดสอบฟังก์ชันเข้าและถอดรหัส 51
6.3 การทดสอบการสังเคราะห์วงจร 54
6.4 การจำลองการสังเคราะห์วงจร 55
6.5 การทดสอบทางไฟฟ้า 57
6.6 สรุป 57
7. สรุปผลการวิจัยและข้อเสนอแนะ 58
7.1 บทนำ 58

7.2 สรุปผลการวิจัย 58
7.3 การเปรียบเทียบคุณสมบัติกับงานวิจัยอื่น 59
7.4 บทวิจารณ์และข้อเสนอแนะ 59
บรรณานุกรม 62
ภาคผนวก
ภาคผนวก ก แผนผังทางเวลาของวงจรรวมต้นแบบ 65
ภาคผนวก ข แผนผังทางไฟฟ้าของวงจรต่างๆ 94
ภาคผนวก ค โปรแกรมรหัสต้นฉบับ 102
ภาคผนวก ง รายงานสมบูรณ์ของการสังเคราะห์วงจร 115
ประวัติผู้เขียน 118

รายการตาราง

ตาราง	หน้า
1.1 หน่วยงานที่นำรหัสลับไอดีอีเอไปใช้งาน	2
1.2 การเปรียบเทียบสมรรถนะวงจรรวมตามรายงานวิจัยของ E.Caspi และคณะ	5
2.1 การกระจายรหัสกุญแจสำหรับการเข้ารหัสไอดีอีเอ	8
2.2 รหัสกุญแจสำหรับการเข้าและถอดรหัสไอดีอีเอ	18
5.1 การเปรียบเทียบคุณลักษณะของวงจรบวกด้วยวิธีการต่างๆ	44
5.2 การเปรียบเทียบคุณลักษณะของวงจรคูณด้วยวิธีการต่างๆ	46
6.1 สัญญาณควบคุมวงจรเข้าและถอดรหัส	53
6.2 ผลที่คำนวณได้จากโปรแกรมสำหรับการเข้ารหัส	53
6.3 ผลที่คำนวณจากโปรแกรมสำหรับการถอดรหัส	54
7.1 คุณสมบัติของวงจรรวมต้นแบบสำหรับการประมวลผลรหัสลับไอดีอีเอ	59

รายการภาพประกอบ

ภาพประกอบ	หน้า
2.1 แผนผังการเข้ารหัสแบบสมมาตร	7
2.2 แผนผังการเข้ารหัสแบบอสมมาตร	7
2.3 แผนผังรหัสลับไอดีอีเอ	9
2.4 กราฟการคณนาโครงสร้างของคูณบวก	15
2.5 กราฟการคณนาอาวัตการในฟังก์ชัน $I_n(\bullet, Z_B)$	15
2.6 กราฟการคณนาของ 2 รอบแรกในการวิเคราะห์แบบอนุพันธ์	19
3.1 แผนภูมิสายงานของโปรแกรมระบบรหัสลับไอดีอีเอ	22
3.2 ฟังก์ชันการสร้างรหัสกุญแจย่อย 52 ชุด	23
3.3 ฟังก์ชันตัวดำเนินการมอดุโลวิธีปรับปรุงของผลคูณ	25
3.4 รหัสเทียมของขั้นตอนวิธียุคลิดสำหรับการหาตัวหารร่วมมาก	25
3.5 ฟังก์ชันขั้นตอนวิธีของยุคลิดสำหรับการหาตัวหารร่วมมาก	26
3.6 รหัสเทียมของขั้นตอนทวินามวิธียุคลิดสำหรับการหาตัวหารร่วมมาก	26
3.7 ฟังก์ชันขั้นตอนทวินามวิธีของยุคลิดสำหรับการหาค่าผกผันมอดุโล วิธีปรับปรุงของผลคูณ	27
3.8 ฟังก์ชันการหาค่าผกผันมอดุโลของผลบวก	27
3.9 ฟังก์ชันการกระจายกุญแจย่อย 52 ชุดสำหรับการถอดรหัส	28
3.10 ฟังก์ชันการประมวลผลรหัส	28
4.1 โครงสร้างพื้นฐานภายในเอฟพีจีเอของบริษัท Xilinx	30
4.2 แผนผังภายในซีแอลบี	31
4.3 แผนผังภายในไอโอบี	33
4.4 แผนผังการเชื่อมโยงซีแอลบี	34
4.5 โครงสร้างของพีเอสเอ็ม	34
4.6 การเชื่อมโยงพีเอสเอ็มและพีไอพีของซีแอลบี	35
4.7 การเชื่อมโยงแบบการเชื่อมโยงเส้นเดี่ยวและเส้นคู่	36
4.8 การเชื่อมโยงแบบ 4 เส้น	37
4.9 การเชื่อมโยงโดยตรง	38
4.10 การเชื่อมโยงวงแหวนแบบเวอร์ซา	38

รายการภาพประกอบ(ต่อ)

ภาพประกอบ	หน้า
4.11 การเชื่อมโยงแบบแปดเส้น	39
4.12 แผนผังการเชื่อมโยงไอโอบี	40
4.13 วงจรทดลองต้นแบบใช้เอฟพีจีเอรุ่น XC4062XLA09-HQ240	41
5.1 ขั้นตอนการออกแบบสำหรับเอฟพีจีเอ	42
5.2 แผนภาพเอฟเอสเอ็มของวงจรควบคุมวงรอบการเข้า/ถอดรหัส	46
5.3 แผนภาพเอฟเอสเอ็มของการกระจายกุญแจย่อย	47
5.4 แผนภาพเอฟเอสเอ็มของการหาค่าผกผันด้วยขั้นตอนทวินามวิธีของยุคลิด	48
5.5 แผนผังวงจรต้นแบบสำหรับรหัสลับไอดีอีเอ	49
6.1 การจำลองสัญญาณด้วยโปรแกรม ModelSIM สำหรับการเข้ารหัส	51
6.2 การจำลองสัญญาณด้วยโปรแกรม ModelSIM สำหรับการถอดรหัส	52
6.3 ขั้นตอนการสังเคราะห์วงจร	54
6.4 ผลการทดสอบเพื่อหาอัตราการส่งผ่านข้อมูลสูงสุด	55
6.5 แผนผังทางไฟฟ้าการทดสอบภายใน	56
6.6 แผนผังทางเวลาของการทดสอบจากฐานเวลาใน XC4000	56
6.7 การโปรแกรมเอฟพีจีเอ	57
6.8 อุปกรณ์เชื่อมต่อ Xchecker	57
7.1 การเชื่อมโยงแบบอนุกรมสำหรับการประมวลผลรหัสลับไอดีอีเอ	61
ก1 ผลการจำลองสัญญาณทางเวลาของวงจรที่สังเคราะห์ขึ้น	66
ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส	67
ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส	82
ข1 แผนผังทางไฟฟ้าของชุดทดสอบการประมวลผลรหัสลับด้วยสัญญาณนาฬิกาภายใน	95
ข2 แผนผังทางไฟฟ้าของวงจรรวมต้นแบบเอฟพีจีเอเบอร์ XC4062XLA09-HQ240	96

ตัวย่อและสัญลักษณ์

$\oplus$ = ตัวดำเนินการเอ็กซ์คลูซีฟออร์เกตแบบบิต (bitwise Ex-OR)

$\boxplus$ = ตัวดำเนินการมอดุโลของผลบวก (addition modulo)

$\odot$ = ตัวดำเนินการวิธีปรับปรุงมอดุโลของผลคูณ(modified multiplication modulo)

$\mathbb{Z}_n$ = เซตของจำนวนเต็มบวกใดๆที่มีสมาชิกเป็น {0,1,2,3,...} โดยสมาชิกแต่ละตัวจะเป็นเศษที่เหลือจากการหารด้วย n

$\in$ = เป็นสมาชิกของ

$\subset$ = เป็นเซตย่อย

$\approx$ = สมสัณฐาน (isomorphism)

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของหัวข้อวิจัย

การแลกเปลี่ยนข้อมูลสารสนเทศด้วยสื่ออิเล็กทรอนิกส์เช่นอินเตอร์เน็ต เครือข่ายท้องถิ่น โครงข่ายข้อมูลดิจิตอล ฯลฯ จะอาศัยการเข้ารหัสเพื่อปกปิดและซ่อนพลางข้อมูล การศึกษากลไกและกระบวนการเข้ารหัสจึงได้ถูกบรรจุในหลักสูตรการเรียนการสอนในสถาบันการศึกษาในต่างประเทศในรูปแบบของการศึกษาวิธีการสร้างรหัสลับ การวิเคราะห์ และการประยุกต์ใช้งาน และจากการสำรวจในเบื้องต้น ยังไม่พบว่าสถาบันการศึกษาในประเทศได้บรรจุหลักสูตรการเรียนการสอนในเรื่องดังกล่าวอย่างเป็นรูปธรรม จึงทำให้การใช้งานรหัสลับในประเทศไทยไม่เกิดประสิทธิภาพและมีประสิทธิผลอย่างสมบูรณ์ เนื่องจากขาดความรู้ ความเข้าใจ ในการเลือกใช้ตรวจสอบความสามารถในการปกปิดซ่อนพรางข้อมูลและความแข็งแกร่งในการต่อต้านระบบการวิเคราะห์รหัสโดยปราศจากกุญแจที่ถูกต้อง

การเข้ารหัสแบบดีอีเอส (Data Encryption Standard - DES) ซึ่งได้รับรองเป็นมาตรฐานการเข้ารหัสเพื่อใช้งานในรัฐบาลกลางของสหรัฐอเมริกาเป็นรหัสที่เชื่อว่ามีความปลอดภัยสูง แต่ด้วยเงื่อนไขข้อตกลงองค์การความร่วมมือแอนแลนติกเหนือ (North Atlantic Treaty Organization- NATO) รหัสลับดีอีเอสถูกจำกัดการใช้งานเฉพาะกลุ่มประเทศอเมริกาเหนือเท่านั้น ประเทศอื่นๆที่ไม่ได้ร่วมในกลุ่มดังกล่าวจึงมีการพัฒนารหัสลับแบบต่างๆใช้งานขึ้นเองโดยมีโครงสร้างและตัวดำเนินการคล้ายกับรหัสลับดีอีเอส (DES-like) อาทิ Blowfish, FEAL, GOST, REDOC และ SAFER รหัสลับบางระบบได้ถูกวิจัยและพัฒนากระบวนการวิเคราะห์รหัสลับ จนทำให้ถูกลดความน่าเชื่อถือในการปกปิดและซ่อนพลางข้อมูล ดังมีรายงานการประชุมและงานวิจัยเกี่ยวกับการวิเคราะห์รหัสลับแบบตั้งแต่ปี ค.ศ. 1990 ถึง ปี ค.ศ. 1998 มากกว่า 20 รายงาน

ไอดีอีเอ (International Data Encryption Algorithm - IDEA) เป็นผลงานวิจัยของ Xuejia Lai และคณะในปี ค.ศ. 1990 ในเริ่มต้นใช้ชื่อว่า "พีอีเอส" (Proposed Encryption Standard - PES) เพื่อแก้ไขข้อจำกัดของรหัสลับดีอีเอสที่อนุญาตใช้งานเชิงพาณิชย์ด้วยรหัสกุญแจ (key) ไม่เกิน 64 บิต นอกกลุ่มประเทศอเมริกาเหนือ ในปีถัดมา Eli Biham และคณะได้เสนองานวิจัยเกี่ยวกับการวิเคราะห์รหัสลับเป็นผลให้รหัสลับพีอีเอสเกิดความไม่สมบูรณ์ จึงมีการปรับปรุงกลไกการเข้ารหัสใหม่เพื่อให้ยากต่อการวิเคราะห์ รหัสลับใหม่นี้คือ "ไอพีอีเอส" (Improved Proposed

Encryption Standard - IPES) และปี ค.ศ. 1992 Xuejia Lai ได้รับทุนสนับสนุนงานวิจัยนี้จาก สถาบันเทคโนโลยีแห่งรัฐบาลกลางสวิสเซอร์แลนด์ (Swiss Federal Institute of Technology - ETH) ทำให้รหัสลับนี้ถูกปรับปรุงเพื่อการใช้งานสำหรับประเทศในกลุ่มยุโรป โดยใช้ชื่อว่า International Data Encryption Algorithm หรือ IDEA (A.J. Menezes. *et al.* ,1997 :263-265)

รหัสลับไอดีอีเอเป็นรหัสลับแบบสมมาตรที่มีกระบวนการเข้าและถอดรหัสลับของข้อมูลโดยใช้กุญแจเดียวกันระหว่างผู้รับและผู้ส่ง กลไกการเข้าและถอดรหัสอาศัยตัวดำเนินการทางคณิตศาสตร์แบบกลุ่ม (group operation) ในทฤษฎีจำนวน (number theory) มีรหัสกุญแจ 128 บิตสำหรับการประมวลผลสัญญาณข้อมูลจึงมีความปลอดภัยสูงเมื่อเทียบกับรหัสลับสมมาตรแบบอื่นๆ และยังไม่มีรายงานการวิจัยใดที่แสดงให้เห็นว่ารหัสลับนี้ถูกวิเคราะห์ได้อย่างสมบูรณ์ ดังนั้นรหัสลับไอดีอีเอจึงได้รับการยอมรับและถูกนำไปใช้ในหน่วยงานสำคัญต่างๆ ดังแสดงในตารางที่ 1.1

ตารางที่ 1.1 หน่วยงานที่นำรหัสลับไอดีอีเอไปใช้งาน

ประเภทกิจการ	รายนาม
อุตสาหกรรม	Ascom Autelca (Switzerland) Audi AG (Germany) BMW Rolls Royce GmbH (Germany) Deutsche Telekom (Germany) Mitsubishi Electric (Japan) Motorola (USA) Nissin Electric Co. (Japan) Siemens (Germany) Swiss Online (Switzerland) Volkswagen AG (Germany)
ระบบคอมพิวเตอร์และสื่อสาร	Cray Communications (Denmark) FTP Software (USA) Highware Inc. (Belgium) IBM (USA) Lange Electronic (Germany) Lemcom Systems Inc. (USA) / PGP Inc. (USA) Lightning Instrumentation (Switzerland) Marx Datentechnik (Germany) Netscape Communications (USA) Network Systems Corp. (USA) RVS Datentechnik (Germany) Utimaco Safeware (Germany) X*Press Information Services (USA)
ธนาคาร ประกันภัย บริษัท	DBS Bank (Singapore) Union Bank of Switzerland Swiss Bank Corporation

ตารางที่ 1.1 หน่วยงานที่นำรหัสลับไอดีอีเอไปใช้งาน (ต่อ)

ประเภทกิจการ	รายนาม
หน่วยงานราชการ สถานพยาบาล	Dirección General Impositiva (Argentina) Federal Defense Department (Switzerland) Kassenärztliche Bundesvereinigung (Germany)
ขนส่งมวลชน	Munich Airport (Germany)
อื่นๆ	Reuters Asia (Hong Kong)

(ที่มา : B.Schneier,1996)

1.2 การตรวจเอกสาร

เนื่องจากรหัสลับไอดีอีเอใช้ตัวดำเนินการทางคณิตศาสตร์แบบกลุ่ม ในทฤษฎีจำนวน จึงมีการประยุกต์ใช้งานรหัสลับไอดีอีเอในรูปของโปรแกรมคอมพิวเตอร์อยู่หลายแบบ ลักษณะดังกล่าวทำให้อัตราการส่งผ่านข้อมูลไม่มากนัก เช่น การพัฒนาด้วยภาษาซีบนเครื่อง VAX-9000 จะมีอัตราการส่งผ่านข้อมูล 355.5 กิโลบิตต่อวินาที และเมื่อใช้โปรแกรมเดียวกันประมวลผลบนเครื่อง Sun SPARCstation 2 จะมีอัตราการส่งผ่าน ข้อมูล 400 กิโลบิตต่อวินาที (R.Zimmermann *et al.*,1994:306) การพัฒนาด้วยภาษาซีและแอสเซมบลีแบบ 4 ชุด (4-way IDEA) บนหน่วยประมวลผลกลาง Pentium II ที่สัญญาณนาฬิกา 450 เมกกะเฮิรตซ์ ด้วยสถาปัตยกรรม MMX (MultiMedia eXtensions) ได้อัตราการส่งข้อมูล 23.6 เมกกะบิตต่อวินาที (H.Lipmaa ,1998:248-263) เป็นต้น ทั้งนี้เพราะการออกแบบวงจรอิเล็กทรอนิกส์สำหรับตัวดำเนินการต่างๆในกลไกการเข้ารหัสแบบไอดีอีเอนั้นมีความยุ่งยากและซับซ้อน สำหรับรายงานวิจัยการออกแบบวงจรรวมเพื่อสร้างระบบการเข้ารหัสแบบไอดีอีเอใน ต่างประเทศมีดังนี้

1.2.1 ค.ศ. 1991 A. Curiger และคณะได้เสนองานวิจัยเรื่อง "Regular VLSI-architecture for multiplication modulo (2^n+1)" ซึ่งรายงานการออกแบบวงจรรวมสำหรับตัวดำเนินการมอดุโลผลคูณโดยใช้เทคโนโลยี double-metal CMOS 1.2 μm (CMN12) ในการออกแบบ ประเมินพื้นที่ได้ 1.51 ตารางมิลลิเมตร บรรจุทรานซิสเตอร์จำนวน 12028 ตัว สามารถประมวลผลโดยใช้สัญญาณนาฬิกา 2 ลูก (A.Curiger *et al.*, 1991:990-994)

1.2.2. ค.ศ. 1993 A. Curiger และคณะได้เสนองานวิจัยเรื่อง "VINCI: VLSI implementation of the new block cipher IDEA" โดยใช้เทคโนโลยี CMN12 ซึ่งเป็นผลจาก

งานวิจัยในปี ค.ศ. 1991 มีอัตราการส่งผ่านข้อมูล 155 เมกกะบิตต่อวินาที เมื่อทำงานที่สัญญาณนาฬิกา 25 เมกกะเฮิรตซ์ (A.Curiger *et al.*, 1993:15.5.1-15.5.4)

1.2.3 ค.ศ. 1994 R.Zimmerman และคณะได้เสนองานวิจัยเรื่อง "A 177 Mb/s VLSI Implementation of the International Data Encryption Algorithm" โดยใช้เทคโนโลยี CMN12 ในการออกแบบ ประเมินพื้นที่ได้ 107.8 ตารางมิลลิเมตร บรรจุทรานซิสเตอร์จำนวน 251,000 ตัว ที่มีอัตราการส่งผ่านข้อมูล 177.8 เมกกะบิตต่อวินาที เมื่อทำงานที่สัญญาณนาฬิกา 25 เมกกะเฮิรตซ์ (R.Zimmermann *et al.*, 1994: 303-307)

1.2.4 ค.ศ. 1995 S. Wolter และคณะได้เสนองานวิจัยเรื่อง "On the VLSI Implementation of the International Data encryption Algorithm IDEA" โดยใช้สถาปัตยกรรมแบบอนุกรมท่อ (pipeline) 10 ชุด และใช้ตัวดำเนินการวิธีปรับปรุงมอดุโลผลคูณ 4 ชุด ตัวดำเนินการคูณใช้วิธีการของ Booth และแผนผังต้นไม้ของ Wallace (Booth multipler and Wallace tree) ยังผลให้ได้วงจรรวมสำหรับการเข้ารหัสที่มีอัตราการส่งผ่านข้อมูล 355 เมกกะบิตต่อวินาที เมื่อทำงานที่สัญญาณนาฬิกา 50 เมกกะเฮิรตซ์ (S.wolter, 1995:397-400)

1.2.5 ค.ศ. 1996 E. Caspi และคณะได้เสนองานวิจัยเรื่อง "IDEA as a benchmark for reconfigurable computing" โดยบรรจุเฉพาะตัวดำเนินการวิธีปรับปรุงมอดุโลผลคูณลงในวงจรรวมแบบต่างๆแล้วเปรียบเทียบสมรรถนะจากสัดส่วนอัตราการส่งผ่านข้อมูลต่อสัญญาณนาฬิกา และสัดส่วนอัตราการส่งผ่านข้อมูลต่อพื้นที่ได้ผลดังตารางที่ 1.2(E.Caspi *et al.*, 1996:1-13)

1.2.6 ค.ศ. 1999 E. Mosanya และคณะได้เสนองานวิจัยเรื่อง "Cryptobooster: A Reconfigurable and Modular Cryptographic Coprocessor" ซึ่งรายงานการออกแบบวงจรสำหรับการเข้ารหัสแบบไอดีอีเอด้วยเทคโนโลยีวงจรรวมที่สามารถโปรแกรมซ้ำได้จำพวกเอฟพีจีเอ (Field Programmable Gate Array - FPGA) เพื่อใช้เป็นอุปกรณ์ประมวลผลร่วม (coprocessor) ผลการออกแบบสามารถส่งผ่านข้อมูลด้วยอัตรา 200 เมกกะบิตต่อวินาทีต่อการประมวลผล 1 รอบการเข้ารหัส และเมื่อใช้สถาปัตยกรรมอนุกรมท่อ 59 ชุดวางการเข้ารหัสทั้ง 9 รอบ จะได้อัตราส่งผ่านข้อมูลสูงกว่า 1500 เมกกะบิตต่อวินาที ผลของงานวิจัยนี้ได้จดลิขสิทธิและสิทธิบัตรร่วมกันระหว่าง สถาบันเทคโนโลยีแห่งรัฐบาลกลางสวิสเซอร์แลนด์ (International patent PCT/CH91/00117) และ Lausanne and Lightning Instrumentation ประเทศสหรัฐอเมริกา (U.S. Patent #5,254703) นอกจากนี้คณะผู้วิจัยได้ออกแบบทดลองออกแบบวงจรนี้

ด้วยเทคโนโลยีไมโครอิเล็กทรอนิกส์ขนาด 0.25 ไมครอน (micron) ขณะอยู่ในระหว่างดำเนินทดสอบขั้นสุดท้าย (E. Mosanya *et al.*,1999 :246-256)

ตารางที่ 1.2 การเปรียบเทียบสมรรถนะวงจรรวมตามรายงานวิจัยของ E. Caspi และคณะ

วงจรรวม	อัตราการส่งผ่านข้อมูล/สัญญาณนาฬิกา (เมกกะบิต/เมกกะเฮิรตซ์)	อัตราการส่งผ่านข้อมูล/พื้นที่ (เมกกะบิต/ตารางมิลลิเมตร)
XC4005	0.065 (ที่ 7.3เมกกะเฮิรตซ์)	0.0020
GRAP*	0.561 (ที่ 140 เมกกะเฮิรตซ์)	0.0570
MC56166**	0.021 (ที่ 60 เมกกะเฮิรตซ์)	-
UltraSparce I	0.093 (ที่ 167 เมกกะเฮิรตซ์)	0.0031
sT0***	0.140 (ที่ 40 เมกกะเฮิรตซ์)	0.0050

*,*** *GRAP และ T0 เป็นผลงานวิจัยของห้องปฏิบัติการ BRASS ของคณะผู้วิจัย*

** *MC56166 เป็นวงจรรวมประมวลผลสัญญาณดิจิตอลของบริษัทโมโตโลรา ซึ่งไม่มีรายละเอียดของพื้นที่ที่ใช้งานจริงหลังจากบรรจุวงจรตัวดำเนินการลงไป*

(ที่มา : E.Caspi *et al.*, 1996)

สำหรับประเทศไทย ยังไม่มีรายงานการประชุมหรืองานวิจัยเกี่ยวกับวงจรรวมสำหรับการเข้ารหัสไอดีอีเอ คงมีแต่งานวิจัยที่ใกล้เคียงกล่าวคือการออกแบบวงจรอุปกรณ์สำหรับการประมวลผลรหัสลับแบบดีอีเอสเท่านั้น

1.3 วัตถุประสงค์

1.3.1 เพื่อศึกษา ออกแบบและทดสอบโปรแกรมรหัสลับไอดีอีเอให้เกิดผล อันจะนำไปสู่การอ้างอิงและสนับสนุนผลการทดสอบวงจรรวมต้นแบบ

1.3.2 เพื่อออกแบบและสร้างวงจรรวมต้นแบบสำหรับการเข้าและถอดรหัสข้อมูล 64 บิต ด้วยรหัสกุญแจ 128 บิต แบบ 9 รอบ ตามข้อกำหนดของรหัสลับแบบไอดีอีเอ

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ความรู้ความเข้าใจเกี่ยวกับรหัสลับไอดีอีเอ

1.4.2 วงจรรวมต้นแบบสำหรับรหัสลับไอดีอีเอซึ่งยังไม่มีการออกแบบในประเทศไทย

1.5 ขอบเขตของการวิจัย

1.5.1 ออกแบบและสร้างวงจรรวมต้นแบบเพื่อการประมวลผลรหัสลับไอดีอีเอ โดยใช้เอฟพีจีเอ ของบริษัท Xilnx ในตระกูล XC4000

1.6 ขั้นตอนและวิธีดำเนินงานวิจัย

1.6.1 ศึกษาข้อมูลเกี่ยวกับรหัสลับไอดีอีเอ

1.6.2 เขียนโปรแกรมทดสอบการเข้ารหัส ถอดรหัสไอดีอีเอ

1.6.3 ศึกษาและออกแบบวงจรสำหรับการเข้าและถอดรหัสไอดีอีเอ

1.6.4 สร้างวงจรรวมต้นแบบและทดสอบรหัสลับไอดีอีเอโดยใช้เอฟพีจีเอตระกูล XC4000

1.6.5 สรุปผลขั้นสุดท้าย

1.6.6 ดำเนินการสอบวิทยานิพนธ์

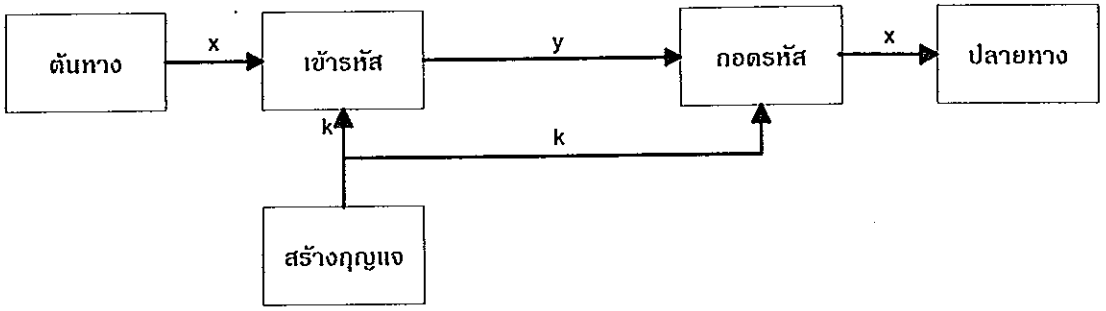
1.5.7 แก้ไขวิทยานิพนธ์ฉบับสมบูรณ์

บทที่ 2

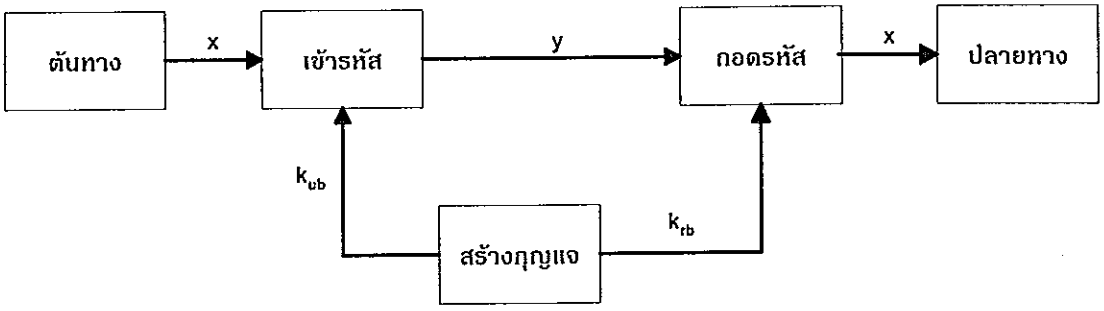
ขั้นตอนและวิธีการของรหัสลับไอดีอีเอ

2.1 บทนำ

การเข้ารหัสจะจำแนกเป็นสองประเภทได้แก่ ประเภทแรกการเข้ารหัสแบบสมมาตร (symmetry) หรือการเข้ารหัสกุญแจเดียว (single-key) บางทฤษฎีเรียกเป็นการเข้ารหัสกุญแจส่วนบุคคล (private-key)) แสดงในภาพประกอบที่ 2.1 และประเภทที่สองการเข้ารหัสแบบอสมมาตร (asymmetry) หรือการเข้ารหัสกุญแจสาธารณะ (public-key) แสดงในภาพประกอบที่ 2.2



ภาพประกอบ 2.1 แผนผังการเข้ารหัสแบบสมมาตร



ภาพประกอบ 2.2 แผนผังการเข้ารหัสแบบอสมมาตร

รหัสลับแบบสมมาตรนี้จะจำแนกได้เป็นการเข้ารหัสแบบสายน้ำ (stream cipher) ที่นิยมใช้ได้แก่ RC4 , SEAL , A5 , ISAAC , SOBER , WAKE , Sapphire และ PIKE การเข้ารหัสแบบสายน้ำนี้มีข้อด้อยคือ ถูกถอดรหัสได้ง่าย แต่มีข้อเด่นคือสามารถใช้กับระบบประมวลผลเวลาจริง (real-time processing system) ได้ สำหรับการเข้ารหัสแบบบล็อก

(block cipher) ที่นิยมใช้ได้แก่ AES , Blowfish , CAST , DES , FEAL , GOST , ICE , IDEA , Lucifer , MISTY , RC5 , RC2 , REDOC, Safer , Shark , Skipjack และ Square ซึ่งมีความปลอดภัยสูงกว่าการเข้ารหัสแบบสายน้ำ แต่ยังคงประสิทธิภาพเสมือนหนึ่งประมวลผลเวลาจริง สามารถประยุกต์เป็นวงจรอิเล็กทรอนิกส์ได้

สำหรับรหัสลับแบบอสมมาตรนั้น ที่นิยมใช้ได้แก่ Diffie-Hellman , Cramer-Shoup , RSA , XTR , LUC , McEliece , NTRU , RPK , Knapsacks , Chor-Rivest , Elliptic Curves และ Lattice Reduction รหัสลับแบบนี้มีการคำนวณที่ซับซ้อนและใช้เวลาในการประมวลผลมากกว่ารหัสลับแบบสมมาตร แต่กระนั้นก็มีความปลอดภัยในการซ่อนพลางข้อมูลได้เป็นอย่างดี จึงมีการประยุกต์ใช้ในรูปแบบของโปรแกรมคอมพิวเตอร์เป็นส่วนใหญ่ (B.Schneier,1996 :233-239)

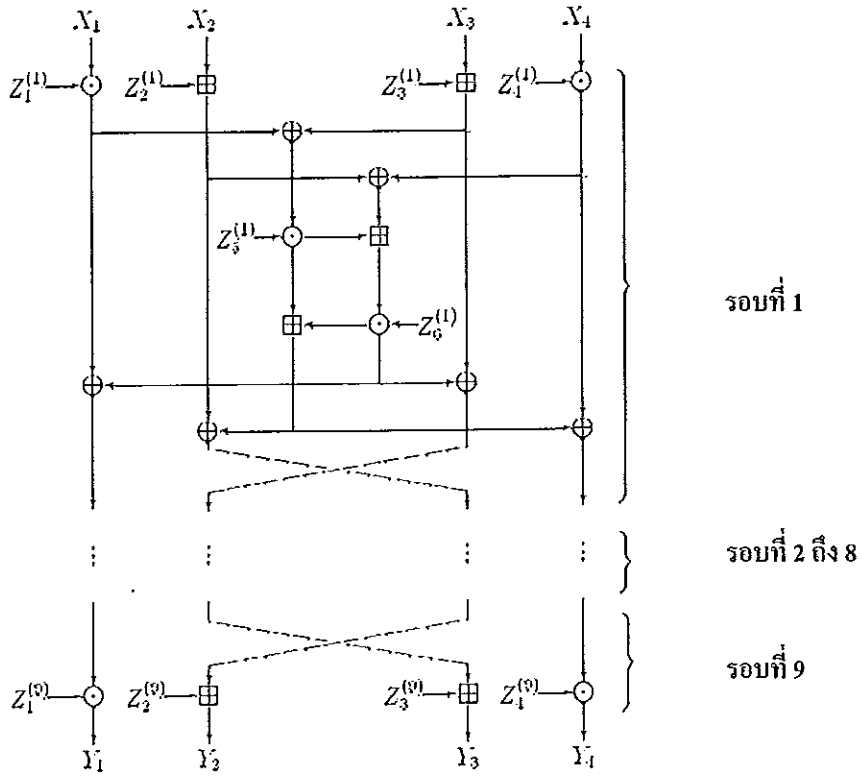
2.2 หลักการของรหัสลับไอดีอีเอ

ภาพประกอบที่ 2.3 แสดงแผนผังของรหัสลับไอดีอีเอรับข้อมูลก่อนการเข้ารหัส (plaintext) ขนาด 64 บิตและรหัสกุญแจขนาด 128 บิต และได้ผลลัพธ์เป็นรหัสลับขนาด 64 บิต แผนผังดังกล่าวจะถูกใช้ทั้งการเข้ารหัสและถอดรหัส มีการกำหนดสัญลักษณ์และสัญกรต่างๆดังนี้ (X.Lai and J.Massey , 1990)

ข้อมูลก่อนเข้ารหัสขนาด 64 บิตจะถูกแยกเป็นกลุ่มย่อยๆตามลำดับของเลขนัยสำคัญเป็นกลุ่มๆละ 16 บิต เป็น X_1, X_2, X_3 และ X_4 ตามลำดับ และรหัสกุญแจขนาด 128 บิตจะถูกแบ่งเป็น 8 กลุ่มย่อยได้แก่ $Z_1^{(1)}, Z_2^{(1)}, Z_3^{(1)}, Z_4^{(1)}, Z_5^{(1)}, Z_6^{(1)}, Z_7^{(1)}$ และ $Z_8^{(1)}$ ตามลำดับ ในการเข้ารหัสจะทำ 9 รอบซึ่งต้องการรหัสกุญแจ 52 ชุด ดังนั้นกุญแจย่อยที่เหลือจะได้จากการเลื่อนแบบวงรอบไปทางซ้าย (cyclic shifted left) 25 ตำแหน่งของกุญแจย่อยในรอบก่อนหน้านี้ ดังแสดงไว้ในตารางที่ 2.1

ตารางที่ 2.1 การกระจายรหัสกุญแจสำหรับการเข้ารหัสไอดีอีเอ

รอบที่	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6
1	0-15	16-31	32-47	48-63	64-79	80-95
2	96-111	112-127	25-40	41-56	57-72	73-88
3	89-104	105-120	121-8	9-24	50-65	66-81
4	82-97	98-113	114-1	2-17	18-33	34-49
5	75-90	91-106	107-122	123-10	11-26	27-42
6	43-58	59-74	100-115	116-3	4-19	20-35
7	36-51	52-67	68-83	84-99	125-12	13-28
8	29-44	45-60	61-76	77-92	93-108	109-124
9	22-37	38-53	54-69	70-85	-	-



ภาพประกอบที่ 2.3 แผนผังของรหัสลับไอดีอีเอ

(ที่มา :X.Lai et al,1990:393)

โดยที่

$\oplus$ เป็นตัวดำเนินการเอ็กซ์คลูซีฟออร์แบบบิต (bitwise Ex-OR)

$\boxplus$ เป็นตัวดำเนินการมอดุโลของผลบวก (addition modulo) 2^{16} หรือ (a+b) mod (2^{16}) หรือ a $\boxplus$ b

$\odot$ เป็นตัวดำเนินการวิธีปรับปรุงมอดุโลของผลคูณ(modified multiplication modulo) หรือ (a•b) mod ($2^{16}+1$) หรือ a $\odot$ b

X_i เป็น 16 บิตของข่าวสาร (plaintext)

Y_i เป็น 16 บิตของรหัสลับ (cipher)

$Z_i^{(r)}$ เป็น 16 บิตรหัสกุญแจรอบที่ r ตัวที่ I

2.3 ความรู้เบื้องต้นเกี่ยวทฤษฎีจำนวนและพีชคณิตนามธรรม (K.H.Rosen,1992)

พื้นฐานที่สำคัญเกี่ยวกับรหัสลับและการเข้ารหัสจะใช้เซตของหน่วยในวงจำนวนจริงที่เป็นกลุ่มภายใต้ตัวดำเนินการใดๆ โดยที่สมาชิกของเซตไม่เป็นกลุ่มวัฏจักร (cyclic group) ทั้งนี้เพื่อให้ได้ผลการส่งผ่านระหว่างฟังก์ชันเป็นค่าสมสัณฐาน (isomophic) กันซึ่งจะมีความเป็นหนึ่งเดียวภายใต้เซตนั้นๆ ถ้ากำหนดสัญกร $\mathbb{Z}_n$ เป็นเซตของจำนวนเต็มบวกใดๆที่มีสมาชิกเป็น {0,1,2,3,...} โดยสมาชิกแต่ละตัวจะเป็นเศษที่เหลือจากการหารด้วย n จะมีนิยามต่างๆได้แก่

บทนิยามที่ 1 ถ้า $a \neq 0$ และ $b=ac$ แล้ว จะเรียกว่า b หารด้วย a ลงตัว และเขียนแทนด้วย a|b โดยจะเรียก a ว่าเป็นตัวหาร (divisor) ของ b และเรียก b ว่าเป็นพหุคูณ (multiple) ของ a ซึ่งมีคุณสมบัติดังนี้

1. 1|a ทุกๆ a หรือ a|a เป็นจริง
2. ถ้า a|b และ b|c แล้ว a|c เป็นจริง
3. ถ้า a|b และ a|c แล้ว a|(bx+cy) ทุกๆจำนวนเต็ม x,y เป็นจริง
4. ถ้า a|b และ b|a แล้ว $a=\pm b$ เป็นจริง

บทนิยามที่ 2 ถ้า $p \in \mathbb{Z}_n$ และ $p \neq 0$ แล้ว p จะเป็นจำนวนเฉพาะ (prime) ก็ต่อเมื่อ $\pm 1|p$ และ $\pm p|p$ เท่านั้น

บทนิยามที่ 3 ถ้า a และ $b \neq 0$ แล้ว จะเรียก c ซึ่งเป็นจำนวนเต็มที่มากที่สุดเพียงจำนวนเดียวที่ c|a และ c|b ว่า เป็นจำนวนหารร่วมมาก (Greatest Common Divisor-GCD) เขียนแทนด้วย $c = GCD(a,b)$ ซึ่งมีคุณสมบัติดังนี้

1. $c>0$ เป็นจริง
2. ถ้า $d \in Z_n$ ซึ่ง d|a และ d|b จะได้ว่า d|c เป็นจริง
3. ถ้า $GCD(a,b)=1$ แล้ว a และ b จะเป็นจำนวนเฉพาะต่อกัน (relatively prime)

บทนิยามที่ 4 ถ้า $n,q \in \mathbb{Z}_n$ จะเรียก และ $q \leq n$ เมื่อ q เป็นจำนวนเฉพาะต่อกันกับ n ว่าเป็นค่าของฟังก์ชันออยเลอร์ฟี (Euler phi-function) เขียนแทนด้วย $\varphi(n) = q$

บทนิยามที่ 5 ถ้า $m \in \mathbb{Z}_n$ และ m|(a-b) แล้ว จะเรียกว่า a สมภาคกับ b ภายใต้มอดุโล n และเขียนแทนด้วย $a \equiv b \pmod{m}$

บทนิยามที่ 6 ถ้า $a,m \in \mathbb{Z}_n$ และเป็นจำนวนเฉพาะต่อกัน จะเรียกจำนวนเต็มบวก x ที่น้อยที่สุด ซึ่งเป็นขนาดของ a ภายใต้มอดุโล m ว่า $a^x=1(\text{mod } m)$ และเขียนแทนด้วย $ORD_m(a)$

บทนิยามที่ 7 ถ้า r และ n เป็นจำนวนเฉพาะต่อกันโดยที่ $n>0$ และ $ORD_n(r) = \varphi(n)$ แล้ว จะเรียก r ว่าเป็น รากปฐมฐานมอดุโล n

บทนิยามที่ 8 ถ้า A เป็นเซตไม่ว่างใดๆ และ * เป็นฟังก์ชันจาก AxA ไป A ($*: AxA \rightarrow A$) จะเรียก * ว่าเป็นตัวดำเนินการทวิภาค (binary operation) บน A และมีคุณสมบัติดังนี้

1. * จะถูกนิยามสำหรับทุก $(a,b) \subset AxA$ อย่างแจ่มชัด
2. $a,b \in A$ และ $a * b \in A$

บทนิยามที่ 9 ถ้า $\mathbb{G}$ เป็นเซตไม่ว่างและ * เป็นตัวดำเนินการทวิภาคบน $\mathbb{G}$ จะเขียนแทนกลุ่มของ $\mathbb{G}$ ได้ว่า $(\mathbb{G}, *)$ ซึ่งมีคุณสมบัติดังนี้

- * มีคุณสมบัติการเปลี่ยนกลุ่ม
- มีสมาชิก e ใน $\mathbb{G}$ ซึ่ง $a*e = e*a = a$ ทุกๆค่าของ a ที่เป็นสมาชิกของ $\mathbb{G}$ เรียก e ว่า สมาชิกเอกลักษณ์ของ $\mathbb{G}$ ภายใต้ตัวดำเนินการ * เขียนแทนสมาชิกเอกลักษณ์นี้ว่า e_G
- มีสมาชิก b ใน $\mathbb{G}$ ซึ่ง $a*b = b*a = e_G$ แล้วจะเรียก b ว่าเป็นตัวผกผันของ a เขียนแทนตัวผกผันนี้ของ a ด้วย a^{-1}

บทนิยามที่ 10 ถ้าสมาชิกทุกตัวของ $\mathbb{G}$ ใน $(\mathbb{G}, *)$ มีคุณสมบัติการสลับที่จะเรียก $(\mathbb{G}, *)$ ว่า กลุ่มอาบีเลียน (Abelian group)

บทนิยามที่ 11 ถ้า a สมาชิกของ $\mathbb{G}$ ใน $(\mathbb{G}, *)$ ซึ่ง $\mathbb{G} = \{a^n | n \in \mathbb{Z}_n\}$ แล้วจะเรียก $(\mathbb{G}, *)$ ว่ากลุ่มวัฏจักร (cyclic group) และเรียก a ว่า ตัวก่อกำเนิด (generator) สำหรับ $\mathbb{G}$ เขียนแทนด้วย <a>

บทนิยามที่ 12 ถ้า $\mathbb{S}$ เป็นเซตไม่ว่างและ * เป็นตัวดำเนินการบน $\mathbb{S}$ โดยที่ $(a,b) \subset \mathbb{S}$ และมีคุณสมบัติไม่ครบตามนิยามการเป็นกลุ่ม จะเรียก $(\mathbb{S}, *)$ ว่ากึ่งกลุ่ม (quasigroup) ของ $\mathbb{S}$ ก็ต่อเมื่อ $a*x=b$ และ $y*a=b$ มีเพียงผลเฉลยเพียงหนึ่งเดียวเมื่อ $x,y \in \mathbb{Z}_n$

บทนิยามที่ 13 ถ้า $(\mathbb{S}, *)$ เป็นกึ่งกลุ่ม (quasigroup) แต่มีคุณสมบัติการเปลี่ยนกลุ่มได้ จะเรียก $(\mathbb{S}, *)$ ว่ากลุ่ม

บทนิยามที่ 14 ให้ θ เป็นการส่งผ่านค่าจากกลุ่ม $(\mathbb{S}_1, *_1)$ และ $(\mathbb{S}_2, *_2)$ จะเรียก θ ว่า ฟังก์ชันสาทิสสัณฐาน (homomorphism) และกล่าวว่ากลุ่ม $\mathbb{S}_1$ และ $\mathbb{S}_2$ สมสัณฐาน (isomorphic) กัน เขียนแทนด้วย $\mathbb{S}_1 \approx \mathbb{S}_2$

บทนิยามที่ 15 ถ้า $(\mathbb{S}_1, *_1)$ และ $(\mathbb{S}_2, *_2)$ เป็นกึ่งกลุ่มและมีส่งผ่านค่าหนึ่งต่อหนึ่งแบบทั่วถึง (bijective) ของฟังก์ชัน $\theta,\varnothing,\psi : \mathbb{S}_1 \rightarrow \mathbb{S}_2$ แล้ว จะเรียกว่าเป็นกลุ่มสมนัย (isotopic group) โดยที่ $\theta(x) *_2 \varnothing(y)=\psi(x*_2y)$ ทุกๆค่าของ x,y ใน S_1และเรียกฟังก์ชัน $\theta,\varnothing,\psi$ ว่าเป็นสมาชิก สมนัย (isotopism) ของ $(\mathbb{S}_1, *_1)$ บน $(\mathbb{S}_2, *_2)$

บทนิยามที่ 16 ถ้ากึ่งกลุ่ม 2 กลุ่มมีสมาชิกสมนัยกันและเป็นกลุ่มสมนัยซึ่งกันและกันแล้ว กึ่งกลุ่มทั้งสองจะเป็นกลุ่มสมสัณฐาน แต่กลุ่มสมสัณฐานไม่จำเป็นต้องเป็นกลุ่มสมนัยซึ่งกันและกัน

บทนิยามที่ 17 ถ้า $\mathbb{R} \subset \mathbb{Z}_n$ และเป็นเซตไม่ว่าง โดย + เป็นตัวดำเนินการทวิภาคของการบวกภายในเซต $\mathbb{R}$ และ $\bullet$ เป็นตัวดำเนินการทวิภาคของการคูณภายในเซต $\mathbb{R}$ แล้วจะเรียก $(\mathbb{R},+, \bullet)$ ว่าเป็นวงโดยมีกลุ่มเอกฐานของวง R ที่เป็นสมาชิกของ $\mathbb{Z}_n$ เป็น $\mathbb{Z}_n^*$ ก็ต่อเมื่อมีคุณสมบัติดังนี้

- $(\mathbb{R},+)$ เป็นกลุ่มอาบีเลียน
- สำหรับทุกๆ $a,b,c \in \mathbb{R}$ จะสอดคล้องกับกฎการแจกแจงไปทางซ้ายและกฎการแจกแจงไปทางขวา

บทนิยามที่ 18 ถ้า $\mathbb{R}$ เป็นวงที่มีสมาชิกเอกลักษณ์ภายใต้การคูณจะเรียกวง $\mathbb{R}$ ว่าวงซึ่งมีหนึ่ง

บทนิยามที่ 19 ถ้า $\mathbb{R}$ เป็นวงซึ่งมีหนึ่งและมีสมาชิกที่เป็นตัวผกผันการคูณจะเรียกตัวผกผันการคูณนั้นว่าหน่วย (unit) ของวง $\mathbb{R}$

