ระบบการเข้ารหัสลับและถอดรหัสลับ แบบ ECC

ELLIPTIC CURVE CRYPTOSYSTEM

โดย

นายสาธิต จันทร์ศิริ

นายเอกภพ ภูธราพันธ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ระบบการเข้ารหัสลับและถอดรหัสลับ แบบ ECC

ELLIPTIC CURVE CRYPTOSYSTEM

โดย

นายสาธิต จันทร์ศิริ 44015034

นายเอกภพ ภูธราพันธ์ 44015046

อาจารย์ที่ปรึกษา

ดร. พรชัย ทรัพย์นิธิ

อาจารย์ ศรวัฒน์ ชิวปรีชา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ระบบการเข้ารหัสและถอดรหัสลับ แบบ ECC
ELLIPTIC CURVE CRYPTOSYSTEM

โดย นายสาธิต จันทร์ศิริ 44015034
นาย เอกภพ ภูธราพันธ์ 44015046

อาจารย์ที่ปรึกษา ดร.พรชัย ทรัพย์นิธิ
อาจารย์ศรวัฒน์ ชิวปรีชา

บทคัดย่อ

โครงงานนี้นำเสนอการสร้างระบบเข้ารหัสและถอดรหัสข้อมูลแบบ ECC (ELLIPTIC CURVE CRYPTOSYSTEM) ซึ่งทำหน้าที่เป็น ส่วนของเข้ารหัสและถอดรหัสข้อมูลในสมาร์ทการ์ด โดยในการออกแบบได้ใช้ภาษา VHDL ซึ่งเป็นภาษา ที่ทำการบรรยายลักษณะการทำงานของฮาร์ดแวร์ได้เป็นอย่างดี โดยไม่อิงกับเทคโนโลยีเฉพาะผู้ผลิตรายใด และ ทำการทดสอบการทำงานบนอุปกรณ์ FPGAs ซึ่งวิธีการเข้ารหัสแบบ ECC มีข้อดีคือ ขนาดของกุญแจ (keys) มีขนาดเล็ก เมื่อเทียบกับการเข้ารหัสแบบอื่นในระดับความปลอดภัยที่เท่ากัน ทำให้ประหยัดเนื้อที่หน่วยความจำของสมาร์ทการ์ดในการเก็บกุญแจ

Abstract

This project designs an encription and decryption method known as "Elliptic Curve Cryptosystem" for a Smart card application. A hardware implementation is to be tested on FPGAs using the hardware language VHDL. Compared with other encryption/decryption systems, the ECC generates keys with smaller size given comparable level of security. The method, consequently, reduces hardware memory for key storage

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ระบบการเข้ารหัสและถอดรหัสลับ แบบ ECC**

ELLIPTIC CURVE CRYPTOSYSTEM

ผู้จัดทำ

นาย สาธิต จันทร์ศิริ 44015034

นาย เอกภพ ภูธราพันธ์ 44015046

..... อาจารย์ที่ปรึกษา

(ดร.พรชัย ทรัพย์นิธิ)

..... อาจารย์ที่ปรึกษา

(อาจารย์ศรวัฒน์ ชิวปรีชา)

สารบัญ

สารบัญรูป

รูปที่ 7.27 แสดงส่วนแสดงผลการถอดรหัสข้อความ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ที่มาของโครงงาน

โดยทั่วไปสมาร์ทการ์ดถูกนำไปใช้งานในรูปแบบธุรกิจทางการเงินเช่น ธุรกิจธนาคาร ต่อจากนั้น ก็เริ่มใช้ในภาคอุตสาหกรรม ธุรกิจข้อมูลเฉพาะบุคคล ธุรกิจประกันภัย และอื่นๆ เนื่องจากการใช้งานที่ง่าย, สะดวก และ มีราคาถูก มีส่วนช่วยให้สมาร์ทการ์ดเป็นที่นิยมและแพร่หลายอย่างรวดเร็ว โดยเริ่มแรกขนาดความจุของหน่อยความจำบนสมาร์ทการ์ดเก็บข้อมูลได้น้อย ต่อมาด้วยเทคโนโลยี VLSI ทำให้สามารถสร้างหน่วยความจำที่มีขนาดความจุเพิ่มมากขึ้น แต่เนื่องจากขนาดของกุญแจสาธารณะที่ใช้งานมีขนาดใหญ่ จึงทำให้สิ้นเปลืองหน่วยความจำเพื่อใช้สำหรับการเก็บค่าของกุญแจสาธารณะนี้ ดังนั้นจึงมีความต้องการที่จะลดขนาดของกุญแจสาธารณะ ที่ใช้งานกับสมาร์ทการ์ดให้มีขนาดเล็กลง และสามารถใช้งานได้อย่างปลอดภัยตามที่มาตรฐานกำหนด ดังนั้นจึงเกิดแนวคิดที่นำเอาระบบเข้ารหัสลับแบบ Elliptic Curve (ECC : Elliptic Curve Cryptosystem) มาประยุกต์ใช้งานกับสมาร์ทการ์ด เนื่องจากขนาดของกุญแจที่เล็กกว่ามาก แต่ให้ความปลอดภัยที่เท่ากันเมื่อเทียบกับรหัสลับระบบอื่นๆ อีกทั้งในปัจจุบันได้มีการทำวิจัยกันออกมามากมาย จนทำให้ ECC นั้นสามารถสร้างให้วงจรมีขนาดเล็กและทำงานได้เร็วมาก ซึ่งมีความเหมาะสมเป็นอย่างยิ่งที่จะนำใช้งานบนสมาร์ทการ์ด

ปกติการรักษาความปลอดภัยของข้อมูลนั้นจะต้องเข้ารหัสข้อมูลก่อนที่จะเก็บลงในหน่วยความจำ ในส่วนของสมาร์ทการ์ดนั้นได้เก็บกุญแจสาธารณะซึ่งใช้สำหับการเข้ารหัสข้อมูล และกุญแจส่วนตัวซึ่งใช้ในการถอดรหัสข้อมูล เอาไว้ในหน่วยความจำของสมาร์ทการ์ด ในการส่งข้อมูลนั้นผู้ส่งจะเข้ารหัสข้อมูลด้วยกุญแจสาธารณะของผู้รับที่ผู้ส่งต้องการจะส่งถึง และ ในส่วนของการถอดรหัส เมื่อมีเอกสารที่ถูกเข้ารหัสด้วยกุญแจสาธารณะของผู้ถือบัตรส่งมา ผู้ถือบัตรจะสามารถถอดรหัสได้ด้วยกุญแจส่วนตัวที่เก็บอยู่ในบัตรนั่นเอง

โครงงานนี้ได้ทำการจำลองระบบการเข้ารหัสลับแบบECCสำหรับใช้งานบนสมาร์ทการ์ดซึ่งใดฉิตศาสตร์แบบ Elliptic Curve Discrete Logarithm Problem บนคณิตศาสตร์สนามจำกัด(Finite Field) โดยในส่วนของการออกแบบตัวประมวลผลข้อมูลนั้นทำการออกแบบด้วยภาษาVHDLและใคอมพิวเตอร์ช่วยในการจำลองการทำงานพร้อมทั้งทำการสังเคราะห์วงจรเป็นระดับเกตหลังจากนั้นจึงนำมาสร้างเป็นฮาร์ดแวร์ด้วยเทคโนโลยี VLSI เพื่อจำลองการทำงานของสมาร์ทการ์ดต่อไป

1.2 วัตถุประสงค์

1.2.1) เพื่อศึกษากระบวนการเข้ารหัสและถอดรหัสลับระบบกุญแจสาธารณะ ที่นำมาจำลองการทำงานบนสมาร์ทการ์ด

1.2.2) เพื่อศึกษาวิธีการเข้าเข้ารหัสและถอดรหัสลับระบบกุญแจสาธารณะที่เหมาะสม

1.2.3) เพื่อศึกษาถึงวิธีการออกแบบวงจรรวมด้วยภาษา VHDL และวิธีทดสอบผลการทำงานของวงจรด้วยอุปกรณ์ FPGA

1.3 ขอบเขตของโครงงาน

โครงงานนี้ได้จำลองการทำงานของสมาร์ทการ์ด ให้มีฟังก์ชั่นเข้ารหัสและถอดรหัสอยู่ภายใน โดยใช้อักกอริทึมของที่เป็นมาตรฐานของECCพร้อมทั้งได้ทำออกแบบหน่วยความจำที่ใช้จึงทำให้สามารถเข้ารหัสและถอดรหัสข้อมูลได้จากอุปกรณ์เอฟพีจีเอได้โดยตรง

1.4 ผลที่คาดว่าจะได้รับ

1.4.1) เพื่อพัฒนาทักษะในการออกแบบวงจรรวมด้วยภาษาวีเอชดีแอล และการใช้งานอุปกรณ์เอจีเอ

1.4.2) เพื่อพัฒนาความรู้ความเข้าใจกระบวนการในการเข้ารหัสและถอดรหัส

1.4.3) เพื่อให้โครงงานที่สร้างขึ้นนี้เป็นต้นแบบที่ให้ผู้สนใจนำไปศึกษาและพัฒนาต่อไป

