ระบบรักษาความปลอดภัยของข้อมูลบนสมาร์ทการ์ดโดยใช้ ECC

DATA SECURITY SYSTEM ON SMART CARD USING ECC

โดย

นาย กงกฤช รัตโนดม

นาย ประพันธ์ ลี้กุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ระบบรักษาความปลอดภัยของข้อมูลบนสมาร์ทการ์ดโดยใช้ ECC

DATA SECURITY SYSTEM ON SMART CARD USING ECC

โดย

นาย คงกฤช รัตโนดม 45015002

นาย ประพัน ลี้กุล 45015014

อาจารย์ที่ปรึกษา

ดร. พรชัย ทรัพย์นิธิ

อ. ศรวัฒน์ ชิวปรีชา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ระบบรักษาความปลอดภัยของข้อมูลบนสมาร์ทการ์ดโดยใช้ ECC**

DATA SECURITY SYSTEM ON SMART CARD USING ECC

ผู้จัดทำ

1. นายคงกฤช รัตโนดม 45015002
2. นายประพัน ลี้กุล 45015014

..... อาจารย์ที่ปรึกษา

(ดร. พรชัย ทรัพย์นิธิ)

..... อาจารย์ที่ปรึกษา

(อาจารย์ ศรวัฒน์ ชิวปรีชา)

ระบบรักษาความปลอดภัยของข้อมูลบนสมาร์ตการ์ดโดยใช้ ECC

Data security system on Smart Card using ECC

โดย นายคงกฤช รัตโนดม 45015002
นายประพัน ลี้กุล 45015014

อาจารย์ที่ปรึกษา ดร.พรชัย ทรัพย์นิธิ
อาจารย์ ศรวัฒน์ ชีวปรีชา

บทคัดย่อ

โครงงานนี้นำเสนอ การเข้ารหัสและถอดรหัส เพื่อรักษาความปลอดภัยของข้อมูลบนสมาร์ตการ์ดแบบ ECC (Elliptic Curve Cryptosystem) ซึ่งทำให้สามารถลดจำนวน Public Key ที่ใช้ เมื่อเทียบกับการเข้ารหัสและถอดรหัสแบบอื่น จึงทำให้ประหยัดเนื้อที่ ในหน่อยความจำของสมาร์ตการ์ด แต่ยังคงมีความปลอดภัยสูง เมื่อเทียบกับความปลอดภัยที่ได้จากระบบอื่นๆ ในระดับ Public Key ที่เท่ากันเช่น RSA , EDL (Elgamal Discrete Logarithm) และได้จำลองการทำงานของการเข้ารหัสและถอดรหัส ที่พัฒนาโดยภาษา VHDL รวมทั้งทดสอบการทำงานจริงบนอุปกรณ์ FPGA

Abstract

This project implements encryption and decryption method known as "Elliptic Curve Cryptosystem"(ECC) for Data security system for the smart card application. The ECC requires smaller key size for the same security level compared with other encryption method. A hardware implementation is to be test on FPGA using the hardware language VHDL.

สารบัญ

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ที่มาของโครงงาน

ในปัจจุบัน สมาร์ทการ์ด มีอยู่มากมายหลายชนิด และมีการนำไปใช้งานอย่างแพร่หลายมากขึ้น อีกทั้ง ยังได้มีความพยายามผลักดันให้มีการนำสมาร์ทการ์ดเป็นมาตรฐานในเรื่องของการทำธุรกรรมด้านการเงินการธนาคาร ด้วยการสร้างข้อกำหนด EMV (Europay,Master,Visa) ในการใช้งานในด้านสมาร์ทการ์ดในด้านการทำธุรกรรม และการเงินการธนาคารร่วมกันทั้งหมด ซึ่งอีกไม่นานเราจะได้เห็นบัตรเครดิตในรูปของสมาร์ทการ์ดอย่างแน่นอน นอกจากนั้นก็ยังมีการใช้งานในภาคธุรกรรมอุตสาหกรรม ข้อมูลเฉพาะบุคคลเช่น บัตรประชาชน บัตรสุขภาพ และธุระกิจประกันภัย เป็นต้น ทั้งนี้ขึ้นอยู่กับความคิดในการประยุกต์ใช้งานตามความเหมาะสม แต่จุดเด่นที่ทำให้สมาร์ทการ์ดเหนือกว่าบัตรธรรมดา คือ สมาร์ทการ์ดสามารถประมวลผลได้ด้วยตนเอง และมีระบบรักษาความปลอดภัยของข้อมูล โดยเริ่มแรกขนาดความจุของหน่วยความจำบนสมาร์ทการ์ดเก็บข้อมูลได้น้อย ต่อมาด้วยเทคโนโลยี VLSI ทำให้สามารถสร้างหน่วยความจำที่มีขนาดความจุเพิ่มมากขึ้น แต่เนื่องจากขนาดของ Public – Key ที่ใช้งานมีขนาดใหญ่ จึงทำให้สิ้นเปลืองหน่วยความจำเพื่อใช้สำหรับการเก็บค่าของ Public – Key นี้ ดังนั้นจึงมีความต้องการที่จะลดขนาดของ Public – Key ที่ใช้งานกับสมาร์ทการ์ดให้มีขนาดเล็กลง และสามารถใช้งานได้อย่างปลอดภัยตามที่มาตรฐานกำหนด ดังนั้นจึงเกิดแนวคิดที่นำเอาระบบเข้ารหัสลับแบบ Elliptic Curve (ECC : Elliptic Curve Cryptosystem) มาประยุกต์ใช้งานกับสมาร์ทการ์ด เนื่องจากขนาดของกุญแจที่เล็กกว่ามาก แต่ให้ความปลอดภัยที่เท่ากันเมื่อเทียบกับรหัสลับระบบอื่น ๆ อีกทั้งในปัจจุบันที่เล็กกว่ามาก แต่ให้ความปลอดภัยที่เท่ากันเมื่อเทียบกับรหัสลับระบบอื่น ๆ อีกทั้งในปัจจุบันได้มีการทำวิจัยกันออกมามากมาย จนทำให้ ECC นั้นสามารถสร้างให้วงจรมีขนาดเล็กและทำงานได้รวดเร็วมาก ซึ่งมีความเหมาะสมเป็นอย่างยิ่งที่จะนำไปใช้งานบนสมาร์ทการ์ด

ปกติการรักษาความปลอดภัยของข้อมูลนั้นจะต้องเข้ารหัสข้อมูลก่อนที่จะเก็บลงหน่วยความจำในส่วนของสมาร์ทการ์ดนั้นได้เก็บ Public – Key ซึ่งใช้สำหรับการเข้ารหัสข้อมูลด้วย Public – Key ของผู้ที่ส่งต้องการจะส่งถึง และในส่วนของการถอดรหัส เมื่อมีเอกสารที่ถูกเข้ารหัสด้วย Public – Key ของผู้ถือบัตรส่งมา ผู้ถือบัตรจะสามารถถอดรหัสได้ด้วยกุญแจส่วนตัวที่เก็บอยู่ในบัตรนั่นเอง

โครงงานนี้ได้ทำการจำลองระบบการเข้ารหัสลับแบบ ECC สำหรับใช้งานบนสมาร์ทการ์ดซึ่งในคณิตศาสตร์แบบ Elliptic Curve Discrete Logarithm Problem บนคณิตศาสตร์สนามจำกัด (Finite Field) โดยในส่วนของการออกแบบตัวประมวลผลข้อมูลนั้นทำการออกแบบด้วยภาษา VHDL และคอมพิวเตอร์ช่วยในการจำลองการทำงานพร้อมทั้งทำการสังเคราะห์วงจรเป็นระดับเกตหลังจากนั้นจึงนำมาสร้างเป็นฮาร์ดแวร์ด้วยเทคโนโลยี VLSI เพื่อจำลองการทำงานของสมาร์ทการ์ดต่อไป

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษากระบวนการเข้ารหัสและถอดรหัสลับ ในส่วนที่เป็น Public – Key ที่นำมาจำลองการทำงานบนสมาร์ทการ์ด

1.2.2 เพื่อศึกษาวิธีการเข้ารหัสและถอดรหัสระบบกุญแจสาธารณะที่เหมาะสม

1.2.3 เพื่อศึกษาถึงวิธีการออกแบบวงจรรวมด้วยภาษา VHDL และวิธีทดสอบผลการทำงานของวงจรด้วยอุปกรณ์ FPGA

1.3 ขอบเขตของโครงงาน

โครงงานนี้ได้จำลองการทำงานของสมาร์ทการ์ด ให้มีฟังก์ชั่นเข้ารหัสและถอดรหัสอยู่ภายในโดยใช้อักกอริทึมของที่เป็นมาตรฐานของ ECC พร้อมทั้งได้ทำออกแบบหน่วยความจำที่ใช้จึงทำให้สามารถเข้ารหัสและถอดรหัสข้อมูลได้จากอุปกรณ์ FPGA ได้โดยตรง

1.4 ผลที่คาดว่าจะได้รับ

1.4.1 เพื่อพัฒนาทักษะในการออกแบบวงจรรวมด้วยภาษา VHDL และการใช้งานอุปกรณ์ FPGA

1.4.2 เพื่อพัฒนาความรู้ ความเข้าใจกระบวนการในการเข้ารหัสและถอดรหัส

1.4.3 เพื่อให้โครงงานที่สร้างขึ้นนี้เป็นต้นแบบที่ให้ผู้สนใจนำไปศึกษาและพัฒนาต่อไป

บทที่ 2
สมาร์ทการ์ด

2.1 ประวัติความเป็นมาของสมาร์ทการ์ด

การใช้งานสมาร์ทการ์ดนั้น เริ่มต้นเกิดจากการใช้งานบัตรที่อยู่ในรูปของบัตรพลาสติกซึ่ง เรียกกว่าพลาสติกการ์ด (Plastic Card) ซึ่งเริ่มใช้ในปี ค.ศ. 1950 โดยบริษัท Diner Club ซึ่งสมาชิกผู้ถือบัตรสามารถใช้บริการตามภัตตาคาร ร้านอาหาร หรือโรงแรมที่ซึ่งบริษัทนี้เป็นสมาชิกอยู่ ต่อมีปี ค.ศ. 1951 บริษัท American Express ก็ได้นำบัตรนี้มาให้บริการแก่สมาชิกของบริษัท ซึ่งการใช้งานของทั้งสองบริษัทนั้นเป็นลักษณะของการให้สิทธิพิเศษกับสมาชิก แต่ที่จริงแล้วการใช้บัตรนี้เป็นบัตรเครดิตนั้นได้เริ่มใช้งานตั้งแต่ปี ค.ศ. 1940 ในธนาคารของสหรัฐอเมริกา ซึ่งจะให้สิทธิบัตรพิเศษแก่ลูกค้าของธนาคาร โดยที่ลูกค้าของธนาคารสามารถติดต่อกับธนาคารสาขาต่าง ๆ ในสหรัฐอเมริกาได้สะดวกและรวดเร็ว จึงเรียกการทำธุรกิจโดยใช้บัตรแบบนี้ว่า "วีซ่า" ต่อมาในปี ค.ศ. 1966 ก็ได้มีการใช้บัตรนี้ในประเทศอังกฤษ โดยใช้งานร่วมกับบัตรวีซ่าในสหรัฐอเมริกา และปี ค.ศ. 1972 ธนาคารยักษ์ใหญ่ 4 ธนาคารคือ ธนาคาร Lloyds ธนาคาร Nation Westminster ธนาคาร Midland และธนาคาร Royal ก็ได้ร่วมมือกันเพื่อที่จะใช้บัตรนี้ติดต่อธุรกิจทางการเงิน โดยใช้ชื่อของ Master card ต่อมาเพื่อความสะดวกสบายให้กับลูกค้าในการรับบริการถอนเงินด่วนจึงมีการใช้บัตรนี้กับตู้บริการเงินด่วน คือ Automatic Teller Machine (ATM) ซึ่งถือเป็นที่แพร่หลายในเวลาต่อมา

การใช้งานสมาร์ทการ์ดนั้น เริ่มใช้งานในปี ค.ศ.1970 ที่ประเทศฝรั่งเศส โดยเกิดแนวคิดที่ต้องการใช้บัตรแทนจำนวนเงินซึ่งเรียกว่า Chip – Card หลังจากนั้นก็ได้มีการพัฒนาให้มีการทำงานร่วมกับวงจรอิเลคทรอนิกส์ ซึ่งเรียกว่า Integrated Circuit Card (ICC) และในทุกวันนั้น ใช้งานสมาร์ทการ์ดกันอย่างแพร่หลายในวงสังคม ผลจากการใช้บัตรในการทำธุรกรรมต่าง ๆ นั้น ทำให้ชีวิตสะดวกสบายมากขึ้น รวดเร็วขึ้น ซึ่งเราสามารถนำเอาสมาร์ทการ์ดไปใช้งานต่าง ๆ ได้ดังเช่น ธุรกิจทางการเงิน ข้อมูลทางสุขภาพ ข้อมูลเฉพาะบุคคล การรักษาความปลอดภัย ธุรกิจสื่อสาร บัตรเอทีเอ็มสำหรับเบิกเงินสด บัตรชมภาพยนต์ บัตรสำหรับซื้อตั๋วรถไฟฟ้า รถโดยสารประจำทาง อื่น ๆ อีกมากมาย ทุกวันนั้นบัตรสมาร์ทการ์ดได้ถูกพัฒนามากขึ้น แต่จุดหนึ่งที่มีความสำคัญและน่าสนใจมากที่สุดสำหรับการใช้งานบัตรสมาร์ทการ์ดคือ ทางด้านความปลอดภัยของการใช้านทั้งระบบ ซึ่งถึงแม้ว่าจะมีการพัฒนามากแล้วก็ตามแต่ก็ยังป้องกันได้ไม่เต็ม 100 เปอร์เซ็นต์ แต่ที่แน่นอนคือว่าบัตรพลาสติกแถบแม่เหล็กที่ใช้กันแพร่หลายในปัจจุบันอย่างแน่นอน ซึ่งอาจกล่าวได้ว่าสมาร์ทการ์ดนั้นกำลังจะเข้ามาที่ส่วนสำคัญเกี่ยวกับการดำเนินชีวิตประจำวันนั่นเอง

2.2 องค์ประกอบภายในสมาร์ทการ์ด

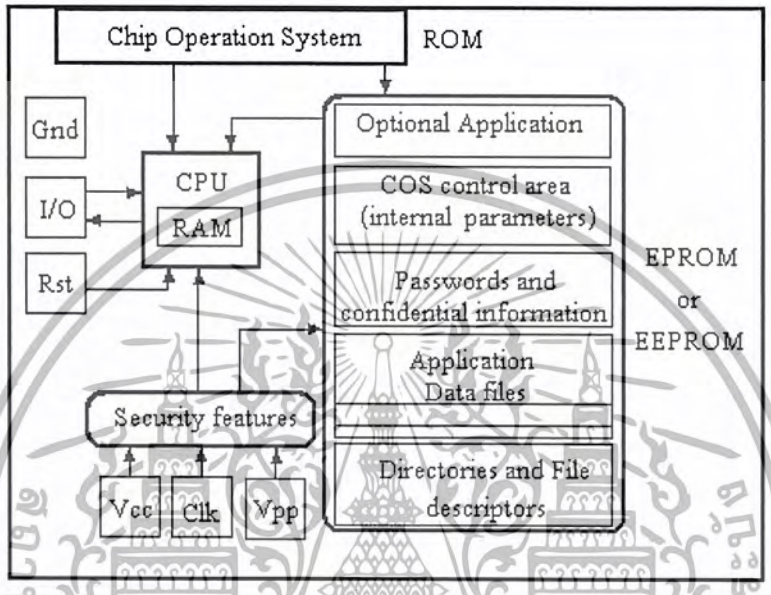
สมาร์ทการ์ด คือ อุปกรณ์เก็บข้อมูลแบบพกพาซึ่งสามารถแสดงข้อกำหนดและคุณสมบัติเฉพาะตัว พร้อมทั้งระดับความปลอดภัยได้ ซึ่งสมาร์ทการ์ดเกิดขึ้นจากการใช้งานเทคโนโลยีอิเล็กทรอนิกส์ ทำการออกแบบตัวประมวลผลสำหรับการใช้งานดังกล่าว และเนื่องจากการพัฒนา

เทคโนโลยีทางด้านอิเลคทรอนิกส์ ทำให้สมาร์ทการ์ด มีความจุของข้อมูลเพิ่มมากขึ้น ความเร็วในการประมวลผลสูงขึ้นนั่นเอง

2.2.1 โครงสร้างพื้นฐานของสมาร์ทการ์ด

สมาร์ทการ์ด สามารถแบ่งโครงสร้างออกเป็น 2 ส่วนใหญ่ ๆ ด้วยกันคือ

- ไมโครโปรเซสเซอร์
- หน่วยความจำ



รูปที่ 2.1 แสดงโครงสร้างพื้นฐานของสมาร์ทการ์ด

2.2.1.1 ไมโครโปรเซสเซอร์

สถาปัตยกรรมในส่วนนี้นั้นประกอบด้วย CPU RAM ROM และ EEPROM หน่วยความจำ ROM ทำหน้าที่เก็บชุดคำสั่งซึ่งเป็นขั้นตอนการทำงานของระบบ CPU ทำหน้าที่ประมวลผลข้อมูลโดยที่หน่วยความจำ RAM ทำหน้าที่เป็นรีจีสเตอร์ช่วยสำหรับการประมวลผลข้อมูล และข้อมูลที่ได้ประมวลผลเสร็จเรียบร้อยและได้ถูกเก็บในหน่วยความจำ EEPROM โดยทั่วไปนั้นขนาดของหน่วยความจำ EEPROM มีขนาดเป็น 4 เท่าของหน่วยความจำ

2.2.1.2 หน่วยความจำ

ส่วนของหน่วยความจำนั้นส่วนที่เก็บข้อมูลของสมาร์ทการ์ด โดยสามารถออกแบบได้ 3 ส่วน ดังนี้

- พื้นที่เปิด (Open Zone) คือ พื้นที่ที่ใช้สำหรับเก็บข้อมูลของบริษัทผู้ผลิต Serial Number ข้อมูลของผู้ถือบัตรที่สามารถเปิดเผยได้
- พื้นที่ปกปิด (Secret Zone) คือพื้นทีที่ใช้สำหรับเก็บข้อมูลปกปิดส่วนบุคคล
- พื้นที่ใช้งาน (User Zone) คือพื้นที่เก็บข้อมูลที่ได้ทำการเข้ารหัส

2.3 การ์ดหน่วยความจำและการ์ดไมโครโปรเซสเซอร์ชนิดต่าง ๆ

สมาร์ทการ์ดปัจจุบันมีอยู่ 2 ชนิด คือ การ์ดหน่วยความจำ (Memory Card) และการ์ดไมโครโปรเซสเซอร์ (Microprocessor Card) ดังรูปที่ 2.1 การ์ดหน่วยความจำนั้นทำหน้าที่ง่าย ๆ เพียงเก็บข้อมูล (Store Data) และสามารถอ่านออกมาได้คล้าย ๆ กับแผ่นฟลอปปี้ดิสก์ขนาดเล็กอันหนึ่ง แต่มีความปลอดภัยสูงกว่า ส่วนการ์ดไมโครโปรเซสเซอร์นั้นจะมีจุดเด่นกว่าคือ สามารถเพิ่ม ลบ หรือจัดการกับหน่วยความจำภายในการ์ดด้วย ดังนั้นการ์ดนี้จึงมีคุณสมบัติคล้ายกับคอมพิวเตอร์ขนาดเล็ก ๆ ทีเดียว ซึ่งภายในการ์ดจะมีระบบปฏิบัติการ (input / out port operation system) และหน่วยความจำ (Memory) อยู่ภายใน รวมถึงการรักษาความปลอดภัยในการใช้งานก็ถูกติดตั้งไว้ด้วย

2.4 สมาร์ทการ์ดแบบมีคอนแทกต์และไม่มีคอนแทกต์

2.4.1 สมาร์ทการ์ดแบบมีคอนแทกต์

สมาร์ทการ์ดสามารถแบ่งออกเป็น 2 ประเภทใหญ่ ๆ คือ ชนิดมีคอนแทกต์ (Contact Smart Card) และชนิดไม่มีคอนแทกต์ (Contactless Smart Card) สำหรับสมาร์ทการ์ดแบบมีคอนแทกต์นั้นต้องใช้งานโดยการสอดบัตรเข้ากับเครื่องอ่านบัตร (Smart card Reader) บัตรชนิดนี้จะมีเพลตสีทองขนาดเล็ก (Small Gold Plate) ที่ด้านหน้าของบัตรซึ่งแตกต่างจากบัตรเครดิตแบบใช้แถบแม่เหล็กที่พบเห็นได้ทั่วไปซึ่งมีแถบแม่เหล็กอยู่ด้านหลังบัตร เมื่อการ์ดถูกสอดเข้ากับเครื่องอ่านบัตรทำให้เกิดการสัมผัสที่คอนแทกต์ของบัตร ทำให้เกิดการเชื่อมต่อทางไฟฟ้า เพื่อใช้ในการรับส่งข้อมูลเข้าและออกจากชิบบนบัตรได้ ลักษณะของบัตรสมาร์ทการ์ดแบบมีคอนแทกต์ ดังแสดงรูปที่ 2.2

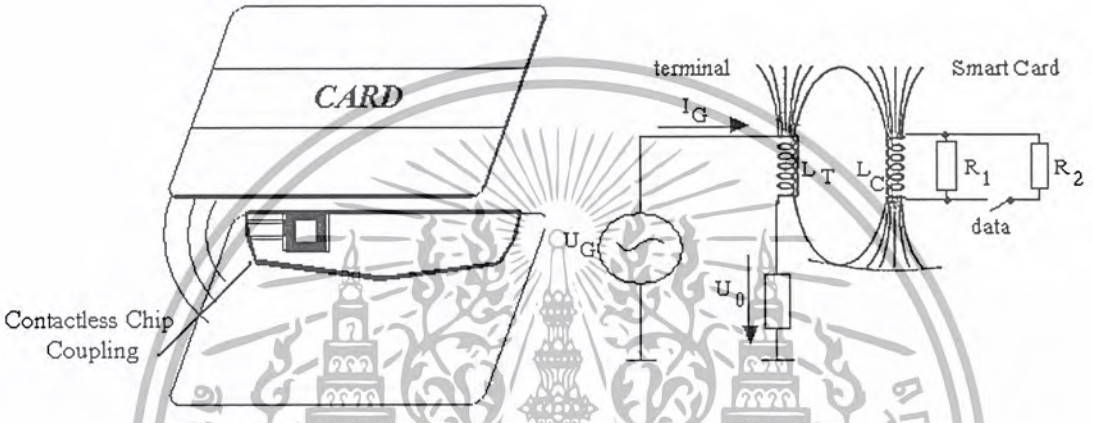
รูปที่ 2.2 สมาร์ทการ์ดแบบคอนแทกต์

2.4.2 สมาร์ทการ์ดแบบไม่มีคอนแทกต์

บัตรสมาร์ทการ์ดอีกชนิดหนึ่งคือชนิดไม่มีคอนแทกต์ (Contactlass) มีความพิเศษกว่าตรงที่มีการรับส่งข้อมูลระหว่างชิปภายในบัตรกับภายนอก โดยใช้การส่งผ่านข้อมูลทางอากาศโดยการเหนี่ยวนำ (Coupling Loop) ซึ่งถูกติดตั้งอยู่ภายในบัตร ดังนั้นถ้าดูจากภายนอกตัวบัตรจะมีลักษณะเหมือนกับบัตรเครดิตพลาสติกทั่วไป แตกต่างกันเพียงแต่ว่าสมาร์ทการ์ดแบบนี้ไม่มีไมโครชิปอิเล็กทรอนิกส์และส่วนของการสื่อสารข้อมูลโดยการเหนี่ยวนำติดตั้งอยู่ภายใน ซึ่งอุปกรณ์เหล่านี้ทำให้การ์ดสามารถ

ติดต่อสื่อสารข้อมูลโดยปราศจากการสัมผัส บัตรสมาร์ทการ์ดแบบนี้จึงเหมาะอย่างยิ่งสำหรับงานที่ต้องการความเร็วในการใช้งาน เช่น ใช้เป็นตัวสำหรับระบบรถไฟฟ้าหรือการฝ่าเข้าทางด่วนพิเศษที่มีผู้ใช้มาก ๆ เป็นต้น

นอกจากบัตรสมาร์ทการ์ดทั้งสองแบบดังกล่าวแล้ว ปัจจุบันยังมีการผลิตสมาร์ทการ์ดแบบผสมหรือที่เรียกว่า คอมบิการ์ด (Combi Card) ออกมาใช้งานอีกด้วย โดยบัตรแบบนี้เป็นบัตรใบเดียวแต่ทำหน้าที่เป็นทั้งสมาร์ทการ์ดแบบมีการสัมผัส และสมาร์ทการ์ดแบบไม่มีการสัมผัสเพื่อเพิ่มความสะดวกและประโยชน์ในการใช้งานมากขึ้น



รูปที่ 2.3 สมาร์ทการ์ดแบบไม่มีคอนแทกต์

2.5 มาตรฐานของสมาร์ทการ์ด

ในปี ค.ศ. 1981 ได้มีการกำหนดมาตรฐานการส่งผ่านข้อมูลของสมาร์ทการ์ด โดยใช้มาตรฐานของ AFNOR (Association Francais de Normalisation) และได้นำมาตรฐานนี้มากำหนดเป็นมาตรฐานของ ISO ซึ่งปัจจุบันมีมาตรฐานพื้นฐานที่เกี่ยวข้องกับสมาร์ทการ์ดชนิดของคอนแทกต์ถูกกำหนดตามมาตรฐาน ISO 7816 ซีรีส์ซึ่งมีอยู่ทั้งหมด 10 หมวด ในขณะที่บัตรชนิดที่ไม่มีคอนแทคถูกกำหนดตามมาตรฐาน ISO 1443 มาตรฐานเหล่านี้จะเป็นตัวกำหนดคุณสมบัติต่าง ๆ ของสมาร์ทการ์ด ที่มีการผลิตออกมาใช้งานครอบคลุมตั้งแต่ลักษณะภายนอกโดยทั่วไป กระบวนการทางไฟฟ้า ทางกลไกและการอินเตอร์เฟส ต่อไปนี้เป็นรายละเอียดมาตรฐาน ISO 7816 ซีรีส์ทั้ง 10 หมวด

- IS 7816 – 1 (1987) : Physical Characteristics Amendment 1
  (1998) : Revised edition March 1998
- IS 7816 – 2 (1998) : Dimension and location of contacts Revised edition March 1998
- IS 7816 – 3 (1989) : Electronic Signal and Transmission Protocol Amendment
  (1992) : Protocol T = Amendment 2

(1994) : Revision of Protocol Type Selection Amendment 3

(1998) : Introduction of Voltage ICCS

- IS 7816 – 4 (1995) : Inter industry commands and responses Amendment 1

(1998) : Revision Secure Messaging

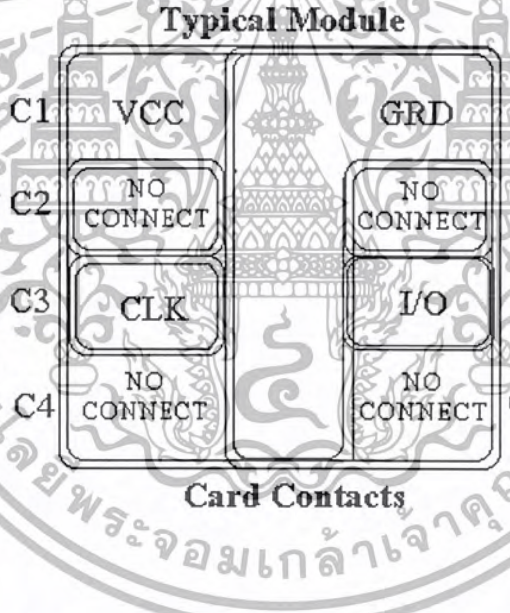
- IS 7816 – 5 (1994) : Registration system for application identifiers Amendment 1

(1996) : Registration of identifiers

- IS 7816 – 6 (1995) : Data element for interchange Amendment 1

(DIS) : Registration of IC Manufactures

- IS 7816 – 7 (1998) : Smart Card Query Language commands
- DIS 7816 – 8 : Inter – industry Security Commands
- CD 7816 – 9 : Inter – industry Enhanced Commands
- ISO 7816 – 10 (1999) : Synchronous cards



รูปที่ 2.4 แสดงมาตรฐานขาสัญญาณของบัตรสมาร์ทการ์ด

2.6 มาตรฐานสำหรับการรักษาความปลอดภัยในบัตรสมาร์ทการ์ด

ความเป็นจริงแล้วในปัจจุบันนี้ มีการนำรูปแบบและเทคนิคในการรักษาความปลอดภัยบนสมาร์ทการ์ดมาใช้งานหลายรูปแบบซึ่งในที่นี้คงจะไม่ได้กล่าวถึงเทคนิคต่าง ๆ ที่นำมาใช้งาน แต่กล่าวเฉพาะรูปแบบและลักษณะการควบคุมได้ 2 แบบ คือ แบบแรกเป็นการควบคุมว่าผู้ใดจะสามารถเข้าถึงข้อมูลต่าง ๆ ในบัตรได้อย่างไรบ้าง และแบบที่ 2 คือควบคุมระดับของการเข้าถึงข้อมูลในบัตร

ในแบบแรกนั้นหมายถึงเป็นการใช้ระบบความปลอดภัยควบคุมว่าต้องการให้ใครบ้างสามารถเข้าถึงข้อมูลได้ ซึ่งแบ่งได้เป็น 3 ระดับ

1. เข้าถึงได้ทุกคน คือ บัตรสมาร์ตการ์ดประเภทนี้จะไม่มีการใช้รหัสผ่าน (Password) ผู้ใดที่ถือบัตรนี้สามารถเข้าถึงข้อมูลได้ทันที แต่อาจเป็นข้อมูลบางส่วนเท่านั้นก็ได้ เช่น ข้อมูลเกี่ยวกับชื่อ มีเลือดกรุ๊ปใดเป็นต้น ซึ่งเสมือนบัตร Medicard ที่สามารถอ่านข้อมูลได้ โดยไม่ต้องใช้รหัสผ่าน
2. เข้าถึงได้เฉพาะผู้ถือบัตร คือ บัตรแบบนี้ส่วนใหญ่มีการใช้รหัสผ่านสำหรับผู้ถือบัตรร่วมด้วย ซึ่งมักเรียกกันว่า PIN (Personal Identification Number) ที่อาจมีความยาว 4 – 5 หลัก โดยป้อนผ่านทางคีย์แพด ระบบนี้ถ้าหากมีผู้ที่พยายามเข้าถึงข้อมูลภายในบัตรนี้ โดยพยายามป้อนรหัสผ่านซึ่งถ้าป้อนรหัสผิดเกิน 3 ครั้ง บัตรจะทำการล็อคตัวเอง และถ้าทำการปลอดล็อคนี้ต้องใช้รหัสผ่านตัวอื่น ซึ่งมีความซับซ้อนกว่ามาป้อนแทน จึงกลับมาสู่สภาวะปกติได้
3. เข้าถึงได้เฉพาะบุคคลที่ 3 เท่านั้น คือ สมาร์ทการ์ดบางชนิดสามารถเข้าถึงข้อมูลได้เฉพาะบุคคลที่ 3 เท่านั้น ซึ่งโดยส่วนใหญ่ก็คือ ผู้ออกบัตรให้เท่านั้น เช่นบัตรสมาร์ทการ์ดที่ใช้แทนเงินอิเล็กทรอนิกส์ สามารถโหลดข้อมูลใหม่ได้เฉพาะธนาคารที่ออกบัตรให้เท่านั้นเป็นต้น

การรักษาความปลอดภัยสนแบบที่ 2 คือ การจำกัดการเข้าถึงข้อมูลภายในบัตร โดยสามารถแบ่งการเข้าถึงข้อมูลภายในบัตรออกเป็นส่วน ๆ ซึ่งส่วนใหญ่แบ่งออกเป็น 4 ระดับ

1. ข้อมูลที่สามารถอ่านได้เท่านั้น (Read Only) คือ สามารถอ่านข้อมูลในบัตรได้เท่านั้น
2. ข้อมูลที่สามารถเพิ่มได้เท่านั้น (Added Only) คือ สามารถเขียนข้อมูลลงไปในบัตรได้เท่านั้น
3. ข้อมูลที่สามารถอัพเดทได้เท่านั้น (Updated Only) คือ เก็บข้อมูลใหม่ได้เท่านั้นโดยที่ข้อมูลเก่าถูกลบทิ้ง
4. ข้อมูลที่ไม่อนุญาตเข้าถึงได้เลย คือ ไม่สามารถอ่านหรือเขียนข้อมูลภายในบัตรได้เลย

บัตรโดยทั่วไป หรือบัตรสมาร์ทการ์ดบางประเภทควบคุมการใช้ข้อมูลที่อยู่ในบัตรโดยการใช้รหัสผ่าน ซึ่งมีแต่เพียงผู้ถือบัตรเท่านั้นที่ทราบ แต่วิธีนี้หากข้อมูลตรงส่วนนี้จำเป็นต้องถูกส่งผ่านทางสายโทรศัพท์หรือทางคลื่นวิทยุแล้ว คงจะนับได้ว่าการรักษาความปลอดภัยเพียงเท่านี้คงไม่เพียงพอ วิธีหนึ่งที่สามารถนำมาใช้เพื่อเพิ่มความปลอดภัยขึ้น คือ การใช้กระบวนการไซเฟอร์ริ่ง (ciphering) ซึ่งจะมีการทำงานคล้ายกับการแปลภาษาจากข้อมูลหนึ่งไปเป็นภาษาอื่นที่ไม่รู้จัก สมาร์ทการ์ดบางระบบจะถูกบรรจุระบบไซเฟอร์ริ่งและดีไซเฟอร์ริ่ง ซึ่งก็คือการเข้ารหัสและการถอดรหัสเพื่อทำการแปลข้อมูลให้กลับมาเหมือนเดิม ดังนั้นการส่งผ่านข้อมูลจึงทำได้อย่างสมบูรณ์ไม่มีอะไรผิดเพี้ยน

สมาร์ทการ์ดสามารถใช้ระบบไซเฟอร์ริ่งนี้ ทำการแปลข้อมูลที่แตกต่างกันได้มากนับหลายพันล้านรูปแบบและจะทำการสุ่มรูปแบบกันใหม่ทุกครั้งที่มีการสื่อสาร ด้วยวิธีการนี้จึงทำให้เรามั่นใจได้ว่าการใช้งานบัตรสมาร์ทการ์ดในการประยุกต์ในงานทุกรูปแบบและในบางประเภทที่ต้องการความปลอดภัยสูง เช่น การใช้แทนเงินสด

2.7 อนาคตของสมาร์ทการ์ด

คุณสมบัติที่เด่นที่สำคัญของสมาร์ทการ์ดก็คือ เป็นอุปกรณ์อิเล็กทรอนิกส์ที่สามารถพกพาได้ในกระเป๋าสตางค์ของทุกคน ซึ่งมีหน้าที่หลักคือจัดเก็บและจัดการกับข้อมูลต่าง ๆ ในรูปแบบอิเล็กทรอนิคส์ได้ ความฉลาดของสมาร์ทการ์ดคือ วงจรอิเล็กทรอนิกส์ที่ถูกติดตั้งอยู่ภายในการ์ดพลาสติกซึ่งสามารถที่จะปกปิดข้อมูลพร้อมกับการจัดการข้อมูลภายในได้ และในอนาคตด้วย เทคโนโลยีคล้ายกันนี้อาจนำพาไปสู่กาติดตั้งในรูปแบบอื่น ๆ เช่น ในกุญแจ นาฬิกา แว่นตา แหวน และอุปกรณ์อื่น ๆ ที่เป็นสิ่งใกล้ตัวที่เราต้องใช้อยู่ทุก ๆ วัน และในทุกวันนี้ก็มีการใช้เทคโนโลยี สมาร์ทการ์ดนี้เป็นสมาร์ทคีย์ (Smart Key) ในการใช้งานกับระบบเคเบิลทีวีที่มีการบอกรับสมาชิกกันแล้ว การพัฒนาในอีกด้านหนึ่งที่น่าสนใจกว่า คือ การพัฒนาทางเทคโนโลยีสมาร์ทการ์ดแบบไม่มีคอนแทคต์ ซึ่งเหมาะมากสำหรับการนำไปสร้างเป็นอุปกรณ์ที่เราเรียกกว่าแท็ก (Tag) อุปกรณีประเภทแท็กนั้นมีการทำงานคล้ายกับสมาร์ทการ์ดแบบไม่มีคอนแทคต์มากเพียงแต่อาจจะไม่ได้อยู่ในรูปแบบของบัตรพลาสติกเท่านั้น แต่อาจเป็นรูปแบบของแหวนคล้องหรือป้ายติดสินค้า ซึ่งปัจจุบันมีการนำมาใช้งานกันบ้างแล้วเช่น การนำไปติดเข้ากับถังก๊าซ รถยนต์ สัตว์เป็นต้น ส่วนใหญ่ทำหน้าที่ในการจัดเก็บข้อมูลที่เกี่ยวข้องกับสิ่งนั้น ๆ ไว้ และอาจเป็นไปได้ที่ผู้ควบคุมอาจทำการเปลี่ยนแปลงข้อมูลภายในโดยไม่ต้องนำมาแก้ไขทีละชิ้นก็ได้ นอกจากนั้นสมาร์ทการ์ดยังอาจนำไปใช้งานร่วมกับระบบรักษาความปลอดภัยชั้นสูง ซึ่งในปัจจุบันเราเรียกกันว่าระบบไบโอเมตริก (Biometrics) โดยการใช้นิ้วมือ ฝ่ามือ เรตินาของตา หรือใช้เสียงในการระบุและแยกแยะแต่ละบุคคล ในอนาคตอาจมีการใช้ข้อมูลทางอิเล็กทรอนิกส์ที่บรรจุอยู่ในสมาร์ทการ์ดมาร่วมประมวลผลข้อมูลอีกด้วย สมาร์ทการ์ดจึงถือได้ว่าเป็นเทคโนโลยีใหม่ที่น่าติดตามและมีความสัมพันธ์ต่อการดำรงชีวิตของผู้คนนับหลายล้านคน ซึ่งในขณะนี้ก็เริ่มเป็นที่ประจักษ์แล้วเมื่อมีการใช้งานสมาร์ทการ์ดในการซื้อสินค้าต่าง ๆ ในร้านค้าหรือเมื่อใช้บัตรนี้ไปพบแพทย์

2.8 แนวคิดในการพัฒนาสมาร์ทการ์ด

ในปัจจุบันมีการใช้งานสมาร์ทการ์ดกันอย่างแพร่หลาย อีกรั้งความต้องการใช้งานสมาร์ทการ์ดเพื่อที่จะเก็บข้อมูลเฉพาะบุคคลมีเพิ่มมากขึ้นเช่น ข้อมูลบัตรผู้ป่วยในโรงพยาบาล จึงส่งผลให้ความต้องการเพิ่มความจุในการเก็บข้อมูลบนสมาร์ทการ์ดเพิ่มมากขึ้น ดังนั้นจึงเกิดแนวคิดที่จะเพิ่มขนาดของหน่วยความจำบนบัตรนั่นเอง เนื่องจากขนาดของกุญแจรหัสที่ใช้งานในปัจจุบันมีขนาดใหญ่มาก จึงทำให้เกิดการสิ้นเปลืองเนื้อที่บนหน่วยความจำบนสมาร์ทการ์ดซึ่งใช้สำหรับเก็บ Public – Key และยังทำให้วงจรออกแบบมีขนาดใหญ่ ดังนั้นจึงเกดแนวคิดที่จะทำการลดขนาดของ Public – Key โดยที่ระดับความปลอดภัยยังคงใช้งานได้ดี พร้อมกับวิธีการเข้ารหัสข้อมูลเพื่อทำให้ความจุของข้อมูลบนบัตรสมาร์ทการ์ดเพิ่มมากขึ้น

บทที่ 3
ภาษาวีเอชดีแอล

3.1 แนะนำวีเอชดีแอล (Introduction to VHDL)

ในช่วงฤดูร้อนของปี 1981 สถาบันเพื่อป้องกัน (The Institute for Defence Analysis) ในสหรัฐอเมริกาได้จัดตั้งคณะทำงานขึ้นคณะหนึ่ง เพื่อทำการพัฒนาภาษาที่ใช้ในการบรรยายหรืออธิบายรูปแบบการทำงานและความสัมพันธ์ ของอุปกรณ์ฮาร์ดแวร์แบบใหม่ขึ้น ผลการทำงานของคณะทำงานชุดนี้ได้ก่อให้เกิดภาษาการบรรยายฮาร์ดแวร์ขึ้น เรียกกว่า VHDL (VHDL Hardware Description Language) โดย VHDL เป็นชื่อย่อของแผนกหนึ่งของสถาบันที่ทำงานเกี่ยวกับวงจรรวมที่มีความเร็วสูงมาก (Very High Speed Intergrated Circuit) ต่อมาในปี 1985 IEEE ได้ทำการผลักดันให้ VHDL กลายเป็นภาษาที่เป็นมาตรฐานและมีการยอมรับกันอย่างกว้างขวางในวงการอุตสาหกรรมคอมพิวเตอร์ด้วยความสามารถของ VHDL ในด้านของการกำหนดพฤติกรรมของการทำงานของวงจร ทำให้นักออกแบบสามารถกำหนดรูปแบบพฤติกรรมการทำงานได้ทั้งวงจรของดิจิตอลทั่ว ๆ ไป และในระบบที่แตกต่างกันออกไป เช่น พฤติกรรมการทำงานของระบบเรดาร์หรือพฤติกรรมการทำงานของระบบเครือข่ายในประสาทในสมองมนุษย์ได้ ข้อดีมีหลักที่สำคัญของ VHDL ก็คือภาษานี้จะสามารถถูกใช้ได้ตลอดในทุก ๆ ระดับขั้นตอนการออกแบบที่แตกต่างกันได้ นั่นคือ ในกระบวนการออกแบบระดับสูง (System Level) จนถึงระดับที่ต่ำกว่า (Lower hardware level) สามารถใช้ภาษาเดียวกันได้โดยตลอด ทำให้เพิ่มประสิทธิภาพในการติดต่อระหว่างกลุ่มที่ทำงานร่วมกันได้เป็นอย่างดี

3.1.1 ข้อกำหนด (VHDL Requirement)

ในเอกสารของ DoD (Department of Defense Requirement for Hardware Description Language) ซึ่งออกมาในเดือนมกราคม ปี 1983 ได้ตั้งขอกำหนดสำหรับภาษา VHDL

3.1.1.1 ลักษณะทั่วไป (Generation Features)