2.4 กลไกการเข้ารหัสไอดีอีเอ (D.R.Stinson.1995)

กลไกที่สำคัญของการเข้ารหัสไอดีอีเอนี้เกิดจากการรวมตัวดำเนินการที่ให้ผลของพีชคณิตเชิงกลุ่มไม่ซ้ำค่าเมื่อองค์ประกอบของกลุ่มเป็นจำนวนเดียวกัน ตัวดำเนินการดังกล่าวได้แก่ ตัวดำเนินการแบบบิตของฟังก์ชันเอ็กคลูซีฟออร์ ตัวดำเนินการมอดุโลของผลบวก และตัวดำเนินการวิธีปรับปรุงมอดุโลของผลคูณ กำหนดสัญกรต่างๆเพื่ออธิบายความสัมพันธ์เชิงจำนวนได้ดังนี้

$\mathbb{Z}_{2^{16}}$	แทนวงของมอดุโล 2^{16} ที่เป็นจำนวนจริงบวก
$(\mathbb{Z}^*_{2^{16}+1}, *)$	แทนกลุ่มการคูณของสมาชิกที่ไม่เป็นศูนย์ของสนาม $\mathbb{Z}_{2^{16}+1}$
$(\mathbb{Z}_{2^{16}}, +)$	แทนกลุ่มการบวกของวงมอดุโล 2^{16}
$(\mathbb{F}_2^{16}, \oplus)$	แทนกลุ่มของ แบบ 16 บิตภายใต้มอดุโล 2 ที่เกิดจากตัวดำเนินการ Ex-OR
$d(i)$	แทนการส่งผ่านค่าโดยตรง (direct mapping) จาก $Z^*_{2^{16}+1}$ ไปบน $\mathbb{Z}_{2^{16}}$ โดยที่ $d(i) = i$ เมื่อ $i \neq 2^{16}$ และ $d(2^{16}) = 0$

อาศัยบทนิยามที่กล่าวมาตอนต้น จะถูกกำหนดเป็นทฤษฎีบทประกอบจะได้ว่า

ทฤษฎีบทประกอบที่ 1 กึ่งกลุ่มของ $(\mathbb{F}_2^{16}, \oplus)$ และ $(\mathbb{Z}_{2^{16}}, +)$ จะไม่มีสมาชิกที่สมนัยกันและไม่เป็นกลุ่มสมนัยกันเนื่องจาก $(\mathbb{Z}_{2^{16}}, +)$ เป็นกลุ่มวัฏจักรแต่ $(\mathbb{F}_2^{16}, \oplus)$ ไม่เป็นกลุ่มวัฏจักร

ทฤษฎีบทประกอบที่ 2 กึ่งกลุ่มของ $(\mathbb{F}_2^{16}, \oplus)$ และ $(\mathbb{Z}^*_{2^{16}+1}, *)$ จะไม่มีสมาชิกที่สมนัยกันและไม่เป็นกลุ่มสมนัยกันเนื่องจาก $(\mathbb{Z}^*_{2^{16}+1}, *)$ และ $(\mathbb{Z}_{2^{16}}, +)$ เป็นกลุ่มวัฏจักรที่สมนัยกัน

ทฤษฎีบทประกอบที่ 3 ถ้า a และ b เป็นสมาชิกภายใต้วง $\mathbb{Z}_{2^{16}}$ ที่เกิดจากการส่งผ่านค่าโดยตรงของ $d(i)$ แล้ว $(a * b) \bmod (2^{16}+1) = d[(d^{-1}(a) * d^{-1}(b)) \bmod (2^{16}+1)]$ เมื่อ $d^{-1}(i)$ เป็นฟังก์ชันผกผันการส่งผ่านค่าของ $d(i)$ ภายใต้กลุ่มวัฏจักร

ทฤษฎีบทประกอบที่ 4 ถ้า a และ b เป็นสมาชิกภายใต้สนาม $\mathbb{Z}_{2^{16}+1}$ ที่เกิดจากการส่งผ่านค่าโดยตรงของ *d(i)* แล้ว $(a + b) \bmod (2^{16}) = d[(d^{-1}(a) + d^{-1}(b)) \bmod 2^{16}]$ เมื่อสมาชิกของสนามไม่เป็นศูนย์ แต่ *d(i)* จะมีค่าเป็นศูนย์ก็ต่อเมื่อสนามชิกของสนาม $\mathbb{Z}_{2^{16}+1}$ เป็นศูนย์ โดยที่ $d^{-1}(i)$ เป็นฟังก์ชันผกผันการส่งผ่านค่าของ *d(i)* ภายใต้กลุ่มวัฏจักร

ทฤษฎีบทประกอบที่ 5 จะไม่มีการกระทำระหว่างตัวดำเนินการคู่ใดที่ยังผลให้กฎการกระจายเป็นจริง

ทฤษฎีบทประกอบที่ 6 จะไม่มีการกระทำระหว่างตัวดำเนินการคู่ใดที่ยังผลให้กฎการสลับที่ของตัวดำเนินการเป็นจริง

จากบทนิยามและทฤษฎีบทประกอบที่ได้กล่าวมาแล้ว จะเห็นได้ว่าการประมวลผลในแต่ละตัวดำเนินการจะมีความไม่เป็นเชิงเส้น และถูกทำให้ซับซ้อนขึ้นด้วยการประมวลผลอีก 9 รอบ

2.5 กลไกการถอดรหัส

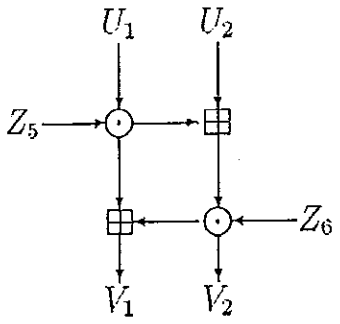
ข้อดีของรหัสลับไอดีอีเอคือการใช้กระบวนการเข้ารหัสและถอดรหัสที่เหมือนกัน แตกต่างกันเฉพาะข้อมูลเข้าและการแปลงรหัสกุญแจเท่านั้น เรียกกลไกนี้ว่า *"การแปลงภาวะคล้าย"* (similarity transformation)

ในการพิจารณากลไกการถอดรหัสนั้นจะอ้างอิงขั้นตอนเพียงรอบที่ 1 ของการเข้ารหัสตามภาพประกอบที่ 2.3 จะสังเกตได้ว่า ฟังก์ชันที่ได้ในแต่ละรอบจะเป็นกลุ่มของรหัสลับ (cipher group) ที่ประกอบขึ้นจากผลรวมของอาวัตนาการรหัสลับ (involution cipher) และอาวัตของการเรียงสับเปลี่ยน (involutary permutation) ซึ่งเป็นอัตสัณฐาน (automorphism) ของกลุ่ม $\left(\mathbb{F}_2^{64}, \otimes\right)$ และสามารถเขียนความสัมพันธ์ของข่าวสารและกุญแจได้ดังสมการที่ 2.1

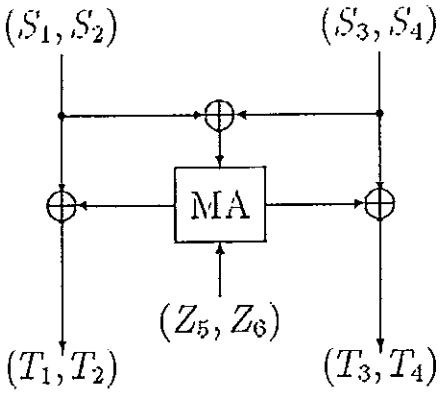
$$X \otimes Z_A = (X_1 \odot Z_1, X_2 \boxplus Z_2, X_3 \boxplus Z_3, X_4 \odot Z_4) \qquad 2.1$$

และถ้ากำหนดให้ $P_I(X)$ เป็นวิธีการเรียงสับเปลี่ยนบน X ที่ทำให้เกิดการไขว้กันระหว่างผลลัพธ์ที่เกิดจาก X_2 และผลลัพธ์ที่เกิดจาก X_3 ของ $X = (X_1, X_2, X_3, X_4)$

ในขั้นตอนสุดท้ายของแต่ละรอบแล้ว จะพบว่า P_I คืออาวัตนาการที่ยังผลให้ $P_I(X \otimes Z_A) = P_I(X) \otimes P_I(Z_A)$ หมายความถึง P_I จะทำให้เกิดความเป็นอัตสัณฐานอย่างสมบูรณ์สมบูรณ์ของกลุ่ม $(\mathbb{F}_2^{64}, \otimes)$



ภาพประกอบที่ 2.4 กราฟการคณนาโครงสร้างของคูณบวก (Multiplication addition – MA)



ภาพประกอบที่ 2.5 กราฟการคณนาของอาวัตการในฟังก์ชัน $I_n(\bullet, Z_B)$

ในภาพประกอบที่ 2.4 และ 2.5 จะแสดงความสัมพันธ์ของข้อมูลเข้าขนาด 64 บิต (S_1, S_2, S_3, S_4) และข้อมูลออกขนาด 64 บิต (T_1, T_2, T_3, T_4) ตามความสัมพันธ์ของฟังก์ชัน $I_n(\bullet, Z_B)$ โดยมีตัวควบคุมเป็นกุญแจ 32 บิต $Z_B = (Z_5, Z_6)$ ที่เป็นอาวัตนาการ ถ้าตรึง $Z_B = (Z_5, Z_6)$ ไว้แล้วจะได้ความสัมพันธ์ของค่าผกผันในฟังก์ชัน $I_n(\bullet, Z_B)$ เป็นค่าเดิมและคุณสมบัติผกผันในตัว (self-inverse) นี้จะเป็นข้อเท็จจริงที่สอดคล้องตามเงื่อนไขที่ว่า "*ผลของตัวดำเนินการเอ็กซ์คลูซีฟออร์ระหว่าง* (S_1, S_2) *และ* (S_3, S_4) *จะต้องเท่ากับผลของตัวดำเนินการ Ex-OR ระหว่าง* (T_1, T_2) *และ* (T_3, T_4)" จึงทำให้เกิดบทนิยามแบบอุปนัย (inductive definition) สำหรับกุญแจในการถอดรหัสสำหรับแต่ละรอบ

ในการหารหัสกุญแจตัวอื่นๆ $Z_A = (Z_1, Z_2, Z_3, Z_4)$ จะใช้คุณสมบัติของกลุ่มวัฏจักรของ $(\mathbb{Z}_{2^{16}}, +)$ และ $(\mathbb{Z}^*_{2^{16}+1}, *)$ โดยการหาค่าผกผันของตัวดำเนินการ ดังนั้นกุญแจสำหรับการถอดรหัส Z'_A จึงสามารถเขียนเป็นสมการที่ 2.2 ได้ว่า

$$Z'_A = (Z_1^{-1}, -Z_2, -Z_3, Z_4^{-1}) \qquad 2.2$$

การหาค่า $-Z$ ซึ่งเป็นค่าผกผันมอดุโลของผลบวกจะทำได้โดยหาผลต่างระหว่างค่ามากที่สุดของกลุ่ม $(\mathbb{Z}_{2^{16}}, +)$ และค่า Z ตามความสัมพันธ์ในสมการที่ 2.3 ที่ว่า

$$-Z = 2^{16} + \overline{Z} \text{ หรือ } -Z + \overline{Z} = 2^{16} \qquad 2.3$$

เมื่อ $\overline{Z}$ เป็นส่วนเติมเต็มเลขฐานสอง (2's complementary) ของ Z

เมื่อเติมตัวดำเนินการมอดุโลด้วย 2^{16} ทั้งสองข้างของสมการ 2.3 จะได้ผลดังสมการที่ 2.4

$$0 \equiv \left(-Z + \overline{Z}\right) \bmod 2^{16} = 2^{16} \bmod 2^{16} \qquad 2.4$$

ดังนั้น $-Z$ คือค่าที่ได้ผลของตัวดำเนินการ $-Z \boxplus Z = 0$

สำหรับการหาค่า Z^{-1} ทำได้โดยการใช้บทนิยามหารร่วมมากโดยเขียนอยู่ในรูปสมการไดโอแฟนไทน์ (Diophantine equatioin) โดยที่ ถ้า $a,b \in \mathbb{Z}_n$ และ $a \le b$ แล้ว จะได้ความสัมพันธ์ตามสมการที่ 2.5 เป็น

$$\mathrm{GCD}(a, b) = \mathrm{GCD}(b, a \bmod b) = ax + by \qquad 2.5$$

เมื่อให้ a เป็นรหัสกุญแจของตัวดำเนินการวิธีปรับปรุงมอดุโลของผลคูณและ b เป็นตัวเลขแฟร์มาต์ (Fermet number) $2^{16}+1$ ที่มีอันดับ 4 ($F_n=2^{2^n}+1$) จะได้ความสัมพันธ์ดังแสดงในสมการที่ 2.6 คือ

$$\mathrm{GCD}(Z, 2^{16}+1) = Z \cdot x + (2^{16}+1) \cdot y \qquad 2.6$$

เนื่องจากตัวเลขแฟร์มาต์ $2^{16}+1$ เป็นค่าจำนวนเฉพาะตามทฤษฎีของออยเลอร์-ฟี ทำให้ข้อมูลขนาด 16 บิตทุกตัวจะเป็นจำนวนเฉพาะต่อกันกับตัวเลขแฟร์มาต์อันดับ 4 เสมอ และตัวเลขแฟร์มาต์ดังกล่าวยังเป็นจำนวนเฉพาะสมบูรณ์อีกด้วย ยังผลของสมการ 2.6 จะมีค่าสมนัยกับ 1 และเขียนความสัมพันธ์ต่อไปดังสมการที่ 2.7 ได้ว่า

$$(2^{16}+1)\cdot y \equiv Z\cdot x+1 \text{ หรือ } 1 \equiv y(2^{16}+1)+\overline{Z\cdot x} \qquad 2.7$$

เมื่อ $\overline{Z\cdot x}$ เป็นส่วนเติมเต็มเลขฐานสองของ $Z\cdot x$ ภายใต้กลุ่ม $\left(\mathbb{Z}^{*}_{2^{16}+1},*\right)$

เมื่อเติมตัวดำเนินการมอดุโลด้วย $2^{16}+1$ ทั้งสองข้างของสมการ 2.7 จะได้ความสัมพันธ์ตามสมการที่ 2.8 และ 2.9 เป็น

$$1 \equiv 1 \bmod (2^{16}+1) \qquad 2.8$$

หรือ
$$\begin{aligned}&\left\{\left(y\cdot(2^{16}+1)\right)+\overline{Z\cdot x}\right\} \bmod (2^{16}+1)\\ &\equiv \left(y\cdot(2^{16}+1)\right)\bmod(2^{16}+1)+\left(\overline{Z\cdot x}\right)\bmod(2^{16}+1)\\ &\equiv 0+\left(\overline{Z\cdot x}\right)\bmod(2^{16}+1) \qquad 2.9\end{aligned}$$

ใช้ผลการอุปนัยจากสมการ 2.8 และ 2.9 จะได้ค่า x เป็นค่าผกผันมอดุโลวิธีปรับปรุงของผลคูณตามสมการที่ 2.10

$$1 \equiv \left(\overline{Z\cdot x}\right)\bmod(2^{16}+1) \text{ หรือ } \left(\overline{Z\cdot x}\right)\equiv 1 \bmod(2^{16}+1) \qquad 2.10$$

ดังนั้นเมื่อให้ x แทนค่ากุญแจผกผัน Z^{-1} จะได้ผลของตัวดำเนินการ $Z \odot Z^{-1}=1$ จะสังเกตได้ว่าส่วนเติมเต็มเลขฐานสองของ $Z\cdot x$ ภายใต้กลุ่ม $\left(\mathbb{Z}^{*}_{2^{16}+1},*\right)$ นี้จะต้องให้ค่าจำนวนเฉพาะเท่านั้นจึงได้ให้ผลสอดคล้องกับสมการ 2.10 กรณียกเว้นที่ $Z\cdot x=0$ จะใช้ค่า Z^{-1} เป็นศูนย์ ตารางที่ 2.2 แสดงความสัมพันธ์ของรหัสกุญแจย่อยในการเข้าและถอดรหัสที่รอบต่างๆ

ตารางที่ 2.2 รหัสกุญแจสำหรับการเข้าและถอดรหัสไอดีอีเอ

การเข้ารหัส

รอบที่ 1	$Z_1^{(1)}$	$Z_2^{(1)}$	$Z_3^{(1)}$	$Z_4^{(1)}$	$Z_5^{(1)}$	$Z_6^{(1)}$
รอบที่ 2	$Z_1^{(2)}$	$Z_2^{(2)}$	$Z_3^{(2)}$	$Z_4^{(2)}$	$Z_5^{(2)}$	$Z_6^{(2)}$
รอบที่ 3	$Z_1^{(3)}$	$Z_2^{(3)}$	$Z_3^{(3)}$	$Z_4^{(3)}$	$Z_5^{(3)}$	$Z_6^{(3)}$
รอบที่ 4	$Z_1^{(4)}$	$Z_2^{(4)}$	$Z_3^{(4)}$	$Z_4^{(4)}$	$Z_5^{(4)}$	$Z_6^{(4)}$
รอบที่ 5	$Z_1^{(5)}$	$Z_2^{(5)}$	$Z_3^{(5)}$	$Z_4^{(5)}$	$Z_5^{(5)}$	$Z_6^{(5)}$
รอบที่ 6	$Z_1^{(6)}$	$Z_2^{(6)}$	$Z_3^{(6)}$	$Z_4^{(6)}$	$Z_5^{(6)}$	$Z_6^{(6)}$
รอบที่ 7	$Z_1^{(7)}$	$Z_2^{(7)}$	$Z_3^{(7)}$	$Z_4^{(7)}$	$Z_5^{(7)}$	$Z_6^{(7)}$
รอบที่ 8	$Z_1^{(8)}$	$Z_2^{(8)}$	$Z_3^{(8)}$	$Z_4^{(8)}$	$Z_5^{(8)}$	$Z_6^{(8)}$
รอบที่ 9	$Z_1^{(9)}$	$Z_2^{(9)}$	$Z_3^{(9)}$	$Z_4^{(9)}$		

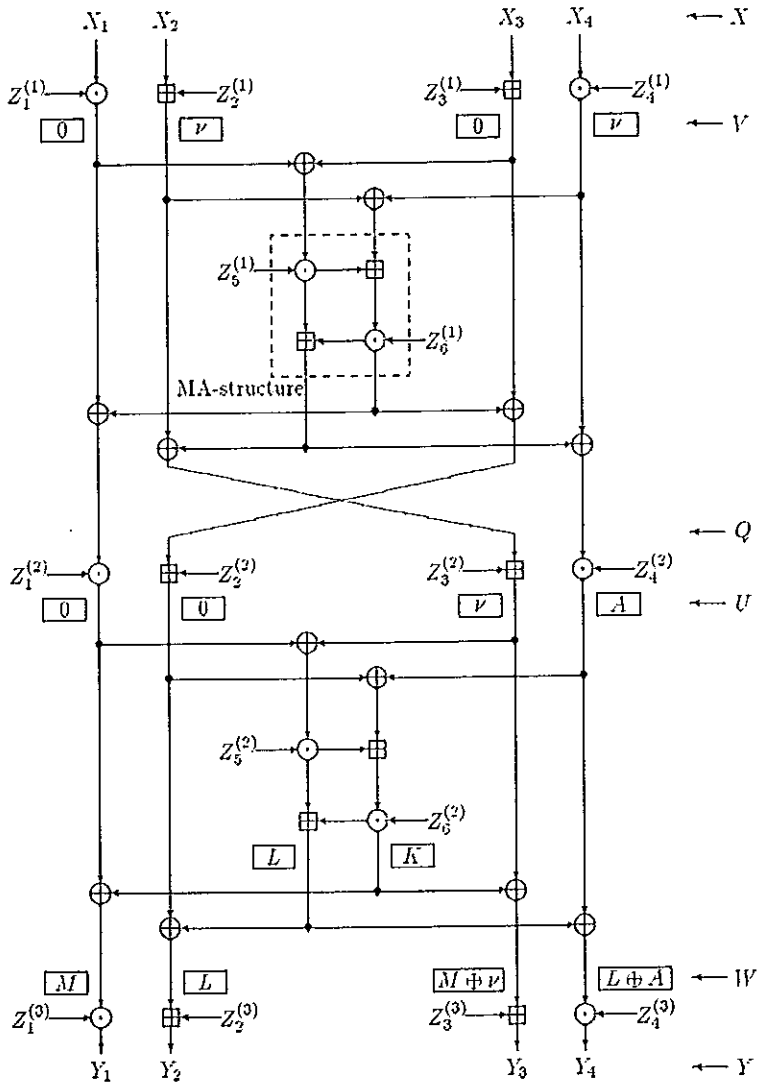
การถอดรหัส

รอบที่ 1	$Z_1^{(9)^{-1}}$	$-Z_2^{(9)}$	$-Z_3^{(9)}$	$Z_4^{(9)^{-1}}$	$Z_5^{(8)}$	$Z_6^{(8)}$
รอบที่ 2	$Z_1^{(8)^{-1}}$	$-Z_2^{(8)}$	$-Z_3^{(8)}$	$Z_4^{(8)^{-1}}$	$Z_5^{(7)}$	$Z_6^{(7)}$
รอบที่ 3	$Z_1^{(7)^{-1}}$	$-Z_2^{(7)}$	$-Z_3^{(7)}$	$Z_4^{(7)^{-1}}$	$Z_5^{(6)}$	$Z_6^{(6)}$
รอบที่ 4	$Z_1^{(6)^{-1}}$	$-Z_2^{(6)}$	$-Z_3^{(6)}$	$Z_4^{(6)^{-1}}$	$Z_5^{(5)}$	$Z_6^{(5)}$
รอบที่ 5	$Z_1^{(5)^{-1}}$	$-Z_2^{(5)}$	$-Z_3^{(5)}$	$Z_4^{(5)^{-1}}$	$Z_5^{(4)}$	$Z_6^{(4)}$
รอบที่ 6	$Z_1^{(4)^{-1}}$	$-Z_2^{(4)}$	$-Z_3^{(4)}$	$Z_4^{(4)^{-1}}$	$Z_5^{(3)}$	$Z_6^{(3)}$
รอบที่ 7	$Z_1^{(3)^{-1}}$	$-Z_2^{(3)}$	$-Z_3^{(3)}$	$Z_4^{(3)^{-1}}$	$Z_5^{(2)}$	$Z_6^{(2)}$
รอบที่ 8	$Z_1^{(2)^{-1}}$	$-Z_2^{(2)}$	$-Z_3^{(2)}$	$Z_4^{(2)^{-1}}$	$Z_5^{(1)}$	$Z_6^{(1)}$
รอบที่ 9	$Z_1^{(1)^{-1}}$	$-Z_2^{(1)}$	$-Z_3^{(1)}$	$Z_4^{(1)^{-1}}$		

โดยที่ Z^{-1} คือค่าที่ได้ผลของ $Z \odot Z^{-1} = 1$ และ $-Z$ คือค่าที่ได้ผลของ $-Z \boxplus Z = 0$

2.6 ความปลอดภัยของระบบรหัสลับไอดีอีเอ

หลังจากการทำเสนอการวิเคราะห์รหัสกุญแจด้วยวิธีการหาอนุพันธ์ (differential cryptanalysis) ทำให้ระบบรหัสลับไอดีอีเอถูกพัฒนาเพื่อต้านทานต่อการวิเคราะห์ด้วยระบบดังกล่าว ผลของการปรับปรุงทำให้ระบบรหัสลับไอดีอีเอมีค่าความน่าจะเป็นในการวิเคราะห์ประมาณ 2^{-18} ในรอบแรก และค่าความน่าจะเป็นลดต่ำกว่า 2^{-86} ในการทำซ้ำ 3 รอบ และจะต่ำกว่าเมื่อทำซ้ำจนครบ 9 รอบ (X.Lai and J.Massey, 1991: 30)



ภาพประกอบที่ 2.6 กราฟการคณนาของ 2 รอบแรกในการวิเคราะห์แบบอนุพันธ์
(ที่มา :J.Daemen *et al*,1992: 421)

เมื่อทำการวิเคราะห์ด้วยวิธีการหาอนุพันธ์สำหรับ 2.5 รอบ (J.Daemen *et al.*,1992 :421) ซึ่งแสดงไว้ในภาพประกอบที่ 2.4 โดยเริ่มต้นใช้กระบวนการแจกแจงกรณี (exhaustive process) กับกุญแจย่อยที่ 4 ในรอบที่ 1 ($Z_4^{(1)}$) เมื่อกำหนดให้ข้อมูลเข้า 16 บิตแรก (X_1) และ 16 บิตในชุดที่สาม (X_3) เป็นศูนย์ทั้ง 16 ตัว ส่วนข้อมูลเข้าในชุดที่ 2 (X_2) และข้อมูลเข้าในชุดที่ 4 (X_4) เป็นค่าใดๆ จะพบความสัมพันธ์ของกุญแจย่อย $Z_1^{(3)}$ ในเริ่มต้นชุดข้อมูลที่ 1 ของรอบที่ 3 ซึ่งแทนด้วยสัญกร M โดยจะสัมพันธ์กับกุญแจย่อย $Z_3^{(3)}$ ในค่าเริ่มต้นชุดข้อมูลที่ 3 ของรอบที่ 3 ซึ่ง

แทนด้วยสัญกร $M \oplus \nu$ ในความสัมพันธ์ดังกล่าวนี้เมื่อสุ่มข้อมูลจำนวน 2^{10} ข้อมูลป้อนให้เป็นค่า υ จะได้ความสัมพันธ์ของกุญแจย่อย $Z_2^{(1)}, Z_4^{(1)}, Z_1^{(3)}$ และ $Z_3^{(3)}$ เป็นค่าตั้งต้น และเมื่อได้ค่า $Z_1^{(3)}$ และ $Z_3^{(3)}$ จะสามารถคำนวณย้อนกลับไปหาค่ากุญแจย่อย $Z_1^{(2)}$ ได้จากสมการ

$$K = \left((Z_1^{(2)} \cdot (A \oplus \upsilon)) \bmod (2^{16}+1)\right) \oplus \left(\left(A \cdot Z_1^{(2)}\right) \bmod (2^{16}+1)\right) \quad 2.11$$

ทั้งนี้จะต้องอาศัยการสุ่มข้อมูล A แล้วแทนค่าความสัมพันธ์ลงในสมการ 2.11 และผลของสมการนี้จะให้ค่ากุญแจย่อย $Z_4^{(2)}$ โดยการนิรนัยแล้วนำไปหากุญแจย่อยอื่นๆต่อโดยอาศัยความสัมพันธ์จากสมการ

$$Z_5^{(i)} = W_1^{-1} \cdot \left(W_2 + (2^{16} - W_3)\right) \quad 2.12$$

$$Z_6^{(i)} = W_3 \cdot \left(W_4 + \left(\left(Z_5^{(2)} \cdot W_1\right) \bmod (2^{16}+1)\right)\right)^{-1} \quad 2.13$$

$$Z_2^{(2)} = 2^{16} - \left(V_3 \oplus Q_1 \oplus V_1\right) \quad 2.14$$

$$Z_3^{(2)} = 2^{16} - \left(V_2 \oplus Q_4 \oplus V_4\right) \quad 2.15$$

เมื่อ เวกเตอร์ $\hat{W}, \hat{Q}$ และ $\hat{V}$ ได้จากการแจกแจงแบบสุ่มในวง $\mathbb{Z}_{2^{16}}$

ผลการวิเคราะห์ระบบรหัสลับไอดีอีเอด้วยวิธีนี้จะต้องใช้ข้อมูลข่าวสาร 2^{42} ชุด ทำการเข้ารหัส 2^{106} ครั้งจึงจะได้ความน่าจะเป็นในการวิเคราะห์กุญแจมากกว่า 80 %

และเมื่อมีการเสนอรายงานการวิจัยเกี่ยวกับการวิเคราะห์รหัสลับแบบเชิง ทำให้มีความสนใจที่จะวิเคราะห์ระบบรหัสลับไอดีอีเอด้วยวิธีการเชิงเส้นเช่นกันแต่ไม่ประสบความสำเร็จจนกระทั่งมีการผสานหลายวิธีเข้าด้วยกัน วิธีการหาอนุพันธ์ตัดปลาย (truncated-differential) และวิธีการหาอนุพันธ์-ประมาณค่าเชิงเส้น (differential-linear) ได้ถูกทดลอง (J.Borst *et al.*, 1997 : 10) และพบว่าให้ประสิทธิภาพสูงสุดในการวิเคราะห์รหัสกุญแจก็ยังไม่สามารถวิเคราะห์รหัสกุญแจของระบบรหัสลับไอดีอีเอได้สมบูรณ์ กล่าวคือทำการวิเคราะห์เพียง 3.5 รอบ ได้ผลความน่าจะเป็นสูงสุด 86% เมื่อสุ่มข้อมูล 2^{56} โดยทำการเข้ารหัส 2^{67} ครั้งด้วยวิธีการหาอนุพันธ์ตัดปลาย และได้ผลสมบูรณ์เมื่อใช้ข้อมูลสุ่ม 2^{29} ชุดทำการเข้ารหัส 2^{44} ครั้งด้วยวิธีการหาอนุพันธ์-ประมาณค่าเชิงเส้นสำหรับการเข้ารหัส 3 รอบ และไม่สามารถอนุมาน ใช้กับวิธีการนี้ได้สำหรับการทำซ้ำ 9 รอบ จึงถือได้ว่าระบบรหัสลับไอดีอีเอมีความปลอดภัยสูงเมื่อเทียบการระบบรหัสลับอื่นๆในกลุ่มเดียวกัน

2.7 รหัสกุญแจที่อ่อนแอ

รหัสกุญแจที่อ่อนแอจะนิยามได้จากผลของตัวรหัสลับที่มีค่าความน่าจะเป็นใกล้ขอบ(ค่าความน่าจะเป็นเข้าใกล้ค่า 0 หรือ 1) สังเกตจากรหัสกุญแจย่อยที่ทำให้เกิดผลของ $\hat{V} = (0, \upsilon, o, \upsilon)$ ตามภาพประกอบที่ 2.4 จะทำให้เหลือรหัสกุญแจย่อยที่ไม่ทราบค่าเพียง 51 บิต จากรหัสกุญแจย่อย 128 บิต ตำแหน่งทั้ง 51 ได้แก่ บิตที่ 26 - 40 ,72 - 83 และบิตที่ 99 –112 และเมื่อพิจารณาการกระจายกุญแจย่อยแล้วยังสามารถคำนวณหาความสัมพันธ์ของกุญแจย่อย $Z_1^{(9)}$ โดยการสุ่ม 12 บิตต่ำ (LSB) ซึ่งเหลือค่าความน่าจะเป็นสูงสุดเพียง 2^{-4} เท่านั้น และผลดังกล่าวทำให้ทราบ 3 บิตในกุญแจย่อย $Z_2^{(9)}$ ทราบ 12 บิตในกุญแจย่อย $Z_2^{(9)}$ และทราบ 7 บิตในกุญแจย่อย $Z_4^{(8)}$ และในที่สุดจะเหลือกุญแจย่อยที่ต้องแจกแจงด้วยการสุ่มเพียง 17 บิตเท่านั้น (J.Daemon *et al.*,1993: 87) และได้มีการเสนอรายงานการวิจัยให้ใช้รหัสกุญแจ 144 บิตสำหรับระบบรหัสลับไอดีอีเอโดย 16 บิตที่เพิ่มขึ้นมาจะเรียกว่ากุญแจอัลฟา (α-key) โดยที่กุญแจย่อย 52 ชุดหลังการกระจายค่านี้ทุกตัวเมื่อดำเนินปฏิบัติการด้วยกุญแจอัลฟาแล้วจะไม่ทำให้ค่าในกุญแจย่อยนั้นเป็นศูนย์ เขียนความสัมพันธ์ของ กุญแจย่อยใหม่ที่ได้ดำเนินปฏิบัติการกับกุญแจอัลฟาได้ว่า (J.Daemon *et al.*,1994 :230-231)

$$\hat{Z}_i^{(r)} = \alpha \oplus Z_i^{(r)} \qquad 2.16$$

2.8 สรุป

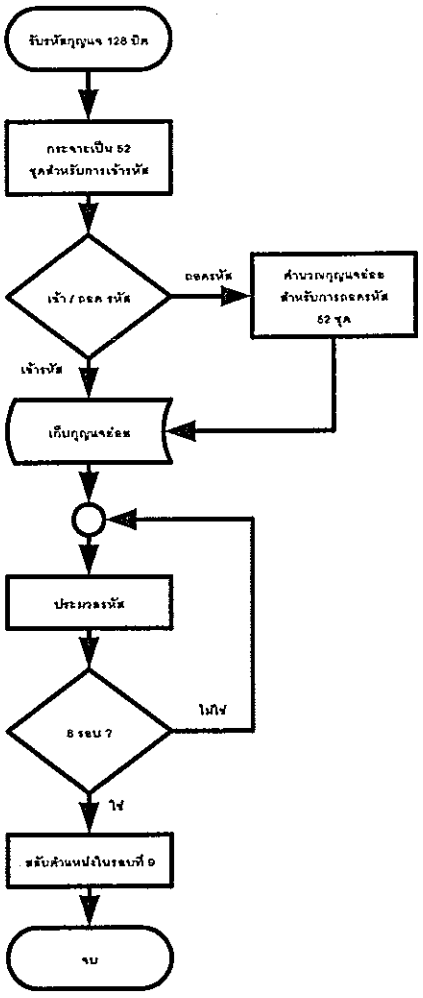
การศึกษากระบวนการและขั้นตอนของรหัสลับไอดีอีเอในเชิงความสัมพันธ์ทางคณิตศาสตร์ โดยอาศัยทฤษฎีจำนวนและพีชคณิตนามธรรม จะทำให้ทราบถึงกลไกและประสิทธิภาพของระบบรหัสลับอันจะมีประโยชน์ในการนำไปออกแบบและพัฒนาตัวดำเนินการในขั้นตอนต่างๆ เพื่อนำไปเขียนโปรแกรมคอมพิวเตอร์ในลำดับต่อไป

บทที่ 3

โปรแกรมคอมพิวเตอร์สำหรับรหัสลับไอดีอีเอ

3.1 บทนำ

ผู้วิจัยได้ทำการเขียนโปรแกรมคอมพิวเตอร์เพื่อทดสอบการทำงานระบบรหัสลับไอดีอีเอ ในการวิจัยนี้เลือกใช้ภาษาซี เนื่องจากเป็นภาษาโครงสร้างที่มีความเหมาะสมในการพัฒนางานทางด้านคณิตศาสตร์และงานวิศวกรรม กำหนดให้โปรแกรมทำงานในระบบเลขฐานสอง และทำซ้ำ 9 รอบ เพื่อให้เป็นไปตามข้อกำหนดของรหัสลับไอดีอีเอ แสดงแผนภูมิสายงานไว้ในภาพประกอบที่ 3.1



ภาพประกอบที่ 3.1 แผนภูมิสายงานของโปรแกรมระบบรหัสลับไอดีอีเอ

ผลของโปรแกรมนี้จะนำไปสู่การอ้างอิงและสนับสนุนผลการทดสอบวงจรรวมต้นแบบในการประมวลผลต่อไป

3.2 การสร้างรหัสกุญแจย่อย 52 ชุด

ในส่วนนี้รหัสกุญแจ 128 บิตจะถูกกระจายให้เป็นรหัสกุญแจย่อย 52 ชุด ชุดละ 16 บิต รวม 832 บิตดังตารางที่ 2.1 ซึ่งใช้วิธีการเลื่อนตำแหน่งครั้งละ 25 บิต วิธีการดังกล่าวมีความไม่สะดวกจึงได้ปรับปรุงใหม่ในมีความกระชับ และสะดวกขึ้นโดยเขียนเป็นฟังก์ชันในภาษาซีดังแสดงไว้ในภาพประกอบที่ 3.2

```
void Expandkey(u_int16 *ukey, u_int16 *key)
{
        int i;
        for (i=0; i<8; i++) key[i]=ukey[i];
        for (i=8; i<52; i++) {
                if ((i & 7) < 6)
                        key[i]=(((key[i-7] & 127) << 9) | (key[i-6] >> 7));
                else if ((i & 7) == 6)
                        key[i]=(((key[i-7] & 127) << 9) | (key[i-14] >> 7));
                else
                        key[i]=(((key[i-15] & 127) << 9) | (key[i-14] >> 7);
        }
}
```

ภาพประกอบที่ 3.2 ฟังก์ชันการสร้างรหัสกุญแจย่อย 52 ชุด

ตัวแปร ukey และตัวแปร key เป็นข้อมูล unsigned int ชนิดแถวลำดับ (array) 1 มิติ และประกาศเป็นตัวแปรชนิดตัวชี้ (pointer) โดยที่ ukey เป็นรหัสกุญแจตัวละ 16 บิตจำนวน 8 ตัวรวม 128 บิต ส่วน key รหัสกุญแจที่ได้ถูกกระจายให้เป็นรหัสกุญแจย่อย 52 ชุด ชุดละ 16 บิต รวม 832 บิต

การดำเนิน (i&7) คือการพิจารณาลำดับของกุญแจในตำแหน่งโดยลำดับที่ (i&7)<6 หมายถึงกุญแจในลำดับที่ key[8] ,...,key[13] ,key[16] ,...key[21],key[24],...key[29],key [32],...,key[37], key[40],...,key[45] และ key[48],...,key[52] จากตารางที่ 2.1 จะสังเกตได้ว่า กุญแจในตำแหน่งที่ 8 จะได้จาก การเลื่อนตำแหน่งกุญแจย่อยที่ 2 (key[2]) จากบิตต่ำไปบิตสูง 9 ตำแหน่ง นั่นคือบิตที่ 25 ใน key[2] จะถูกยกเป็นบิตแรกใน key[8] และบิตที่ 26 จนถึงบิตที่ 31ซึ่งเป็น 7 บิตหลังใน key[2] จะกลายเป็น 7 บิตแรกของ key[8] และเมื่อ

สังเกตต่อใน key[3] จะพบว่า 9 บิตแรกของ key[3] จะปรากฏเป็น 9 บิตหลังใน key[8] และเป็นเช่นนี้ตลอดในกุญแจย่อยที่กล่าวมาข้างต้น และเมื่อพิจารณาเงื่อนไข (i&7)=6 จะเป็นการพิจารณากุญแจย่อยในลำดับที่ key[14],key[22] ,.... สำหรับเงื่อนไขอื่นๆจะพิจารณากุญแจย่อยในลำดับที่ key[15],key[23],... โดยมีกระบวนการเป็นไปคล้ายการพิจารณากับเงื่อนไขข้างต้น ผู้วิจัยจึงอาศัยหลักการนี้ปรับปรุงระเบียบวิธีการกระจายรหัสกุญแจ และเมื่อทดสอบจะได้ค่าเช่นเดียวกับการกระจายด้วยวิธีเลื่อน แบบวงรอบไปทางบิตต่ำ 25 ตำแหน่งโดยตรง ซึ่งเป็นไปตามข้อกำหนดของรหัสลับไอดีอีเอ

3.3 ตัวดำเนินการมอดุโลของผลบวกและมอดุโลวิธีปรับปรุงของผลคูณ

ในส่วนของตัวดำเนินการมอดุโลของผลบวกนั้นเมื่อเขียนด้วยภาษาซีโดยกำหนดขนาดตัวแปรเป็น unsined int ทำให้ผลของตัวดำเนินการบวกถูกจำกัดขนาดเป็น 16 บิตตามเงื่อนไขของตัวแปรโดยทันที

ในส่วนของตัวดำเนินการมอดุโลวิธีปรับปรุงของผลคูณนั้นจะใช้ "ขั้นตอนต่ำ-สูงสำหรับการคูณ (low-high algorithm for multiplication)" เพื่อแก้ปัญหาการใช้ตัวดำเนินการมอดุโลขนาด 17 บิต ที่ว่า

<u>ทฤษฎีบทช่วยที่</u> 1 ขั้นตอนต่ำ-สูงสำหรับการคูณ

ถ้า $a,b \in \mathbb{Z}_{2^{16}}$ และ $a,b \neq 0$ แล้ว $(a \cdot b) \bmod \left(2^{16}+1\right)$ จะสามารถหาได้จาก

$$(a \cdot b)\bmod 2^{16} - (a \cdot b)\operatorname{div} 2^{16}$$
$$\text{เมื่อ } (a \cdot b)\bmod 2^{16} \geq (a \cdot b)\operatorname{div} 2^{16}$$

หรือ
$$(a \cdot b)\bmod 2^{16} - (a \cdot b)\operatorname{div} 2^{16} + 2^{16} + 1$$
$$\text{เมื่อ } (a \cdot b)\bmod 2^{16} < (a \cdot b)\operatorname{div} 2^{16} \qquad (3.1)$$

เนื่องจากผลคูณของ a,b จะมีขนาด 32 บิต ดังนั้นเมื่อบรรจุลงในตัวแปร 16 บิต จะให้ความหมายเช่นเดียวกับ $(a \cdot b) \bmod 2^{16}$ สำหรับ $(a \cdot b)\operatorname{div} 2^{16}$ นั้นหาได้จากการเลื่อนตำแหน่งไปทางขวา 16 ตำแหน่งแล้วบรรจุลงในตัวแปร 16 บิต ทำให้การประมวลผลตัว

ดำเนินการวิธีปรับปรุงมอดุโลของผลคูณลดขั้นตอนลงและมีความกระทัดรัดมากขึ้น ซึ่งแสดงเป็นฟังก์ชันของภาษาซีในภาพประกอบที่ 3.3

```
u_int16 mul(u_int16 x, u_int16 y)
{
        u_int32 p;
        p=(u_int32)x*y;
        if (p == 0)
                x = 65537l-x-y;
        else {
                x = p >> 16;
                y = p;
                x = y-x;
                if (y < x) x += 65537l;
        }
        return x;
}
```

ภาพประกอบที่ 3.3 ฟังก์ชันตัวดำเนินการมอดุโลวิธีปรับปรุงของผลคูณ

3.4 ตัวดำเนินการค่าผกผันมอดุโลวิธีปรับปรุงของผลคูณและค่าผกผันมอดุโลของผลบวก

ในบทที่ 2 กล่าวถึงการหาค่าผกผันมอดุโลวิธีปรับปรุงของผลคูณว่าสามารถหาได้จากการใช้ฟังก์ชันหารร่วมมากซึ่งจะใช้ขั้นตอนวิธีของยุคลิด (extended Euclidean algorithm) สำหรับการหาตัวหารร่วมมาก (A.J.Menezes *et al* ,1997: 67) ดังแสดงรหัสเทียม (pseudocode) ของขั้นตอนดังกล่าวในภาพประกอบที่ 3.4 และ แสดงตัวอย่างฟังก์ชันในการพัฒนาด้วยโปรแกรมคอมพิวเตอร์ภาษาซี ในภาพประกอบที่ 3.5

INPUT $x,y ; x \geq y > 0$
OUTPUT d =GCD (a,b) and intergers x,y satisfying ax+by=d
1. If $b = 0$ then
   set $d \leftarrow a , x \leftarrow 1 , y \leftarrow 0$, and return (d,x,y)
2. Set $x_2 \leftarrow 1 , x_1 \leftarrow 0, y_2 \leftarrow 0 , y_1 \leftarrow 1$
3. while $b > 0$ do the following :
   3.1 $q \leftarrow \lfloor a / b \rfloor , r \leftarrow a - qb , x \leftarrow x_2 - qx_1 , y \leftarrow y_2 - qy_1$
   3.2 $a \leftarrow b , b \leftarrow r , x_2 \leftarrow x_1 , x_1 \leftarrow x , y_2 \leftarrow y , y_1 \leftarrow y$
4. Set $d \leftarrow a , x \leftarrow x , y \leftarrow y$
5. Return (d,x,y)

ภาพประกอบที่ 3.4 รหัสเทียมของขั้นตอนวิธียุคลิดสำหรับการหาตัวหารร่วมมาก

```
u_int16 mulinv(u_int16 b)
{       long a, q, r, x, y, t;
        if(b ==0)
        y=0;
        else {  a=65537l; y=1;  x=0;
                do {    r=(a%b);
                        q=(a-r)/b;
                        if(r==0) {      if(y<0)  y=65537l+y;
                                } else {  a=b;  b=r; t=y;
                                        y=x-q*y; x=t;    }
                } while (r!=0);
            }
        return (u_int16)y;
}
```

ภาพประกอบที่ 3.5 ฟังก์ชันขั้นตอนวิธีของยุคลิดสำหรับการหาตัวหารร่วมมาก

เนื่องจากตัวดำเนินการหารจะต้องใช้หลายวงรอบเวลาการประมวลผลในคอมพิวเตอร์ ดังนั้นผู้วิจัยจึงปรับปรุงโดยใช้ขั้นตอนทวินามวิธีของยุคลิด (binary extended Euclidean algorithm) ซึ่งใช้วิธีการเลื่อนตำแหน่งไปทางซ้ายและการหารแบบมอดุโลสองเพื่อลดวงรอบเวลาการประมวลผลในคอมพิวเตอร์ แสดงรายละเอียดของรหัสเทียมในภาพประกอบที่ 3.6 และฟังก์ชันในการพัฒนาด้วยโปรแกรมคอมพิวเตอร์ภาษาซีในภาพประกอบที่ 3.7 (A.J.Menezes *et al.*,1997 :608-611)

INPUT: two positive integers x and y.
OUTPUT: integers a, b,and v such that ax + by = v,where v = gcd(x; y).

1. g ←1.
2. While x and y are both even, do the following: x← x/2, y← y/2, g← 2g.
3. u ←x, v← y, A ←1, B← 0, C← 0, D← 1.
4. While u is even do the following:
   4.1 u ←u/2.
   4.2 If A ≡ B ≡ 0 (mod 2) then A ←A/2, B← B/2; otherwise, A← (A + y)/2, B← (B - x)/2.
5. While v is even do the following:
   5.1 v ←v/2.
   5.2 If C ≡ D ≡ 0 (mod 2) then C← C/2, D← D/2; otherwise, C← (C +y)/2, D (D _ x)=2.
6. If u ≥ v then u ←u - v, A← A - C, B← B - D;
   otherwise, v ←v - u, C ←C - A, D ←D - B.
7. If u = 0,then a C, b D, and return(a; b; g * v); otherwise, go to step 4.