บทที่ 2
สมาร์ทการ์ด

2.1 ประวัติความเป็นมาของสมาร์ทการ์ด

การใช้งานสมาร์ทการ์ดนั้น เริ่มเกิดต้นจากการใช้งานบัตรที่อยู่ในรูปของบัตรพลาสติกซึ่ง เรียกว่า พลาสติกการ์ด (Plastic Card) ซึ่งเริ่มใช้ในปี ค.ศ. 1950 โดยบริษัท Dinerclup ซึ่งสมาชิกผู้ถือบัตรสามารถใช้บริการตาม ภัตตาคาร ร้านอาหาร หรือโรงแรมที่ซึ่งบริษัทนี้เป็นสมาชิกอยู่ ต่อมาปี ค.ศ. 1951 บริษัทAmerican Express ก็ได้นำบัตรนี้มาให้บริการแก่สมาชิกของบริษัท ซึ่งการใช้งานของทั้งสองบริษัทนั้นเป็นลักษณะของการให้สิทธิพิเศษกับสมาชิก แต่ที่จริงแล้วการใช้บัตรนี้เป็นบัตรเครดิตนั้นได้เริ่มใช้งานตั้งแต่ปี ค.ศ. 1940 ในธนาคารของสหรัฐอเมริกา ซึ่งจะให้สิทธิพิเศษแก่ลูกค้าของธนาคาร โดยที่ลูกค้าของธนาคารสามารถติดต่อกับธนาคารสาขาต่างๆ ในสหรัฐอเมริกาได้สะดวกและรวดเร็ว จึงเรียกการทำธุรกิจโดยใช้บัตรแบบนี้ว่า "วีซ่า" ต่อมาในปี ค.ศ. 1966 ก็ได้มีการใช้บัตรนี้ในประเทศอังกฤษ โดยใช้งานร่วมกับบัตร วีซ่าในสหรัฐอเมริกาและปี ค.ศ. 1972 ธนาคารยักษ์ใหญ่ 4 ธนาคารคือ ธนาคาร Lloyds ธนาคาร Nation Westminster ธนาคาร Midland และ ธนาคาร Royal ก็ได้ร่วมมือกันเพื่อที่จะใช้บัตรนี้ติดต่อธุรกิจทางการเงิน โดยใช้ชื่อของ Master card ต่อมาเพื่อเพิ่มความสะดวกสบายให้กับลูกค้าในการรับบริการถอนเงินด่วนจึงมีการใช้บัตรนี้กับตู้บริการเงินด่วน คือ Automatic Teller Machine (ATM) ซึ่งเป็นที่แพร่หลายในเวลาต่อมา

การใช้งานสมาร์ทการ์ดนั้น เริ่มใช้งานในปี ค.ศ. 1970 ที่ประเทศฝรั่งเศส โดยเกิดแนวคิดที่ต้องการใช้บัตรแทนจำนวนเงินซึ่งเรียกว่า Chip - Card หลังจากนั้นก็ได้มีการพัฒนาให้มีการทำงานร่วมกับวงจรอิเลคทรอนิกส์ซึ่งเรียกว่า Integrated Circuit Card (ICC) และในทุกวันนี้ ใช้งานสมาร์ทการ์ดกันอย่างแพร่หลายในวงสังคม ผลจากการใช้บัตรในการทำธุรกรรมต่างๆ นั้นทำให้ชีวิตสะดวกสบายมากขึ้น รวดเร็วขึ้น ซึ่งเราสามารถนำเอาสมาร์ทการ์ดไปใช้งานต่างๆ ได้ดังเช่น ธุรกิจทางการเงิน ข้อมูลทางสุขภาพ ข้อมูลเฉพาะบุคคล การรักษาความปลอดภัย ธุรกิจสื่อสาร บัตรเอทีเอ็มสำหรับเบิกเงินสด บัตรชมภาพยนต์ บัตรสำหรับซื้อตั๋วรถไฟฟ้า รถโดยสารประจำทาง อื่นๆ อีกมากมาย ทุกวันนี้บัตรสมาร์ทการ์ดได้ถูกพัฒนามากขึ้น แต่จุดหนึ่งที่มีความสำคัญและน่าสนใจมากที่สุดสำหรับการใช้งานบัตรสมาร์ทการ์ดคือ ทางด้านความปลอดภัยของการใช้งานทั้งระบบ ซึ่งถึงแม้ว่าจะมีการพัฒนาไปมากแล้วก็ตามแต่ก็ยังป้องกันได้ไม่เต็ม 100 เปอร์เซ็นต์ แต่ที่แน่นอนคือดีกว่าบัตรพลาสติกแถบแม่เหล็กที่ใช้กันแพร่หลายในปัจจุบันอย่างแน่นอน ซึ่งอาจกล่าวได้ว่าสมาร์ทการ์ดนั้นกำลังจะเข้ามาทีส่วนสำคัญเกี่ยวกับการดำเนินชีวิตประจำวันนั่นเอง

2.2 องค์ประกอบภายในสมาร์ทการ์ด

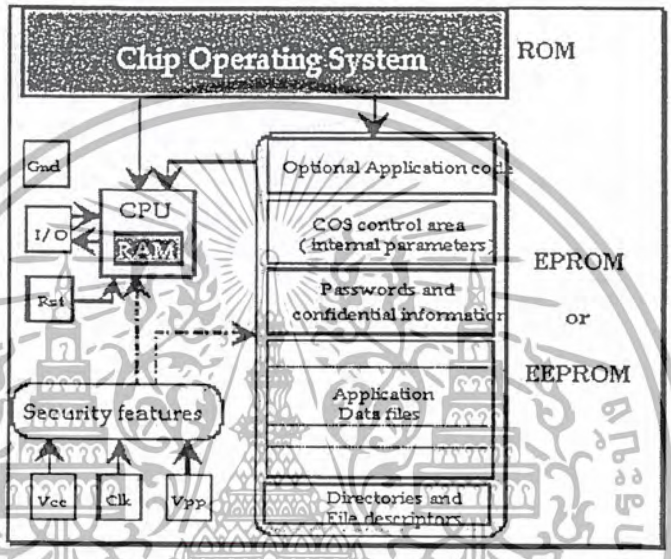
สมาร์ทการ์ด คือ อุปกรณ์เก็บข้อูลแบบพกพาซึ่งสามารรถแสดงข้อกำหนดและคุณสมบัติเฉพาะตัว พร้อมทั้งระดับความปลอดภัยได้ ซึ่งสมาร์ทการ์ดเกิดขึ้นจากการใช้งานเทคโนโลยีอิเล็กทรอนิกส์ ทำการออกแบบตัวประมวลผลสำหรับการใช้งานดังกล่าวและเนื่องจากการพัฒนาเทคโนโลยีทางด้าน

อิเล็กทรอนิกส์นี้ ทำให้สมาร์ทการ์ด มีความจุของข้อมูลเพิ่มมากขึ้น ความเร็วในการประมวลผลสูงขึ้นนั่นเอง

2.2.1 โครงสร้างพื้นฐานของสมาร์ทการ์ด

สมาร์ทการ์ด สามารถแบ่งโครงสร้างออกเป็น 2 ส่วนใหญ่ๆ ด้วยกันคือ

- ไมโครโปรเซสเซอร์
- หน่วยความจำ



รูปที่ 2.1 แสดงโครงสร้างพื้นฐานของสมาร์ทการ์ด

2.2.1.1 ไมโครโปรเซสเซอร์

สถาปัตยกรรมในส่วนนี้นั้นประกอยด้วย CPU RAM ROM และ EEPROM หน่วยความจำ ROM ทำหน้าที่เก็ยชุดคำสั่งซึ่งเป็นขั้นตอนการทำงานของระบบ CPU ทำหน้าที่ประมวลผลข้อมูลโดยที่หน่วยความจำ RAM ทำหน้าที่เป็นรีจิสเตอร์ช่วยสำหรับการประมวลผลข้อมูล และข้อมูลที่ได้ประมวลผลเสร็จเรียบร้อยและได้ถูกเก็บในหน่วยความจำ EEPROM โดยทั่วไปนั้นขนาดของหน่วยความจำ EEPROM มีขนาดเป็น 4 เท่าของหน่วยความจำ RAM

2.1.1.2 หน่วยความจำ

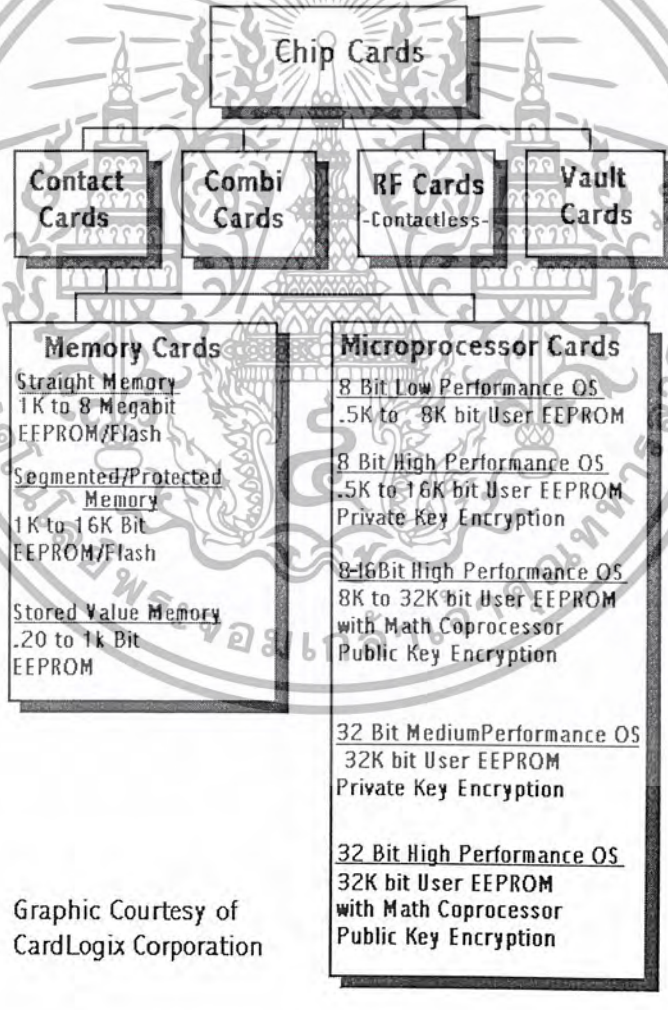
ส่วนของหน่วยความจำนั้นส่วนที่เก็บข้อมูลของสมาร์ทการ์ด โดยสามารถออกแบบได้ 3 ส่วนดังนี้

- พื้นที่เปิด (Open Zone) คือ พื้นที่ที่ใช้สำหรับเก็บข้อมูลของบริษัทผู้ผลิต Serial Number ข้อมูลของผู้ถือบัตรที่สามารถเปิดเผยได้
- พื้นที่ปกปิด (Secret Zone) คือพื้นที่ที่ใช้สำหรับเก็บข้อมูลปกปิดส่วนบุคคล