เอกสารของ DoD กำหนดไว้ว่า VHDL เป็นภาษาสำหรับการออกแบบและบรรยายของฮาร์ดแวร์ ซึ่งหมายถึงความสามารถในการอธิบายและออกแบบในระดับสูง โดยมีความสามารถในการเลียนแบบ (Simulation) การสังเคราะห์ (Synthesis) และการทดสอบ (Testing) นอกจากนั้น VHDL ยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์คือ ระบบจนถึงระดับเกทอีกด้วย เนื่องจากในการทำงานของระบบดิจิตอลจริงๆ ทุก ๆ องค์ประกอบในระบบไม่ว่าเล็กหรือใหญ่จะทำงานไปพร้อม ๆ กัน ซึ่งในเรื่องของความพร้อมในการทำงานนี้ถือว่าเป็นข้อกำหนดที่สำคัญอย่างหนึ่ง ใน VHDL ด้วยเช่นกัน (สำหรับในภาษาที่ใช้ในการบรรยายฮาร์ดแวร์แล้ว ความพร้อมเพรียงที่จะหมายถึงทุก ๆ คำสั่งองค์ประกอบเกท หรือวงจรต่าง ๆ จะถูกนำมาปฏิบัติทั้งหมด ดังนั้นในตอนท้ายแล้วก็จะดูเหมือนว่าได้ปฏิบัติไปพร้อม ๆ กัน)

3.1.1.2 สนับสนุนการออกแบบลำดับขั้น (Support for Design Hierachy)

การออกแบบลำดับขั้น เป็นลักษณะที่สำคัญอย่างหนึ่ง การออกแบบที่มีหลายๆ ระดับในการออกแบบจะประกอบด้วยส่วนการบรรยายเชื่อมต่อและส่วนการบรรยายหน้าที่การทำงาน หน้าที่การทำงานของระบบก็สามารถกำหนดได้ด้วยตนเองหรือถูกกำหนดโดยโครงนร้างที่ประกอบย่อย ๆ ลงไปได้เช่นกัน แต่ที่ระดับล่างสุดองค์ประกอบต้องถูกบรรยายหน้าที่การทำงานด้วยตัวมันเองและไม่สามารถกำหนดการทำงานโดยลักษณะแบบโครงสร้างได้

3.1.1.3 ไลบารี (Library Support)

VHDL ได้สนับสนุนการมีไลบรารีเพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษระและการทำงานของอุปกรณ์พื้นฐานไว้ในระบบไลบรารี หรือจะใช้ไลบรารีที่ระบบได้จัดเตรียมไว้แล้วก็ได้ โมเดลและการบรรยายที่ถูกต้องจะถูกเก็บไว้ในไลบรารี หลังจากที่ได้ผ่าการคอมไพล์เรียบร้อยแล้วเพื่อให้ผู้ออกแบบคนอื่น ๆ สามารถนำไปใช้ได้ด้วย

3.1.1.4 ลำดับคำสั่ง (Sequential Statement)

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการโดยพร้อมเพรียงกันจะเป็นคุณสมบัติที่สำคัญของ VHDL ก็ตาม ตัวภาษาเองได้มีการจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วย เมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ประกอบที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้ว ผู้ออกแบบก็ยังสามารถบรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรมที่ประกอบด้วยโครงสร้างแบบ case if-then-else และloop ทั่ว ๆ ไปได้ การบรรยายแบบลำดับคำสั่งทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้สะดวกและง่ายขึ้น อย่างไรก็ตามโครงสร้างทั้งหมดของ VHDL ก็คงเป็นการทำงานพร้อมเพรียงกันเช่นเดิม

3.1.1.5 การกำหนดคุณสมบัติ (Generic Design)

นอกจากการกำหนดอินพุทและอินเอาท์พุทแล้ว เงื่อนไขอื่น ๆ ก็มีผลต่อการปฏิบัติหน้าที่ของอุปกรณ์ฮาร์ดแวร์ด้วยเช่นกัน สิ่งนี้รวมถึงสภาพแวดล้อมและลักษณะทางกายภาพของอุปกรณ์นั้น ๆ ภาษาสำหรับการออกแบบที่ดีควรจะสามารถให้ผู้ออกแบบกำหนดคุณสมบัติของอุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนดขนาด ลักษณะทางกายภาพ เวลา โหลด และเงื่อนไขทางสภาพแวดล้อมอื่น ๆ ความสามารถในการกำหนดคุณสมบัติก็เป็นส่วนหนึ่งที่มีอยู่ในภาษา VHDL ด้วยเช่นกัน

3.1.1.6 ชนิดของข้อมูล (Type Declaration and usage)

VHDL สามารถกำหนดชนิดของข้อมูลไม่เพียงแต่ชนิด BIT และ BOOLEAN เท่านั้น แต่ยังสามารถกำหนดชนิดของข้อมูลเป็นจำนวนเต็ม จำนวนจริง จุดทศนิยมและชนิดลำดับ การนับ (Enumerate Type) หรือแม้แต่ชนิดของข้อมูลที่ผู้ออกแบบกำหนดขึ้นใส่เองก็ได้

3.1.1.7 โปรแกรมย่อย (Use of subprogram)

ความสามารถในการใช้ฟังก์ชันและโพรซีเจอร์ (Procedure) เป็นข้อกำหนดอีกอย่างหนึ่งใน VHDL เราสามารถที่จะใช้โปรแกรมในการเปลี่ยนแปลงชนิดของข้อมูล การกำหนดหน่วยของลอจิก (Logic) การกำหนดตัวกระทำต่าง ๆ ทั้งเท่าและใหม่หรืออะไรก็ตามได้เช่นเดียวกับการเขียนโปรแกรมทั่วไป

3.1.1.8 การควบคุมเวลา (Timing Control)

VHDL อนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ตามต้องการ การตรวจสอบ การออกแบบเกท หรือการหน่วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอยเหตุการณ์ (Event) นอกจากนี้ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

3.1.1.9 การกำหนดแบบโครงสร้าง (Structural Specification)

การกำหนดโครงสร้างขององค์ประกอบสามารถกระทำได้ในทุก ๆ ระดับของการออกแบบการกำหนดโครงสร้างขององค์ประกอบรวมที่เกิดจากองค์ประกอบย่อยที่ต่างกันหรือเหมือนกันก็เป็นเรื่องข้อกำหนดมาตรฐานอย่างหนึ่งเช่นกัน

3.2 ความสามารถของภาษา VHDL (Capability)

- ตัวภาษา VHDL สามารถใช้เป็นสื่อกลางในการแลกเปลี่ยนระหว่างผู้ผลิตชิพกับ ผู้ออกแบบ (CAD Tools)
- ใช้เป็นสื่อกลางในการแลกเปลี่ยนสื่อสารระหว่างซีเออี (CAE) และซีเอดีทูล (CAD Tool) เช่น ตัวภาษาซ้อสโค้ด (SOURCE CODE) ของ VHDL สามารถคอมไพล์เลอร์ (Compiler) และซีมูเลเตอร์ (Simurator) ได้หลายตัวแตกต่างกัน
- ภาษา VHDL สนับสนุนการออกแบบ แบบบนล่าง (Top Down Design) และแบบล่างขึ้นบน (Bottom Up Design) หรือผสมกันทั้งสองแบบ
- ตัวภาษา VHDL เป็นแบบทั่วไป (Generic) ไม่อิงเทคโนโลยีอันใดอันหนึ่ง ในขณะเดียวกันก็สนับสนุนหลายเทคโนโลยี
- ตัวภาษา VHDL สามารถอ่านและทำความเข้าใจโดยมนุษย์
- สนับสนุนการออกแบบทั้งระบบเชิงโครนัส (Synchronous) และอะซิงโครนัส (Asynchronous)
- ตัวภาษา VHDL เป็นภาษามาตรฐานรับรองโดย IEEA และ ANSI ทำให้โมเดลที่ออกแบบโดยภาษา VHDL สามารถเคลื่อนย้ายไปยังระบบใด ๆ ก็ได้ และสามารถนำกลับมาใช้ได้ใหม่
- สามารถเขียนโมเดลได้ขนาดไม่จำกัด ไม่มีข้อจำกัดในตัวภาษาเรื่องขนาดของโมเดล (ขึ้นอยู่กับซอฟแวร์)

- ภาษา VHDL สนับสนุนการเขียนถึง 3 รูปแบบ ได้แก่ แบบบีเฮฟวีเออร์ (Behavioral Style) แบบสตรัคเจอรัล (Structural Style) แบบดาต้าโฟลว์ (Data Flow) หรือสามารถเขียนรวมกันได้ทั้ง 3 รูปแบบ

- สนับสนุนการออกแบบวงจรขนาดใหญ่โดยใช้ความสามารถของส่วนประกอบ (Component) ฟังก์ชั่นโพรซีเดอร์ (Function Procedure) และแพคเกต (Package)

- สามารถอธิบายตัวแปรที่เกี่ยวกับฟังก์ชั่นทางด้านเวลา เช่น Propagation delay, Min-Max delay, Setup, Holding Time สามารถอธิบายได้โดยตัวภาษา

- ภาษา VHDL เป็นมาตรฐานที่ใช้โดยบริษัทและผู้ออกแบบหลาย ๆ แห่ง ฉะนั้นจึงง่ายที่จะทำความเข้าใจถึงแม้ว่าจะมาจากแหล่งต่าง ๆ

- โมเดลที่สร้างขึ้นสามารถจำลองการทำงานได้ เพราะว่าตัวแปรภาษาได้ตรวจสอบตัวแปรทางซิมูเลชั่นซีเมนติกไว้ด้วย

3.3 หลักการสร้างโมเดลโดยใช้ภาษา VHDL (General VHDL Modelling Principles)

VHDL เป็นภาษาที่ใช้สำหรับอธิบายการทำงานของฮาร์ดแวร์ในรูปแบบฟอร์มที่อ่านเข้าใจได้ ซึ่งจะช่วยในการสร้างและออกแบบวงจรรวมดิจิตอลและส่วนประกอบต่าง ๆ อาจใช้อธิบายได้ทั้งระบบเพียงบางส่วน ซึ่งอยู่ในรูปของ (Component Block) จากนั้นจะทำการจำลองการทำงาน (Simulate) โดยที่รูปแบบนั้นยังไม่ได้สร้างขึ้นจริงหรือเพียงแต่อยู่ในรูปของคำอธิบายเท่านั้น (Textual Format) หลังจากการจำลองการทำงานจนได้ตามที่ต้องการจึงนำไปทำการ Synthesis เพื่อให้ได้วงจรเกทเลเวลต่อไป ประโยชน์จริงของการใช้วงจร VHDL เป็น Design Tools แทนการสร้างต้นแบบ (Prototype) ขึ้นมาจริง คือเราสามารถอธิบาย Product Idea, Product, Proposal, Product Specification ในรูปของ Text จากนั้นก็นำคอมไพล์เพื่อดู Timing การทำงานแล้วแก้ไข (Refine) จนกว่าจะได้ Specification ตามต้องการ เมื่อ product ได้ผลตามที่ต้องการแล้วจึงนำไปสู่การสังเคราะห์ (Synthesis) เพื่อให้ได้เกทเลเวล Schematic เพื่อนำไปสร้างเป็นต้นแบบจริงต่อไป ซึ่งต้นแบบที่สร้างนั้นทำงานได้จริงเพราะได้ทำการ Simulate เรียบร้อยแล้ว เป็นการลดเวลาและค่าใช้จ่ายในการสร้างแบบได้มาก

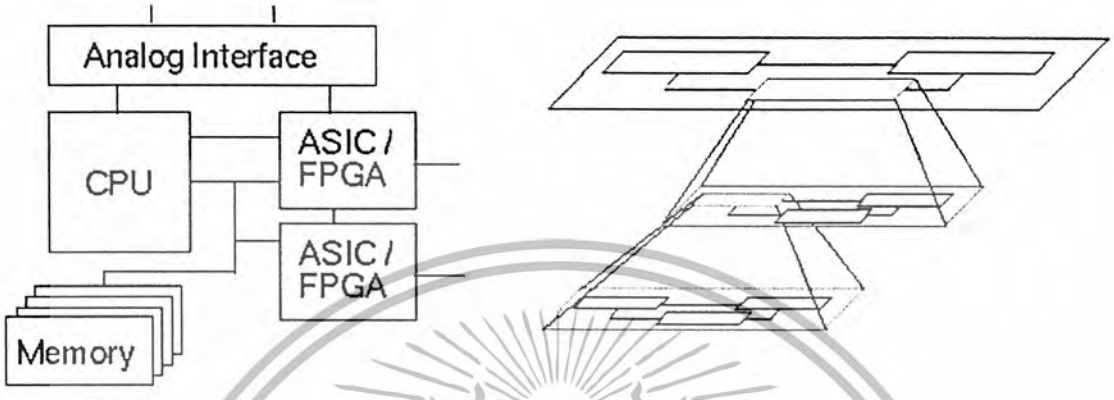
ตัวภาษา VHDL สนับสนุนหลักการต่าง ๆ ให้เขียนแก้ไขและบำรุงรักษาวงจรดิจิตอลที่มีความซับซ้อนให้เป็นไปอย่างรวดเร็วและมีประสิทธิภาพโดยมีหลักการดังนี้

3.3.1 Top Down Design

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน เช่น ASIC (Application Specific Integrated Circuit) วิศวกรหรือผู้ออกแบบมักจะมองรูปแบบให้อยู่ในรูปของ Block Diagram เสียก่อนก่อนที่จะย่อยรูปแบบให้ถึงรายละเอียดต่อไป ซึ่งภาษา VHDL นั้น อนุญาตให้อธิบายการทำงานของแต่ละ Block วิเคราะห์การทำงาน จัดการแก้ไขและปรับปรุงการทำงานจากการวิเคราะห์เพื่อให้ได้การทำงานตามที่ต้องการ ก่อนที่จะทำการออกแบบให้ละเอียดลึกลงไปในขั้นตอนต่อไป การแก้ไขในขั้นตอนนี้จะทำให้ลดค่าใช้จ่ายกว่าการแก้ไขในช่วงของการพัฒนาในระดับสร้างซิลิกอนชิพ

3.3.2 Modularity

Modularity คือ หลักการในการแยกส่วน (Partitioning) ฮาร์ดแวร์ ออกเป็นส่วนย่อยเล็กลงไป ซึ่งปกติการทำงานของฮาร์ดแวร์ใหญ่ ๆ ต้องประกอบด้วยฮาร์ดแวร์ย่อย ๆ ลงไปดังรูปที่ 3.1

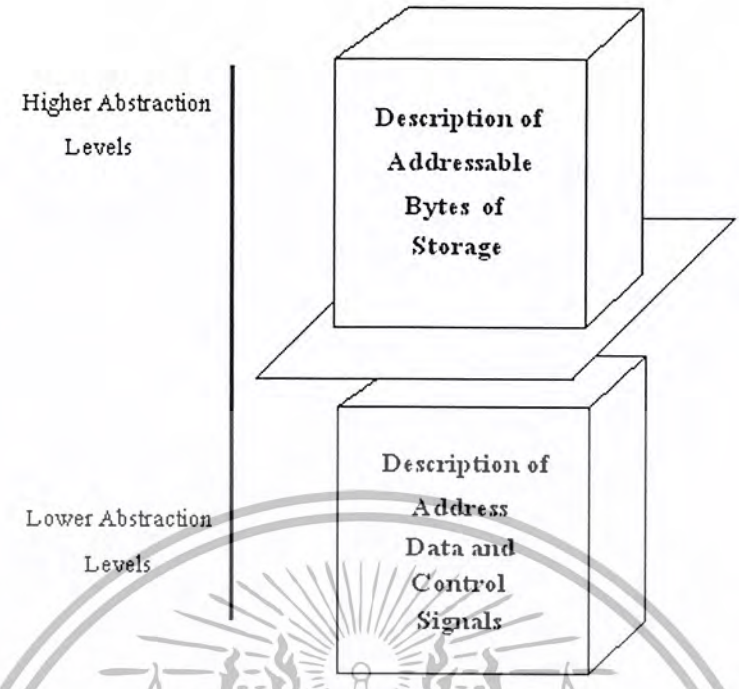


รูปที่ 3.1 การแบ่งย่อยในระดับราบของการออกแบบฮาร์ดแวร์

จากรูปได้แสดงวงจรทั้งหมดในรูปเดียว (Flatten Design) หลังจากนั้นตัดเป็นส่วนย่อยๆ เล็กลงมา เมื่อเราออกแบบโดยใช้ภาษา VHDL หน้าที่การทำงานของแต่ละส่วนต้องสามารถอธิบายได้โดยโมดูลของโค้ด (คล้ายฟังก์ชั่นหรือโพรซีเตอร์) ซึ่งแสดงการทำงานของส่วนย่อยนั้นอย่างชัดเจน ซึ่งการแยกรูปใหญ่ ๆ ออกเป็นส่วนย่อย ๆ นี้ ทำให้ง่ายต่อการจัดการและง่ายต่อการทำความเข้าใจ

รูปที่ 3.1 แสดง Hierarchy Method โดยการแยกส่วนรูปออบเป็นส่วนย่อย ๆ ส่วนบนสุด อธิบายการทำงานของ Shifter ส่วนล่าง ๆ ลงมา คือการแยกของ Shifter ออกเป็นฟลิบฟลอบจาก ฟลิบฟลอบแยกเป็น NAND เกท ภายใน Shifter ได้อธิบายถึงการทำงานโดยใช้การต่อกันของฟลิบฟลอบก็เกิดจาก NAND เกท ซึ่งมีการอธิบายการทำงานอยู่ภายใน โดยแต่ละโมดูลจะมีคำอธิบายการทำงานในตัวของมันเองอยู่แล้ว คำอธิบายในแต่ละโมดูลมีไว้เพื่อสามารถใช้ฟลิบฟลอบโมดูลก็อธิบายการเชื่อมต่อไว้อย่างดี ทำให้สามารถเชื่อมต่อกับ NAND เกท ในระดับล่างสุดได้ ประโยชน์อย่างหนึ่งของการแยกส่วนฟลิบฟลอบและ NAND เกท ออกจากกัน เนื่องจากทำให้ง่ายในการใช้ Nand เกท ตัวนี้ในรูปแบบไฮเวลตัวอื่น ๆ ทำให้ออกไปใช้งานได้อีก และลดความซับซ้อนในการใช้อุปกรณ์เพื่อแก้ไขการทำงานของ Shifter ง่ายขึ้น โดยปราศจากการแก้ไขฟลิบฟลอบและ NAND เกท ประโยชน์ที่ได้จากการทำ Modularity นี้ ทำให้รูปแบบที่ออกแบบง่ายต่อการเข้าใจและแก้ไขได้เสมอ

3.3.3 **Abstraction** คำนิยามของรูปแบบ จะอธิบายการทำงานของตัวรูปแบบมากกว่าจะอธิบายว่าพัฒนาตัวรูปแบบนั้นได้อย่างไร หลักการนี้มีความสำคัญอย่างใกล้ชิดกับหลักการของ Modularity ฟลิบฟลอบ เป็นการนิยามในการใช้ NAND เกท และ Shifter เป็นนิยามในการใช้ฟลิบฟลอบ

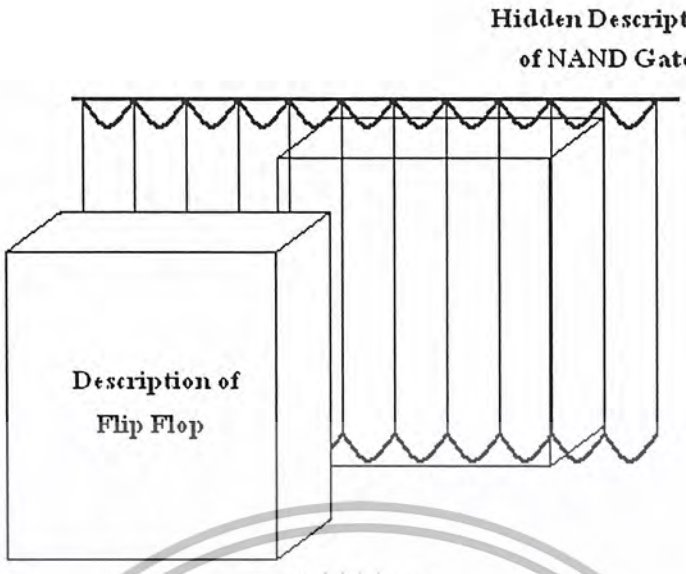


รูปที่ **3.2** Applying Abstraction to a ROM Description

รูปที่ 3.2 แสดงถึงการอธิบายการทำงานของรูปแบบโดยใช้ VHDL ในหลาย ๆ ระดับของการนิยาม ROM (Read Only Memory) อธิบายโดยใช้ภาษาระดับสูง แสดงถึงตำแหน่งต่าง ๆ ซึ่งเก็บข้อมูลไว้ในตำแหน่งนั้น ๆ ที่ระดับนี้ไม่ต้องสนใจถึง Address Line Data Line และ Control line เราสามารถพุ่งจุดสนใจไปที่ขนาดของข้อมูลโดยไม่ต้องคำนึงถึงสัญญาณควบคุมต่าง ๆ ภายในเพราะว่าส่วนนั้นจะถูกจัดการเองในระดับที่ต่ำลงมา ในระดับล่างลงมาเราสามารถอธิบายการทำงานของสัญญาณแต่ละเส้นภายในการที่จะอ่านข้อมูลหรือโปรแกรมข้อมูลใน ROM ถ้าต้องการเปลี่ยนค่าข้อมูลภายใน ROM ควรแก้ระดับที่สูงขึ้นมาจะง่ายกว่าการควบคุมสัญญาณภายใน จะเห็นว่าแต่ละระดับมีความเหมาะสมแตกต่างกันออกไป ง่ายต่อการแก้ไขโดยการใช้ประโยชน์ของ Abstraction

3.3.4 Information Hiding

เมื่อทำการเขียน VHDL code ขึ้นมาเพื่ออธิบายการทำงานของฮาร์ดแวร์ตัวหนึ่ง บางครั้งอาจต้องการที่จะซ่อนรายละเอียดการพัฒนาโมดูลนั้นโดยไม่ต้องการให้ส่วนโมดูลอื่น ๆ รู้การทำงานภายใน Information Hiding มีประโยชน์คือ ทำให้รูปแบบภาษา VHDL นั้น สามารถจัดการและอ่านเข้าใจได้ง่าย หลักการนี้จะสนับสนุนหลักการ Abstraction คือสนใจรายละเอียดในการใช้งานมากกว่าจะสนใจว่ารูปแบบนั้นจะถูกสร้างขึ้นมาอย่างไร เป็นต้น การซ่อนรายละเอียดภายในโมดูลทำให้ความสนใจของผู้ออกแบบนั้นสนใจไปในส่วนที่สำคัญมากกว่าในส่วนที่ไม่น่าสนใจจะซ่อนไว้และเข้าถึงไม่ได้ โดยแสดงดังรูปที่ 3.3



รูปที่ 3.3 การซ่อนรายละเอียดที่ไม่จำเป็นของระดับ NAND เกท

อธิบายการทำงานของฟลิบฟลอบไม่ต้องสนใจว่า NAND เกท จะทำงานอย่างไร จะต่อกันภายในอย่างไร โดย NAND เกท สามารถเขียนขึ้นแล้วคอมไพล์เก็บไว้ในไลบรารี่ ผู้ที่ออกแบบฟลิบฟลอบระดับสูงขึ้นมาเพียงแต่ต้องรู้ว่า จะเชื่อมต่ออินพุท / เอาท์พุทของ NAND เกท มาใช้งานได้อย่างไร โดยไม่ต้องสนใจว่า NAND เกท จะถูกสร้างและพัฒนามาอย่างไร ประโยชน์อีกอย่างหนึ่งคือป้องกันข้อมูลภายใน ในกรณีที่แจกจ่าย VHDL โมเดลไปยังที่อื่นทำให้เราป้องกันทรัพย์สินทางปัญญาได้อีกระดับหนึ่ง

3.3.5 Uniforminity

Uniforminity เป็นหลักอีกอย่างหนึ่งที่ช่วยในการอธิบายฮาร์ดแวร์ด้วยภาษา VHDL หมายถึง การสร้างโมดูลของรหัส ในลักษณะคล้ายกัน โดยใช้ตัวภาษา VHDL Building Block ทำให้เกิดการเขียนรหัสที่ดีอย่างเช่น มีการใช้ย่อหน้า มีคำอธิบาย (Comment) เป็นต้น ทำให้การพัฒนาโมดูลทำความเข้าใจง่าย

3.4 องค์ประกอบพื้นฐานใน VHDL (Basic concept VHDL)

รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบใน VHDL ประกอบด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม (Architecture) ดังแสดงในรูปที่ 3.7 การบรรยายเชื่อมต่อจะขึ้นต้นด้วยคำ ENTITY ตามด้วยชื่อขององค์ประกอบและคำ IS ภายในบรรยายถึงพอร์ตการติดต่อ อินพุท/เอาท์พุท พอร์ตขององค์ประกอบ ส่วนลักษณะกายภาพนอก อื่น ๆ เช่น เวลา อุณหภูมิ ก็สามารถเข้าไปในส่วนนี้ได้เช่นกัน ในส่วนของการทำงานขององค์ประกอบ หน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญา อินพุท/เอาท์พุท และพารามิเตอร์อื่น ๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูป 3.4 การบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังคำว่า BEGIN เป็นต้น

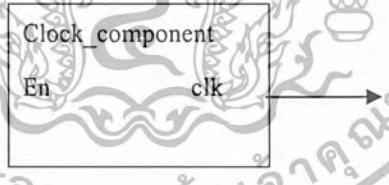
```
ENTITY  component_name IS
        Input and output poports.
        Physical and otheparameter'
END component_name;
```

```
ARCHITECTURE identifier OF component_name IS
Declaration
BEGIN
    Specification of functionality of the component
    Interms of its input linea and as influenced
END indentifier;
```

รูปที่ 3.4 แสดงการกำหนดเชื่อมต่อและสถาปัตยกรรม

3.4.1 การกำหนดการเชื่อมต่อ (Interface Description)

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ ในระดับนี้จะต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบภายนอกอื่น ๆ ดังตัวอย่างในรูปที่ 3.5 บรรทัดแรกเป็นการกำหนดชื่อองค์ประกอบซึ่งกำหนดให้เป็นชื่อ clock componene ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ในวงเล็บ IN และ OUT กำหนดโหมดของสัญญาณเป็นอินพุทหรือเอาท์พุท BIT แสดงชนิดของข้อมูล




```
ENTITY clock_component IS
    PORT (en : IN BIT; clk : OUT ?BIT)
END clock_component;
```

รูปที่ 3.5 แสดงบล็อคไดอะแรกมและการบรรยายการเชื่อมต่อของ clock component

3.4.2 การกำหนดรูปแบบของการบรรยาย (Architecture Description)

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ การบรรยายสามารถกำหนดค่าของสัญญาณเอาท์พุทในเทอมของ clock component ในรูปที่ 3.6 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมมี en เป็น

อินพุท และมี clk เป็น Process เป็นค่าเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม ภายใน Process กำหนดให้ periodic เป็นตัวแปรที่มีค่าเริ่มต้นเป็น "0" ถ้าสัญญาณ en มีค่าเป็น "1" ค่าของ periodic ถูกคอมพาลีเมนต์และส่งค่าให้กับ clk ซึ่งเป็นสัญญาณเอาท์พุทคำสั่ง WAIT กำหนดให้สัญญาณทีคาบเป็นเวลา 1 ไมโครวินาที

```
ARCHITECTURE behavioral OF clk_component IS
BEGIN
        PROCESS
        VARIABLE peridic : BIT : = '0';
        BEGIN
            IF en = '1' THEN
            Clk <= periodic;
            END PDOCESS;
END clk_component;
```

รูปที่ 3.6 แสดงการบรรยายเชิงพฤติกรรมของ Clock_component

3.4.3 หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่าง ๆ ตลอดจนโปรแกรมย่อยที่เป็นประโยชน์ต่อการเขียนรูปแบบการบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนของแพ็กเก็ต ซึ่งหน่วยการออกแบบต่าง ๆ เช่น หน่วยการออกแบบ Entity หน่วยการออกแบบสถาปัตยกรรมหรือ หน่วยการออกแบบแพ็กเกจอื่น ๆ สามารถเรียกข้อมูลเหล่านี้ไปใช้ได้ นอกจากนั้นสิ่งที่นิยมทำกันมากคือการนำรูปแบบมาตรฐานต่าง ๆ เช่น อุปกรณ์มาตรฐาน มาเก็บไว้ในรูปของแพ็กเกจ ที่ทุกคนสามารถเข้าถึงได้ ตามปกติแล้วแพ็กเกจ (Package body) เนื่องจากแพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษา VHDL สามารถกระทำได้ด้วยชุดคำสั่ง

3.4.3.1 PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ ได้แก่ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจ สำหรับประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่าและพฤติกรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือ จุดเชื่อมต่อ หรือพอร์ต ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้น โดยทั่วไปแล้วแพ็กเกจสามรถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศชนิด (Type) หรือสัญญาณ เช่นเดียวกับส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมีส่วนของการประกาศแพ็กเกจ แต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จากรูปแบบอื่นได้

```
Package package_name IS
        Package_declarative_part
End package_name;
```

รูปที่ 3.7 โครงสร้างทั่วไปของส่วนการประกาศแพ็กเกจ

3.4.3.2 PACKAGE BODY

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อย ๆ นั้น ได้ถูกประกาศไปแล้วในส่วนของการประกาศแพ็กเกจ จะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึงการกำหนดค่าคงที่ต่าง ๆ อันได้แก่ ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อนในแพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการประกาศชื่อที่เป็นโปรแกรมย่อยหรือค่าคงที่ การเขียนบอดีแพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในรูปที่ 3.8

```
Package Body package_name Is
        declarative_part
End package_name;
```

รูปที่ 3.8 โครงสร้างของบอดี้แพ็กเกจ

3.4.4 หน่วยการออกแบบ Configuration

ดังที่ทราบกันแล้วว่าระบบดิจิตอลรูปแบบหนึ่งไม่ว่าจะเป็นอะไรก็ตาม จะสามารถมีหน่วยการออกแบบ Entity ได้เพียงหนึ่งเดียวเท่านั้น ซึ่งในหน่วยการออกแบบ Entity หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมีหน่วยการออกแบบ Configuration มาเพื่อกำหนดการใช้ Configuration ของการประกอบ Entity กับหน่วยการออกแบบสถาปัตยกรรมหน่วยใด ๆ เข้าด้วยกัน

```
Configuration identifier Of entity_nane Is
        Configuration_declarative_part
End;
```

รูปที่ 3.9 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

3.4.5 โปรแกรมย่อย (Subprogram)

การใช้ฟังก์ชั่นและโพรซีเดอร์ VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่ว ๆ ไป ค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อย อาจมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่น ถ้าเราให้ฟังก์ชั่นแทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรตรรกะจริง ๆ ในขณะที่เราใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูลหรือในการคำนวนค่าหน่วงเวลา แล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์

รูปที่ 3.10 แสดงการใช้ฟังก์ชั่น โพรซีเดอร์เพื่อเปลี่ยนข้อมูลชนิด 8 บิต เป็นค่าจำนวนเต็ม รูปที่ 3.11 แสดงการใช้ฟังก์ชั่นโดยกำหนดให้ x เป็นตัวแปรชนิดบิตแทนการกระทำในสมการบูลีน

```
Type byte Is Array (7 downto 0) Of bit;
...
Procedure  byte_to_integer (ib : In byte; oi : Out integer) IS
                    Variable result : integer : = 0;
Begin
                    For I In To 7 Loop
                        If ib (i) = '1' Then
                            Result := result +2**i;
                        End if;
                    End Loop;
                    Oi := result;
```

รูปที่ 3.10 แสดงการใช้โพรซีเดอร์

```
Funtion f(a,b,c:Bit)Return bit IS
        VARIABLE x:bit
Begin
        x:=((not a) and (not b) and c);
        Return x;
End f;
```

รูปที่ 3.11 แสดงการใช้ฟังก์ชัน

3.4.6 โอเปอร์เรเตอร์ (VHDL OPERATORS)

การบรรยายเชิงพฤติกรรมใน VHDL ก็มีตัวกระทำทางลอจิกและคณิตศาสตร์เช่นเดียวกันกับภาษาซอฟแวร์ทั่วไปดังรูปที่ 3.12

PREDEFINED OPERATORS

LOGICAL OPERATOR : NOT AND OR NAND NOR XOR

OPERAND TYPE : BIT BOOLEAN

RESULT TYPE : BIT BOOLEAN

RELATONAL OPERATORS : = /= < <= > >=

OPERAND TYPE : BIT BOOLEAN

RESULT TYPE : BOOLEAN

ARITHMETIC OPERATORS : + - * / ** MOD REM ABS

OPERAND TYPE : INTEGER REAL PHYSICAL

RESULT TYPE : INTEGER REAL PHYSICAL

CONCANTINATION OPERATORS : &

OPERAND TYPE : ARRAY OF ANY TYPE

RESULT TYPE : ARRAY OF ANY TYPE

รูปที่ 3.12 แสดงตัวกระทำใน VHDL

3.4.7 เวลาและความพร้อมเพรียง (Timing and Concurrency)

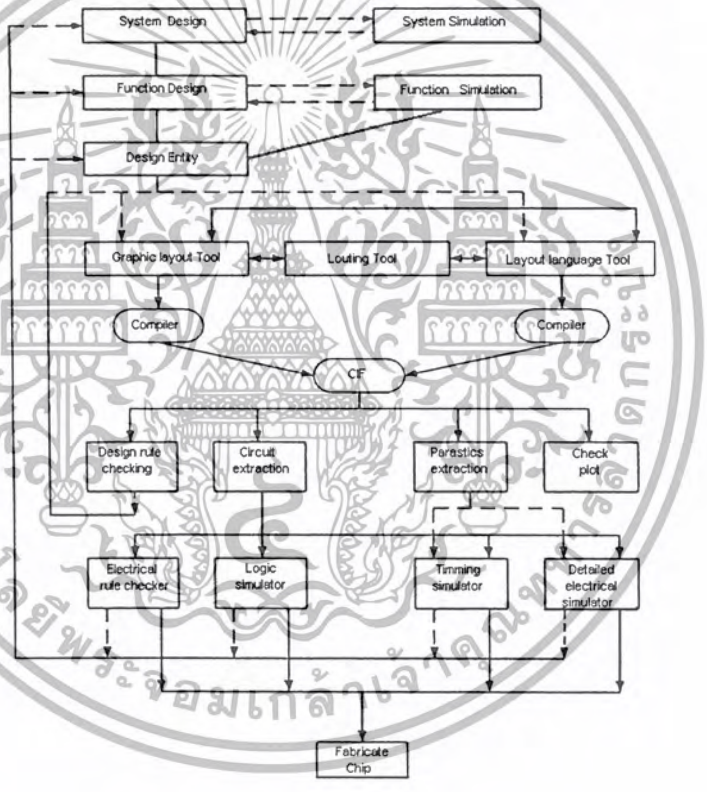
ในวงจรอิเล็กทรอนิกส์ อุปกรณ์ทุกตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ (always active) และจะมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยเสมอในทุก ๆ เหตุการณ์ที่เกิดขึ้น VHDL เป็นภาษาที่ได้รับการออกแบบมาเพื่อสามารถบรรยายรูปแบบและการพ้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงาน ที่อยู่ภายในส่วนของสถาปัตยกรรม การบรรยายจะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โพรเซสที่มีการทำงานภายในเป็นแบบลำดับคำสั่งก็ตาม หากมีหลาย ๆ โพรเซสอยู่ภายในโครงสร้างเดียวกันทุก ๆ โพรเซสก็จะทำงานไปพร้อม ๆ กันด้วย

3.4.8 สัญญาณและตัวแปร (Signals and Variable)

ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวคิดเบื้องต้นจนกระทั่งออกมาเป็นอุปกรณ์ฮาร์ดแวร์ที่ใช้งานได้ จะต้องผ่านขั้นตอนต่าง ๆ และในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์สุดท้ายในแต่ละขั้น ทำการเพิ่มเติมความจำเป็นและเข้ากระบวนการออกแบบในขั้นนั้นต่อไป

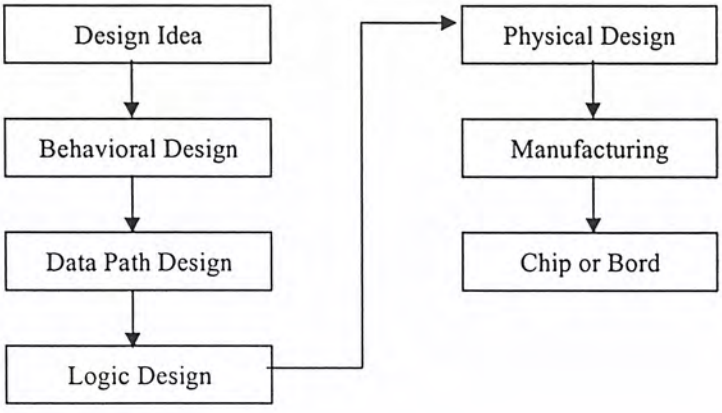
3.5 โครงสร้างของ VHDL

ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวความคิดเบื้องต้น จนกระทั่งได้ออกมาเป็นอุปกรณ์ฮาร์ดแวร์ที่ใช้งาน จะต้องผ่านขั้นตอนต่าง ๆ มากมาย และในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพทธ์สุดท้ายในแต่ละขั้น ทำการเพิ่มเติมความจำเป็นและเข้ากระบวนการออกแบบในขั้นต่อไป โครงสร้างการออกแบบวีเอชดีแอลสามารถเขียนเป็นแผนผังดังรูปที่ 3.13



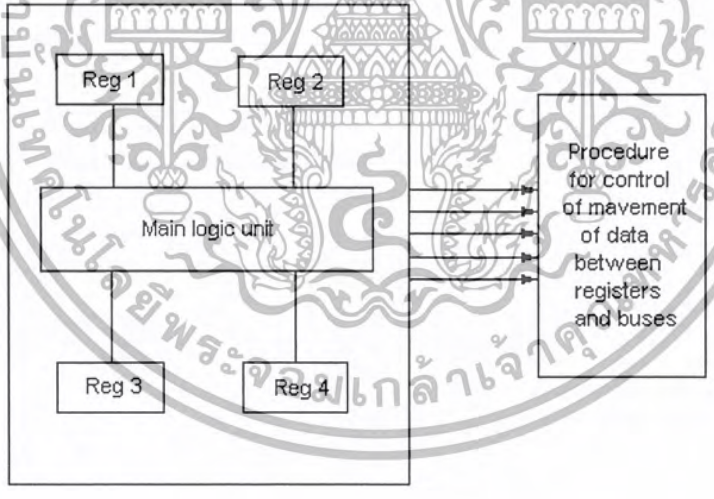
รูปที่ 3.13 รูปแสดงโครงสร้างการออกแบบวงจรมารวมโดยใช้ VHDL

รูปที่ 3.14 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิตอลทั่ว ๆ ไป ซึ่งขั้นแรกผู้ออกแบบกำหนดแนวคิดในการออกแบบเสียก่อนและทำการพัฒนาให้สามรถนำมาใช้ได้อย่างสมบูรณ์ ดังนั้นในขั้นตอนนี้จึงจำเป็นที่ผู้ออกแบบจะต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบซึ่งอาจเป็นผังแสดงแบบหรือรหัสคำสั่งเทียม (Pseudo Code)



รูปที่ 3.14 แสดงขั้นตอนการออกแบบระบบดิจิตอล

ขั้นตอนต่อไปเป็นการออกแบบระบบเส้นทางของข้อมูล ผู้ออกแบบจะต้องกำหนดส่วนประกอบรีจีสเตอร์ ผู้ออกแบบจะกำหนดส่วนของรีจีสเตอร์ (Register) และวงจรตรรกะ (Logic) ที่จำเป็นทั้งหมดที่ประกอบกันเป็นระบบที่สมบูรณ์ แต่ละองค์ประกอบสามารถเชื่อมต่อกันด้วยบัสหนึ่งหรือสองทิศทาง (Unidirectional or Bitdirectional Bus) กระบวนการควบคุมในการเคลื่อนย้ายข้อมูลระหว่างรีจิสเตอร์และวงจรตรรกะจะขึ้นอยู่กับพฤติกรรมของระบบที่กำหนดไว้ ดังรูปที่ 3.15



รูปที่ 3.15 แสดงการออกแบบระบบเส้นทางของข้อมูล

การออกแบบวงจรตรรกะจะเป็นขั้นตอนต่อไป การออกแบบในขั้นนี้เกี่ยวข้องกับการใช้เกทพื้นฐาน และฟลิบฟลอบเป็นส่วนของอุปกรณ์แยกต่าง ๆ ได้แก่ รีจีสเตอร์เก็บข้อมูลวงจรตรรกะและส่วนควบคุมฮาร์ดแวร์ในสุดท้ายจะได้ออกมาเป็นเครือข่ายของการโยงใยระหว่างเกทและฟลิบฟลอบนั่นเอง ต่อมาเป็นขั้นตอนของการเปลี่ยนแปลงเครือข่ายโยงใยในขั้นตอนที่แล้วให้เป็นทรานซิสเตอร์และเลย์เอาท์

(Transistor list and layout) ในขั้นตอนนี้เกี่ยวข้องกับการจัดวางเกทและฟลิบฟลอยแทนด้วยทรานซิสเตอร์หรือไลบรารีเซล ขั้นตอนสุดท้ายเป็นการส่งระบบที่ออกแบบไว้ไปทำการเจือสารที่โรงงานเพื่อผลิตออกมาเป็นวงจรรวมในที่สุด

บทที่ 4
ระบบการเข้ารหัสลับ (Cryptosystem)

4.1 หลักการเบื้องต้นของระบบเข้ารหัสลับ

ระบบการเข้ารหัสลับ คือ การนำข้อมูลข่าวสาร ไปแปลงจนทำให้มีคนเพียงจำนวนหนึ่งเท่านั้นที่สามารถเข้าใจถึงความหมายเดิมได้ เราเรียกข้อมูลเดิม, ขบวนการแปลงข่าวสารและข่าวสารที่ถูกแปลงแล้วว่า เอกสารปกติ (Plaintext) การเข้ารหัสลับ (Encryption) และ เอกสารลับ (Ciphertext) ตามลำดับ ส่วนขบวนการที่ใช้ในการแปลงเอกสารลับให้กลับเป็นเอกสารเดิมเรียกกว่า การถอดรหัส (Decryption) การเข้ารหัสลับจะประกอบด้วยอัลกอริทึม และตัวแปรต่าง ๆ ที่เรามักแทนด้วยสัญลักษณ์ ที่เรียกกว่า "กุญแจรหัส (Key)" ในการสื่อสารถึงแม้ว่าช่องสัญญาณจะถูกดังฟัง (Wiretap) หากผู้ดังฟังไม่มีกุญแจรหัสก็ไม่สามารถจะถอดรหัสกลับมาเป็นเอกสารปกติเดิมได้โดยง่าย