ภาพประกอบที่ 3.6 รหัสเทียมของขั้นตอนทวินามวิธียุคลิดสำหรับการหาตัวหารร่วมมาก

```
u_int16 mulinv (u_int y)
{       long  u, v, a, b, c, d ;
        u = 655371; v = y ; a = 1 ; b = 0 ; c = 0 ; d = 1 ;
        do {
                do {    u = u mod2 ;
                        if ((a mod 2 == b mod 2) == 0) {
                                a = a mod 2;   b = b mod 2;
                        } else {  a = (a+y ) mod 2;  b = (b-x ) mod 2 ;
                        }
                } while (u mod 2 == 0) ;
                do {    v = v mod 2;
                        if (( c mod 2 == d mod 2) == 0 ) {
                                c = c mod 2; d = d mod 2 ;
                        }else {   c = (c + y) mod 2; d = (d - x)mod 2;
                        }
                } while (v mod 2 == 0) ;
                if u >= v {u = u-v ; a = a-c ; b = b-d ;
                } else {  v = v-u; c = c-a ;d = d-b;
                }
        }while ( u = 0) ;
return (u_int16)d;
}
```

ภาพประกอบที่ 3.7 ฟังก์ชันขั้นตอนทวินามวิธีของยุคลิด
สำหรับการหาค่าผกผันมอดุโลวิธีปรับปรุงของผลคูณ

สำหรับการหาค่าผกผันมอดุโลของผลบวกสามารถพัฒนาเป็นฟังก์ชันในโปรแกรมคอมพิวเตอร์ภาษาซีได้ดังแสดงในภาพประกอบที่ 3.8

```
u_int16 addinv(u_int16 x)
{       return 0-x;
}
```

ภาพประกอบที่ 3.8 ฟังก์ชันการหาค่าผกผันมอดุโลของผลบวก

3.5 การสร้างรหัสกุญแจสำหรับการถอดรหัส

เมื่อได้พัฒนาฟังก์ชันสำหรับการหาค่าผกผันมอดุโลวิธีปรับปรุงของผลคูณและค่าผกผันมอดุโลของผลบวกแล้ว ผู้วิจัยได้พัฒนาฟังก์ชันเพิ่มเติมเพื่อจัดลำดับรหัสกุญแจให้สอดคล้องกับการถอดรหัสตามตารางที่ 2.2 และแสดงฟังก์ชันนี้ในภาพประกอบที่ 3.9 เมื่อตัวแปร

in เป็นรหัสกุญแจ 52 ชุดสำหรับการเข้ารหัส และตัวแปร out เป็นรหัสกุญแจ 52 ชุดสำหรับการถอดรหัสโดยอ้างเป็นตัวแปรชนิดตัวชี้

```
void Invertkey(u_int16 *in, u_int16 *out)
{       u_int16 t1, t2, t3, t4, round;
        u_int16 *p;
        p= out + 52; /* We work backwards */
        t1 = mulinv(*in++);         t4 = mulinv(*in++);
        t2 = addinv(*in++);         t3 = addinv(*in++);
        *--p = t4; *--p = t3; *--p = t2; *--p = t1;
        for (round = 1; round < 8; round++) {
                t1 = *in++; t2 = *in++; *--p = t2;    *--p = t1;
                t1 = mulinv(*in++);    t4 = mulinv(*in++);
                t2 = addinv(*in++);    t3 = addinv(*in++);
                *--p = t4;
                *--p = t2; /* NB: Order */
                *--p = t3;
                *--p = t1;
        }
        t1 = *in++; t2 = *in++; *--p = t2; *--p = t1;
        t1 = mulinv(*in++);  t4 = mulinv(*in++);
        t2 = addinv(*in++);  t3 = addinv(*in++);
        *--p = t4; *--p = t3; *--p = t2; *--p = t1;
}
```

ภาพประกอบที่ 3.9 ฟังก์ชันการกระจายกุญแจย่อย 52 ชุดสำหรับการถอดรหัส

```
void cip(u_int16 input[4], u_int16 output[4], u_int16 key[52])
{       int i;
        u_int16 x1, x2, x3, x4, t2, t3;
        x1 = input[0];     x2 = input[1];     x3 = input[2];     x4 = input[3];
        for (i = 0; i < 8; i++) {
                x1 = mul(x1, key[0]);       x4 = mul(x4, key[3]);
                x2 += key[1];               x3 += key[2];
                t3 = x3;            x3 ^= x1;          x3 = mul(x3, key[4]);
                t2 = x2;            x2 ^= x4;          x2 += x3;          x2 = mul(x2, key[5]);
                x3 += x2;           x1 ^= x2;          x4 ^= x3;          x2 ^= t3;          x3 ^= t2;
                key += 6;
        }
        x1 = mul(x1, key[0]);       x4 = mul(x4, key[3]);
        x3 += key[1];               x2 += key[2];
        output[0] = x1;    output[1] = x3;    output[2] = x2;    output[3] = x4;
}
```

ภาพประกอบที่ 3.10 ฟังก์ชันการประมวลผลรหัส

3.6 ส่วนประมวลผลรหัส

เนื่องจากระบบรหัสลับไอดีอีเอเป็นระบบสมมาตร ดังนั้นฟังก์ชันในการเข้ารหัสและฟังก์ชันในการถอดรหัสจะใช้เป็นฟังก์ชันเดียวกัน ภาพประกอบที่ 3.10 แสดงฟังก์ชันการเข้าและถอดรหัสโดยมีตัวแปร input เป็นข้อมูลเข้าชนิดแถวลำดับ (array) 1 มิติ ตัวละ 16 บิต จำนวน 4 ตัวรวม 64 บิต และตัวแปร key เป็นรหัสกุญแจชนิดแถวลำดับ (array) 1 มิติ ตัวละ 16 บิตจำนวน 52 ตัวรวม 832 บิต และได้ผลลัพธ์ในรูปของตัวแปร output เป็นชนิดแถวลำดับ (array) 1 มิติ ตัวละ 16 บิตจำนวน 4 ตัว รวม 64 บิต

3.7 สรุป

จากผลการศึกษาและทดลองเขียนโปรแกรมภาษาซีโดยผู้วิจัยได้พัฒนาขั้นตอนวิธีต่างๆในระบบรหัสลับไอดีอีเอได้แก่ การปรับปรุงวิธีการกระจายรหัสกุญแจย่อย การใช้ขั้นตอนสูงต่ำเพื่อลดขั้นตอนในการคำนวณค่าจากตัวดำเนินการมอดุโลวิธีปรับปรุงของผลคูณ และใช้ฟังก์ชันหารร่วมมากซึ่งพัฒนาจากขั้นตอนทวินามวิธีของยุคลิดในการคำนวณค่าผกผันของตัวดำเนินการมอดุโลวิธีปรับปรุงของผลคูณ ซึ่งทั้งหมดได้แสดงโปรแกรมรหัสต้นฉบับไว้ในภาคผนวก ค2

บทที่ 4

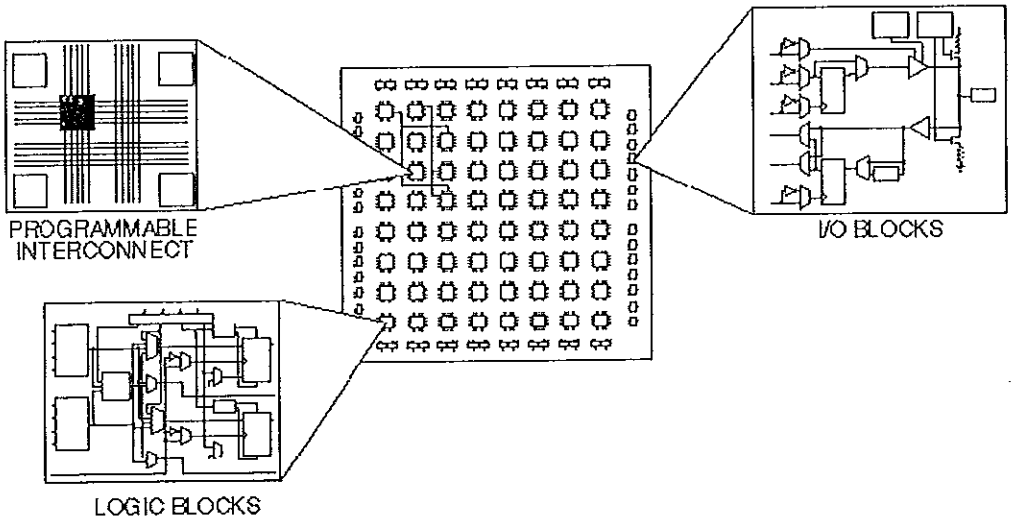
สถาปัตยกรรมฮาร์ดแวร์ของวงจรรวมต้นแบบ

4.1 บทนำ

ในงานวิจัยนี้จะใช้อุปกรณ์ดิจิตอลชนิดโปรแกรมซ้ำได้ (reconfigurable logic device) ชนิดเอฟพีจีเอ (Field Programmable Gate Array - FPGA) เป็นต้นแบบของการออกแบบวงจรดิจิตอล และเอฟพีจีเอที่ใช้เป็นของบริษัท Xilinx ในตระกูล XC4000

4.2 โครงสร้างและสถาปัตยกรรมภายในเอฟพีจีเอ

เอฟพีจีเอ XC4000 เป็นอุปกรณ์ดิจิตอลชนิดโปรแกรมซ้ำได้ที่มีหน่วยความจำแบบสถิตย์ (Static RAM – SRAM) เพื่อสร้างฟังก์ชันการทำงานวงจรตรรกะเชิงจัดหมู่ (combinatorial logic function) และจะมีสวิตช์ (switches) หรือมัลติเพล็กเซอร์ (multiplexers) เชื่อมต่อ (routing resources) แต่ละวงจรเข้าด้วย ดังแสดงในภาพประกอบที่ 4.1 และมีส่วนประกอบที่สำคัญดังนี้

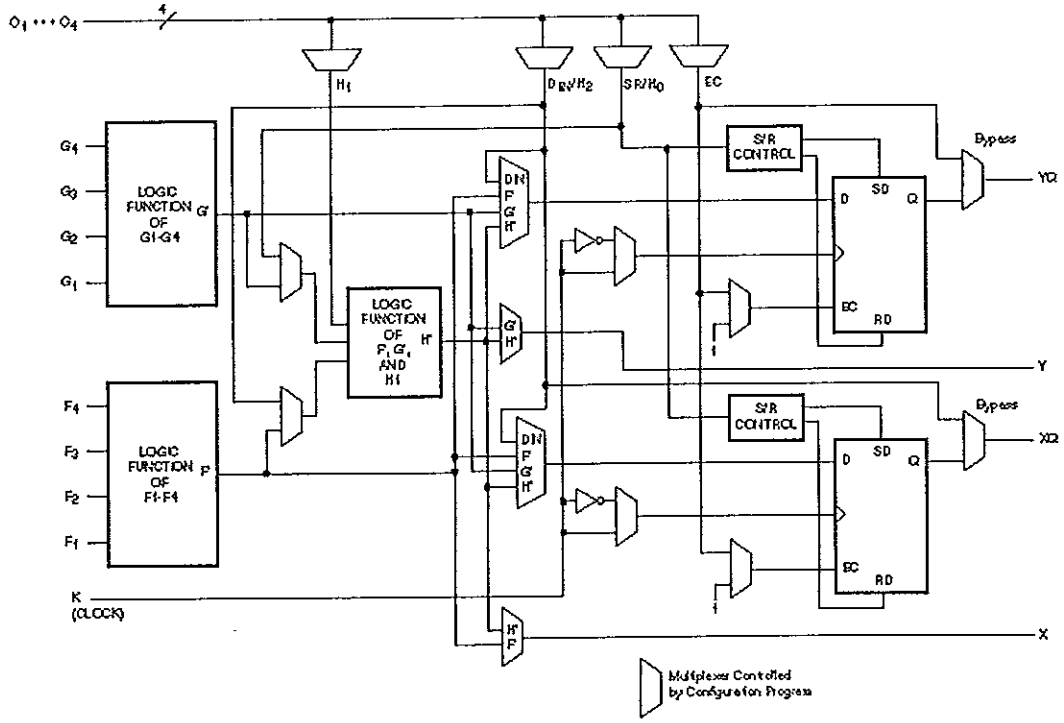


ภาพประกอบที่ 4.1 โครงสร้างพื้นฐานภายในเอฟพีจีเอของบริษัท Xilinx
(ที่มา : Xilinx,1999)

4.2.1 Configurable Logic Block (CLB)

เป็นเซลล์พื้นฐานของวงจรประกอบด้วยตัวกำเนิดฟังก์ชันจำนวน 3 ตัวคือ F, G และ H โดยฟังก์ชัน F และ G เพียงหน่วยความจำที่มีแอดเดรสขนาด 4 บิตและบิตข้อมูล 1 บิต (16X1 RAM) ซึ่งจะเก็บพฤติกรรมเชิงจัดหมู่ของสัญญาณขาเข้าสามารถแทนฟังก์ชันตรรกะได้ถึง $2^{(2^4)}$ ฟังก์ชันหรือประมาณ 65,536 รูปแบบ มีสัญญาณขาออกเป็น F' และ G' ตามลำดับ ส่วนฟังก์ชัน H จะมีอินพุตจำนวน 3 ตัว โดยรับสัญญาณจาก F',G' และจากสัญญาณขาเข้าภายนอกซีแอลบี

ฟลิปฟลอปจำนวน 2 ตัวโดยการควบคุมของสัญญาณนาฬิกา (k) สัญญาณ EC (clock enable) และสัญญาณเซ็ท/รีเซ็ท (S/R) จะถูกต่อขันดับจากฟังก์ชัน F,G และ H เพื่อใช้คงค่าสัญญาณขาออกจากตัวกำเนิดฟังก์ชันและสัญญาณขาเข้าอิสระ (Din) จากภายนอก สำหรับสัญญาณขาออกของซีแอลบีจะมี 4 ตัวคือ Y, YQ, X และ XQ โดย X สามารถป้อนกลับเป็นสัญญาณขาเข้าสำหรับฟังก์ชัน F หรือ H ส่วน Y สามารถป้อนกลับเป็นสัญญาณขาเข้าของฟังก์ชัน G หรือ H ได้เช่นเดียวกันดังแสดงในภาพประกอบที่ 4.2



ภาพประกอบที่ 4.2 แผนผังภายในของซีแอลบี

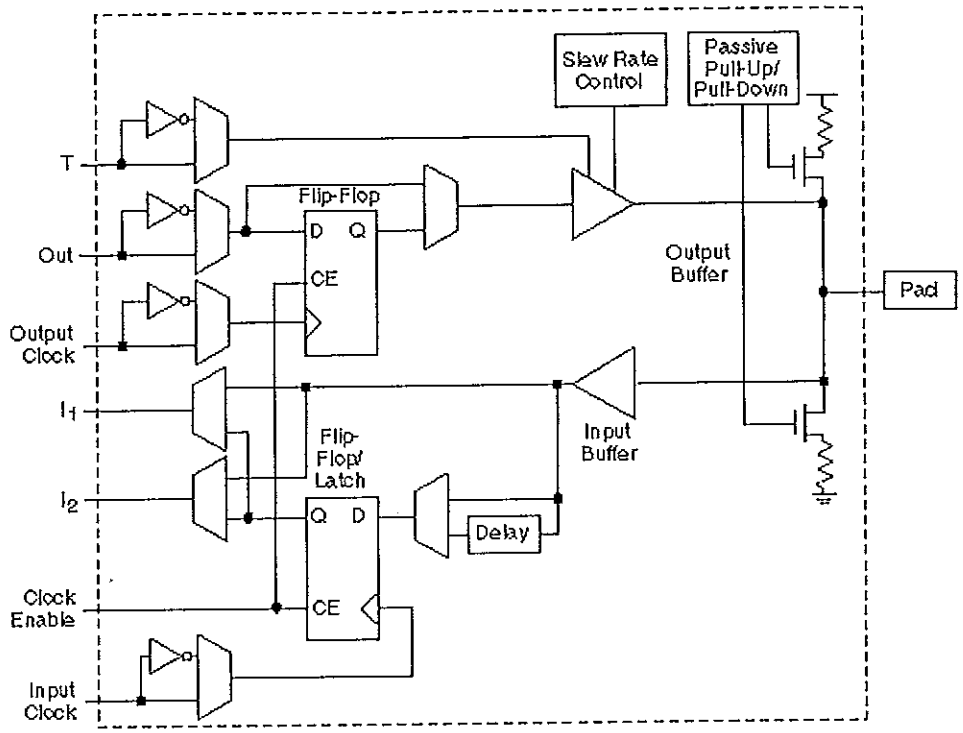
(ที่มา : Xilinx,1999)

4.2.2 Input Output Block (IOB)

ไอโอบีเป็นส่วนที่ทำให้เอฟพีจีเอสามารถติดต่อกับจุดเชื่อมโยงขาภายนอก (pad) ได้ และสามารถกำหนดคุณลักษณะให้เป็นการนำสัญญาณเข้าหรือออกเพียงทิศทางเดียว หรือส่งผ่านข้อมูลได้ 2 ทิศทาง จากภาพประกอบที่ 4.3 แสดงแผนผังภายในของไอโอบี ข้อมูลที่เข้ามาทางขาเชื่อมต่อภายนอกเพื่อส่งผ่านไปยัง CLB สัญญาณจะผ่านวงจรกันซน (buffer) และแยกออกเป็น 2 สาย สายแรกจะต่อโดยตรงกับ วงจรมัลติเพล็กเซอร์กลายเป็นสัญญาณ I_1 และ I_2 เรียกการรับสัญญาณที่เดินทางลักษณะนี้ว่า การรับข้อมูลโดยตรง สายที่ 2 จะผ่านวงจรฟลิบ-ฟลอบและต่ออันดับกับวงจรมัลติเพล็กเซอร์เพื่อส่งต่อให้กับ I_1 และ I_2 เรียกการรับสัญญาณที่เดินทางลักษณะนี้ว่าสัญญาณแบบรีจิสเตอร์ ซึ่งมีประโยชน์ในการคงค่าสัญญาณตามการควบคุมของสัญญาณนาฬิกา (clock enable และ input clock) การกำหนดคุณลักษณะของการคงค่านี้จะเป็นไปได้ 2 แบบ ได้แก่ การคงค่าด้วยขอบขา (edge-trigged latch) และการคงค่าด้วยระดับสัญญาณ (level-sensitivity latch)

สำหรับสัญญาณที่ส่งจากซีแอลบีจะต่อกับไอโอบีที่ขา O และสามารถส่งออกไปยังขาสัญญาณภายนอกได้ 2 วิธีคือส่งโดยตรงและส่งโดยค้างค่าด้วยวงจรฟลิบฟลอบเช่นเดียวกับการรับข้อมูล ก่อนสัญญาณจะถูกส่งไปยังขาภายนอก สัญญาณจะผ่านวงจรกันซนซึ่งเป็นวงจรสามสถานะ โดยมีสัญญาณ T และวงจรควบคุมอัตราการเปลี่ยนแรงดัน (slew rate control) ก่อนจะส่งให้วงจรขับเพื่อให้จ่ายกระแสได้ 12-24 มิลลิแอมป์

วงจรควบคุมอัตราการเปลี่ยนแรงดันนี้จะช่วยลดการโด่งของสัญญาณ (overshoot) ซึ่งเกิดจากภาระตัวเก็บประจุ (capacitive load) ซึ่งมีค่าระหว่าง 400-600 พิโกฟารัด (pF) อันจะทำให้เกิดการกระเพื่อมของแรงดัน 1.5 โวลต์เมื่อทำการสวิตช์ในช่วงเวลาน้อยกว่า 5 นาโนวินาที ซึ่งเป็นสาเหตุของความผิดเพี้ยนในการส่งข้อมูลสำหรับลักษณะอื่นๆของไอโอบีที่สำคัญ เช่น สามารถควบคุมการหน่วงเวลาสำหรับชดเชยการประวิงความล่าช้าของสัญญาณนาฬิกา

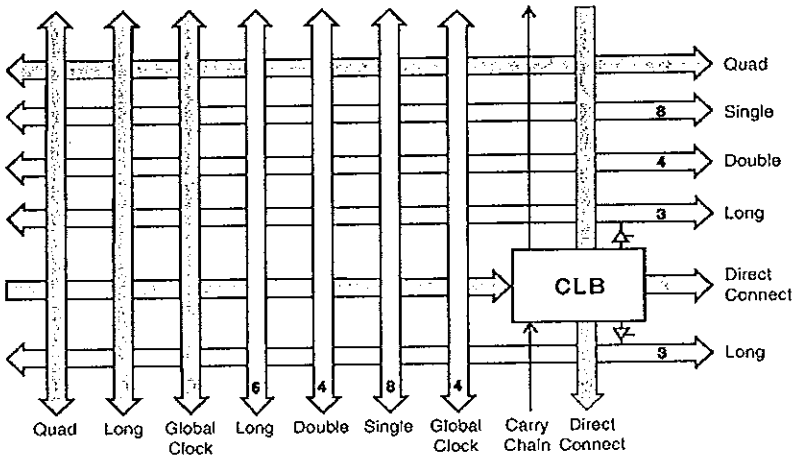


ภาพประกอบที่ 4.3 แผนผังภายในของไอโอบี

(ที่มา : Xilinx,1999)

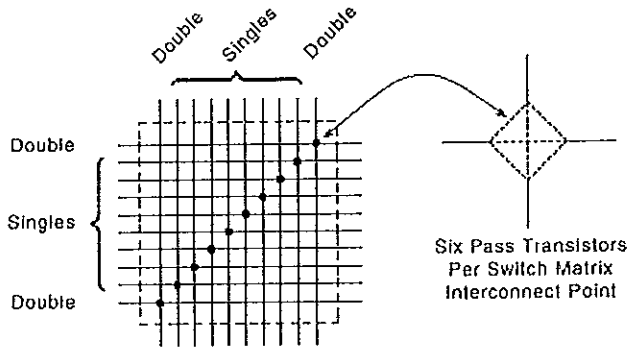
4.2.3 การเชื่อมโยงภายใน

การเชื่อมโยงทางไฟฟ้าภายในเอฟพีจีเอจะแบ่งเป็น 2 กลุ่มคือกลุ่มที่เชื่อมโยงระหว่างซีแอลบีกับซีแอลบี และกลุ่มที่เชื่อมโยงระหว่างซีแอลบีกับไอโอบี โดยมีการเชื่อมโยงแบบต่างๆ ดังแสดงในภาพประกอบที่ 4.4 ซึ่งมีทั้งการเชื่อมโยงในแนวขวางและแนวตั้งโดยมีเมตริกของสวิตช์ชนิดโปรแกรมได้ (Programmable Switch Matrix – PSM) หรือบางครั้งเรียกว่าจุดต่อร่วมชนิดโปรแกรมได้ (Programmable Interconnection Point – PIP) เป็นตัวกำหนดเส้นทางของสัญญาณพีเอสเอ็มหรือพีไอพีนี้จะประกอบขึ้นจากวงจรทรานซิสเตอร์ที่นำกระแส 2 ทิศทาง จำนวน 6 ตัวดังแสดงในภาพประกอบที่ 4.5 และ 4.6



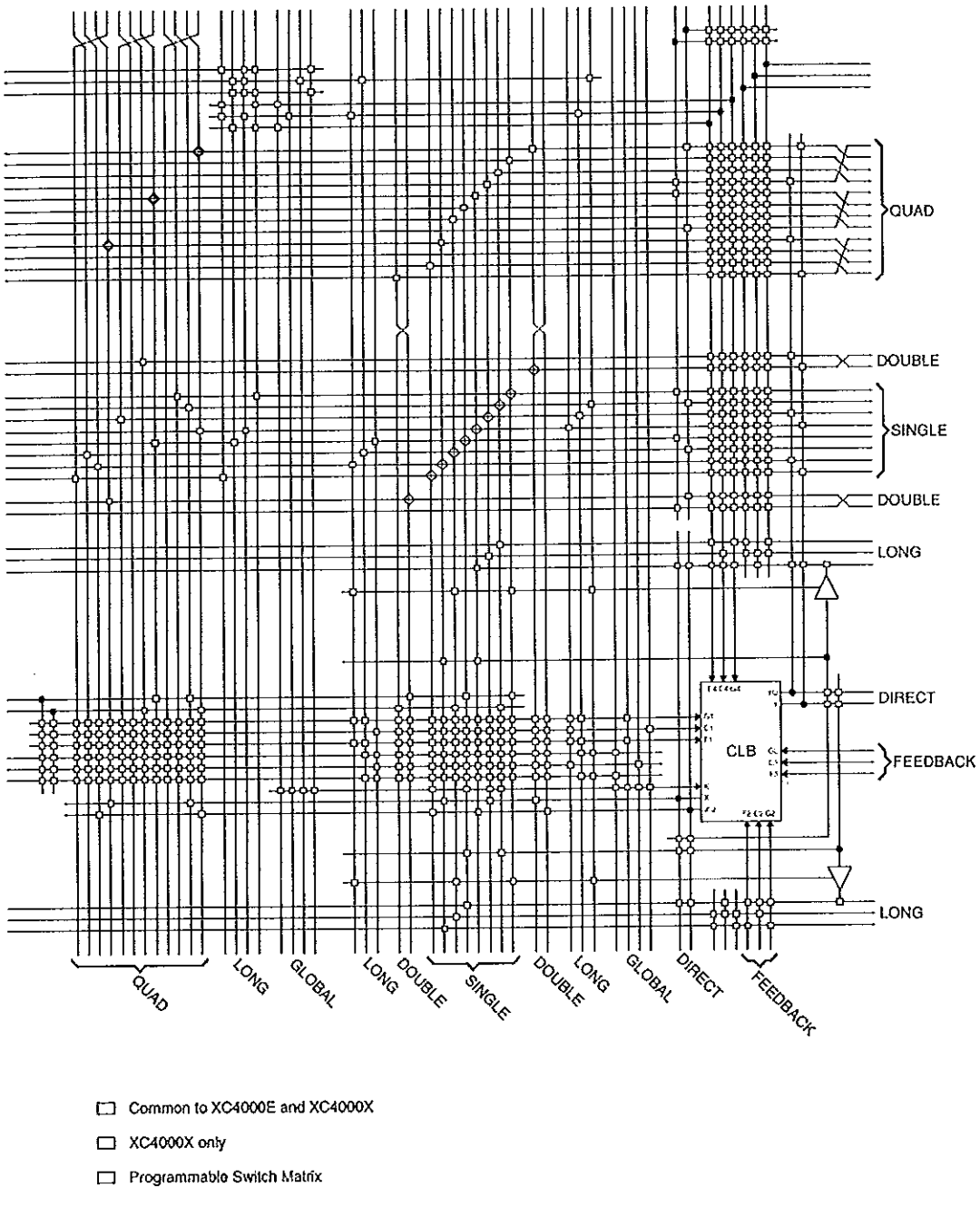
ภาพประกอบที่ 4.4 แผนผังการเชื่อมโยงซีแอลบี

(ที่มา : Xilinx,1999)



ภาพประกอบที่ 4.5 โครงสร้างของพีเอสเอ็ม

(ที่มา : Xilinx,1999)



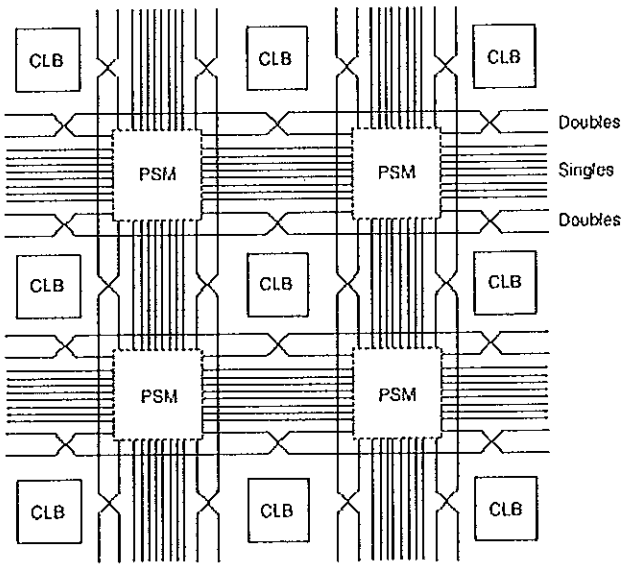
ภาพประกอบที่ 4.6 การเชื่อมโยงพีเอสเอ็มและพีไอพีของซีแอลบี

(ที่มา : Xilinx,1999)

4.2.3.1**การเชื่อมโยงเส้นเดี่ยว เส้นคู่ และ สี่เส้น** (single-length line ,double-length line และ quad line)

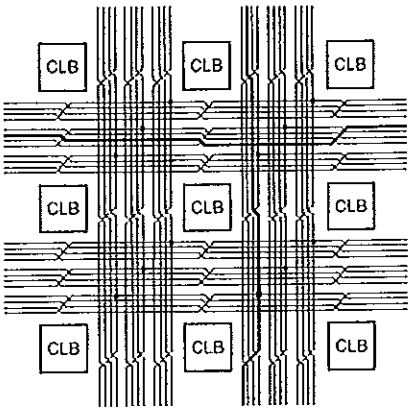
การเชื่อมโยงเส้นเดี่ยว (single-length line) และเส้นคู่ (double-length line) เป็นการเชื่อมโยงมาตรฐานระหว่างซีแอลบีประกอบด้วยเส้นทางในแนวตั้งและแนวขวางอย่างละ 8 เส้นสำหรับการเชื่อมโยงเส้นเดี่ยว ในแนวตั้งและแนวขวางอย่างละ 4 เส้นสำหรับเส้นคู่

ความแตกต่างที่เห็นได้การเชื่อมโยงเส้นเดี่ยวจะเชื่อมต่อกับซีแอลบีทุกชุด ในขณะที่การเชื่อมโยงเส้นคู่จะเชื่อมซีแอลบีชุดเว้นชุด ดังแสดงในภาพประกอบที่ 4.7 ซึ่งการเชื่อมโยงเส้นคู่จะช่วยลดการล่าช้าของการส่งสัญญาณได้ดีกว่าการเชื่อมโยงเส้นเดี่ยว เนื่องจากผ่านพีเอสเอ็มจำนวนน้อยกว่า



ภาพประกอบที่ 4.7 การเชื่อมโยงแบบการเชื่อมโยงเส้นเดี่ยวและเส้นคู่
(ที่มา : Xilinx,1999)

สำหรับการเชื่อมโยงแบบ 4 เส้นซึ่งจะมีเฉพาะ XC4000x จะเชื่อมโยงซีแอลบีห่างกัน 4 ชุดดังแสดงในภาพประกอบที่ 4.8 ด้วยเหตุผลเช่นเดียวกับการเชื่อมโยงเส้นคู่



ภาพประกอบที่ 4.8 การเชื่อมโยงแบบ 4 เส้น

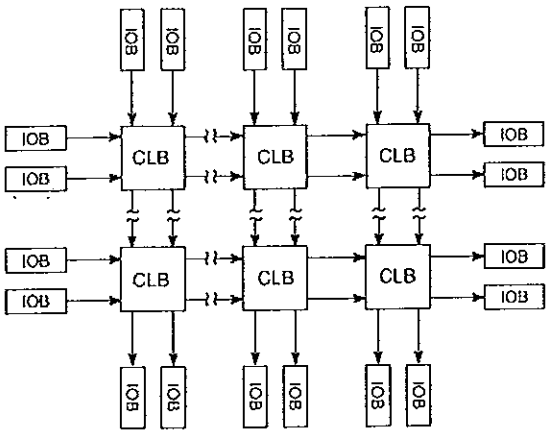
(ที่มา : Xilinx,1999)

4.2.3.2 **การเชื่อมโยงแบบลองไลน์** (longline)

การเชื่อมโยงแบบลองไลน์เป็นการเชื่อมโยงเพื่อให้ส่งสัญญาณระหว่างซีแอลบีที่อยู่ห่างกันมากโดยใช้ วงจรบัฟเฟอร์ 3 สถานะเป็นตัวช่วยขับและกำหนดสถานะการเชื่อมโยงแทนการใช้พีเอสเอ็มใน XC4000x จะมีตัวต้านทานดึงขึ้น (pull-up resister) เพื่อรักษาเสถียรภาพของระดับสัญญาณ เมื่อเปรียบเทียบความล่าช้าของสัญญาณที่เกิดขึ้นในการเชื่อมโยงแบบลองไลน์กับการเชื่อมโยงแบบสี่เส้นพบว่าการเชื่อมโยงแบบลองไลน์ จะมีการล่าช้าของสัญญาณมากกว่าการเชื่อมโยงแบบสี่เส้น

4.2.3.3 **การเชื่อมโยงโดยตรง** (direct interconnection)

การเชื่อมโยงชนิดนี้มีเฉพาะ XC4000x โดยมีลักษณะพิเศษคือเป็นการเชื่อมโยงที่ไม่สามารถโปรแกรมใหม่ได้และเชื่อมต่อไอโอบีกับซีแอลบี หรือซีแอลบีกับซีแอลบีที่อยู่ใกล้กัน เท่านั้นดังแสดงรายละเอียดในภาพประกอบที่ 4.9

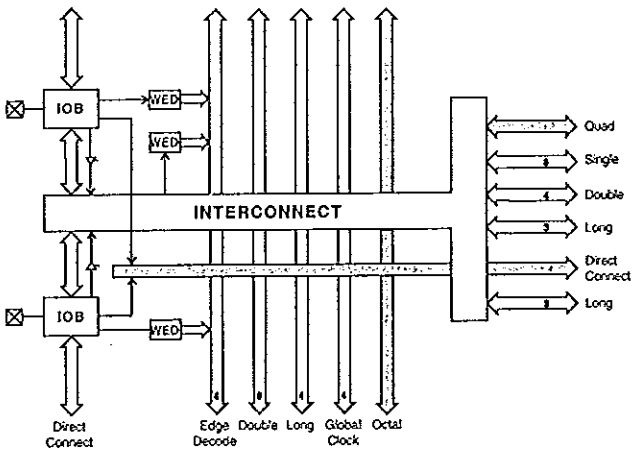


ภาพประกอบที่ 4.9 การเชื่อมโยงโดยตรง

(ที่มา : Xilinx,1999)

4.2.3.4 **การเชื่อมโยงวงแหวนแบบเวอร์ซา (versaRing) และการเชื่อมโยงแบบแปดเส้น (octal line)**

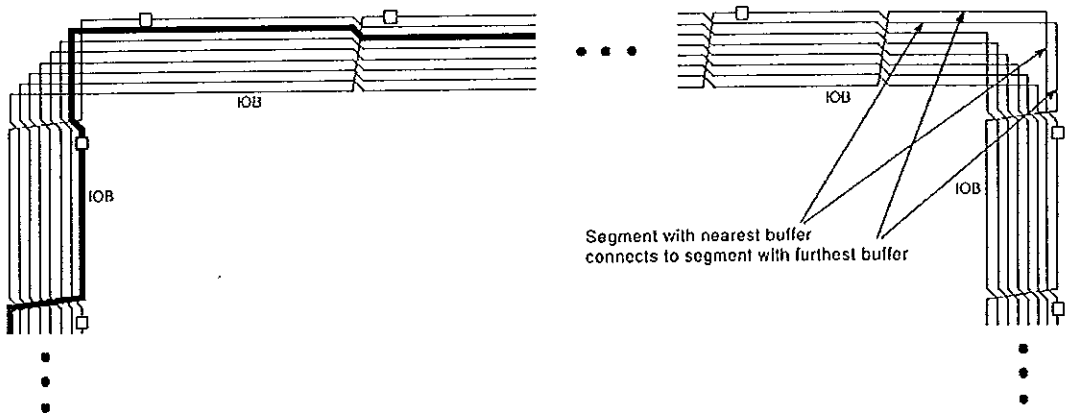
เป็นการเชื่อมโยงที่มีเฉพาะไอโอบีเท่านั้น เนื่องจากการวางตัวของไอโอบีมีข้อจำกัดให้วางตัวได้เฉพาะขอบของตัวถังและจะมีไอโอบีเพียง 2 ชุดเท่านั้นที่จะเชื่อมโยงกับซีแอลบี 1 ชุดจึงทำให้เกิดอุปสรรคในการกำหนดตำแหน่งซีแอลบีการเชื่อมโยงด้วยการเชื่อมโยงวงแหวนแบบเวอร์ซาจะเป็นการเชื่อมโยงไอโอบีแต่ละชุดเข้าด้วยกัน เพื่อให้เกิดความยืดหยุ่นในการสลับตำแหน่งขาของการออกแบบได้ ภาพประกอบที่ 4.10 แสดงการเชื่อมโยงด้วยการเชื่อมโยง วงแหวนแบบเวอร์ซาของไอโอบี 2 ชุดซึ่งมีลักษณะคล้ายกับการเชื่อมโยงของซีแอลบี



ภาพประกอบที่ 4.10 การเชื่อมโยงวงแหวนแบบเวอร์ซา

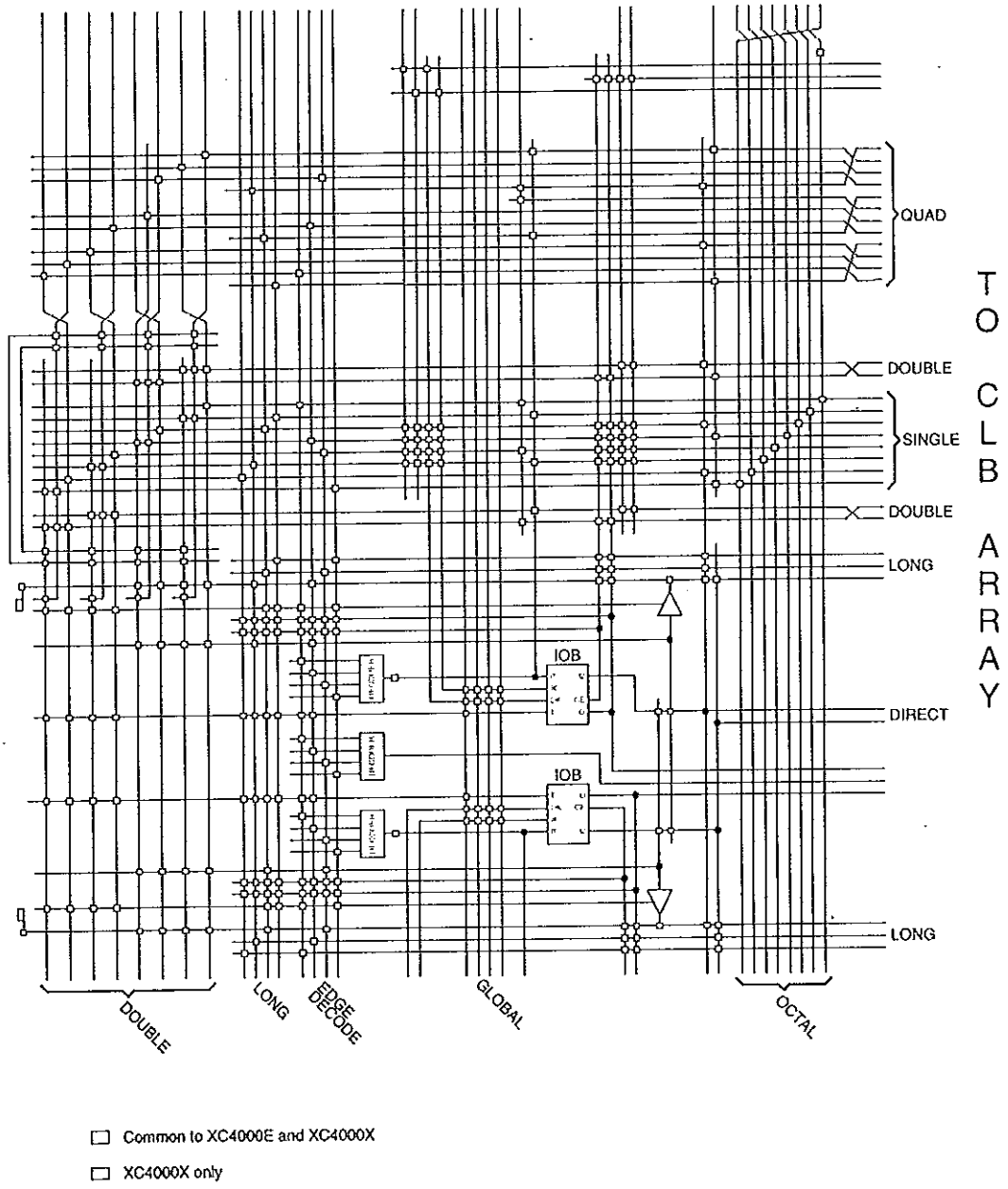
(ที่มา : Xilinx,1999)

สิ่งที่แตกต่างกับการเชื่อมโยงซีแอลบีคือการเชื่อมโยงแบบแปดเส้นซึ่งจะแบ่งไอโอบีออกเป็นส่วนๆตามการต่อกับซีแอลบีกำหนดให้ 1 ส่วนประกอบ 8 ซีแอลบีหรือไอโอบี16 ชุด ดังนั้นการเชื่อมโยงแบบแปดเส้นจะเป็นการเชื่อมโยงไอโอบีภายในแต่ละส่วนและจะเชื่อมโยงไอโอบีห่างกัน 8 ชุดดังแสดงในภาพประกอบที่ 4.11



ภาพประกอบที่ 4.11 การเชื่อมโยงแบบแปดเส้น
(ที่มา : Xilinx,1999)

ในภาพประกอบที่ 4.12 จะแสดงแผนผังทางไฟฟ้าของไอโอบี 2 ชุดกับซีแอลบี 1 ชุด ซึ่งจะมีการเชื่อมโยงที่คล้ายกับการเชื่อมโยงระหว่างซีแอลบี แต่จะเพิ่มเติมสัญญาณการตรวจขอบขาของสัญญาณ (edge decoder) เพื่อใช้กำหนดคุณลักษณะของการรับส่งข้อมูลผ่านขาสัญญาณภายนอก

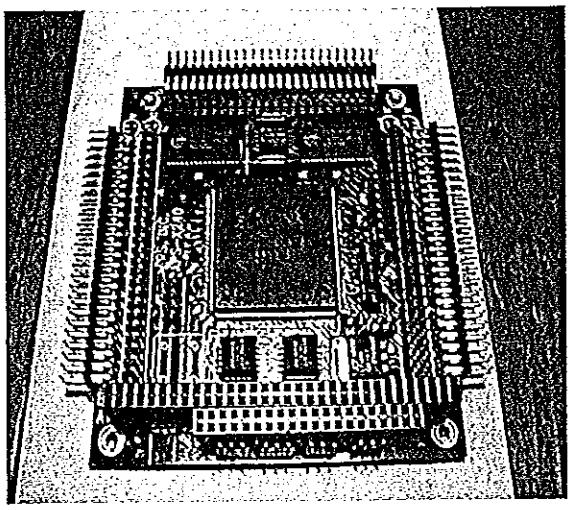


ภาพประกอบที่ 4.12 แผนผังการเชื่อมโยงไอโอบี

(ที่มา : Xilinx,1999)

4.3 การเลือกวงจรฮาร์ดแวร์เพื่อการสร้างต้นแบบ

ในงานวิจัยนี้เลือกใช้เอฟพีจีเอรุ่น XC4062XLA09-HQ240 เป็นวงจรรวมต้นแบบประกอบด้วย 5,472 ตรรกะเซลล์หรือ 2,304 ซีแอลบี เทียบเท่าวงจรดิจิตอลที่ประกอบด้วยเกต 62,000 ตัว มีจำนวนขาสัญญาณและขาควบคุมรวม 240ขา ดังแสดงในภาพประกอบที่ 4.13



ภาพประกอบที่ 4.13 วงจรทดลองต้นแบบใช้เอฟพีจีเอรุ่น XC4062XLA09-HQ240

4.4 สรุป

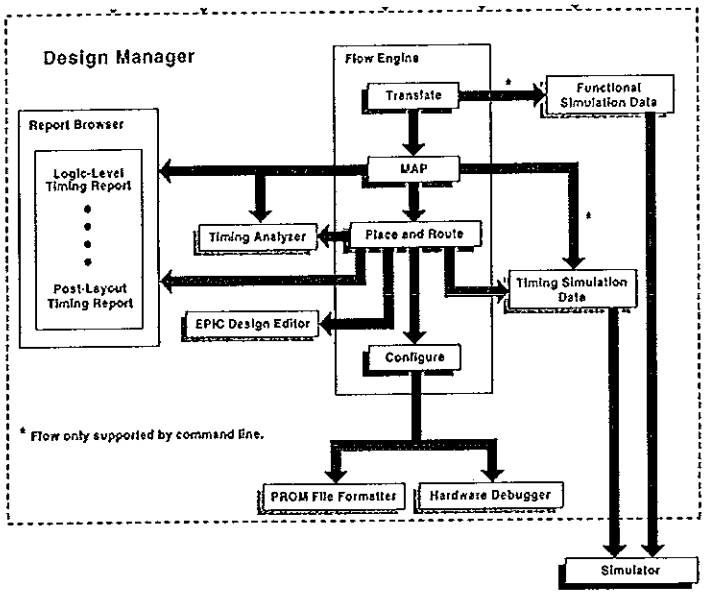
จากการศึกษาสถาปัตยกรรมฮาร์ดแวร์และโครงสร้างภายในของวงจรรวมต้นแบบที่เป็นเอฟพีจีเอ จะเป็นแนวทางที่สำคัญในการเลือกวิธีการออกแบบวงจรดิจิตอล ให้มีความเหมาะสมกับวงจรรวมต้นแบบ อันจะนำไปสู่การออกแบบที่มีประสิทธิภาพในลำดับต่อไป

บทที่ 5

การออกแบบวงจรดิจิตอล

5.1 บทนำ

การออกแบบวงจรดิจิตอลสำหรับงานวิจัยนี้จะใช้ภาษาบรรยายฮาร์ดแวร์ (hardware description langauge -HDL) ร่วมกับการออกแบบด้วยแผนผังทางไฟฟ้า (schematic capture) แล้วทำการสังเคราะห์วงจร ทดสอบและทวนสอบการทำงานภายใต้โปรแกรม Foundation ตามคำแนะนำของผู้ผลิต ก่อนทดสอบจริงด้วยระดับสัญญาณไฟฟ้าบนเอฟพีจีเอ ดังแสดงในภาพประกอบที่ 5.1



ภาพประกอบที่ 5.1 ขั้นตอนการออกแบบสำหรับเอฟพีจีเอ
(ที่มา : Xilinx,1999)

5.2 การออกแบบด้วยภาษาบรรยาย

ปัจจุบันภาษาบรรยายถูกใช้เป็นเครื่องมือในการออกแบบวงจรดิจิตอลเนื่องจากสามารถใช้ได้ง่าย สะดวกและแปลงเป็นผลลัพธ์ได้ใกล้เคียงกับวงจรจริง จึงทำให้มีความรวดเร็ว

ในการพัฒนางานทางด้านนี้ ตัวอย่างเช่นภาษาไอเอสพีเอส, ภาษาเวอริล็อก, ภาษาเอลลา และภาษาวีเอชดีแอล เป็นต้น สำหรับงานวิจัยนี้เลือกใช้การบรรยายวงจรด้วยภาษาวีเอชดีแอล (VHSIC Hardware Description Language - VHDL) ในรุ่นที่ประกาศเป็นมาตรฐาน IEEE1993 เนื่องจากสามารถงานร่วมกับโปรแกรม Foundation ได้เป็นอย่างดี