- พื้นที่ใช้งาน (User Zone) คือพื้นที่เก็บข้อมูลที่ได้ทำการเข้ารหัส

2.3 การ์ดหน่วยความจำและการ์ดไมโครโปรเซสเซอร์ชนิดต่างๆ

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

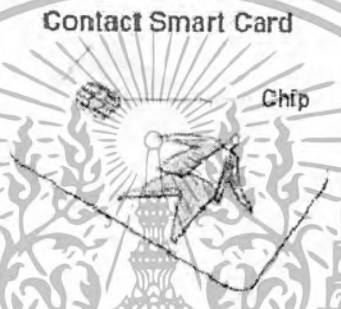


รูปที่ 2.2 แสดงสมาร์ทการ์ดชนิดต่าง

2.4 สมาร์ทการ์ดแบบมีคอนแทกต์และไม่มีคอนแทกต์

2.4.1 สมาร์ทการ์ดแบบมีคอนแทกต์

สมาร์ทการ์ดสามารถแบ่งออกเป็น 2 ประเภทใหญ่ๆ คือ ชนิดมีคอนแทกต์ (Contract Smart Card) และชนิดไม่มีคอนแทกต์ (Contractless Smart Card) สำหรับสมาร์ทการ์ดแบบมีคอนแทกต์นั้นต้องใช้งาน โดยการสอดบัตรเข้ากับเครื่องอ่านบัตร (Smart card Reader) บัตรชนิดนี้จะมีเพลตสีทองขนาดเล็ก (Small Gold Plate) ที่ด้านหน้าของบัตรซึ่งแตกต่างจากบัตรเครดิตแบบใช้แถบแม่เหล็กที่พบเห็นได้ทั่วไปซึ่งมีแถบแม่เหล็กอยู่ทางด้านหลังบัตร เมื่อการ์ดถูกสอดเข้ากับเครื่องอ่านบัตรทำให้เกิดการสัมผัสที่คอนแทกต์ของบัตร ทำให้เกิดการเชื่อมต่อทางไฟฟ้า เพื่อใช้ในการรับส่งข้อมูลเข้าและออก จากชิบบนบัตรได้ ลักษณะของบัตรสมาร์ทการ์ดแบบมีคอนแทกต์ ดังแสดงรูปที่ 2.3

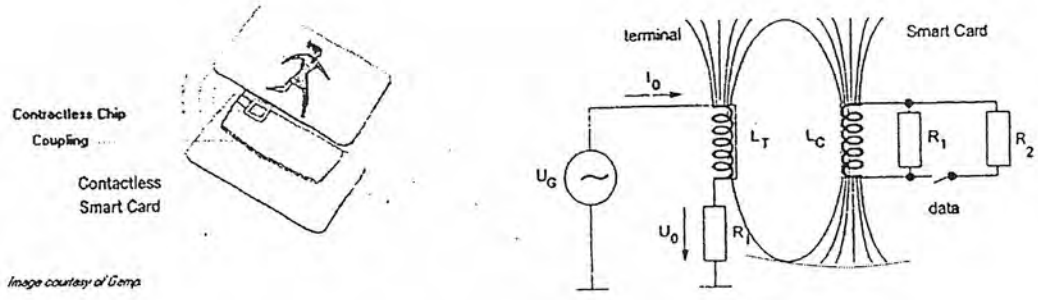


รูปที่ 2.3 สมาร์ทการ์ดแบบคอนแทกต์

2.4.2 สมาร์ทการ์ดแบบไม่มีคอนแทกต์

บัตรสมาร์ทการ์ดอีกชนิดหนึ่งคือชนิดไม่มีคอนแทกต์ (Contactlass) มีความพิเศษกว่าตรงที่มีการรับส่งข้อมูลระหว่างชิปภายในบัตรกับภายนอก โดยใช้การส่งผ่านข้อมูลทางอากาศโดยการเหนี่ยวนำ (Coupling Loop) ซึ่งถูกติดตั้งอยู่ภายในบัตร ดังนั้นถ้าดูจากภายนอกตัวบัตรจะมีลักษณะเหมือนกับบัตรเครดิตพลาสติกทั่วไป แตกต่างกันเพียงแต่ว่าสมาร์ทการ์ดแบบนี้ไม่มีไมโครชิปอิเล็กทรอนิกส์และส่วนของการสื่อสารข้อมูลโดยการเหนี่ยวนำติดตั้งอยู่ภายใน ซึ่งอุปกรณ์เหล่านี้ทำให้การ์ดสามารติดต่อสื่อสารข้อมูลโดยปราศจากการสัมผัส บัตรสมาร์ทการ์ดแบบนี้จึงเหมาะอย่างยิ่งสำหรับงาน ที่ต้องการความรวดเร็วในการใช้งานเช่น ใช้เป็นตัวสำหรับระบบรถไฟฟ้าหรือการผ่านเข้าทางด่วนพิเศษที่มีผู้ใช้มากๆ เป็นต้น

นอกจากบัตรสมาร์ทการ์ดทั้งสองแบบดังกล่าวแล้ว ปัจจุบันยังมีการผลิตสมาร์ทการ์ดแบบผสมหรือที่เรียกว่า คอมบิการ์ด (Combi Card) ออกมาใช้งานอีกด้วย โดยบัตรแบบนี้เป็นบัตรใบเดียวแต่ทำหน้าที่เป็นทั้งสมาร์ทการ์ดแบบมีการสัมผัส และสมาร์ทการ์ดแบบไม่มีการสัมผัสเพื่อเพิ่มความสะดวกและประโยชน์ในการใช้งานมากขึ้น



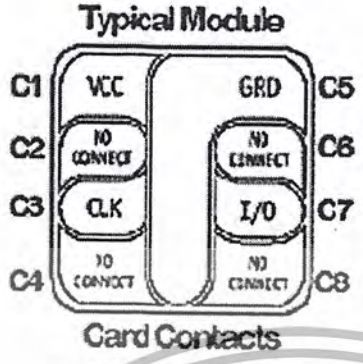
รูปที่ 2.4 สมาร์ทการ์ดแบบไม่มีคอนแทกต์

2.5 มาตราฐานของสมาร์ทการ์ด

ในปี ค.ศ. 1981 ได้มีการกำหนดมาตรฐานการส่งผ่านข้อมูลของสมาร์ทการ์ด โดยใช้มาตราฐานของ AFNOR (Association Francais de Normalisation) และได้นำมาตราฐานนี้มากำหนดเป็นมาตราฐานของ ISO ซึ่งปัจจุบันมีมาตราฐานพื้นฐานที่เกี่ยวข้องกับสมาร์ทการ์ดชนิดของคอนแทคต์ถูกกำหนดตามมาตราฐาน ISO 7816 ซีรีส์ซึ่งมีอยู่ทั้งหมด 10 หมวด ในขณะที่บัตรชิดที่ไม่มีคอนแทคต์จะถูกกำหนดตามมาตราฐาน ISO 1443 มาตราฐานเหล่านี้จะเป็นตัวกำหนดคุณสมบัติต่างๆ ของสมาร์ทการ์ด ที่มีการผลิตออกมาใช้งานครอบคลุมตั้งแต่ ลักษณะภายนอกโดยทั่วไป กระบวนการทางไฟฟ้า ทางกลไกและการอินเตอร์เฟส ต่อไปนี้เป็นรายละเอียดมาตราฐาน 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 = 1 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 Manufacturers
- 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 2.5 แสดงมาตราฐานขาสัญญาณของบัตรสมาร์ทการ์ด

2.6 มาตราฐานสำหรับการรักษาความปลอดภัยในบัตรสมาร์ทการ์ด

ความเป็นจริงแล้วในปัจจุบันนี้ มีการนำรูปแบบและเทคนิคในการรักษาความปลอดภัยบนสมาร์ทการ์ดมาใช้งานหลายรูปแบบซึ่งในที่นี้คงจะไม่ได้กล่าวถึงเทคนิคต่างๆ ที่นำมาใช้งาน แต่กล่าวเฉพาะรูปแบบและลักษณะหรือระดับการรักษาความปลอดภัยที่มีใช้กันอยู่ ซึ่งการเข้าถึงข้อมูลต่างๆ บนบัตรได้นั้นถูกแบ่งออกตามลักษณะการควบคุมได้ 2 แบบคือ แบบแรกเป็นการควบคุมว่าผู้ใดจะสามารถเข้าถึงข้อมูลต่างๆ ในบัตรได้อย่างไรบ้าง และแบบที่ 2 คือควบคุมระดับของการเข้าถึงข้อมูลในบัตร

ในแบบแรกนั้นหมายถึงเป็นการใช้ระบบความปลอดภัยควบคุมว่าต้องการให้ใครบ้างสามารถเข้าถึงข้อมูลได้ ซึ่งแบ่งได้เป็น 3 ระดับ

1. เข้าถึงได้ทุกคน คือ บัตรสมาร์ทการ์ดประเภทนี้จะไม่มีการใช้รหัสผ่าน (Password) ผู้ใดที่ถือบัตรนี้สามารถเข้าถึงข้อมูลได้ทันที แต่อาจเป็นข้อมูลบางส่วนเท่านั้นก็ได้ เช่น ข้อมูลเกี่ยวกับชื่อ มีเลือดกรุ๊ปใด เป็นต้น ซึ่งเสมือนบัตร Medicard ที่สามารถอ่านข้อมูลได้ โดยไม่ต้องใช้รหัสผ่าน
2. เข้าถึงได้เฉพาะผู้ถือบัตร คือ บัตรแบบนี้ส่วนใหญ่มีการใช้รหัสผ่านสำหรับผู้ถือบัตรร่วมด้วยซึ่งมักเรียกกันว่า PIN (Personal Identification Number) ที่อาจมีความยาว 4-5 หลัก โดยป้อนผ่านทางคีย์แพด ระบบนี้ถ้าหากมีผู้ที่พยายามเข้าถึงข้อมูลภายในบัตรนี้ โดยพยายามป้อนรหัสผ่าน ซึ่งถ้าป้อนรหัสผิดเกิน 3 ครั้งบัตรจะทำการล็อคตัวเอง และถ้าทำการปลดล็อคนี้ต้องใช้รหัสผ่านตัวอื่น ซึ่งมีความซับซ้อนกว่ามาป้อนแทน จึงกลับมาสู่สภาวะปกติได้