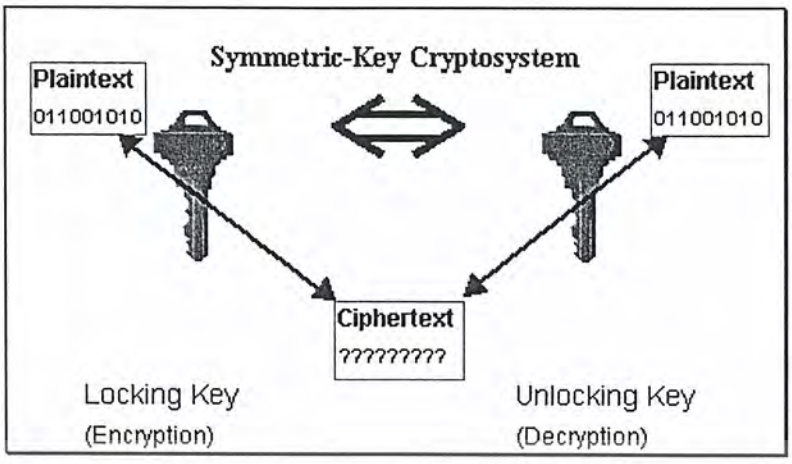
การโจมตี (Attack) เพื่อทำลายรหัส คือ การที่ผู้บุกรุกพยายามที่จะถอดรหัสลับที่ได้จากการดักฟังโดยไม่มีกุญแจรหัส แต่อาจมีข้อมูลบางส่วนที่เกี่ยวกับเอกสารที่ถูกเข้ารหัส ความยากง่ายของการทำลายรหัสนั้นขึ้นอยู่กับข้อมูลที่สามารถหาได้ และวิธีการต่าง ๆที่ใช้ในการวิเคราะห์รหัส (Crytanalysis) รหัสลับถือว่ามีความปลอดภัยสูงจะต้องมีคุณสมบัติต่าง ๆ เช่น ต้องใช้กุญแจที่มีขนาดใหญ่ (เช่น จำนวนบิทมาก) ต้องมีภูมิต้านทานต่อการวิเคราะห์รหัสที่มีประสิทธิภาพชนิดต่างๆ ที่รู้จักกันในปัจจุบันฯลฯ อย่างไรก็ดีรหัสที่มีคุณสมบัติเหล่านี้ครบ ไม่จำเป็นต้องเป็นรหัสที่มีความปลอดภัยสูง รหัสลับจำนวนไม่น้อยที่ดูเหมือนจะมีความปลอดภัยสูง ในครั้งแรกแต่ทำลายอย่างง่าย ๆ ในภายหลัง

การเข้ารหัสลับในยุคใหม่ได้รับความสนใจอย่างสูงเมื่อการเข้ารหัสแบบ DES (Data Encryption Standard) ปรากฏตัวขึ้น การที่รัฐบาลสหรัฐยอมรับอย่างเป็นทางการให้ DES เป็นมาตรฐานการเข้ารหัสลับยิ่งเป็นการกระตุ้นให้มีการศึกษา และวิจัยเกี่ยวกับการเข้ารหัสอย่างมาก หลังจากนั้นการค้นพบรหัสลับระบบกุญแจสาธารณะรวมทั้งการนำรหัสลับมาใช้ในกระบวนการพิสูจน์ความเป็นเจ้าของ (Athenication) ทางอิเล็กทรอนิกส์ ยิ่งทำให้ขอบเขตการใช้งานขยายวงกว้างออกไปอย่างมาก

ระบบการเข้ารหัสลับสามารถแบ่งออกได้เป็น 2 ประเภทใหญ่ๆ คือ รหัสลับระบบกุญแจปกปิด (Secret-Key Cryptosystem) และ รหัสลับระบบกุญแจสาธารณะ (Public-Key Cryptosystem)

4.2 รหัสสลับระบบกุญแจปกปิด

รหัสระบบกุญแจปกปิดเป็นรหัสแบบดังเดิมที่ใช้กันมานานแล้ว ในระบบนี้ทั้งผู้ส่งและผู้รับจะต้องมีกุญแจรหัสที่เหมือนกัน โดยที่กุญแจนี้จะต้องถูกปกปิดเป็นความลับระหว่างผู้ส่งและผู้รับเท่านั้น ผู้ส่งจะใช้กุญแจรหัสในขบวนการเข้ารหัส ส่วนผู้รับจะใช้กุญแจในการถอดรหัส



รูปที่ 4.1 รหัสระบบกุญแจปกปิด

4.2.1 หลักการทำงานของรหัสระบบกุญแจปกปิด

การเข้ารหัสระบบดิจิตอลในปัจจุบันเป็นแบบที่เรียกว่าการเข้ารหัส แบบบล็อค (Block Cipher) การเข้ารหัสแบบบล็อคคือการเข้ารหัสลับโดยการแบ่งข้อมูลออกเป็นบล็อค โดยที่แต่ละบล็อคมีขนาดเท่ากัน โดยเครื่องเข้ารหัสจะทำการเข้ารหัสด้วยอัลกอริทึมเดียวกันทีละบล็อค โดยทั่วไปแล้วยิ่งขนาดของบล็อคใหญ่เท่าไหร่ก็จะสามารถสร้างรหัสที่ปลอดภัยได้มากยิ่งขึ้น เมื่อเกือบ 50 ปีที่แล้ว C.E. Shannon ปรมาจารย์ผู้ให้กำเนิดวิชา "ทฤษฎีข่าวสาร" (Information Theory)" ได้ให้นิยามหลักการสำคัญของการเข้ารหัสลับแบบบล็อคไว้ ซึ่งยังคงใช้ได้จนกระทั่งปัจจุบัน ดังต่อไปนี้

โดยทั่วไปแล้วรหัสลับจะมีฟังก์ชันการทำงานที่สำคัญ 2 อย่าง คือ คอนฟิวส์ชัน (Confusion) และดิฟฟิวส์ชัน (Diffusion) ในการเข้ารหัส คอนฟิวส์ชันของรหัสจะทำหน้าที่ซ่อนความสัมพันธ์ระหว่างส่วนประกอบสำคัญ 3 ส่วน คือ เอกสารปกติ เอกสารลับ และกุญแจรหัส คอนฟิวส์ชันของรหัสที่ดีจะต้องทำให้ความสัมพันธ์เหล่านี้ยุ่งยากในเชิงสถิติ จนเครื่องมือวิเคราะห์รหัสที่มีประสิทธิภาพดีเพียงใดก็ไม่สามารถทำงานได้ ส่วนหน้าที่ของดิฟฟิวส์ชัน คือกระจายผลที่แต่ละบิตของกุญแจละเอกสารปกติที่มีต่อเอกสารลับให้มากที่สุดเท่าที่จะทำได้ ดิฟฟิวส์ชันยังมีผลในการซ่อนความสัมพันธ์ในเชิงสถิติซึ่งทำให้การวิเคราะห์รหัสยากขึ้นด้วย

ที่จริงแล้วการทำคอนฟิวส์ชันเพียงอย่างเดียวก็เพียงพอแล้วสำหรับรหัสที่ดี นั่นคือเราสามารถสร้างรหัสลับที่ยากต่อการวิเคราะห์โดยการใช้วิธีการเปิดตรง (Look – up Table) เพียงอย่างเดียว ซึ่งก็คือการใช้หน่วยความจำในการเข้ารหัสนั่นเอง ปัญหาอยู่ที่ว่าเมื่อขนาดที่ใช้แบ่งเอกสารมีขนาดใหญ่ขึ้น ขนาดของหน่วยความจำที่ต้องการใช้ก็มากขึ้นจนไม่เหมาะสมในทางปฏิบัตินั่นเอง

ตรงนี้เองที่ดิฟฟิวส์ชันถูกนำมาแก้ปัญหา โดยการเข้ารหัสแต่ละครั้งนั้น แทนที่จะใช้ตาราง (หน่วยความจำ) ขนาดใหญ่เพียงอย่างเดียว เราจะทำการคอนฟิวส์ชันโดยการใช้ตารางที่เล็กกว่าแล้วนำมาผสมกับการดิฟฟิวส์ชันหลาย ๆ ครั้งด้วยรูปแบบต่าง ๆ กัน การดิฟฟิวส์ชันอาจทำได้ง่าย ๆ โดยการสลับตำแหน่ง (Permutation) เท่านั้น การเข้ารหัสสลับแบบบล็อคที่อธิบายดังกล่าวเรียกว่า Product Cipher รหัส

สลับระบบกุญแจปกปิดที่มีใช้กันอยู่ในปัจจุบันในส่วนใหญ่จะใช้หลักการของ Product Cipher ในจำนวนนี้รหัส DES เป็นรหัสที่แพร่หลายมากที่สุด และยังคงได้รับความเชื่อถือจนกระทั่งปัจจุบัน

4.2.2 รหัสลับระบบกุญแจปกปิด DES

รหัส DES ก็เป็น Product Cipher ชนิดหนึ่งซึ่งถูกพัฒนาโดยบริษัทไอบีเอ็ม หลังจากนั้นจึงได้รับการยอมรับให้เป็นมาตรฐานการเข้ารหัสของรัฐบาลสหรัฐ DES มีข้อมูลขาเข้าและขาออกเท่ากันคือ 64 บิท มีกุญแจรหัสความยาว 56 บิท นอกจากหลักการของคอนฟิวส์ชันและดิฟฟิวส์ชันของ Product Cipher แล้ว DES ยังใช้หลักการพิเศษบางประการในการออกแบบหนึ่งในหลักการนั้นคือการทำขบวนการง่าย ๆ ซ้ำไปมาหลาย ๆ รอบเพื่อเพิ่มความแข็งแกร่งของรหัส

ลักษณะการทำซ้ำไปมาในแต่ละรอบของ DES ใช้หลักการที่เรียกกว่า Fiestel Cipher ในการเข้ารหัสแบบ Fiestel Cipher นั้นในแต่ละระบบล็อคข้อมูลความยาว n บิท จะถูกแบ่งออกเป็น 2 ส่วน คือ L และ R โดยในแต่ละส่วนมีความยาว n/2 บิทเท่ากัน ความสัมพันธ์ของข้อมูลขาออกในรอบที่ iกับi-1 ดังต่อไปนี้

$$L_i = R_{i-1} \qquad (4.1)$$

$$R_{i-1} = L_{i-1} \text{XOR } f(R_{i-1} K_{i-1}) \qquad (4.2)$$

โดยที่ XOR คือการทำ exclusive or, K_i คือส่วนของกุญแจที่ถูกใช้ในรอบที่ i และ f คือฟังก์ชั่นที่ใช้ในแต่ละรอบ การเลือกฟัง f นั้นมีผลต่อความแข็งแกร่งต่อการวิเคราะห์รหัส เพราะ f คือการคอนฟิวส์ชันนั่นเอง ส่วนการทำซ้ำคือการดิฟฟิวส์ชัน ซึ่งยิ่งทำมากรอบเท่าไหร่ความสัมพันธ์ที่แต่ละบิทของกุญแจรหัสและข้อมูลขาเข้าจะถูกกระจายไปยังข้อมูลขาออกมากยิ่งขึ้น สำหรับ DES นั้น จำนวนรอบที่ทำคือ 16 กรอบ

ข้อดีของ DES ที่สำคัญที่สุดประการหนึ่ง คือ ความง่ายของวงจรเข้า และถอดรหัส จาก (4.1) และ (2) จะเห็นว่าในแต่ละรอบมีการทำงานที่เหมือนกันจึงง่ายต่อการออกแบบวงจรให้รวดเร็ว และประหยัด ยิ่งไปกว่านั้น จาก (4.1) และ (4.2) เราจะได้ว่า

$$L_{i-1} \text{XOR } f(R_{i-1} K_{i-1}) \text{ XOR } f(R_{i-1} K_{i-1}) = L_{i-1} \qquad (4.3)$$

นั่นคือถ้าเรามีกุญแจทั้งหมด เราก็จะสามารถถอดรหัสได้ด้วยขบวนการ (วงจร) เดียวกันกับขบวนการ (วงจร) เข้ารหัส ด้วยเหตุนี้เองนอกจาก DES แล้วหลักการของ Fiestel Cipher ยังถูกนำไปใช้ในการออกแบบ Product Cipher ชนิดต่าง ๆ เช่น FEAL, LOKI ฯลฯ

ถึงแม้ว่าอัลกอริทึมในการเข้ารหัสของ DES จะถูกเผยแพร่ต่อสาธารณะชนมากกว่า 15 ปีแล้วก็ตาม และได้มีการพยายาที่จะทำลายรหัส DES อย่างนับไม่ถ้วน แต่ DES ยังคงได้รับการยอมรับ จนกระทั่งปัจจุบันว่าเป็นการเข้ารหัสลับที่มีความปลอดภัยสูงมากชนิดหนึ่ง ในปี 1993 ได้มีการประมาณว่าจะต้องใช้เงินประมาณ 1 ล้านดอลลาร์เพื่อเป็นค่าใช้จ่ายในการสร้างเครื่องพิมพ์คอมพิวเตอร์เฉพาะทางที่ใช้ใน

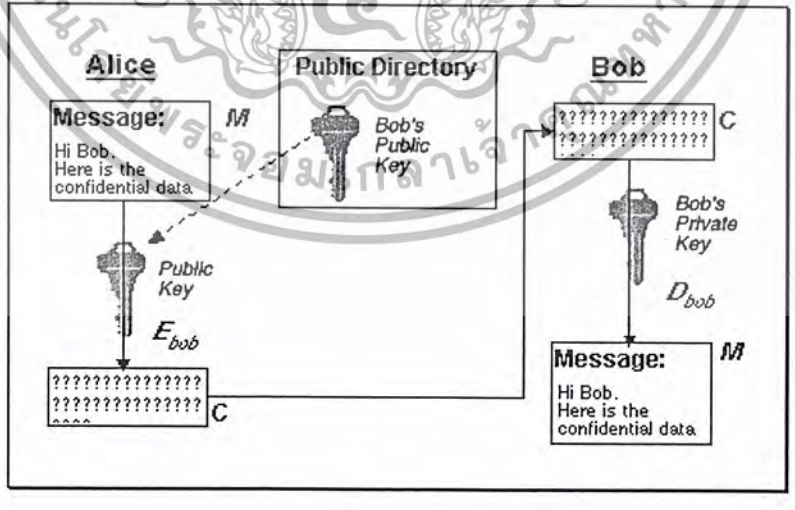
การวิเคราะห์ DES และสำหรับการส่งข้อความที่ปลอดภัยสูงนั้น การใช้ DES ต่อกันเป็นโครงข่าย เช่น Tripple-DES ซึ่งใช้การเข้ารหัสแบบ DES 3 รอบโดยใช้กุญแจ 3 ดอกจะยิ่งเพิ่มความแข็งแกร่งให้กับ DES เป็นอย่างมาก

ปัญหาใหญ่ของรหัสลับในระบบกุญแจปกปิด คือการที่ทั้งผู้รับและผู้ส่งจะต้องใช้กุญแจที่เหมือนกัน โดยกุญแจนั้นจะต้องเก็บเป็นความลับ ปล่อยให้ตกอยู่ในมือของคนอื่นไม่ได้ ดังนั้นหากผู้รับและผู้ส่งอยู่ห่างจากกันในทางกายภาพ เราจำเป็นต้องให้บุคคลที่สามที่ไว้ใจได้ หรือช่องสื่อสารที่ปลอดภัยในการส่งผ่านกุญแจ เพราะถ้าหากกุญแจนี้ถูกแอบลักลอบดังฟังหรือเปิดอ่านระหว่างการขนส่งแล้ว ผู้ลักลอบดังกฟังก็สามารถถอดรหัสลับได้ ทำให้ระบบรักษาความปลอดภัยไร้ความหมายโดยสิ้นเชิง การสร้าง การเก็บและการเปลี่ยนกุญแจรหัส มักรวมเรียกกันว่าระบบจัดการกุญแจรหัส ในการออกแบบรหัสลับระบบกุญแจปกปิดนั้นการจัดการกุญแจรหัสที่ปลอดภัยมักจะเป็นปัญหาที่ยากที่สุด

4.3 รหัสลับระบบกุญแจสาธารณะ

4.3.1 หลักการทำงานของรหัสลับระบบกุญแจสาธารณะ

รหัสลับระบบกุญแจสาธารณะถูกค้นพบครั้งแรกโดย Whitefield Diffie และ Martin Hellman ในปี ค.ศ. 1976 เพื่อแก้ปัญหาการจัดการกุญแจของรหัสลับระบบกุญแจปกปิด ในรหัสลับระบบกุญแจสาธารณะนี้สมาชิกแต่ละคนจะต้องมีกุญแจ 2 ชนิด คือ กุญแจส่วนตัว (Private Key) และกุญแจสาธารณะ (Public Key) โดยที่กุญแจส่วนตัวจะถูกเก็บไว้เป็นความลับ ส่วนกุญแจสาธารณะนั้นจะเปิดเผยให้ใครก็ได้ที่ต้องการส่งเอกสารให้แก่ตน การทำงานของรหัสลับมีหลักการว่าข้อมูลที่ถูกเข้ารหัสด้วยกุญแจสาธารณะของผู้ใด จะถูกถอดรหัสได้ด้วยกุญแจส่วนตัวของผู้นั้นเท่านั้น การทำงานของกุญแจรหัสรับระบบกุญแจสาธารณะสามารถอธิบายการทำงานได้ดังนี้



รูปที่ 4.2 รหัสลับระบบกุญแจสาธารณะ

จากรูปที่ 4.2 สมมติว่า Alice ต้องการส่งข้อความถึง Bob ก่อนอื่น Alice ต้องมีกุญแจสาธารณะของ Bob ซึ่งอาจจะขอจาก Bob โดยตรงหรือจากองค์กรที่ให้บริการ จากนั้น Alice ก็จะใช้กุญแจสาธารณะของ Bob ในการเข้ารหัสลับแล้วส่งไปใช้ Bob จากนั้น Bob จะถอดรหัสลับโดยการใช้กุญแจส่วนตัวของตน ดังนั้นจะไม่มีใครถอดรหัสได้นอกจาก Bob ขณะเดียวกัน ๆ ก็สามารถส่งข้อความที่ถูกเข้ารหัสลับด้วยกุญแจสาธารณะของ Bob ได้โดยไม่ต้องมีกุญแจส่วนตัวของ Bob จะเห็นว่าในระบบกุญแจสาธารณะนั้นผู้ส่งไม่จำเป็นต้องมีระบบขนส่งกุญแจที่ไว้ใจได้ ทำให้ลดความยุ่งยากในระบบการจัดการกุญแจรหัสลงอย่างมาก

อย่างไรก็ดี สิ่งสำคัญที่สุดที่ควรคำนึงถึงสำหรับรหัสลับระบบกุญแจสาธารณะคือ ความปลอดภัยของมันขึ้นอยู่กับว่าการที่เราจะสามารถคำนวณกุญแจส่วนตัวจากกุญแจสาธารณะได้ยากเพียงไร รหัสลับระบบกุญแจสาธารณะที่ดีนั้น การคำนวณกุญแจสวนตัวจากกุญแจสาธารณะจะต้องทำได้ยากมากหรือไม่อาจทำได้เลย นอกจากรหัสลับของ Deffie และ Hellman แล้วรหัสลับระบบกุญแจสาธารณะหลายแบบได้ถูกพัฒนาขึ้น แต่ที่ใช้กันแพร่หลายมากที่สุด คือ รหัส RSA เพราะนอกจากการเข้ารหัสเอกสารแล้วมันสามารถนำไปใช้การการลงนามระบบดิจิตอลได้ด้วย RSA แตกต่างจากรหัสลับของ Diffie และ Hellman ตรงที่ กุญแจสาธารณะของรหัสที่ Deffie และ Hellman คิดขึ้นนั้นจะเปลี่ยนแปลงทุกครั้งที่มีกรติดต่อสื่อสาร ดังนั้นก่อนส่ง Alice จำเป็นต้องติดต่อกับ Bob ทุกครั้งเพื่อขอกุญแจสาธารณะใหม่เสมอในขณะที่กุญแจสาธารณะของ RSA จะไม่มีการเปลี่ยนแปลงในการติดต่อสื่อสารแต่ละครั้ง ทำให้ลดขั้นตอนและความสิ้นเปลืองในการสื่อสารลงได้มาก แต่เนื่องจากกุญแจสาธารณะไม่มีการเปลี่ยนแปลงรหัส RSA จึงจำเป็นต้องอาศัยองค์กรที่ไว้ใจได้เป็นผู้รับรอง และเผยแพร่กุญแจสาธารณะเหล่านั้น RSA ผ่านการทดสอบจากนักวิชาการและผู้เชี่ยวชาญเป็นเวลานาน และยังคงได้รับความเชื่อถือว่าเป็นรหัสที่มีความปลอดภัยสูงมากชนิดหนึ่ง

4.3.2 รหัสลับระบบกุญแจสาธารณะแบบ RSA

รหัส RSA ถูกคิดค้นขึ้นในปี ค.ศ. 1977 โดยชื่อ RSA นั้นได้มาจากอักษรตัวแรกของนามสกุลของผู้ร่วมกันคิดค้นคือ Ron River, Adi Shamir และ Leonard Adleman หลักการทำงานของ RSA มีดังต่อไปนี้

ถ้าให้ p และ q เป็น prime number ที่มีค่ามาก โดยที่ n = pq เรียกว่าโมดูลัส (Modolus) จากนั้นจึงเลือก e ที่มีค่าน้อยกว่า n และไม่สามารถหาร (p-1)(n=1) ได้ลงตัวถ้าให้ d เป็นส่วนกลับของ e ในคณิตศาสตร์โมดูโลฐาน (p-1)(n=1) นั่นคือ

$$ed \bmod (p-1)(n=1) = 1 \qquad (4.4)$$

ในรหัส RSA (n,e) คือกุญแจสาธารณะ ส่วน d คือกุญแจส่วนตัว เมื่อได้ค่าเหล่านี้แล้ว ค่า p และ q จะต้องเป็นความลับหรือถูกทำลายในทันที

และเพราะมีแต่ Bob เท่านั้นที่รู้ค่า d จึงมีแต่ Bob เท่านั้นจึงจะถอดรหัสได้ คุณสมบัติที่สำคัญประการหนึ่งของ RSA คือ จากสมการ (4.4) ในทางกลับกันถ้าเราเข้ารหัสด้วยกุญแจส่วนตัว เราก็จะสามารถ

ถอดรหัสด้วยกุญแจสาธารณะได้ด้วยเช่นกัน คุณสมบัติข้อนี้เองทำให้ RSA มีประโยชน์มาก เพราะสามารถนำไปลงลายเซ็นดิจิตอลได้

วิธีการทำลายรหัส RSA ที่รู้จักกันดีคือ การหาค่า d นั่นเอง จากสมการ (4.4) เราอาจหาค่า d ได้หากรู้ค่า p และ q แต่เนื่องจาก p และ q เป็น prime number ที่ pq = n ดังนั้นการทำลายรหัส RSA ขึ้นอยู่กับการแยกตัวประกอบของ n นั่นเอง แต่วิธีการแยกตัวประกอบของ n นั้น ไม่ง่ายเลยถ้าหาก n มีค่ามาก R. Rivest ได้คำนวณไว้ในปี ค.ศ. 1992 ว่าจะต้องใช้เงินประมาณ 8.3 ล้านเหรียญสหรัฐในการแยกตัวประกอบของ n ที่มีความยาว 512 บิต ค่านี้อาจลดลงได้ในอนาคต ดังนั้นสำหรับข้อมูลที่มีความสำคัญมากกว่าอาจจำเป็นต้องใช้ n ที่มีค่ามากถึง 700 หรือ 1000 บิตก็ได้

จะเห็นว่าไม่ว่าการเข้ารหัสหรือถอดรหัส RSA ก็จำเป็นต้องใช้การยกกำลังในในระบบโมดูโล ซึ่งการยกกำลังนั้นสามารถทำได้โดยใช้วงจรคูณระบบโมดูโลมาต่ออนุกรมกัน ดังนั้น ความเร็วของทั้งการเข้าและถอดรหัสจึงขึ้นอยู่กับความเร็วของวงจรโมดูโลเป็นอย่างมาก นี่เองเป็นจุดอ่อนข้อหนึ่งของ RSA เมื่อเทียบกับรหัสลับระบบกุญแจปกปิด เช่น DES เพราะปัจจุบันการคูณในระบบโมดูโลยังค่อนข้างยุ่งยาก เมื่อเทียบกับการแทนค่าหรือสลับตำแหน่งใน DES ได้มีการประมาณกันว่าสำหรับ n muj,u8;k,pk; 512 บิต DES จะเร็วกว่า RSA ประมาณ 100 เท่า ถ้าทำการเข้ารหัสด้วยซอฟแวร์ และอาจเร็วกว่าถึง 1000 หรือ 1000 เท่า ซึ่งแล้วแต่ลักษณะการออกแบบของวงจรหากทำการเข้ารหัสด้วยฮาร์ดแวร์ โดยทั่วไปแล้วเราต้องการใช้วงจรเข้ารหัสเร็วกว่าการถอดรหัสดังนั้นเราจึงมักเลือกให้ e มีค่าน้อยกว่า d และยิ่งกว่านั้นเรามักให้ e ของสมาชิกทุกคนมีค่าเดียวกันเพื่อให้ฮาร์ดแวร์ของวงจรเข้ารหัสสำหรับสมาชิกแต่ละคนมีลักษณะคล้ายกัน

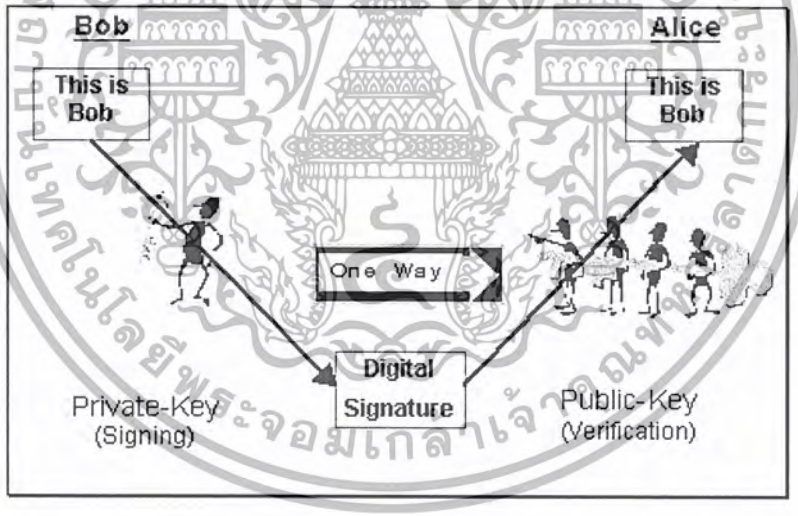
เนื่องจาก DES และ RSA มีข้อดีข้อเสียที่แตกต่างกัน จึงไม่จำเป็นว่ารหัสชนิดใดจะเหมาะสมในทุกสถานการณ์ โดยทั่วไปแล้ว DES จะถูกใช้ในการเข้ารหัสข้อมูลขนาดใหญ่เพราะรวดเร็วกว่าในขณะที่ RSA จะถูกใช้ในระบบสื่อสารที่ไม่ยาวนัก แต่ต้องการความปลอดภัยสูงในบางครั้ง RSA ยังถูกใช้ร่วมกับ DES เพื่อเสริมจุดเด่นซึ่งกันและกัน เช่นตัวเอกสารจริงจะถูกเข้ารหัสด้วย DES โดยที่กุญแจรหัส DES จะถูกเข้าด้วย RSA แล้วส่งไปด้วยกันหรือส่งไปก่อนแต่ในบางครั้ง DES อย่างเดียวก็พอแล้วหากการแลกเปลี่ยนกุญแจสามารถทำได้อย่างปลอดภัยพอ หรือในกรณีที่ผู้ส่งและผู้รับเป็นบุคคลคนเดียวกัน เช่น ฮาร์ดดิสก์ในคอมพิวเตอร์ส่วนตัว หรือข้อมูลส่วนตัวในสมาร์ทการ์ด

รหัส RSA เป็นรหัสลับระบบกุญแจสาธารณะที่รู้จักกันแพร่หลายมากที่สุดระบบหนึ่งในปัจจุบัน รหัส RSA ทั้งที่เป็นซอฟแวร์และฮาร์ดแวร์ ถูกนำไปใช้ในองค์กรต่าง ๆ มากมายทั่วโลก รหัสลับในมาตรฐาน X.509 ของ CCITT และ ISO 9676 นั้น ล้วนเป็นรหัสลับตระกูลเดียวกันกับรหัส RSA นอกจากนี้รหัส RSA ยังถูกกำหนดให้เป็นส่วนหนึ่งของมาตรฐานของ SWIFT (Society for Worldwide Interbank Financial Telecommunications)

4.4 การพิสูจน์ความเป็นเจ้าของ (Authentication) และ ลายเซ็นดิจิตอล (Digital Signature)

ในการสื่อสารระบบดิจิตอล การพิสูจน์ความเป็นเจ้าของคือขบวนการที่ทำให้ผู้รับมั่นใจได้ว่าข้อมูลที่ได้รับนั้นมาจากใครและผู้นั้นเป็นผู้ส่งจริงหรือเปล่า ตัวอย่างง่าย ๆ ที่เห็นได้ชัดได้แก่การป้อนรหัสลับพร้อมกับบัตร ATM ของธนาคาร หากเราคิดว่าเจ้าของเท่านั้นที่รู้รหัสลับประจำตัว การตรวจสอบว่าผู้ใช้ใส่รหัสถูกหรือไม่นั้นก็ถือเป็นการพิสูจน์ว่าผู้ใช้เป็นเจ้าของหรือไม่ ได้อย่างหนึ่ง การพิสูจน์ความเป็นเจ้าของทางอิเล็กทรอนิคสีสามารถทำได้โดยการใช้ขบวนการเข้ารหัสลับ ซึ่งอาจเป็นได้ทั้งรหัสลับ ระบบกุญแจปกปิด เช่น DES หรือรหัสลับระบบกุญแจสาธารณะ เช่น RSA แต่สำหรับการพิสูจน์ความเป็นเจ้าของทางอิเล็กทรอนิคส์ที่ใช้รหัสลับระบบกุญแจสาธารณะนั้นยังหมายถึงการลงลายเซ็นดิจิตอลด้วย

ในการพิสูจน์ความเป็นเจ้าของทางดิจิตอลนั้น เราอาจเปรียบเทียบได้กับการรับรองเอกสารทั่วไปก็ได้ โดยในการรับรองจะต้องมีการลงลายเซ็นจากผู้ส่งในเอกสาร ซึ่งสำหรับเอกสารทางดิจิตอลนั้นส่วนที่จะเป็นการลงลายเซ็นต์นั้นจะต้องเป็นข้อมูลเพิ่มเติมพิเศษที่จะต้องได้รับการยอมรับกันทั่วไป และยากต่อการปลอมแปลง ในขณะเดียวกันผู้รับหรือบุคคลที่สามก็สามารถพิสูจน์ได้ว่าเป็นของจริง ดังนั้นการรับรองที่ปลอดภัยนั้นจะต้องประกอบด้วย 2 ขบวนการใหญ่ ๆ คือ การลงลายเซ็นที่ไม่สามารถปฏิเสธในภายหลังได้ว่าตัวเองไม่ได้ลงนาม (Non-Repudiation) การใช้รหัสลับในการส่งข้อความที่จำเป็นจะต้องมีการพิสูจน์ความเป็นเจ้าของโดยอาจทำได้ดังวิธีต่อไปนี้



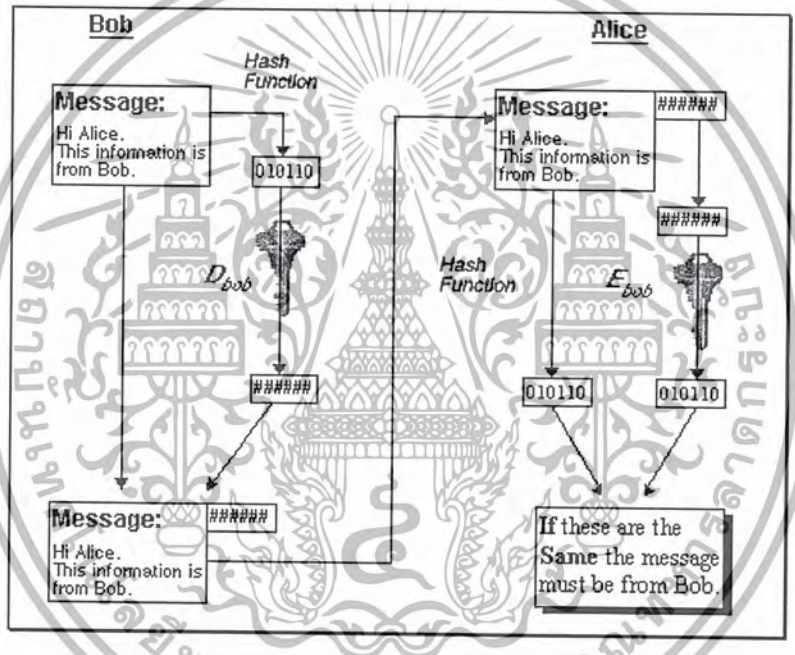
รูปที่ 4.3 การลงลายเซ็นดิจิตอลและการพิสูจน์ความเป็นเจ้าของ

ถ้า Bob ต้องการส่งเอกสารให้ Alice พร้อมกับรับรองว่าตนเองเป็นผู้เขียน นอกจากตัวเอกสารจริงแล้ว Bob จะเข้ารหัสลับเอกสารนั้นก่อนแล้วจึงส่งไปพร้อมกับเอกสารตัวจริง เมื่อ Alice ได้รับเขาจะพิสูจน์ว่าเป็นเอกสารของ Bob จริง ได้โดยการถอดรหัสเอกสารรหัสลับแล้วนำมาเปรียบเทียบกับเอกสารตัวจริงที่ส่งมาด้วยว่าตรงกันหรือไม่ ถ้าตรงกันก็ถือว่าถูกต้อง

จะสังเกตได้ว่าถ้าเราใช้รหัสลับระบบกุญแจปกปิด ทั้ง Bob และ Alice จะต้องมีกุญแจที่เหมือนกันซึ่งทำให้มีปัญหาในการนำส่งกุญแจอย่างปลอดภัย หรือจำเป็นต้องให้ Bob มาพิสูจน์ด้วยตัวเอง แต่ถ้าเราใช้

รหัสระบบกุญแจสาธารณะ โดยเฉพาะอย่างยิ่ง RSA ในตอนส่ง Bob ก็จะเข้ารหัสโดยใช้กุญแจส่วนตัวของตนเองในการเข้ารหัส และ Alice ก็จะใช้กุญแจสาธารณะของ Bob ในการพิสูจน์ ทำให้ลดความยุ่งยากในการนำส่งกุญแจไปได้ นอกจากนี้ใคร ๆ ก็สามารถพิสูจน์ลายเซ็นได้ว่าเป็นของจริงหรือไม่ โดยไม่ต้องมีกุญแจส่วนตัวของ Bob โดยทั่วไปแล้วสำหรับกรณีที่ใช้รหัส RSA ในการลงลายเซ็นดิจิตอลนั้น ก็เช่นเดียวกับการเข้ารหัสลับ คือเรามักให้ e มีขนาดน้อยกว่า d และ e มีค่าเท่ากัน สำหรับสมาชิกทุกคนนั่นหายความว่าขบวนการลงนามนั้นใช้เวลามากกว่าขบวนการพิสูจน์

อย่างไรก็ดี เนื่องจากวงจรเข้ารหัสและถอดรหัสลับระบบกุญแจสาธารณะยุ่งยากมากกว่ารหัสลับระบบกุญแจปกปิดมาก ในอุปกรณ์ที่มีขนาดเล็ก ราคาถูก เช่น สมาร์ทการ์ด ยังคงนิยมใช้รหัสลับระบบกุญแจปกปิด เช่น DES อยู่ แต่ในอนาคตเมื่ออุปกรณ์มีราคาต่ำลงอุปกรณ์เหล่านี้มีแนวโน้มที่จะเปลี่ยนมาใช้รหัสระบบกุญแจสาธารณะ



รูปที่ 4.4 การลงลายเซ็นดิจิตอลด้วยการใช้ฟังค์ชั่นแฮส

ข้อสังเกตอีกประการหนึ่งในการลงลายเซ็นดิจิตอลแบบที่อธิบายมาแล้วนั้น เราต้องส่งทั้งเอกสารปกติตัวจริง และเอกสารรหัสลับไปด้วยกัน เพื่อเป็นการยืนยันว่าเป็นการลงลายเซ็นต์ดิจิตอลที่ออกให้กับเอกสารชิ้นที่ส่งไปจริง ซึ่งนอกจากจะเป็นการสูญเสียแบด์วิท์ในการสื่อสารแล้วยังเสียเวลามากในการถอดรหัสโดยเฉพาะอย่างยิ่งเมื่อเอกสารที่ต้องการพิสูจน์ความเป็นเจ้าของนั้นมีขนาดใหญ่มาก ปัจจุบันเราแก้ปัญหานี้ โดยการใช้ "ฟังก์ชันแฮสทิศทางเดียว" (One – Way Hash Function)"

ฟังก์ชันแฮสทิศทางเดียวคือฟังก์ชันที่ทำการเปลี่ยนเอกสารที่มีความยาวขนาดต่าง ๆ ให้เป็นข้อมูลขนาดสั้น ๆ ค่าหนึ่งที่มีความยาวคงที่เรียกว่า "ไดเจสของเอกสาร (Massage Digest)" โดยทิศทางเดียวของ

ฟังก์ชันแฮส หมายความว่า เราไม่สามารถนำไดเจสเหล่านั้นมาตีความใด ๆ ที่เกี่ยวข้องกับเอกสารเดิม หรือเพื่อเปลี่ยนแปลงกลับไปเป็นเอกสารเดิมได้เลย

ดังนั้นจากคุณสมบัติของฟังก์ชันแฮสทิศทางเดียว ในการลงลายเซ็นดิจิตอลนั้นแทนที่เราจะเข้ารหัสลับกับเอกสารโดยตรง เราสามารถเข้ารหัสไดเจสของเอกสารแทนแล้วส่งไปพร้อมกับเอกสารปกติได้ และในการพิสูจน์ความเป็นเจ้าของและการลงลายเซ็นต์ระบบดิจิตอลมีความสำคัญมาก เพิ่มความปลอดภัยในระบบรักษาความปลอดภัยของข้อมูลแล้ว ยังมีประโยชน์มหาศาลต่อระบบการเงิน อำนาจแทนการลงนามจริงในเอกสารต่าง ๆ เช่น การทำสัญญา การสั่งจ่ายเงินหรือธนาคารอาจใช้รับรองข้อมูลที่ใช้แทนเงินในระบบดิจิตอล เป็นต้น

4.5 การจัดการกุญแจรหัส (Key Management) และการแลกเปลี่ยนกุญแจรหัส (Key Exchange) ในระบบดิจิตอล

ดังที่กล่าวมาแล้ว ระบบจะจัดการกุญแจรหัสที่ปลอดภัยเป็นสิ่งสำคัญมาก เพราะในทางปกิบัติการจัดการกุญแจรหัสมักจะตกเป็นเป้าหมายในการจุดจมเพื่อทำลายความปลอดภัยของระบบโครงข่ายข้อมูลมากกว่าการจู่โจมทำลายรหัสลับโดยตรงเสียอีก โดยทั่วไปแล้วระบบโครงข่ายข้อมูลทางดิจิตอลในองค์กรหรือหน่วยงานจะประกอบไปด้วยคอมพิวเตอร์ส่วนตัวและอุปกรณ์ปลายทางอื่น ๆ ที่ต่อเข้ากับระบบสื่อสารขององค์กรที่จัดให้มีการใช้รหัสลับเพื่อรักษาความปลอดภัยของข้อมูลในองค์กร และส่วนของสมาชิกนั้นจะต้องมีระบบจัดการกุญแจซึ่งมีคุณสมบัติดังต่อไปนี้

1. กุญแจรหัสจะต้องถูกแจกจ่ายหรือแลกเปลี่ยนแก่สมาชิกใด ๆ เพื่อให้ผู้รับและผู้ส่งสามารถทำการเข้ารหัสและถอดรหัสลับ หรือทำการรับรองและพิสูจน์ในขบวนการพิสูจน์ความเป็นเจ้าของในระหว่างการติดต่อสื่อสารได้
2. จะต้องมีวิธีที่ปลอดภัยที่จะป้องกันไม่ให้ผู้บุกรุกดังฟัง หรือลอบนำกุญแจไปคัดลอกระหว่างการสื่อสารจนกระทั่งสามารถปลอมเป็นสมาชิก หรือแอบอ้างเป็นสมาชิกผู้หนึ่งผู้ใดได้
3. จะต้องมีวิธีปลอดภัยที่ทำให้สมาชิกสามารถตรวจสอบได้ว่ากุญแจรหัสที่ได้รับนั้นเป็นของสมาชิกที่ต้องการติดต่อสื่อสารด้วย

ระบบจัดการกุญแจที่ง่ายที่สุด (แต่ทำปลอดภัยได้ยากมากที่สุด) คือการใช้รหัสลับกุญแจปกปิดร่วมกับช่องเก็บสัญญาณซึ่งสามรถแลกเปลี่ยนกุญแจรหัสที่มีความเร้นลับและปลอดภัยพอที่จะป้องกันอาชญากรรมในข้อ 2 และ 3 อย่างไรก็ดีการจัดหาช่องสัญญาณที่ปลอดภัยอย่างแท้จางนั้นแทบเป็นสิ่งที่เป็นไปไม่ได้ ระบบที่ยังคงใช้การจัดการกุญแจแบบนี้ในปัจจุบันนั้น มีความสามารถป้องกันความปลอดภัยได้เพียงระดับหนึ่งเท่านั้น ระบบ ATM ของธนาคารที่ใช้กันทั่วไปทุกวันนี้ก็เป็นตัวอย่างที่ดี เพราะมีผู้เชี่ยวชาญรวมทั้งคดีตัวอย่างที่เกิดขึ้นในต่างประเทศจำนวนมาก ซึ่งชี้ให้เห็นว่า มันเปราะบาง มากกต่อการถูกโจมตีด้วยการแอบดังฟังทางสายโทรศัพท์ที่ถูกใช้ในการสื่อสารระหว่างเครื่อง ATM และศูนย์คอมพิวเตอร์ของธนาคาร

การใช้รหัสลับระบบกุญแจสาธารณะสามารถแก้ปัญหาการบุกรุกในข้อ 2 ไปได้เพราะกุญแจที่ใช้ในการแลกเปลี่ยนกุญแจสาธารณะไม่ใช่กุญแจส่วนตัว จึงไม่สามารถใช้ในการถอดรหัสหรือการลงนาม

ระบบดิจิตอลได้ อย่างไรก็ตามรหัสลับระบบกุญแจสาธารณะเพียงอย่างเดียว โดยเฉพาะ RSA ไม่อาจป้องกันการปลอมแปลงในข้อ 3 ได้