แบบแผนโครงสร้างทางภาษาวีเอชดีแอลที่ใช้ในงานวิจัยนี้จะประกอบด้วย 2 ลักษณะคือ การบรรยายพฤติกรรม (behavior description) ซึ่งเป็นการอธิบายถึงพฤติกรรมของวงจรหรือแต่ละชิ้นส่วนตามคุณลักษณะที่ต้องการพีชคณิตแบบบูลีน และอีกแบบแผนหนึ่งคือการบรรยายโครงสร้างเป็นการแสดงความสัมพันธ์การเชื่อมต่อระหว่างชิ้นส่วนต่างๆที่ประกอบรวมกันเป็นโครงสร้างที่ซับซ้อน แบบแผนทั้งสองนี้สนับสนุนการออกแบบจากบนลงล่าง (top-down design) ได้เป็นอย่างดี รวมถึงการสนับสนุนการออกแบบชนิดโครงข่าย (hierarchy) ด้วย

และจากโครงสร้างของรหัสลับไอดีอีเอในภาพประกอบที่ 2.1 ของบทที่ 2 ทำให้จำแนกวงจรดิจิตอลที่จะออกแบบได้เป็นวงจรย่อยๆได้ดังนี้

5.2.1 วงจรมอดุโลของผลบวกขนาด 16 บิต และวงจรหาค่าผกผันมอดุโลของผลบวก

วงจรในส่วนนี้ผู้วิจัยได้ทำการศึกษาทฤษฎีวงจรบวกได้แก่

- วงจรบวกแบบฟลูแอดเดอร์และส่วนเติมเต็มของสอง (full adder and 2's complementary) ดังแสดงความสัมพันธ์ด้วยพีชคณิตบูลีนในสมการที่ 5.1 (I.Koren,1993:c5-1)

$$
\begin{array}{l|l}
g = ab \quad \text{(generate)} & c^0 = ab \\
p = a \oplus b \quad \text{(propagate)} & c^1 = a + b \\
s = a \oplus b \oplus c_{in} = p \oplus c_{in} & \\
c_{out} = ab + ac_{in} + bc_{in} = ab + (a \oplus b)c_{in} & \\
\quad = g + pc_{in} = \overline{p}g + pc_{in} = \overline{p}a + pc_{in} & \\
\quad = \overline{c}_{in}c^0 + c_{in}c^1 &
\end{array}
\qquad (5.1)
$$

- วงจรบวกแบบแคร์รีลุค-อเฮดและส่วนเติมเต็มของสอง (carry-lookahead adder and 2's complementary) ดังแสดงความสัมพันธ์ด้วยพีชคณิตบูลีนในสมการที่ 5.2 (I.Koren,1993: c5-2)

$$
\begin{aligned}
c_0 &= c'_0 \\
c_1 &= g_0 + p_0 c'_0 \\
c_2 &= g_1 + p_1 g_0 + p_1 p_0 c'_0 \\
c_3 &= g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c'_0 \\
g'_3 &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 \\
p'_3 &= p_3 p_2 p_1 p_0
\end{aligned}
\qquad
\begin{aligned}
&\text{preprocessing}: g_i = a_i b_i \,,\; p_i = a_i \oplus b_i \\
&\text{postprocessing}: s_i = p_i \oplus c_i
\end{aligned}
\qquad (5.2)
$$

- ฟังก์ชันการบวกตามค่าโดยปริยายของวีเอชดีแอล
- วงจรบวกแบบแผนผังไฟฟ้าตามข้อแนะนำของบริษัท Xilinx

เมื่อทำการสร้างเป็นวงจรบวก และทดสอบผลทางฟังก์ชันได้ผลดังแสดงในตารางที่ 5.1 และด้วยเหตุผลของการใช้ข้อกำหนดเกี่ยวกับขนาดและพื้นที่ของวงจรสำหรับการออกแบบ ในงานวิจัยนี้จึงเลือกใช้วงจรมอดุโลของผลบวกที่ได้จากฟังก์ชันการบวกตามค่าโดยปริยายของวีเอชดีแอล

ตารางที่ 5.1 การเปรียบเทียบคุณลักษณะของวงจรบวกด้วยวิธีการต่างๆ

	Full adder		*Carry-lookahead*		*VHDL Function*	*Xilinx Lib.*
	VHDL	*Schematic*	*VHDL*	*Schematic*		
CLB	14	21	12	12	9	10
CLK	ไม่มี CLK	- ไม่มี CLK	- 2 CLK	- 2 CLK	ไม่มี CLK	- ไม่มี CLK

5.2.2 วงจรวิธีปรับปรุงมอดุโลของผลคูณ

วงจรในส่วนนี้ได้พัฒนาขึ้นตามทฤษฎีบทช่วยในหัวข้อ3.1 เรื่องขั้นตอนต่ำ-สูงสำหรับการคูณ และผู้วิจัยได้ทำการศึกษาทฤษฎีวงจรคูณได้แก่

- วงจรคูณแบบอเรย์ (array multiply) ดังแสดงความสัมพันธ์ด้วยพีชคณิตบูลีนในสมการที่ 5.3

$$
\begin{array}{cccccccc}
 & & & & a_0b_3 & a_0b_2 & a_0b_1 & a_0b_0 \\
 & & & a_1b_3 & a_1b_2 & a_1b_1 & a_1b_0 & \\
 & & a_2b_3 & a_2b_2 & a_2b_1 & a_2b_0 & & \\
+ & a_3b_3 & a_3b_2 & a_3b_1 & a_3b_0 & & & \\
\hline
p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
\qquad P = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_i b_j 2^{i+j}
\qquad (5.3)
$$

- วงจรคูณของ Baught & Wooley (Baught & Wooley multiply) ดังแสดงความสัมพันธ์ด้วยพีชคณิตบูลีนในสมการที่ 5.4 (J.Pihl and J.E.Aas, 1996)

$$
\begin{array}{cccccccccc}
 & & & & & \overline{a}_0b_3 & a_0b_2 & a_0b_1 & a_0b_0 & \\
 & & & & \overline{a}_1b_3 & a_1b_2 & a_1b_1 & a_1b_0 & & \\
 & & & \overline{a}_2b_3 & a_2b_2 & a_2b_1 & a_2b_0 & & & \\
 & & a_3b_3 & a_3\overline{b}_2 & a_3\overline{b}_1 & a_3\overline{b}_0 & & & & \\
 & & \overline{a}_3 & & & a_3 & & & & \\
+ & 1 & \overline{b}_3 & & & b_3 & & & & \\
\hline
 & p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0 &
\end{array}
$$

1 neg. bit : $-a+b+c_{in} = 2c_{out} - s$

2 neg. bits : $a-b-c_{in} = -2c_{out} + s$

$$s = a \oplus b \oplus c_{in}$$

$$c_{out} = \overline{a}b + \overline{a}c_{in} + bc_{in}$$

(5.4)

- วงจรคูณวิธีปรับปรุงของ Booth และแผนผังต้นไม้ของ Wallace (modified Booth & Wallace tree multiply) ดังแสดงความสัมพันธ์ด้วยพีชคณิตบูลีนในสมการที่ 5.5 (R.Zimmermann,1999:148 -157)

$$\bullet\ \bullet\ \bullet\ \underbrace{x_7\ x_6}_{y_7y_6}\ \underbrace{x_5\ x_4}_{y_5y_4}\ \underbrace{x_3\ x_2}_{y_3y_2}\ \underbrace{x_1\ x_0}_{y_1y_0}\ (x_{-1})$$

x_i	x_{i-1}	x_{i-2}	y_i	y_{i-1}	operation	comments
0	0	0	0	0	+0	string of zeros
0	1	0	0	1	$+A$	a single 1
1	0	0	$\bar{1}$	0	$-2A$	beginning of 1's
1	1	0	0	$\bar{1}$	$-A$	beginning of 1's
0	0	1	0	1	$+A$	end of 1's
0	1	1	1	0	$+2A$	end of 1's
1	0	1	0	$\bar{1}$	$-A$	a single 0
1	1	1	0	0	+0	string of 1's

$x_{i\_1} + x_{i\_2} - 2x_i$

(5.5)

- วงจรคูณแบบแผนผังไฟฟ้าตามข้อแนะนำของบริษัท Xilinx

เมื่อทำการสร้างเป็นวงจรคูณ และทดสอบผลทางฟังก์ชันได้ผลดังแสดงในตารางที่ 5.2 จะสังเกตได้ว่าวงจรคูณแบบแผนผังไฟฟ้าตามข้อแนะนำของบริษัท Xilinx ถึงแม้จะใช้พื้นที่และมีขนาดของวงจรเล็กกว่าวิธีอื่นแต่จะต้องรอสัญญาณนาฬิกาในช่วงเวลาแฝง (latency time) มาก ซึ่งจะส่งผลต่อความเร็วในการประมวลผลของระบบ ดังนั้นผู้วิจัยจึงเลือกใช้วงจรคูณในกระบวนวิธีการออกแบบของ Baught & Wooley เป็นส่วนประมวลผลวิธีปรับปรุงมอดุโลของผลคูณ

ตารางที่ 5.2 การเปรียบเทียบคุณลักษณะของการคูณด้วยวิธีการต่าง ๆ

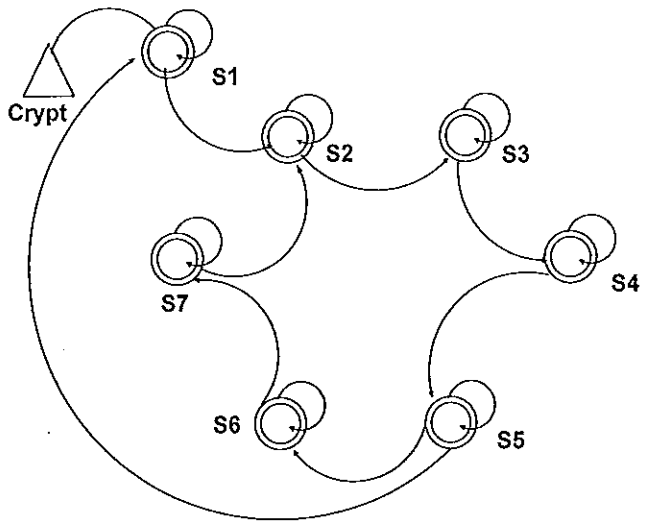
Array	*Baught & Wooley*	*Booth & Wallace*	*Xilinx Lib.*
- ใช้พื้นที่ 463 CLB - ไม่มี CLK	- ใช้พื้นที่ 248 CLB - ไม่มี CLK	- ใช้พื้นที่ 298 CLB - ใช้เวลา 4 CLK - ใช้ register latch ในแต่ละขั้นของ Wallace tree	-ใช้พื้นที่ 234 CLB - ใช้เวลา 5 CLK

5.2.3 วงจรเอ็กคลูซีพออร์เกต

ผู้วิจัยได้ทำการสร้างต้นแบบด้วยภาษาบรรยายเปรียบเทียบกับการสร้างด้วยผังไฟฟ้า พบว่าผลการออกแบบทั้งสองจะใช้พื้นที่ 16 CLB เท่ากัน

5.2.4 วงจรควบคุมวงรอบการทำงาน

วงจรในส่วนนี้ ผู้วิจัยใช้การออกแบบด้วยวิธีการบรรยายด้วยแผนภาพเอฟเอสเอ็ม (finite state machine – FSM) เนื่องจากแต่ละสถานะของแต่ละกระบวนการย่อยมีปฏิสัมพันธ์ที่ซับซ้อนและอาศัยการโต้ตอบในแต่ละปฏิสัมพันธ์เพื่อให้เกิดการประมวลผลที่เข้าจังหวะกันจึงเป็นการยากที่จะบรรยายพฤติกรรมของวงจร ภาพประกอบที่ 5.2 แสดงการออกแบบวงจรควบคุมวงรอบการทำงานด้วยแผนภาพเอฟเอสเอ็ม

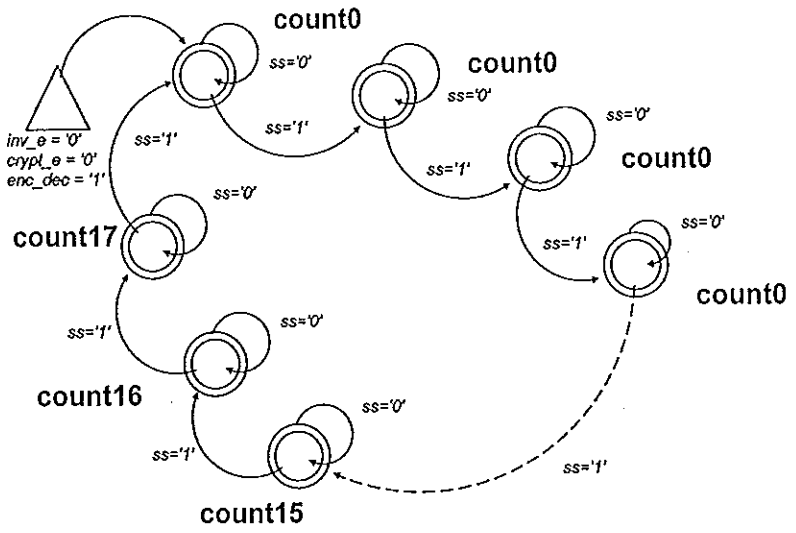


ภาพประกอบที่ 5.2 แผนภาพเอฟเอสเอ็มของวงจรควบคมวงรอบการเข้า/ถอดรหัส

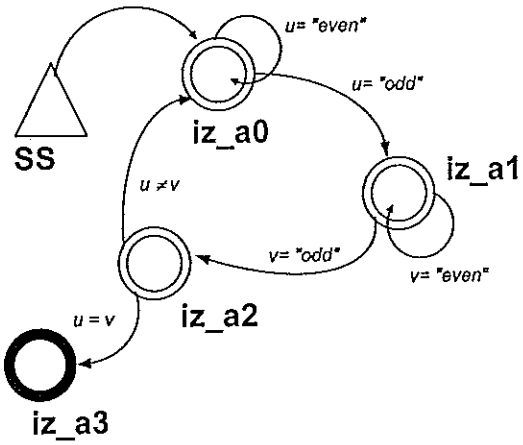
จากภาพประกอบที่ 5.2 เมื่อใช้ทฤษีการออกแบบวงจรของ Moor และ Murry (S.Sjoholm *et al.*, 1997) สำหรับปฏิสัมพันธ์ของแต่ละสถานะ โดยแต่ละสถานะใช้รหัสเกรย์ 3 บิตเพื่อกำหนด 7 สถานะสำหรับการประมวลผล 9 รอบ และใช้การเหลื่อมเวลาในแต่ละสถานะเป็นช่วงรอการประมวลผล การเปลี่ยนสถานะจะต้องรอการเปลี่ยนค่าของสัญญาณนาฬิกาเป็นขอบขาขึ้นและจะเริ่มทำงานในสถานะนั้นๆก็ต่อเมื่อสัญญาณนาฬิกาเป็น "1" สถานะ s1 จะรับรหัสกุญแจ โดยที่สถานะ s2 จะประมวลผลรหัสกุญแจ 1 และ 2, สถานะ s 4 จะประมวลกุญแจ 3 และ 4, สถานะ s6 จะประมวลกุญแจ 5 และ 6, สถานะ s3, s5 และ s7 จะประมวลผลตามตัวดำเนินการเอ็กคลูซีฟออร์ของผลลัพธ์จากสถานะ s2, s4 และ s6 ตามลำดับ สถานะ s2 ถึง s8 จะทำการวนจนครบ 8 ครั้ง ส่วนรอบที่ 9 จะประมวลผลจาก s2 ถึง s5 แล้วจึงกระโดดกลับมาทำที่ s1 ใหม่ วงรอบของการทำงานจะดำเนินไปตามลำดับขั้นตราบเท่าที่สัญญาณ crypt ยังคงเป็นตรรกะต่ำ (ลอจิกเป็นค่า "0") และสามารถขัดจังหวะการทำงานเพื่อเริ่มต้นการที่สถานะ s1 ได้ใหม่เมื่อสัญญาณ crypt เป็นตรรกะสูง (ลอจิกเป็นค่า "1")

5.2.5 วงจรหาค่าผกผันของกุญแจย่อยในการถอดรหัส

วงจรในส่วนนี้มีความซับซ้อนมากกว่าวงจรควบคุมวงรอบเนื่องจากประกอบด้วยเอฟเอสเอ็ม 2 ชุดทำงานซ้อนกัน ชุดหลักเป็นการควบคุมการกระจายกุญแจเพื่อคำนวณค่าผกผันวิธีปรับปรุงมอดุโลของผลคูณ 18 กุญแจ และชุดรองเป็นการคำนวณค่าผกผันมอดุโลของผลคูณด้วยขั้นตอนทวินามวิธีของยุคลิด ดังแสดงในภาพประกอบที่ 5.3 และ 5.4



ภาพประกอบที่ 5.3 แผนภาพเอฟเอสเอ็มของการกระจายกุญแจย่อย



ภาพประกอบที่ 5.4 แผนภาพเอฟเอสเอ็มของการหาค่าผกผันด้วยขั้นตอนทวินามวิธีของยุคลิด

จากภาพประกอบที่ 5.3 สามารถอธิบายการทำงานของวงจรในส่วนนี้ได้ว่า เมื่อสัญญาณควบคุม inv_e , crypt_e เป็นตรรกะต่ำ และสัญญาณ enc_dec เป็นตรรกะสูง วงจรในส่วนนี้จะส่งค่าของกุญแจย่อยไปประมวลผลด้วยขั้นตอนทวินามวิธีของยุคลิด ดังแสดงในภาพประกอบที่ 5.4 ในเวลาเดียวกันนั้น สัญญาณ ss จะถูกกำหนดให้เป็นค่าตรรกะต่ำเมื่อการคำนวณค่าผกผันด้วยขั้นตอนทวินามวิธีของยุคลิด และสัญญาณ ss จะถูกกำหนดให้เป็นค่าตรรกะสูงเมื่อค่าผกผันมอดุโลผลคูณในแต่ละกุญแจย่อยถูกคำนวณเสร็จเรียบร้อยแล้ว และยังผลให้เกิดการเปลี่ยนสถานะในการกระจายกุญแจย่อยต่อไป

สำหรับการหาค่าผกผันมอดุโลของผลบวกส่งส่วนเติมเต็มของกุญแจย่อยนั้นๆ ไปยังวงจรมอดุโลผลบวกจะกระทำเมื่อการคำนวณค่าผกผันมอดุโลของผลคณในกุญแจย่อยทุกตัวถูกคำนวณเสร็จเรียบร้อยแล้ว

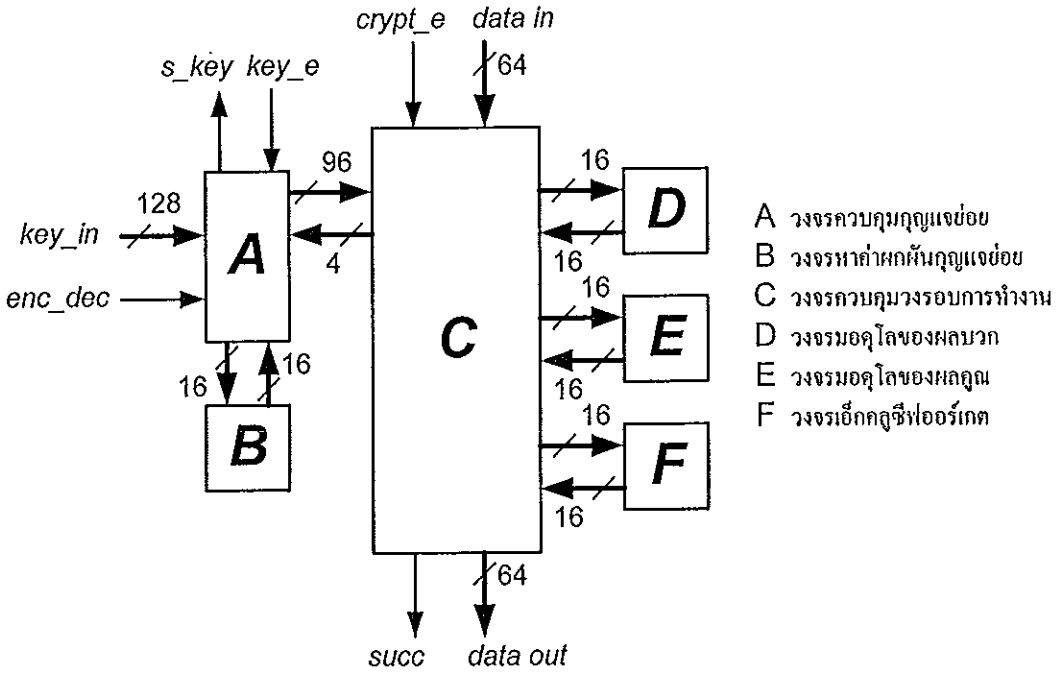
5.3 การออกแบบด้วยแผนผังไฟฟ้า

วงจรในส่วนนี้ ผู้วิจัยได้เลือกใช้อุปกรณ์จากคลังสำเร็จของโปรแกรม Foundation ตามคำแนะนำของผู้ผลิต และบางส่วนได้จากการสร้างใหม่ภายใต้โปรแกรมสนับสนุนของ Foundation ได้แก่โปรแกรม CoreGEN และโปรแกรม LogicBLOX จากนั้นทำการเชื่อมโยงสายสัญญาณกับวงจรต่างๆ ที่สร้างจากภาษาบรรยาย วงจรในส่วนนี้ได้แก่

- วงจรกำหนดค่าและตรวจสอบเงื่อนไขเริ่มต้นสำหรับเอฟพีจีเอ (startup & boundary scan)
- วงจรการเชื่อมโยงขาภายนอก (pad)

5.4 การสังเคราะห์วงจร

วงจรในส่วนต่างๆที่ได้ออกแบบมาจะถูกแปลงและสังเคราะห์ให้เป็นแผนผังไฟฟ้า จากนั้นจึงจะทำการเชื่อมโยงวงจรย่อยทั้งหมดเข้าด้วยกันตามแผนผังในภาพประกอบที่ 5.5



ภาพประกอบที่ 5.5 แผนผังวงจรต้นแบบสำหรับรหัสลับไอดีอีเอ

5.5 สรุป

จากผลการออกแบบวงจรดิจิตอล ผู้วิจัยได้เลือกที่จะออกแบบ 2 วิธีคือการออกแบบด้วยภาษาบรรยายฮาร์ดแวร์ซึ่งได้แก่การออกแบบ วงจรมอดุโลของผลบวก วงจรวิธีปรับปรุงมอดุโลของผลคูณ วงจรเอ็กคลูซีฟออร์เกต วงจรควบคุมวงรอบการทำงานและวงจรหาค่าผกผันมอดุโล

ของผลคูณ ส่วนออกแบบด้วยแผนผังทางไฟฟ้านั้นจะใช้สำหรับวงจรกำหนดค่าเริ่มต้นสำหรับเอฟพีจีเอและการเชื่อมโยงขาภายนอก

ในการทดลองออกแบบนั้นผู้วิจัยเลือกใช้ฟังก์ชันตามค่าโดยปริยายของภาษาวีเอชดีแอลสำหรับวงจรบวก และวงจรเอ็กคลูซีฟออร์เกต และเลือกออกแบบวงจรวิธีปรับปรุงมอดุโลของผลคูณด้วยวิธีการของ Baught และ Wooley สำหรับวงจรควบคุมวงรอบการทำงานและวงจรหาค่าผกผันมอดุโลของผลคูณใช้การออกแบบด้วยเอฟเอสเอ็มก่อนแล้วจึงแปลเป็นภาษาบรรยายเพื่อใช้เกิดความสะดวกในการออกแบบ โดยทั้งหมดนี้ได้แสดงโปรแกรมรหัสต้นฉบับภาษาวีเอชดีแอลไว้ในภาคผนวก ค1

บทที่ 6

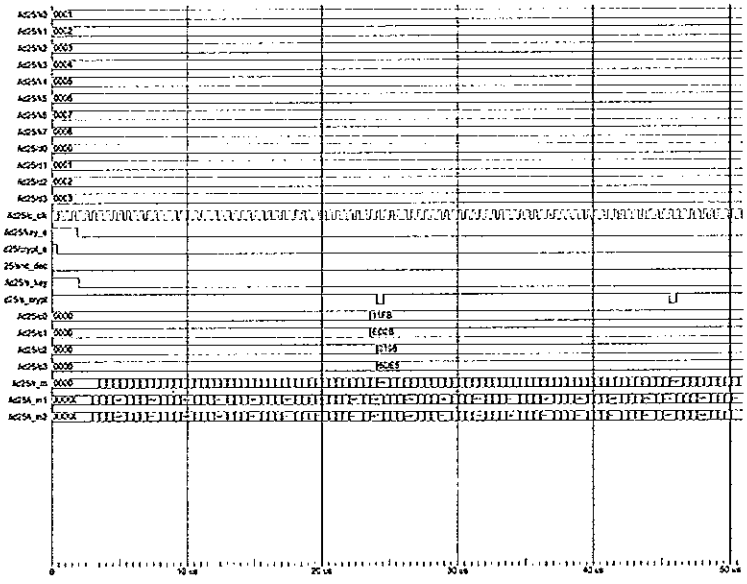
ผลการทดลอง

6.1 บทนำ

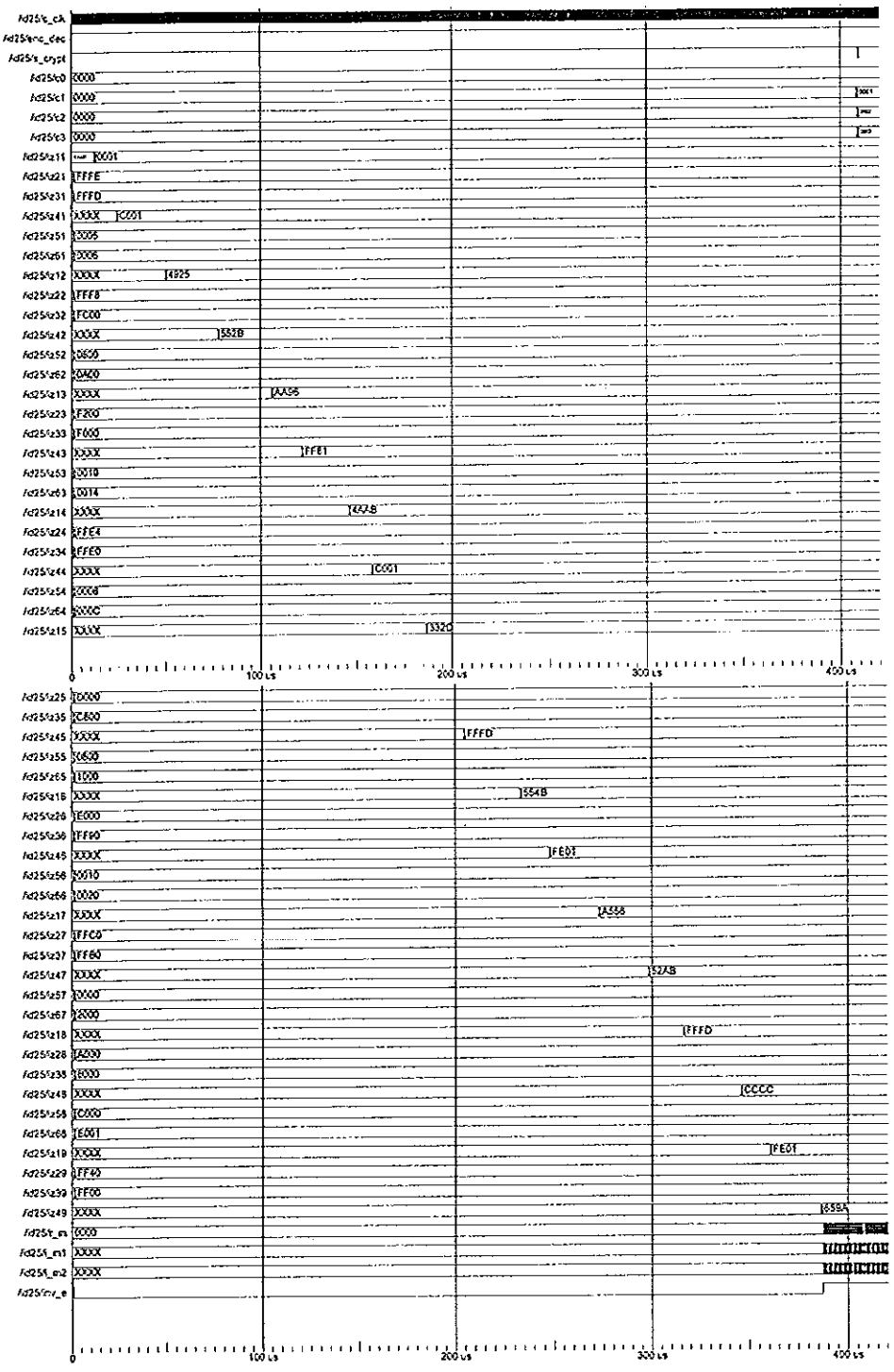
จากผลการออกแบบวงจรดิจิตอล วงจรทั้งหมดถูกสังเคราะห์เพื่อทดสอบความถูกต้องและค่าคงตัวทางเวลาที่ใช้จริง ทั้งนี้จะใช้เงื่อนไขการบังคับทางเวลา (timing constraint) เพื่อหาเส้นทางวิกฤต (critical path) จากสัญญาณผลลัพธ์และสัญญาณขาเข้า ผลดังกล่าวจะเป็นตัวกำหนดความเร็วสูงสุดที่แต่ละหน่วยดำเนินการจะปฏิบัติงานสำเร็จ

6.2 การทดสอบฟังก์ชันเข้าและถอดรหัส

วงจรในส่วนนี้ประกอบด้วย วงจรควบคุมวงรอบการทำงาน วงจรควบคุมกุญแจย่อย วงจรประมวลผลวิธีปรับปรุงมอดุโลของผลคูณ วงจรมอดุโลของผลบวก และวงจรเอ็กซ์คลูซีฟออร์เกต ดังแสดงผลการทดสอบภาพประกอบที่ 6.1 และ 6.2 โดยใช้โปรแกรม ModelSIM แสดงแผนผังสมบูรณ์ไว้ในภาคผนวก และใช้สัญญาณควบคุมดังแสดงในตารางที่ 6.1 เปรียบเทียบกับค่าที่คำนวณได้ด้วยโปรแกรมภาษาซีในตารางที่ 6.2 และ 6.3 เมื่อ K_r^i เป็นกุญแจย่อยที่ i รอบที่ r และ X_r^i เป็นผลลัพธ์การคำนวณย่อยที่ i รอบที่ r



ภาพประกอบที่ 6.1 การจำลองสัญญาณด้วยโปรแกรม ModelSIM สำหรับการเข้ารหัส



ภาพประกอบที่ 6.2 การจำลองสัญญาณด้วยโปรแกรม ModelSIM สำหรับการถอดรหัส

ตารางที่ 6.1 สัญญาณควบคุมวงจรเข้าและถอดรหัส

ชื่อสัญญาณ	ทิศทาง	หน้าที่
K7[15:0] – K0[15:0]	เข้า	รหัสกุญแจ 16 บิตจำนวน 8 ชุดรวม 128 บิต
D3[15:0] – D0[15:0]	เข้า	ข้อมูลดิบก่อนเข้า/ถอดรหัส 16 บิตจำนวน 4 ชุดรวม 64 บิต
C_CLK	เข้า	สัญญาณนาฬิกา
KEY_E	เข้า	สัญญาณควบคุมการส่งรหัสกุญแจ
CRYPT_E	เข้า	สัญญาณควบคุมการเริ่มต้นวงรอบการทำงาน
ENC_DEC	เข้า	สัญญาณควบคุมเลือกแบบการเข้า หรือ ถอดรหัส
S_KEY	ออก	สัญญาณตอบรับการแปลงรหัสกุญแจย่อยเสร็จสมบูรณ์
S_CRYPT	ออก	สัญญาณตอบรับการผลการเข้า/ถอดเสร็จสมบูรณ์
C3[15:0] – C0[15:0]	ออก	ข้อมูลผลลัพธ์เข้า/ถอดรหัส 16 บิตจำนวน 4 ชุดรวม 64 บิต
Z11[15:0] – Z49[15:0]	-	ข้อมูลกุญแจย่อยสำหรับการเข้ารหัส 52 ชุด
IZ11[15:0] – IZ49[15:0]	-	ข้อมูลกุญแจย่อยสำหรับการถอดรหัส 52 ชุด
I_M1 , I_M2	-	ข้อมูลเข้าสำหรับตัวดำเนินการวิธีปรับปรุงมอดุโลของผลคูณ
R_M	-	ผลลัพธ์ที่ได้จากตัวดำเนินการวิธีปรับปรุงมอดุโลของผลคูณ

ตารางที่ 6.2 ผลที่คำนวณจากโปรแกรมสำหรับการเข้ารหัส

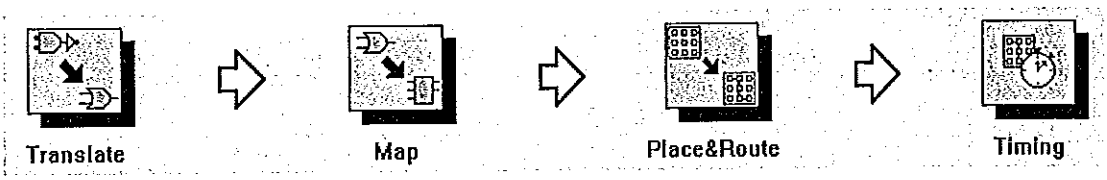
รหัสกุญแจ $K = (1, 2, 3, 4, 5, 6, 7, 8)$							ข้อมูลดิบ $X = (0, 1, 2, 3)$			
รอบที่	K_r^1	K_r^2	K_r^3	K_r^4	K_r^5	K_r^6	X_r^1	X_r^2	X_r^3	X_r^4
1	0001	0002	0003	0004	0005	0006	00f0	00f5	010a	0105
2	0007	0008	0400	0600	0800	0a00	222f	21b5	f45e	e959
3	0c00	0e00	1000	0200	0010	0014	0f86	39be	8ee8	1173
4	0018	001c	0020	0004	0008	000c	57df	ac58	c65b	ba4d
5	2800	3000	3800	4000	0800	1000	8e81	ba9c	f77f	3a4a
6	1800	2000	0070	0080	0010	0020	6942	9409	e21b	1c64
7	0030	0040	0050	0060	0000	2000	99d0	c7f6	5331	620e
8	4000	6000	8000	a000	c000	e001	0a24	0098	ec6b	4925
9	0080	00c0	0100	0140	—	—	11fb	ed2b	0198	6de5

ตารางที่ 6.3 ผลที่คำนวณจากโปรแกรมสำหรับการถอดรหัส

รหัสกุญแจ K = (1, 2, 3, 4, 5, 6, 7, 8)							ข้อมูลดิบ X = (11fb, ed2b, 0198, 6de5)			
รอบที่	K_r^1	K_r^2	K_r^3	K_r^4	K_r^5	K_r^6	X_r^1	X_r^2	X_r^3	X_r^4
1	fe01	ff40	ff00	659a	c000	e001	d98d	d331	27f6	82b8
2	fffd	8000	a000	cccc	0000	2000	bc4d	e26b	9449	a576
3	a556	ffb0	ffc0	52ab	0010	0020	0aa4	f7ef	da9c	24e3
4	554b	ff90	e000	fe01	0800	1000	ca46	fe5b	dc58	116d
5	332d	c800	d000	fffd	0008	000c	748f	8f08	39da	45cc
6	4aab	ffe0	ffe4	c001	0010	0014	3266	045e	2fb5	b02e
7	aa96	f000	f200	ff81	0800	0a00	0690	050a	00fd	1dfa
8	4925	fc00	fff8	552b	0005	0006	0000	0005	0003	000c
9	0001	fffe	fffd	c001	—	—	0000	0001	0002	0003

6.3 การทดสอบการสังเคราะห์วงจร

เมื่อทำการจำลองสัญญาณจนได้ผลที่ถูกต้องแล้ว จึงทำการสังเคราะห์วงจรตามเทคโนโลยีของ XC4062XLA09-HQ240 โดยใช้โปรแกรม Foundation ซึ่งแสดงลำดับขั้นตอนการสังเคราะห์วงจรในภาพประกอบที่ 6.3 ผลการสังเคราะห์วงจรนี้จะประเมินเวลาประวิงและเส้นทางวิกฤตสำหรับการแก้ไขและปรับปรุงการออกแบบให้เป็นไปตามเงื่อนไขด้านขนาดของวงจรและเวลาในการประมวล จากผลการสังเคราะห์จะใช้ทรัพยากรไป 1520 ซีแอลบี และสามารถประมวลผลได้เมื่อให้สัญญาณนาฬิกามีคาบไม่น้อยกว่า 130 นาโนวินาที

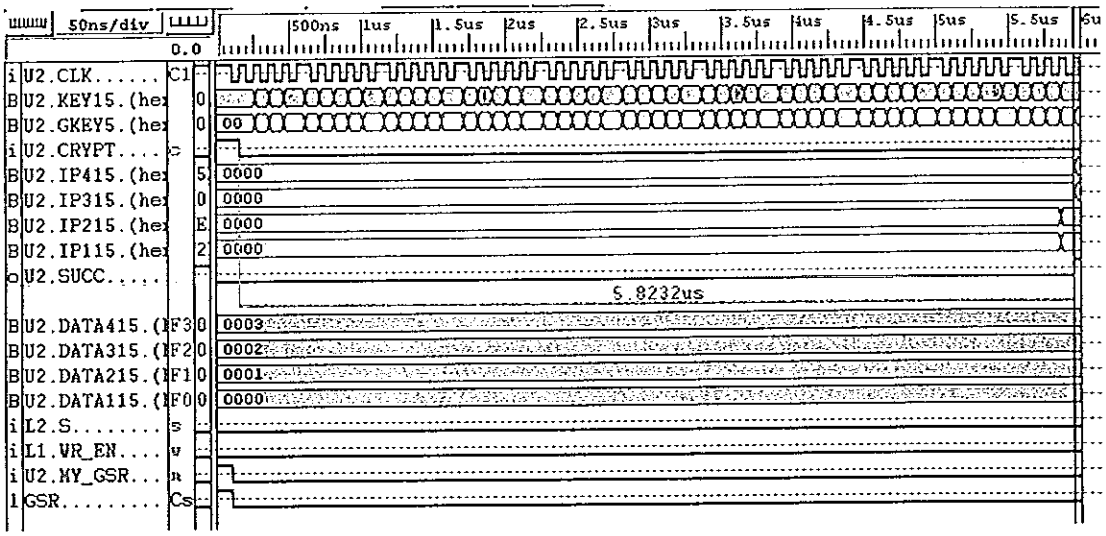


ภาพประกอบที่ 6.3 ขั้นตอนการสังเคราะห์วงจร

6.4 การจำลองผลการสังเคราะห์วงจร

6.4.1 การหาอัตราการส่งผ่านข้อมูลสูงสุด

วงจรที่ผ่านการสังเคราะห์จะถูกจำลองสัญญาณใหม่เพื่อทดสอบความสมบูรณ์ทางฟังก์ชันภายใต้เงื่อนไขการประวิงเวลาและเส้นทางวิกฤต ในการทดสอบนี้จะกำหนดรหัสกุญแจเป็นค่าคงที่แต่เปลี่ยนสัญญาณข้อมูลให้มีความแตกต่างกัน รวมถึงการเปลี่ยนความถี่ของสัญญาณนาฬิกาเพื่อหาค่าความถี่สูงสุดที่ระบบยังคงประมวลผลได้ถูกต้องอันจะนำไปสู่การหาค่าอัตราการส่งผ่านข้อมูลสูงสุด จากผลการทดสอบสามารถเข้า/ถอดรหัสอัตราการส่งผ่านข้อมูลสูงสุดเป็น 13.3 เมกกะบิตต่อวินาทีด้วยใช้สัญญาณนาฬิกาจากภายนอกความถี่ 9.1 เมกกะเฮิรตซ์ (สัญญาณคาบเป็น 110 นาโนวินาที) ซึ่งสูงกว่าค่าที่รายงานในผลการสังเคราะห์วงจรและแสดงผลการทดสอบในภาพประกอบที่ 6.4

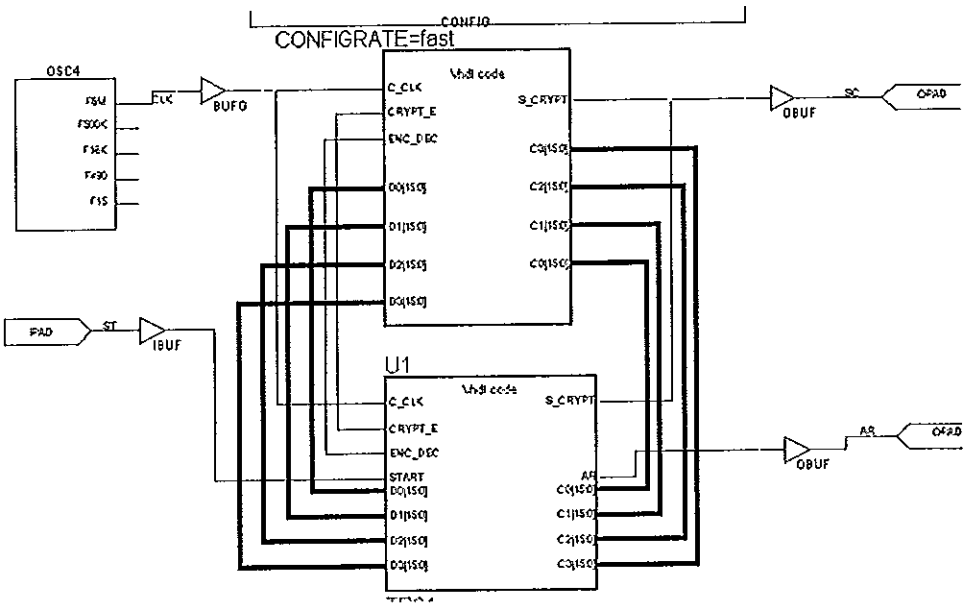


ภาพประกอบที่ 6.4 ผลการทดสอบเพื่อหาอัตราการส่งผ่านข้อมูลสูงสุด

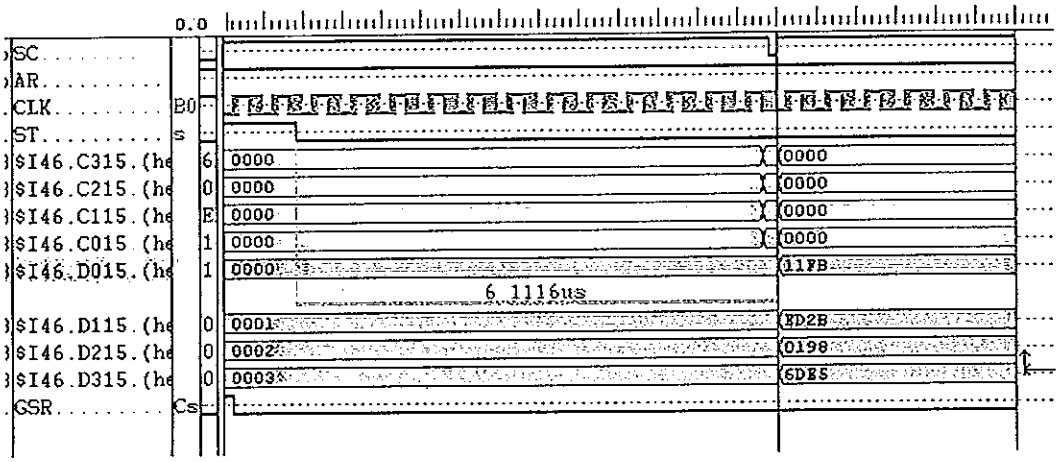
6.4.2 การทดสอบโดยใช้สัญญาณนาฬิกาภายใน

การทดสอบเบื้องต้นได้ทำการสร้างชุดทดสอบภายใน (loopback testbench) ดังแสดงในภาพประกอบที่ 6.5 และใช้สัญญาณนาฬิกาจากฐานเวลาภายใน XC4000 ความถี่ 8 เมกกะเฮิรตซ์ (คาบ 125 นาโนวินาที) แล้วประเมินอัตราการส่งผ่านข้อมูลจากสัญญาณการเข้า-ถอดรหัสเสร็จสิ้นในแต่ละรอบ(SC) และการตรวจสอบความถูกต้องของรหัส (AR) จาก

ผลการทดสอบจะได้อัตราการส่งผ่านข้อมูลเข้ารหัสเป็น 10.5 เมกกะบิตต่อวินาที(Mbit/s) ดังแสดงในภาพประกอบที่ 6.6



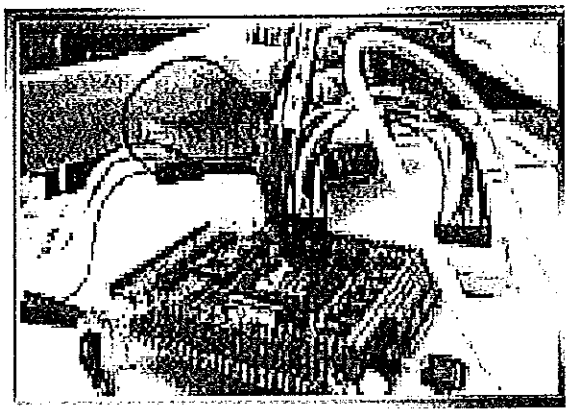
ภาพประกอบที่ 6.5 แผนผังทางไฟฟ้าการทดสอบภายใน



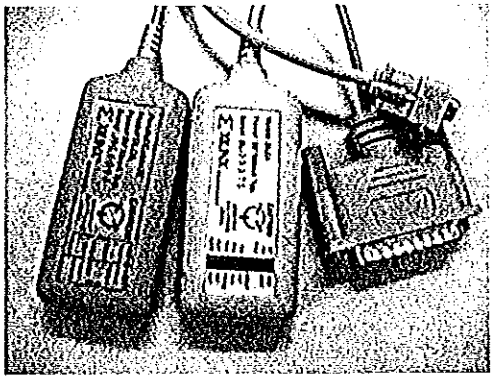
ภาพประกอบที่ 6.6 แผนผังทางเวลาของการทดสอบจากฐานเวลาใน XC4000

6.5 การทดสอบทางไฟฟ้า

ในการทดสอบนี้จะทำการโปรแกรมวงจรที่ออกแบบไว้ผ่านอุปกรณ์เชื่อมต่อ Xchecker ผ่านทางอนุกรม ดังแสดงในภาพประกอบที่ 6.7 และ 6.8 จากนั้นจึงทำการวัดสัญญาณ AR ซึ่งเป็นสัญญาณยืนยันความถูกต้องของข้อมูลทดสอบ