3. เข้าถึงได้เฉพาะบุคคลที่ 3 เท่านั้น คือ สมาร์ทการ์ดบางชนิดสามารถเข้าถึงข้อมูลได้เฉพาะบุคคลที่ 3 เท่านั้นซึ่งโดยส่วนใหญ่ก็คือ ผู้ออกบัตรให้เท่านั้น เช่นบัตรสมาร์ทการ์ดที่ใช้แทนเงินอิเล็กทรอนิกส์ สามารถโหลดข้อมูลใหม่ได้เฉพาะธนาคารที่ออกบัตรให้เท่านั้นเป็นต้น

การรักษาความปลอดภัยในแบบที่ 2 คือการจำกัดการเข้าถึงข้อมูลภายในบัตร โดยสามารถแบ่งการเข้าถึงข้อมูลภายในบัตรออกเป็นส่วนๆ ซึ่งส่วนใหญ่แบ่งออกเป็น 4 ระดับ

1. ข้อมูลที่สามารถอ่านได้เท่านั้น (Read Only) คือสามารถอ่านข้อมูลในบัตรได้เท่านั้น
2. ข้อมูลที่สามารถเพิ่มได้เท่านั้น (Added Only) คือสามารถเขียนข้อมูลลงไปในบัตรได้เท่านั้นนั้น
3. ข้อมูลที่สามารถอัพเดทได้เท่านั้น (Updated Only) คือเก็บข้อมูลใหม่ได้เท่านั้นโดยที่ข้อมูลเก่าถูกลบทิ้ง
4. ข้อมูลที่ไม่อนุญาตเข้าถึงได้เลย คือไม่สามารถอ่านหรือเขียนข้อมูลภายในบัตรได้เลย

บัตรโดยทั่วไปหรือบัตรสมาร์ทการ์ดบางประเภทควบคุมการใช้ข้อมูลที่อยู่ในบัตรโดยการใช้รหัสผ่าน ซึ่งมีแต่เพียงผู้ถือบัตรเท่านั้นที่ทราบ แต่วิธีนี้หากข้อมูลตรงส่วนนี้จำเป็นต้องถูกส่งผ่านทางสายโทรศัพท์หรือทางคลื่นวิทยุแล้ว คงจะนับได้ว่าการรักษาความปลอดภัยเพียงเท่านี้คงไม่เพียงพอ วิธีหนึ่งที่สามารถนำมาใช้เพื่อเพิ่มความปลอดภัยขึ้นคือ การใช้กระบวนการไซเฟอร์ริ่ง (ciphering) ซึ่งจะมีการทำงานคล้ายกับการแปลภาษาจากข้อมูลหนึ่งไปเป็นภาษาอื่นที่ไม่รู้จัก สมาร์ทการ์ดบางระบบจะถูกบรรจุระบบไซเฟอร์ริ่งและดีไซเฟอร์ริ่ง ซึ่งก็คือการเข้ารหัสและการถอดรหัสเพื่อทำการแปลข้อมูลให้กลับมาเหมือนเดิม ดังนั้นการส่งผ่านข้อมูลจึงทำได้อย่างสมบูรณ์ไม่มีอะไรผิดเพี้ยน

สมาร์ทการ์ดสามารถใช้ระบบไซเฟอร์ริ่งนี้ ทำการแปลข้อมูลที่แตกต่างกันได้มากนับหลายพันล้านรูปแบบและจะทำการสุ่มรูปแบบกันใหม่ทุกครั้งที่มีการสื่อสาร ด้วยวิธีการนี้จึงทำให้เรามั่นใจได้ว่าการใช้งานบัตรสมาร์การ์ดในการประยุกต์ในงานทุกรูปแบบและในบางประเภทที่ต้องการความปลอดภัยสูง เช่น การใช้แทนเงินสด

2.7 อนาคตของสมาร์ทการ์ด

คุณสมบัติที่เด่นที่สำคัญของสมาร์ทการ์ดก็คือ เป็นอุปกรณ์อิเล็กทรอนิกส์ที่สามารถพกพาได้ในกระเป๋าสตางค์ของทุกคน ซึ่งมีหน้าที่หลักคือจัดเก็บและจัดการกับข้อมูลต่างๆ ในรูปแบบอิเล็กทรอนิกส์ได้ ความฉลาดของสมาร์ทการ์ดคือ วงจรอิเล็กทรอนิกส์ที่ถูกติดตั้งอยู่ภายในการ์ดพลาสติกซึ่งสามารถที่จะปกปิดข้อมูลพร้อมกับการจัดการข้อมูลภายในได้ และในอนาคตด้วย เทคโนโลยีคล้ายกันนี้อาจนำพาไปสู่การติดตั้งในรูปแบบอื่นๆ เช่นในกุญแจ นาฬิกา แว่นตา แหวน และอุปกรณ์อื่นๆ ที่เป็นสิ่งใกล้ตัวที่เราต้องใช้อยู่ทุกๆ วัน และในทุกวันนี้ก็มีการใช้เทคโนโลยี สมาร์ทการ์ดนี้เป็นสมาร์ทคีย์ (Smart Key) ในการใช้งานกับระบบเคเบิลทีวีที่มีการบอกรับสมาชิกกันแล้ว การพัฒนาในอีกด้านหนึ่งที่น่าสนใจ

กว่าคือการพัฒนาทางเทคโนโลยีสมาร์ทการ์ดแบบไม่มีคอนแทคต์ ซึ่งเหมาะมากสำหรับการนำไปสร้างเป็นอุปกรณ์ที่เราเรียกว่าแท็ก (Tag) อุปกรณ์ประเภทแท็กนั้นมีการทำงานคล้ายกับสมาร์ทการ์ดแบบไม่มีคอนแทคคต์มากเพียงแต่อาจจะไม่ได้อยู่ในรูปแบบของบัตรพลาสติกเท่านั้น แต่อาจเป็นรูปแบบของแหวนคล้องหรือป้ายติดสินค้าซึ่งปัจจุบันมีการนำมาใช้งานกันบ้างแล้วเช่นการนำไปติดเข้ากับถังก๊าซ รถยนต์ สัตว์ เป็นต้น ส่วนใหญ่ทำหน้าที่ในการจัดเก็บข้อมูลที่เกี่ยวข้องกับสิ่งนั้นๆ ไว้ และอาจเป็นไปได้ที่ผู้ควบคุมอาจทำการเปลี่ยนแปลงข้อมูลภายในโดยไม่ต้องนำมาแก้ไขทีละชิ้นก็ได้ นอกจากนั้นสมาร์ทการ์ดยังอาจนำไปใช้งานร่วมกับระบบรักษาความปลอดภัยชั้นสูง ซึ่งในปัจจุบันเราเรียกกันว่าระบบไบโอเมตริก (Biometrics) โดยการใช้นิ้วมือ ฝ่ามิอ เรตินาของตา หรือใช้เสียงในการระบุและแยกแยะแต่ละบุคคล ในอนาคตอาจมีการใช้ข้อมูลทางอิเล็กทรอนิกส์ที่บรรจุอยู่ในสมาร์ทการ์ดมาร่วมประมวลผลข้อมูลอีกด้วย สมาร์ทการ์ดจึงถือได้ว่าเป็นเทคโนโยลีใหม่ที่น่าติดตามและมีความสัมพันธ์ต่อการดำรงค์ชีวิตของผู้คนนับหลายล้านคน ซึ่งในขณะนี้ก็เริ่มเป็นที่ประจักษ์แล้วเมื่อมีการใช้งานสมาร์ทการ์ดในการซื้อสินค้าต่างๆ ในร้านค้าหรือเมื่อใช้บัตรนี้ไปพบแพทย์

2.8 แนวคิดในการพัฒนาสมาร์ทการ์ด

ในปัจจุบันมีการใช้งานสมาร์ทการ์ดกันอย่างแพร่หลาย อีกทั้งความต้องการใช้งานสมาร์ทการ์ดเพื่อที่จะเก็บข้อมูลเฉพาะบุคคลมีเพิ่มมากขึ้นเช่น ข้อมูลบัตรผู้ป่วยในโรงพยาบาล จึงส่งผลให้ความต้องการเพิ่มความจุในการเก็บข้อมูลบนสมาร์ทการ์ดเพิ่มมากขึ้น ดังนั้นจึงเกิดแนวคิดที่จะเพิ่มขนาดของหน่วยความจำบนบัตรนั่นเอง เนื่องจากขนาดของกุญแจรหัสที่ใช้งานในปัจจุบันมีขนาดใหญ่มาก จึงทำให้เกิดการสิ้นเปลืองเนื้อที่บนหน่วยความจำบนสมาร์ทการ์ดซึ่ง ใช้สำหรับเก็บ Public-Key และยังทำให้วงจรที่ออกแบบมีขนาดใหญ่ ดังนั้นจึงเกิดแนวคิดที่จะทำการลดขนาดของ Public-Key โดยที่ระดับความปลอดภัยยังคงใช้งานได้ดี พร้อมกับวิธีการเข้ารหัสข้อมูลเพื่อทำให้ความจุของข้อมูลบนบัตรสมาร์ทการ์ดเพิ่มมากขึ้น

บทที่ 3
ภาษาวีเอชดีแอล

3.1 แนะนำวีเอชดีแอล (Introduction to VHDL)