ตัวอย่างเช่น ในระบบที่ใช้รหัส RSA ถ้าผู้บุกรุกมีความชำนาญสามารถเข้ามาในระบบได้ แล้วลักลอบเปลี่ยนกุญแจของ Bob ด้วยกุญแจที่ทำขึ้นเอง หากไม่มีระบบการป้องกันที่รัดกุมเพียงพอ ที่จะตรวจสอบว่ากุญแจสาธารณะที่อยู่ในฐานข้อมูลนั้นเป็นของจริงหรือไม่ ผู้บุกรุกก็สามารถแอบอ้างเป็น Bob ได้อย่างง่ายดาย ด้วยกุญแจที่ทำปลอมผู้บุกรุกสามารถที่จะอ่านเอกสารที่เป็นความลับของสมาชิกคนอื่นที่ต้องการส่งให้ Bob ได้ หรืออาจแอบอ้างการเป็น Bob โดยการใช้ลายเซ็นต์ปลอมในการส่งเอกสารต่าง ๆ เช่น ใบสั่งของ หรือ โอนจ่ายเงินจากบัญชี ของ Bob ได้

ปัจจุบันมาตรฐานที่นิยมใช้กันมากที่สุดเพื่อป้องกันการบุกรุกชนินี้คือจะต้องมีหน่วยงานเชื่อถือได้ ทำการออกใบรังรองให้หน่วยงานที่ว่านี้ก็คือ หน่วยงานหรือองค์กรที่ได้รับสิทธิการรับรอง(Certificate Authority : CA) ส่วนใบรับรองที่องค์กรนี้จะออกให้เรียกว่าใบรับรองดิจิตอล (Digital Certificate) การรับรองของ CA ไม่ใช้ให้เลยทันที แต่ผู้ใช้จะต้องทำการสมัครขอใช้บริการ ซึ่งทาง CA ก็จะทำ การตรวจสอบข้อมูลเพื่อมายืนยันตัวบุคคลที่ใช้บริการว่ามีตัวตนอยู่หรือไม่ หากผ่านเกณฑ์การตรวจสอบ CA ก็จะออกใบรับรองดิจิตอลให้ มาตรฐานของใบรับรองที่ยอมรับกันมากที่สุดอันหนึ่งคือ X.509 ของ CCITT

บทที่ 5

Elliptic Curve Cryptosystem

5.1 ECC คืออย่างไร

เนื่องจากระบบรักษาความปลอดภัยของข้อมูลนั้นถือเป็นหัวใจสำคัญของสมาร์ทการ์ด และเนื่องด้วยขนาดที่จำกัดของสมาร์ทการ์ดทำให้การจัดการพื้นที่ในสมาร์ทการ์ดให้มีประสิทธิภาพที่สุด จึงเป็นสิ่งสำคัญที่ต้องพิจารณาในระยะแรกนั้นการเข้ารหัสลับในสมาร์ทการ์ดนั้นใช้รหัสลับแบบกุญแจปกปิดแต่เพียงอย่างเดียว ซึ่งทำงานได้รวดเร็ว แต่ว่าความหลากหลายในการใช้งานมีน้อย จนเมื่อมีรหัสลับระบบกุญแจสาธารณะเกิดขึ้น เช่น RSA ก็ได้มีแนวความคิดที่จะนำรหัสระบบกุญแจสาธารณะมาใช้งานบนสมาร์ทการ์ด แต่จากบทที่แล้ว จากอัลกอริทึมของ RSA นั้น แสดงให้เห็นว่าการออกแบบวงจรเข้ารหัสนั้นต้องใช้การยกระดับโมดูลล ทำให้วงจรมีขนาดใหญ่มากจนไม่เหมาะที่จะนำมาสร้างบนแอพพลิเคชั่นขนาดเล็กและราคาถูก อย่างเช่น สมาร์ทการ์ด จนเมื่อปี ค.ศ. 1985 Victor Miller และ Neal ได้นำเอาการเข้ารหัสลับแบบ Elliptic Curve มาประยุกต์ใช้งานร่วมกับอัลกอริทึมของรหัสลับระบบกุญแจสาธารณะแบบอื่น ๆ เช่น ร่วมกับรหัสลับแบบ ElGamal เรียกว่า EC(Elliptic Curve) ElGamal โดยนำมาประยุกต์ใช้ในส่วนของกุญแจรหัส เนื่องจาก ECC นั้นมีขนาดกุญแจเล็กกว่ารหัสลับระบบกุญแจสาธารณะแบบอื่น ๆ

ตารางที่ 5.1 เปรียบเทียบขนาดกุญแจที่เล็กกว่ารหัสระหว่าง RSA กับ ECC

MIPS years	RSA/DSA Key size	ECC Key size	RSA/DSA Key size ratio
3×10^4	512	106	5:1
2×10^8	768	132	6:1
3×10^{11}	1,024	160	7:1
3×10^{20}	2,048	210	10:1

จากตารางที่ 5.1 แสดงการเปรียบเทียบเวลาที่ใช้ในการทำลายระบบ RSA DSA (Digital Signature Algorithm) และ ECC โดยเปลี่ยนแปลงตามขนาดของบิตที่ใช้งาน ซึ่งค่าในการคำนวณจะแสดงในรูป MIPS years ซึ่งก็คือเวลาที่ใช้ในการคำนวณใน 1 ปี บนขีดความสามารถในการประมวลผลของเครื่องคำนวณที่ 1 ล้านคำสั่งต่อ 1 วินาที (One Million Instructions per Second) ค่าที่ยอมรับกันเป็นมาตรฐานทั่วไปคือ 10^{12} MIPSyears ซึ่งจากรูปทำให้ RSA และ DSA ควรเริ่มใช้งานที 1,024 บิต และ ECC ควรเริ่มใช้งานที่ 160 บิต ซึ่งเห็นได้อย่างชัดเจนว่าขนาดของกุญแจรหัสของ ECC นั้นเล็กกว่าของ RSA และ DSA มาก และเมื่อเพิ่มระดับความปลอดภัยให้สูงขึ้นจะทำให้ขนาดความแตกต่างของกุญแจรหัสจะเพิ่มมากขึ้น เช่น เมื่อใช้งาน ECC ที่ 600 บิต ขณะที่ RSA และ DSA นั้นต้องใช้ถึง 21,000 บิต ทำให้เห็นได้ว่าการเลือกใช้ ECC บนสมาร์ทการ์ดนั้นทำให้ลดการสิ้นเปลืองหน่วยความจำในการเก็บกุญแจ

รหัสลงไปได้มาก และอัลกอริทึมของ ECC นั้นใช้คณิตศาสตร์ได้หลายแบบในการสร้าง ซึ่งสามารถสร้างได้หลายวิธีแล้วแต่คณิตศาสตร์ที่เลือกใช้ ในปัจจุบันมีงานวิจัยของ ECC ออกมาอย่างต่อเนื่องทำให้ปัจจุบันสามารถสร้าง ECC ให้วงจรให้มีขนาดเล็ก สามารถทำงานได้อย่างรวดเร็วทั้งในแบบซอฟแวร์และฮาร์ดแวร์

ตารางที่ 5.2 แสดงกำลังที่ใช้สำหรับทำลายระบบ ECC โดยใช้วิธีการของ Pollard rho method ซึ่งเปลี่ยนแปลงตามค่าของ n

Field size (in bits)	Size of n (in bits)	$(\lambda n/2)^{1/2}$	MIPS years
163	160	2^{8}	9.6×10^{11}
191	186	2^{93}	7.9×10^{15}
239	234	2^{117}	1.6×10^{23}
359	354	2^{117}	1.5×10^{41}
431	426	2^{213}	1.0×10^{52}

5.2 กาลัวฟิลด์ (Galois Field Arithmetic)

ในระบบการสื่อสารข้อมูล (Data Communication) นั้นข้อมูลต่าง ๆ แสดงออกมาเป็นสัญญาณดิจิตอล ซึ่งมีสัญลักษณ์เพียง "0" และ "1" โดยการกระทำกับสัญญาณเหล่านั้นต้องมีการกระทำโอเปอร์เรชั่นซึ่งประกอบด้วย การบวก การลบ การคูณ (Multiplication) และการหาส่วนกลับ (Inverse) ดังนั้น จึงมีคณิตศาสตร์เพื่อใช้สำหรับการทำโอเปอร์เรชั่นดังกล่าว และเรียกคณิตศาสตร์นั้นว่า Finite Field (Galois Field) การบวกและการคูณดังกล่าวเรียกว่าการบวกและการคูณแบบโมดูโล 2 โดยที่ผลลัพธ์ที่ได้จากทำโอเปอร์เรชั่นบวก ลบ คูณ และอินเวอร์ส ต้องถูกโมดูโลด้วย 2 จึงได้ผลลัพธ์ก็คือ เศษที่เหลือจากหารด้วย 2 นั่นเอง

ตารางที่ 5.3 การบวกและการคูณแบบคณิตศาสตร์

$0 \oplus 0$	$0 \otimes 0$
$0 \oplus 1$	$0 \otimes 1$
$1 \oplus 0$	$1 \otimes 0$
$1 \oplus 1$	$1 \otimes 1$

$\oplus$ XOR $\oplus$ AND

$GF(2^m)$ เป็นเซตซึ่งมีสมาชิกอยู่จำนวน 2^m ตัวที่ไม่ซ้ำกัน โดยที่ m เป็นจำนวนเต็ม และเนื่องจากมีเซตที่ไม่ซ้ำกันจำนวน 2^m นี้เองจึงทำให้สามารถนำ $GF(2^m)$ ไปประยุกต์ใช้ในระบบการสื่อสารแบบต่าง ๆ มากมาย เช่น การแก้ไขข้อมูลผิดพลาดแบบ BCH Coding (Bose – Chaudhuri-Hocquenghem codes) แบบ Reed Solomon Coding และอื่น ๆ ดังนั้นสมาชิกจำนวน 2^m ตัวของ $GF(2^m)$ สามารถหาได้ตามวิธีการต่อไปนี้ อันดับแรกเริ่มจากการกำหนดค่าของ Primitive Polynomial p(x) อันดับที่ m โดยที่ Primitive Polynomial ดังกล่าวก็คือพหุนามที่ไม่สามารถแยกตัวประกอบออกได้กำลังที่ m หรือเรียกว่า Irreducible Polynomial นั่นเองถ้ากำหนดให้ α เป็นรากของโพลิโนเมียล กล่าวคือ $P(\alpha) = 0$ ค่ายกกำลังของ a มีค่าได้ถึง $2^m - 2$ ค่าที่แตกต่างกัน โดยค่ายกกำลังของ α^{2m-1} จะเท่ากับ 1 ใหม่ ดังนี้ 0, 1 α, α^2, ..., α^{2m-2} จึงเป็นเซตของ $GF(2^m)$ และเรียก $GF(2^m)$ ว่า Field โดยเรียกสมาชิกของ Field ว่า Field Element โดยแต่ละอิลิเมนท์ที่เป็น 1, α, α^2,..., α^{m-1} ตัวอย่างสำหรับ m = 4, $p(x) = x^4 + x + 1$ เป็น Irreducible Polynomial ของ Field $GF(2^4)$ และให้ α เป็นรากของโพลิโนเมียล p $p(x) = x^4 + x + 1$ ดังนั้น $\alpha^4 = \alpha + 1$, α สามารถยกกำลังที่ให้ค่าแตกต่างกัน $2^4-2 = 14$ ค่า ดังนั้นการกระจายเทอมกำลังต่าง ๆ ของ a ทำได้ดังตารางที่ 5.4

ตารางที่ 5.4 กาลัวส์ฟิลด์ 2^4 อีลีเมนท์ ($GF(2^4)$) ซึ่ง $p(x) = x^4+x+1$

ฟิลด์อีลีเมนท์ ($GF(2^4)$)	เลขฐานสอง
0	0000
1	0001
α	0010
$\alpha+1$	0011
α^2	0100
α^2+1	0101
$\alpha^2+\alpha$	0110
$\alpha^2+\alpha+1$	0111
α^3	1000
$\alpha^3+\alpha$	1001
$\alpha^3+\alpha$	1010
$\alpha^3+\alpha+1$	1011
$\alpha^3+\alpha^2$	1100
$\alpha^3+\alpha^2+1$	1101
$\alpha^3+\alpha^2+\alpha$	1110
$\alpha^3+\alpha^2+\alpha+1$	1111

อีลีเม้นท์ α นี้ จะเรียกกว่าไพรมีทีฟอีลีเม้นท์ (Primitive Element) ของ $GF(2^m)$ ได้โดยอีลีเม้นท์ใด ๆ ของ $GF(2^m)$ ที่ค่ายกกำลังของอีลีเม้นท์นั้นไม่เท่ากับศูนย์บน $GF(2^m)$ แล้วจะเรียกอีลีเม้นท์ดังกล่าวว่าไพรมีทีฟ (Primitive) ตัวอย่าง ค่ายกกำลังของ α ที่ α^4 บน $GF(2^m)$ ในตารางที่ 5.3 คือ

ตารางที่ 5.5 ตัวอย่างของกำลังของ α^4 ของ $(GF(2^4)$

$(\alpha^4)^0=1$	$(\alpha^4)^1=\alpha$	$(\alpha^4)^1=\alpha^8$
$(\alpha^4)^3=\alpha^{12}$	$(\alpha^4)^4=\alpha^{16}$	$(\alpha^4)^5=\alpha^{20}=\alpha^5$
$(\alpha^4)^6=\alpha^{24}=\alpha^9$	$(\alpha^4)^7=\alpha^{28}=\alpha^{13}$	$(\alpha^4)^8=\alpha^{32}=\alpha^2$
$(\alpha^4)^9=\alpha^{36}=\alpha^6$	$(\alpha^4)^{10}=\alpha^{40}=\alpha^{10}$	$(\alpha^4)^{11}=\alpha^{44}=\alpha^{14}$
$(\alpha^4)^{12}=\alpha^{48}=\alpha^3$	$(\alpha^4)^{13}=\alpha^{52}=\alpha^7$	$(\alpha^4)^{14}=\alpha^{56}=\alpha^{11}$

พบว่าทั้ง 15 อิลิเม้นท์ มีค่าไม่เท่ากับ 0 ของ $GF(2^4)$ ดังนั้น a^4 จึงเป็นไพรมีทีฟอิลีเม้นท์ของ $GF(2^4)$ ส่วนอีลีเม้นท์ a^4 ไม่เป็นไพรมีทีฟอิลีเม้นท์เพราะค่าของ (a^4) มีค่าเท่ากับ 0 ดังนั้น โพลิโนเมียล p(x) ที่มีอันดับเท่ากับ m ที่สร้างเซตซึ่งมีสมาชิกเท่ากับ 2^m ตัวจะถูกเรียกว่าไพรมีทีฟโพลิโนเมียล จากวิธีการหาค่าไพรมีทีฟโพลิโนเมียลจึงพิสูจน์ได้ว่าแต่ละจำนวนเต็มบวก m จะมีอย่างน้อยที่สุด 1 ไพรมีทีฟโพลิโนเมียล ไพรมีทีฟโพลิโนเมียลต่าง ๆ พอสรุปมาได้ดังตารางที่ 5.6

ตารางที่ 5.6 ไพรมีทีฟโพลิโนเมียลอันดับต่างๆ

อันดับที่ m	ไพรมีทีฟโพลิโนเมียล
3	x^3+x+1
4	x^4+x+1
5	x^5+x^2+1
6	x^6+x+1
7	x^7+x^3+1
8	$x^8+x^4+x^3+x^2+1$
9	x^9+x^4+1
10	$x^{10}+x^3+1$
11	$x^{11}+x^2+1$
12	$x^{12}+x^6+x^4+x+1$
13	$x^{13}+x^4+x^3+x+1$
14	$x^{14}+x^{10}+x^6+x+1$

ตารางที่ 5.6 ไพรมีทีฟโพลิโนเมียลอันดับต่างๆ(ต่อ)

15	$x^{15}+x+1$
16	$x^{16}+x^{12}+x^{3}+x+1$
17	$x^{17}+x^{3}+1$
18	$x^{18}+x^{7}+1$
19	$x^{19}+x^{5}+x^{2}+x+1$
20	$x^{20}+x^{3}+1$
21	$x^{21}+x^{2}+1$
22	$x^{22}+x+1$
23	$x^{23}+x^{5}+1$
24	$x^{24}+x^{7}+x^{2}+x+1$

5.3 โอเปอร์เรชั่นของกาลัวฟิลด์

โอเปอร์เรชั่นทางคณิตศาสตร์ของกาลัวฟิลด์ $GF(2^m)$ สามารถแสดงในรูปของโพลิโนเมียล ซึ่งมี p(x) เป็นพหุนามลดทอนไม่ได้ (Irreducible Polynomial)

5.3.1 โอเปอร์เรชั่นการบวก

ถ้า $A(x) = (a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\ldots+a_1x+a_0) = \sum^{m=1} a_kx^k$ และ

$B(x) = (b_{m-1}x^{m-1}+b_{m-2}+x^{m-2}+\ldots+b_1x+a_0) = \sum^{m=1} a_kx^k$ โดยที่ $a,b \in GF(2^m)$

สามารถแสดงโอเปอร์เรชั่นการบวกได้ดังนี้

$$A(x)+B(x) = C(x) = \sum_{i=0}^{m-1} cix^i \qquad (5.1)$$

โดยที่ $c_i = (a_i+b_i) \bmod 2$

5.3.2 โอเปอร์เรชั่นการคูณ

ถ้า $A(x) = (a_{m-1}x^{m-1}+a_{m-2}+x^{m-2}+\ldots+ a_2x^2+a_1x+a_0) = \sum^{m=1} a_kx^k$ และ

$B(x) = (b_{m-1}x^{m-1}+b_{m-2}+x^{m-2}+\ldots+ b_2x^2+b_1x+a_0) = \sum^{m=1} a_kx^k$ โดยที่ $a,b \in GF(2^m)$

แสดงโอเปอร์เรชั่นการบวกได้ดังนี้

$$A(x).B(x) = R(x) \bmod p(x) = \sum_{i=0}^{m-1} cix^i \qquad (5.2)$$

5.3.3 โอเปอร์เรชั่นอินเวอร์ส

ถ้า a,c $\in GF(2^m)$ ซึ่งไม่เท่ากับศูนย์อินเวอร์สของ a สามารถแสดงได้ดังนี้

$$A(x)^{-1} = \frac{1}{C(x)} \qquad (5.3)$$

โดยที่ $A(x) = (a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1x + a_0) = \sum_{i=0}^{m-1} a_k x^k$ และ

$$C(x) = (c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \ldots + c_1x + c_0) = \sum_{i=0}^{m-1} c_k x^k$$

ตัวอย่าง 5.1 แสดงโอเปอร์เรชั่นของกาลัวฟิลด์

ถ้า $p(x) = x^4 + x + 1$ เป็นพหุนามลดทอนไม่ได้บน $GF(2^4)$ สามารถแสดงอีลีเม้นท์ได้ 16 ตัวดังนี้

0	(0000)	1	(0001)	x	(0010)	x+1	(0011)
x^2	(0100)	x^2+1	(0101)	x^2+x	(0110)	x^2+x+1	(0111)
x^3	(1000)	x^3+1	(1001)	x^3+x	(1010)	x^3+x+1	(1011)
x^3+x^2	(1100)	x^3+x^2+1	(1101)	x^3+x^2+1	(1110)	x^3+x^2+x+1	(1111)

กำหนดให้ $A(x) = x^3 + x^2 + 1$ และ $B(x) = x^3 + 1$ จากสมการที่ (5.1) สามารถแสดงโอเปอร์เรชั่นการบวกได้ดังนี้

$$A(x)+B(x) = C(x)$$
$$(x^3 + x^2 + 1) + (x^3 + 1) = x^2$$

ดังนั้นผลบวกของ A(x) และ B(x) มีค่าเท่ากับ x^2

กำหนดให้ $A(x) = x^3 + x^2 + 1$ และ $B(x) = x^3 + 1$ จากสมการที่ (5.2) สามารถแสดงโอเปอร์เรชั่นการคูณได้ดังนี้

$$A(x)+B(x) = R(x) \bmod p(x)$$
$$(x^3 + x^2 + 1).(x^3 + 1) = x^6 + x^5 + x^2 + 1$$
$$= (x^6 + x^5 + x^2 + 1) \bmod (x^4 + x + 1)$$
$$= (x^3 + x^2 + x + 1)$$

ดังนั้นผลบวกของ A(x) และ B(x) มีค่าเท่ากับ $x^3 + x^2 + x + 1$

กำหนดให้ $A(x)\ x^3 + x^2 + 1$ จากสมการที่ (5.3) สามารถแสดงโอเปอร์เรชั่นอินเวอร์สได้ดังนี้

$$A(x)^{-1} = \frac{1}{C(x)}$$

$$= \frac{1}{x^3 + x^2 + 1}$$

$$= x^2$$

ดังนั้นผลบวกของ A(x) และ B(x) มีค่าเท่ากับ x^2

5.4 Elliptic Curve Over F_2m

5.4.1 สมการ Elliptic Curve Over F_2m

elliptic curve E over F_2m สามารถแสดงความสัมพันธ์ในรูปของสมการ Non – Supersigular ดังต่อไปนี้

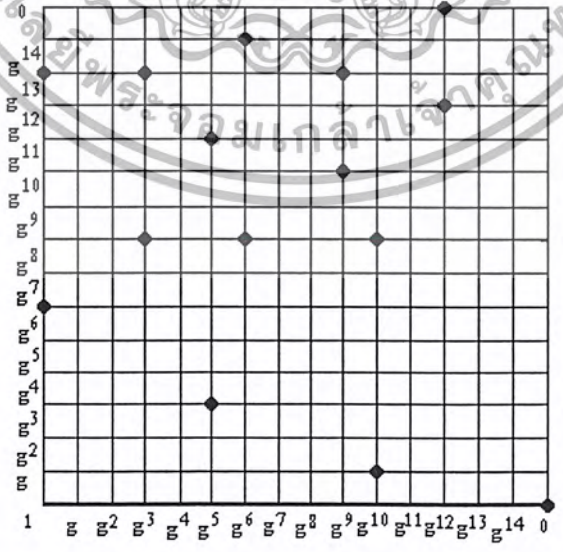
$$y^2+x = x^3 + ax^2 + b \bmod f(x) \qquad (5.4)$$

โดยที่ $a,b \in F_2m$ และ $b \neq 0$ เซ็ตของ $E(F_2m)$ จะประกอบไปด้วยจุด (x,y) โดยที่ $x \in F_2m$, $y \in F_2m$ ที่ทำให้สมการ (1) เป็นจริง และอีกจุดหนึ่งคือจุด 0 เรียกกว่า Point at infinity

ตัวอย่างที่ 5.2 พิจารณา F_24 ซึ่งมี $f(x) = x^4 + x + 1$ จากสมการที่ (1) ให้ $a = g^4$ และ $b = 1$ เราจะได้จุดทั้งหมดของ $E(F_24)$ และจุด 0 ดังนี้

(0,1)	$(1,g^6)$	$(1,g^{13})$	(g^3,g^8)	(g^3,g^{13})
(g^5,g^3)	(g^5,g^{11})	(g^6,g^8)	(g^6,g^{14})	(g^9,g^{10})
(g^9,g^{13})	(g^{10},g)	(g^{10},g^8)	$(g^{12},0)$	(g^{12},g^{12})

ซึ่งแสดงค่าเหล่านี้ได้ด้วยกราฟดังรูปที่ 5.1



รูปที่ 5.1 ตำแหน่งของอิลิเมนที่เกิดจากสมการ $y^2 + xy = x^3 + g^4x^2 + 1 \mod (x^4 + x + 1)$

5.4.2 กฎการบวกของ Elliptic curve

1. $P + O = O + P =$ ทุกค่าของ $P \in E(F_2m)$
2. ถ้า $P = (x,y) \in E(F_2m)$ ดังนั้น $(x,y) + (x,x+y) = O$ เพราะว่าจุด $(x,x+y)$ นี้แทนได้ด้วย P ซึ่งเป็นค่าติดลบของ P
3. ถ้า $P = (x,y) \in E(F_2m)$ ดังนั้น $(x,y) + (x_2,y_2) \in E(F_2m)$ โดยที่ $P \neq \pm Q$ แล้วเราจะได้ $P + Q = (x_3,y_3)$ ซึ่งเราเรียกว่า Point addition โดยที่

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$$

$$y_3 = \lambda.(x_1+x_3) + x_3 + y_1$$

$$\lambda = (y_1+y_2)/(x_1+x_2)$$

4. ให้ $P = (x_1,y_1) \in E(F_2m)$ โดยที่ $P \neq -P$ เราจะได้ $2P = (x_2,y_3)$ ซึ่งเราเรียกว่า Point doubling

$$x_3 = (x_1)^2 + b/(x_1)^2$$

$$y_2 = x_1^2 + [(x_1 + y_1 / (x_1)].x_3 + x_3$$

5.5 การเลือกใช้โพลิโนเมียลลดรูป

ถ้าโพลิโนเมียลอยู่ในรูป $x^m + x^k + 1$ โดยที่ $1 \leq k \leq m-1$ แล้ว จะเรียกโพลิโนเมียลนี้ว่า Trinomial over F2 และถ้าอยู่ในรูป $x^m + x^{k3} + x^{k2} + x^{k1} + 1$ โดยที่ $1 \leq k1 \leq k2 \leq k3 \leq m-1$ จะเรียกโพลิโนเมียลนี้ว่า Pentanomial over F_2 ตามมาตรฐานของ ANSI X9.62 จะมีกฎในการเลือกโพลิโนเมียลลดรูปดังนี้

1. ถ้า Irreducible trinomial ดีกรี m ของ F2 มีอยู่จริงแล้ว โพลิโนเมียลลดรูปต้องเป็น Irreducible trinomial ซึ่งอยู่ในรูป $x^m + x^k + 1$ ให้เลือกค่า k ที่น้อยที่สุด
2. ถ้า Irreducible trinomial ดีกรี m ของ F_2 ไม่มีอยู่ โพลิโนเมียลลดรูปต้องเป็น Irreducible pentanomial ซึ่งอยู่ในรูป $x^m + x^{k3} + x^{k2} + x^{k1} + 1$ ให้เลือกค่า k ดังนี้
   - พิจารณาเฉพาะ k3 ให้มีค่าน้อยที่สุดก่อน แล้วค่อยพิจารณา k2, k1
   - พิจารณาเฉพาะ k3, k2 ให้มีค่าน้อยที่สุดก่อน แล้วค่อยพิจารณา k2, k1
   - พิจารณา k3, k2, k1 ให้มีค่าน้อยที่สุด

การเลือกโพลิโนเมียลลดรูปของ F_2m ในรูปของ Polynomial Basic ตามมาตรฐานของ SEC แสดงในตารางที่ 5.7

ตารางที่ 5.7 แสดงโพลิโนเมียลลดรูปดีกรีต่าง ๆ ตามมาตรฐานของ SEC

Field	Reduction Polynomia (s)
F_2113	$x^{113} + x^9 + 1$
F_2131	$x^{113} + x^8 + x^3 + x^2 + 1$
F_2163	$x^m + x^7 + x^6 + x^3 + 1$
F_2193	$X^{193} + x^{15} + 1$
F_2233	$X^{233} + x^{74} + 1$
F_2239	$X^{239} + x^{36} + 1$ or $x^{239} + x^{158} + 1$
F_2283	$X^{283} + x^{12} + x^7 + x^5 + 1$
F_2409	$X^{409} + x^{87} + 1$
F_2571	$X^{571} + x^{10} + x^5 + x^2 + 1$

5.6 อัลกอริทึม Elliptic Curve ElGamal

อัลกอริทึมเบื้องต้นของ Elliptic Curve ที่รู้จักกันดีอัลกอริทึมหนึ่งคือ อัลกอริทึม Elliptic Curve ElGamal หรือเรียกสั้น ๆ ว่า EC ElGamal โดยในโครงงานนี้จะใช้อัลกอริทึม EC ElGamal เป็นอัลกอริทึมที่แสดงการเข้ารหัสและถอดรหัสเนื่องจากหลักการที่ง่าย ทำให้สามารถมองเห็นถึงการเอาหลังการของ Elliptic Curve ซึ่งจะเป็นพื้นฐานในการศึกษาอัลกอริทึมมาตรฐานอื่น ๆ ในหัวข้อ 5.6 ซึ่งจะเห็นว่าจะมีการนำเอา Elliptic Curve ไปใช้ในการสร้างกุญแจแบบต่าง ๆ

หลักการทำงานของอัลกอริทึม EC ElGamal แสดงดังตารางที่ 5.8 โดยเบื้องต้นนั้นทุกคนที่ใช้อัลกอริทึม EC ElGamal ทุกคนต้องทำการสร้างกุญแจส่วนตัว และกุญแจสาธารณะ จากนั้นทำการนำเอากุญแจสาธารณะไปขอใบรับรองจากองค์กรที่ทำหน้าที่รับรอง ซึ่งจะใช้กุญแจสาธารณะของคนนั้นเข้ารหัสดังแสดงในอัลกอริทึม เมื่อ B ต้องการส่งข้อมูลลับมาให้ A อย่างแรกที่จะต้องทำคือขอกุญแจสาธารณะคือ A_T และพารามิเตอร์ต่างๆ แล้วนำมาเข้ารหัสตามตารางการทำงานดังกล่าว จากนั้นเมื่อส่งข้อความลับ (C_1,C_2) ที่ B ได้ทำการเข้ารหัสไปใช้ A จากนั้น A จะทำการถอดรหัสออกมาได้โดยใช้กุญแจส่วนตัวของ a โดยทำการคูณแบบ Scalar คือ $a.C_1$ คือ Point C_1 บวกกัน a ครั้ง

ตารางที่ 5.8 แสดงอัลกอริทึมการเข้ารหัสแบบ Elliptic Curve EIGamal

A ทำการสร้างกุญแจเข้ารหัส	
ขั้นตอนที่ 1	เลือก Field (m)
ขั้นตอนที่ 2	กำหนด E ซึ่งก็คือ สมการ $y^2+xy = x^3 + ax^2 + b \bmod f(x)$ โดยเลือกค่า $a,b \in 2^m$
ขั้นตอนที่ 3	สุ่มคู่ลำดับ P ที่ทำให้ E เป็นจริง

ตารางที่ 5.8 แสดงอัลกอริทึมการเข้ารหัสแบบ Elliptic Curve EIGamal(ต่อ)

ขั้นตอนที่ 4	เลือกเลขจำนวนเต็ม a ซึ่งอยู่ในช่วง [2,m-1]
ขั้นตอนที่ 5	คำนวณ $A_T = aP$
ขั้นตอนที่ 6	โดยที่ a เป็นกุญแจส่วนตัว และ A_T,E,m,P เป็นกุญแจสาธารณะของ A
B ทำการเข้ารหัส	
ขั้นตอนที่ 1	ขอกุญแจสาธารณะจาก A_T คือ A_T,E,m,P
ขั้นตอนที่ 2	สุ่มเลขจำนวนเต็ม k ซึ่งอยู่ในช่วง [2,m-1] ซึ่งต้องสุ่มใหม่ทุกครั้ง
ขั้นตอนที่ 3	คำนวณ $C_1 = kP$ และ $KA_T = k(aP) = (xy)$
ขั้นตอนที่ 4	คำนวณ $C_2 = P_m + kA_T$ โดยที่เราสมมติว่า P_m เป็นเอกสารที่นำมาเข้ารหัส $P_m \in E$
ขั้นตอนที่ 5	ส่ง C_1 ,C_2 ไปให้ A
A ทำการถอดรหัส	
ขั้นตอนที่ 1	คำนวณ $a C_1 = a(kP) = (x,y)$
ขั้นตอนที่ 2	A ถอดรหัส โดยการคำนวณ $P_m = C_2 , -aC_1$

ตัวอย่างที่ 5.3 แสดงการเข้ารหัสแบบ EC Elgamal โดยที่ $E : y^2+xy = x^3 = (1100)x^2 + (0001)$ mod (10011) , a = (0111), P = (1100,0101), m = 4 bit , Pm = (0110,1000)

- A ทำการสร้างกุญแจสาธารณะ

$$A = a.p\ (0111).(1100,0101) = (1010,0111)$$

- B ทำการสุ่มค่า k ซึ่งต้องสุ่มทุกครั้ง ซึ่งการสุ่มทุกครั้งทำให้รหัสที่ใช้ในการเข้ารหัสจะเปลี่ยนไปทุกครั้งแม่ว่าจะใช้กุญแจสาธารณะของ A ค่าเดิม ทำให้รหัสมีความปลอดภัยมากขึ้น ซึ่งเป็นข้อดีของ ECC โดยสมมติว่า k = (1011)

5.7 อัลกอริทึมมาตรฐานของ ECC

5.7.1 อัลกอริทึม Diffie – Hellman

อัลกอริทึมของ Diffie-Hellman นั้นเป็นอัลกอริทึมพื้นฐานของรหัสลับระบบกุญแจสาธารณะที่มีจุดประสงค์เพื่อใช้ทำกุญแจปกปิดร่วมกัน โดยที่ A(Alice) และ B(Bob) นั้น ต้องยอมรับการใช้พารามิเตอร์ที่ใช้ในการเข้ารหัสร่วมกัน เช่น ขนาดของกุญแจรหัส ชนิดของคณิตศาสตรีที่ใช้ คู่ลำดับที่ใช้ เป็นต้น ซึ่งหลักการทำงานแสดงตารางที่ 5.9

ตารางที่ 5.9 อัลกอริทึมของ Diffie Hellman

ขั้นตอนที่ 1	ทั้ง A และ B ต้องทำการสุ่มกุญแจส่วนตัว ซึ่งจะได้ 'k_A' เป็นกุญแจส่วนตัวของ A และ 'k_B' เป็นกุญแจส่วนตัวของ B
ขั้นตอนที่ 2	A ต้องทำการคำนวณ k_AG แล้วส่งไปให้ B ในทำนองเดียวกัน B ต้องทำการคำนวณ k_BG แล้วส่งไปให้ A ในทำนองเดียวกัน
ขั้นตอนที่ 3	ทั้ง A และ B ต่างคำนวณกุยแจปกปิดที่ใช้ร่วมกันคือ $Q = k_Ak_BG = k_Bk_AG$

5.7.2 อัลกอริทึม ECIES

อัลกอริทึม ECIES (Ellptic Curve Integrated Encryption Scheme) นั้นเป็นการเข้ารหัสแบบ Elgamal แบบหนึ่งที่มีประสิทธิภาพมาก และใช้กันเป็นมาตรฐานในการเข้ารหัสและถอดรหัสลับเอกสาร โดยถ้า B ต้องการส่งเอกสาร (m_1,m_2) B จะเข้ารหัสข้อความ (m_1,m_2) ด้วยกุญแจสาธารณะของ A แล้วส่งไปให้ A เมื่อ A ได้รับเอกสารลับ A จะสามารถถอดรหัสได้ด้วยกุญแจส่วนตัวของตัวเอง การทำงานของอัลกอริทึม ECIES เป็นตารางที่ 5.10

ตารางที่ 5.10 อัลกอริทึม ECIES

A ทำการสร้างกุญแจเข้ารหัส	
ขั้นตอนที่ 1	เลือก Field (n)
ขั้นตอนที่ 2	กำหนด E ซึ่งก็คือ สมการ $y^2+xy = x^3 + ax^2 + b \bmod f(x)$ โดยเลือกค่า $a,b \in 2^m$
ขั้นตอนที่ 3	สุ่มคู่ลำดับ G ที่ทำให้ E เป็นจริง
ขั้นตอนที่ 4	เลือกเลขจำนวนเต็ม d ซึ่งอยู่ในช่วง [1,n-1]
ขั้นตอนที่ 5	คำนวณ Q = dG
ขั้นตอนที่ 6	โดยที่ d เป็นกุญแจส่วนตัว และ Q,E,n,G เป็นกุญแจสาธารณะของ A
B ทำการเข้ารหัส	
ขั้นตอนที่ 1	ขอกุญแจสาธารณะจาก A คือ Q,E,n,G
ขั้นตอนที่ 2	สุ่มเลขจำนวนเต็ม k ซึ่งอยู่ในช่วง [1,n-1] ซึ่งต้องสุ่มใหม่ทุกครั้ง
ขั้นตอนที่ 3	คำนวณ P = kG และ kQ = k(aG) = (x,y) ถ้า $x = 0 \pmod n$ หรือ $y = 0 \pmod n$ ให้เริ่มขั้นตอนที่ 2 ใหม่
ขั้นตอนที่ 4	คำนวณ $C_1 = m_1.x \bmod n$ และ $c_2 = m_2.y \bmod n$ ให้เริ่มขั้นตอนที่ 2 ใหม่
ขั้นตอนที่ 5	ส่ง (C_1, C_2, P) ไปให้ A
A ทำการถอดรหัส	

ตารางที่ 5.10 อัลกอริทึม ECIES(ต่อ)

ขั้นตอนที่ 1	คำนวณ a P = a(kG) = (x,y)
ขั้นตอนที่ 2	A ถอดรหัสโดยการคำนวณ $m_1 = c_1.x^{-1} \bmod n$ และ $m2 = c_2.y^{-1} \bmod n$

5.7.3 อัลกอริทึม ESDSA

อัลกอริทึมของ ECDSA นั้น ใช้ในการลงลายเซ็นต์ดิจิตอล และการตรวจสอบความถูกต้องของข้อมูล ซึ่งถ้า A ต้องการส่งเอกสารไปใช้ B เพื่อเป็นการยืนยันว่าเป็น A จริง A ต้องส่งลายเซ็นต์ดิจิตอลไปพร้อมกับข้อมูลด้วย โดยการนำเอกสารไปเข้าฟังก์ชันแฮส ซึ่งจะได้ไดเจสของเอกสาร จากนั้น A จะเข้ารหัสไดเจสของข่าวสารด้วยกุญแจส่วนตัว ซึ่งจะได้ลายเซ็นต์ดิจิตอล จากนั้นจึงส่งลายเซ็นต์ดิจิตอลไปพร้อมกับเอกสารปกติส่งไปให้ B ซึ่ง B จะทำการตรวจสอบลายเซ็นตีและความถูกต้องของข้อมูลด้วยกุญแจสาธารณะของ A การทำงานของอัลกอริทึมของ ECDSA แสดงตารางที่ 5.11

ตารางที่ 5.11 อัลกอริทึมของ ECDSA

A ทำการสร้างกุญแจเข้ารหัส	
ขั้นตอนที่ 1	เลือก Field (n)
ขั้นตอนที่ 2	กำหนด E ซึ่งก็คือ สมการ $y^2+xy = x^3 + ax^2 + b \bmod f(x)$ โดยเลือกค่า $a,b \in 2^m$
ขั้นตอนที่ 3	สุ่มคู่ลำดับ G ที่ทำให้ E เป็นจริง
ขั้นตอนที่ 4	เลือกเลขจำนวนเต็ม d ซึ่งอยู่ในช่วง [1,n-1]
ขั้นตอนที่ 5	คำนวณ Q = dG
ขั้นตอนที่ 6	โดยที่ d เป็นกุญแจส่วนตัว และ Q,E,n,G เป็นกุญแจสาธารณะของ A
A ทำการสร้างลายเซ็นดิจิตอลจากเอกสาร m	
ขั้นตอนที่ 1	สุ่มเลขจำนวนเต็ม k ซึ่งอยู่ในช่วง [1,n-1] ซึ่งต้องสุ่มใหม่ทุกครั้ง
ขั้นตอนที่ 2	คำนวณ $k(G) = (x_1, y_1)$
ขั้นตอนที่ 3	คำนวณ $r = x_1 \bmod n$ (โดยที่ x_1 เป็นเลขจำนวนเต็ม) ถ้า r = 0 ให้เริ่มขั้นตอนที่ 1 ใหม่
ขั้นตอนที่ 4	คำนวณ $k^{-1} \bmod n$
ขั้นตอนที่ 5	คำนวณ $s = k^{-1} h(m) + dr\} \bmod n$ โดยที่ h เป็นฟังก์ชันแฮส ถ้า s = 0 ให้เริ่มขั้นตอนที่ 1 ใหม่
ขั้นตอนที่ 6	ซึ่งจะได้ลายเซ็นต์ดิจิตอลของเอกสาร m คือ (r,s)
B ตรวจสอบความถูกต้องของข้อมูล	
ขั้นตอนที่ 1	B ขอกุญแจสาธารณะของ A คือ Q,E,n,G
ขั้นตอนที่ 2	ตรวจสอบว่า r และ s อยู่ในช่วง $[1,n^{-1}]$

ตารางที่ 5.11 อัลกอริทึมของ ECDSA(ต่อ)

ขั้นตอนที่ 3	คำนวณ w = s-1 mod n และ h(m)
ขั้นตอนที่ 4	คำนวณ u_1 = h(m) w mod n และ u_2 = rw mod n
ขั้นตอนที่ 5	คำนวณ $u_1G + u_2Q = (x_0, y_0)$ และ v = x_2 mod n
ขั้นตอนที่ 6	B จะยอมรับเอกสารนี้ถ้า v = r

ตารางที่ 5.12 มาตรฐานที่สอดคล้องกันของ ECC

Standard	Schemes included
ANSI X9.62	ECDSA
ANSI X9.63	ECDSA, ECHD, ECMQV
FIPS 186 – 2	ECDSA
IEEE P 1363	ECDSA, ECHD, ECMQV
IEEE P 1363A	ECDSA
IPSec	ECDSA, ECHD, ECMQV
ISO 1488 – 3	ECDSA
ISO 15946	ECDSA, ECHD, ECMQV

5.8 การออกแบบระบบ ECC และระบบต่าง ๆ

5.8.1 การออกแบบวงจรคำนวณทางคณิตศาสตร์จำกัด (Finite Field Arithmetic)

ในระบบของ ECC ที่ใช้คณิตศาสตร์สนามจัดในรูปของ Binary Field ในการแทนความสัมพันธ์ทางคณิตศาสตร์ของสมาชิกใน Elliptic Curve จะมีการทำโอเปอร์เรชั่นทางคณิตศาสตร์อยู่ 3 อย่างที่สำคัญ คือ การคูณ การหากำลังสอง และการหาส่วนกลับ โดยการออกแบบโอเปอร์เรเตอร์ ของวงจรคูณ วงจรกำลังสอง และวงจรหาส่วนกลับนี้จะมีผลต่อขนาดและความเร็วในการทำงานของระบบเป็นอย่างมาก ซึ่งสามารถออกแบบได้ดังต่อไปนี้