ภาพประกอบที่ 6.7 การโปรแกรมเอฟพีจีเอ



ภาพประกอบที่ 6.8 อุปกรณ์เชื่อมต่อ XChecker

6.6 สรุป

จากผลการทดลอง ผู้วิจัยสามารถสังเคราะห์และโปรแกรมวงจรดิจิตอลที่ได้ออกแบบไว้ลงในวงจรรวมต้นแบบและสามารถทำงานได้ตามข้อกำหนดของระบบรหัสลับไอดีอีเอ

บทที่ 7

สรุปผลการวิจัยและข้อเสนอแนะ

7.1 บทนำ

เมื่อพิจารณาผลการวิจัยโดยรวม อันมีวัตถุประสงค์ที่สำคัญประการแรก เพื่อศึกษาออกแบบและทดสอบโปรแกรมรหัสลับไอดีอีเอให้เกิดผลอันจะนำไปสู่การอ้างอิงและสนับสนุนผลการทดสอบวงจรรวมต้นแบบ และประการที่สอง เพื่อออกแบบและสร้างวงจรรวมต้นแบบสำหรับการเข้าและถอดรหัสข้อมูล 64 บิต ด้วยรหัสกุญแจ 128 บิต แบบ 9 รอบ ตามข้อกำหนดของรหัสลับแบบไอดีอีเอ นั้น

การดำเนินการเพื่อบรรลุวัตถุประสงค์ประการแรก จำเป็นต้องศึกษาทฤษฎีจำนวน และพีชคณิตนามธรรมเพื่อให้ใจถึงความสัมพันธ์ในตัวดำเนินการต่างๆที่จะนำมาใช้ในการเข้า/ถอดรหัสลับ และทำการประยุกต์ทฤษฎีในลักษณะของโปรแกรมคอมพิวเตอร์ที่ใช้ภาษาซี เพื่อตรวจสอบและยืนยันความถูกต้อง รวมไปถึงการนำผลไปใช้เทียบเคียงกับการออกแบบวงจรรวมในลำดับต่อไป และได้มีการพัฒนาเพื่อเพิ่มประสิทธิโดยนำทฤษฎีบทสำหรับการคำนวณค่าทวินามเพื่อคำนวณจำนวนเต็มหน่วย เพื่อให้เกิดความสมบูรณ์ในการนำไปออกแบบเป็นวงจรดิจิตอลในลำดับต่อไป

สำหรับวัตถุประสงค์ประการที่สองนั้น ได้วางโครงร่างการดำเนินงานเป็น 2 ส่วน คือ การพัฒนาวงจรดิจิตอลด้วยภาษาบรรยายฮาร์ดแวร์ และการพัฒนาวงจรดิจิตอลด้วยแผนผังทางไฟฟ้า โดยทั้งสองส่วนนี้ได้ดำเนินการภายใต้โปรแกรม Foundation ซึ่งสนับสนุนการพัฒนาวงจรรวมแบบโปรแกรมซ้ำได้ซนิดเอฟพีจีเอ โดยผู้วิจัยได้ทำการทดสอบทั้งฟังก์ชันการทำงาน และผลทางเวลา เพื่อนำไปสร้างเป็นวงจรรวมต้นแบบสำหรับการประมวลผลรหัสลับไอดีอีเอ

7.2 สรุปผลการวิจัย

ผลการวิจัยการออกแบบวงจรรวมสำหรับการประมวลผลรหัสลับไอดีอีเอ ซึ่งได้ออกแบบทั้งในส่วนประมวลผลรหัสกุญแจย่อย และส่วนประมวลผลการเข้าและถอดรหัส บรรจุในวงจรรวมซนิดโปรแกรมซ้ำได้แบบเอฟพีจีเอ โดยวงจรรวมต้นแบบใช้เอฟพีจีเอเบอร์ XC4062XLA09 มีตัวถังเป็นเซอรามิกทนความร้อน (high heat dissipation - HHD) รูปทรงสี่เหลี่ยมจัตุรัสขาราบรอบตัว

(quad flat pack – QFP) 240 ขา ทำงานที่แรงดันไฟเลี้ยง 3.3 โวลต์ ผลการออกแบบจะได้วงจรรวมซึ่งมีคุณลักษณะดังแสดงในตารางที่ 7.1

ตารางที่ 7.1 คุณสมบัติของวงจรรวมต้นแบบสำหรับการประมวลผลรหัสลับไอดีอีเอ

	ใช้สัญญาณนาฬิกาภายนอก	ใช้สัญญาณนาฬิกาภายใน
ความถี่สัญญาณนาฬิกา	9.1 เมกกะเฮิรตซ์	8 เมกกะเฮิรตซ์
จำนวนทรัพยากร *	1381 ซีแอลบี (ร้อยละ 59.94)	1383 ซีแอลบี (ร้อยละ 60.03)
อัตราการส่งผ่านข้อมูล	13.3 เมกกะบิตต่อวินาที **	10.5 เมกกะบิตต่อวินาที **
ขนาดสัญญาณข้อมูลเข้า/ออก	64 บิต	64 บิต

* *ไม่รวม testbench*

** *คำนวณจากการประมวลผลโดยไม่เปลี่ยนรหัสกุญแจ*

7.3 การเปรียบเทียบคุณสมบัติกับงานวิจัยอื่น

จากผลในตารางที่ 7.1 เมื่อเปรียบเทียบกับการพัฒนาด้วยภาษาซีบนเครื่องคอมพิวเตอร์ VAX-9000 ซึ่งมีอัตราการส่งผ่านข้อมูล 355.5 กิโลบิตต่อวินาที (kbit/s) และบนเครื่องคอมพิวเตอร์ Sun SPARCstation 2 ซึ่งมีอัตราการส่งผ่านข้อมูล 400 กิโลบิตต่อวินาที (kbit/s) พบว่าระบบรหัสลับที่ได้ออกแบบในงานวิจัยนี้มีอัตราการส่งผ่านข้อมูลที่สูงกว่าหลายเท่า และเมื่อเปรียบเทียบกับการพัฒนาด้วยภาษาซีและแอสเซมบลีแบบ 4 ชุด (4-way IDEA) บนตัวประมวลผล Pentium II 450 เมกกะเฮิรตซ์ ด้วยสถาปัตยกรรม MMX (MultiMedia eXtensions) แม้ว่ามีอัตราการส่งผ่านข้อมูล 23.6 เมกกะบิตต่อวินาที (Mbit/s) แต่วงจรมีขนาดใหญ่และซับซ้อนกว่า รวมถึงการใช้สัญญาณนาฬิกาที่สูงกว่างานวิจัยนี้มาก แต่ยังให้ประสิทธิภาพต่ำกว่าการวิจัยวงจรรวมในต่างประเทศที่ใช้การออกแบบด้วยวิธีไมโครอิเล็กทรอนิกส์และใช้สถาปัตยกรรมขนาน

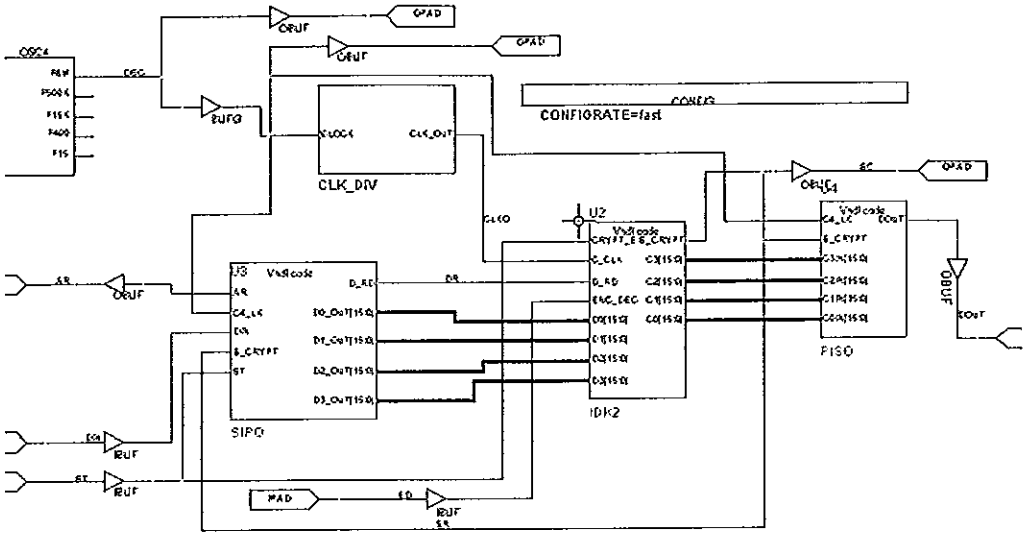
7.4 บทวิจารณ์และข้อเสนอแนะ

เนื่องจากอุปกรณ์โปรแกรมซ้ำได้แบบเอฟพีจีเอจะมีความคล่องตัวและมีความยืดหยุ่นสูงสำหรับการออกแบบวงจรรวมต้นแบบ แต่อุปกรณ์ที่มีความจุสูงจะมีราคาแพงมากจึงอาจทำให้ราคาต้นทุนต่อประสิทธิภาพสูงตามไปด้วย สำหรับงานวิจัยนี้สามารถลดราคาต้นทุนของวงจรรวม

ให้ต่ำลงโดยใช้เอฟพีจีเอที่มีขนาดเล็กของบริษัท Xilinx เบอร์ XC4020XLA09-HQ208 ซึ่งมีความจุประมาณ 784 ซีแอลบีบรรจุเฉพาะวงจรคำนวณค่าจากตัวดำเนินการวิธีปรับปรุงมอดุโลของผลคูณ วงจรมอดุโลของผลบวก วงจรเอ็กซ์คลูซีฟออร์เกตขนาด 16 บิต วงจรควบคุมการทำงานแบบวงรอบและหน่วยความจำสำหรับการเก็บรหัสกุญแจย่อย สำหรับตัวประมวลผลรหัสกุญแจย่อยสำหรับการเข้าและถอดรหัส สามารถบรรจุลงบนไมโครคอนโทรลเลอร์ จะให้ราคาต้นทุนลดต่ำลงกว่ากึ่งหนึ่งของงานวิจัยนี้ ในขณะที่ความเร็วในการประมวลผลรหัสลับลดลงไปเพียง 1 – 2 เมกกะบิตต่อวินาทีเท่านั้น โดยระบบจะใช้เวลาประมวลผลมากกว่าเดิมเฉพาะการคำนวณรหัสกุญแจย่อยที่มีการเปลี่ยนแปลงไปเท่านั้น (สาวิตร์ ตัณฑนุช และชัชวาล ยนต์หงส์ ,2543 :469-472)

ในการออกแบบวงจรดิจิตอลนั้นจะต้องทำความเข้าใจกับสถาปัตยกรรมของเอฟพีจีเอ การสังเคราะห์วงจรที่ได้จากภาษาบรรยายเพียงเดียวนั้นอาจจะไม่ได้ผลสมบูรณ์ทางไฟฟ้าแม้ว่าจะได้คุณลักษณะทางฟังก์ชันตรงตามที่ต้องการเนื่องจากกระบวนการแปลความจากรีจิสเตอร์เป็นวงจรตรรกะ (registers transfer logics - RTLs) ไม่สามารถถ่ายทอดเป็นวงจรตรรกะเชิงจัดหมู่ได้อย่างสมบูรณ์ ส่วนการออกแบบด้วยแผนผังทางไฟฟ้านั้นจะก่อให้เกิดความยุ่งยากและซับซ้อน ดังนั้นการเลือกวิธีการออกแบบที่เหมาะสมในแต่ละวงจรย่อยจะทำให้กระบวนการสังเคราะห์วงจรเกิดความสมบูรณ์และได้ประสิทธิภาพการออกแบบที่ดี (สาวิตร์ ตัณฑนุช และชัชวาล ยนต์หงส์ ,2543 :465-468)

และเนื่องจากบัสของข้อมูลนี้มีขนาด 64 บิต ในการประยุกต์ใช้งานจริง ควรมีการพัฒนาให้สามารถเชื่อมโยงสัญญาณข้อมูลกับอุปกรณ์ภายนอกแบบอนุกรมเพื่อลดขนาดของบัสและป้องกันการเกิดการรบกวนแบบสัญญาณไขว้แทรก (crosstalk) ดังแสดงตัวอย่างวงจรในภาพประกอบที่ 7.1



ภาพประกอบที่ 7.1 การเชื่อมโยงแบบอนุกรมสำหรับระบบประมวลผลรหัสลับไอดีอีเอ

บรรณานุกรม

สาวิตร์ ตัณฑนุช และ ชัชวาลย์ ยนต์หงส์. 2543. "การเปรียบเทียบสมรรถนะสถาปัตยกรรมรหัสลับ DES บน FPGA ในอนุกรม XC4000E/X", รายงานการประชุมวิชาการวิศวกรรมไฟฟ้า ครั้งที่ 23, 465-468 : มหาวิทยาลัยเชียงใหม่

สาวิตร์ ตัณฑนุช และ ชัชวาลย์ ยนต์หงส์. 2543. "การพัฒนาระบบรหัสลับ IDEA ด้วย FPGA และไมโครคอนโทรลเลอร์", รายงานการประชุมวิชาการวิศวกรรมไฟฟ้า ครั้งที่ 23, 469-472: มหาวิทยาลัยเชียงใหม่

Borst, J. 1997. "Differential-Linear Cryptanalysis of IDEA" ,Technical Report ESATCOSIC Report 96-2, Department of Electrical Engineering, Leuven: Katholieke University , 1-13.

Caspi, E. and Weaver, N. 1996. "IDEA as a Benchmark for Reconfigurable Computing", Technical report of BRASS research group. California: University of Berkeley,1-13.

Curiger, A., Bonnenberg, H. and Kaeslin, H. 1991. "Regular VLSI Architectures for Multiplication modulo", IEEE Journal of Solid-State Circuits, Vol.26 , Issue 7, 990 -- 994.

Curiger, A., Bonnenberg, H., Zimmermann, R., Felber, N., Kaeslin, H. and Fichtner, W. . 1993. "VINCI:VLSI Implementation of the New Block Cipher IDEA", The IEEE CICC'93 , Sandiago, 15.5.5-15.5.4.

Daemen, J., Govaerts, R. and Vandewalle, J. 1992. "Cryptanalysis of 2.5 Rounds of IDEA (Extended Abstract) ",Computer Security ESORICS92, European Symposium on Research in Computer security Proceeding. Springer-Verlag, 419-434.

Daemen, J., Govaerts, R. and Vandewalle, J. 1993. "Block Ciphers Based on Modular Arithmetic", State and Progress of Research in Cryptography. EUROCRYPT'93 Proceeding. Springer-Verlag, 80-89

Daemen, J., Govaerts, R. and Vandewalle, J. 1994. "Weak Keys For IDEA" ,Advance in Cryptography. EUROCRYPT'93 Proceeding .Springer-Verlag, 224-231

Koren, I. 1993. Computer Arithematic Algorithms. 1st ed. New Jersey : Prentice Hall Inc.

Lai, X. and Massey, J. 1990. "A proposal for a New Block Cipher", Advance in Cryptography. EUROCRYPT'90 Proceeding .Springer-Verlag ,quoted in A.J.Menezes , P.C.van Oorschot and S.A. Vanstone . 1997. Handbook of Applied Cryptography. New York:CRC Press, 263-264

Lai, X. and Massey, J. 1991. "Markov Ciphers and Differtial Cryptanalysis", Advance in Cryptography. EUROCRYPT'91 Proceeding .Springer-Verlag, 17-38

Lipmaa, H. 1998. "IDEA:A Cipher for Multimedia Architectures", Cryptography 1998. Springer-Verlag, 248-263.

Menezes, A.J., Oorschot, P.C.van and Vanstone, S.A. 1997. Handbook of Applied Cryptography. New York:CRC Press.

Mosanya, E., Teuscher, C., Restrepo, H.F., Galley ,P. and Sanchez, E. 1999. "CryptoBooster:A Reconfigurable and Modular Cryptographic Coprocessor", Advance and Embedded Systems'99. International Workshop on Cryptographic Proceeding. Springer-Verlag, 246-256.

Pihl, J.and Aas ,E. J. 1996. "A Multiplier and Squarer Generator for High Performance DSP Applications", 39th Midwest Symposium on Circuits and Systems. Iowa.

Rosen, K.H. 1992. Elementary Number Theory and It's Application. :Addison-Wesley Publishing Company.

Schneier, B. 1996. Applied Cryptography. 2D ed. New York:John Wiley&Sons Inc.

Sjoholm, S.and Lindh, L. 1997. VHDL for DESIGNERS. 1st ed. New Jersey : Prentice Hall Inc.

Stinson, D.R.. 1995. Cryptography Theory and Practice. New York:CRC Press.

Tanthanuch, S. and Yonhong, C. 2000. "A Performance Comparison of DES Architecture using XC4000E/X series, FPGAs" ,1st Seminar on Collaboration in Computer Communications and IC Design Technology in Thailand. Prince of Songkla University. Songklha,Thailand.

Tanthanuch, S. and Yonhong, 2000. "Improvement of IDEA cryptographic systems using FPGA and Microcontroller" , 1st Seminar on Collaboration in Computer Communications and IC Design Technology in Thailand. Prince of Songkla University. Songklha,Thailand.

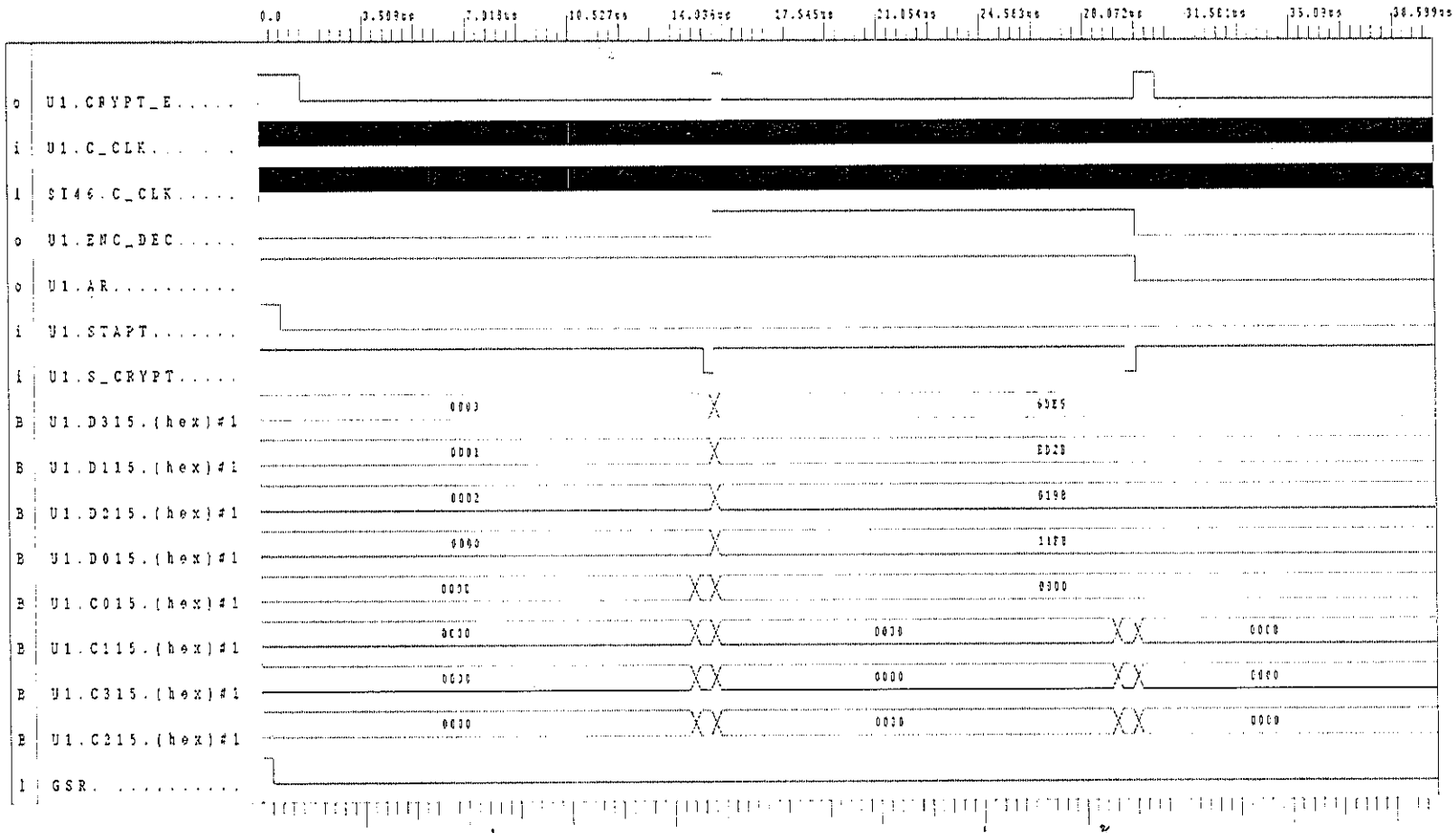
Wolter, S., Matz, H., Laur, R. and Matz, A. 1995. "On VLSI implementation of the IDEA". IEEE International Symposium on Circuits and Systems, 397-400.

Xilinx Inc. 1999. The Programmable Logic Data Book 1999. 1999 Edition.

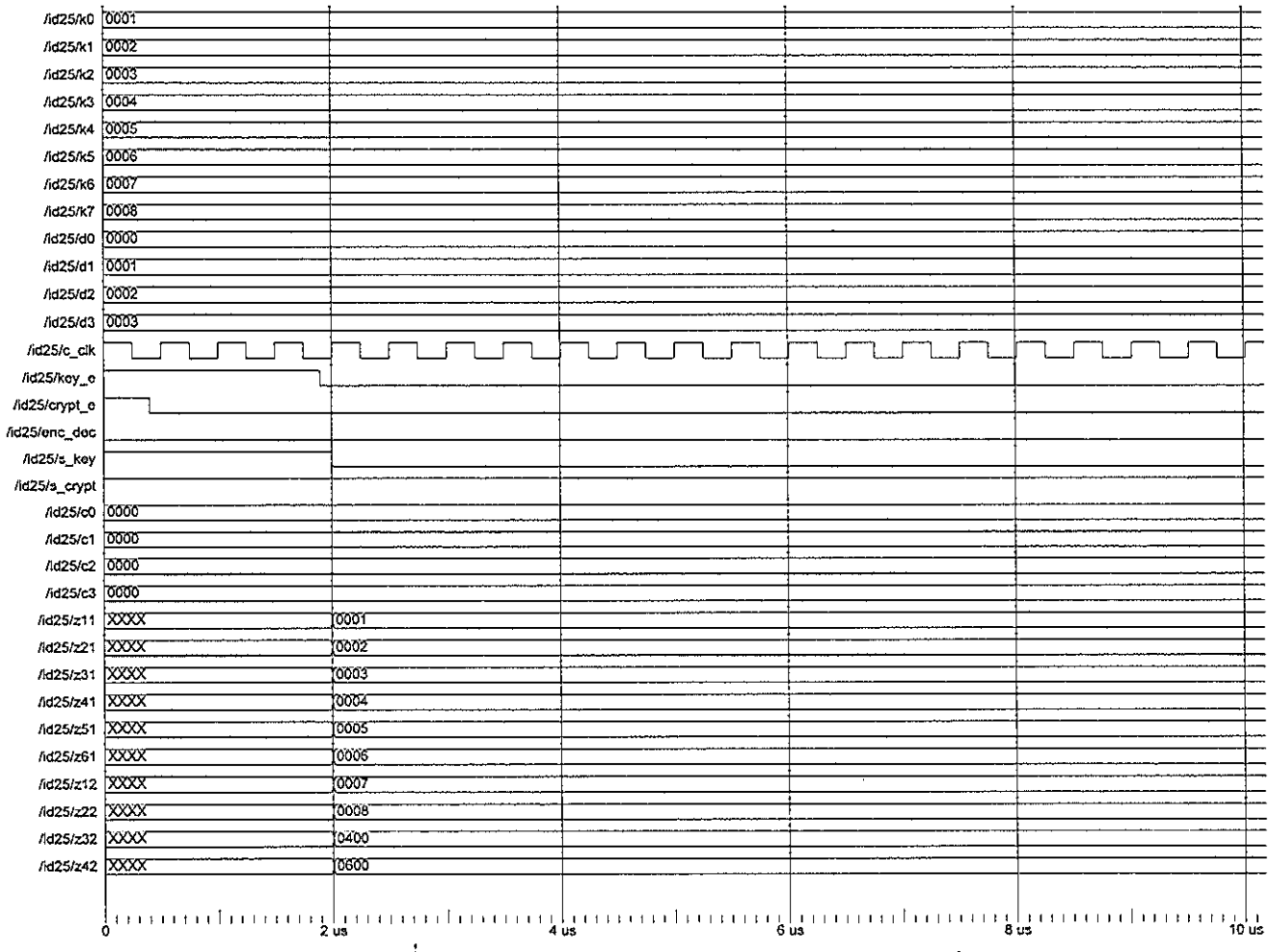
Zimmermann, R., Curiger, A., Bonnenberg, H., Felber, N., Kaeslin, H. and Fichtner, W. . 1994. "A 177 Mbit/s VLSI Implementation of International Data Encryption Algorithm". IEEE Journal of Solid-State Circuits, Vol29, Issue 3, 303-307.

Zimmermann, R. 1999. "Efficient VLSI Implementation of Modulo Addition and Multiplication". 14th IEEE Symposium on Computer Arithematic. Adelaide, Australia,148-157.

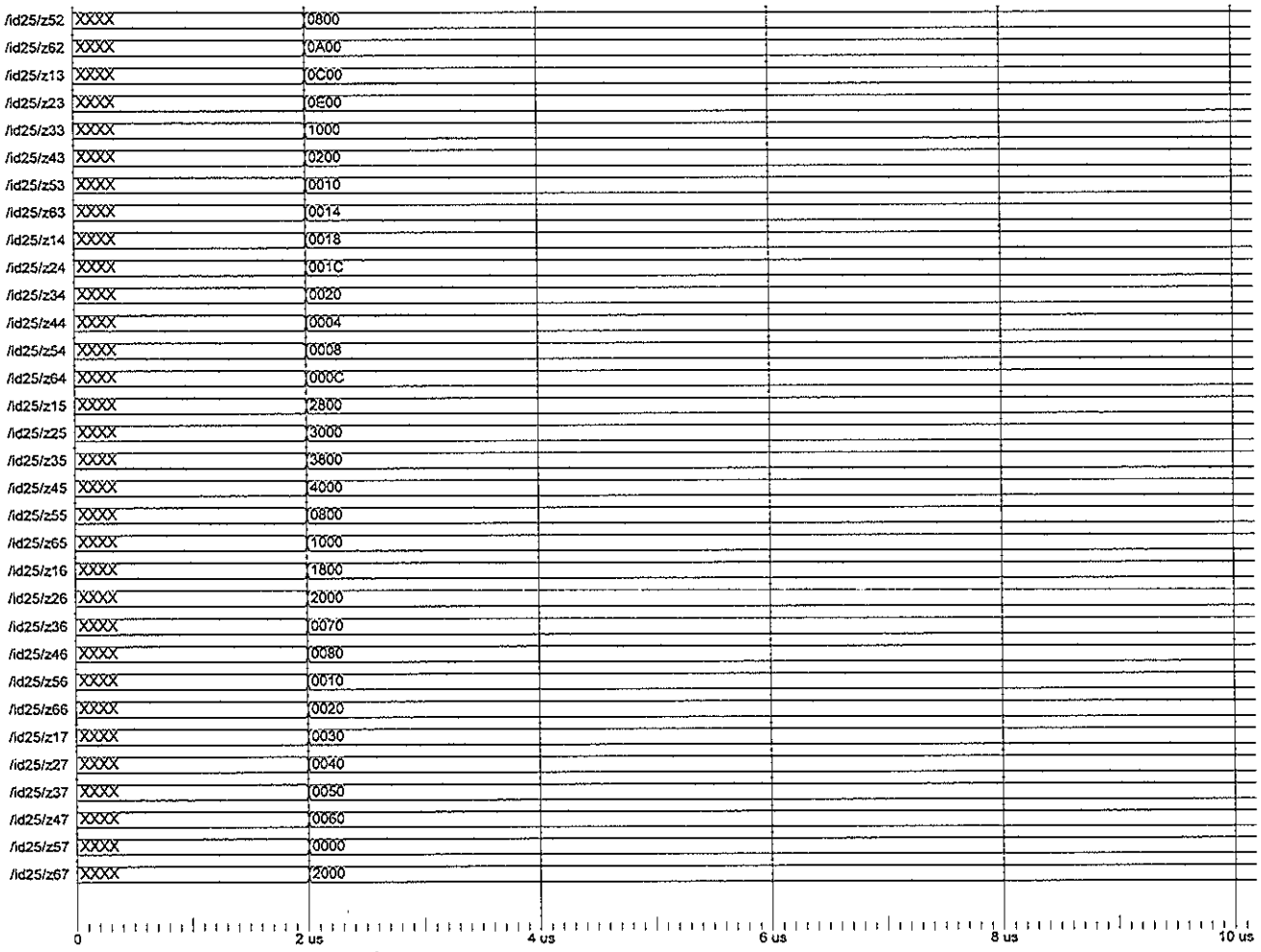
ภาคผนวก ก แผนผังทางเวลาของวงจรรวมต้นแบบ



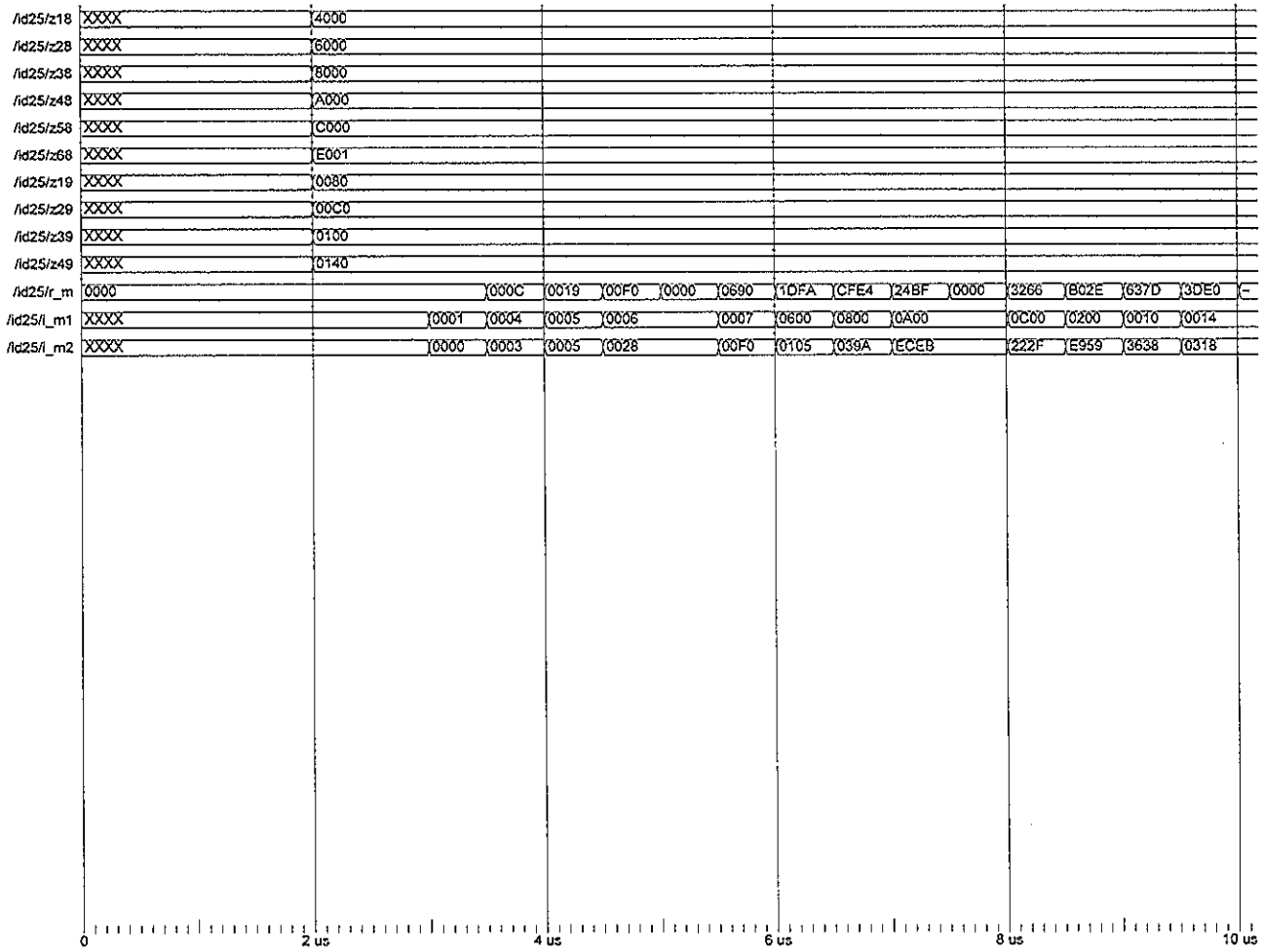
ภาพประกอบที่ ก1 ผลการจำลองสัญญาณทางเวลาจากวงจรที่สังเคราะห์ขึ้น



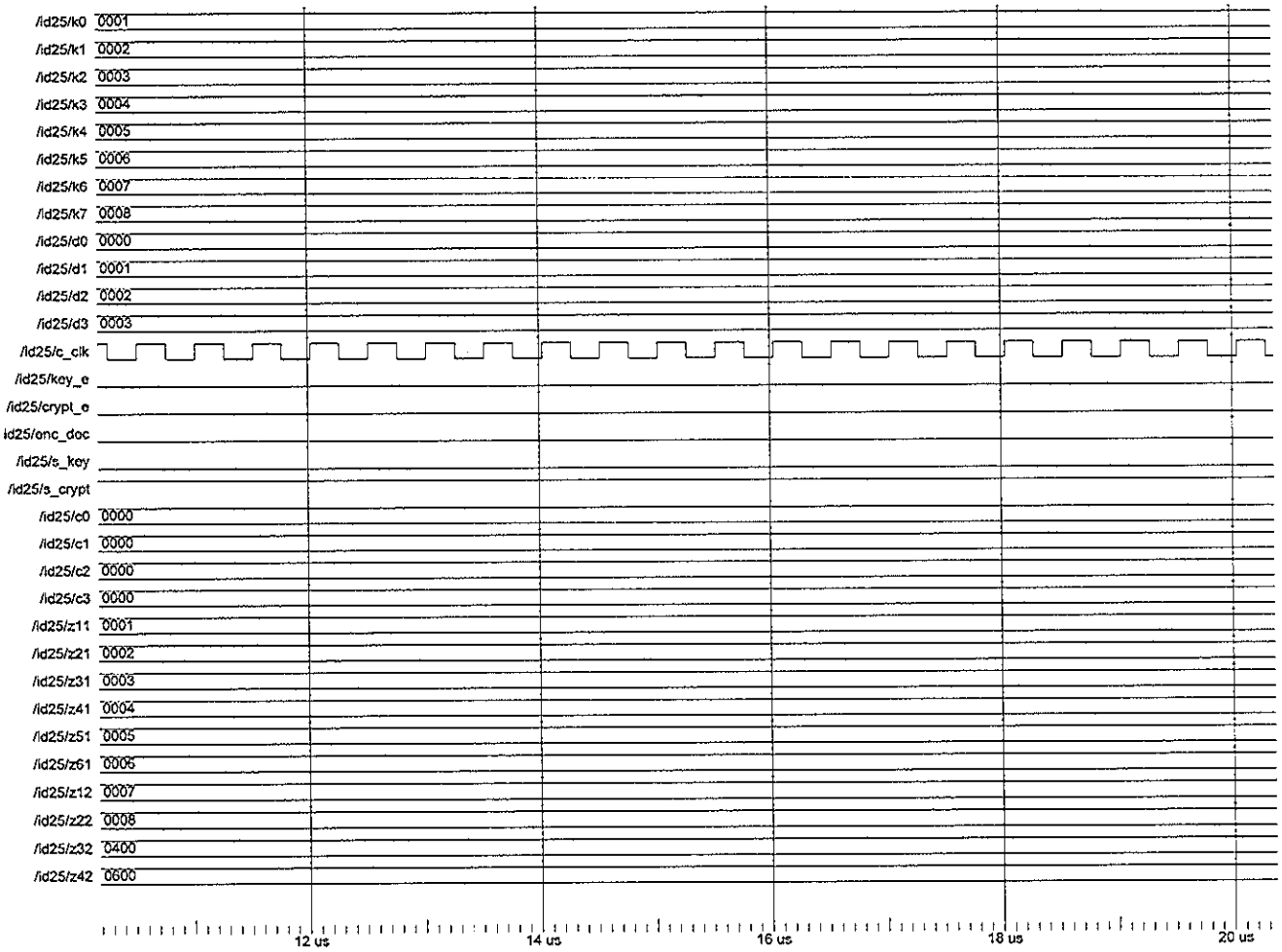
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส



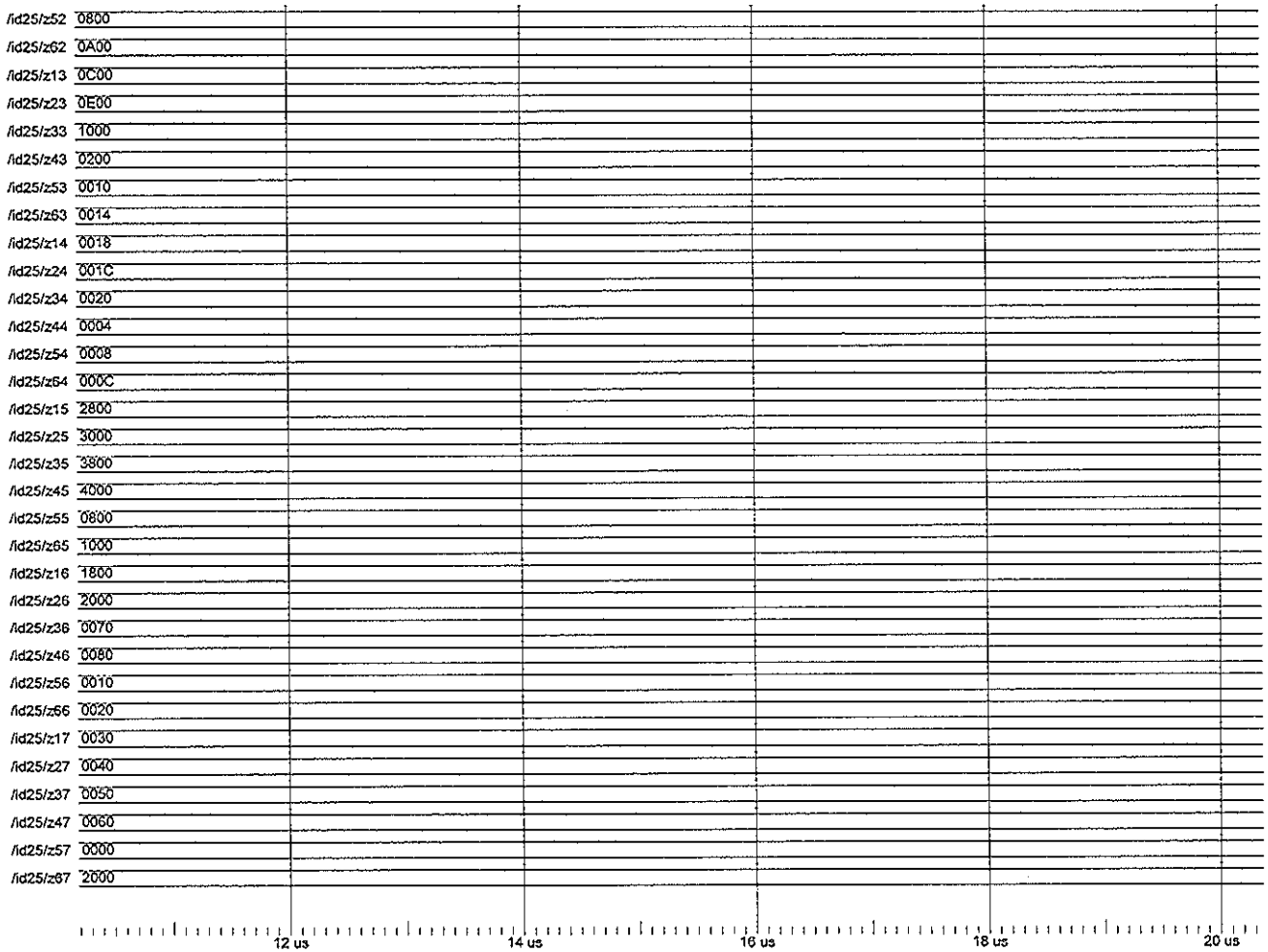
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



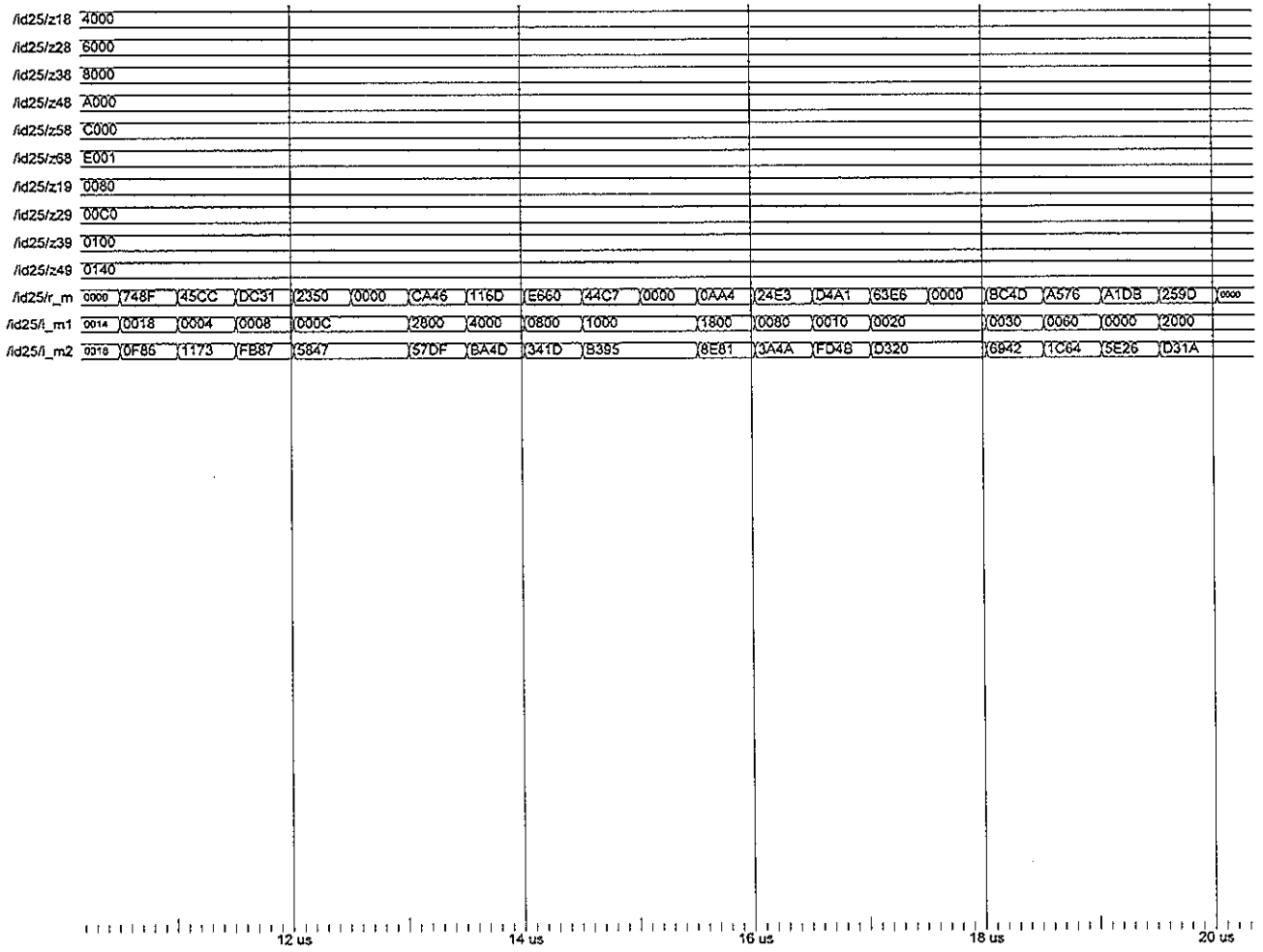
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



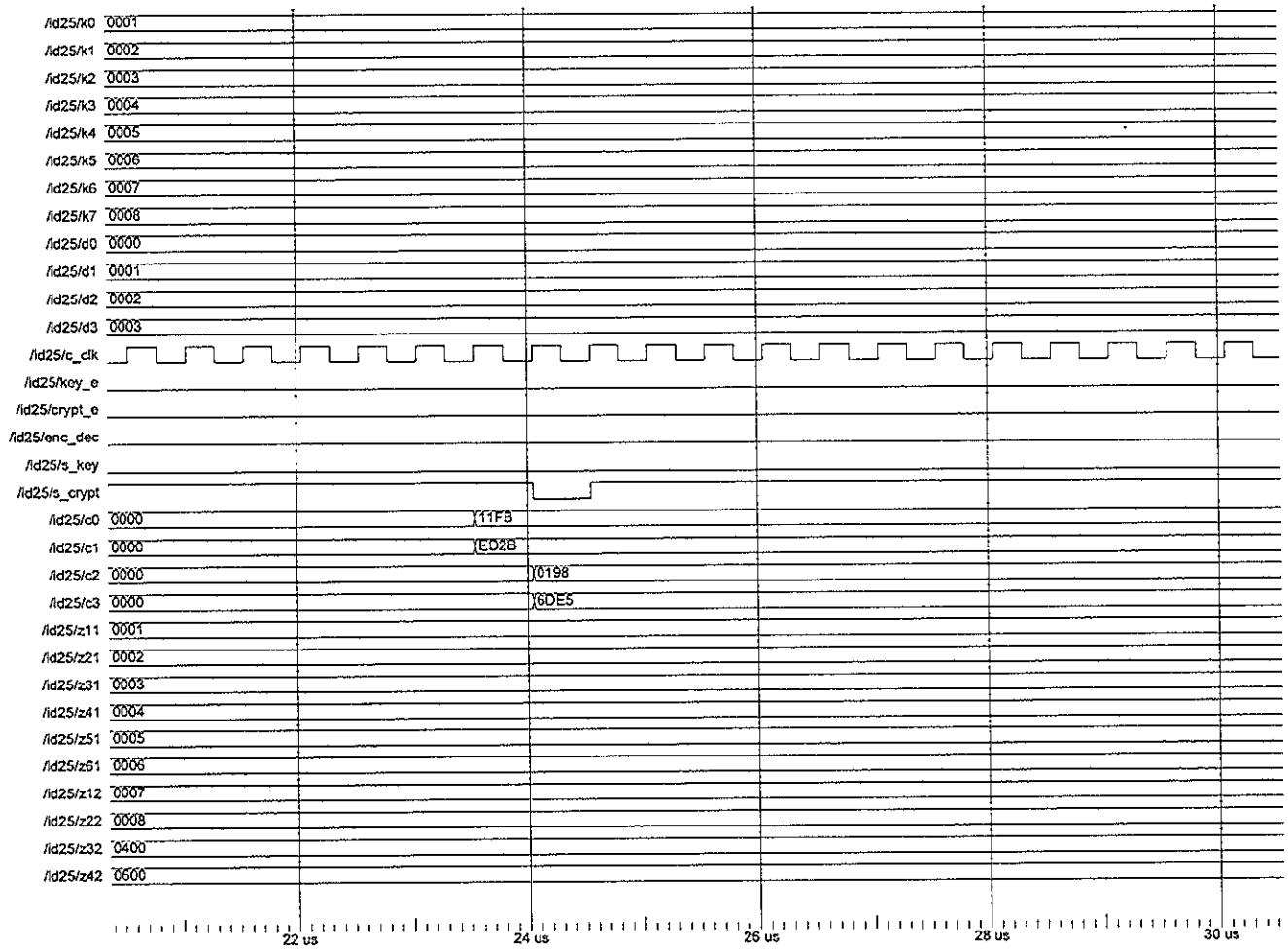
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