ในช่วงฤดูร้อนของปี 1981 สถาบันเพื่อป้องกัน (The Institute for Defence Analysis) ในสหรัฐอเมริกาได้จัดตั้งคณะทำงานขึ้นคฯะหนึ่ง เพื่อทำการพัฒนาภาษาที่ใช้ในการบรรยายหรืออะบายรูปแบบการทำงานและความสัมพันธ์ ของอุปกรณ์ฮาร์ดแวร์แบบใหม่ขึ้น ผลการทำงานของคณะทำงานชุดนี้ได้ก่อให้เกิดภาษาการบรรยายฮาร์ดแวร์ขึ้น เรียกว่า VHDL (VHSIC Hardware Description Languge) โดย VHSIC เป็นชื่อย่อของแผนกหนึ่งของสถาบันที่ทำงานเกี่ยวกับ วงจรรวมที่มีความเร็วสูงมาก (Very High Speed Intergrated Circuit) ต่อมาในปี 1985 IEEE ได้ทำการผลักดันให้ VHDL กลายเป็นภาษาที่เป็นมาตราฐานและมีการยอมรับกันอย่างกว้างขวางในวงการอุตสาหกรรมคอมพิวเตอร์ ด้วยความสามารถของ VHDL ในด้านของการกำหนดพฤตกรรมการทำงานของวงจร ทำให้นักออกแบบสามารถกำหนดรูปแบบพฤติกรรมการทำงานได้ทั้งวงจรของดิจิตอลทั่วๆ ไป และในระบบที่แตกต่างกันออกไป เช่น พฤติกรรมการทำงานของระบบเรดาร์หรือพฤติกรรมการทำงานของระบบเครือข่ายใยประสาทในสมองมนุษย์ได้ ข้อดีมีหลักที่สำคัญของVHDL ก็คือภาษานี้จะสามารถถูกใช้ได้ตลอดในทุกๆ ระดับขั้นตอนการออกแบบที่ต่างกันได้ นั่นคือในกระบวนการออกแบบระดับสูง(System Level) จนถึงระดับที่ต่ำกว่า(Lower harware level) สามารถใช้ภาษาเดียวกันได้โดยตลอด ทำให้เพิ่มประสิทธิภาพในการติดต่อระหว่างกลุ่มที่ทำงานร่วมกันได้เป็นอย่างดี

3.1.1 ข้อกำหนด (VHDL Requirement)

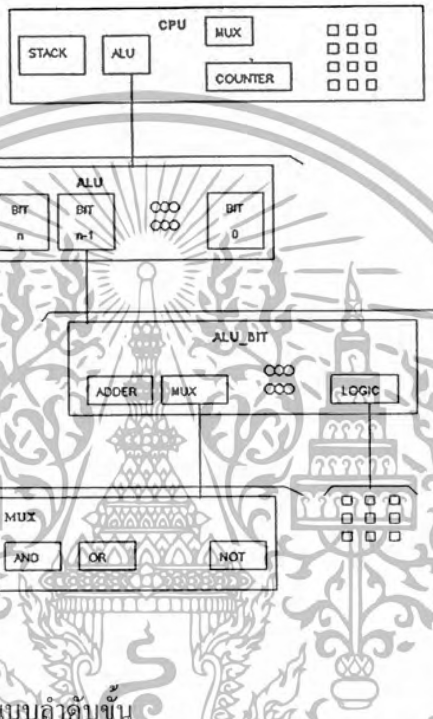
ในเอกสารของ DoD (Department of Defense Requriment for Hardware Description Languge) ซึ่งออกมาในเดือนมกราคมปี 1983 ได้ตั้งข้อกำหนดสำหรับภาษา VHDL ไว้ดังนี้

3.1.1.1 ลักษณะทั่วไป (Generation Features)

เอกสารของ DoD กำหนดไว้ว่า VHDL เป็นภาษาสำหรับการออกแบบและบรรยายของฮาร์ดแวร์ ซึ่งหมายถึงความสามารถในการอธิบายและออกแบบในระดับสูง โดยมีความสามารถในการเลียนแบบ (Simulation) การสังเคราะห์(Synthesis) และการทดสอบ(Testing) นอกจากนั้น VHDL ยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์คือ ระบบจนถึงระดับเกทอีด้วย เนื่องจากในการทำงานของระบบดิจิตอลจริงๆ ทุกๆ องค์ประกอบในระบบไม่ว่าเล็กหรือใหญ่จะทำงานไปพร้อมๆ กัน ซึ่งในเรื่องของความพร้อมในการทำงานนี้ถือว่าเป็นข้อกำหนดที่สำคุญอย่างหนึ่ง ใน VHDL ด้วยเช่นกัน (สำหรับในภาษาที่ใช้ในการบรรยายฮาร์ดแวร์แล้ว ความพร้อมเพรียงจะ หมายถึงทุกๆ คำสั่งองค์ประกอบเกท หรือวงจรต่างๆ จะถูกนำมาปฏิบัติทั้งหมด ดังนั้นในตอนท้ายแล้วก็จะดูเหมือนว่า ได้ปฏิบัติไปพร้อมๆ กัน)

3.1.1.2 สนับสนุนการออกแบบแบบลำดับขั้น (Support for Design Hierarchy)

การออกแบบลำดับขั้น เป็นลักษณะที่สำคัญอย่างหนึ่งการออกแบบที่มีหลายๆ ระดับในการออกแบบจะประกอบด้วยส่วนการบรรยายการเชื่อมต่อและส่วนการบรรยายหน้าที่การทำงาน หน้าที่การทำงานของระบบก็สามารถกำหนดได้ด้วยตนเองหรือถูกกำหนดโดยโครงสร้างที่ประกอบย่อยๆ ลงไปได้เช่นกัน แต่ที่ระดับล่างสุดองค์ประกอบต้องถูกบรรยายหน่าที่การทำงานด้วยตัวมันเองและไม่สามารถกำหนดการทำงานโดยลักษณะแบบโครงสร้างได้



รูปที่ 3.1 แสดงตัวอย่างการออกแบบแบบลำดับขั้น

3.1.1.3 ไลบรารี (Library Support)

VHDL ได้สนับสนุนการมีไลบรารีเพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษณะและการทำงานของอุปกรณ์พื้นฐานไว้ในระบบไลบรารี หรือจะใช้ไลบรารีที่ระบบได้จัดเตรียมไว้แล้วก็ได้โมเดลและการบรรยายที่ถูกต้องควรจะถูกเก็บไว้ในไลบรารี หลังจากที่ได้ผ่านการคอมไพล์เรียบร้อยแล้ว เพื่อให้ผู้ออกแบบคนอื่นๆ สามารถนำไปใช้ได้ด้วย

3.1.1.4 ลำดับคำสั่ง (Sequential Statement)

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการโดยพร้อมเพรียงกันจเป็นคุณสมบัติที่สำคัญของ VHDL ก็ตาม ตัวภาษาเองได้มีการจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วย เมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ปะกอบที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้ว ผูออกแบบก็ยังสามารถบรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดภายในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียน

โปรแกรมที่ประกอบด้วยโครงสร้างแบบ case if-then-else และ loop ทั่วๆ ไปได้ การบรรยายแบบลำดับคำสั่งทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้สะดวกและง่ายขึ้น อย่างไรก็ตาม โครงสร้างทั้งหมดของ VHDL ก็คงเป็นการทำงานพร้อมเพรียงกันเช่นเดิม

3.1.1.5 การกำหนดคุณสมบัติ (Generic Design)

นอกจากการกำหนดอินพุทและอินเอาท์พุทแล้ว เงื่อนไขอื่นๆ ก็มีผลต่อการปฏิบัติหน้าที่ ของอุปกรณ์ฮาร์ดแวร์ด้วยเช่นกัน สิ่งนี้รวมถึงสภาพแวดล้อมและลักษณะทางกายภาพของอุปกรณ์นั้นๆ ภาษาสำหรับการออกแบบที่ดีควรจะสามารถให้ผู้ออกแบบกำหนอคุณสมบัติของอุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนดขนาด ลักษณะทางกายภาพ เวลา โหลดและเงื่อนไขทางสภาพแวดล้อมอื่นๆ 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) การกำหนดตัวกระทำต่างๆ ทั้งเก่าและใหม่หรืออะไรก็ตามได้เช่นเดียวกับการเขียนโปรแกรมทั่วไป

3.1.1.8 การควบคุมเวลา (Timing Control)

VHDL อนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ตามต้องการการตรวจสอบ การออกแบบเกท หรือการหน่วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอยเหตุการณ์(Event) นอกจากนี้ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

3.1.1.9 การกำหนดแบบโครงสร้าง (Structural Specification)

การกำหนดโครงสร้างขององค์ประกอบสามารถกระทำได้ในทุกๆ ระดับของการออกแบบ การกำหนดโครงสร้างขององค์ประกอบรวมที่เกิดจากองค์ประกอยย่อยที่ต่างกันหรือเหมือนกันก็เป็นเรื่องข้อกำหนดมาตราฐานอย่างหนึ่งเช่นกัน

3.2 ความสามารถของภาษาวีเอชดีแอล (Capability)

- ตัวภาษา VHDL สามารถใช้เป็นสื่อกลางในการแลกเปลี่ยนระหว่างผู้ผลิตชิพกับผู้ออกแบบ (CAD Tools)

- ใช้เป็นสื่อกลางในการแลกเปลี่ยนสื่อสารระหว่างซีเออี(CAE) และซีเอดีทูล(CAD Tools) เช่น ตัวภาษาซ้อสโค้ด(SOURCE CODE) ของ VHDL สามารถคอมไพล์เลอร์(Compiler) และซีมูเลเตอร์ (Simurator) ได้หลายตัวแตกต่างกัน

- ภาษา VHDL สนับสนุนการออกแบบ แบบบนลงล่าง(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 สามารถเคลื่อนย้ายไปยังระบบใดๆ ก็ได้ และสามารถนำกลับมาใช้ใหม่ได้

- 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)