5.8.1.1 การออกแบบวงจรคูณ

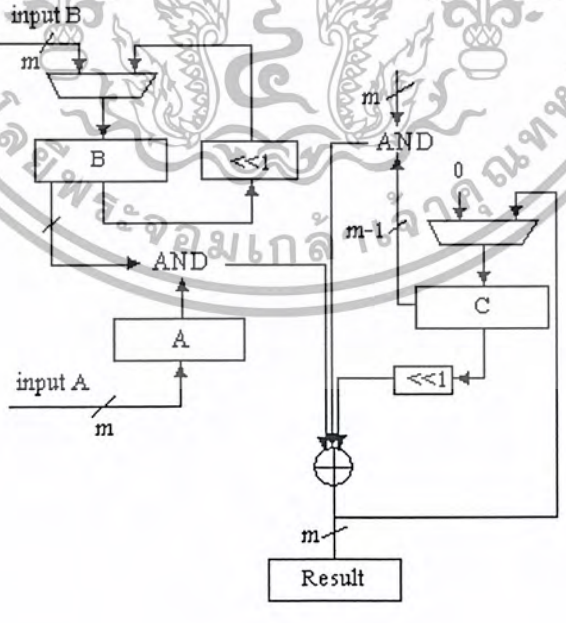
การออกแบบวงจรคูณของ Elliptic Curve นั้นไม่เหมือนกับการออกแบบวงจรคูณทางคณิตศาสตร์ หรือ การออกแบบวงจรดิจิตอลทั่ว ๆไป เนื่องจากการทำโอเปอร์เรชั่นของ Ellptic Curve นั้น ทำงานอยู่บนคณิตศาสตร์สนามจำกัด ซึ่งเมื่อค่าของผลลัพธ์ที่ได้จากการคูณเกินขอบเขตฟิลด์ไป ผลลัพธ์นั้นจะถูกโมดูโล ด้วยค่าขอบเขตของฟิลด์เพื่อให้ค่าที่เกินขอบเขตของฟิลด์นั้น กลับมาอยู่ในฟิลด์ เช่น การคูณด้วยคณิตศาสตร์ธรรมดา เราจะได้ 3 x 6 = 18 แต่ถ้าเป็นคณิตศาสตร์สนามจำกัดแล้วจะต้องมี

ขอบเขตของสนามจำกัด ซึ่งค่าที่ได้จะกลายเป็น 3 x 6 mod 11 = 7 ซึ่งก็คือค่าของเศษที่เกิดจากผลคูณหารด้วยขอบเขตของสนาม การออกแบบวงจรคูณในโครงการนี้ได้ใช้อัลกอริทึม GF(2^m) Multiplication ซึ่งมีการทำงานตามตารางที่ 5.13

ตารางที่ 5.13 อัลกอริทึม GF(2^m) Multiplication

Input : Binary Polynomial A (x) and B (x) $\in$ GF(2^m)	
Output : C(x) = A (x).B (x) mod P(x)	
ขั้นตอนที่ 1	C(x) = 0
ขั้นตอนที่ 2	For I = m-1 to 0 Do
ขั้นตอนที่ 3	- C(x) = C<<1+A(x).b_i
ขั้นตอนที่ 4	- C(x) = C+C_m.P(x)
ขั้นตอนที่ 5	End For
ขั้นตอนที่ 6	Return C (x)

จากตารางที่ 5.13 นั้น m จะเป็นขนาดของบิตที่ใช้ ส่วน P(x) เป็น Polynomial Reduction ซึ่งโครงสร้างทางฮาร์ดแวร์ ของอัลกอริทึม GF(2^m) Multiplication จะแสดงได้ดังรูปที่ 5.2 โดยจะใช้ Clock ในการประมวลผลเท่ากับ m + 1 ลูก



รูปที่ 5.2 แสดงโครงสร้างของวงจรคูณแบบ GF(2^m) Multiplication

5.8.1.2 การออกแบบวงจรกำลังสอง

จากอัลกอริทึม ECC over F_2m ในรูปของ Polynomial basis การคูณกันของสองอิลิเมนต์A,B $\in$ F_2m สามารถแสดงได้ในรูปของสมการ

$$C(x) = A(x).B(x) = \sum_{i=0}^{2m-2} a_i b_{k-1} \quad \text{ซึ่งแทนที่ด้วย } c_k = \sum_{i=0}^{k} a_i b_{k-1} \text{ โดยที่ } 0 \leq k \leq 2m-2 \quad (5.5)$$

จากสมการ (5.5) นั้น ที่ $a_i = 0$ และ $b_i = 0$ สำหรับ i□m ทำให้แต่ละระดับของการการคูณในนั้นอยู่ในรูป AND โอเปอร์เรชั่นของสมการบูลีน จะเห็นได้ว่า ค่าดีกรีที่มากที่สุดที่เกิดจากผลคูณของโพลิโนเมียล C(x) = A(x).B(x) เท่ากับ 2m-2 อย่างไรก็ตาม เราสามารถลดบิตสตริงให้เท่า m ได้โดยการโมดูโลด้วย โพลิโนเมียลลดรูป f(x) ซึ่งอยู่ในรูป

$$x^m \equiv \sum_{i=0}^{2m-1} f_i x^i \bmod f(x) \quad (5.6)$$

สำหรับ f(x) ที่อยู่ในรูปของ Irreducible trinomial คือ $x^m + x^b + 1$ กระบวนการลดรูปสามารถที่จะทำให้มีประสิทธิภาพโดยใช้สัญลักษณ์

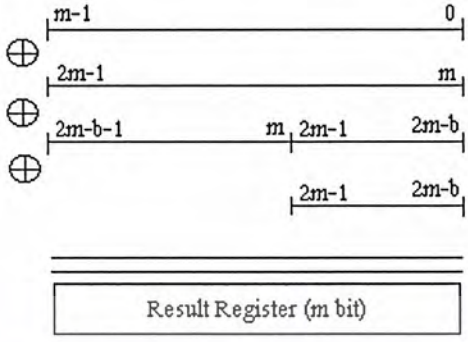
$$x^m \equiv x^b + \bmod f(x)$$
$$x^{m+1} \equiv x^{b+1} + \bmod f(x)$$
$$\vdots$$
$$x^{2m} \equiv x^{b+m} + x^m + \bmod f(x)$$

ซึ่งทำให้

$$\begin{aligned} C(x) &= \sum_{i=o}^{2m-2} c_i x^i \\ &\equiv \sum_{i=o}^{m-1} c_i x^i + \sum_{i=o}^{2m-2} c_i (x^{b+c-m} + x^{i-m}) \bmod f(x) \\ &= \sum_{i=o}^{m-1} c_i x^i + \sum_{i=o}^{m-1-b} c_{i+m} x^{b+i} + \sum_{i=m-b}^{m-1} c_{i+m} x^{b+i} + \sum_{i=o}^{m-1} c_{i+m} x^i \\ &\equiv \sum_{i=o}^{m-1} c_i x^i + \sum_{i=o}^{m-1-b} c_{i+m} x^{b+i} + \sum_{i=m-b}^{m-1} c_{i+m} (x^{2b+i-m} + x^{b+i-m}) \sum_{i=o}^{m-1} c_{i+m} x^i \bmod f(x) \\ &= \sum_{i=o}^{m-1} c_i x^i + \sum_{i=o}^{m-1-b} c_{i+m} x^{b+i} + \sum_{i=0}^{b-1} c_{2i-b+m} x^{b+i} + \sum_{i=o}^{b-1} c_{2i-b+m} x^{b+I} + \sum_{i=o}^{m-1} c_{i+m} x^I \end{aligned} \quad (5.7)$$

ผลลัพธ์จากสมการ (5.7) นั้นจะทำให้มีจำนวน exclusive เท่ากับ 2m+b สำหรับหนึ่งโพลิโนเมียลลดรูป ซึ่งแสดงโครงสร้างดังรูปที่ 5.3 นั่นคือวงจรโพลิโนเมียลลดรูปสามารถนำไปใช้ในการหากำลัง

สองได้โดย การแทรกบิต “0” ระหว่างกลาง เช่น (1111)2 = 1010101 ก่อนนำไปลดรูปด้วยโครงสร้างดังกล่าว ซึ่งส่งผลให้เวลาในการหากำลังสองจะใช้ clock เพียงลูกเดียว



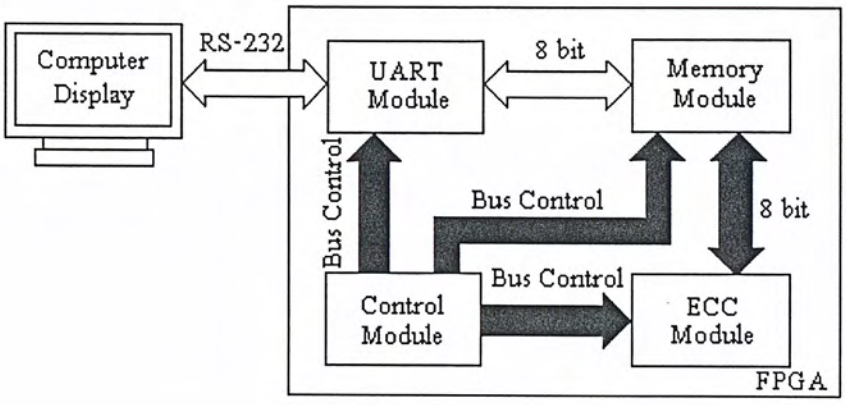
รูปที่ 5.3 แสดงโครงสร้างของการลดรูปโพลิโนเมียล

5.9 การออกแบบระบบ ECC

5.9.1 สถาปัตยกรรมของระบบ ECC

สถาปัตยกรรมของระบบที่ออกแบบประกอบไปด้วยส่วนประกอบต่าง ๆ คือ ส่วนของการสื่อสารข้อมูล (UART Module) ส่วนของหน่วยความจำ (Memory Module) ส่วนประมวลผลของ Elliptic Curve (ECC Module) และส่วนของการควบคุม (Control Module) สามารถแสดงได้ดังรูปที่ 5.4

ในการออกแบบได้ทำการออกแบบให้สมาร์ทการ์ดรับและส่งข้อมูลดิจิตอลจากคอมพิวเตอร์ผ่านทางช่องสื่อสารอนุกรม ตามมาตรฐาน RS-232 โดยใช้ออสซิเลเตอร์ขนาด 1 MHz ทำหน้าที่เป็นสัญญาณนาฬิกาของสมาร์ทการ์ด จากรูปที่ 5.4 ข้อมูลจากคอมพิวเตอร์ที่ส่งเข้ามา จะเข้าไปยังส่วนของการสื่อสารข้อมูล (UART Module) เพื่อทำการแปลงข้อมูลตามโปรโตคอลของ RS-232 ให้เป็นข้อมูลดิจิตอลขนาด 8 บิต แล้วจึงเก็บไว้ยังส่วนของหน่วยความจำ (Memory Module) เพื่อที่จะรอคำสั่งจากหน่วยควบคุม (Control Module) ทำการส่งข้อมูลจากหน่วยความจำไปที่ส่วนประมวลผลของ Ellptic Curve จากนั้นผลลัพธ์ที่ได้จะถูกส่งกลับไปเก็บไว้ยังส่วนของหน่วยความจำ ก่อนที่หน่วยของการสื่อสารข้อมูลจะส่งข้อมูลออกไปยังคอมพิวเตอร์ตามโปรโตคอลของ RS-232



รูปที่ 5.4 แสดงส่วนประกอบต่าง ๆ ของสถาปัตยกรรมเพื่อจำลองการทำงานของสมาร์ทการ์ด

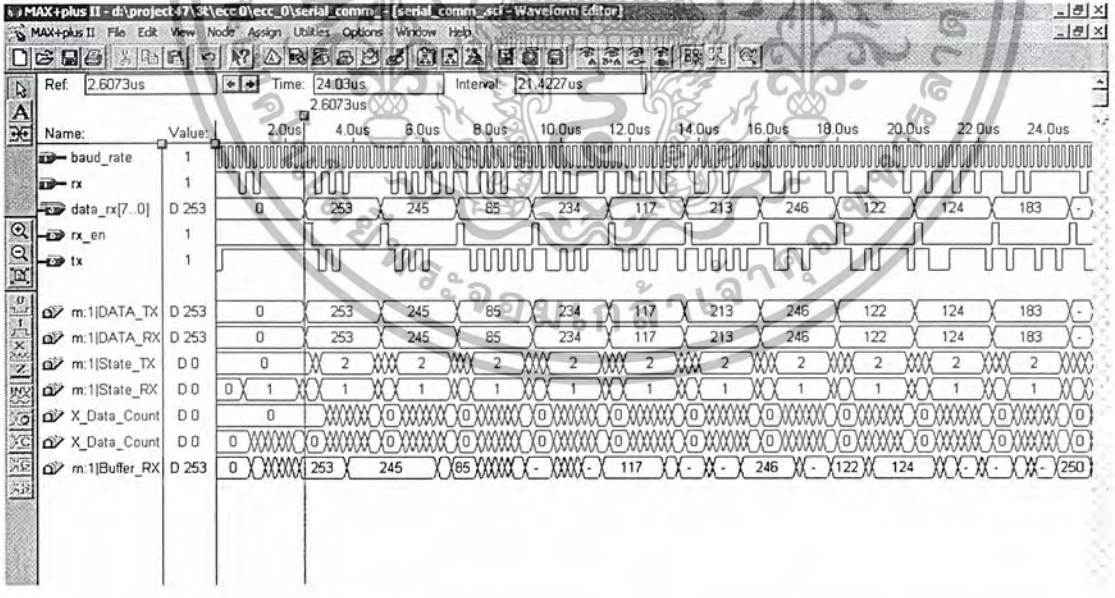
การทำงานของระบบนั้นจะถูกแบ่งออกเป็น 4 ส่วนใหญ่ๆ คือ UART Module, Memory Module, Control Module และ ECC Module โดย UART Module มีหน้าที่ในการควบคุมการรับและการส่งข้อมูลขนาด 8 บิตระหว่างคอมพิวเตอร์ กับ FPGA ส่วนของ Memory Module จะเป็นหน่วยความจำที่ใช้เก็บข้อมูลที่ยังไม่เข้ารหัสและข้อมูลที่เข้ารหัสแล้ว ส่วนของ Control Module จะเป็นส่วนที่ควบคุมการทำงานของระบบทั้งหมด และส่วน ECC Module จะเป็นส่วนของการเข้ารหัสของข้อมูล

บทที่ 6
การทดสอบและผลการทดสอบ

6.1 การทดสอบส่วนการสื่อสารข้อมูล

ส่วนของการสื่อสารข้อมูล จะเป็นการสื่อสารแบบอนุกรม ตามมาตรฐาน RS-232 ระหว่างระบบที่ออกแบบกับคอมพิวเตอร์ โดยใช้ Baud Rate ที่ 110 บิตต่อวินาที ตามมาตรฐานของสมาร์ทการ์ด โดยส่วนของการรับข้อมูลจะรับข้อมูลจากคอมพิวเตอร์ทางขาสัญญาณ RX เมื่อรับข้อมูลจนครบ 8 บิต จะทำการส่งข้อมูลไปยังส่วนต่างๆภายใน FPGA ทางขาสัญญาณ DATA_RX และส่วนของการส่งข้อมูลจาก FPGA ไปยังคอมพิวเตอร์ จะรับข้อมูลเข้ามาทางขา DATA_TX ซึ่งมีขนาด 8 บิต โดยจะมีสัญญาณ TX_EN เป็นตัวควบคุมในการส่ง โดยเมื่อข้อมูลเข้ามาครบ 8 บิต TX_EN จะสั่งให้ส่งข้อมูลออกทางขา TX ไปยังคอมพิวเตอร์

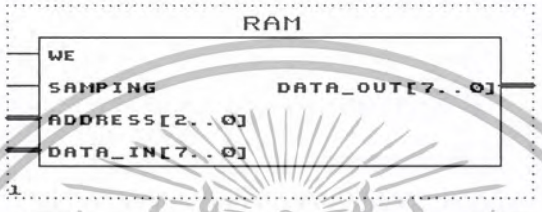
รูปที่ 6.1 แสดงบล็อกไดอะแกรมของส่วนการสื่อสารข้อมูล



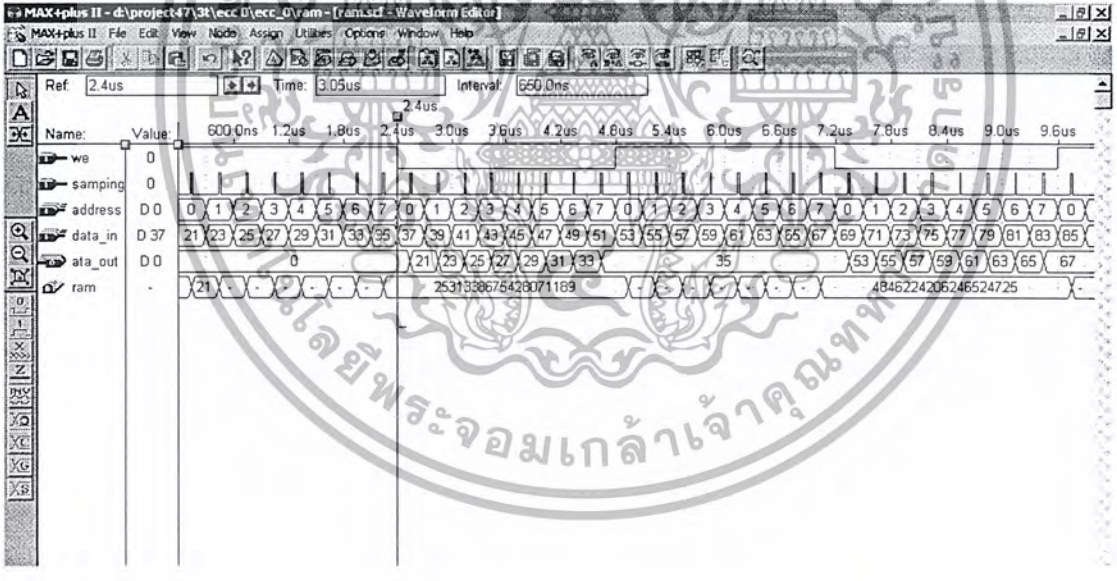
รูปที่ 6.2 แสดง Timming Diagram ของการรับและการส่งข้อมูลระหว่างคอมพิวเตอร์กับ FPGA

6.2 การทดสอบส่วนของหน่วยความจำ

ส่วนของหน่วยความจำจะมีหน้าที่ในการเก็บข้อมูลที่รับมาจากคอมพิวเตอร์ โดยจะรับข้อมูลขนาด 8 บิตเข้ามาทางขา DATA_IN โดยการเขียนหรือการอ่านของหน่วยความจำนี้ จะถูกควบคุมโดยขา WE ถ้าขา WE เป็น "1" จะทำการเขียนข้อมูลที่รับเข้ามาลงไปใน ADDRESS โดย ADDRESS จะถูกสร้างขึ้นจากตัว Control โดย 1 แอดเดรส จะมีค่า 8 บิต แต่ถ้าขา WE มีค่าเป็น "0" จะทำการอ่านข้อมูลที่อยู่ในแอดเดรสนั้นๆ ขึ้นมา โดยการอ่านและการเขียนนั้นจะต้องมีสัญญาณ SAMPING เป็นตัวควบคุมอีกตัวด้วย โดยข้อมูลจะถูกส่งออกมาทางขา DATA_OUT



รูปที่ 6.3 แสดงบล็อกไดอะแกรมของส่วนหน่วยความจำ

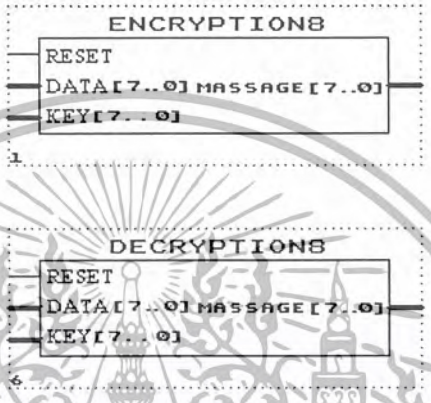


รูปที่ 6.4 แสดง Timming Diagram ของการอ่านและการเขียนข้อมูลในหน่วยความจำ

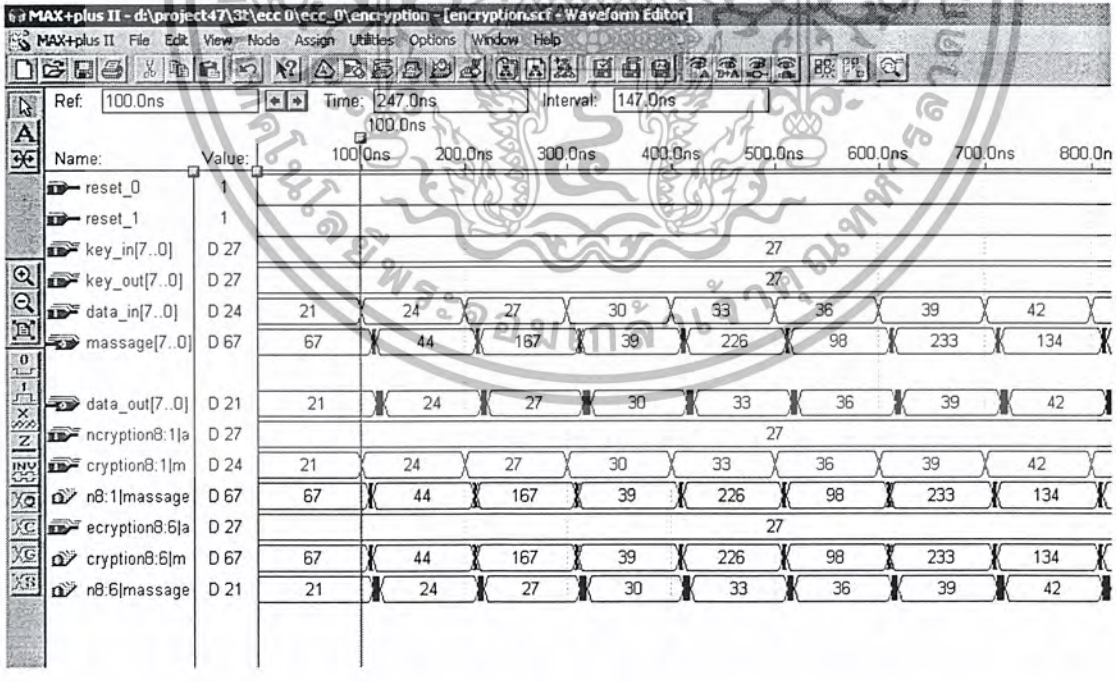
6.3 การทดสอบส่วนของการเข้ารหัสและการถอดรหัส

- การเข้ารหัสเพื่อปกปิดข้อมูล มีขั้นตอนในการเข้ารหัสคือ จะรับข้อมูล กับกุญแจรหัสมาพร้อมกัน แล้วจึงทำการเข้ารหัสข้อมูล โดยใช้วิธีการทางคณิตศาสตร์ ที่มีการทำงานบนสนามจำกัดจะทำให้ ได้ข้อมูลที่ถูกเข้ารหัสออกมา

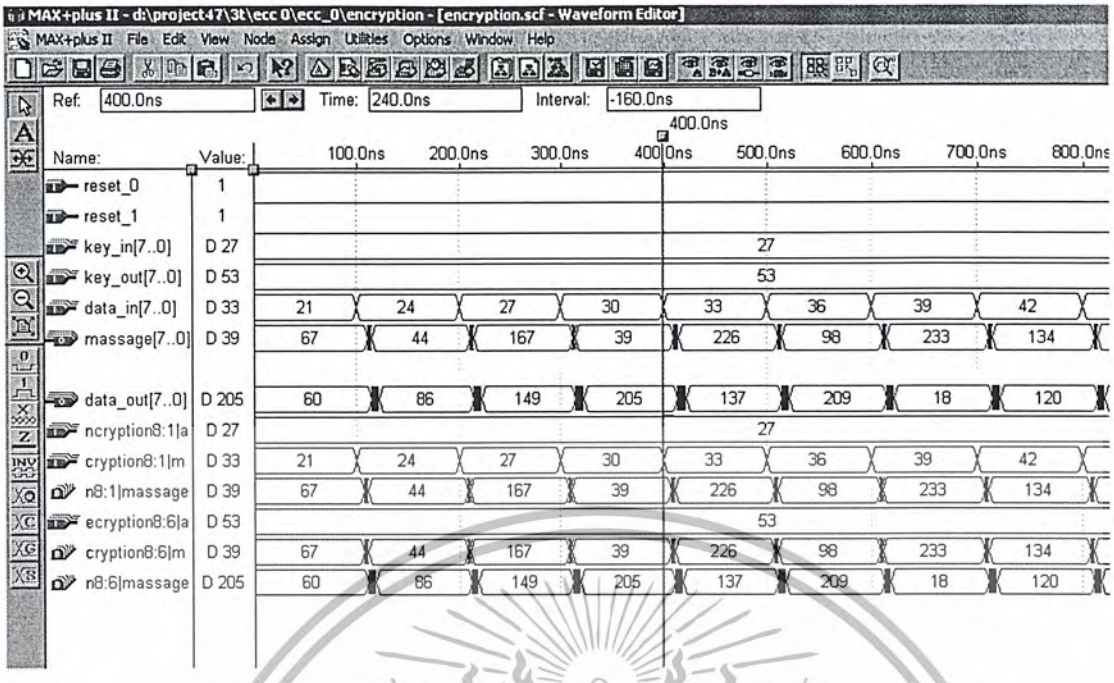
- การถอดรหัสจะทำงานโดย การรับข้อมูลที่ได้ทำการเข้ารหัสแล้วมาพร้อมกับกุญแจที่ถูกต้อง มาทำการถอดรหัสด้วยวิธีการทางคณิตศาสตร์จะได้ข้อมูลแบบเดิมออกมา ถือว่าการถอดรหัสนั้นสมบูรณ์ แต่ถ้าเราใช้กุญแจที่มีค่าไม่ตรงกับตอนที่ทำการเข้ารหัส เราก็ไม่สามารถที่จะถอดรหัสข้อมูลเดิมออกมาได้



รูปที่ 6.5 แสดงบล็อกไดอะแกรมของส่วนการเข้ารหัสและการถอดรหัสของข้อมูล



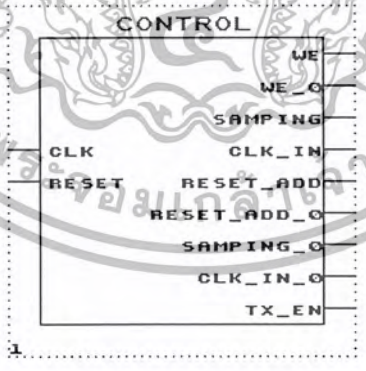
รูปที่ 6.6 แสดง Timming Diagram ของการเข้ารหัสและการถอดรหัสโดยใช้กุญแจตัวเดียวกัน



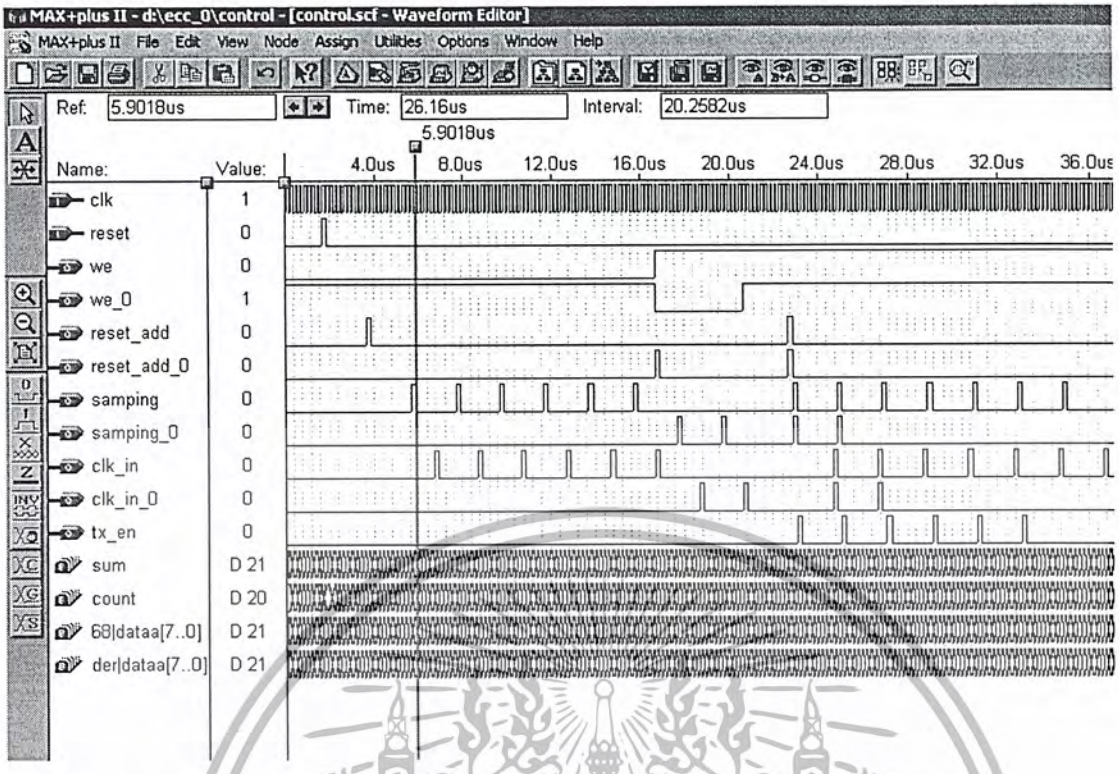
รูปที่ 6.7 แสดง Timming Diagram ของการเข้ารหัสและการถอดรหัสโดยใช้กุญแจคนละตัว

6.4 การทดสอบส่วนของ Control ของระบบ

ในส่วนของ Control จะเป็นตัวควบคุมการทำงานของ UART Module ,Memory Module และ การเข้ารหัสและถอดรหัส ให้สามารถทำงานร่วมกันได้ อย่างมีประสิทธิภาพ

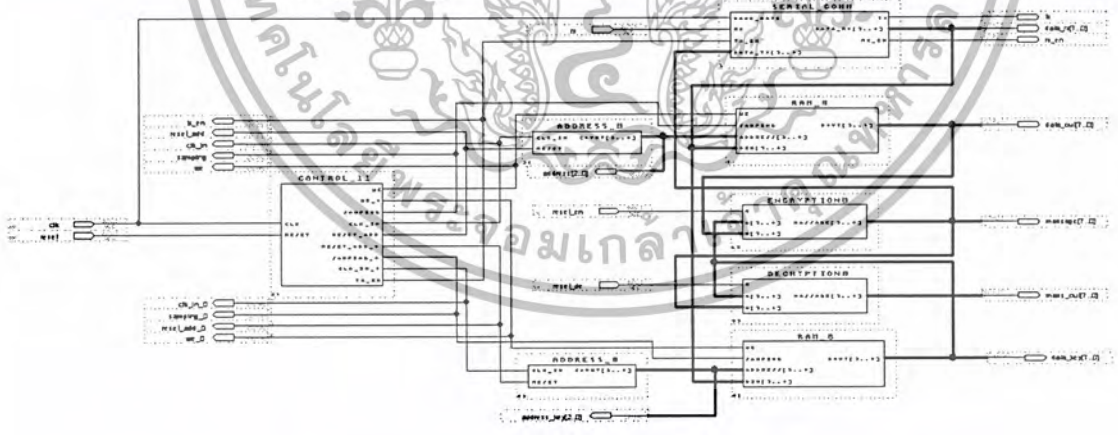


รูปที่ 6.8 แสดงบล็อกไดอะแกรมของส่วน Control

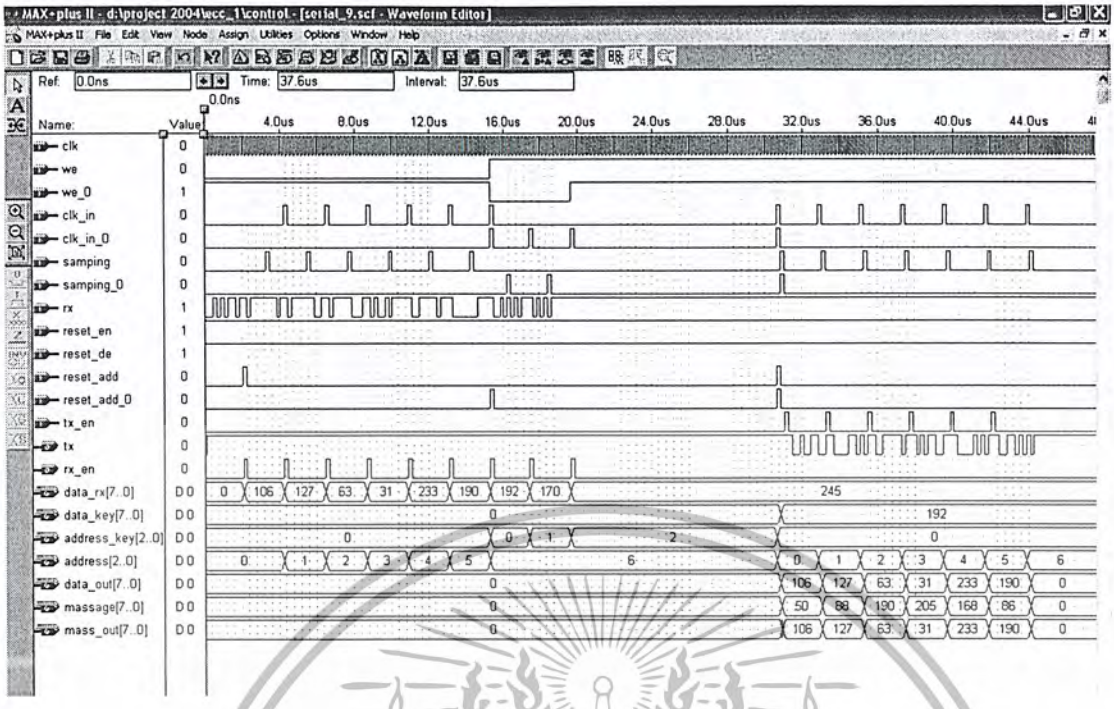


รูปที่ 6.9 แสดง Timming Diagram การทำงานของส่วน Control

6.5 การทดสอบการทำงานของระบบ



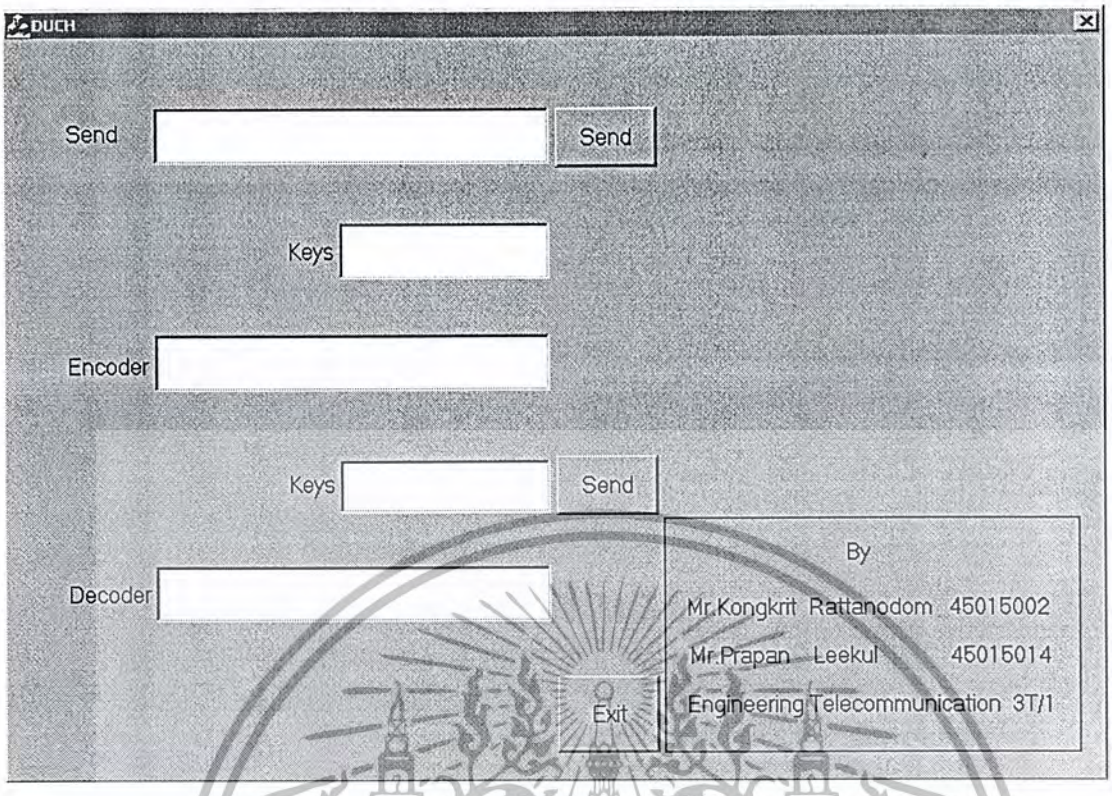
รูปที่ 6.10 แสดงบล็อกไดอะแกรมของส่วนประกอบทั้งหมดของระบบ



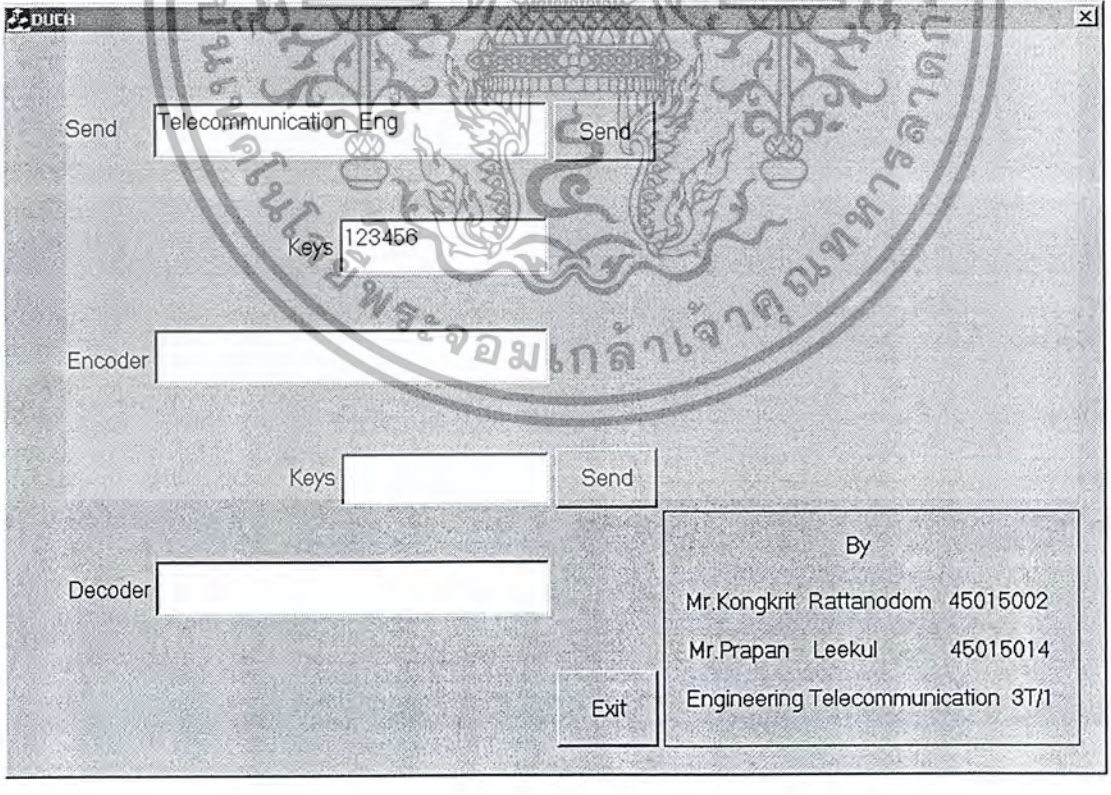
รูปที่ **6.11** แสดง Timming Diagram การทำงานของส่วนประกอบทั้งหมดของระบบ

6.6 ผลการทดลองการเข้าและถอดรหัส

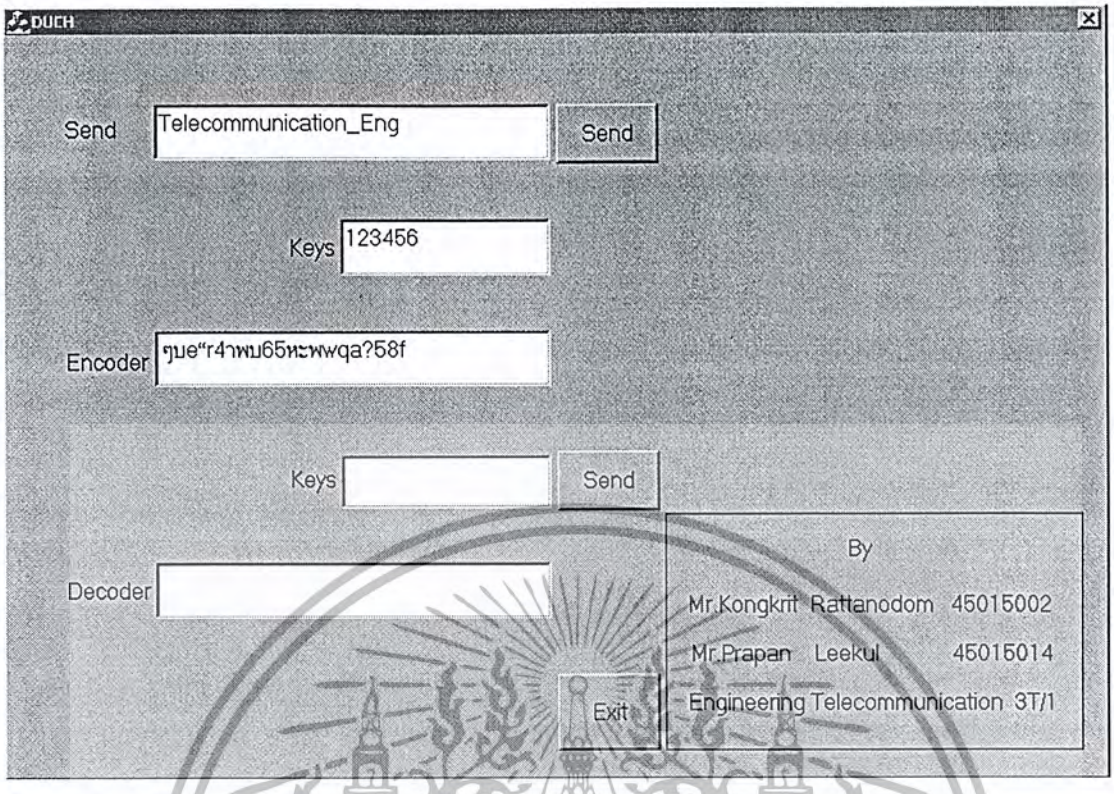
การทดสอบจะทำการโปรแกรมวงจรที่ออกแบบลงบนอุปกรณ์เอฟพีจีเอ โดยจะมีโปรแกรมแสดงผลบนคอมพิวเตอร์เพื่อรับข้อมูลในการเข้าและถอดรหัส โดยมีเราจะทำการป้อนข้อมูลจำนวน 21 ตัวอักษรเข้าไปแล้วก็ป้อนกุญแจรหัสจำนวน 6 ตัวอักษร เพื่อทำการเข้ารหัสข้อมูล โดยเมื่อเราส่งข้อมูลจากคอมพิวเตอร์ไปยังเอฟพีจีเอ เอฟพีจีเอจะทำการเข้ารหัสข้อมูลที่เราส่งไปพร้อมกับส่งค่าที่ทำการเข้ารหัสแล้วออกมายังคอมพิวเตอร์ เพื่อแสดงผลออกทางหน้าจอ และเมื่อเราป้อนกุญแจที่ใช้ในการถอดรหัสเข้าไป เอฟพีจีเอจะทำการถอดรหัส แล้วก็จะทำการส่งข้อมูลที่ถูกถอดรหัสแล้วมาแสดงผลทางหน้าจอ โดยถ้ากุญแจที่เราใช้เข้าและถอดรหัสเป็นกุญแจดอกเดียวกัน ข้อมูลที่เราถอดรหัสออกมาจะมีค่าเท่ากับข้อมูลเดิมที่เราทำการเข้ารหัสเข้าไปในตอนแรก แต่ถ้าเราใช้กุญแจคนละดอกกันข้อมูลที่ถูกถอดรหัสออกมาจะมีค่าไม่ตรงกับค่าที่เราทำการเข้ารหัสในตอนแรก โดยจะแสดงผลการทำงานดังรูปต่อไปนี้