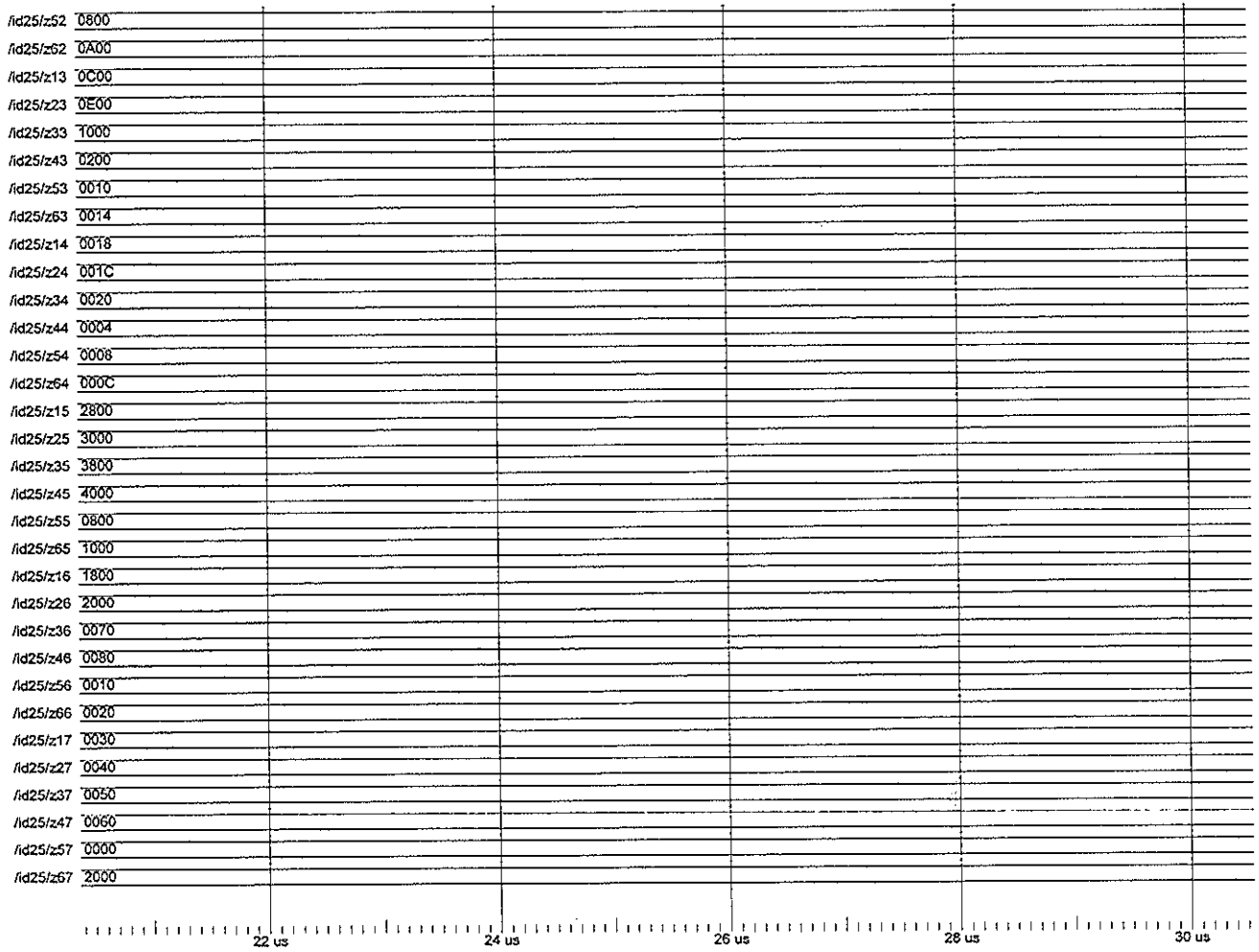
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



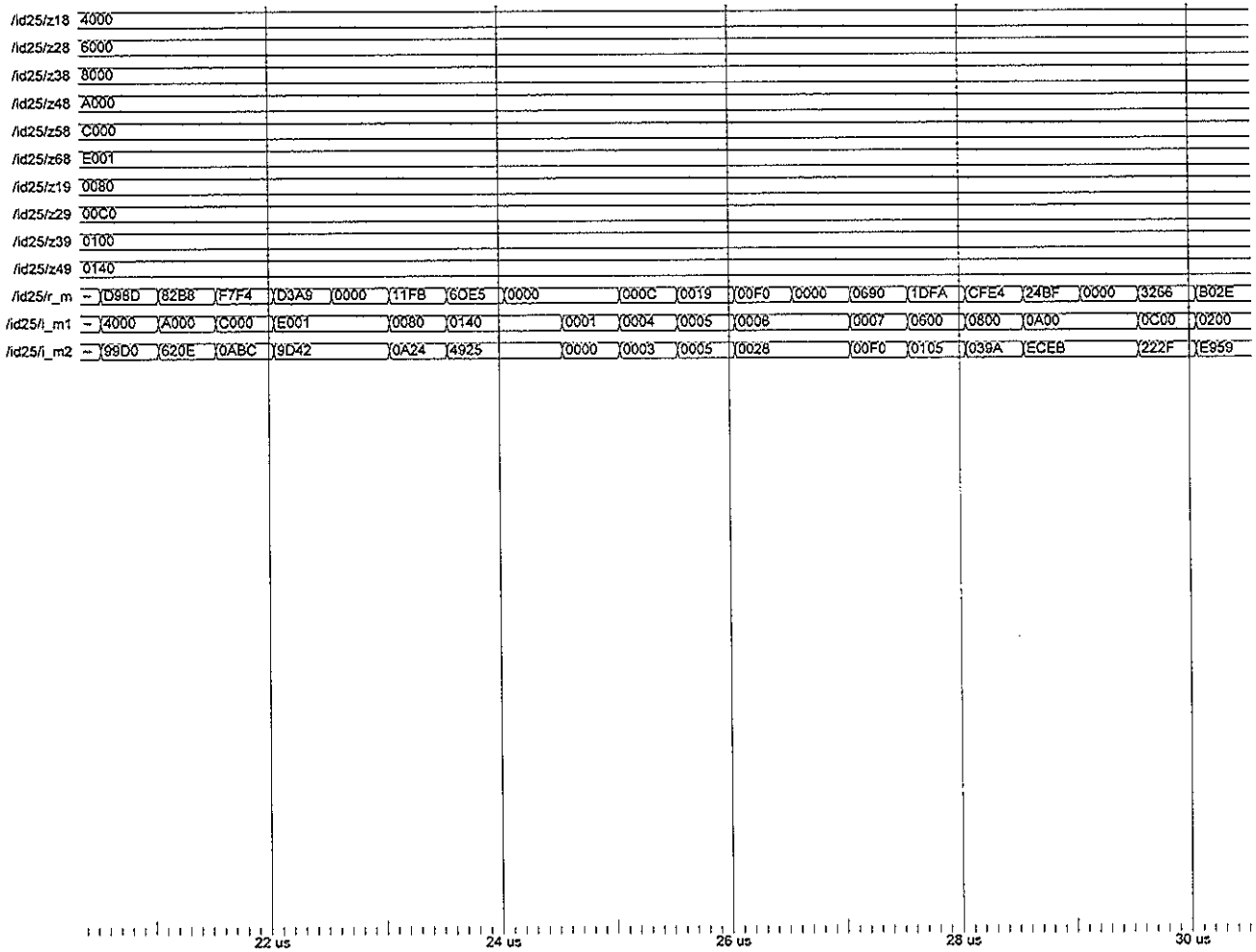
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



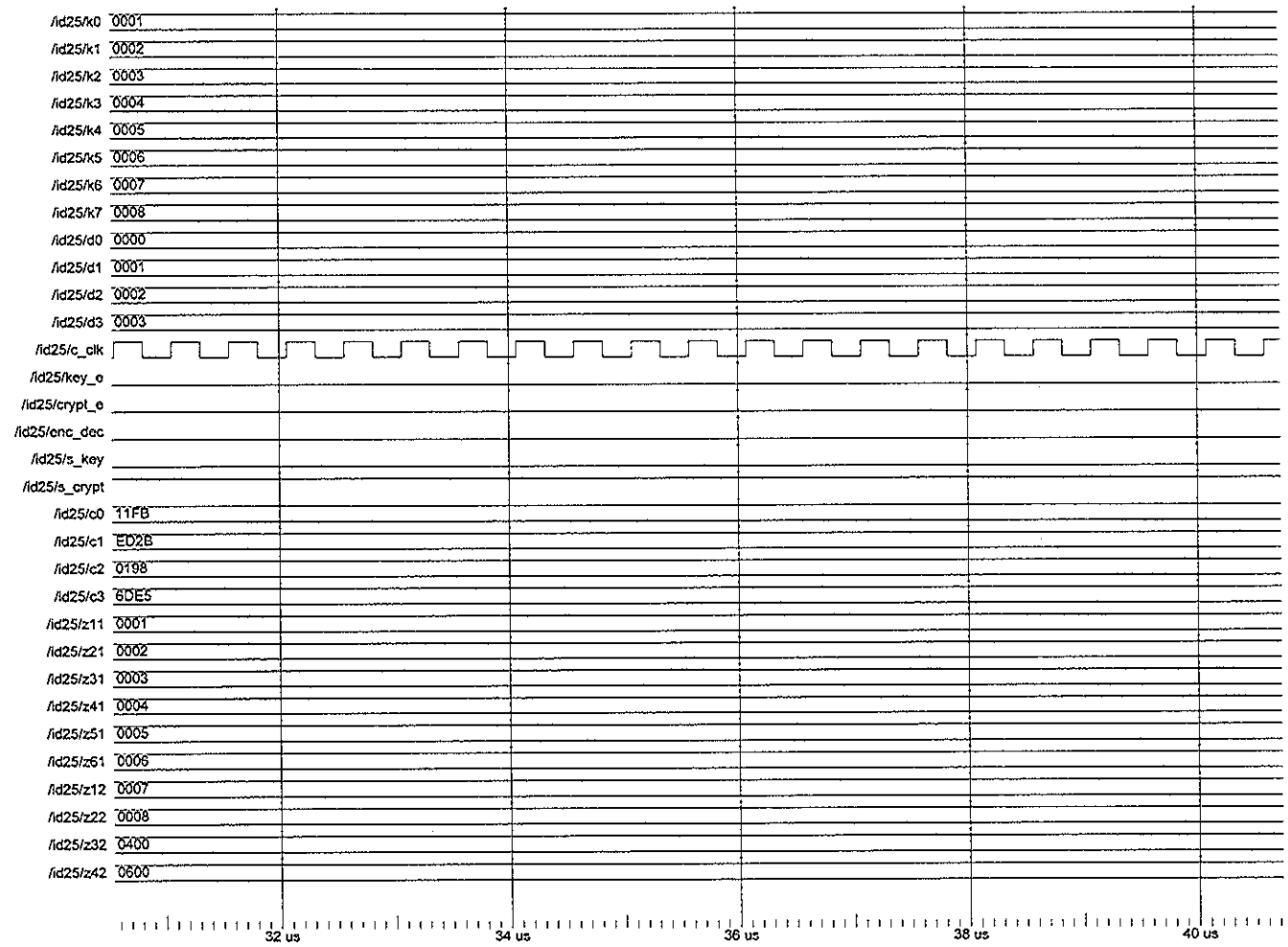
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



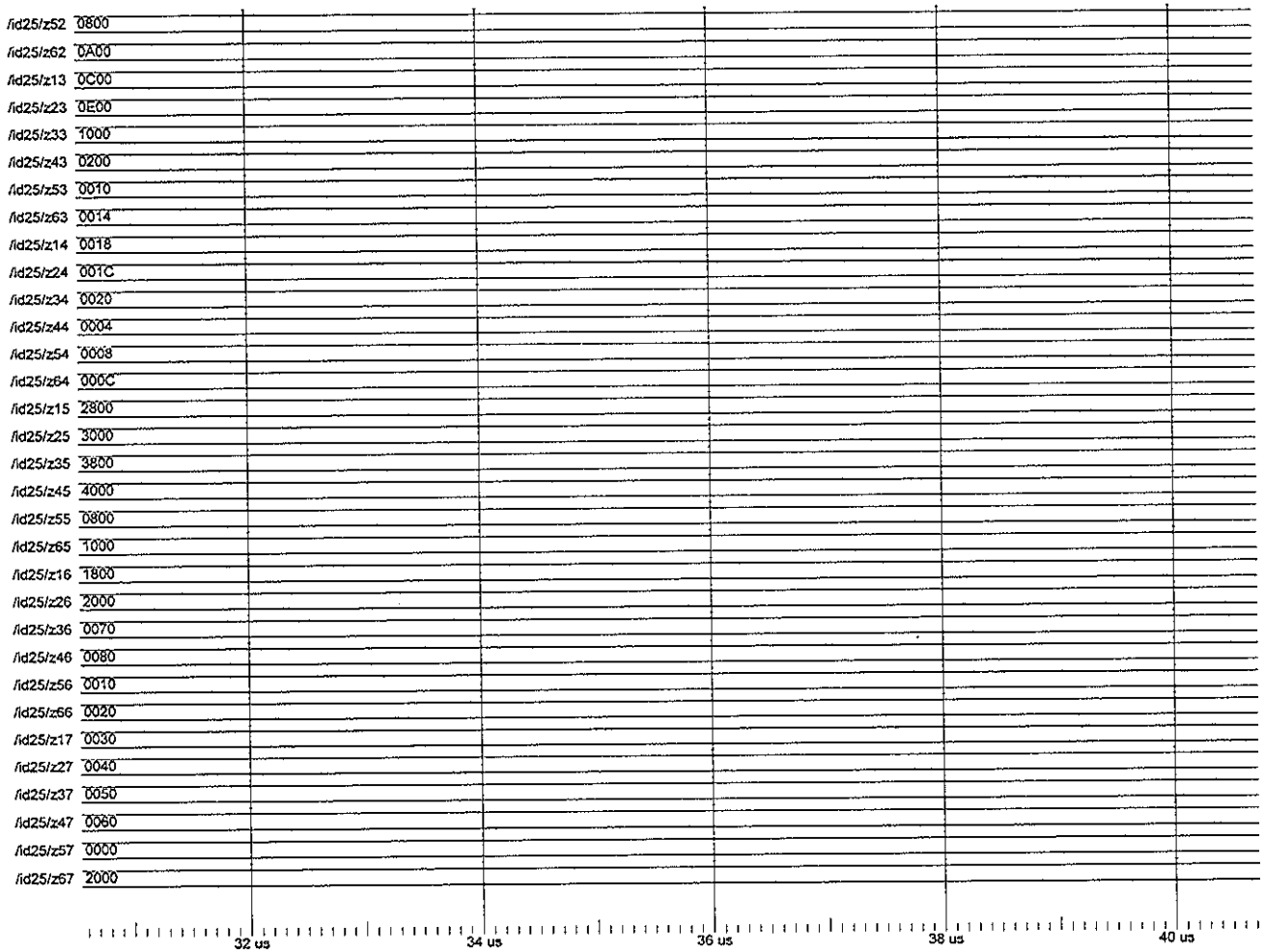
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางพังก์ชันของการเข้ารหัส (ต่อ)



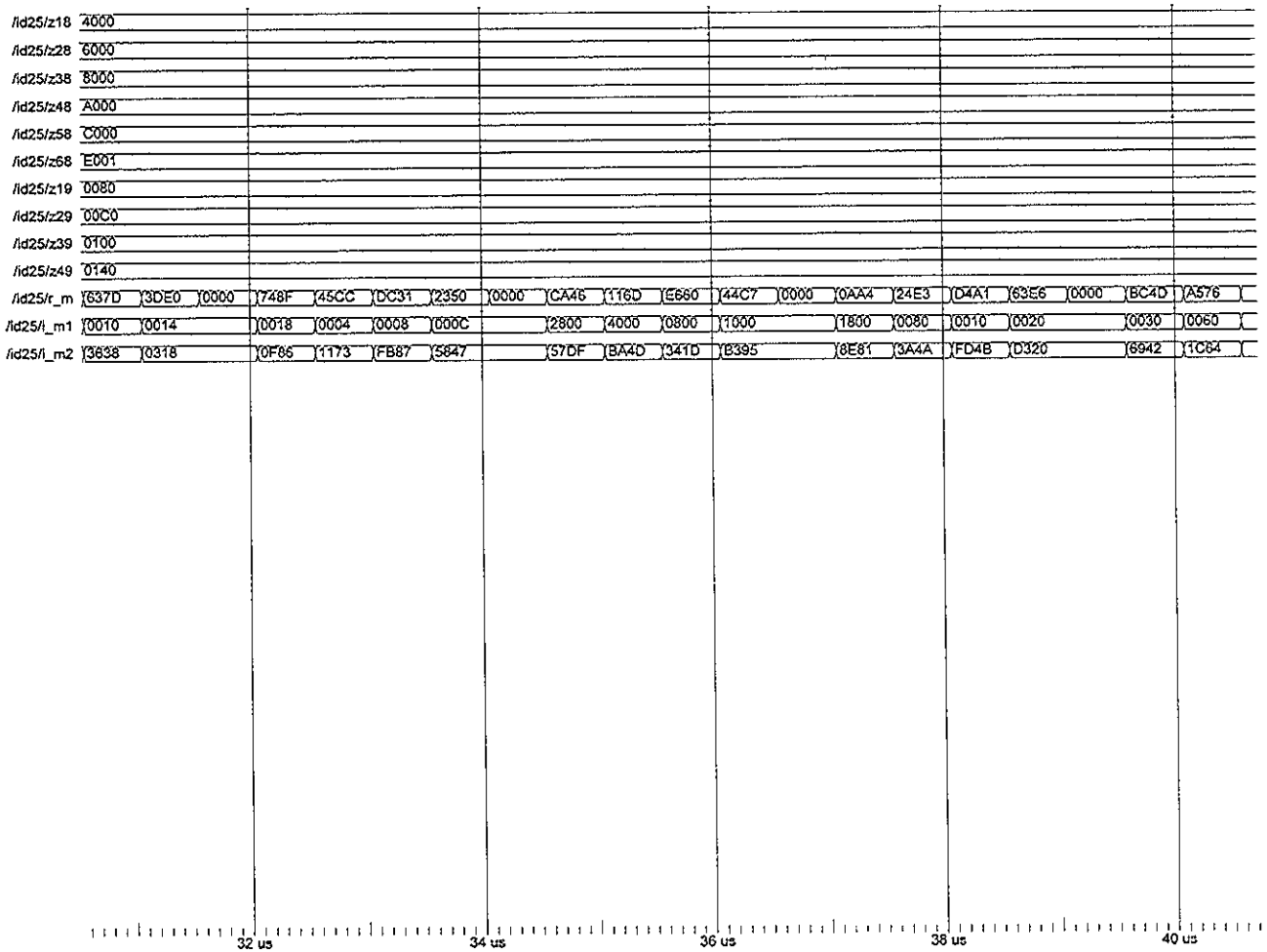
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



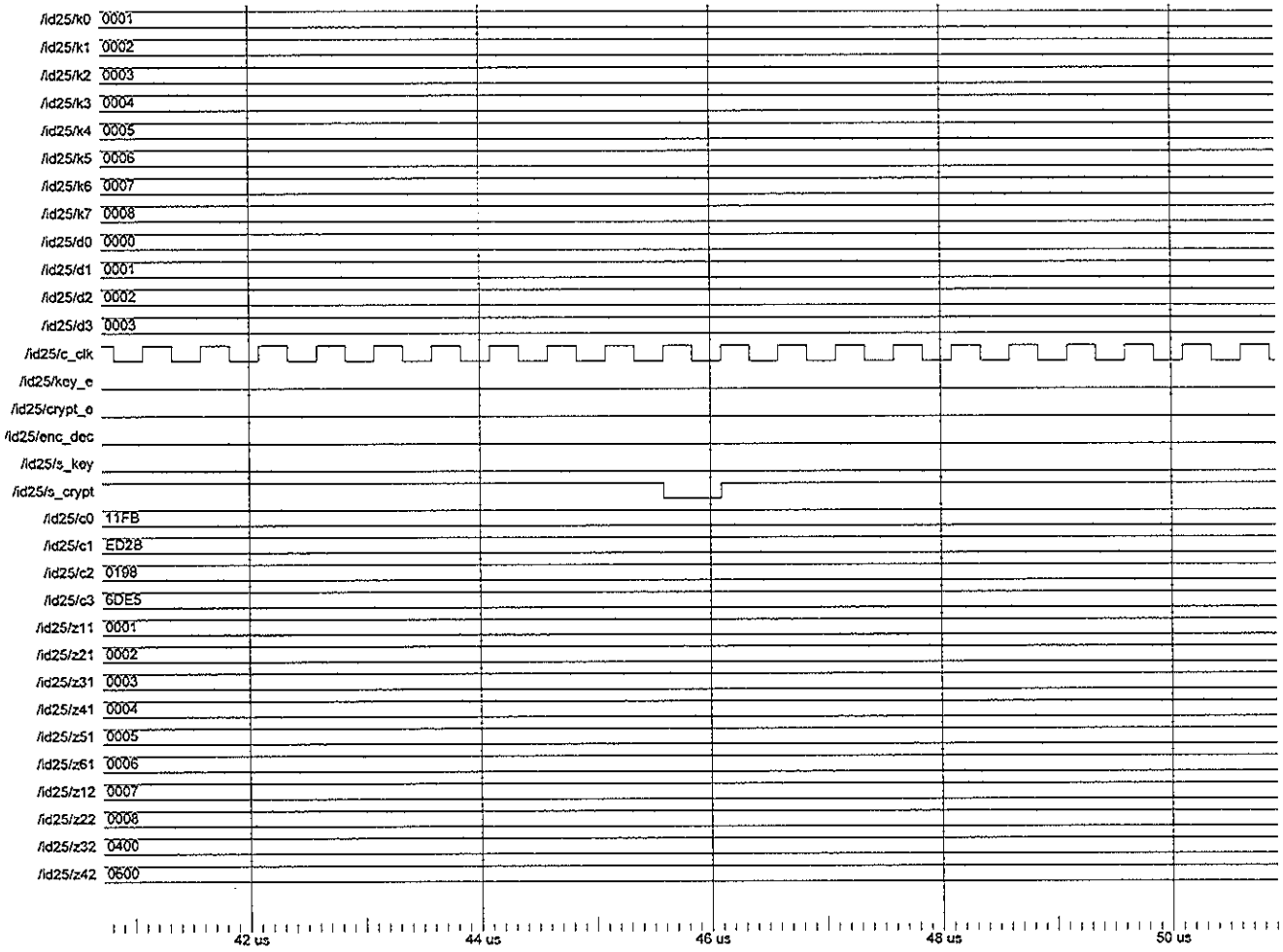
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



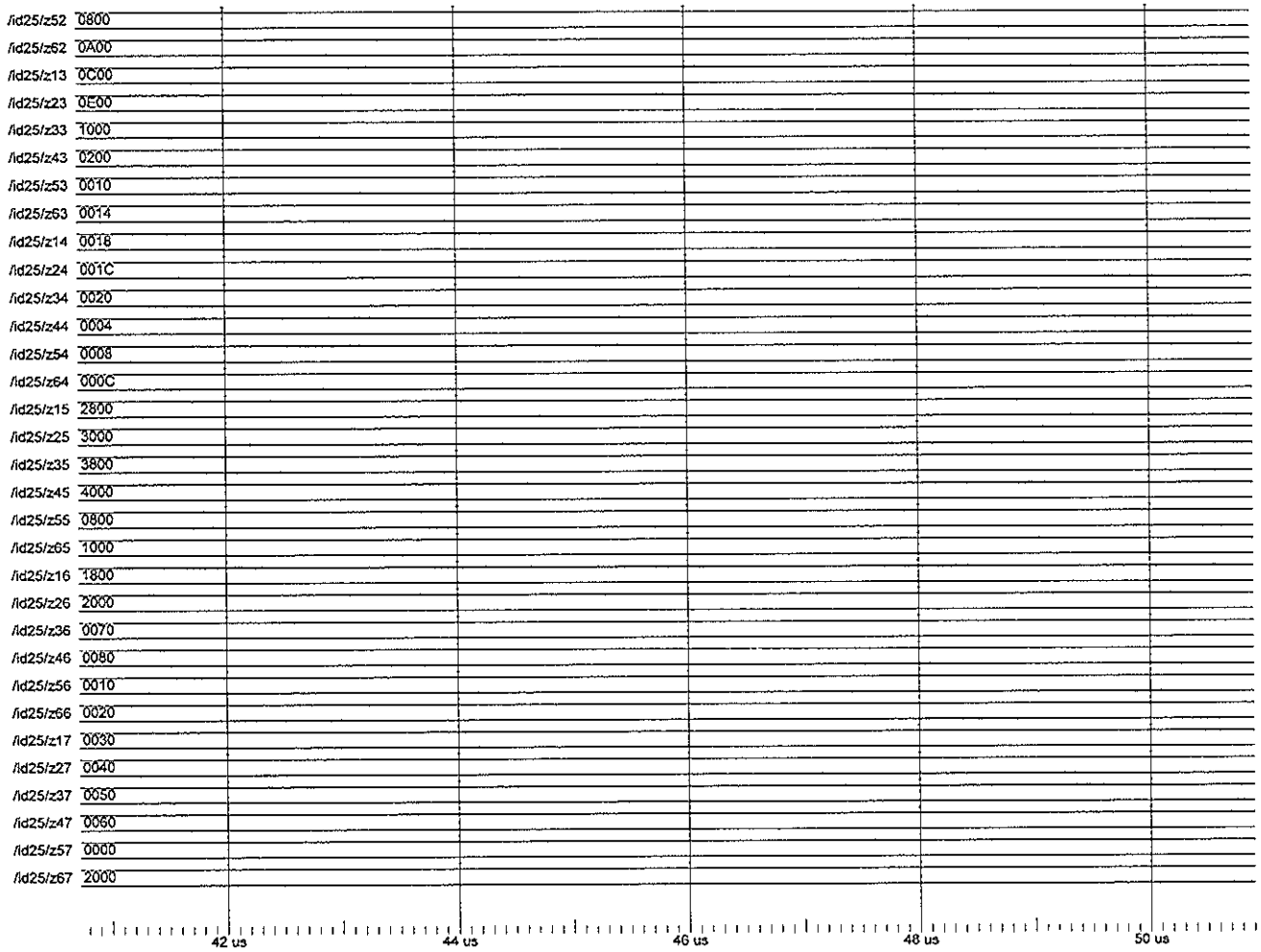
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



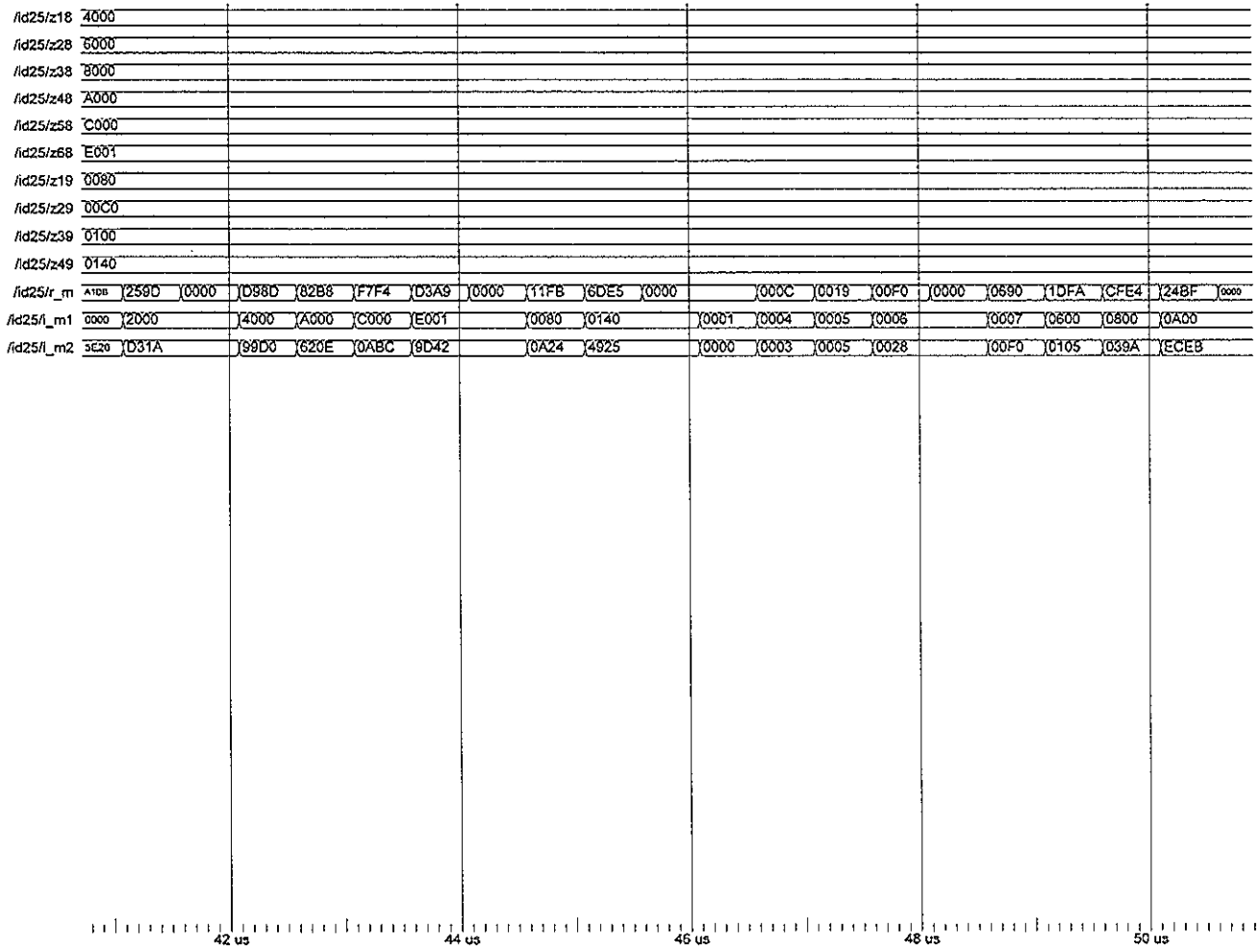
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



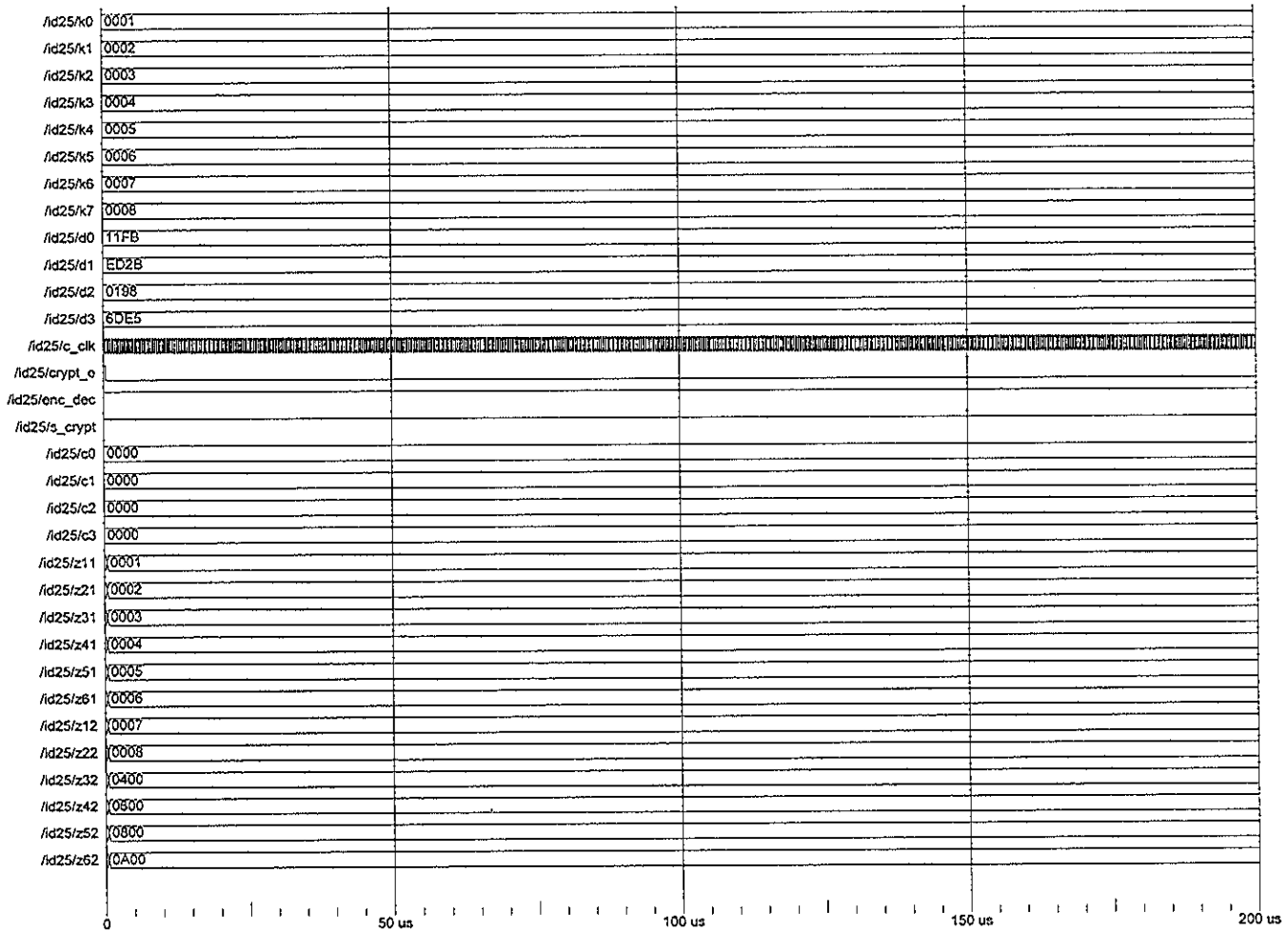
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



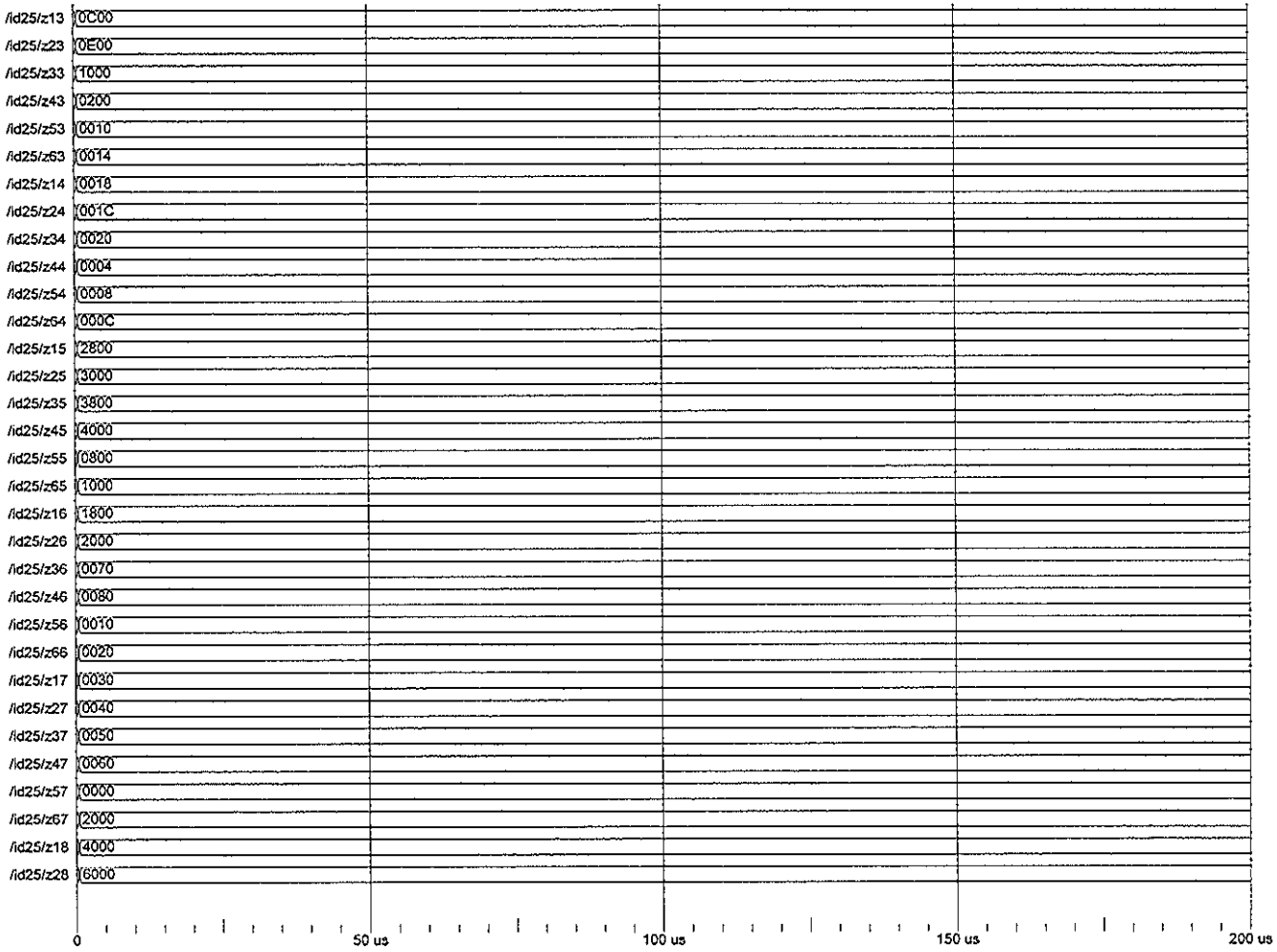
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



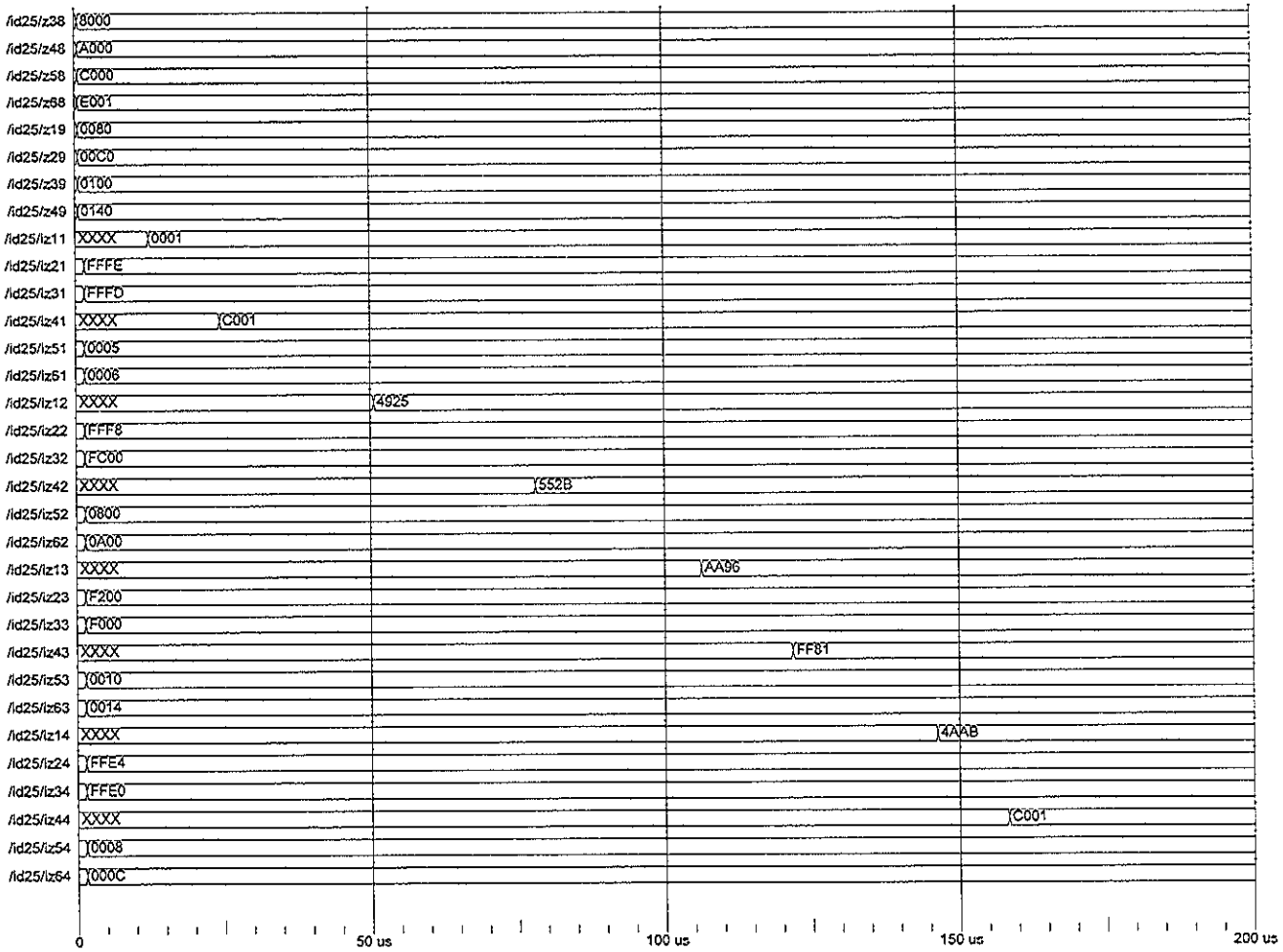
ภาพประกอบที่ ก2 ผลการจำลองสัญญาณทางฟังก์ชันของการเข้ารหัส (ต่อ)



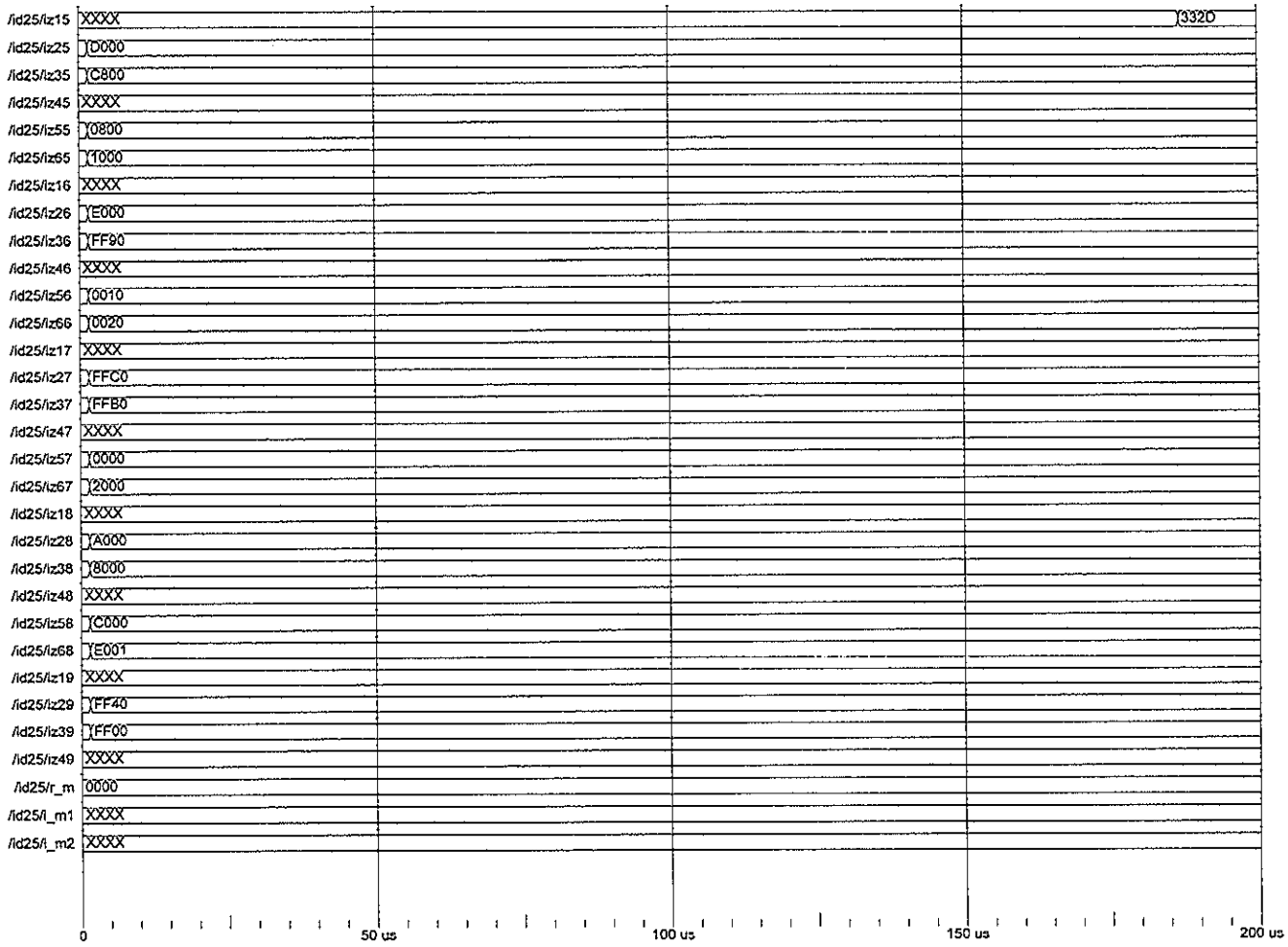
ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส