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

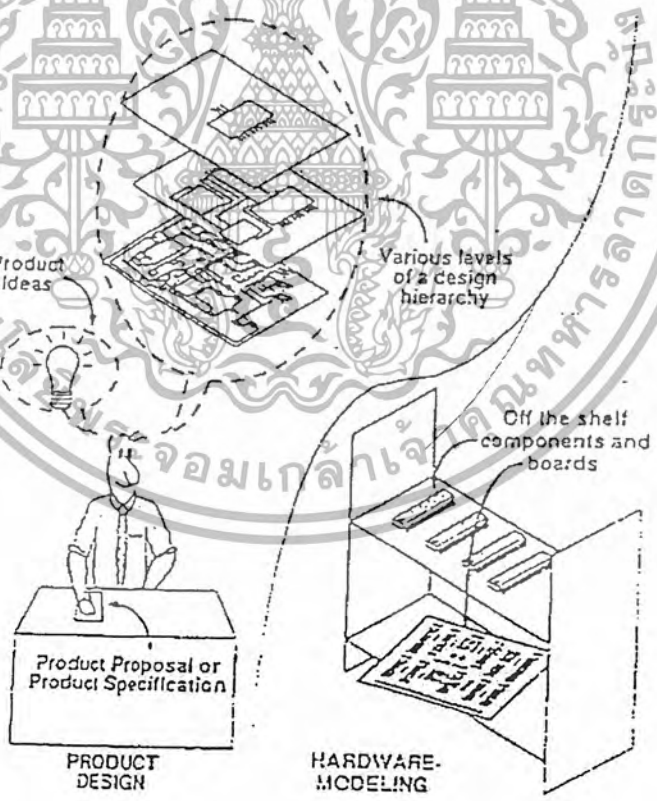
- ภาษา VHDL เป็นมาตรฐานที่ใช้โดยบริษัทและผู้ออกแบบหลายๆ แห่ง ฉะนั้นจึงง่ายที่จะทำความเข้าใจถึงแม้ว่าจะมาจากแหล่งต่างๆ

- โมเดลที่สร้างขึ้นสามารถจำลองการทำงานได้ เพราะว่าตัวแปรภาษาได้ตรวจสอบตัวแปรทางซิมูเลชั่นซีเมนติกไว้ด้วย

3.3 หลักการสร้างโมเดลโดยใช้ภาษาวีเอชดีแอล (General VHDL Modelling Principles)

วีเอชดีแอลเป็นภาษาที่ใช้สำหรับอธิบายการทำงานของฮาร์ดแวร์ในรูปแบบฟอร์มที่อ่านเข้าใจได้ ซึ่งจะช่วยในการสร้างและออกแบบวงจรรวมดิจิตอลและส่วนประกอบต่างๆ อาจใช้อธิบายทั้งระบบเพียงบางส่วน ซึ่งอยู่ในรูปของ (Component Block) จากนั้นก็ทำการจำลองการทำงาน(Simulate) โดยที่รูปแบบนั้นยังไม่ได้สร้างขึ้นจริงหรือเพียงแต่อยู่ในรูปของคำอธิบายเท่านั้น(Textual Format) หลังจากจำลองการทำงานจนได้ตามที่ต้องการจึงนำไปทำการSynthesis เพื่อให้ได้วงจรเกทเลเวลต่อไป ประโยชน์จริงของการใช้วงจร วีเอชดีแอล เป็น Design Tools แทนการสร้างตนแบบ (Prototype) ขึ้นมาจริง คือ เราสามารถอธิบาย Product Idea , Product Proposal , product Specification ในรูปแบบของ Text จากนั้นก็นำคอมไพล์เพื่อดู Timing การทำงานแล้วแก้ไข (Refine) จนกว่าจะได้ Specification ตามต้องการ เมื่อ product ได้ผลตามที่ต้องการแล้วจึงนำไปสู่การสังเคราะห์(Synthesis) เพื่อให้ได้เกทเลเวล Schematic เพื่อนำไปสร้างเป็นต้นแบบจริงต่อไป ซึ่งต้นแบบที่สร้างนั้นทำงานได้จริงเพราะได้ทำการ Simulate เรียบร้อยแล้ว เป็นการลดเวลาและค่าใช้จ่ายในการสร้างแบบได้มาก

ตัวภาษา VHDL สนับสนุนหลักการต่างๆ ให้เขียนแก้ไขและบำรุงรักษาวงจรดิจิตอลที่มีความซับซ้อนให้เป็นไปอย่างรวดเร็วและมีประสิทธิภาพโดยมีหลักการดังนี้



รูปที่ 3.2 สิ่งต่างๆ ที่สามารถอธิบายได้ด้วย VHDL

3.3.1 Top Down Design

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน เช่น ASIC (Application Specific Integrated Circuit) วิศวกรหรือผู้ออกแบบมักจะมองรูปแบบให้อยู่ในรูปของ Block Diagram เสียก่อน ก่อนที่จะย่อยรูปแบบให้ถึงรายละเอียดต่อไป ซึ่งภาษา VHDL นั้นอนุญาตให้อธิบายการทำงานของแต่ละ Block วิเคราะห์การทำงาน จัดการแก้ไขและปรับปรุงการทำงานจากการวิเคราะห์เพื่อให้ได้การทำงานตามที่ต้องการ ก่อนที่จะทำการออกแบบให้ละเอียดลึกลงไปในขั้นตอนต่อไป การแก้ไขในขั้นตอนนี้จะทำให้ลดค่าใช้จ่ายกว่าการแก้ไขในช่วงของการพัฒนาในระดับสร้างซิลิกอนชิพ

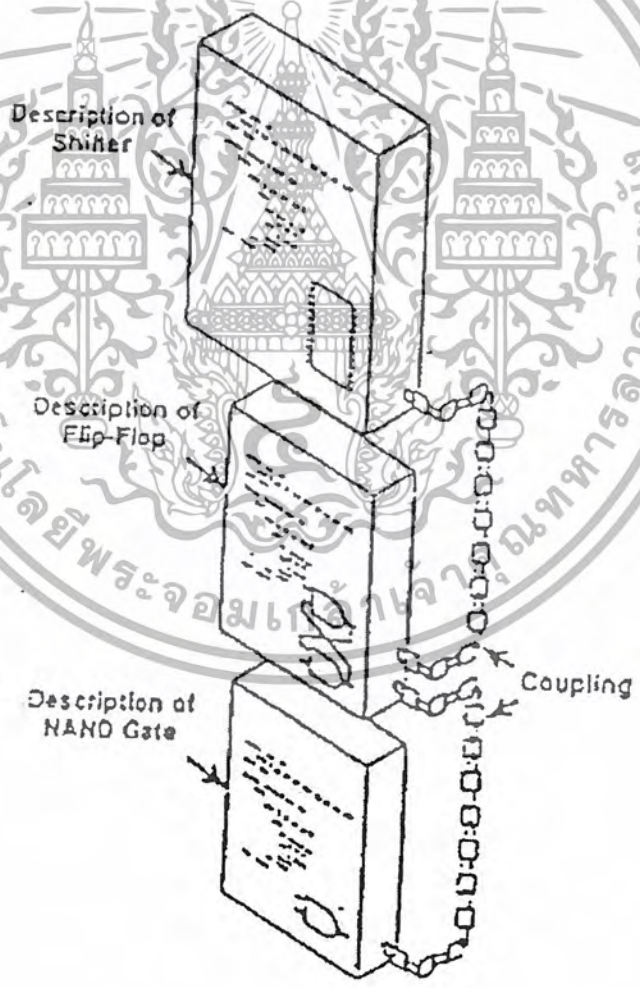
3.3.2 Modularity

Modularity คือ หลักการในการแยกส่วน (Partitioning) ฮาร์ดแวร์ ออกเป็นส่วนย่อยเล็กลงไป ซึ่งปกติการทำงานของฮาร์ดแวร์ใหญ่ๆ ต้องประกอบด้วยฮาร์ดแวร์ย่อยๆ ลงไปดังรูปที่ 3.3

รูปที่ 3.3 การแบ่งย่อยในระดับราบของการออกแบบฮาร์ดแวร์

จากรูปได้แสดงวงจรทั้งหมดในรูปเดียว (Flatten Design) หลังจากนั้นตัดเป็นส่วนย่อยๆ เล็กลงมา เมื่อเราออกแบบโดยใช้ภาษา VHDL หน้าที่การทำงานของแต่ละส่วนต้องสามารถอธิบายได้โดยโมดูลของโค้ด(คล้ายฟังก์ชั่นหรือโพรซีเดอร์) ซึ่งแสดงการทำงานของส่วนย่อยนั้นอย่างชัดเจน ซึ่งการแยกรูปใหญ่ๆ ออกเป็นส่วนย่อยๆ นี้ ทำให้ง่ายต่อการจัดการและง่ายต่อการทำความเข้าใจ