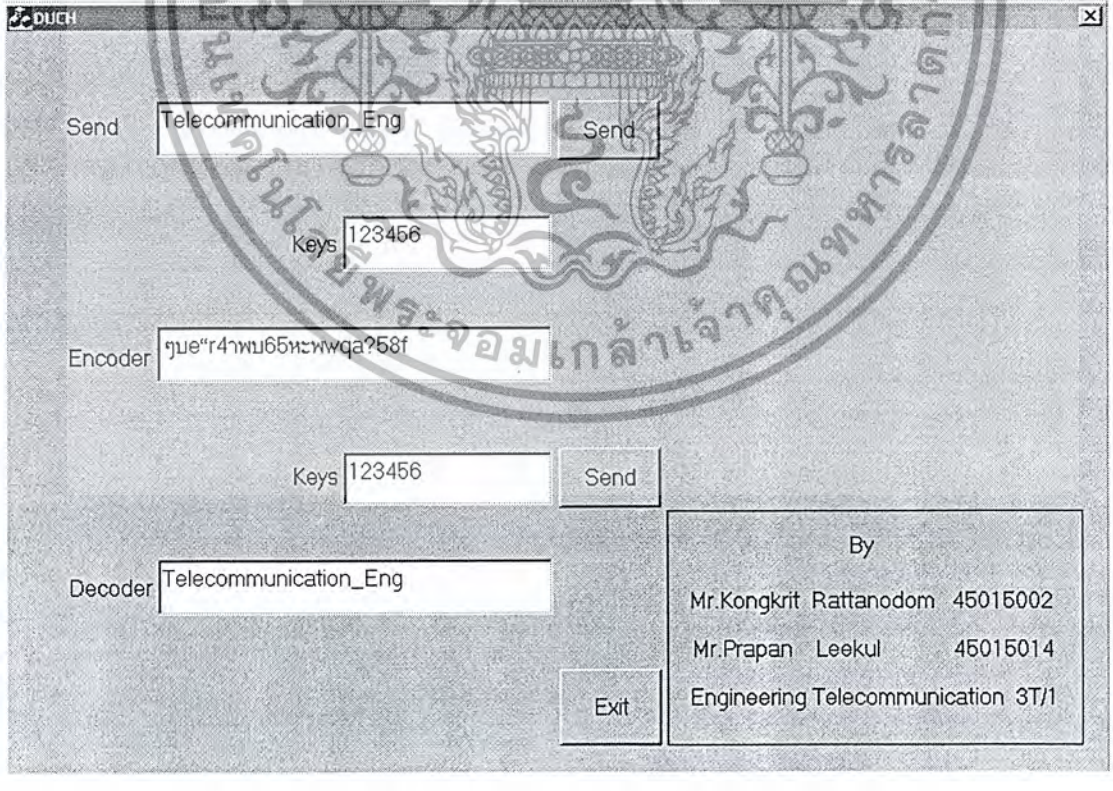
รูปที่ 6.12 แสดงหน้าจอส่วนแสดงผลการเข้าและถอดรหัส



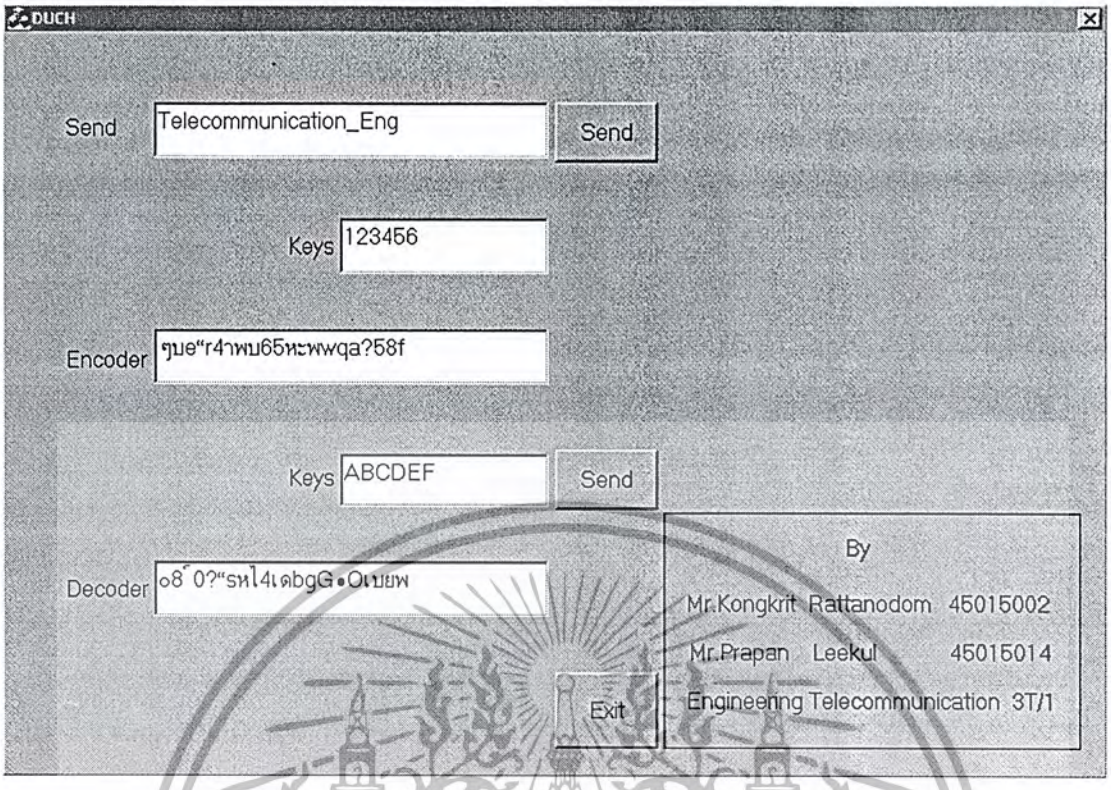
รูปที่ 6.13 แสดงค่าข้อมูลและรหัสที่เราจะทำการเข้ารหัส



รูปที่ 6.14 แสดงค่าที่เราทำการเข้ารหัสเรียบร้อยแล้ว



รูปที่ 6.15 แสดงค่าที่เราทำการถอดรหัสข้อมูลด้วยกุญแจดอกเดียวกัน



รูปที่ 6.16 แสดงค่าที่เราทำการถอดรหัสข้อมูลด้วยกุญแจที่แตกต่างกัน

บทที่ 7
บทสรุปและวิจารณ์

ในปัจจุบันมีการใช้งานสมาร์ทการ์ดอย่างแพร่หลาย โดยที่ระบบรักษาความปลอดภัยถือว่าเป็นหัวใจสำคัญของสมาร์ทการ์ด แต่ด้วยโครงสร้างของสมาร์ทการ์ดที่มีขนาดเล็ก เราจึงต้องออกแบบวงจรที่มีขนาดเล็กเพื่อให้ใช้งานได้เหมาะสมกับสมาร์ทการ์ด

ในโครงงานนี้ได้ทำการออกแบบวงจรโดยใช้ภาษาวีเอสดีแอล ซึ่งเป็นภาษาที่ใช้ในการบรรยายหรืออธิบายรูปแบบการทำงานและความสัมพันธ์ของอุปกรณ์ฮาร์ดแวร์ทั้งในระดับเกทจนถึงระดับดิจิตอลที่ซับซ้อน ทั้งนี้เนื่องจากภาษาวีเอสดีแอล เป็นภาษาที่เหมาะในการนำมาใช้ในการเลียนแบบการทำงานของอุปกรณ์ อีกทั้งยังไม่ขึ้นกับเทคโนโลยี ทำให้ประหยัดเวลาและค่าใช้จ่ายในการออกแบบ จึงได้นำมาใช้กันอย่างกว้างขวางในวงการอุตสาหกรรม รูปแบบของภาษาวีเอสดีแอลประกอบด้วย 2 ส่วนใหญ่ๆ ได้แก่ ส่วนของภาษาซีเควนเชียล (Sequential Language) และภาษาคอนเคอร์เรนท์ (Concurent Language) อีกทั้งการออกแบบสามารถใช้ได้ทั้งสองแบบรวมกัน นอกจากนี้ตัวภาษายังสามารถอธิบายถึงการเชื่องต่อระหว่างระบบย่อยเข้าด้วยกันเพื่อให้เป็นระบบใหญ่ได้และสามารถกำหนดรูปแบบไวยกรณ์ (Syntex) อีกทั้งยังมีการตรวจสอบความหมายของภาษาว่าจะซิมูเลท (Simulate) ได้หรือไม่ เพราะโปรแกรมที่เขียนโดยภาษาวีเอสดีแอลต้องผ่านการซิมูเลทเพื่อตรวจสอบการทำงาน ฉะนั้นในการคอมไพล์ (Compile) จะมีการตรวจสอบทั้งวงจรและซิมูเลชั่นซีแมนติก (Sementic) จึงทำให้สะดวกในการใช้งาน

ระบบเข้ารหัสนั้นจะสามารถแบ่งได้ 2 ประเภทใหญ่ๆคือ แบบรหัสลับกุญแจปกปิด และรหัสลับกุญแจสาธารณะ โดยในโครงงานนี้ ได้ใช้ระบบการเข้ารหัสแบบ Elliptic Curve Cryptosystem(ECC) ซึ่งเป็นรหัสลับแบบกุญแจสาธารณะ เนื่องจากความซับซ้อนของคณิตศาสตร์แบบ Elliptic Curve ทำให้ขนาดของกุญแจรหัสที่ใช้ในการเข้ารหัสมีขนาดเล็กเมื่อเทียบกับการเข้ารหัสลับในแบบอื่น โดยที่ระดับความปลอดภัยยังคงมีความเท่าเทียมกัน และเนื่องจากใช้กุญแจที่มีขนาดเล็กจึงทำให้มีความไวในการประมวลผลสูงและใช้พลังงานต่ำ จึงเหมาะกับสมาร์ทการ์ดซึงมีขาดเล็กและมีพื้นที่ๆจำกัด

ปัญหาที่พบในการทำงาน ส่วนมากจะเกิดจากการที่ ออสซิลเรเตอร์ในวงจรผลิตความถี่ที่มีค่าไม่คงที่ทำให้ค่า Baud Rate ที่ใช้ในวงจรมีการผิดเพี้ยนบ้าง ทำให้เวลาที่เราทำการส่งข้อมูลระหว่างคอมพิวเตอร์กับเอฟพีจีเอ เกิดความผิดพลาดได้ในบางครั้ง

ภาคผนวก

โปรแกรม SERIAL_COMM

```
library ieee;
use ieee.std_logic_1164.all;
-------------------------------------------------------------
entity SERIAL_COM is
port(
      BAUD_RATE : in std_logic;
      RX : in std_logic;
      TX_EN : in std_logic;
      DATA_TX : in std_logic_vector(7 downto 0);
      TX : out std_logic;
      DATA_RX : out std_logic_vector(7 downto 0);
      RX_EN : out std_logic
           );
end SERIAL_COM;
-------------------------------------------------------------
architecture rtl of SERIAL_COM is
type State_Type_RX is (Idel,ReceiveData,Stop);
type State_Type_TX is (Idel,Start,TransData,Stop);
signal State_RX : State_Type_RX := Idel;
signal State_TX : State_Type_TX;
begin
-------------------------------------------------------------
      process(BAUD_RATE,TX_EN,DATA_TX)
      variable TX_Data_Count : integer range 0 to 7 := 0;
      begin
-------------------------------------------------------------
            if BAUD_RATE'Event and BAUD_RATE = '1' then
                  case State_TX is
                        when Idel =>
                              TX <= '1';
                              if TX_EN = '1' then
                                    TX_Data_Count := 0;
                                    State_TX <= Start;
                              else
                                    TX_Data_Count := 0;
                                    State_TX <= Idel;
                              end if;
                        when Start =>
                              TX <= '0';
                              TX_Data_Count := 0;
                              State_TX <= TransData;
                        when TransData =>
                              if TX_Data_Count = 7 then
                                    TX <= DATA_TX(TX_Data_Count);
                                    State_TX <= Stop;
                              else
                                    TX <= DATA_TX(TX_Data_Count);
                                    TX_Data_Count :=TX_Data_Count+1;
                                    State_TX <= TransData;
                              end if;
                        when Stop =>
                              TX <= '1';
                              TX_Data_Count := 0;
                              State_TX <= Idel;
                        when others =>
                              TX_Data_Count := 0;
                              TX <= '1';

                              State_TX <= Idel;
                  end case;
            end if;
      end process;
-------------------------------------------------------------------
      process(BAUD_RATE,RX)
      variable RX_Data_Count : integer range 0 to 7 := 0;
      variable Buffer_RX : std_logic_vector(7 downto 0);
```

```
      begin
            if BAUD_RATE'Event and BAUD_RATE = '1' then
                  case State_RX is
                        when Idel =>
                              RX_EN <= '0';
                              if RX = '0' then
                                    RX_Data_Count := 0;
                                    State_RX <= ReceiveData;
                              else
                                    RX_Data_Count := 0;
                                    State_RX <= Idel;
                              end if;
                        when ReceiveData =>
                              RX_EN <= '0';
                              if RX_Data_Count = 7 then
                                    Buffer_RX(RX_Data_Count) := RX;
                                    State_RX <= Stop;
                              else
                                    Buffer_RX(RX_Data_Count) := RX;
                                    RX_Data_Count := RX_Data_Count+1;
                                    State_RX <= ReceiveData;
                              end if;
                        when Stop =>
                              RX_Data_Count := 0;
                              DATA_RX <= Buffer_RX;
                              RX_EN <= '1';

                              State_RX <= Idel;
                         when others =>
                              RX_Data_Count := 0;
                              RX_EN <= '0';
                                    State_RX <= Idel;
                  end case;
            end if;
      end process;
end;
-------------------------------------------------------------------------
```

โปรแกรมสร้าง Address

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity address is
port(clk,clk_in,reset:in std_logic;
     count:out std_logic_vector(4 downto 0));
end address;
architecture rtl of address is
 signal sum : std_logic_vector(4 downto 0);
begin
-------------------------------------------------
 process(clk,reset)
 begin
        if (reset = '1') then
        sum   <= (others => '0');
        elsif (clk'event and clk = '1') then
        if clk_in = '1' then
        if sum < "11111" then
       sum <= sum + '1';
             else
              sum <= "00000";
             end if;
        end if;
       end if;
       count <= sum;
       end process;
```

```
  end rtl;
```

โปรแกรม การเข้ารหัสลับ

```
library ieee;
use ieee.std_logic_1164.all;
entity encryption_0 is
port(    r : in std_logic;
       a,m : in std_logic_vector(7 downto 0);
   massage :out std_logic_vector(7 downto 0));
end encryption_0;
architecture behave of encryption_0 is
 begin
 process(a)
  variable b : std_logic_vector(7 downto 0);
  variable c : std_logic_vector(7 downto 0);
  variable d : std_logic_vector(7 downto 0);
  variable j : std_logic_vector(7 downto 0);
  variable k : std_logic_vector(7 downto 0);
  variable mas:std_logic_vector(7 downto 0);
 begin
   if r = '1' then --2
    b(7) := (a(6) and a(6)) xor (a(0) and a(7)) xor (a(7) and a(0)) xor
(a(1) and a(6)) xor (a(6) and a(1)) xor (a(2) and a(5)) xor (a(5) and
a(2)) xor (a(3) and a(4)) xor (a(4) and a(3)) xor (a(4) and a(7)) xor
(a(7) and a(4)) xor (a(5) and a(6)) xor (a(6) and a(5)) xor (a(5) and
a(7)) xor (a(7) and a(5)) xor (a(6) and a(7)) xor (a(7) and a(6));
    b(6) := (a(3) and a(3)) xor (a(5) and a(5)) xor (a(6) and a(6)) xor
(a(0) and a(6)) xor (a(6) and a(0)) xor (a(1) and a(5)) xor (a(5) and
a(1)) xor (a(2) and a(4)) xor (a(4) and a(2)) xor (a(3) and a(7)) xor
(a(7) and a(3)) xor (a(4) and a(6)) xor (a(6) and a(4)) xor (a(4) and
a(7)) xor (a(7) and a(4)) xor (a(5) and a(6)) xor (a(6) and a(5)) xor
(a(5) and a(7)) xor (a(7) and a(5));
    b(5) := (a(5) and a(5)) xor (a(0) and a(5)) xor (a(5) and a(0)) xor
(a(1) and a(4)) xor (a(4) and a(1)) xor (a(2) and a(3)) xor (a(3) and
a(2)) xor (a(2) and a(7)) xor (a(7) and a(2)) xor (a(3) and a(6)) xor
(a(6) and a(3)) xor (a(4) and a(5)) xor (a(5) and a(4)) xor (a(3) and
a(7)) xor (a(7) and a(3)) xor (a(4) and a(6)) xor (a(6) and a(4)) xor
(a(4) and a(7)) xor (a(7) and a(4)) xor (a(5) and a(6)) xor (a(6) and
a(5));
    b(4) := (a(2) and a(2)) xor (a(4) and a(4)) xor (a(5) and a(5)) xor
(a(7) and a(7)) xor (a(0) and a(4)) xor (a(4) and a(0)) xor (a(1) and
a(3)) xor (a(3) and a(1)) xor (a(1) and a(7)) xor (a(7) and a(1)) xor
(a(2) and a(6)) xor (a(6) and a(2)) xor (a(3) and a(5)) xor (a(5) and
a(3)) xor (a(2) and a(7)) xor (a(7) and a(2)) xor (a(3) and a(6)) xor
(a(6) and a(3)) xor (a(4) and a(5)) xor (a(5) and a(4)) xor (a(3) and
a(7)) xor (a(7) and a(3)) xor (a(4) and a(6)) xor (a(6) and a(4));
    b(3) := (a(4) and a(4)) xor (a(6) and a(6)) xor (a(0) and a(3)) xor
(a(3) and a(0)) xor (a(1) and a(2)) xor (a(2) and a(1)) xor (a(1) and
a(7)) xor (a(7) and a(1)) xor (a(2) and a(6)) xor (a(6) and a(2)) xor
(a(3) and a(5)) xor (a(5) and a(3)) xor (a(2) and a(7)) xor (a(7) and
a(2)) xor (a(3) and a(6)) xor (a(6) and a(3)) xor (a(4) and a(5)) xor
(a(5) and a(4)) xor (a(4) and a(7)) xor (a(7) and a(4)) xor (a(5) and
a(6)) xor (a(6) and a(5)) xor (a(5) and a(7)) xor (a(7) and a(5));
    b(2) := (a(1) and a(1)) xor (a(4) and a(4)) xor (a(5) and a(5)) xor
(a(6) and a(6)) xor (a(0) and a(2)) xor (a(2) and a(0)) xor (a(1) and
a(7)) xor (a(7) and a(1)) xor (a(2) and a(6)) xor (a(6) and a(2)) xor
(a(3) and a(5)) xor (a(5) and a(3)) xor (a(3) and a(7)) xor (a(7) and
a(3)) xor (a(4) and a(6)) xor (a(6) and a(4)) xor (a(5) and a(7)) xor
(a(7) and a(5)) xor (a(6) and a(7)) xor (a(7) and a(6));
    b(1) := (a(7) and a(7)) xor (a(0) and a(1)) xor (a(1) and a(0)) xor
(a(2) and a(7)) xor (a(7) and a(2)) xor (a(3) and a(6)) xor (a(6) and
a(3)) xor (a(4) and a(5)) xor (a(5) and a(4)) xor (a(6) and a(7)) xor
(a(7) and a(6));
```

```
    b(0) := (a(0) and a(0)) xor (a(4) and a(4)) xor (a(6) and a(6)) xor
(a(7) and a(7)) xor (a(1) and a(7)) xor (a(7) and a(1)) xor (a(2) and
a(6)) xor (a(6) and a(2)) xor (a(3) and a(5)) xor (a(5) and a(3)) xor
(a(5) and a(7)) xor (a(7) and a(5)) xor (a(6) and a(7)) xor (a(7) and
b(6));
   end if;
   if r = '1' then --4
    c(7) := (b(6) and b(6)) xor (b(0) and b(7)) xor (b(7) and b(0)) xor
(b(1) and b(6)) xor (b(6) and b(1)) xor (b(2) and b(5)) xor (b(5) and
b(2)) xor (b(3) and b(4)) xor (b(4) and b(3)) xor (b(4) and b(7)) xor
(b(7) and b(4)) xor (b(5) and b(6)) xor (b(6) and b(5)) xor (b(5) and
b(7)) xor (b(7) and b(5)) xor (b(6) and b(7)) xor (b(7) and b(6));
    c(6) := (b(3) and b(3)) xor (b(5) and b(5)) xor (b(6) and b(6)) xor
(b(0) and b(6)) xor (b(6) and b(0)) xor (b(1) and b(5)) xor (b(5) and
b(1)) xor (b(2) and b(4)) xor (b(4) and b(2)) xor (b(3) and b(7)) xor
(b(7) and b(3)) xor (b(4) and b(6)) xor (b(6) and b(4)) xor (b(4) and
b(7)) xor (b(7) and b(4)) xor (b(5) and b(6)) xor (b(6) and b(5)) xor
(b(5) and b(7)) xor (b(7) and b(5));
    c(5) := (b(5) and b(5)) xor (b(0) and b(5)) xor (b(5) and b(0)) xor
(b(1) and b(4)) xor (b(4) and b(1)) xor (b(2) and b(3)) xor (b(3) and
b(2)) xor (b(2) and b(7)) xor (b(7) and b(2)) xor (b(3) and b(6)) xor
(b(6) and b(3)) xor (b(4) and b(5)) xor (b(5) and b(4)) xor (b(3) and
b(7)) xor (b(7) and b(3)) xor (b(4) and b(6)) xor (b(6) and b(4)) xor
(b(4) and b(7)) xor (b(7) and b(4)) xor (b(5) and b(6)) xor (b(6) and
b(5));
    c(4) := (b(2) and b(2)) xor (b(4) and b(4)) xor (b(5) and b(5)) xor
(b(7) and b(7)) xor (b(0) and b(4)) xor (b(4) and b(0)) xor (b(1) and
b(3)) xor (b(3) and b(1)) xor (b(1) and b(7)) xor (b(7) and b(1)) xor
(b(2) and b(6)) xor (b(6) and b(2)) xor (b(3) and b(5)) xor (b(5) and
b(3)) xor (b(2) and b(7)) xor (b(7) and b(2)) xor (b(3) and b(6)) xor
(b(6) and b(3)) xor (b(4) and b(5)) xor (b(5) and b(4)) xor (b(3) and
b(7)) xor (b(7) and b(3)) xor (b(4) and b(6)) xor (b(6) and b(4));
    c(3) := (b(4) and b(4)) xor (b(6) and b(6)) xor (b(0) and b(3)) xor
(b(3) and b(0)) xor (b(1) and b(2)) xor (b(2) and b(1)) xor (b(1) and
b(7)) xor (b(7) and b(1)) xor (b(2) and b(6)) xor (b(6) and b(2)) xor
(b(3) and b(5)) xor (b(5) and b(3)) xor (b(2) and b(7)) xor (b(7) and
b(2)) xor (b(3) and b(6)) xor (b(6) and b(3)) xor (b(4) and b(5)) xor
(b(5) and b(4)) xor (b(4) and b(7)) xor (b(7) and b(4)) xor (b(5) and
b(6)) xor (b(6) and b(5)) xor (b(5) and b(7)) xor (b(7) and b(5));
    c(2) := (b(1) and b(1)) xor (b(4) and b(4)) xor (b(5) and b(5)) xor
(b(6) and b(6)) xor (b(0) and b(2)) xor (b(2) and b(0)) xor (b(1) and
b(7)) xor (b(7) and b(1)) xor (b(2) and b(6)) xor (b(6) and b(2)) xor
(b(3) and b(5)) xor (b(5) and b(3)) xor (b(3) and b(7)) xor (b(7) and
b(3)) xor (b(4) and b(6)) xor (b(6) and b(4)) xor (b(5) and b(7)) xor
(b(7) and b(5)) xor (b(6) and b(7)) xor (b(7) and b(6));
    c(1) := (b(7) and b(7)) xor (b(0) and b(1)) xor (b(1) and b(0)) xor
(b(2) and b(7)) xor (b(7) and b(2)) xor (b(3) and b(6)) xor (b(6) and
b(3)) xor (b(4) and b(5)) xor (b(5) and b(4)) xor (b(6) and b(7)) xor
(b(7) and b(6));
    c(0) := (b(0) and b(0)) xor (b(4) and b(4)) xor (b(6) and b(6)) xor
(b(7) and b(7)) xor (b(1) and b(7)) xor (b(7) and b(1)) xor (b(2) and
b(6)) xor (b(6) and b(2)) xor (b(3) and b(5)) xor (b(5) and b(3)) xor
(b(5) and b(7)) xor (b(7) and b(5)) xor (b(6) and b(7)) xor (b(7) and
b(6));
   end if;
   if r = '1' then --8
    d(7) := (c(6) and c(6)) xor (c(0) and c(7)) xor (c(7) and c(0)) xor
(c(1) and c(6)) xor (c(6) and c(1)) xor (c(2) and c(5)) xor (c(5) and
c(2)) xor (c(3) and c(4)) xor (c(4) and c(3)) xor (c(4) and c(7)) xor
(c(7) and c(4)) xor (c(5) and c(6)) xor (c(6) and c(5)) xor (c(5) and
c(7)) xor (c(7) and c(5)) xor (c(6) and c(7)) xor (c(7) and c(6));
    d(6) := (c(3) and c(3)) xor (c(5) and c(5)) xor (c(6) and c(6)) xor
(c(0) and c(6)) xor (c(6) and c(0)) xor (c(1) and c(5)) xor (c(5) and
c(1)) xor (c(2) and c(4)) xor (c(4) and c(2)) xor (c(3) and c(7)) xor
(c(7) and c(3)) xor (c(4) and c(6)) xor (c(6) and c(4)) xor (c(4) and
c(7)) xor (c(7) and c(4)) xor (c(5) and c(6)) xor (c(6) and c(5)) xor
(c(5) and c(7)) xor (c(7) and c(5));
    d(5) := (c(5) and c(5)) xor (c(0) and c(5)) xor (c(5) and c(0)) xor
(c(1) and c(4)) xor (c(4) and c(1)) xor (c(2) and c(3)) xor (c(3) and
```

```
c(2)) xor (c(2) and c(7)) xor (c(7) and c(2)) xor (c(3) and c(6)) xor
(c(6) and c(3)) xor (c(4) and c(5)) xor (c(5) and c(4)) xor (c(3) and
c(7)) xor (c(7) and c(3)) xor (c(4) and c(6)) xor (c(6) and c(4)) xor
(c(4) and c(7)) xor (c(7) and c(4)) xor (c(5) and c(6)) xor (c(6) and
c(5));
    d(4) := (c(2) and c(2)) xor (c(4) and c(4)) xor (c(5) and c(5)) xor
(c(7) and c(7)) xor (c(0) and c(4)) xor (c(4) and c(0)) xor (c(1) and
c(3)) xor (c(3) and c(1)) xor (c(1) and c(7)) xor (c(7) and c(1)) xor
(c(2) and c(6)) xor (c(6) and c(2)) xor (c(3) and c(5)) xor (c(5) and
c(3)) xor (c(2) and c(7)) xor (c(7) and c(2)) xor (c(3) and c(6)) xor
(c(6) and c(3)) xor (c(4) and c(5)) xor (c(5) and c(4)) xor (c(3) and
c(7)) xor (c(7) and c(3)) xor (c(4) and c(6)) xor (c(6) and c(4));
    d(3) := (c(4) and c(4)) xor (c(6) and c(6)) xor (c(0) and c(3)) xor
(c(3) and c(0)) xor (c(1) and c(2)) xor (c(2) and c(1)) xor (c(1) and
c(7)) xor (c(7) and c(1)) xor (c(2) and c(6)) xor (c(6) and c(2)) xor
(c(3) and c(5)) xor (c(5) and c(3)) xor (c(2) and c(7)) xor (c(7) and
c(2)) xor (c(3) and c(6)) xor (c(6) and c(3)) xor (c(4) and c(5)) xor
(c(5) and c(4)) xor (c(4) and c(7)) xor (c(7) and c(4)) xor (c(5) and
c(6)) xor (c(6) and c(5)) xor (c(5) and c(7)) xor (c(7) and c(5));
    d(2) := (c(1) and c(1)) xor (c(4) and c(4)) xor (c(5) and c(5)) xor
(c(6) and c(6)) xor (c(0) and c(2)) xor (c(2) and c(0)) xor (c(1) and
c(7)) xor (c(7) and c(1)) xor (c(2) and c(6)) xor (c(6) and c(2)) xor
(c(3) and c(5)) xor (c(5) and c(3)) xor (c(3) and c(7)) xor (c(7) and
c(3)) xor (c(4) and c(6)) xor (c(6) and c(4)) xor (c(5) and c(7)) xor
(c(7) and c(5)) xor (c(6) and c(7)) xor (c(7) and c(6));
    d(1) := (c(7) and c(7)) xor (c(0) and c(1)) xor (c(1) and c(0)) xor
(c(2) and c(7)) xor (c(7) and c(2)) xor (c(3) and c(6)) xor (c(6) and
c(3)) xor (c(4) and c(5)) xor (c(5) and c(4)) xor (c(6) and c(7)) xor
(c(7) and c(6));
    d(0) := (c(0) and c(0)) xor (c(4) and c(4)) xor (c(6) and c(6)) xor
(c(7) and c(7)) xor (c(1) and c(7)) xor (c(7) and c(1)) xor (c(2) and
c(6)) xor (c(6) and c(2)) xor (c(3) and c(5)) xor (c(5) and c(3)) xor
(c(5) and c(7)) xor (c(7) and c(5)) xor (c(6) and c(7)) xor (c(7) and
c(6));
   end if;
   if r = '1' then --12
    j(7) := (d(6) and c(6)) xor (d(0) and c(7)) xor (d(7) and c(0)) xor
(d(1) and c(6)) xor (d(6) and c(1)) xor (d(2) and c(5)) xor (d(5) and
c(2)) xor (d(3) and c(4)) xor (d(4) and c(3)) xor (d(4) and c(7)) xor
(d(7) and c(4)) xor (d(5) and c(6)) xor (d(6) and c(5)) xor (d(5) and
c(7)) xor (d(7) and c(5)) xor (d(6) and c(7)) xor (d(7) and c(6));
    j(6) := (d(3) and c(3)) xor (d(5) and c(5)) xor (d(6) and c(6)) xor
(d(0) and c(6)) xor (d(6) and c(0)) xor (d(1) and c(5)) xor (d(5) and
c(1)) xor (d(2) and c(4)) xor (d(4) and c(2)) xor (d(3) and c(7)) xor
(d(7) and c(3)) xor (d(4) and c(6)) xor (d(6) and c(4)) xor (d(4) and
c(7)) xor (d(7) and c(4)) xor (d(5) and c(6)) xor (d(6) and c(5)) xor
(d(5) and c(7)) xor (d(7) and c(5));
    j(5) := (d(5) and c(5)) xor (d(0) and c(5)) xor (d(5) and c(0)) xor
(d(1) and c(4)) xor (d(4) and c(1)) xor (d(2) and c(3)) xor (d(3) and
c(2)) xor (d(2) and c(7)) xor (d(7) and c(2)) xor (d(3) and c(6)) xor
(d(6) and c(3)) xor (d(4) and c(5)) xor (d(5) and c(4)) xor (d(3) and
c(7)) xor (d(7) and c(3)) xor (d(4) and c(6)) xor (d(6) and c(4)) xor
(d(4) and c(7)) xor (d(7) and c(4)) xor (d(5) and c(6)) xor (d(6) and
c(5));
    j(4) := (d(2) and c(2)) xor (d(4) and c(4)) xor (d(5) and c(5)) xor
(d(7) and c(7)) xor (d(0) and c(4)) xor (d(4) and c(0)) xor (d(1) and
c(3)) xor (d(3) and c(1)) xor (d(1) and c(7)) xor (d(7) and c(1)) xor
(d(2) and c(6)) xor (d(6) and c(2)) xor (d(3) and c(5)) xor (d(5) and
c(3)) xor (d(2) and c(7)) xor (d(7) and c(2)) xor (d(3) and c(6)) xor
(d(6) and c(3)) xor (d(4) and c(5)) xor (d(5) and c(4)) xor (d(3) and
c(7)) xor (d(7) and c(3)) xor (d(4) and c(6)) xor (d(6) and c(4));
    j(3) := (d(4) and c(4)) xor (d(6) and c(6)) xor (d(0) and c(3)) xor
(d(3) and c(0)) xor (d(1) and c(2)) xor (d(2) and c(1)) xor (d(1) and
c(7)) xor (d(7) and c(1)) xor (d(2) and c(6)) xor (d(6) and c(2)) xor
(d(3) and c(5)) xor (d(5) and c(3)) xor (d(2) and c(7)) xor (d(7) and
c(2)) xor (d(3) and c(6)) xor (d(6) and c(3)) xor (d(4) and c(5)) xor
(d(5) and c(4)) xor (d(4) and c(7)) xor (d(7) and c(4)) xor (d(5) and
c(6)) xor (d(6) and c(5)) xor (d(5) and c(7)) xor (d(7) and c(5));
```

```
    j(2) := (d(1) and c(1)) xor (d(4) and c(4)) xor (d(5) and c(5)) xor
(d(6) and c(6)) xor (d(0) and c(2)) xor (d(2) and c(0)) xor (d(1) and
c(7)) xor (d(7) and c(1)) xor (d(2) and c(6)) xor (d(6) and c(2)) xor
(d(3) and c(5)) xor (d(5) and c(3)) xor (d(3) and c(7)) xor (d(7) and
c(3)) xor (d(4) and c(6)) xor (d(6) and c(4)) xor (d(5) and c(7)) xor
(d(7) and c(5)) xor (d(6) and c(7)) xor (d(7) and c(6));
    j(1) := (d(7) and c(7)) xor (d(0) and c(1)) xor (d(1) and c(0)) xor
(d(2) and c(7)) xor (d(7) and c(2)) xor (d(3) and c(6)) xor (d(6) and
c(3)) xor (d(4) and c(5)) xor (d(5) and c(4)) xor (d(6) and c(7)) xor
(d(7) and c(6));
    j(0) := (d(0) and c(0)) xor (d(4) and c(4)) xor (d(6) and c(6)) xor
(d(7) and c(7)) xor (d(1) and c(7)) xor (d(7) and c(1)) xor (d(2) and
c(6)) xor (d(6) and c(2)) xor (d(3) and c(5)) xor (d(5) and c(3)) xor
(d(5) and c(7)) xor (d(7) and c(5)) xor (d(6) and c(7)) xor (d(7) and
c(6));
   end if;
   if r = '1' then --13
    k(7) := (j(6) and a(6)) xor (j(0) and a(7)) xor (j(7) and a(0)) xor
(j(1) and a(6)) xor (j(6) and a(1)) xor (j(2) and a(5)) xor (j(5) and
a(2)) xor (j(3) and a(4)) xor (j(4) and a(3)) xor (j(4) and a(7)) xor
(j(7) and a(4)) xor (j(5) and a(6)) xor (j(6) and a(5)) xor (j(5) and
a(7)) xor (j(7) and a(5)) xor (j(6) and a(7)) xor (j(7) and a(6));
    k(6) := (j(3) and a(3)) xor (j(5) and a(5)) xor (j(6) and a(6)) xor
(j(0) and a(6)) xor (j(6) and a(0)) xor (j(1) and a(5)) xor (j(5) and
a(1)) xor (j(2) and a(4)) xor (j(4) and a(2)) xor (j(3) and a(7)) xor
(j(7) and a(3)) xor (j(4) and a(6)) xor (j(6) and a(4)) xor (j(4) and
a(7)) xor (j(7) and a(4)) xor (j(5) and a(6)) xor (j(6) and a(5)) xor
(j(5) and a(7)) xor (j(7) and a(5));
    k(5) := (j(5) and a(5)) xor (j(0) and a(5)) xor (j(5) and a(0)) xor
(j(1) and a(4)) xor (j(4) and a(1)) xor (j(2) and a(3)) xor (j(3) and
a(2)) xor (j(2) and a(7)) xor (j(7) and a(2)) xor (j(3) and a(6)) xor
(j(6) and a(3)) xor (j(4) and a(5)) xor (j(5) and a(4)) xor (j(3) and
a(7)) xor (j(7) and a(3)) xor (j(4) and a(6)) xor (j(6) and a(4)) xor
(j(4) and a(7)) xor (j(7) and a(4)) xor (j(5) and a(6)) xor (j(6) and
a(5));
    k(4) := (j(2) and a(2)) xor (j(4) and a(4)) xor (j(5) and a(5)) xor
(j(7) and a(7)) xor (j(0) and a(4)) xor (j(4) and a(0)) xor (j(1) and
a(3)) xor (j(3) and a(1)) xor (j(1) and a(7)) xor (j(7) and a(1)) xor
(j(2) and a(6)) xor (j(6) and a(2)) xor (j(3) and a(5)) xor (j(5) and
a(3)) xor (j(2) and a(7)) xor (j(7) and a(2)) xor (j(3) and a(6)) xor
(j(6) and a(3)) xor (j(4) and a(5)) xor (j(5) and a(4)) xor (j(3) and
a(7)) xor (j(7) and a(3)) xor (j(4) and a(6)) xor (j(6) and a(4));
    k(3) := (j(4) and a(4)) xor (j(6) and a(6)) xor (j(0) and a(3)) xor
(j(3) and a(0)) xor (j(1) and a(2)) xor (j(2) and a(1)) xor (j(1) and
a(7)) xor (j(7) and a(1)) xor (j(2) and a(6)) xor (j(6) and a(2)) xor
(j(3) and a(5)) xor (j(5) and a(3)) xor (j(2) and a(7)) xor (j(7) and
a(2)) xor (j(3) and a(6)) xor (j(6) and a(3)) xor (j(4) and a(5)) xor
(j(5) and a(4)) xor (j(4) and a(7)) xor (j(7) and a(4)) xor (j(5) and
a(6)) xor (j(6) and a(5)) xor (j(5) and a(7)) xor (j(7) and a(5));
    k(2) := (j(1) and a(1)) xor (j(4) and a(4)) xor (j(5) and a(5)) xor
(j(6) and a(6)) xor (j(0) and a(2)) xor (j(2) and a(0)) xor (j(1) and
a(7)) xor (j(7) and a(1)) xor (j(2) and a(6)) xor (j(6) and a(2)) xor
(j(3) and a(5)) xor (j(5) and a(3)) xor (j(3) and a(7)) xor (j(7) and
a (3)) xor (j(4) and a(6)) xor (j(6) and a(4)) xor (j(5) and a(7)) xor
(j(7) and a(5)) xor (j(6) and a(7)) xor (j(7) and a(6));
    k(1) := (j(7) and a(7)) xor (j(0) and a(1)) xor (j(1) and a(0)) xor
(j(2) and a(7)) xor (j(7) and a(2)) xor (j(3) and a(6)) xor (j(6) and
a(3)) xor (j(4) and a(5)) xor (j(5) and a(4)) xor (j(6) and a(7)) xor
(j(7) and a(6));
    k(0) := (j(0) and a(0)) xor (j(4) and a(4)) xor (j(6) and a(6)) xor
(j(7) and a(7)) xor (j(1) and a(7)) xor (j(7) and a(1)) xor (j(2) and
a(6)) xor (j(6) and a(2)) xor (j(3) and a(5)) xor (j(5) and a(3)) xor
(j(5) and a(7)) xor (j(7) and a(5)) xor (j(6) and a(7)) xor (j(7) and
a(6));
   end if;
 ---------------------------------------------------------------------
   if r = '1' then
    mas(7) := (m(6) and k(6)) xor (m(0) and k(7)) xor (m(7) and k(0))
xor (m(1) and k(6)) xor (m(6) and k(1)) xor (m(2) and k(5)) xor (m(5)
```