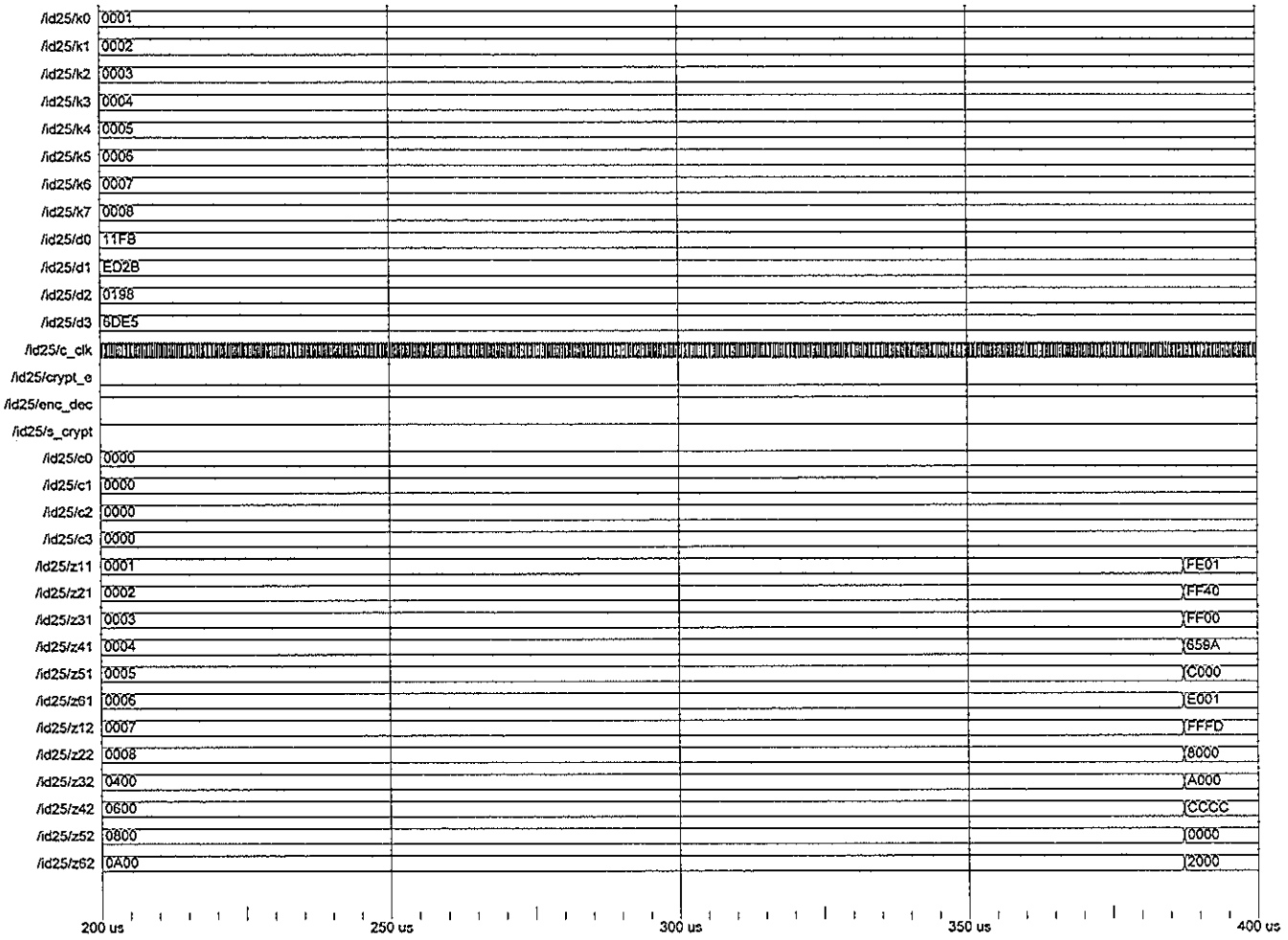
ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)



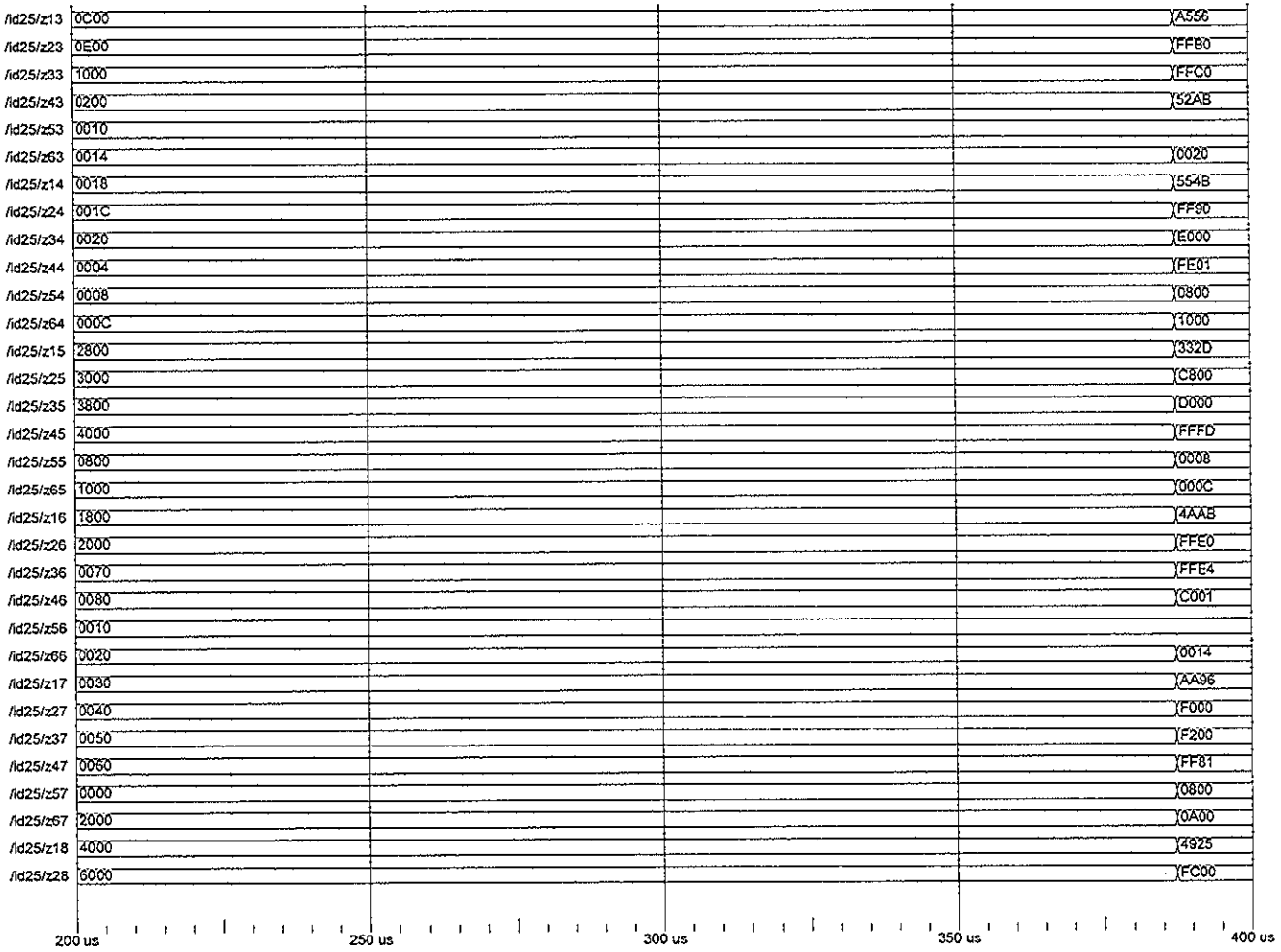
ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)



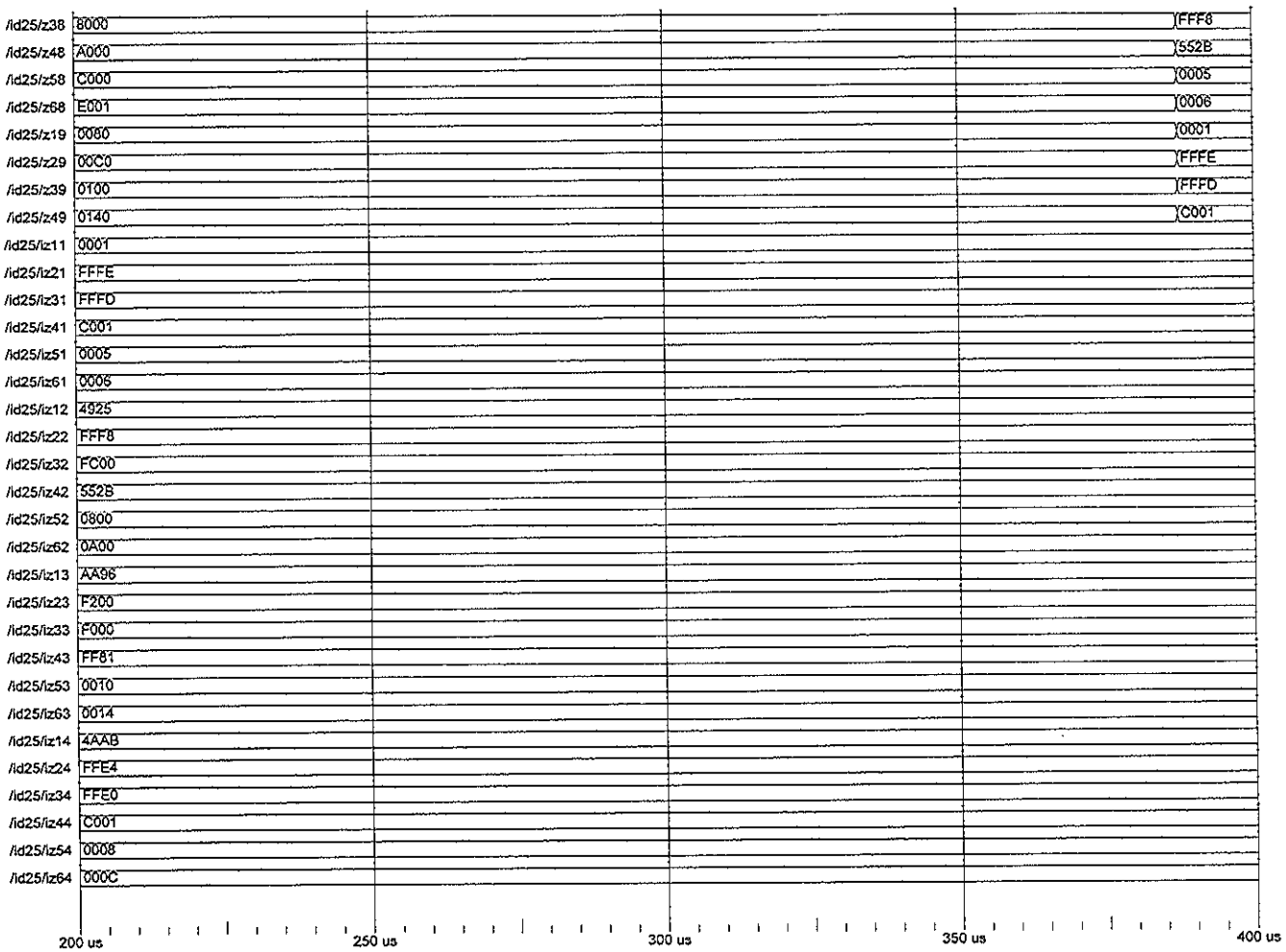
ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)



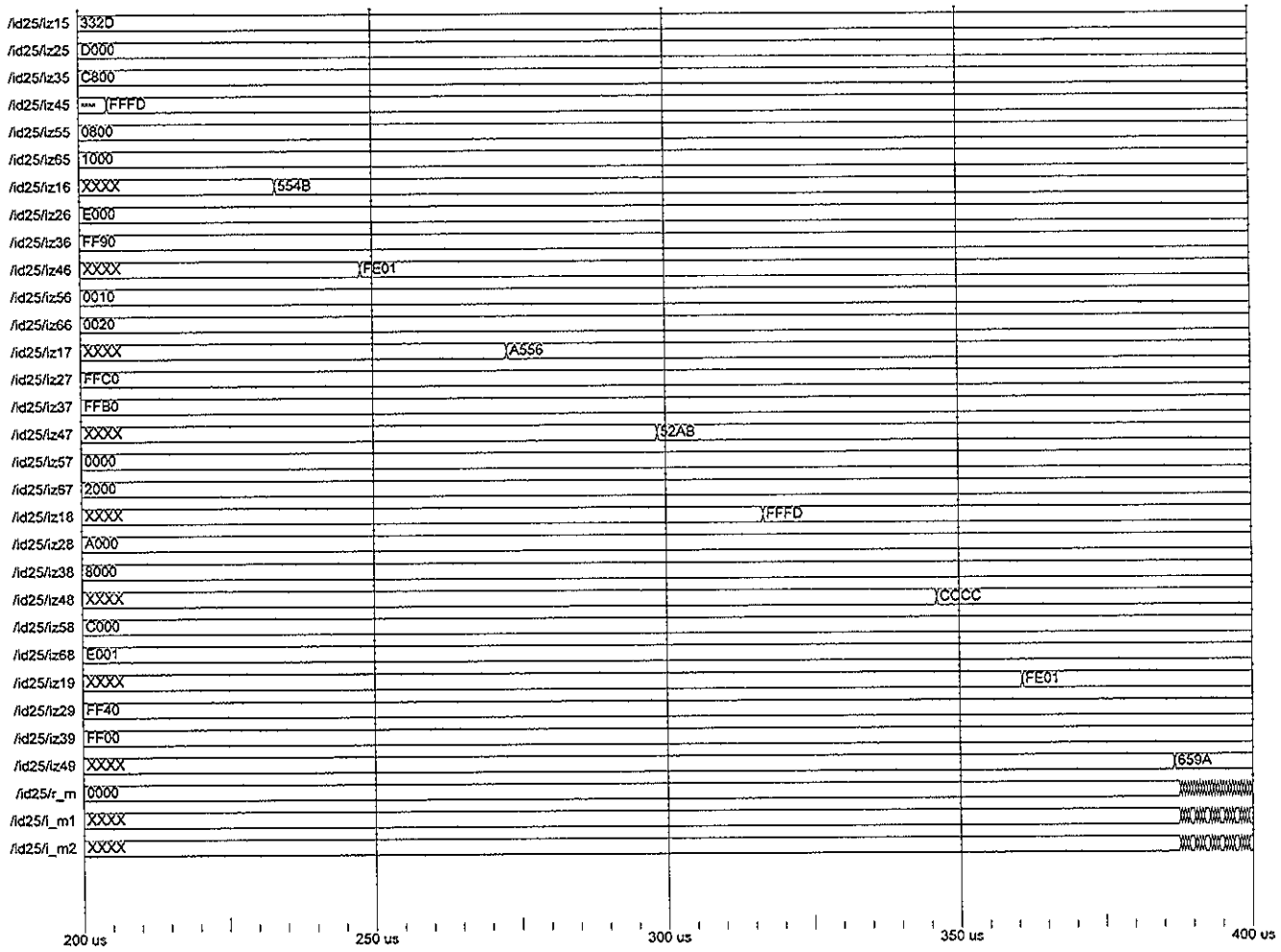
ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)



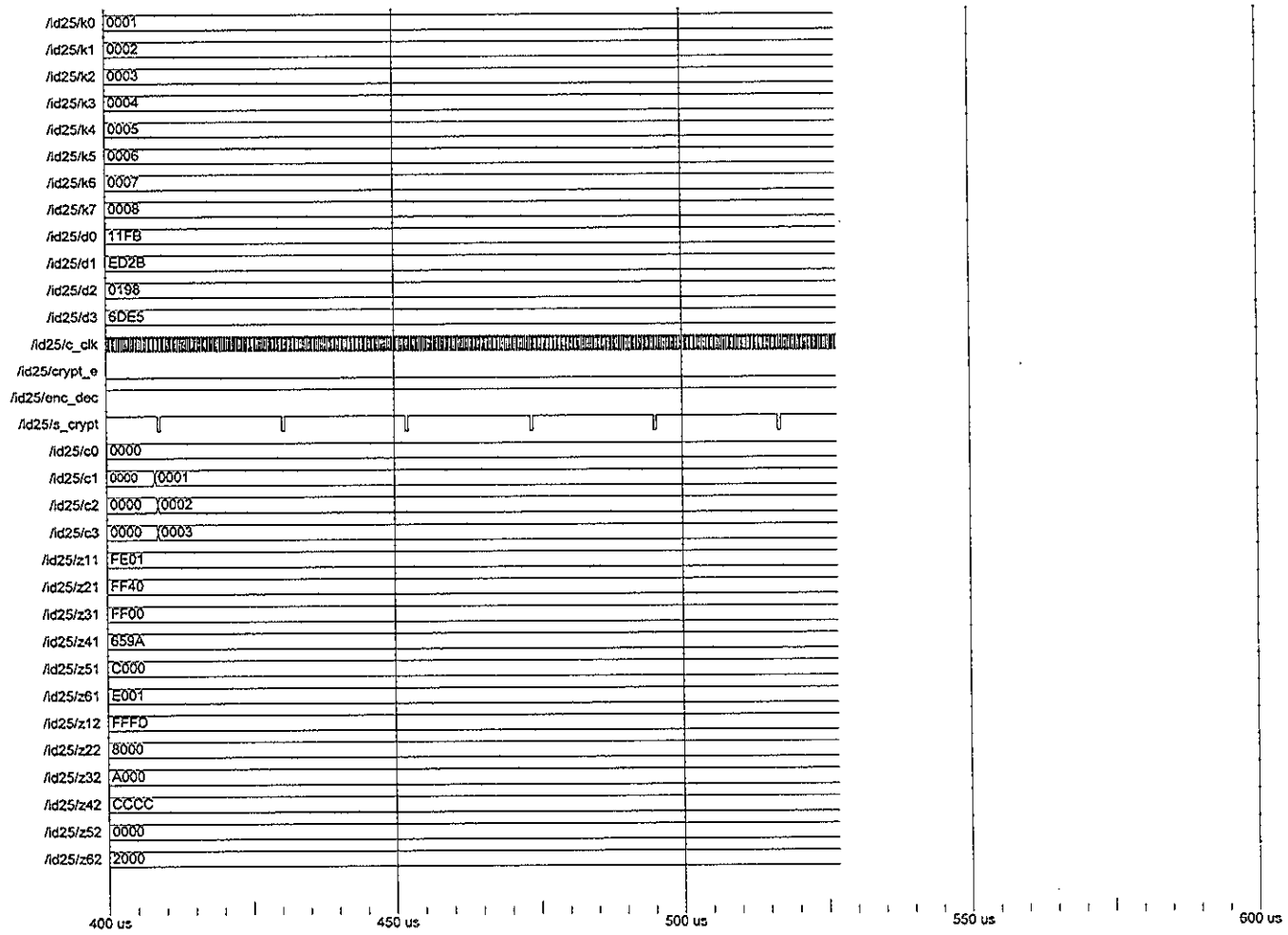
ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)



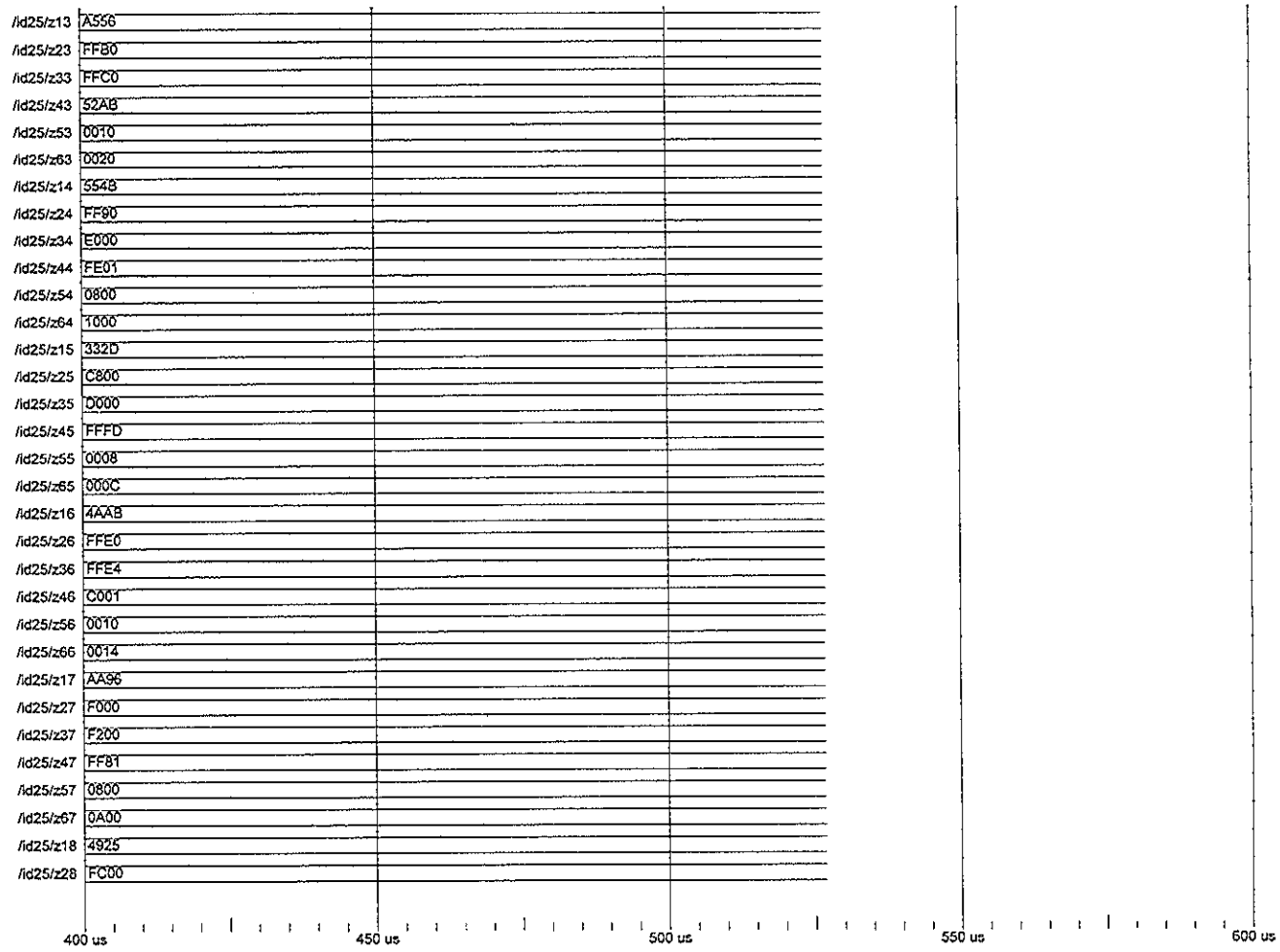
ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)



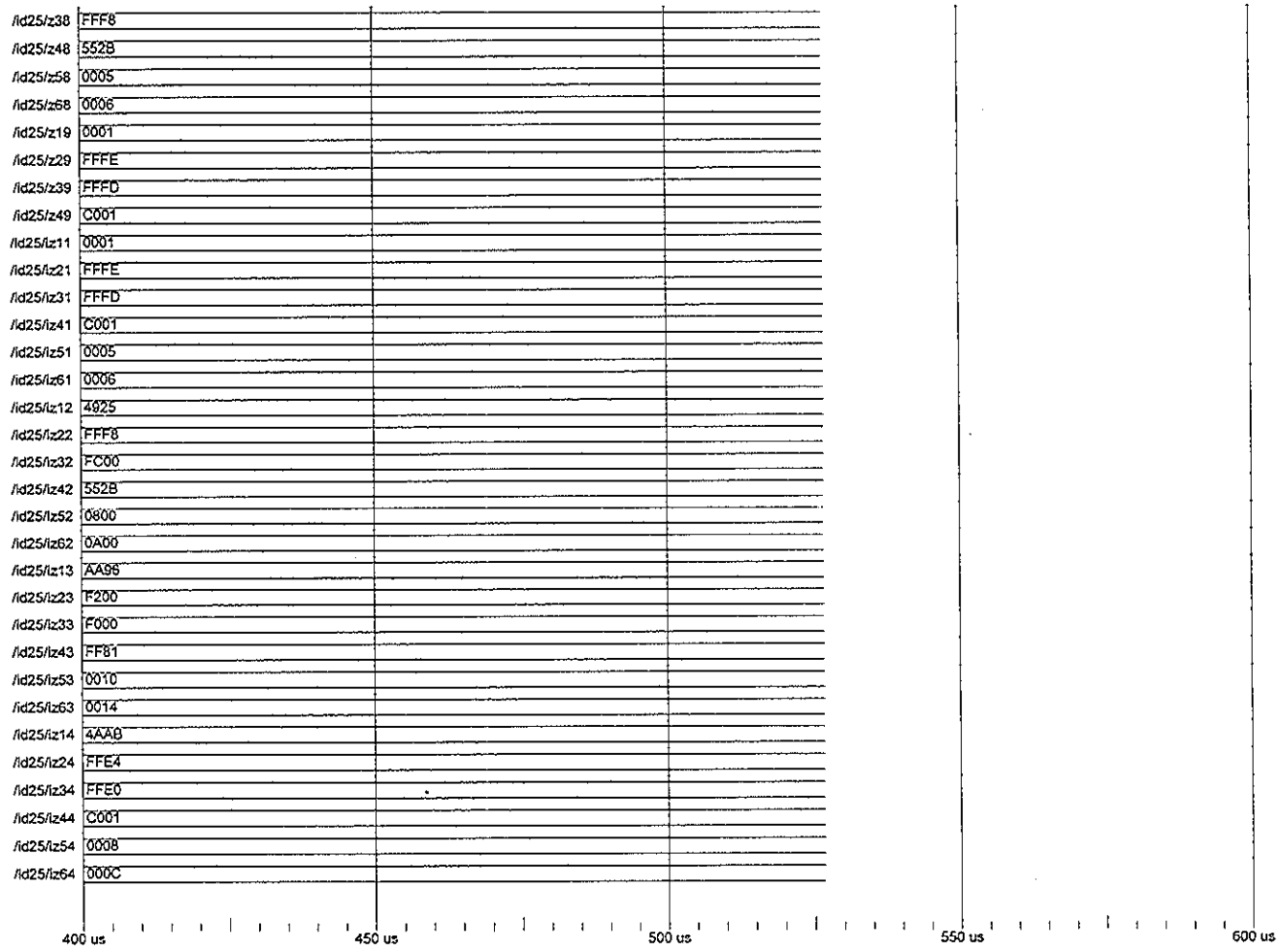
ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)



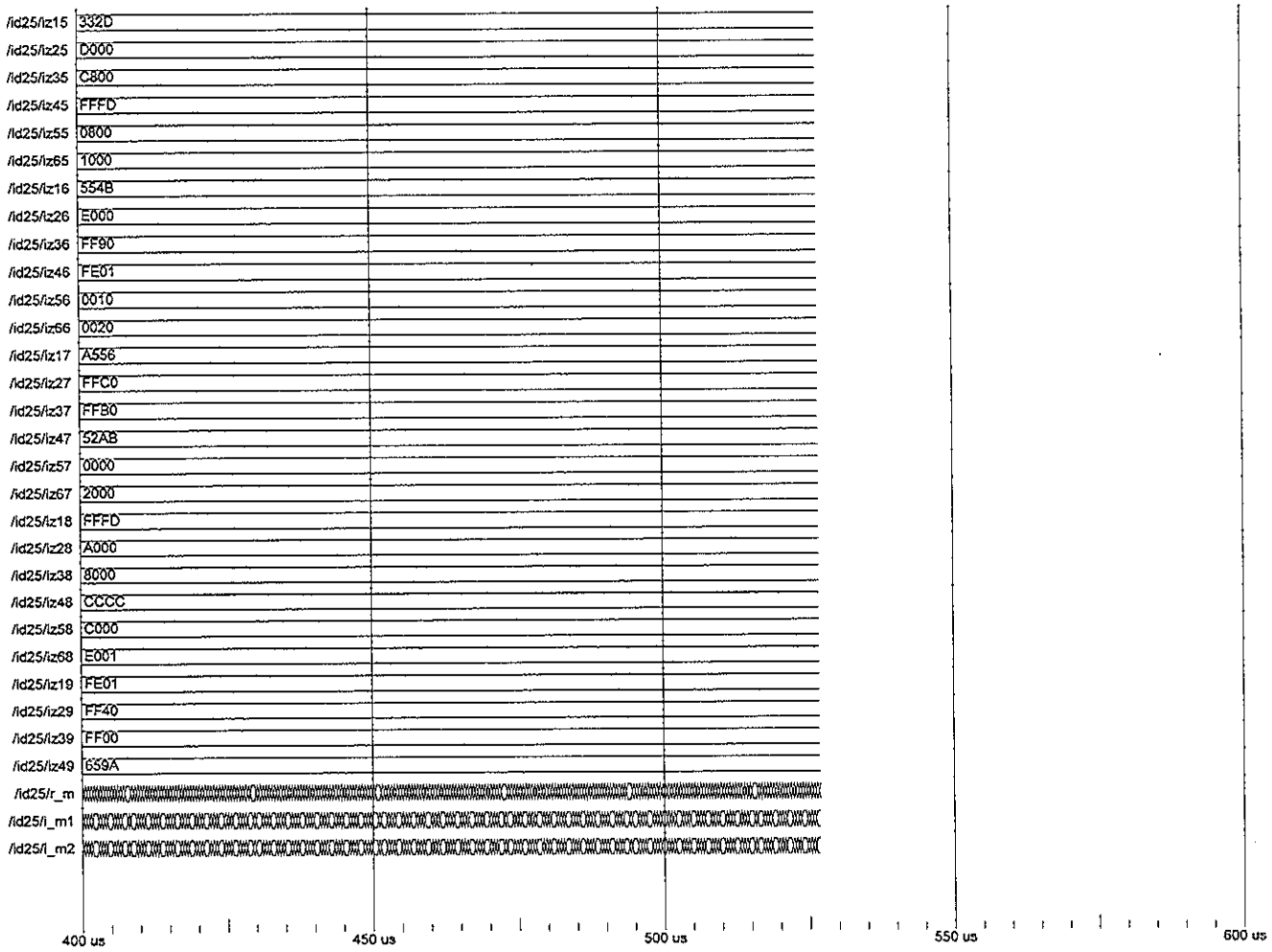
ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)



ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)

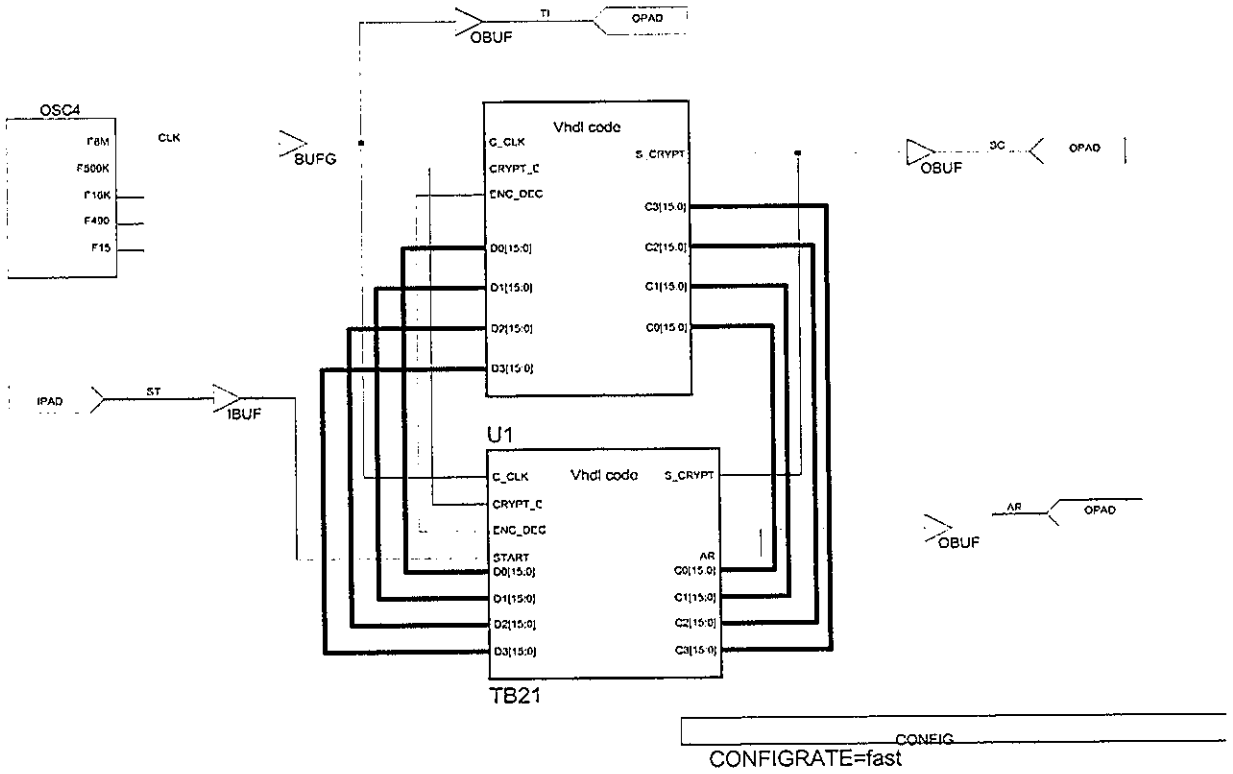


ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)

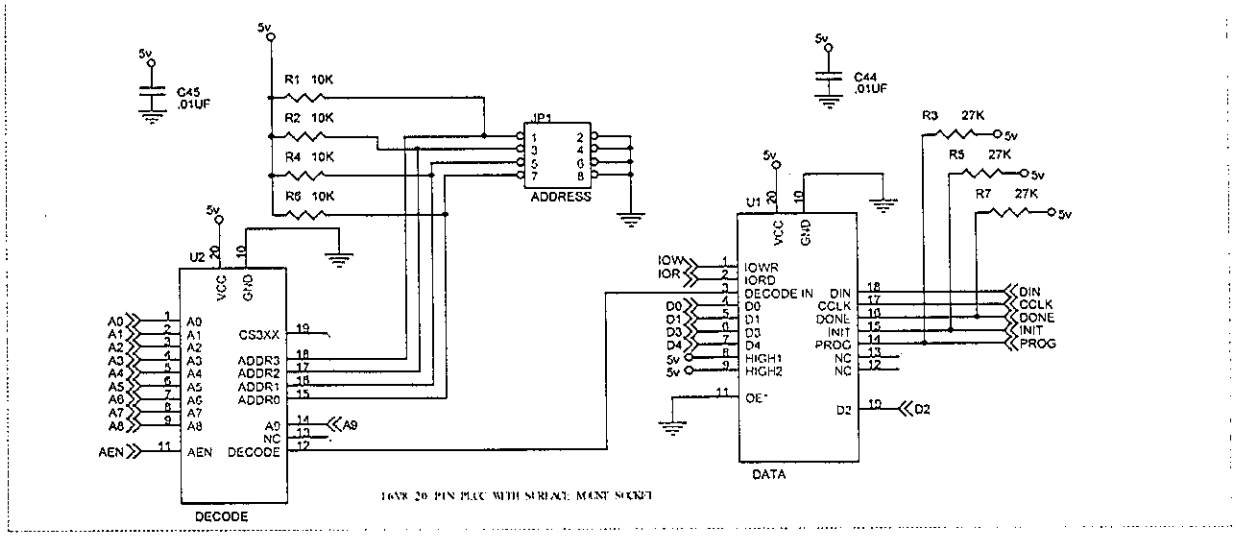


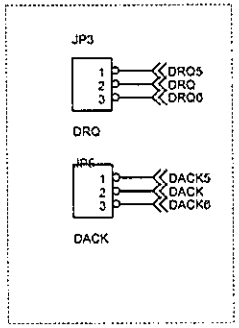
ภาพประกอบที่ ก3 ผลการจำลองสัญญาณทางฟังก์ชันของการถอดรหัส (ต่อ)

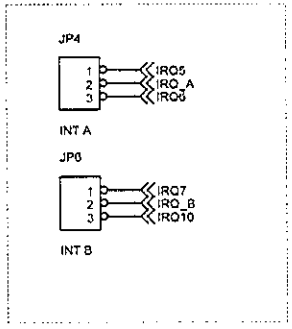
ภาคผนวก ข แผนผังทางไฟฟ้าของวงจรต่าง ๆ

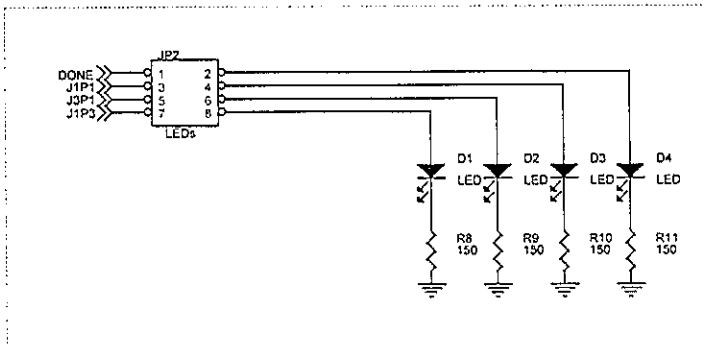


ภาพประกอบที่ ข1 แผนผังทางไฟฟ้าของชุดทดสอบการเข้า/ถอดรหัสด้วยสัญญาณนาฬิกาภายใน

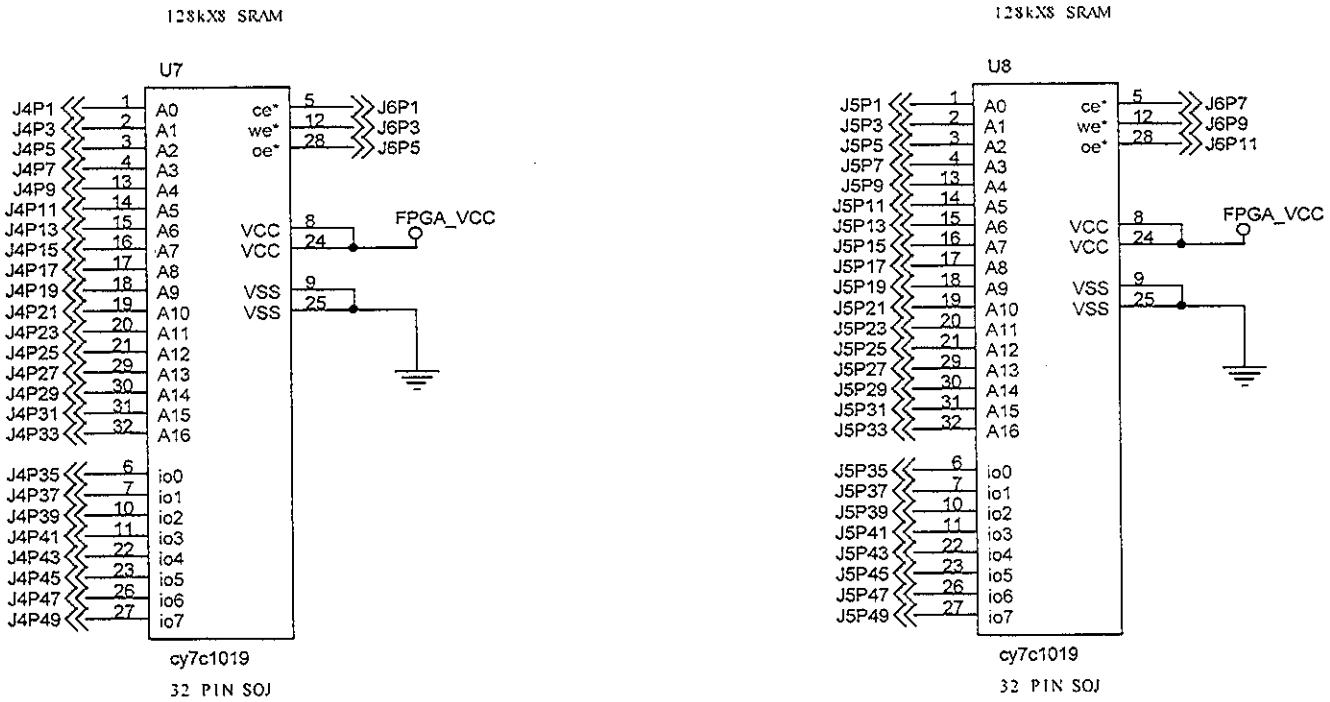




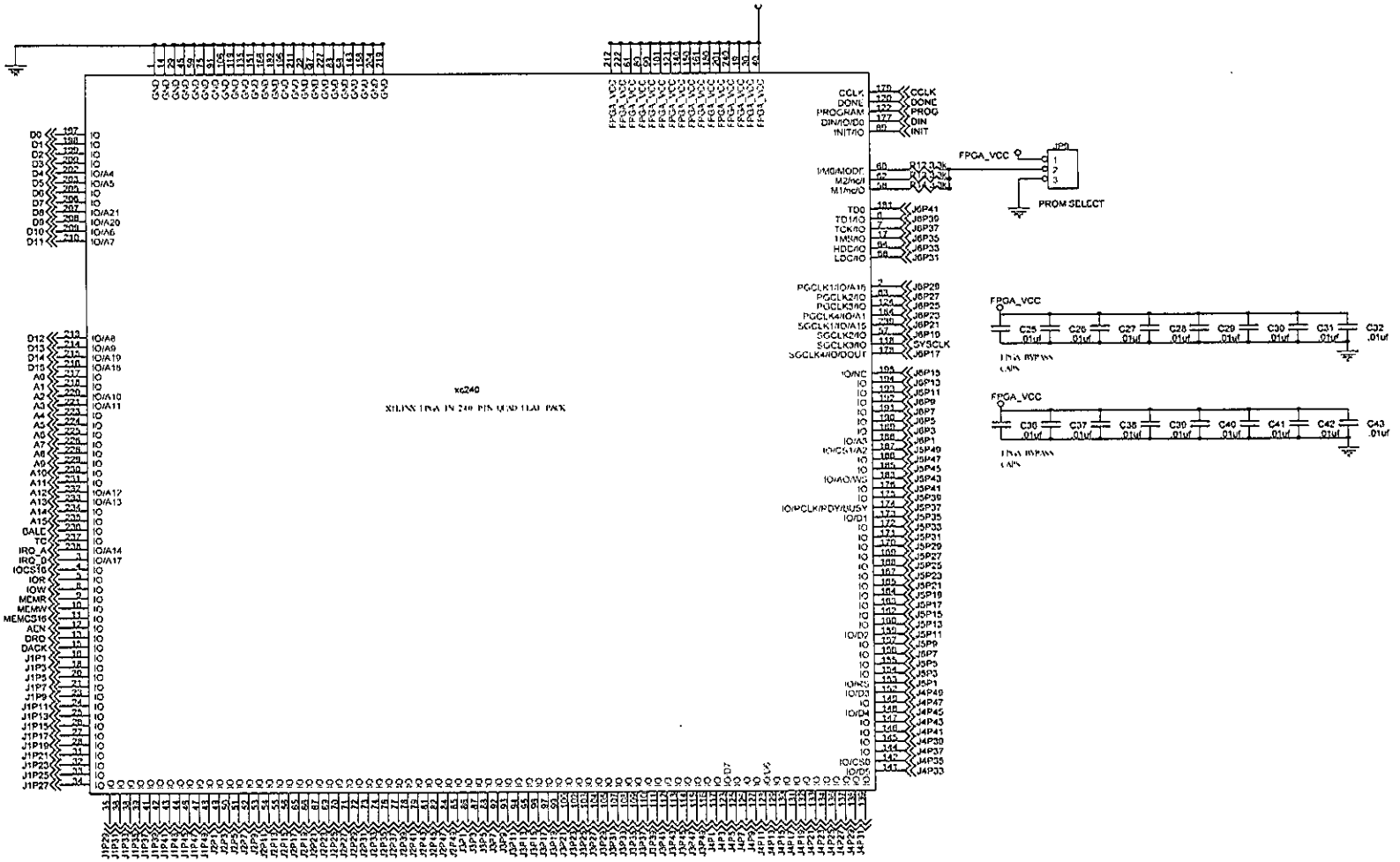




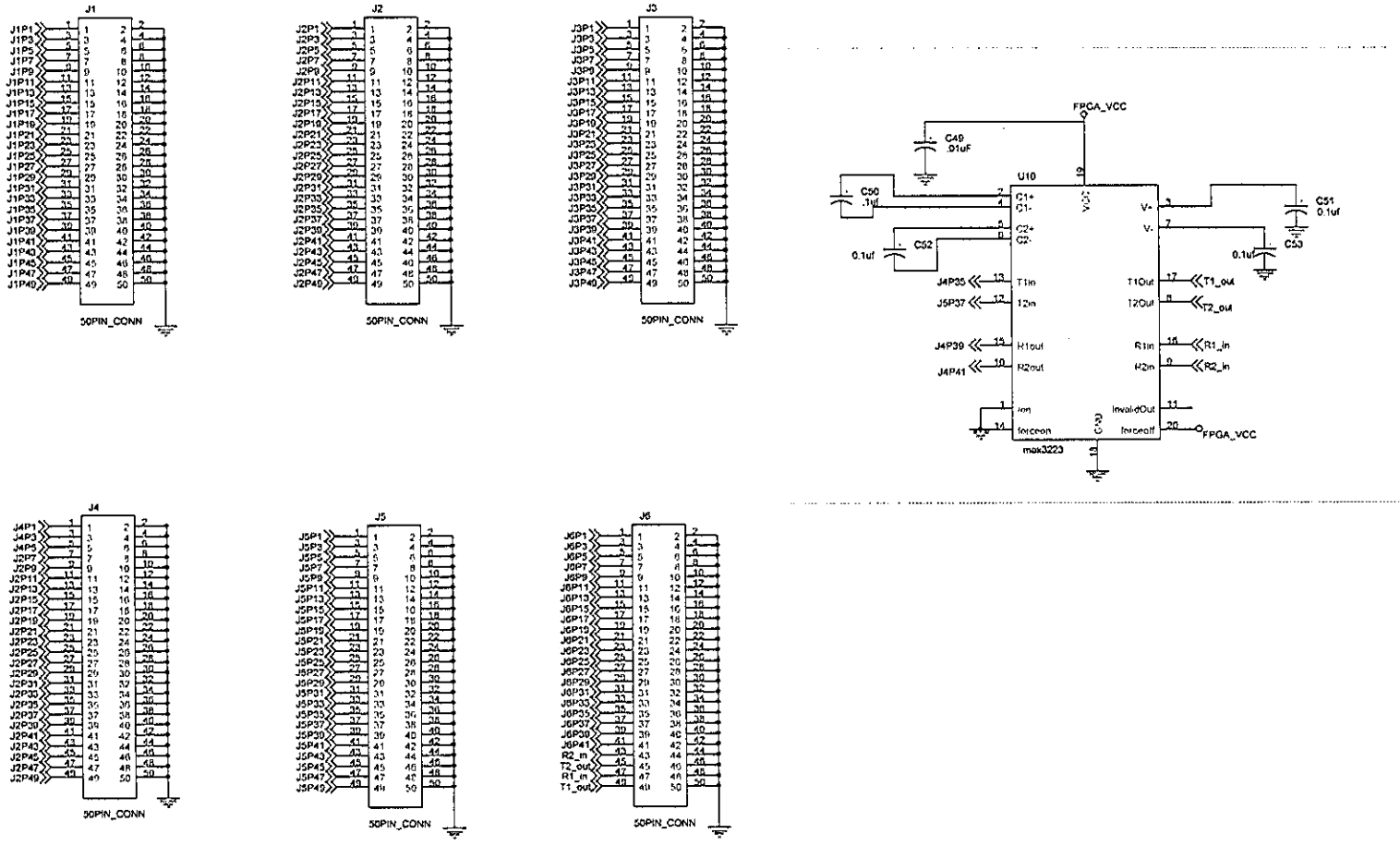
ภาพประกอบที่ ข2 แผนผังทางไฟฟ้าของวงจรรวมต้นแบบเอฟพีจีเอเบอร์ XC4062XLA09- HQ240