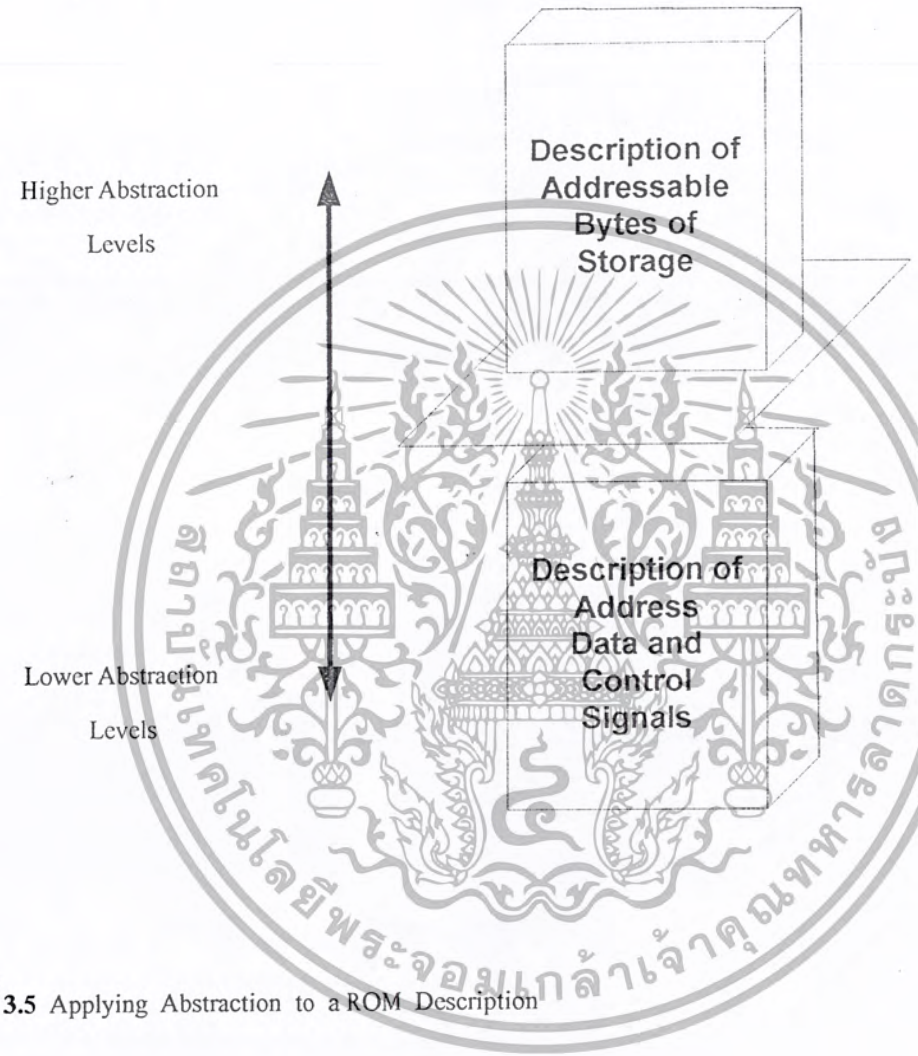
รูปที่ 3.3 แสดง Hierarchy Method โดยการแยกส่วนรูปแบบออกเป็นส่วนย่อยๆ ส่วนบนสุดอธิบายการทำงานของ Shifter ส่วนล่างๆ ลงมา คือการแยกของ Shifter ออกเป็นฟลิบฟลอบจาก ฟลิบฟลอบแยกเป็น NAND เกท ภายใน Shifter ได้อธิบายการทำงานโดยใช้การต่อกันของ ฟลิบฟลอบก็เกิดจาก NAND เกท ต่อกัน 2 ตัวในระดับที่ต่ำลงมาอีกก็เป็น NAND เกท ซึ่งมีการอธิบายการทำงานอยู่ภายใน โดยแต่ละโมดูลจะมีคำอธิบายการทำงานในตัวของมันเองอยู่แล้วคำอธิบายในแต่ละโมดูลมีไว้เพื่อสามารถใช้ฟลิบฟลอบโมดูลก็อธิบายการเชื่อมต่อไว้อย่างดีทำให้สามารถเชื่อมต่อกับ NAND เกท ในระดับล่างสุดได้ ประโยชน์อย่างหนึ่งของการแยกส่วนฟลิบฟลอบและ NAND เกท ออกจากกันเนื่องจากทำให้ง่ายในการใช้ NAND เกท ตัวนี้ในรูปแบบไฮเลเวลตัวอื่นๆ ทำให้ออกไปใช้งานได้อีกและลดความซับซ้อนในการการใช้อุปกรณ์เพื่อ แก้ไข การทำงานของ Shifter ง่ายขึ้นโดยปราศจากการแก้ไขฟลิบฟลอบและ NAND เกท ประโยชน์ที่ได้จากการทำ Modularity นี้ ทำให้รูปแบบที่ออกแบบง่ายต่อการเข้าใจและแก้ไขได้เสมอ



รูปที่ 3.4 การแบ่งแบบ Hierarchy ของ VHDL Shifter Description

3.3.3 Abstraction

คำนิยามของรูปแบบ จะอธิบายการทำงานของตัวรูปแบบมากกว่าจะอธิบายว่าพัฒนาตัวรูปแบบนั้นได้อย่างไร หลักการนี้มีความสำคัญอย่างใกล้ชิดกับหลักการของ Modularity ในรูปที่ 3.4 ฟลิบฟลอบ เป็นการนิยามในการใช้ NAND เกท และ Shifter เป็นนิยามในการใช้ฟลิบฟลอบ

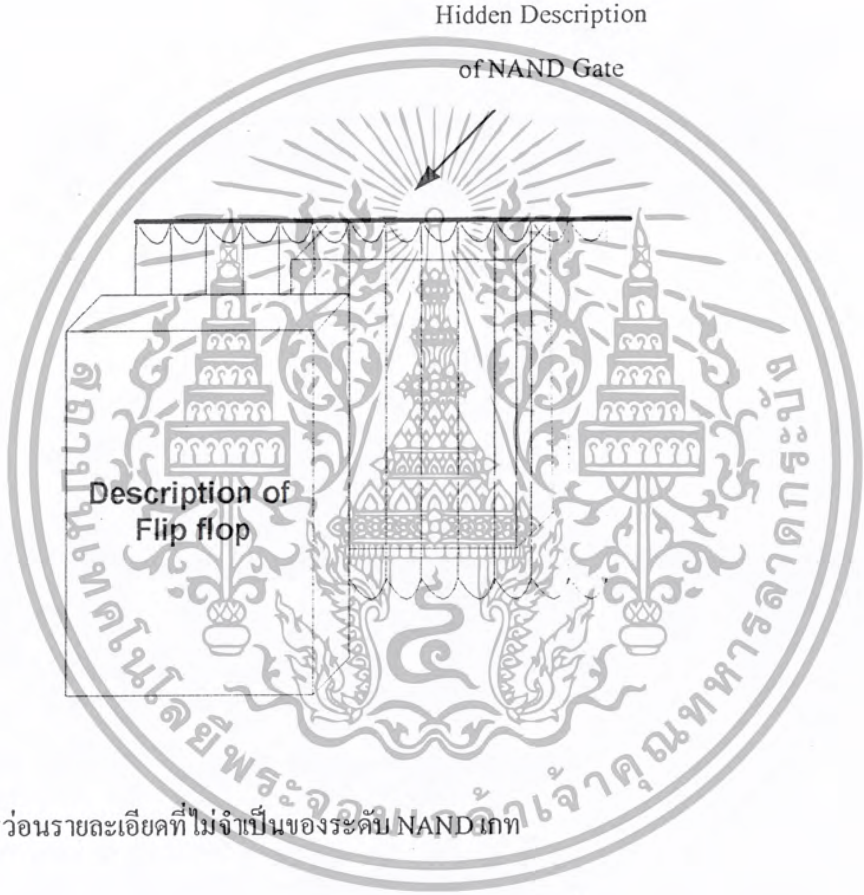


รูปที่ 3.5 Applying Abstraction to a ROM Description

รูป3.5 แสดงถึงการอธิบายการทำงานของรูปแบบโดยใช้ VHDL ในหลายๆ ระดับของการนิยาม ROM (Read Only Memory) อธิบายโดยใช้ภาษาระดับสูง แสดงถึงตำแหน่งต่างๆ ซึ่งเก็บข้อมูลไว้ในตำแหน่งนั้นๆ ที่ระดับนี้ไม่ต้องสนใจถึง Address Line , Data Line และ Control line เราสามารถพุ่งจุดสนใจไปที่ขนาดของข้อมูลโดยไม่ต้องคำนึงสัญญาณควบคุมต่างๆ ภายในเพราะว่าส่วนนั้นจะถูกจัดการเองในระดับที่ต่ำลงมา ในระดับล่างลงมาเราสามารถอธิบายการทำงานของสัญญาณแต่ละเส้นภายในการที่จะอ่านข้อมูลหรือโปรแกรมข้อมูลใน ROM ถ้าต้องการเปลี่ยนค่าข้อมูลภายใน ROM ควรแก้ระดับที่สูงขึ้นมาจะง่ายกว่าการควบคุมสัญญาณภายใน จะเห็นว่าแต่ละระดับมีความเหมาะสมแตกต่างกันออกไปง่ายต่อการแก้ไขโดยการใช้ประโยชน์ของ Abstraction

3.3.4 Information Hiding

เมื่อทำการเขียน VHDL Code ขึ้นมาเพื่ออธิบายการทำงานของฮาร์ดแวร์ตัวหนึ่ง บางครั้งอาจต้องการที่จะซ่อนรายละเอียดการพัฒนาโมดูล นั้นโดยไม่ต้องการให้ส่วนโมดูลอื่นๆ รู้การทำงานภายใน Information Hiding มีประโยชน์คือ ทำให้รูปแบบภาษา VHDL นั้นสามารถจัดการและอ่านเข้าใจได้ง่าย หลักการนี้จะสนับสนุนหลักการ Abstraction คือสนใจรายละเอียดในการใช้งานมากกว่าจะสนใจว่ารูปแบบนั้นจะถูกสร้างขึ้นมาอย่างไร เป็นต้น การซ่อนรายละเอียดภายในโมดูลทำให้ความสนใจของผู้ออกแบบนั้นสนใจไปในส่วนที่สำคัญมากกว่าในส่วนที่ไม่น่าสนใจจะซ่อนไว้และเข้าถึงไม่ได้ ดังรูปที่3.6