```
and k(2)) xor (m(3) and k(4)) xor (m(4) and k(3)) xor (m(4) and k(7))
xor (m(7) and k(4)) xor (m(5) and k(6)) xor (m(6) and k(5)) xor (m(5)
and k(7)) xor (m(7) and k(5)) xor (m(6) and k(7)) xor (m(7) and k(6));
    mas(6) := (m(3) and k(3)) xor (m(5) and k(5)) xor (m(6) and k(6))
xor (m(0) and k(6)) xor (m(6) and k(0)) xor (m(1) and k(5)) xor (m(5)
and k(1)) xor (m(2) and k(4)) xor (m(4) and k(2)) xor (m(3) and k(7))
xor (m(7) and k(3)) xor (m(4) and k(6)) xor (m(6) and k(4)) xor (m(4)
and k(7)) xor (m(7) and k(4)) xor (m(5) and k(6)) xor (m(6) and k(5))
xor (m(5) and k(7)) xor (m(7) and k(5));
    mas(5) := (m(5) and k(5)) xor (m(0) and k(5)) xor (m(5) and k(0))
xor (m(1) and k(4)) xor (m(4) and k(1)) xor (m(2) and k(3)) xor (m(3)
and k(2)) xor (m(2) and k(7)) xor (m(7) and k(2)) xor (m(3) and k(6))
xor (m(6) and k(3)) xor (m(4) and k(5)) xor (m(5) and k(4)) xor (m(3)
and k(7)) xor (m(7) and k(3)) xor (m(4) and k(6)) xor (m(6) and k(4))
xor (m(4) and k(7)) xor (m(7) and k(4)) xor (m(5) and k(6)) xor (m(6)
and k(5));
    mas(4) := (m(2) and k(2)) xor (m(4) and k(4)) xor (m(5) and k(5))
xor (m(7) and k(7)) xor (m(0) and k(4)) xor (m(4) and k(0)) xor (m(1)
and k(3)) xor (m(3) and k(1)) xor (m(1) and k(7)) xor (m(7) and k(1))
xor (m(2) and k(6)) xor (m(6) and k(2)) xor (m(3) and k(5)) xor (m(5)
and k(3)) xor (m(2) and k(7)) xor (m(7) and k(2)) xor (m(3) and k(6))
xor (m(6) and k(3)) xor (m(4) and k(5)) xor (m(5) and k(4)) xor (m(3)
and k(7)) xor (m(7) and k(3)) xor (m(4) and k(6)) xor (m(6) and k(4));
    mas(3) := (m(4) and k(4)) xor (m(6) and k(6)) xor (m(0) and k(3))
xor (m(3) and k(0)) xor (m(1) and k(2)) xor (m(2) and k(1)) xor (m(1)
and k(7)) xor (m(7) and k(1)) xor (m(2) and k(6)) xor (m(6) and k(2))
xor (m(3) and k(5)) xor (m(5) and k(3)) xor (m(2) and k(7)) xor (m(7)
and k(2)) xor (m(3) and k(6)) xor (m(6) and k(3)) xor (m(4) and k(5))
xor (m(5) and k(4)) xor (m(4) and k(7)) xor (m(7) and k(4)) xor (m(5)
and k(6)) xor (m(6) and k(5)) xor (m(5) and k(7)) xor (m(7) and k(5));
    mas(2) := (m(1) and k(1)) xor (m(4) and k(4)) xor (m(5) and k(5))
xor (m(6) and k(6)) xor (m(0) and k(2)) xor (m(2) and k(0)) xor (m(1)
and k(7)) xor (m(7) and k(1)) xor (m(2) and k(6)) xor (m(6) and k(2))
xor (m(3) and k(5)) xor (m(5) and k(3)) xor (m(3) and k(7)) xor (m(7)
and k(3)) xor (m(4) and k(6)) xor (m(6) and k(4)) xor (m(5) and k(7))
xor (m(7) and k(5)) xor (m(6) and k(7)) xor (m(7) and k(6));
    mas(1) := (m(7) and k(7)) xor (m(0) and k(1)) xor (m(1) and k(0))
xor (m(2) and k(7)) xor (m(7) and k(2)) xor (m(3) and k(6)) xor (m(6)
and k(3)) xor (m(4) and k(5)) xor (m(5) and k(4)) xor (m(6) and k(7))
xor (m(7) and k(6));
    mas(0) := (m(0) and k(0)) xor (m(4) and k(4)) xor (m(6) and k(6))
xor (m(7) and k(7)) xor (m(1) and k(7)) xor (m(7) and k(1)) xor (m(2)
and k(6)) xor (m(6) and k(2)) xor (m(3) and k(5)) xor (m(5) and k(3))
xor (m(5) and k(7)) xor (m(7) and k(5)) xor (m(6) and k(7)) xor (m(7)
and k(6));
   end if;
    massage <= mas;
 end process;
end behave;
```

โปรแกรมการถอดรหัสลับ

```
library ieee;
use ieee.std_logic_1164.all;
entity decryption_0 is
port(  r : in std_logic;
       a,m : in std_logic_vector(7 downto 0);
       massage :out std_logic_vector(7 downto 0));
end decryption_0;
architecture behave of decryption_0 is
  signal l : std_logic_vector(7 downto 0);
begin
process
  variable b : std_logic_vector(7 downto 0);
  variable c : std_logic_vector(7 downto 0);
  variable d : std_logic_vector(7 downto 0);
```

```
  variable j : std_logic_vector(7 downto 0);
  variable k : std_logic_vector(7 downto 0);
  variable mas:std_logic_vector(7 downto 0);
 begin
   if r = '1' then --2
    b(7) := (a(6) and a(6)) xor (a(0) and a(7)) xor (a(7) and a(0)) xor
(a(1) and a(6)) xor (a(6) and a(1)) xor (a(2) and a(5)) xor (a(5) and
a(2)) xor (a(3) and a(4)) xor (a(4) and a(3)) xor (a(4) and a(7)) xor
(a(7) and a(4)) xor (a(5) and a(6)) xor (a(6) and a(5)) xor (a(5) and
a(7)) xor (a(7) and a(5)) xor (a(6) and a(7)) xor (a(7) and a(6));
    b(6) := (a(3) and a(3)) xor (a(5) and a(5)) xor (a(6) and a(6)) xor
(a(0) and a(6)) xor (a(6) and a(0)) xor (a(1) and a(5)) xor (a(5) and
a(1)) xor (a(2) and a(4)) xor (a(4) and a(2)) xor (a(3) and a(7)) xor
(a(7) and a(3)) xor (a(4) and a(6)) xor (a(6) and a(4)) xor (a(4) and
a(7)) xor (a(7) and a(4)) xor (a(5) and a(6)) xor (a(6) and a(5)) xor
(a(5) and a(7)) xor (a(7) and a(5));
    b(5) := (a(5) and a(5)) xor (a(0) and a(5)) xor (a(5) and a(0)) xor
(a(1) and a(4)) xor (a(4) and a(1)) xor (a(2) and a(3)) xor (a(3) and
a(2)) xor (a(2) and a(7)) xor (a(7) and a(2)) xor (a(3) and a(6)) xor
(a(6) and a(3)) xor (a(4) and a(5)) xor (a(5) and a(4)) xor (a(3) and
a(7)) xor (a(7) and a(3)) xor (a(4) and a(6)) xor (a(6) and a(4)) xor
(a(4) and a(7)) xor (a(7) and a(4)) xor (a(5) and a(6)) xor (a(6) and
a(5));
    b(4) := (a(2) and a(2)) xor (a(4) and a(4)) xor (a(5) and a(5)) xor
(a(7) and a(7)) xor (a(0) and a(4)) xor (a(4) and a(0)) xor (a(1) and
a(3)) xor (a(3) and a(1)) xor (a(1) and a(7)) xor (a(7) and a(1)) xor
(a(2) and a(6)) xor (a(6) and a(2)) xor (a(3) and a(5)) xor (a(5) and
a(3)) xor (a(2) and a(7)) xor (a(7) and a(2)) xor (a(3) and a(6)) xor
(a(6) and a(3)) xor (a(4) and a(5)) xor (a(5) and a(4)) xor (a(3) and
a(7)) xor (a(7) and a(3)) xor (a(4) and a(6)) xor (a(6) and a(4));
    b(3) := (a(4) and a(4)) xor (a(6) and a(6)) xor (a(0) and a(3)) xor
(a(3) and a(0)) xor (a(1) and a(2)) xor (a(2) and a(1)) xor (a(1) and
a(7)) xor (a(7) and a(1)) xor (a(2) and a(6)) xor (a(6) and a(2)) xor
(a(3) and a(5)) xor (a(5) and a(3)) xor (a(2) and a(7)) xor (a(7) and
a(2)) xor (a(3) and a(6)) xor (a(6) and a(3)) xor (a(4) and a(5)) xor
(a(5) and a(4)) xor (a(4) and a(7)) xor (a(7) and a(4)) xor (a(5) and
a(6)) xor (a(6) and a(5)) xor (a(5) and a(7)) xor (a(7) and a(5));
    b(2) := (a(1) and a(1)) xor (a(4) and a(4)) xor (a(5) and a(5)) xor
(a(6) and a(6)) xor (a(0) and a(2)) xor (a(2) and a(0)) xor (a(1) and
a(7)) xor (a(7) and a(1)) xor (a(2) and a(6)) xor (a(6) and a(2)) xor
(a(3) and a(5)) xor (a(5) and a(3)) xor (a(3) and a(7)) xor (a(7) and
a(3)) xor (a(4) and a(6)) xor (a(6) and a(4)) xor (a(5) and a(7)) xor
(a(7) and a(5)) xor (a(6) and a(7)) xor (a(7) and a(6));
    b(1) := (a(7) and a(7)) xor (a(0) and a(1)) xor (a(1) and a(0)) xor
(a(2) and a(7)) xor (a(7) and a(2)) xor (a(3) and a(6)) xor (a(6) and
a(3)) xor (a(4) and a(5)) xor (a(5) and a(4)) xor (a(6) and a(7)) xor
(a(7) and a(6));
    b(0) := (a(0) and a(0)) xor (a(4) and a(4)) xor (a(6) and a(6)) xor
(a(7) and a(7)) xor (a(1) and a(7)) xor (a(7) and a(1)) xor (a(2) and
a(6)) xor (a(6) and a(2)) xor (a(3) and a(5)) xor (a(5) and a(3)) xor
(a(5) and a(7)) xor (a(7) and a(5)) xor (a(6) and a(7)) xor (a(7) and
b(6));
   end if;
   if r = '1' then --4
    c(7) := (b(6) and b(6)) xor (b(0) and b(7)) xor (b(7) and b(0)) xor
(b(1) and b(6)) xor (b(6) and b(1)) xor (b(2) and b(5)) xor (b(5) and
b(2)) xor (b(3) and b(4)) xor (b(4) and b(3)) xor (b(4) and b(7)) xor
(b(7) and b(4)) xor (b(5) and b(6)) xor (b(6) and b(5)) xor (b(5) and
b(7)) xor (b(7) and b(5)) xor (b(6) and b(7)) xor (b(7) and b(6));
    c(6) := (b(3) and b(3)) xor (b(5) and b(5)) xor (b(6) and b(6)) xor
(b(0) and b(6)) xor (b(6) and b(0)) xor (b(1) and b(5)) xor (b(5) and
b(1)) xor (b(2) and b(4)) xor (b(4) and b(2)) xor (b(3) and b(7)) xor
(b(7) and b(3)) xor (b(4) and b(6)) xor (b(6) and b(4)) xor (b(4) and
b(7)) xor (b(7) and b(4)) xor (b(5) and b(6)) xor (b(6) and b(5)) xor
(b(5) and b(7)) xor (b(7) and b(5));
    c(5) := (b(5) and b(5)) xor (b(0) and b(5)) xor (b(5) and b(0)) xor
(b(1) and b(4)) xor (b(4) and b(1)) xor (b(2) and b(3)) xor (b(3) and
b(2)) xor (b(2) and b(7)) xor (b(7) and b(2)) xor (b(3) and b(6)) xor
(b(6) and b(3)) xor (b(4) and b(5)) xor (b(5) and b(4)) xor (b(3) and
```

```
b(7)) xor (b(7) and b(3)) xor (b(4) and b(6)) xor (b(6) and b(4)) xor
(b(4) and b(7)) xor (b(7) and b(4)) xor (b(5) and b(6)) xor (b(6) and
b(5));
    c(4) := (b(2) and b(2)) xor (b(4) and b(4)) xor (b(5) and b(5)) xor
(b(7) and b(7)) xor (b(0) and b(4)) xor (b(4) and b(0)) xor (b(1) and
b(3)) xor (b(3) and b(1)) xor (b(1) and b(7)) xor (b(7) and b(1)) xor
(b(2) and b(6)) xor (b(6) and b(2)) xor (b(3) and b(5)) xor (b(5) and
b(3)) xor (b(2) and b(7)) xor (b(7) and b(2)) xor (b(3) and b(6)) xor
(b(6) and b(3)) xor (b(4) and b(5)) xor (b(5) and b(4)) xor (b(3) and
b(7)) xor (b(7) and b(3)) xor (b(4) and b(6)) xor (b(6) and b(4));
    c(3) := (b(4) and b(4)) xor (b(6) and b(6)) xor (b(0) and b(3)) xor
(b(3) and b(0)) xor (b(1) and b(2)) xor (b(2) and b(1)) xor (b(1) and
b(7)) xor (b(7) and b(1)) xor (b(2) and b(6)) xor (b(6) and b(2)) xor
(b(3) and b(5)) xor (b(5) and b(3)) xor (b(2) and b(7)) xor (b(7) and
b(2)) xor (b(3) and b(6)) xor (b(6) and b(3)) xor (b(4) and b(5)) xor
(b(5) and b(4)) xor (b(4) and b(7)) xor (b(7) and b(4)) xor (b(5) and
b(6)) xor (b(6) and b(5)) xor (b(5) and b(7)) xor (b(7) and b(5));
    c(2) := (b(1) and b(1)) xor (b(4) and b(4)) xor (b(5) and b(5)) xor
(b(6) and b(6)) xor (b(0) and b(2)) xor (b(2) and b(0)) xor (b(1) and
b(7)) xor (b(7) and b(1)) xor (b(2) and b(6)) xor (b(6) and b(2)) xor
(b(3) and b(5)) xor (b(5) and b(3)) xor (b(3) and b(7)) xor (b(7) and
b(3)) xor (b(4) and b(6)) xor (b(6) and b(4)) xor (b(5) and b(7)) xor
(b(7) and b(5)) xor (b(6) and b(7)) xor (b(7) and b(6));
    c(1) := (b(7) and b(7)) xor (b(0) and b(1)) xor (b(1) and b(0)) xor
(b(2) and b(7)) xor (b(7) and b(2)) xor (b(3) and b(6)) xor (b(6) and
b(3)) xor (b(4) and b(5)) xor (b(5) and b(4)) xor (b(6) and b(7)) xor
(b(7) and b(6));
    c(0) := (b(0) and b(0)) xor (b(4) and b(4)) xor (b(6) and b(6)) xor
(b(7) and b(7)) xor (b(1) and b(7)) xor (b(7) and b(1)) xor (b(2) and
b(6)) xor (b(6) and b(2)) xor (b(3) and b(5)) xor (b(5) and b(3)) xor
b(5) and b(7)) xor (b(7) and b(5)) xor (b(6) and b(7)) xor (b(7) and
b(6));
   end if;
   if r = '1' then --8
    d(7) := (c(6) and c(6)) xor (c(0) and c(7)) xor (c(7) and c(0)) xor
(c(1) and c(6)) xor (c(6) and c(1)) xor (c(2) and c(5)) xor (c(5) and
c(2)) xor (c(3) and c(4)) xor (c(4) and c(3)) xor (c(4) and c(7)) xor
(c(7) and c(4)) xor (c(5) and c(6)) xor (c(6) and c(5)) xor (c(5) and
c(7)) xor (c(7) and c(5)) xor (c(6) and c(7)) xor (c(7) and c(6));
    d(6) := (c(3) and c(3)) xor (c(5) and c(5)) xor (c(6) and c(6)) xor
(c(0) and c(6)) xor (c(6) and c(0)) xor (c(1) and c(5)) xor (c(5) and
c(1)) xor (c(2) and c(4)) xor (c(4) and c(2)) xor (c(3) and c(7)) xor
(c(7) and c(3)) xor (c(4) and c(6)) xor (c(6) and c(4)) xor (c(4) and
c(7)) xor (c(7) and c(4)) xor (c(5) and c(6)) xor (c(6) and c(5)) xor
(c(5) and c(7)) xor (c(7) and c(5));
    d(5) := (c(5) and c(5)) xor (c(0) and c(5)) xor (c(5) and c(0)) xor
(c(1) and c(4)) xor (c(4) and c(1)) xor (c(2) and c(3)) xor (c(3) and
c(2)) xor (c(2) and c(7)) xor (c(7) and c(2)) xor (c(3) and c(6)) xor
(c(6) and c(3)) xor (c(4) and c(5)) xor (c(5) and c(4)) xor (c(3) and
c(7)) xor (c(7) and c(3)) xor (c(4) and c(6)) xor (c(6) and c(4)) xor
(c(4) and c(7)) xor (c(7) and c(4)) xor (c(5) and c(6)) xor (c(6) and
c(5));
    d(4) := (c(2) and c(2)) xor (c(4) and c(4)) xor (c(5) and c(5)) xor
(c(7) and c(7)) xor (c(0) and c(4)) xor (c(4) and c(0)) xor (c(1) and
c(3)) xor (c(3) and c(1)) xor (c(1) and c(7)) xor (c(7) and c(1)) xor
(c(2) and c(6)) xor (c(6) and c(2)) xor (c(3) and c(5)) xor (c(5) and
c(3)) xor (c(2) and c(7)) xor (c(7) and c(2)) xor (c(3) and c(6)) xor
(c(6) and c(3)) xor (c(4) and c(5)) xor (c(5) and c(4)) xor (c(3) and
c(7)) xor (c(7) and c(3)) xor (c(4) and c(6)) xor (c(6) and c(4));
    d(3) := (c(4) and c(4)) xor (c(6) and c(6)) xor (c(0) and c(3)) xor
(c(3) and c(0)) xor (c(1) and c(2)) xor (c(2) and c(1)) xor (c(1) and
c(7)) xor (c(7) and c(1)) xor (c(2) and c(6)) xor (c(6) and c(2)) xor
(c(3) and c(5)) xor (c(5) and c(3)) xor (c(2) and c(7)) xor (c(7) and
c(2)) xor (c(3) and c(6)) xor (c(6) and c(3)) xor (c(4) and c(5)) xor
(c(5) and c(4)) xor (c(4) and c(7)) xor (c(7) and c(4)) xor (c(5) and
c(6)) xor (c(6) and c(5)) xor (c(5) and c(7)) xor (c(7) and c(5));
    d(2) := (c(1) and c(1)) xor (c(4) and c(4)) xor (c(5) and c(5)) xor
(c(6) and c(6)) xor (c(0) and c(2)) xor (c(2) and c(0)) xor (c(1) and
c(7)) xor (c(7) and c(1)) xor (c(2) and c(6)) xor (c(6) and c(2)) xor
```

```
(c(3) and c(5)) xor (c(5) and c(3)) xor (c(3) and c(7)) xor (c(7) and
c(3)) xor (c(4) and c(6)) xor (c(6) and c(4)) xor (c(5) and c(7)) xor
(c(7) and c(5)) xor (c(6) and c(7)) xor (c(7) and c(6));
    d(1) := (c(7) and c(7)) xor (c(0) and c(1)) xor (c(1) and c(0)) xor
(c(2) and c(7)) xor (c(7) and c(2)) xor (c(3) and c(6)) xor (c(6) and
c(3)) xor (c(4) and c(5)) xor (c(5) and c(4)) xor (c(6) and c(7)) xor
(c(7) and c(6));
    d(0) := (c(0) and c(0)) xor (c(4) and c(4)) xor (c(6) and c(6)) xor
(c(7) and c(7)) xor (c(1) and c(7)) xor (c(7) and c(1)) xor (c(2) and
c(6)) xor (c(6) and c(2)) xor (c(3) and c(5)) xor (c(5) and c(3)) xor
(c(5) and c(7)) xor (c(7) and c(5)) xor (c(6) and c(7)) xor (c(7) and
c(6));
   end if;
   if r = '1' then --12
    j(7) := (d(6) and c(6)) xor (d(0) and c(7)) xor (d(7) and c(0)) xor
(d(1) and c(6)) xor (d(6) and c(1)) xor (d(2) and c(5)) xor (d(5) and
c(2)) xor (d(3) and c(4)) xor (d(4) and c(3)) xor (d(4) and c(7)) xor
(d(7) and c(4)) xor (d(5) and c(6)) xor (d(6) and c(5)) xor (d(5) and
c(7)) xor (d(7) and c(5)) xor (d(6) and c(7)) xor (d(7) and c(6));
    j(6) := (d(3) and c(3)) xor (d(5) and c(5)) xor (d(6) and c(6)) xor
(d(0) and c(6)) xor (d(6) and c(0)) xor (d(1) and c(5)) xor (d(5) and
c(1)) xor (d(2) and c(4)) xor (d(4) and c(2)) xor (d(3) and c(7)) xor
(d(7) and c(3)) xor (d(4) and c(6)) xor (d(6) and c(4)) xor (d(4) and
c(7)) xor (d(7) and c(4)) xor (d(5) and c(6)) xor (d(6) and c(5)) xor
(d(5) and c(7)) xor (d(7) and c(5));
    j(5) := (d(5) and c(5)) xor (d(0) and c(5)) xor (d(5) and c(0)) xor
(d(1) and c(4)) xor (d(4) and c(1)) xor (d(2) and c(3)) xor (d(3) and
c(2)) xor (d(2) and c(7)) xor (d(7) and c(2)) xor (d(3) and c(6)) xor
(d(6) and c(3)) xor (d(4) and c(5)) xor (d(5) and c(4)) xor (d(3) and
c(7)) xor (d(7) and c(3)) xor (d(4) and c(6)) xor (d(6) and c(4)) xor
(d(4) and c(7)) xor (d(7) and c(4)) xor (d(5) and c(6)) xor (d(6) and
c(5));
    j(4) := (d(2) and c(2)) xor (d(4) and c(4)) xor (d(5) and c(5)) xor
(d(7) and c(7)) xor (d(0) and c(4)) xor (d(4) and c(0)) xor (d(1) and
c(3)) xor (d(3) and c(1)) xor (d(1) and c(7)) xor (d(7) and c(1)) xor
(d(2) and c(6)) xor (d(6) and c(2)) xor (d(3) and c(5)) xor (d(5) and
c(3)) xor (d(2) and c(7)) xor (d(7) and c(2)) xor (d(3) and c(6)) xor
(d(6) and c(3)) xor (d(4) and c(5)) xor (d(5) and c(4)) xor (d(3) and
c(7)) xor (d(7) and c(3)) xor (d(4) and c(6)) xor (d(6) and c(4));
    j(3) := (d(4) and c(4)) xor (d(6) and c(6)) xor (d(0) and c(3)) xor
(d(3) and c(0)) xor (d(1) and c(2)) xor (d(2) and c(1)) xor (d(1) and
c(7)) xor (d(7) and c(1)) xor (d(2) and c(6)) xor (d(6) and c(2)) xor
(d(3) and c(5)) xor (d(5) and c(3)) xor (d(2) and c(7)) xor (d(7) and
c(2)) xor (d(3) and c(6)) xor (d(6) and c(3)) xor (d(4) and c(5)) xor
(d(5) and c(4)) xor (d(4) and c(7)) xor (d(7) and c(4)) xor (d(5) and
c(6)) xor (d(6) and c(5)) xor (d(5) and c(7)) xor (d(7) and c(5));
    j(2) := (d(1) and c(1)) xor (d(4) and c(4)) xor (d(5) and c(5)) xor
(d(6) and c(6)) xor (d(0) and c(2)) xor (d(2) and c(0)) xor (d(1) and
c(7)) xor (d(7) and c(1)) xor (d(2) and c(6)) xor (d(6) and c(2)) xor
(d(3) and c(5)) xor (d(5) and c(3)) xor (d(3) and c(7)) xor (d(7) and
c(3)) xor (d(4) and c(6)) xor (d(6) and c(4)) xor (d(5) and c(7)) xor
(d(7) and c(5)) xor (d(6) and c(7)) xor (d(7) and c(6));
    j(1) := (d(7) and c(7)) xor (d(0) and c(1)) xor (d(1) and c(0)) xor
(d(2) and c(7)) xor (d(7) and c(2)) xor (d(3) and c(6)) xor (d(6) and
c(3)) xor (d(4) and c(5)) xor (d(5) and c(4)) xor (d(6) and c(7)) xor
(d(7) and c(6));
    j(0) := (d(0) and c(0)) xor (d(4) and c(4)) xor (d(6) and c(6)) xor
(d(7) and c(7)) xor (d(1) and c(7)) xor (d(7) and c(1)) xor (d(2) and
c(6)) xor (d(6) and c(2)) xor (d(3) and c(5)) xor (d(5) and c(3)) xor
(d(5) and c(7)) xor (d(7) and c(5)) xor (d(6) and c(7)) xor (d(7) and
c(6));
   end if;
   if r = '1' then --13
    k(7) := (j(6) and a(6)) xor (j(0) and a(7)) xor (j(7) and a(0)) xor
(j(1) and a(6)) xor (j(6) and a(1)) xor (j(2) and a(5)) xor (j(5) and
a(2)) xor (j(3) and a(4)) xor (j(4) and a(3)) xor (j(4) and a(7)) xor
(j(7) and a(4)) xor (j(5) and a(6)) xor (j(6) and a(5)) xor (j(5) and
a(7)) xor (j(7) and a(5)) xor (j(6) and a(7)) xor (j(7) and a(6));
```

```
    k(6) := (j(3) and a(3)) xor (j(5) and a(5)) xor (j(6) and a(6)) xor
(j(0) and a(6)) xor (j(6) and a(0)) xor (j(1) and a(5)) xor (j(5) and
a(1)) xor (j(2) and a(4)) xor (j(4) and a(2)) xor (j(3) and a(7)) xor
(j(7) and a(3)) xor (j(4) and a(6)) xor (j(6) and a(4)) xor (j(4) and
a(7)) xor (j(7) and a(4)) xor (j(5) and a(6)) xor (j(6) and a(5)) xor
(j(5) and a(7)) xor (j(7) and a(5));
    k(5) := (j(5) and a(5)) xor (j(0) and a(5)) xor (j(5) and a(0)) xor
(j(1) and a(4)) xor (j(4) and a(1)) xor (j(2) and a(3)) xor (j(3) and
a(2)) xor (j(2) and a(7)) xor (j(7) and a(2)) xor (j(3) and a(6)) xor
(j(6) and a(3)) xor (j(4) and a(5)) xor (j(5) and a(4)) xor (j(3) and
a(7)) xor (j(7) and a(3)) xor (j(4) and a(6)) xor (j(6) and a(4)) xor
(j(4) and a(7)) xor (j(7) and a(4)) xor (j(5) and a(6)) xor (j(6) and
a(5));
    k(4) := (j(2) and a(2)) xor (j(4) and a(4)) xor (j(5) and a(5)) xor
(j(7) and a(7)) xor (j(0) and a(4)) xor (j(4) and a(0)) xor (j(1) and
a(3)) xor (j(3) and a(1)) xor (j(1) and a(7)) xor (j(7) and a(1)) xor
(j(2) and a(6)) xor (j(6) and a(2)) xor (j(3) and a(5)) xor (j(5) and
a(3)) xor (j(2) and a(7)) xor (j(7) and a(2)) xor (j(3) and a(6)) xor
(j(6) and a(3)) xor (j(4) and a(5)) xor (j(5) and a(4)) xor (j(3) and
a(7)) xor (j(7) and a(3)) xor (j(4) and a(6)) xor (j(6) and a(4));
    k(3) := (j(4) and a(4)) xor (j(6) and a(6)) xor (j(0) and a(3)) xor
(j(3) and a(0)) xor (j(1) and a(2)) xor (j(2) and a(1)) xor (j(1) and
a(7)) xor (j(7) and a(1)) xor (j(2) and a(6)) xor (j(6) and a(2)) xor
(j(3) and a(5)) xor (j(5) and a(3)) xor (j(2) and a(7)) xor (j(7) and
a(2)) xor (j(3) and a(6)) xor (j(6) and a(3)) xor (j(4) and a(5)) xor
(j(5) and a(4)) xor (j(4) and a(7)) xor (j(7) and a(4)) xor (j(5) and
a(6)) xor (j(6) and a(5)) xor (j(5) and a(7)) xor (j(7) and a(5));
    k(2) := (j(1) and a(1)) xor (j(4) and a(4)) xor (j(5) and a(5)) xor
(j(6) and a(6)) xor (j(0) and a(2)) xor (j(2) and a(0)) xor (j(1) and
a(7)) xor (j(7) and a(1)) xor (j(2) and a(6)) xor (j(6) and a(2)) xor
(j(3) and a(5)) xor (j(5) and a(3)) xor (j(3) and a(7)) xor (j(7) and
a(3)) xor (j(4) and a(6)) xor (j(6) and a(4)) xor (j(5) and a(7)) xor
(j(7) and a(5)) xor (j(6) and a(7)) xor (j(7) and a(6));
    k(1) := (j(7) and a(7)) xor (j(0) and a(1)) xor (j(1) and a(0)) xor
(j(2) and a(7)) xor (j(7) and a(2)) xor (j(3) and a(6)) xor (j(6) and
a(3)) xor (j(4) and a(5)) xor (j(5) and a(4)) xor (j(6) and a(7)) xor
(j(7) and a(6));
    k(0) := (j(0) and a(0)) xor (j(4) and a(4)) xor (j(6) and a(6)) xor
(j(7) and a(7)) xor (j(1) and a(7)) xor (j(7) and a(1)) xor (j(2) and
a(6)) xor (j(6) and a(2)) xor (j(3) and a(5)) xor (j(5) and a(3)) xor
(j(5) and a(7)) xor (j(7) and a(5)) xor (j(6) and a(7)) xor (j(7) and
a(6));
   end if;
 ---------------------------------------------------------------------
   if r = '1' then
   case k is
      when "00000001" => l <= "00000001";
      when "00000010" => l <= "10001110";
      when "00000100" => l <= "01000111";
      when "00001000" => l <= "10101101";
      when "00010000" => l <= "11011000";
      when "00100000" => l <= "01101100";
      when "01000000" => l <= "00110110";
      when "10000000" => l <= "00011011";
      when "00011101" => l <= "10000011";
      when "00111010" => l <= "11001111";
      when "01110100" => l <= "11101001";
      when "11101000" => l <= "11111010";
      when "11001101" => l <= "01111101";
      when "10000111" => l <= "10110000";
      when "00010011" => l <= "01011000";
      when "00100110" => l <= "00101100";
      when "01001100" => l <= "00010110";
      when "10011000" => l <= "00001011";
      when "00101101" => l <= "10001011";
      when "01011010" => l <= "11001011";
      when "10110100" => l <= "11101011";
      when "01110101" => l <= "11111011";
      when "11101010" => l <= "11110011";
```

```
when "11001001" => l <= "11110111";
when "10001111" => l <= "11110101";
when "00000011" => l <= "11110100";
when "00000110" => l <= "01111010";
when "00001100" => l <= "00111101";
when "00011000" => l <= "10010000";
when "00110000" => l <= "01001000";
when "01100000" => l <= "00100100";
when "11000000" => l <= "00010010";
when "10011101" => l <= "00001001";
when "00100111" => l <= "10001010";
when "01001110" => l <= "01000101";
when "10011100" => l <= "10101100";
when "00100101" => l <= "01010110";
when "01001010" => l <= "00101011";
when "10010100" => l <= "10011011";
when "00110101" => l <= "11000011";
when "01101010" => l <= "11101111";
when "11010100" => l <= "11111001";
when "10110101" => l <= "11110010";
when "01110111" => l <= "01111001";
when "11101110" => l <= "10110010";
when "11000001" => l <= "01011001";
when "10011111" => l <= "10100010";
when "00100011" => l <= "01010001";
when "01000110" => l <= "10100110";
when "10001100" => l <= "01010011";
when "00000101" => l <= "10100111";
when "00001010" => l <= "11011101";
when "00010100" => l <= "11100000";
when "00101000" => l <= "01110000";
when "01010000" => l <= "00111000";
when "10100000" => l <= "00011100";
when "01011101" => l <= "00001110";
when "10111010" => l <= "00000111";
when "01101001" => l <= "10001101";
when "11010010" => l <= "11001000";
when "10111001" => l <= "01100100";
when "01101111" => l <= "00110010";
when "11011110" => l <= "00011001";
when "10100001" => l <= "10000010";
when "01011111" => l <= "01000001";
when "10111110" => l <= "10101110";
when "01100001" => l <= "01010111";
when "11000010" => l <= "10100101";
when "10011001" => l <= "11011100";
when "00101111" => l <= "01101110";
when "01011110" => l <= "00110111";
when "10111100" => l <= "10010101";
when "01100101" => l <= "11000100";
when "11001010" => l <= "01100010";
when "10001001" => l <= "00110001";
when "00001111" => l <= "10010110";
when "00011110" => l <= "01001011";
when "00111100" => l <= "10101011";
when "01111000" => l <= "11011011";
when "11110000" => l <= "11100011";
when "11111101" => l <= "11111111";
when "11100111" => l <= "11110001";
when "11010011" => l <= "11110110";
when "10111011" => l <= "01111011";
when "01101011" => l <= "10110011";
when "11010110" => l <= "11010111";
when "10110001" => l <= "11100101";
when "01111111" => l <= "11111100";
when "11111110" => l <= "01111110";
when "11100001" => l <= "00111111";
when "11011111" => l <= "10010001";
when "10100011" => l <= "11000110";
```

```
when "01011011" => l <= "01100011";
when "10110110" => l <= "10111111";
when "01110001" => l <= "11010001";
when "11100010" => l <= "11100110";
when "11011001" => l <= "01110011";
when "10101111" => l <= "10110111";
when "01000011" => l <= "11010101";
when "10000110" => l <= "11100100";
when "00010001" => l <= "01110010";
when "00100010" => l <= "00111000";
when "01000100" => l <= "10010010";
when "10001000" => l <= "01001001";
when "00001101" => l <= "10101010";
when "00011010" => l <= "01010101";
when "00110100" => l <= "10100100";
when "01101000" => l <= "01010010";
when "11010000" => l <= "00101001";
when "10111101" => l <= "10011010";
when "01100111" => l <= "01001101";
when "11001110" => l <= "10101000";
when "10000001" => l <= "01010100";
when "00011111" => l <= "00101010";
when "00111110" => l <= "00010101";
when "01111100" => l <= "10000100";
when "11111000" => l <= "01000010";
when "11101101" => l <= "00100001";
when "11000111" => l <= "10011110";
when "10010011" => l <= "01001111";
when "00111011" => l <= "10101001";
when "01110110" => l <= "11011010";
when "11101100" => l <= "01101101";
when "11000101" => l <= "10111000";
when "10010111" => l <= "01011100";
when "00110011" => l <= "00101110";
when "01100110" => l <= "00010111";
when "11001100" => l <= "10000101";
when "10000101" => l <= "11001100";
when "00010111" => l <= "01100110";
when "00101110" => l <= "00110011";
when "01011100" => l <= "10010111";
when "10111000" => l <= "11000101";
when "01101101" => l <= "11101100";
when "11011010" => l <= "01110110";
when "10101001" => l <= "00111011";
when "01001111" => l <= "10010011";
when "10011110" => l <= "11000111";
when "00100001" => l <= "11101101";
when "01000010" => l <= "11111000";
when "10000100" => l <= "01111100";
when "00010101" => l <= "00111110";
when "00101010" => l <= "00011111";
when "01010100" => l <= "10000001";
when "10101000" => l <= "11001110";
when "01001101" => l <= "01100111";
when "10011010" => l <= "10111101";
when "00101001" => l <= "11010000";
when "01010010" => l <= "01101000";
when "10100100" => l <= "00110100";
when "01010101" => l <= "00011010";
when "10101010" => l <= "00001101";
when "01001001" => l <= "10001000";
when "10010010" => l <= "01000100";
when "00111001" => l <= "00100010";
when "01110010" => l <= "00010001";
when "11100100" => l <= "10000110";
when "11010101" => l <= "01000011";
when "10110111" => l <= "10101111";
when "01110011" => l <= "11011001";
when "11100110" => l <= "11100010";
```

```
when "11010001" => l <= "01110001";
when "10111111" => l <= "10110110";
when "01100011" => l <= "01011011";
when "11000110" => l <= "10100011";
when "10010001" => l <= "11011111";
when "00111111" => l <= "11100001";
when "01111110" => l <= "11111110";
when "11111100" => l <= "01111111";
when "11100101" => l <= "10110001";
when "11010111" => l <= "11010110";
when "10110011" => l <= "01101011";
when "01111011" => l <= "10111011";
when "11110110" => l <= "11010011";
when "11110001" => l <= "11100111";
when "11111111" => l <= "11111101";
when "11100011" => l <= "11110000";
when "11011011" => l <= "01111000";
when "10101011" => l <= "00111100";
when "01001011" => l <= "00011110";
when "10010110" => l <= "00001111";
when "00110001" => l <= "10001001";
when "01100010" => l <= "11001010";
when "11000100" => l <= "01100101";
when "10010101" => l <= "10111100";
when "00110111" => l <= "01011110";
when "01101110" => l <= "00101111";
when "11011100" => l <= "10011001";
when "10100101" => l <= "11000010";
when "01010111" => l <= "01100001";
when "10101110" => l <= "10111110";
when "01000001" => l <= "01011111";
when "10000010" => l <= "10100001";
when "00011001" => l <= "11011110";
when "00110010" => l <= "01101111";
when "01100100" => l <= "10111001";
when "11001000" => l <= "11010010";
when "10001101" => l <= "01101001";
when "00000111" => l <= "10111010";
when "00001110" => l <= "01011101";
when "00011100" => l <= "10100000";
when "00111000" => l <= "01010000";
when "01110000" => l <= "00101000";
when "11100000" => l <= "00010100";
when "11011101" => l <= "00001010";
when "10100111" => l <= "00000101";
when "01010011" => l <= "10001100";
when "10100110" => l <= "01000110";
when "01010001" => l <= "00100011";
when "10100010" => l <= "10011111";
when "01011001" => l <= "11000001";
when "10110010" => l <= "11101110";
when "01111001" => l <= "01110111";
when "11110010" => l <= "10110101";
when "11111001" => l <= "11010100";
when "11101111" => l <= "01101010";
when "11000011" => l <= "00110101";
when "10011011" => l <= "10010100";
when "00101011" => l <= "01001010";
when "01010110" => l <= "00100101";
when "10101100" => l <= "10011100";
when "01000101" => l <= "01001110";
when "10001010" => l <= "00100111";
when "00001001" => l <= "10011101";
when "00010010" => l <= "11000000";
when "00100100" => l <= "01100000";
when "01001000" => l <= "00110000";
when "10010000" => l <= "00011000";
when "00111101" => l <= "00001100";
when "01111010" => l <= "00000110";
```

```
      when "11110100" => l <= "00000011";
      when "11110101" => l <= "10001111";
      when "11110111" => l <= "11001001";
      when "11110011" => l <= "11101010";
      when "11111011" => l <= "01110101";
      when "11101011" => l <= "10110100";
      when "11001011" => l <= "01011010";
      when "10001011" => l <= "00101101";
      when "00001011" => l <= "10011000";
      when "00010110" => l <= "01001100";
      when "00101100" => l <= "00100110";
      when "01011000" => l <= "00010011";
      when "10110000" => l <= "10000111";
      when "01111101" => l <= "11001101";
      when "11111010" => l <= "11101000";
      when "11101001" => l <= "01110100";
      when "11001111" => l <= "00111010";
      when "10000011" => l <= "00011101";
      when "00011011" => l <= "10000000";
      when "00110110" => l <= "01000000";
      when "01101100" => l <= "00100000";
      when "11011000" => l <= "00010000";
      when "10101101" => l <= "00001000";
      when "01000111" => l <= "00000100";
      when "10001110" => l <= "00000010";
      when others => null;
   end case;
   end if;
-------------------------------------------------------
  if r = '1' then
    mas(7) := (m(6) and l(6)) xor (m(0) and l(7)) xor (m(7) and l(0))
xor (m(1) and l(6)) xor (m(6) and l(1)) xor (m(2) and l(5)) xor (m(5)
and l(2)) xor (m(3) and l(4)) xor (m(4) and l(3)) xor (m(4) and l(7))
xor (m(7) and l(4)) xor (m(5) and l(6)) xor (m(6) and l(5)) xor (m(5)
and l(7)) xor (m(7) and l(5)) xor (m(6) and l(7)) xor (m(7) and l(6));
    mas(6) := (m(3) and l(3)) xor (m(5) and l(5)) xor (m(6) and l(6))
xor (m(0) and l(6)) xor (m(6) and l(0)) xor (m(1) and l(5)) xor (m(5)
and l(1)) xor (m(2) and l(4)) xor (m(4) and l(2)) xor (m(3) and l(7))
xor (m(7) and l(3)) xor (m(4) and l(6)) xor (m(6) and l(4)) xor (m(4)
and l(7)) xor (m(7) and l(4)) xor (m(5) and l(6)) xor (m(6) and l(5))
xor (m(5) and l(7)) xor (m(7) and l(5));
    mas(5) := (m(5) and l(5)) xor (m(0) and l(5)) xor (m(5) and l(0))
xor (m(1) and l(4)) xor (m(4) and l(1)) xor (m(2) and l(3)) xor (m(3)
and l(2)) xor (m(2) and l(7)) xor (m(7) and l(2)) xor (m(3) and l(6))
xor (m(6) and l(3)) xor (m(4) and l(5)) xor (m(5) and l(4)) xor (m(3)
and l(7)) xor (m(7) and l(3)) xor (m(4) and l(6)) xor (m(6) and l(4))
xor (m(4) and l(7)) xor (m(7) and l(4)) xor (m(5) and l(6)) xor (m(6)
and l(5));
    mas(4) := (m(2) and l(2)) xor (m(4) and l(4)) xor (m(5) and l(5))
xor (m(7) and l(7)) xor (m(0) and l(4)) xor (m(4) and l(0)) xor (m(1)
and l(3)) xor (m(3) and l(1)) xor (m(1) and l(7)) xor (m(7) and l(1))
xor (m(2) and l(6)) xor (m(6) and l(2)) xor (m(3) and l(5)) xor (m(5)
and l(3)) xor (m(2) and l(7)) xor (m(7) and l(2)) xor (m(3) and l(6))
xor (m(6) and l(3)) xor (m(4) and l(5)) xor (m(5) and l(4)) xor (m(3)
and l(7)) xor (m(7) and l(3)) xor (m(4) and l(6)) xor (m(6) and l(4));
    mas(3) := (m(4) and l(4)) xor (m(6) and l(6)) xor (m(0) and l(3))
xor (m(3) and l(0)) xor (m(1) and l(2)) xor (m(2) and l(1)) xor (m(1)
and l(7)) xor (m(7) and l(1)) xor (m(2) and l(6)) xor (m(6) and l(2))
xor (m(3) and l(5)) xor (m(5) and l(3)) xor (m(2) and l(7)) xor (m(7)
and l(2)) xor (m(3) and l(6)) xor (m(6) and l(3)) xor (m(4) and l(5))
xor (m(5) and l(4)) xor (m(4) and l(7)) xor (m(7) and l(4)) xor (m(5)
and l(6)) xor (m(6) and l(5)) xor (m(5) and l(7)) xor (m(7) and l(5));
    mas(2) := (m(1) and l(1)) xor (m(4) and l(4)) xor (m(5) and l(5))
xor (m(6) and l(6)) xor (m(0) and l(2)) xor (m(2) and l(0)) xor (m(1)
and l(7)) xor (m(7) and l(1)) xor (m(2) and l(6)) xor (m(6) and l(2))
xor (m(3) and l(5)) xor (m(5) and l(3)) xor (m(3) and l(7)) xor (m(7)
and l(3)) xor (m(4) and l(6)) xor (m(6) and l(4)) xor (m(5) and l(7))
xor (m(7) and l(5)) xor (m(6) and l(7)) xor (m(7) and l(6));
```

```
    mas(1) := (m(7) and l(7)) xor (m(0) and l(1)) xor (m(1) and l(0))
xor (m(2) and l(7)) xor (m(7) and l(2)) xor (m(3) and l(6)) xor (m(6)
and l(3)) xor (m(4) and l(5)) xor (m(5) and l(4)) xor (m(6) and l(7))
xor (m(7) and l(6));
    mas(0) := (m(0) and l(0)) xor (m(4) and l(4)) xor (m(6) and l(6))
xor (m(7) and l(7)) xor (m(1) and l(7)) xor (m(7) and l(1)) xor (m(2)
and l(6)) xor (m(6) and l(2)) xor (m(3) and l(5)) xor (m(5) and l(3))
xor (m(5) and l(7)) xor (m(7) and l(5)) xor (m(6) and l(7)) xor (m(7)
and l(6));
   end if;
     massage <= mas;
 end process;
end behave;
```

โปรแกรม หารความถี่