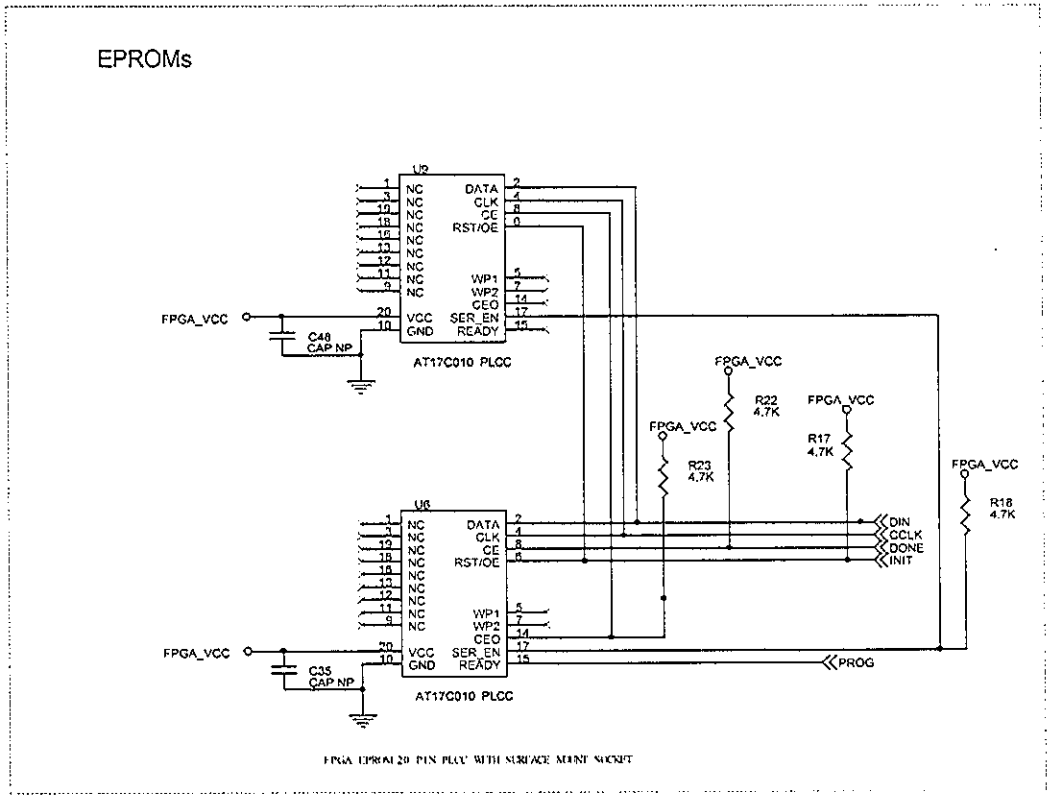
ภาพประกอบที่ ข2 แผนผังทางไฟฟ้าของวงจรรวมต้นแบบเอฟพีจีเอเบอร์ XC4062XLA09- HQ240 (ต่อ)

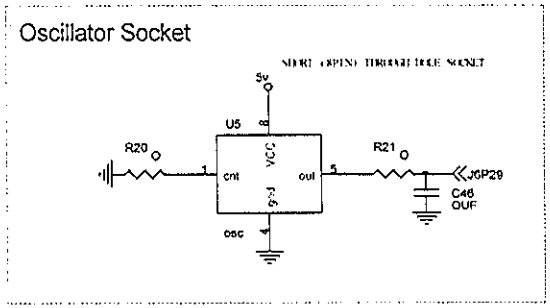


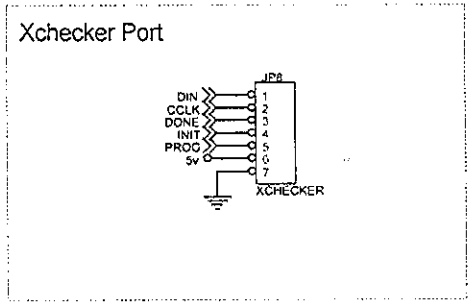
ภาพประกอบที่ ช2 แผนผังทางไฟฟ้าของวงจรรวมต้นแบบเอฟพีจีเอเบอร์ XC4062XLA09- HQ240 (ต่อ)



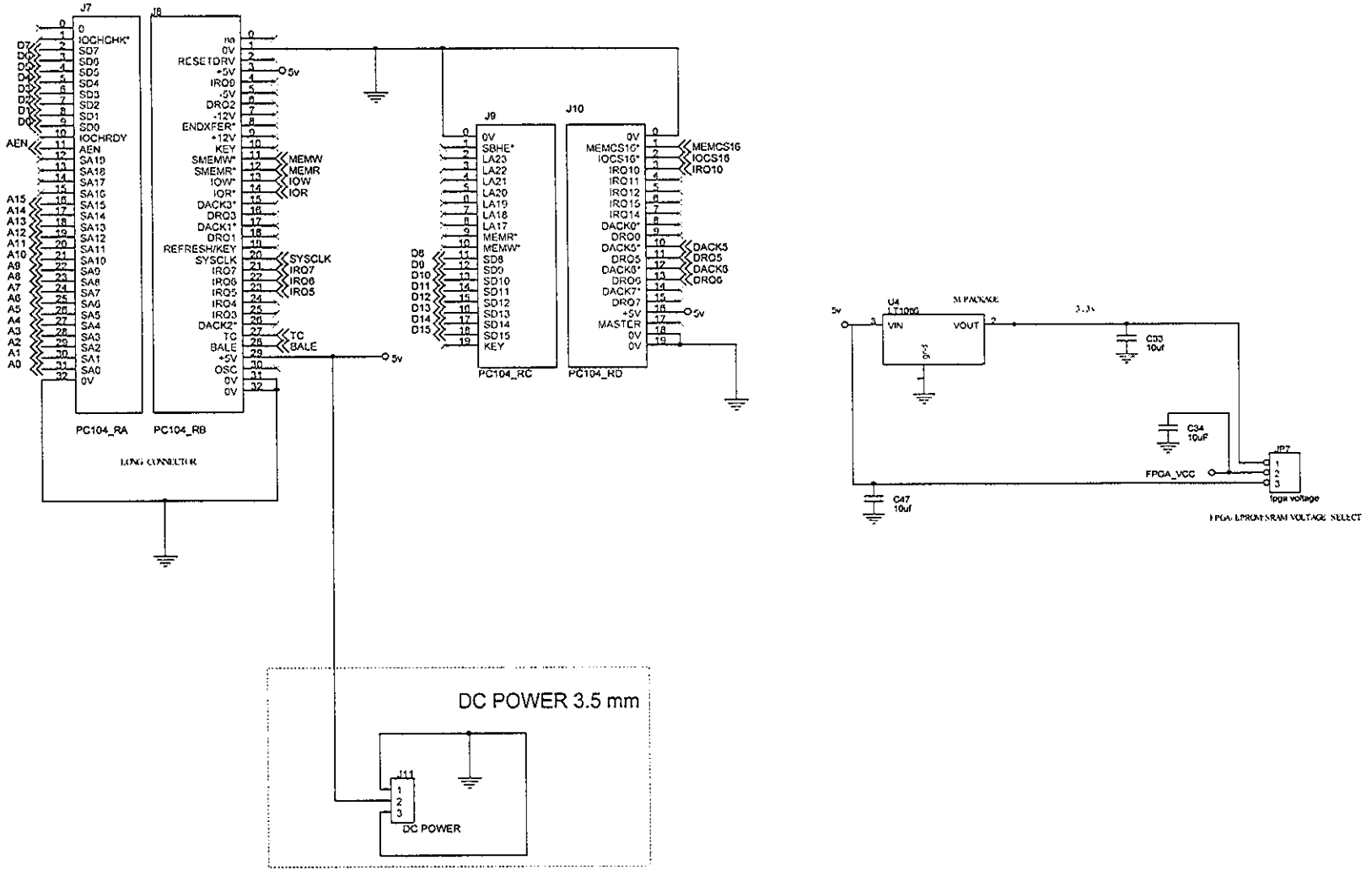
ภาพประกอบที่ ข2 แผนผังทางไฟฟ้าของวงจรรวมต้นแบบเอฟพีจีเอเบอร์ XC4062XLA09- HQ240 (ต่อ)







ภาพประกอบที่ ข2 แผนผังทางไฟฟ้าของวงจรรวมต้นแบบเอฟพีจีเอเบอร์ XC4062XLA09- HQ240 (ต่อ)



ภาพประกอบที่ ข2 แผนผังทางไฟฟ้าของวงจรรวมต้นแบบเอฟพีจีเอเบอร์ XC4062XLA09- HQ240 (ต่อ)

ภาคผนวก ค โปรแกรมรหัสต้นฉบับ

ค1. โปรแกรมรหัสต้นฉบับภาษาบรรยายฮาร์ดแวร์ของวงจรรวมต้นแบบ

```
    -- VHDL IEEE93 support
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   use IEEE.std_logic_arith.all;
4   use IEEE.std_logic_unsigned.all;
5
6   entity id25 is port
7   (
8           k0:in     std_logic_vector (15 downto 0);
9           k1:in     std_logic_vector(15 downto 0);
10          k2:in     std_logic_vector(15 downto 0);
11          k3:in     std_logic_vector(15 downto 0);
12          k4:in     std_logic_vector(15 downto 0);
13          k5:in     std_logic_vector(15 downto 0);
14          k6:in     std_logic_vector(15 downto 0);
15          k7:in     std_logic_vector(15 downto 0);
16          d0:in     std_logic_vector(15 downto 0);
17          d1:in     std_logic_vector(15 downto 0);
18          d2:in     std_logic_vector(15 downto 0);
19          d3:in     std_logic_vector(15 downto 0);
20          c_clk:in std_logic;
21          key_e:in std_logic;
22          crypt_e:in std_logic;
23          enc_dec:in std_logic;
24          s_key: out std_logic;
25          s_crypt: out std_logic;
26          c0:out    std_logic_vector(15 downto 0);
27          c1:out    std_logic_vector(15 downto 0);
28          c2:out    std_logic_vector(15 downto 0);
29          c3:out    std_logic_vector(15 downto 0));
30  end id25;
31
32  architecture behavior of id25 is
33  signal c_clk : std_logic;
34  signal z11,z21,z31,z41,z51,z61 : std_logic_vector(15 downto 0);
35  signal z12,z22,z32,z42,z52,z62 : std_logic_vector(15 downto 0);
36  signal z13,z23,z33,z43,z53,z63 : std_logic_vector(15 downto 0);
37  signal z14,z24,z34,z44,z54,z64 : std_logic_vector(15 downto 0);
38  signal z15,z25,z35,z45,z55,z65 : std_logic_vector(15 downto 0);
39  signal z16,z26,z36,z46,z56,z66 : std_logic_vector(15 downto 0);
40  signal z17,z27,z37,z47,z57,z67 : std_logic_vector(15 downto 0);
41  signal z18,z28,z38,z48,z58,z68 : std_logic_vector(15 downto 0);
42  signal z19,z29,z39,z49                  : std_logic_vector(15 downto 0);
43  signal iz11,iz41,iz12,iz42,iz13: std_logic_vector(15 downto 0);
44  signal iz43,iz14,iz44,iz15,iz45: std_logic_vector(15 downto 0);
45  signal iz16,iz46,iz17,iz47,iz18: std_logic_vector(15 downto 0);
46  signal iz48,iz19,iz49           : std_logic_vector(15 downto 0);
47  signal    r_m,i_m1,i_m2     : std_logic_vector (15 downto 0);
48  signal  inv_e,me,s_inv              : std_logic ;
49
50  constant zero_32 : std_logic_vector :="00000000000000000000000000000000";
51  constant zero_17 : std_logic_vector :="00000000000000000";
52  constant zero_16 : std_logic_vector :="0000000000000000";
53  constant max_16     : std_logic_vector :="1111111111111111";
54  constant two_bin : std_logic_vector :="0000000000000010";
55  constant one_bin : std_logic_vector :="0000000000000001";
56  constant fuyi       : std_logic_vector :="1000000000000000";
57  constant fermat  : std_logic_vector :="0100000000000001";
58
59  begin
60
61  main:process(c_clk,enc_dec,crypt_e,d0,d1,d2,d3,r_m,s_inv,z11,z21,z31,z41,
62  z51,z61,z12,z22,z32,z42,z52,z62,z13,z23,z33,z43,z53,z63,z14,z24,z34,z44,
63  z54,z64,z15,z25,z35,z45,z55,z65,z16,z26,z36,z46,z56,z66,z17,z27,z37,z47,
64  z57,z67,z18,z28,z38,z48,z58,z68,z19,z29,z39,z49,iz11,iz41,iz12,iz42,iz13,
65  iz43,iz14,iz44,iz15,iz45,iz16,iz46,iz17,iz47,iz18,iz48,iz19,iz49)
66
67  variable id0,id1,id2,id3,kc0,kc1,kc2,kc3,kc4,kc5 : std_logic_vector (15 downto 0);
68  variable id_stat: std_logic_vector (2 downto 0);
```

```
69      variable round : std_logic_vector (3 downto 0);
70      variable aprod1,aprod2,aprod3,aprod4: std_logic_vector (15 downto 0);
71      variable mprod1,mprod2,mprod3,mprod4: std_logic_vector (15 downto 0);
72      variable dz11,dz21,dz31,dz41,dz51,dz61 : std_logic_vector(15 downto 0);
73      variable dz12,dz22,dz32,dz42,dz52,dz62 : std_logic_vector(15 downto 0);
74      variable dz13,dz23,dz33,dz43,dz53,dz63 : std_logic_vector(15 downto 0);
75      variable dz14,dz24,dz34,dz44,dz54,dz64 : std_logic_vector(15 downto 0);
76      variable dz15,dz25,dz35,dz45,dz55,dz65 : std_logic_vector(15 downto 0);
77      variable dz16,dz26,dz36,dz46,dz56,dz66 : std_logic_vector(15 downto 0);
78      variable dz17,dz27,dz37,dz47,dz57,dz67 : std_logic_vector(15 downto 0);
79      variable dz18,dz28,dz38,dz48,dz58,dz68 : std_logic_vector(15 downto 0);
80      variable k0,k1,k2,k3,k4,k5,k6,k7 : std_logic_vector(15 downto 0);
81      variable ss :std_logic;
82
83      begin
84      if (crypt_e='1') or (key_e='1') then
85      ------------------------
86      -- init loop control
87      ----------------------
88                  id_stat :="000";
89      elsif (c_clk'event and c_clk='1') then
90       case id_stat is
91                  when "000" =>
92      ------------------------------------------------------------
93      -- init all acknowledge active low (synchr reset)
94      ------------------------------------------------------------
95                  id_stat :="001"; round := "0000";
96                  s_key <='1';  s_crypt <='1'; me<='1';  inv_e<='1';
97                  c0<=zero_16; c1<=zero_16; c2 <= zero_16; c3<=zero_16;
98      -------------
99      -- sub key1
100     -------------
101         dz11:= k0; dz21:=k1; dz31:=k2; dz41:=k3;dz51:=k4; dz61:=k5; dz12:=k6; dz22:=k7;
102                 dz32:=k1(6 downto 0) & k2(15 downto 7);
103                 dz42:=k2(6 downto 0) & k3(15 downto 7);
104                 dz52:=k3(6 downto 0) & k4(15 downto 7);
105                 dz62:=k4(6 downto 0) & k5(15 downto 7);
106                 dz13:=k5(6 downto 0) & k6(15 downto 7);
107                 dz23:=k6(6 downto 0) & k7(15 downto 7);
108                 dz33:=k7(6 downto 0) & k0(15 downto 7);
109                 dz43:=k0(6 downto 0) & k1(15 downto 7);
110                 dz53:=dz42(6 downto 0) & dz52(15 downto 7);
111                 dz63:=dz52(6 downto 0) & dz62(15 downto 7);
112                 dz14:=dz62(6 downto 0) & dz13(15 downto 7);
113                 dz24:=dz13(6 downto 0) & dz23(15 downto 7);
114                 dz34:=dz23(6 downto 0) & dz33(15 downto 7);
115                 dz44:=dz33(6 downto 0) & dz43(15 downto 7);
116                 dz54:=dz43(6 downto 0) & dz32(15 downto 7);
117                 dz64:=dz32(6 downto 0) & dz42(15 downto 7);
118                 dz15:=dz63(6 downto 0) & dz14(15 downto 7);
119                 dz25:=dz14(6 downto 0) & dz24(15 downto 7);
120                 dz35:=dz24(6 downto 0) & dz34(15 downto 7);
121                 dz45:=dz34(6 downto 0) & dz44(15 downto 7);
122                 dz55:=dz44(6 downto 0) & dz54(15 downto 7);
123                 dz65:=dz54(6 downto 0) & dz64(15 downto 7);
124                 dz16:=dz64(6 downto 0) & dz53(15 downto 7);
125                 dz26:=dz53(6 downto 0) & dz63(15 downto 7);
126                 dz36:=dz25(6 downto 0) & dz35(15 downto 7);
127                 dz46:=dz35(6 downto 0) & dz45(15 downto 7);
128                 dz56:=dz45(6 downto 0) & dz55(15 downto 7);
129                 dz66:=dz55(6 downto 0) & dz65(15 downto 7);
130                 dz17:=dz65(6 downto 0) & dz16(15 downto 7);
131                 dz27:=dz16(6 downto 0) & dz26(15 downto 7);
132                 dz37:=dz26(6 downto 0) & dz15(15 downto 7);
133                 dz47:=dz15(6 downto 0) & dz25(15 downto 7);
134                 dz57:=dz46(6 downto 0) & dz56(15 downto 7);
135                 dz67:=dz56(6 downto 0) & dz66(15 downto 7);
136                 dz18:=dz66(6 downto 0) & dz17(15 downto 7);
137                 dz28:=dz17(6 downto 0) & dz27(15 downto 7);
138                 dz38:=dz27(6 downto 0) & dz37(15 downto 7);
139                 dz48:=dz37(6 downto 0) & dz47(15 downto 7);
140                 dz58:=dz47(6 downto 0) & dz36(15 downto 7);
141                 dz68:=dz36(6 downto 0) & dz46(15 downto 7);
```

```
142                 z19<=dz67(6 downto 0) & dz18(15 downto 7);
143                 z29<=dz18(6 downto 0) & dz28(15 downto 7);
144                 z39<=dz28(6 downto 0) & dz38(15 downto 7);
145                 z49<=dz38(6 downto 0) & dz48(15 downto 7);
146     ---------------------
147     -- set encrypt key
148     ---------------------
149         z11<=dz11; z12<=dz12; z13<=dz13; z14<=dz14; z15<=dz15; z16<=dz16; z17<=dz17; z18<=dz18;
150                 z21<=dz21; z22<=dz22; z23<=dz23; z24<=dz24; z25<=dz25; z26<=dz26; z27<=dz27; z28<=dz28;
151                 z31<=dz31; z32<=dz32; z33<=dz33; z34<=dz34; z35<=dz35; z36<=dz36; z37<=dz37; z38<=dz38;
152                 z41<=dz41; z42<=dz42; z43<=dz43; z44<=dz44; z45<=dz45; z46<=dz46; z47<=dz47; z48<=dz48;
153                 z51<=dz51; z52<=dz52; z53<=dz53; z54<=dz54; z55<=dz55; z56<=dz56; z57<=dz57; z58<=dz58;
154                 z61<=dz61; z62<=dz62; z63<=dz63; z64<=dz64; z65<=dz65; z66<=dz66; z67<=dz67; z68<=dz68;
155         s_key <='0';  id_stat :="001";
156
157       when "001" =>
158                 if enc_dec ='1' then
159                   if s_inv ='1' then
160                                 inv_e <='0'; -- activate inv_mul
161                                 id_stat :="001";
162                               else
163                                 inv_e <='1'; -- deactivate inv_mul
164                                 id_stat :="010";
165                               end if;
166                 else
167                               id_stat :="010";
168                 end if;
169                when "010" =>
170                 s_crypt<='1'; ss:='1';
171                               if enc_dec ='0' then
172                                case round is -- encrypt
173                                when "0000" => -- r1 key
174                                kc0:=z11; kc1:=z21; kc2:=z31; kc3:=z41; kc4:=z51; kc5:=z61;
175                                when "0001" => -- r2 key
176                                kc0:=z12; kc1:=z22; kc2:=z32; kc3:=z42; kc4:=z52; kc5:=z62;
177                                when "0010" => -- r3 key
178                                kc0:=z13; kc1:=z23; kc2:=z33; kc3:=z43; kc4:=z53; kc5:=z63;
179                                when "0011" => -- r4 key
180                                kc0:=z14; kc1:=z24; kc2:=z34; kc3:=z44; kc4:=z54; kc5:=z64;
181                                when "0100" => -- r5 key
182                                kc0:=z15; kc1:=z25; kc2:=z35; kc3:=z45; kc4:=z55; kc5:=z65;
183                                when "0101" => -- r6 key
184                                kc0:=z16; kc1:=z26; kc2:=z36; kc3:=z46 ; kc4:=z56; kc5:=z66;
185                                when "0110" => -- r7 key
186                                kc0:=z17; kc1:=z27; kc2:=z37; kc3:=z47; kc4:=z57; kc5:=z67;
187                                when "0111" => -- r8 key
188                                kc0:=z18; kc1:=z28; kc2:=z38; kc3:=z48; kc4:=z58; kc5:=z68;
189                                when "1000" => -- r9 key
190                                kc0:=z19; kc1:=z29; kc2:=z39; kc3:=z49;
191                                when others => null;
192                                end case;
193                               else
194                                case round is -- decrypt
195                                when "0000" => -- r1 key
196                                kc0:=iz19; kc1:=(max_16-z29)+one_bin; kc2:=(max_16-z39)+one_bin;
197                                kc3:=iz49; kc4:=dz58; kc5:=dz68;
198                                when "0001" => -- r2 key
199                                kc0:=iz18; kc1:=(max_16-dz38)+one_bin; kc2:=(max_16-dz28)+one_bin;
200                                kc3:=iz48; kc4:=dz57; kc5:=dz67;
201                                when "0010" => -- r3 key
202                                kc0:=iz17; kc1:=(max_16-dz37)+one_bin; kc2:=(max_16-dz27)+one_bin;
203                                kc3:=iz47; kc4:=dz56; kc5:=dz66;
204                                when "0011" => -- r4 key
205                                kc0:=iz16; kc1:=(max_16-dz36)+one_bin; kc2:=(max_16-dz26)+one_bin;
206                                kc3:=iz46; kc4:=dz55; kc5:=dz65;
207                                when "0100" => -- r5 key
208                                kc0:=iz15; kc1:=(max_16-dz35)+one_bin; kc2:=(max_16-dz25)+one_bin;
209                                kc3:=iz45; kc4:=dz54; kc5:=dz64;
210                                when "0101" => -- r6 key
211                                kc0:=iz14; kc1:=(max_16-dz34)+one_bin; kc2:=(max_16-dz24)+one_bin;
212                                kc3:=iz44; kc4:=dz53; kc5:=dz63;
213                                when "0110" => -- r7 key
214                                kc0:=iz13; kc1:=(max_16-dz33)+one_bin; kc2:=(max_16-dz23)+one_bin;
```

```
                        kc3:=iz43; kc4:=dz52; kc5:=dz62;
                        when "0111" => -- r8 key
                        kc0:=iz12; kc1:=(max_16-dz32)+one_bin; kc2:=(max_16-dz22)+one_bin;
                        kc3:=iz42; kc4:=dz51; kc5:=dz61;
                        when "1000" => -- r9 key
                        kc0:=iz11; kc1:=(max_16-dz21)+one_bin; kc2:=(max_16-dz31)+one_bin;
                        kc3:=iz41;
                        when others => null;
                        end case;
                      end if;

                      if round="0000" then id0:=d0; id1:=d1; id2:=d2; id3:=d3; -- 9 round finish
                      else null;
                      end if;

                      id_stat := "011"; i_m1<=kc0; i_m2<=id0 ; me <='0'; -- activate mul
                      if round ="1000" then aprod1:=id2+kc1;
                      else aprod1:=id1+kc1; aprod3:=aprod1;
                      end if;

            when "011" =>
                      id_stat :="100"; mprod1:=r_m; mprod3:=mprod1; i_m1<=kc3; i_m2<=id3 ; me<='0';
                      if round ="1000" then          aprod2:=id1+kc2;
                      else aprod2:=id2+kc2; aprod4:=aprod2;
                      end if;

    when "100" =>
                      mprod2:=r_m; mprod4:=mprod2;
                      if round = "1000" then
            me<='1'; -- deactivate mul
                       s_crypt <='0';       ss:='0';
                      else
                       i_m1<=kc4 ;i_m2<=(mprod1 xor aprod2) ; me<='0'; -- activate mul
                      end if;
                      id_stat:="101";

            when "101" =>
                      if ss='0' then
                       c0<=mprod1;c1<=aprod1;c3<=mprod2; c2<=aprod2;
                       id_stat := "010" ;round:="0000";
                      else
                       id_stat :="110"; mprod1:=r_m; i_m1<= kc5;
                       i_m2<= (r_m+(aprod1 xor mprod2));
                      end if;

            when "110" =>
                      id_stat :="010"; me<='1'; -- deactivate mul
                      id0 := (r_m xor mprod3); id1 := (r_m xor aprod4);
                      id2 := ((r_m+mprod1) xor aprod3); id3 := ((r_m+mprod1) xor mprod4);
                      round :=round+"0001";

            when others => null;

            end case;

else null;
end if;

end process main;



inv_mul:process (enc_dec,crypt_e,c_clk,inv_e,z11,z41,z12,z42,z13,z43,
z14,z44,z15,z45,z16,z46,z17,z47,z18,z48,z19,z49)

variable c          :std_logic_vector (15 downto 0);
variable ss,ex      :std_logic;
variable u,v,b,d:std_logic_vector (17 downto 0);
variable bs,ds      :std_logic; -- sign bit
variable iz_a       :std_logic_vector (1 downto 0);
variable count      :std_logic_vector (4 downto 0);

begin
```

```
288   if (inv_e='1' and crypt_e='0' and enc_dec='1') then
289    b:="0000000000000000" ; d:="0000000000000000" ; bs:='0' ;ds:='0';
290    u:=fermat; iz_a :="00"; c:= "0000000000000000"; v:=b;
291    count:="00000"; s_inv<='1'; ss:='0';ex:='1';
292   else if ((inv_e='0'and crypt_e='0') and enc_dec='1' ) then
293    if (c_clk'event and c_clk='1') then
294
295   -- polling inv mod mul
296                case count is
297       when "00000" =>
298                count :="00001"; v(15 downto 0):=z11; ex:= ex xor ex;
299
300                when "00001" =>
301                if ss='1' then
302                  count :="00010"; iz11 <=c; v(15 downto 0):=z41;
303                else count :="00001";
304                end if;
305
306                when "00010" =>
307                if ss='1' then
308                        count :="00011";
309                iz41 <=c; v(15 downto 0):=z12;
310                else count :="00010";
311                end if;
312
313                when "00011" =>
314                if ss='1' then
315                        count :="00100"; iz12 <=c; v(15 downto 0):=z42;
316                else count :="00011";
317                end if;
318
319                when "00100" =>
320                if ss='1' then
321                        count :="00101"; iz42 <=c; v(15 downto 0):=z13;
322                else count :="00100";
323                end if;
324
325                when "00101" =>
326                if ss='1' then
327                        count :="00110"; iz13 <=c;        v(15 downto 0):=z43;
328                else count :="00101";
329                end if;
330
331                when "00110" =>
332                if ss='1' then
333                        count :="00111"; iz43 <=c; v(15 downto 0):=z14;
334               else count :="00110";
335               end if;
336
337               when "00111" =>
338               if ss='1' then
339                        count :="01000"; iz14 <=c; v(15 downto 0):=z44;
340               else count :="00111";
341               end if;
342
343               when "01000" =>
344               if ss='1' then
345                        count :="01001"; iz44 <=c; v(15 downto 0):=z15;
346               else count :="01000";
347               end if;
348
349               when "01001" =>
350               if ss='1' then
351                        count :="01010"; iz15 <=c; v(15 downto 0):=z45;
352               else count :="01001";
353               end if;
354
355               when "01010" =>
356               if ss='1' then
357                        count :="01011"; iz45 <=c; v(15 downto 0):=z16;
358              ' else count :="01010";
359               end if;
360
```

```
361                 when "01011" =>
362                 if ss='1' then
363                         count :="01100"; iz16 <=c; v(15 downto 0):=z46;
364                 else count :="01011";
365                 end if;
366
367                 when "01100" =>
368                 if ss='1' then
369                         count :="01101"; iz46 <=c; v(15 downto 0):=z17;
370                 else count :="01100";
371                 end if;
372
373                 when "01101" =>
374                 if ss='1' then
375                         count :="01110"; iz17 <=c; v(15 downto 0):=z47;
376                 else count :="01101";
377                 end if;
378
379                 when "01110" =>
380                 if ss='1' then
381                         count :="01111"; iz47 <=c;      v(15 downto 0):=z18;
382                 else count :="01110";
383                 end if;
384
385                 when "01111" =>
386                 if ss='1' then
387                         count :="10000"; iz18 <=c;      v(15 downto 0):=z48;
388                 else count :="01111";
389                 end if;
390
391                 when "10000" =>
392                 if ss='1' then
393                         count :="10001"; iz48 <=c;      v(15 downto 0):=z19;
394                 else count :="10000";
395                 end if;
396
397                 when "10001" =>
398                 if ss='1' then
399                         count :="10010"; iz19 <=c;      v(15 downto 0):=z49;
400                 else count :="10001";
401                 end if;
402
403                 when "10010" =>
404                 if ss='1' then
405                         count :="10011"; iz49 <=c;      s_inv<='0';
406                 else count :="10010";
407                 end if;
408
409                 when others => null;
410                 end case;
411
412     -- Bin Ext Euc
413      if (ex='0' and(v ="0000000000000000")) then        c:= zero_16;
414      elsif (ex='0' and(v/="0000000000000000"))then
415      case iz_a is
416                 when "00" =>
417                  ss:='0';
418                  if u(0) = '0' then -- even chk
419                   iz_a :="00"; u:="00" & u(16 downto 1) ; --shr for div2
420                         if b(0)='1' then
421                           if bs='1' then b:=fermat+b; -- 2's comp b
422                           else b:=fermat-b;
423                           end if;
424                         bs:='1';
425                         else null;
426                         end if;
427                         b:="00"& b(16 downto 1) ;
428                  else iz_a :="01";
429                  end if;
430
431                 when "01" =>
432                  ss:='0';
433                  if v(0) = '0' then -- even chk
```

```
434                 iz_a :="01"; v:="00" & v(16 downto 1) ; -- shr for div 2
435                 if d(0)='1' then
436                         if ds='1' then d:=fermat+d; -- 2's comp d
437                         else d:=fermat-d;
438                         end if;
439                 ds:='1';
440                 else null;
441                 end if;
442                 d:="00" & d(16 downto 1);
443                 else iz_a :="10";
444                 end if;
445
446                 when "10" =>
447                 iz_a := "00";
448                 if u=v then
449                         ss := '1';
450                         if ds='0' then c:=d(15 downto 0);
451                         else
452                           d:=fermat-d; c:=d(15 downto 0);
453                         end if;
454     -- clr reg
455                         b:= b xor b ; d:="00000000000000000001" ;
456                         bs:= bs xor bs ;ds:= ds xor ds; u:=fermat; v := v xor v;
457                 else
458                         ss:='0';
459                         if (u>v) then
460                          u:=u-v;
461                           if ((bs='0' and ds='0') and b>d) then          b:=b-d;
462                           elsif ((bs='0' and ds='0') and b<d) then       b:=d-b; bs:='1';
463                           elsif bs='0' and ds='1' then b:=b+d;
464                           elsif bs='1' and ds='0' then b:=b+d;
465                           elsif bs='1' and ds='1' and b>d then b:=b-d;
466                           else        b:=d-b; bs:= bs xor bs;
467                           end if;
468                         else
469                          v:=v-u;
470                           if bs='0' and ds='0' and d>b then d:=d-b;
471                           elsif bs='0' and ds='0' and d<b then d:=b-d; ds:='1';
472                           elsif ds='0' and bs='1' then d:=d+b;
473                           elsif ds='1' and bs='0' then d:=d+b;
474                           elsif ds='1' and bs='1' and d>b then d:=d-b;
475                           else d:=b-d; ds:='0';
476                           end if;
477                         end if;
478                 end if;
479
480                 when others =>        null ;
481                 end case;
482       else null;
483       end if;
484      else null;
485      end if;
486     else null;
487     end if;
488    end if;
489    end process inv_mul;
490
491    mult:process (i_m1,i_m2,me,crypt_e)
492
493    variable temp1          : std_logic_vector (31 downto 0);
494    variable l1 : std_logic_vector (16 downto 0);
495    variable l2 : std_logic_vector (16 downto 0);
496    variable l3 : std_logic_vector (16 downto 0);
497    variable l4 : std_logic_vector (17 downto 0);
498    variable result_mult : std_logic_vector (31 downto 0);
499    variable modp,divp    : std_logic_vector(15 downto 0);
500
501    begin
502    if (me='0' and crypt_e='0') then
503    -- init
504      l1 := zero_17;
505      l2 :='0'&i_m1 ; l3:=i_m1 & '0';
506      l4 := l2+('0'&l3);
```

```
507     result_mult:=zero_32;
508     temp1:=zero_32;
509
510   -- Encode
511     case i_m2(1 downto 0) is
512     when "00" => result_mult(16 downto 0):=l1;
513     when "01" => result_mult(16 downto 0):=l2;
514     when "10" => result_mult(16 downto 0):=l3;
515     when others =>result_mult(17 downto 0):=l4;
516     end case;
517
518     case i_m2(3 downto 2) is
519     when "00" => temp1(18 downto 2):=l1;
520     when "01" => temp1(18 downto 2):=l2;
521     when "10" => temp1(18 downto 2):=l3;
522     when others =>temp1(19 downto 2):=l4;
523     end case;
524     result_mult:=temp1+result_mult;
525     temp1:=zero_32;
526
527     case i_m2(5 downto 4) is
528     when "00" => temp1(20 downto 4):=l1;
529     when "01" => temp1(20 downto 4):=l2;
530     when "10" => temp1(20 downto 4):=l3;
531     when others => temp1(21 downto 4):=l4;
532     end case;
533     result_mult:=temp1+result_mult;
534     temp1:=zero_32;
535
536     case i_m2(7 downto 6) is
537     when "00" => temp1(22 downto 6):=l1;
538     when "01" => temp1(22 downto 6):=l2;
539     when "10" => temp1(22 downto 6):=l3;
540     when others => temp1(23 downto 6):=l4;
541     end case;
542     result_mult:=temp1+result_mult;
543     temp1:=zero_32;
544
545     case i_m2(9 downto 8) is
546     when "00" => temp1(24 downto 8):=l1;
547     when "01" => temp1(24 downto 8):=l2;
548     when "10" => temp1(24 downto 8):=l3;
549     when others => temp1(25 downto 8):=l4;
550     end case;
551     result_mult:=temp1+result_mult;
553     temp1:=zero_32;
553
554     case i_m2(11 downto 10) is
555     when "00" => temp1(26 downto 10):=l1;
556     when "01" => temp1(26 downto 10):=l2;
557     when "10" => temp1(26 downto 10):=l3;
558     when others => temp1(27 downto 10):=l4;
559     end case;
560     result_mult:=temp1+result_mult;
561     temp1:=zero_32;
562
563     case i_m2(13 downto 12) is
564     when "00" => temp1(28 downto 12):=l1;
565     when "01" => temp1(28 downto 12):=l2;
566     when "10" => temp1(28 downto 12):=l3;
567     when others => temp1(29 downto 12):=l4;
568     end case;
569     result_mult:=temp1+result_mult;
570     temp1:=zero_32;
571
572     case i_m2(15 downto 14) is
573     when "00" => temp1(30 downto 14):=l1;
574     when "01" => temp1(30 downto 14):=l2;
575     when "10" => temp1(30 downto 14):=l3;
576     when others => temp1(31 downto 14):=l4;
577     end case;
578     result_mult:=temp1+result_mult;
579   -------------------------
```

```
580     -- High low order algo-
581     -------------------------
582       modp:=result_mult(15 downto 0);
583       divp:=result_mult(31 downto 16);
584       if result_mult=zero_32 then r_m<=(max_16-i_m1-i_m2)+two_bin;
585       elsif modp>divp then              r_m<=modp-divp;
586       elsif divp=modp then              r_m<=one_bin;
587       elsif (divp-modp)=one_bin then r_m<=zero_16;
588       else      r_m<=(((max_16-divp)+modp)+two_bin);
589       end if;
590
591     else r_m<=zero_16;
592     end if;
593     end process mult;
594
595     end behavior;
```

ค2. โปรแกรมรหัสต้นฉบับภาษาซีการประมวลผลรหัสลับ

```
1   #include <stdio.h>
2   typedef unsigned int u_int16;
3   typedef unsigned long u_int32;
4   void Idea(u_int16 *in, u_int16 *out, u_int16 *key)
5   {
6           u_int16 x0, x1, x2, x3, t0, t1, round;
7           x0 = *in++;
8           x1 = *in++;
9           x2 = *in++;
10          x3 = *in;
11          for (round = 0; round < 8; round++) {
12                  x0 *= *key++;
13                  x1 += *key++;
14                  x2 += *key++;
15                  x3 *= *key++;
16                  t0  = x1;
17                  t1  = x2;
18                  x2 ^= x0;
19                  x1 ^= x3;
20                  x2 *= *key++;
21                  x1 += x2;
22                  x1 *= *key++;
23                  x2 += x1;
24                  x0 ^= x1;
25                  x3 ^= x2;
26                  x1 ^= t1;
27                  x2 ^= t0;
28          }
29          *out++ = x0 * *key++;
30          *out++ = x2 + *key++;  /* NB: Order */
31          *out++ = x1 + *key++;
32          *out   = x3 * *key;
33  }
34
35  u_int16 mul(u_int16 x, u_int16 y)
36  {
37          u_int32 p;
38          p=(u_int32)x*y;
39          if (p == 0)
40                  x = 65537l-x-y;
41          else {
42                  x = p >> 16;
43                  y = p;
44                  x = y-x;
45                  if (y < x) x += 65537l;
46          }
47          return x;
48  }
49
50  void Expandkey(u_int16 *ukey, u_int16 *key)
51  {
52          int i;
53          for (i=0; i<8; i++) key[i]=ukey[i];
54          for (i=8; i<52; i++) {
55                  if ((i & 7) < 6)
56                          key[i]=(key[i-7] & 127) << 9 | key[i-6] >> 7;
57                  else if ((i & 7) == 6)
58                          key[i]=(key[i-7] & 127) << 9 | key[i-14] >> 7;
59                  else
60                          key[i]=(key[i-15] & 127) << 9 | key[i-14] >> 7;
61          }
62  }
63
64  u_int16 addinv(u_int16 x)
65  {
66          return 0-x;
67  }
68
69  /*
70
```

```
71    u_int16 mulinv (u_int y)
72    {           long  u, v, a, b, c, d ;
73    u = 655371; v = y ; a = 1 ; b = 0 ; c = 0 ; d = 1 ;
74    do {
75    do {        u = u mod2 ;
76    if ((a mod 2 == b mod 2) == 0) {
77                        a = a mod 2;   b = b mod 2;
78    } else {    a = (a+y ) mod 2;  b = (b-x ) mod 2 ;
79    }
80    } while (u mod 2 == 0) ;
81    do {        v = v mod 2;
82                if (( c mod 2 == d mod 2) == 0 ) {
83                        c = c mod 2; d = d mod 2 ;
84    }else {     c = (c + y) mod 2; d = (d - x)mod 2;
85                }
86    } while (v mod 2 == 0) ;
87    if u >= v {u = u-v ; a = a-c ; b = b-d ;
88    } else {    v=v-u ; c = c-a ;  d = d-b;
89    }
90    }while ( u = 0) ;
91    return (u_int16)d;
92    }
93
94    void Invertkey(u_int16 *in, u_int16 *out)
95    {
96              u_int16 t1, t2, t3, t4, round;
97              u_int16 *p;
98               p= out + 52; /* work backwards */
99              t1 = mulinv(*in++);
100             t2 = addinv(*in++);
101             t3 = addinv(*in++);
102             t4 = mulinv(*in++);
103             *--p = t4;
104             *--p = t3;
105             *--p = t2;
106             *--p = t1;
107
108             for (round = 1; round < 8; round++) {
109                       t1 = *in++;
110                       t2 = *in++;
111                       *--p = t2;
112                       *--p = t1;
113
114                       t1 = mulinv(*in++);
115                       t2 = addinv(*in++);
116                       t3 = addinv(*in++);
117                       t4 = mulinv(*in++);
118                       *--p = t4;
119                       *--p = t2; /* NB: Order */
120                       *--p = t3;
121                       *--p = t1;
122             }
123             t1 = *in++;
124             t2 = *in++;
125             *--p = t2;
126             *--p = t1;
127
128             t1 = mulinv(*in++);
129             t2 = addinv(*in++);
130             t3 = addinv(*in++);
131             t4 = mulinv(*in++);
132             *--p = t4;
133             *--p = t3;
134             *--p = t2;
135             *--p = t1;
136   }
137
138   main()
139   {
140             u_int16  Key[8]={ 1, 2, 3, 4, 5, 6, 7, 8 };
141             u_int16  KeyOut[64], InvKeyIn[64], InvKeyOut[64];
142             int i;
143
```

```
144             Expandkey(Key, KeyOut);
145             printf("Key in :\n%X %X %X %X   %X %X %X %X\n",
146                     Key[0], Key[1],Key[2],Key[3],Key[4],Key[5],Key[6],Key[7]);
147             for(i=0; i<52; ++i) {
148                     printf("Key [%d]:%X\n", i+1, KeyOut[i]);
149             }
150             Invertkey(KeyOut, InvKeyOut);
151             for(i=0; i<52; ++i) {
152                     printf("Inverse Key [%d]:%X\n", i+1, InvKeyOut[i]);
153             }
154     }
```

ภาคผนวก ง รายงานสมบูรณ์ของการสังเคราะห์วงจร

Performance Summary

Clock	Requested Period (ns)	Estimated Period (ns)	Max Slack (ns)	Worst Slack (ns)
c_clk	1000.0	59.5	940.5	940.5

Interface information

Input ports:

Port Name	Reference Clock	Max Required Time (ns)	Max Slack(ns)
crypt_e	c_clk	971.8	971.8
d0[15:0]	c_clk	992.4	992.4
d1[15:0]	c_clk	987.6	989.0
d2[15:0]	c_clk	988.8	989.2
d3[15:0]	c_clk	995.6	995.6
enc_dec	c_clk	958.0	958.0
k0[15:0]	c_clk	994.6	994.6
k1[15:0]	c_clk	994.4	994.4
k2[15:0]	c_clk	994.5	994.5
k3[15:0]	c_clk	994.5	994.5
k4[15:0]	c_clk	994.3	994.3
k5[15:0]	c_clk	994.8	994.8
k6[15:0]	c_clk	994.6	994.6
k7[15:0]	c_clk	994.4	994.4

Output port:

Port Name	Reference Clock	Arrival Time (ns)	Max Required Time (ns)	Max Slack(ns)
c0[15:0]	c_clk	2.5	1000	997.5
c0[15:0]	c_clk	2.5	1000	997.5
c0[15:0]	c_clk	2.5	1000	997.5
c0[15:0]	c_clk	2.5	1000	997.5
s_crypt	c_clk	2.5	1000	997.5
s_key	c_clk	2.5	1000	997.5

Start point for paths with Slack Worse than 942.7 ns

Instance	Type	Pin	Time(ns)	Max Slack (ns)
i_m1[15:0]	DFFRE	Q	4.7	941.9

End point for paths with Slack Worse than 942.7 ns

Instance	Type	Pin	Time(ns)	Max Slack (ns)
i_m2[15:0]	DFFRE	D	1001.4	941.9
i_d2[15:0]	DFFRE	D	1001.4	942.5
i_d3[15:0]	DFFRE	D	1001.4	942.5

Critical path with Slack Worse than 940.5 ns

Resource Usage Report

Mapping to part: 4062xlahq240-09
Cell usage:

FD	91 uses	FDC	57 uses	FDPE	1 use
FDCE	1889 uses	FDP	4 uses		

I/O primitives:

IBUF	195 uses	OBUF	1 use	OFDX	65 uses
IFDX	128 uses	BUFG	1 use		

Carry primitives used for arithmetic functions:

FORCE-0	20 uses	INC-FG-CI	70 uses	ADD-FG-CI	127 uses
SUB-FG-CI	98 uses	DEC-FG-0	1 use	SUB-F-CI	5 uses
EXAMINE-CI	3 uses	INC-F-CI	2 uses	FORCE-1	10 uses
INC-FG-1	4 uses	DEC-F-CI	4 uses		

I/O Register bits: 193
Register bits not including I/Os: 2042

Logic Mapping Summary:
FMAPs: 2762 of 4608 (60%)
HMAPs: 666 of 2304 (29%)
Total packed CLBs: 1381 of 2304 (60%)

ประวัติผู้เขียน

ชื่อ นายสาวิตร์ ตัณฑนุช

วัน เดือน ปี เกิด 22 มีนาคม 2517

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมศาสตร์บัณฑิต(วิศวกรรมไฟฟ้าสื่อสาร)	ม.สงขลานครินทร์	2539