```
library ieee;
use ieee.std_logic_1164.all;
-----------------------------------------------------
entity Div_000 is
port(
      Clk_in : in std_logic;
      Clk_out : out std_logic
           );
end Div_000;
-----------------------------------------------------
architecture rtl of Div_000 is
begin
      process(Clk_in)
      variable Clk_temp : std_logic := '0';
      variable count : integer range 0 to 4539 := 0;
      begin
      if Clk_in'Event and Clk_in = '1' then
                  if count < 4539 then
                        count := count + 1;
                        Clk_temp := Clk_temp;
                  else
                        count := 0;
                        Clk_temp := not(Clk_temp);
                  end if;
                  Clk_out <= Clk_temp;
            end if;
      end process;
end rtl;
```

โปรแกรม Control

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity CONTROL_ECC_5 is
 port( clk,input,reset : in std_logic;
       address,samping :out std_logic;
       samping_1,address_1,reset_1,reset_ram,encryp :out std_logic;
       samping_2,address_2,reset_2,decryp,tx_en,clk_out  :out std_logic;
       key_in,encryp_in, decryp_in : in std_logic_vector(7 downto 0);
       key_out,encryp_decryp :out std_logic_vector(7 downto 0)
       );
end CONTROL_ECC_5;
architecture rtl of CONTROL_ECC_5 is
```

```
 type state_type is
(s0,s1,s2,s3,s4,s5,s51,s52,s53,s54,s55,s56,s57,s58,s59,s591,s592,s593,s5
94,s595,s596,s6,s61,s62,s63,s64,s7,s8,s81,s82,s83,s84,s9);
 signal state : state_type;
 signal output_0 : std_logic;
 signal output_1 : std_logic;
 signal output_2 : std_logic;
 signal output_3 : std_logic;
 signal output_4 : std_logic;
 signal output_5 : std_logic;---------6
 signal output_51 : std_logic;
 signal output_52 : std_logic;
 signal output_53 : std_logic;
 signal output_54 : std_logic;
 signal output_55 : std_logic;
 signal output_56 : std_logic;
 signal output_57 : std_logic;
 signal output_58 : std_logic;
 signal output_59 : std_logic;--------15
 signal output_591 : std_logic;
 signal output_592 : std_logic;
 signal output_593 : std_logic;
 signal output_594 : std_logic;
 signal output_595 : std_logic;
 signal output_596 : std_logic;
 signal output_6 : std_logic;
 signal output_61 : std_logic;
 signal output_62 : std_logic;
 signal output_63 : std_logic;
 signal output_64 : std_logic;
 signal output_7 : std_logic;
 signal output_8 : std_logic;
 signal output_81 : std_logic;
 signal output_82 : std_logic;
 signal output_83 : std_logic;
 signal output_84 : std_logic;
 signal output_9 : std_logic;
 signal address_0 : std_logic;
 signal samping_0 : std_logic;
 signal we_0 : std_logic;
 signal reset_0 : std_logic;
 signal sum_0 : std_logic_vector(3 downto 0);
 signal sum_1 : std_logic_vector(3 downto 0);
 signal sum_2 : std_logic_vector(3 downto 0);
 signal sum_3 : std_logic_vector(3 downto 0);
 signal sum_4 : std_logic_vector(3 downto 0);
 signal sum_5 : std_logic_vector(3 downto 0);
 signal sum_51 : std_logic_vector(3 downto 0);
 signal sum_52 : std_logic_vector(3 downto 0);
 signal sum_53 : std_logic_vector(3 downto 0);
 signal sum_54 : std_logic_vector(3 downto 0);
 signal sum_55 : std_logic_vector(3 downto 0);
 signal sum_56 : std_logic_vector(3 downto 0);
 signal sum_57 : std_logic_vector(3 downto 0);
 signal sum_58 : std_logic_vector(3 downto 0);
 signal sum_59 : std_logic_vector(3 downto 0);
 signal sum_591 : std_logic_vector(3 downto 0);
 signal sum_592 : std_logic_vector(3 downto 0);
 signal sum_593 : std_logic_vector(3 downto 0);
 signal sum_594 : std_logic_vector(3 downto 0);
 signal sum_595 : std_logic_vector(3 downto 0);
 signal sum_596 : std_logic_vector(3 downto 0);
 signal sum_6 : std_logic_vector(3 downto 0);
 signal sum_61 : std_logic_vector(3 downto 0);
 signal sum_62 : std_logic_vector(3 downto 0);
 signal sum_63 : std_logic_vector(3 downto 0);
 signal sum_64 : std_logic_vector(3 downto 0);
 signal sum_7 : std_logic_vector(21 downto 0);
 signal sum_8 : std_logic_vector(3 downto 0);
```

```
 signal sum_81 : std_logic_vector(3 downto 0);
 signal sum_82 : std_logic_vector(3 downto 0);
 signal sum_83 : std_logic_vector(3 downto 0);
 signal sum_84 : std_logic_vector(3 downto 0);
 signal sum_9 : std_logic_vector(25 downto 0);
 signal add : std_logic_vector(2 downto 0);
 type ram_type is array(7 downto 0) of std_logic_vector(7 downto 0);
 signal ram : ram_type;
begin
    clk_out <= clk;
 process(clk,reset)
 begin
  if reset = '1' then
     state <= s0;
  elsif clk'event and clk = '1' then
   case state is
    when s0 =>
     if input = '1' then
      state <= s1;
      output_0 <= '1';
     else
      output_0 <= '0';
     end if;
    when s1 =>
     if input = '1' then
      state <= s2;
      output_1 <= '1';
     else
      output_1 <= '0';
     end if;
    when s2 =>
     if input = '1' then
      state <= s3;
      output_2 <= '1';
     else
      output_2 <= '0';
     end if;
    when s3 =>
     if input = '1' then
      state <= s4;
      output_3 <= '1';
     else
      output_3 <= '0';
     end if;
    when s4 =>
      if input = '1' then
       state <= s5;
       output_4 <= '1';
      else
       output_4 <= '0';
      end if;
    when s5 =>
     if input = '1' then
      state <= s51;
       output_5 <= '1';
     else
       output_5 <= '0';
     end if;
    when s51 =>
     if input = '1' then
      state <= s52;
       output_51 <= '1';
     else
       output_51 <= '0';
     end if;
    when s52 =>
     if input = '1' then
      state <= s53;
       output_52 <= '1';
```

```
 else
   output_52 <= '0';
 end if;
when s53 =>
 if input = '1' then
  state <= s54;
   output_53 <= '1';
 else
   output_53 <= '0';
 end if;
when s54 =>
 if input = '1' then
  state <= s55;
   output_54 <= '1';
 else
   output_54 <= '0';
 end if;
when s55 =>
 if input = '1' then
  state <= s56;
   output_55 <= '1';
 else
   output_55 <= '0';
 end if;
when s56 =>
 if input = '1' then
  state <= s57;
   output_56 <= '1';
 else
   output_56 <= '0';
 end if;
when s57 =>
 if input = '1' then
  state <= s58;
   output_57 <= '1';
 else
   output_57 <= '0';
 end if;
when s58 =>
 if input = '1' then
  state <= s59;
   output_58 <= '1';
 else
   output_58 <= '0';
 end if;
when s59 =>
 if input = '1' then
  state <= s591;
   output_59 <= '1';
 else
   output_59 <= '0';
 end if;
when s591 =>
 if input = '1' then
  state <= s592;
   output_591 <= '1';
 else
   output_591 <= '0';
 end if;
when s592 =>
 if input = '1' then
  state <= s593;
   output_592 <= '1';
 else
   output_592 <= '0';
 end if;
when s593 =>
 if input = '1' then
  state <= s594;
```

```
    output_593 <= '1';
  else
    output_593 <= '0';
  end if;
when s594 =>
  if input = '1' then
   state <= s595;
    output_594 <= '1';
  else
    output_594 <= '0';
  end if;
when s595 =>
  if input = '1' then
   state <= s596;
    output_595 <= '1';
  else
    output_595 <= '0';
  end if;
when s596 =>
  if input = '1' then
   state <= s6;
    output_596 <= '1';
  else
    output_596 <= '0';
  end if;
when s6 =>
  if input = '1' then
   state <= s61;
    output_6 <= '1';
  else
    output_6 <= '0';
  end if;
when s61 =>
  if input = '1' then
   state <= s62;
    output_61 <= '1';
  else
    output_61 <= '0';
  end if;
when s62 =>
  if input = '1' then
   state <= s63;
    output_62 <= '1';
  else
    output_62 <= '0';
  end if;
when s63 =>
  if input = '1' then
   state <= s64;
    output_63 <= '1';
  else
    output_63 <= '0';
  end if;
when s64 =>
  if input = '1' then
   state <= s7;
    output_64 <= '1';
  else
    output_64 <= '0';
  end if;
when s7 =>
  if input = '1' then
   state <= s8;
    output_7 <= '1';
  else
    output_7 <= '0';
  end if;
when s8 =>
  if input = '1' then
```

```
      state <= s81;
       output_8 <= '1';
      else
       output_8 <= '0';
     end if;
    when s81 =>
     if input = '1' then
      state <= s82;
       output_81 <= '1';
     else
       output_81 <= '0';
     end if;
    when s82 =>
     if input = '1' then
      state <= s83;
       output_82 <= '1';
     else
       output_82 <= '0';
     end if;
    when s83 =>
     if input = '1' then
      state <= s84;
       output_83 <= '1';
     else
       output_83 <= '0';
     end if;
    when s84 =>
     if input = '1' then
      state <= s9;
       output_84 <= '1';
     else
       output_84 <= '0';
     end if;
    when s9 =>
     if input = '1' then
      state <= s0;
       output_9 <= '1';
     else
       output_9 <= '0';
     end if;
    end case;
   end if;
 end process;
-------------------------------------
process(clk,reset)
 begin
    if reset = '1' then
        sum_0 <= (others => '0');
        sum_1 <= (others => '0');
        sum_2 <= (others => '0');
        sum_3 <= (others => '0');
        sum_4 <= (others => '0');
        sum_5 <= (others => '0');
        sum_51 <= (others => '0');
        sum_52 <= (others => '0');
        sum_53 <= (others => '0');
        sum_54 <= (others => '0');
        sum_55 <= (others => '0');
        sum_56 <= (others => '0');
        sum_57 <= (others => '0');
        sum_58 <= (others => '0');
        sum_59 <= (others => '0');
        sum_591 <= (others => '0');
        sum_592 <= (others => '0');
        sum_593 <= (others => '0');
        sum_594 <= (others => '0');
        sum_595 <= (others => '0');
        sum_596 <= (others => '0');
        sum_6 <= (others => '0');
```

```
        sum_61 <= (others => '0');
        sum_62 <= (others => '0');
        sum_63 <= (others => '0');
        sum_64 <= (others => '0');
        sum_7 <= (others => '0');
        sum_8 <= (others => '0');
        sum_81 <= (others => '0');
        sum_82 <= (others => '0');
        sum_83 <= (others => '0');
        sum_84 <= (others => '0');
        sum_9 <= (others => '0');
    elsif clk'event and clk = '1' then
-------------------------------------------------------------------------
     if output_0 = '1' then
      if (sum_0 < "1111") then
       sum_0 <= sum_0 + '1';
      else
       sum_0 <= "0000";
      end if;
      if (sum_0 = 2) then
         address <= '1';
      else
         address <= '0';
      end if;
      if (sum_0 = 6) then
         samping <= '1';
      else
         samping <= '0';
      end if;
-------------------------------------------------------------------------
     elsif output_1 = '1' then
      if (sum_1 < "1111") then
        sum_1 <= sum_1 + '1';
      else
        sum_1 <= "0000";
      end if;
      if (sum_1 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_1 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_2 = '1' then
      if (sum_2 < "1111") then
        sum_2 <= sum_2 + '1';
      else
        sum_2 <= "0000";
      end if;
      if (sum_2 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_2 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
 -------------------------------------------------------------------------
     elsif output_3 = '1' then
      if (sum_3 < "1111") then
       sum_3 <= sum_3 + '1';
      else
       sum_3 <= "0000";
```

```
      end if;
      if (sum_3 = 2) then
         address <= '1';
      else
         address <= '0';
      end if;
      if (sum_3 = 6) then
         samping <= '1';
      else
         samping <= '0';
      end if;
-----------------------------------------------------------------------
     elsif output_4 = '1' then
      if (sum_4 < "1111") then
        sum_4 <= sum_4 + '1';
      else
        sum_4 <= "0000";
      end if;
      if (sum_4 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_4 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-----------------------------------------------------------------------
     elsif output_5 = '1' then
      if (sum_5 < "1111") then
        sum_5 <= sum_5 + '1';
      else
        sum_5 <= "0000";
      end if;
      if (sum_5 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_5 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-----------------------------------------------------------------------
     elsif output_51 = '1' then
      if (sum_51 < "1111") then
        sum_51 <= sum_51 + '1';
      else
        sum_51 <= "0000";
      end if;
      if (sum_51 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_51 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-----------------------------------------------------------------------
     elsif output_52 = '1' then
      if (sum_52 < "1111") then
        sum_52 <= sum_52 + '1';
      else
        sum_52 <= "0000";
      end if;
```

```
      if (sum_52 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_52 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_53 = '1' then
      if (sum_53 < "1111") then
        sum_53 <= sum_53 + '1';
      else
        sum_53 <= "0000";
      end if;
      if (sum_53 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_53 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_54 = '1' then
      if (sum_54 < "1111") then
        sum_54 <= sum_54 + '1';
      else
        sum_54 <= "0000";
      end if;
      if (sum_54 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_54 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_55 = '1' then
      if (sum_55 < "1111") then
        sum_55 <= sum_55 + '1';
      else
        sum_55 <= "0000";
      end if;
      if (sum_55 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_55 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_56 = '1' then
      if (sum_56 < "1111") then
        sum_56 <= sum_56 + '1';
      else
        sum_56 <= "0000";
      end if;
      if (sum_56 = 2) then
```

```
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_56 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_57 = '1' then
      if (sum_57 < "1111") then
        sum_57 <= sum_57 + '1';
      else
        sum_57 <= "0000";
      end if;
      if (sum_57 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_57 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_58 = '1' then
      if (sum_58 < "1111") then
        sum_58 <= sum_58 + '1';
      else
        sum_58 <= "0000";
      end if;
      if (sum_58 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_58 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_59 = '1' then
      if (sum_59 < "1111") then
        sum_59 <= sum_59 + '1';
      else
        sum_59 <= "0000";
      end if;
      if (sum_59 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_59 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_591 = '1' then
      if (sum_591 < "1111") then
        sum_591 <= sum_591 + '1';
      else
        sum_591 <= "0000";
      end if;
      if (sum_591 = 2) then
        address <= '1';
```

```
      else
        address <= '0';
      end if;
      if (sum_591 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_592 = '1' then
      if (sum_592 < "1111") then
        sum_592 <= sum_592 + '1';
      else
        sum_592 <= "0000";
      end if;
      if (sum_592 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_592 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_593 = '1' then
      if (sum_593 < "1111") then
        sum_593 <= sum_593 + '1';
      else
        sum_593 <= "0000";
      end if;
      if (sum_593 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_593 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_594 = '1' then
      if (sum_594 < "1111") then
        sum_594 <= sum_594 + '1';
      else
        sum_594 <= "0000";
      end if;
      if (sum_594 = 2) then
        address <= '1';
      else
        address <= '0';
      end if;
      if (sum_594 = 6) then
        samping <= '1';
      else
        samping <= '0';
     end if;
-------------------------------------------------------------------------
     elsif output_595 = '1' then
      if (sum_595 < "1111") then
        sum_595 <= sum_595 + '1';
      else
        sum_595 <= "0000";
      end if;
      if (sum_595 = 2) then
        address <= '1';
      else
```

```
        address <= '0';
       end if;
       if (sum_595 = 6) then
         samping <= '1';
       else
         samping <= '0';
      end if;
-------------------------------------------------------------------------
      elsif output_596 = '1' then
       if (sum_596 < "1111") then
         sum_596 <= sum_596 + '1';
       else
         sum_596 <= "0000";
       end if;
       if (sum_596 = 2) then
         address <= '1';
       else
         address <= '0';
       end if;
       if (sum_596 = 6) then
         samping <= '1';
       else
         samping <= '0';
      end if;
-------------------------------------------------------------------------
      elsif output_6 = '1' then
       if (sum_6 < "1111") then
         sum_6 <= sum_6 + '1';
       else
         sum_6 <= "0000";
       end if;
       if (sum_6 = 2) then
         address_0 <= '1';
       else
         address_0 <= '0';
       end if;
       if (sum_6 = 6) then
         samping_0 <= '1';
       else
         samping_0 <= '0';
      end if;
      if (sum_6 < 1) then
        we_0 <= '1';
      elsif (sum_6 > 8) then
        we_0 <= '1';
      else
         we_0 <= '0';
      end if;
-------------------------------------------------------------------------
      elsif output_61 = '1' then
       if (sum_61 < "1111") then
         sum_61 <= sum_61 + '1';
       else
         sum_61 <= "0000";
       end if;
       if (sum_61 = 2) then
         address_0 <= '1';
       else
         address_0 <= '0';
       end if;
       if (sum_61 = 6) then
         samping_0 <= '1';
       else
         samping_0 <= '0';
      end if;
      if (sum_61 < 1) then
        we_0 <= '1';
      elsif (sum_61 > 8) then
        we_0 <= '1';
```

```
      else
         we_0 <= '0';
      end if;
-------------------------------------------------------------------------
      elsif output_62 = '1' then
       if (sum_62 < "1111") then
         sum_62 <= sum_62 + '1';
       else
         sum_62 <= "0000";
       end if;
       if (sum_62 = 2) then
         address_0 <= '1';
       else
         address_0 <= '0';
       end if;
       if (sum_62 = 6) then
         samping_0 <= '1';
       else
         samping_0 <= '0';
      end if;
      if (sum_62 < 1) then
        we_0 <= '1';
      elsif (sum_62 > 8) then
        we_0 <= '1';
      else
         we_0 <= '0';
      end if;
-------------------------------------------------------------------------
      elsif output_63 = '1' then
       if (sum_63 < "1111") then
         sum_63 <= sum_63 + '1';
       else
         sum_63 <= "0000";
       end if;
       if (sum_63 = 2) then
         address_0 <= '1';
       else
         address_0 <= '0';
       end if;
       if (sum_63 = 6) then
         samping_0 <= '1';
       else
         samping_0 <= '0';
      end if;
      if (sum_63 < 1) then
        we_0 <= '1';
      elsif (sum_63 > 8) then
        we_0 <= '1';
      else
         we_0 <= '0';
      end if;
-------------------------------------------------------------------------
      elsif output_64 = '1' then
       if (sum_64 < "1111") then
         sum_64 <= sum_64 + '1';
       else
         sum_64 <= "0000";
       end if;
       if (sum_64 = 2) then
         address_0 <= '1';
       else
         address_0 <= '0';
       end if;
       if (sum_64 = 6) then
         samping_0 <= '1';
       else
         samping_0 <= '0';
      end if;
      if (sum_64 < 1) then
```

```
        we_0 <= '1';
      elsif (sum_64 > 8) then
        we_0 <= '1';
      else
         we_0 <= '0';
      end if;
-------------------------------------------------------------------------------
      elsif output_7 = '1' then
       if (sum_7 < "11111111111111111111") then
        sum_7 <= sum_7 + '1';
       else
        sum_7 <= "00000000000000000000";
       end if;
       if sum_7 = 60 then
         reset_0 <= '1';
       elsif sum_7 = 540 then
         reset_0 <= '1';
       else
         reset_0 <= '0';
       end if;
-----------------------------------
       if sum_7 = 60 then
         reset_ram <= '1';
       elsif sum_7 = 540 then
         reset_ram <= '1';
       else
         reset_ram <= '0';
       end if;
-----------------------------------
       if (sum_7 < 60) then
          encryp <= '0';
       elsif sum_7 > 540 then
          encryp <= '0';
       else
          encryp <= '1';
       end if;
-----------------------------------
       if (sum_7 < 1) then
          we_0 <= '1';
       elsif (sum_7 > 8) then
          we_0 <= '1';
       else
          we_0 <= '0';
       end if;
-----------------------------------
       if (sum_7 = 2) then
          address_0 <= '1';
       elsif sum_7 = 80 then
          address_0 <= '1';
       elsif sum_7 = 100 then
          address_0 <= '1';
       elsif sum_7 = 120 then
          address_0 <= '1';
       elsif sum_7 = 140 then
         address_0 <= '1';
       elsif sum_7 = 160 then
          address_0 <= '1';
      elsif sum_7 = 180 then
          address_0 <= '1';
       else
          address_0 <= '0';
       end if;
-----------------------------------
       if (sum_7 = 6) then
          samping_0 <= '1';
       elsif sum_7 = 82 then
         samping_0 <= '1';
        elsif sum_7 = 102 then
         samping_0 <= '1';
```

```
      elsif sum_7 = 122 then
        samping_0 <= '1';
      elsif sum_7 = 142 then
        samping_0 <= '1';
      elsif sum_7 = 162 then
        samping_0 <= '1';
      elsif sum_7 = 182 then
        samping_0 <= '1';
      else
         samping_0 <= '0';
      end if;
-----------------------------------
      if sum_7 = 60 then
        reset_1 <= '1';
      elsif sum_7 = 540 then
        reset_1 <= '1';
      elsif sum_7 = 1050 then
        reset_1 <= '1';
      else
        reset_1 <= '0';
      end if;
-------------------------------------data
      if (sum_7 = 80) then
         address <= '1';
      elsif sum_7 = 100 then
         address <= '1';
      elsif sum_7 = 120 then
         address <= '1';
      elsif sum_7 = 140 then
         address <= '1';
      elsif sum_7 = 160 then
         address <= '1';
      elsif sum_7 = 180 then
         address <= '1';
      elsif sum_7 = 200 then
         address <= '1';
      elsif sum_7 = 220 then---------8
         address <= '1';
      elsif sum_7 = 240 then
         address <= '1';
      elsif sum_7 = 260 then
         address <= '1';
      elsif sum_7 = 280 then
         address <= '1';
      elsif sum_7 = 300 then
         address <= '1';
      elsif sum_7 = 320 then
         address <= '1';
      elsif sum_7 = 340 then
         address <= '1';
      elsif sum_7 = 360 then---------7+8
         address <= '1';
      elsif sum_7 = 380 then
         address <= '1';
      elsif sum_7 = 400 then
         address <= '1';
      elsif sum_7 = 420 then
         address <= '1';
      elsif sum_7 = 440 then
         address <= '1';
      elsif sum_7 = 460 then---------7+8+5
         address <= '1';
      elsif sum_7 = 480 then
         address <= '1';
      else
         address <= '0';
      end if;
 -------------------------------------data
      if (sum_7 = 80) then
```

```
        address_1 <= '1';
      elsif sum_7 = 100 then
        address_1 <= '1';
      elsif sum_7 = 120 then
        address_1 <= '1';
      elsif sum_7 = 140 then
        address_1 <= '1';
      elsif sum_7 = 160 then
        address_1 <= '1';
      elsif sum_7 = 180 then
        address_1 <= '1';
      elsif sum_7 = 200 then
        address_1 <= '1';
      elsif sum_7 = 220 then----------7
        address_1 <= '1';
      elsif sum_7 = 240 then
        address_1 <= '1';
      elsif sum_7 = 260 then
        address_1 <= '1';
      elsif sum_7 = 280 then
        address_1 <= '1';
      elsif sum_7 = 300 then
        address_1 <= '1';
      elsif sum_7 = 320 then
        address_1 <= '1';
      elsif sum_7 = 340 then
        address_1 <= '1';
      elsif sum_7 = 360 then---------7+8
        address_1 <= '1';
      elsif sum_7 = 380 then
        address_1 <= '1';
      elsif sum_7 = 400 then
        address_1 <= '1';
      elsif sum_7 = 420 then
        address_1 <= '1';
      elsif sum_7 = 440 then
        address_1 <= '1';
      elsif sum_7 = 460 then---------7+8+5
        address_1 <= '1';
      elsif sum_7 = 480 then
        address_1 <= '1';
-------------------------------
      elsif sum_7 = 580 then
        address_1 <= '1';
      elsif sum_7 = 600 then
        address_1 <= '1';
      elsif sum_7 = 620 then
        address_1 <= '1';
      elsif sum_7 = 640 then
        address_1 <= '1';
      elsif (sum_7 = 660) then
        address_1 <= '1';
      elsif sum_7 = 680 then
        address_1 <= '1';
      elsif sum_7 = 700 then
        address_1 <= '1';
      elsif sum_7 = 720 then
        address_1 <= '1';
      elsif sum_7 = 740 then
        address_1 <= '1';
      elsif sum_7 = 760 then
        address_1 <= '1';
      elsif sum_7 = 780 then
        address_1 <= '1';
      elsif sum_7 = 800 then-----------8
        address_1 <= '1';
      elsif sum_7 = 820 then
        address_1 <= '1';
      elsif sum_7 = 840 then
```

```
            address_1 <= '1';
         elsif sum_7 = 860 then
            address_1 <= '1';
         elsif sum_7 = 880 then
            address_1 <= '1';
         elsif sum_7 = 900 then
            address_1 <= '1';
         elsif sum_7 = 920 then
            address_1 <= '1';
         elsif sum_7 = 940 then---------8+6
            address_1 <= '1';
         elsif sum_7 = 960 then
            address_1 <= '1';
         elsif sum_7 = 980 then
            address_1 <= '1';
         else
            address_1 <= '0';
         end if;
-----------------------------------
         if sum_7 = 92 then
           samping_1 <= '1';
         elsif sum_7 = 112 then
           samping_1 <= '1';
         elsif sum_7 = 132 then
           samping_1 <= '1';
          elsif sum_7 = 152 then
           samping_1 <= '1';
         elsif sum_7 = 172 then
           samping_1 <= '1';
          elsif sum_7 = 192 then
           samping_1 <= '1';
         elsif sum_7 = 212 then
           samping_1 <= '1';
          elsif sum_7 = 232 then
           samping_1 <= '1';
         elsif sum_7 = 252 then
           samping_1 <= '1';
         elsif sum_7 = 272 then
           samping_1 <= '1';
          elsif sum_7 = 292 then
           samping_1 <= '1';
         elsif sum_7 = 312 then
           samping_1 <= '1';
          elsif sum_7 = 332 then
           samping_1 <= '1';
         elsif sum_7 = 352 then
           samping_1 <= '1';
          elsif sum_7 = 372 then
           samping_1 <= '1';
          elsif sum_7 = 392 then
           samping_1 <= '1';
         elsif sum_7 = 412 then
           samping_1 <= '1';
          elsif sum_7 = 432 then
           samping_1 <= '1';
         elsif sum_7 = 452 then
           samping_1 <= '1';
          elsif sum_7 = 472 then
           samping_1 <= '1';
          elsif sum_7 = 492 then
           samping_1 <= '1';
         else
            samping_1 <= '0';
         end if;
-----------------------------------
         if sum_7 < 1050 then
           encryp_decryp <= encryp_in;
         end if;
-----------------------------------
```

```
if sum_7 = 581 then
  tx_en <= '1';
elsif sum_7 = 601 then
  tx_en <= '1';
 elsif sum_7 = 621 then
  tx_en <= '1';
elsif sum_7 = 641 then
  tx_en <= '1';
elsif sum_7 = 661 then
  tx_en <= '1';
elsif sum_7 = 681 then
  tx_en <= '1';
elsif sum_7 = 701 then
  tx_en <= '1';
 elsif sum_7 = 721 then
  tx_en <= '1';
elsif sum_7 = 741 then
  tx_en <= '1';
 elsif sum_7 = 761 then
  tx_en <= '1';
elsif sum_7 = 781 then
  tx_en <= '1';
 elsif sum_7 = 801 then-------8
  tx_en <= '1';
elsif sum_7 = 821 then
  tx_en <= '1';
elsif sum_7 = 841 then
  tx_en <= '1';
 elsif sum_7 = 861 then
  tx_en <= '1';
elsif sum_7 = 881 then
  tx_en <= '1';
 elsif sum_7 = 901 then
  tx_en <= '1';
elsif sum_7 = 921 then
  tx_en <= '1';
 elsif sum_7 = 941 then-----8+7
  tx_en <= '1';
 elsif sum_7 = 961 then
  tx_en <= '1';
elsif sum_7 = 981 then
  tx_en <= '1';
else
   tx_en <= '0';
end if;
-------------------------------------------------------------------------
elsif output_8 = '1' then
 if (sum_8 < "1111") then
  sum_8 <= sum_8 + '1';
 else
  sum_8 <= "0000";
 end if;
 if (sum_8 = 2) then
  address_0 <= '1';
 else
  address_0 <= '0';
 end if;
 if (sum_8 = 6) then
  samping_0 <= '1';
 else
  samping_0 <= '0';
 end if;
 if (sum_8 < 1) then
  we_0 <= '1';
 elsif (sum_8 > 8) then
  we_0 <= '1';
 else
  we_0 <= '0';
end if;
```

```
    elsif output_81 = '1' then
     if (sum_8 < "1111") then
       sum_81 <= sum_81 + '1';
     else
       sum_81 <= "0000";
     end if;
     if (sum_81 = 2) then
       address_0 <= '1';
     else
       address_0 <= '0';
     end if;
     if (sum_81 = 6) then
       samping_0 <= '1';
     else
       samping_0 <= '0';
     end if;
     if (sum_81 < 1) then
      we_0 <= '1';
     elsif (sum_81 > 8) then
      we_0 <= '1';
     else
      we_0 <= '0';
    end if;
-------------------------------------------------------------------------
    elsif output_82 = '1' then
     if (sum_82 < "1111") then
       sum_82 <= sum_82 + '1';
     else
       sum_82 <= "0000";
     end if;
     if (sum_82 = 2) then
       address_0 <= '1';
     else
       address_0 <= '0';
     end if;
     if (sum_82 = 6) then
       samping_0 <= '1';
     else
       samping_0 <= '0';
     end if;
     if (sum_82 < 1) then
      we_0 <= '1';
     elsif (sum_82 > 8) then
      we_0 <= '1';
     else
      we_0 <= '0';
    end if;
-------------------------------------------------------------------------
    elsif output_83 = '1' then
     if (sum_83 < "1111") then
       sum_83 <= sum_83 + '1';
     else
       sum_83 <= "0000";
     end if;
     if (sum_83 = 2) then
       address_0 <= '1';
     else
       address_0 <= '0';
     end if;
     if (sum_83 = 6) then
       samping_0 <= '1';
     else
       samping_0 <= '0';
     end if;
     if (sum_83 < 1) then
      we_0 <= '1';
     elsif (sum_83 > 8) then
      we_0 <= '1';
     else
```

```
        we_0 <= '0';
      end if;
-------------------------------------------------------------------------
      elsif output_84 = '1' then
       if (sum_84 < "1111") then
         sum_84 <= sum_84 + '1';
       else
         sum_84 <= "0000";
       end if;
       if (sum_84 = 2) then
         address_0 <= '1';
       else
         address_0 <= '0';
       end if;
       if (sum_84 = 6) then
         samping_0 <= '1';
       else
         samping_0 <= '0';
       end if;
       if (sum_84 < 1) then
        we_0 <= '1';
       elsif (sum_84 > 8) then
        we_0 <= '1';
       else
        we_0 <= '0';
      end if;
-------------------------------------------------------------------------
      elsif output_9 = '1' then
       if (sum_9 < "11111111111111111111111111") then
        sum_9 <= sum_9 + '1';
       else
        sum_9 <= "00000000000000000000000000";
       end if;
       if sum_9 = 60 then
         reset_0 <= '1';
       else
         reset_0 <= '0';
       end if;
-----------------------------------
       if sum_9 = 60 then
         reset_2 <= '1';
       elsif sum_9 = 540 then
         reset_2 <= '1';
       elsif sum_9 = 1050 then
         reset_2 <= '1';
       else
         reset_2 <= '0';
       end if;
-----------------------------------
       if sum_9 = 60 then
         reset_1 <= '1';
       else
         reset_1 <= '0';
       end if;
-----------------------------------
       if (sum_9 = 2) then
          address_0 <= '1';
       elsif (sum_9 = 80) then
          address_0 <= '1';
       elsif sum_9 = 100 then
          address_0 <= '1';
       elsif (sum_9 = 120) then
          address_0 <= '1';
       elsif sum_9 = 140 then
          address_0 <= '1';
       elsif sum_9 = 160 then
          address_0 <= '1';
       elsif sum_9 = 180 then
          address_0 <= '1';
```

```
      else
        address_0 <= '0';
      end if;
-----------------------------------
      if (sum_9 = 6) then
        samping_0 <= '1';
      elsif sum_9 = 82 then
        samping_0 <= '1';
      elsif sum_9 = 102 then
        samping_0 <= '1';
      elsif sum_9 = 122 then
        samping_0 <= '1';
      elsif sum_9 = 142 then
        samping_0 <= '1';
      elsif sum_9 = 162 then
        samping_0 <= '1';
      elsif sum_9 = 182 then
        samping_0 <= '1';
      else
        samping_0 <= '0';
      end if;
-----------------------------------
      if (sum_9 < 1) then
       we_0 <= '1';
      elsif (sum_9 > 8) then
       we_0 <= '1';
      else
       we_0 <= '0';

 end if;
-----------------------------------
      if (sum_9 = 80) then
         address_1 <= '1';
      elsif sum_9 = 100 then
         address_1 <= '1';
      elsif sum_9 = 120 then
         address_1 <= '1';
      elsif sum_9 = 140 then
         address_1 <= '1';
      elsif sum_9 = 160 then
         address_1 <= '1';
      elsif sum_9 = 180 then
         address_1 <= '1';
      elsif sum_9 = 200 then
         address_1 <= '1';
      elsif sum_9 = 220 then------8
         address_1 <= '1';
      elsif sum_9 = 240 then
         address_1 <= '1';
      elsif sum_9 = 260 then
         address_1 <= '1';
      elsif sum_9 = 280 then
         address_1 <= '1';
      elsif sum_9 = 300 then
         address_1 <= '1';
      elsif sum_9 = 320 then
         address_1 <= '1';
      elsif sum_9 = 340 then
         address_1 <= '1';
      elsif sum_9 = 360 then-------7+8
         address_1 <= '1';
      elsif sum_9 = 380 then
         address_1 <= '1';
      elsif sum_9 = 400 then
         address_1 <= '1';
      elsif sum_9 = 420 then
         address_1 <= '1';
      elsif sum_9 = 440 then
         address_1 <= '1';
```

```
      elsif sum_9 = 460 then-------7+8+5
         address_1 <= '1';
      elsif sum_9 = 480 then
         address_1 <= '1';
      else
         address_1 <= '0';
      end if;
-----------------------------------
      if (sum_9 < 60) then
         decryp <= '0';
      elsif sum_9 > 540 then
         decryp <= '0';
      else
         decryp <= '1';
      end if;
-----------------------------------
      if (sum_9 = 80) then
         address_2 <= '1';
      elsif sum_9 = 100 then
         address_2 <= '1';
      elsif sum_9 = 120 then
         address_2 <= '1';
      elsif sum_9 = 140 then
         address_2 <= '1';
      elsif sum_9 = 160 then
         address_2 <= '1';
      elsif sum_9 = 180 then
         address_2 <= '1';
      elsif sum_9 = 200 then
         address_2 <= '1';
      elsif sum_9 = 220 then------8
         address_2 <= '1';
      elsif sum_9 = 240 then
         address_2 <= '1';
      elsif sum_9 = 260 then
         address_2 <= '1';
      elsif sum_9 = 280 then
         address_2 <= '1';
      elsif sum_9 = 300 then
         address_2 <= '1';
      elsif sum_9 = 320 then
         address_2 <= '1';
      elsif sum_9 = 340 then
         address_2 <= '1';
      elsif sum_9 = 360 then-----7+8
         address_2 <= '1';
      elsif sum_9 = 380 then
         address_2 <= '1';
      elsif sum_9 = 400 then
         address_2 <= '1';
      elsif sum_9 = 420 then
         address_2 <= '1';
      elsif sum_9 = 440 then
         address_2 <= '1';
      elsif sum_9 = 460 then-----7+8+5
         address_2 <= '1';
      elsif sum_9 = 480 then
         address_2 <= '1';
-----------------------------------
      elsif sum_9 = 580 then
         address_2 <= '1';
      elsif sum_9 = 600 then
         address_2 <= '1';
      elsif sum_9 = 620 then
         address_2 <= '1';
      elsif (sum_9 = 640) then
         address_2 <= '1';
      elsif (sum_9 = 660) then
         address_2 <= '1';
```

```
elsif sum_9 = 680 then
   address_2 <= '1';
elsif sum_9 = 700 then
   address_2 <= '1';
elsif sum_9 = 720 then
   address_2 <= '1';
elsif sum_9 = 740 then
   address_2 <= '1';
elsif sum_9 = 760 then
   address_2 <= '1';
elsif sum_9 = 780 then
   address_2 <= '1';
elsif sum_9 = 800 then-------8
   address_2 <= '1';
elsif sum_9 = 820 then
   address_2 <= '1';
elsif sum_9 = 840 then
   address_2 <= '1';
elsif sum_9 = 860 then
   address_2 <= '1';
elsif sum_9 = 880 then
   address_2 <= '1';
elsif sum_9 = 900 then
   address_2 <= '1';
elsif sum_9 = 920 then
   address_2 <= '1';
elsif sum_9 = 940 then
   address_2 <= '1';
elsif sum_9 = 960 then
   address_2 <= '1';
elsif sum_9 = 980 then
   address_2 <= '1';
else
   address_2 <= '0';
end if;
-------------------------------------------------------------------------
if sum_9 = 92 then
  samping_2 <= '1';
elsif sum_9 = 112 then
  samping_2 <= '1';
elsif sum_9 = 132 then
  samping_2 <= '1';
 elsif sum_9 = 152 then
  samping_2 <= '1';
elsif sum_9 = 172 then
  samping_2 <= '1';
 elsif sum_9 = 192 then
  samping_2 <= '1';
elsif sum_9 = 212 then
  samping_2 <= '1';
 elsif sum_9 = 232 then------8
  samping_2 <= '1';
elsif sum_9 = 252 then
  samping_2 <= '1';
elsif sum_9 = 272 then
  samping_2 <= '1';
 elsif sum_9 = 292 then
  samping_2 <= '1';
elsif sum_9 = 312 then
  samping_2 <= '1';
 elsif sum_9 = 332 then
  samping_2 <= '1';
elsif sum_9 = 352 then
  samping_2 <= '1';
 elsif sum_9 = 372 then------7+8
  samping_2 <= '1';
 elsif sum_9 = 392 then
  samping_2 <= '1';
elsif sum_9 = 412 then
```

```
        samping_2 <= '1';
       elsif sum_9 = 432 then
        samping_2 <= '1';
      elsif sum_9 = 452 then
        samping_2 <= '1';
       elsif sum_9 = 472 then------7+8+5
        samping_2 <= '1';
       elsif sum_9 = 492 then
        samping_2 <= '1';
      else
         samping_2 <= '0';
      end if;
-----------------------------------
      if sum_9 < 1050 then
        encryp_decryp <= decryp_in;
      end if;
-----------------------------------
      if sum_9 = 581 then
        tx_en <= '1';
      elsif sum_9 = 601 then
        tx_en <= '1';
       elsif sum_9 = 621 then
        tx_en <= '1';
      elsif sum_9 = 641 then
        tx_en <= '1';
      elsif sum_9 = 661 then
        tx_en <= '1';
      elsif sum_9 = 681 then
        tx_en <= '1';
      elsif sum_9 = 701 then
        tx_en <= '1';
       elsif sum_9 = 721 then
        tx_en <= '1';
      elsif sum_9 = 741 then
        tx_en <= '1';
       elsif sum_9 = 761 then
        tx_en <= '1';
      elsif sum_9 = 781 then
        tx_en <= '1';
       elsif sum_9 = 801 then------8
        tx_en <= '1';
      elsif sum_9 = 821 then
        tx_en <= '1';
      elsif sum_9 = 841 then
        tx_en <= '1';
       elsif sum_9 = 861 then
        tx_en <= '1';
      elsif sum_9 = 881 then
        tx_en <= '1';
       elsif sum_9 = 901 then
        tx_en <= '1';
      elsif sum_9 = 921 then
        tx_en <= '1';
       elsif sum_9 = 941 then-------8+7
        tx_en <= '1';
       elsif sum_9 = 961 then
        tx_en <= '1';
      elsif sum_9 = 981 then
        tx_en <= '1';
      else
         tx_en <= '0';
      end if;
     end if;
   end if;
 end process;
-------------------------------------------------------------------------
 process(address_0,clk)
  begin
   if (reset_0 ='1') then
```

```
        add   <= (others=>'0');
   elsif clk'event and clk = '1' then
     if address_0 = '1' then
      if(add < "111")then
            add <= add + '1';
        else
          add <= "000";
      end if;
     end if;
   end if;
 end process;
-------------------------------------------------------------------------
 process(samping_0,add)
 begin
   if( samping_0'event and samping_0 = '1' ) then
    if (we_0 = '0') then
     ram(conv_integer(add)) <= key_in;
     end if;
   end if;
 end process;
     key_out <= ram(conv_integer(add));
-------------------------------------------------------------------------
end;
```

กิตติกรรมประกาศ

ปริญญานิพนธ์ชิ้นนี้ สำเร็จลุล่วงได้ด้วยดี ด้วยความกรุณาเป็นอย่างดียิ่งจาก ดร.พรชัย ทรัพย์นิธิ และ อ.ศรวัฒน์ ชิวปรีชา ที่กรุณาให้ ทั้งในด้านความรู้ ความช่วยเหลือ ให้คำปรึกษาแนะนำ รวมถึงการอนุเคราะห์ เครื่องมือและอุปกรณ์ต่างๆ ตลอดจนตรวจสอบแก้ไขจนสำเร็จลงได้

สุดท้ายนี้ขอกราบขอบคุณ คุณพ่อ คุณแม่ และ เพื่อนทุกคน ที่คอยช่วยเหลือจนการจัดทำปริญญานิพนธ์ครั้งนี้สำเร็จลงได้

คณะผู้จัดทำ

บรรณานุกรม

1. ECC White Paper "THE ELLIPTIC CURVE CRYPTOSYSTEM FOR SMART CARDS." Certicom ,May 1998,www.certicom.com
2. ECC White Paper "REMARK ON THE SECURITY OF THE ELLIPTIC CURVE CRYPTOSYSTEM" Certicom ,July 2000
3. Don Johnson, Alfred Menezes and Scott Vanstone. "The Elliptic Curve Digital Signature Algorithm (ECDSA)" Certicom
4. ผศ.ดร.ลัญฉกร วุฒิสิทธิกุลกิจ, "เทคโนโลยีโทรคมนาคม ทฤษฎีข่าวสารและการเข้ารหัส" กรุงเทพฯ : สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย,2546