รูปที่ 3.6 การว่อนรายละเอียดที่ไม่จำเป็นของระดับ NAND เกท

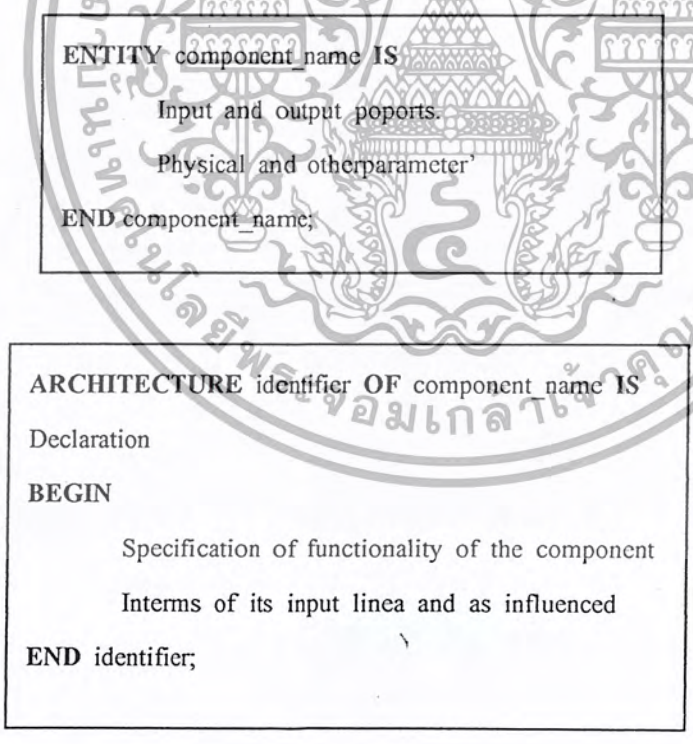
อธิบายการทำงานของฟลิบฟลอบไม่ต้องสนใจว่า NAND เกท จะทำงานอย่างไร จะต่อกันภายในอย่างไร โดย NAND เกท สามารถเขียนขึ้นแล้วคอมไพล์เก็บไว้ในไลบรารี ผู้ที่ออกแบบฟลิบฟลอบระดับสูงขึ้นมา เพียงแต่ต้องรู้ว่า จะเชื่อมตออินพุท/เอาท์พุทของ NAND เกท มาใช้งานได้อย่างไร โดยไม่ต้องสนใจว่า NAND เกท จะถูกสร้างและพัฒนามาอย่างไร ประโยชน์อีกอย่างหนึ่งคือป้องกันข้อมูลภายใน ในกรณีที่แจกจ่าย VHDL โมเดลไปยังที่อื่นทำให้เราป้องกันทรัพย์สินทางปัญญาได้อีกระดับหนึ่ง

3.3.5 Uniforminity

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

3.4 องค์ประกอบพื้นฐานในวีเอชดีแอล (Basic concept in VHDL)

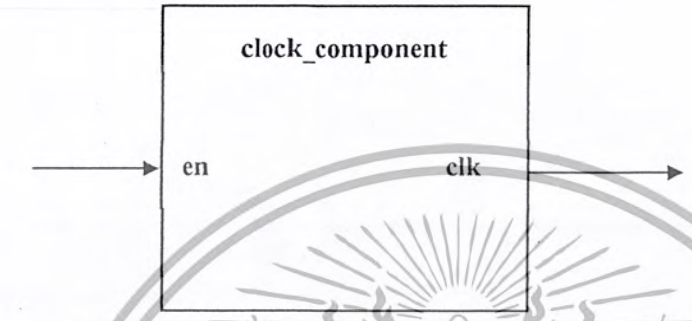
รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบใน VHDL ประกอบด้วยส่วนกำหนดการเชื่อมต่อ(Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม(Architecture) ดังแสดงในรูปที่ 3.7 การบรรยายเชื่อมต่อจะขึ้นต้นด้วยคำ ENTITY ตามด้วยชื่อขององค์ประกอบและคำ IS ภายในบรรยายถึงพอร์ตการติดต่อ อินพุท/เอาท์พุท พอร์ตขององค์ประกอบ ส่วนลักษณะกายภาพนอกอื่นๆ เช่น เวลา อุณหภูมิ ก็สามารถเข้าไปในส่วนนี้ได้เช่นกัน ในส่วนของการกำหนดลักษณะสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้บรรยายหน้าที่การทำงานขององค์ประกอบ หน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณ อินพุท/เอาท์พุท และพารามิเตอร์อื่นๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูป 3.7 การบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังคำว่า BEGIN เป็นต้นไป



รูปที่ 3.7 แสดงการกำหนดเชื่อมต่อและสถาปัตยกรรม

3.4.1 การกำหนดการเชื่อมต่อ (Interface Description)

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ ในระดับนี้จะต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบภายนอกอื่นๆ ดังตัวอย่างในรูปที่ 3.8 บรรทัดแรกเป็นการกำหนดชื่อขององค์ประกอบซึ่งกำหนดให้เป็นชื่อ clock component ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ในวงเล็บ IN และ OUT กำหนดโหมดของสัญญาณเป็นอินพุทหรือเอาท์พุท BIT แสดงชนิดของข้อมูล




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

รูปที่ 3.8 แสดงบล็อคไดอะแกรมและการบรรยายการเชื่อมต่อของ clock component

3.4.2 การกำหนดรูปแบบของการบรรยาย (Architecture Description)

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ การบรรยายสามารถกำหนดค่าของสัญญาณเอาท์พุทในเทอมของ clock component ในรูปที่ 3.9 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมมี en เป็นอินพุทและมี clk เป็น PROCESS เป็นค่าเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม ภายในโพรเซสกำหนดให้ periodic เป็นตัวแปรทีมีค่าเริ่มต้นเป็น “0” ถ้าสัญญาณ en มีค่าเป็น “1” ค่าของ periodic ถูกคอมพลีเมนต์และส่งค่าให้กับ clk ซึ่งเป็นสัญญาณเอาท์พุทคำสั่ง WAIT กำหนดให้สัญญาณทีคาบเป็นเวลา 1 ไมโครวินาที

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

รูปที่ 3.9 แสดงการบรรยายเชิงพฤติกรรมของ Clock_component

3.4.3 หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย ที่เป็นประโยชน์ต่อการเขียนรูปแบบการบรรยายระบบดิจิตอล สามารถเก็บไว้ใน ส่วนของแพ็กเก็ต ซึ่งหน่วยการออกแบบต่างๆ เช่น หน่วยการออกแบบ Entity หน่วยการออกแบบสถาปัตยกรรมหรือ หน่วยการออกแบบแพ็กเกจอื่นๆ สามารถเรียกข้อมูลเหล่านี้ไปใช้ได้ นอกจากนั้นสิ่งที่นิยมทำกันมากคือการนำรูปแบบ มาตรฐานต่างๆ เช่น อุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74XX เป็นต้น) มาเก็บไว้ในรูปของแพ็กเกจ ที่ทุกคนสามารถ เข้าถึงได้ ตามปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (Package declaration)และ ส่วนของบอดี้แพ็กเกจ (Package body) เนื่องจาก แพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่ กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษา VHDL สามารถ กระทำได้ด้วยชุดคำสั่ง

3.4.3.1 PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจ สำหรับนำไปใช้ภายนอกตัวของแพ็กเกจเอง ถ้ามี การประกาศสิ่งใดๆ ในส่วนของส่วนบอดี้แพ็กเกจ แต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่าและพฤติ กรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือ จุดเชื่อมต่อ หรือ พอร์ต ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นได้โดยไม่จำเป็น ต้องมีส่วนบอดี้และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศ ชนิด (Type) หรือสัญญาณ

เช่นเดียวกับ ส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมี ส่วนของการประกาศแพ็กเกจ แต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จาก รูปแบบอื่นได้

```
Package package_name Is
        Package_declarative_part
End package_name;
```

รูปที่ 3.10 โครงสร้างทั่วไปของส่วนการประกาศแพ็กเกจ

3.4.3.2 PACKAGE BODY

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้วในส่วนของการประกาศแพ็กเกจ จะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึง การกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อนในส่วนของการประกาศแพ็กเกจ และถูกกำหนดค่าใน ส่วนของบอดี้แพ็กเกจ ฉะนั้นในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการ ประกาศชื่อที่เป็นโปรแกรมย่อยหรือค่าคงที่ การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในรูปที่ 3.11

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

รูปที่ 3.11 โครงสร้างของบอดี้แพ็กเกจ

3.4.4 หน่วยการออกแบบ Configuration

ดังที่ทราบกันแล้วว่าระบบดิจิตอลรูปแบบหนึ่งไม่ว่าจะเป็นอะไรก็ตาม จะสามารถมีหน่วยการออกแบบ Entity ได้เพียงหนึ่งเดียวเท่านั้น ซึ่งในหน่วยการออกแบบ Entity หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลาย หน่วย ดังนั้นจะต้องมีหน่วยการออกแบบ Configuration มาเพื่อกำหนดการใช้ Configuration ของการประกอบ Entity กับหน่วยการออกแบบสถาปัตยกรรมหน่วยใดๆ เข้าด้วยกัน

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

รูปที่ 3.12 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

3.4.5 โปรแกรมย่อย (Subprogram)

การใช้ฟังก์ชั่นและโพรซีเดอร์ใน VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไป ค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่น ถ้าเราให้ฟังก์ชั่นแทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรตรรกะจริงๆ ในขณะที่เราใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูลหรือในการคำนวณค่า หน่วงเวลา แล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์

รูปที่ 3.13 แสดงการใช้ฟังก์ชั่น โพรซีเดอร์เพื่อเปลี่ยนข้อมูลชนิด 8 บิด เป็นค่าจำนวนเต็ม รูปที่ 3.14 แสดงการใช้ฟังก์ชั่นโดยกำหนดให้ X เป็นตัวแปรชนิดบิดแทนการกระทำในสมการบูลีน

```
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;

End byte_to_integer;
```

รูปที่ 3.13 แสดงการใช้โพรซีเดอร์

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

รูปที่ 3.14 แสดงการใช้ฟังก์ชัน

3.4.6 โอเปอร์เรเตอร์ (VHDL OPERATORS)

การบรรยายเชิงพฤติกรรมใน VHDL ก็มีตัวกระทำทางลอจิกและคณิตศาสตร์เช่นเดียวกันกับภาษาซอฟแวร์ทั่วไปดังรูปที่ 3.15

PREDEFINED OPERATORS

LOGICAL OPERATOR : NOT AND OR NAND NOR XOR

OPERAND TYPE : BIT BOOLEAN

RESULT TYPE : BIT BOOLEAN

RELATIONAL OPERATORS : = /= < <= > >=

OPERAND TYPE : ANY TYPE

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 3.15 แสดงตัวกระทำใน VHDL

3.4.7 เวลาและความพร้อมเพรียง (Timing and Concurrency)

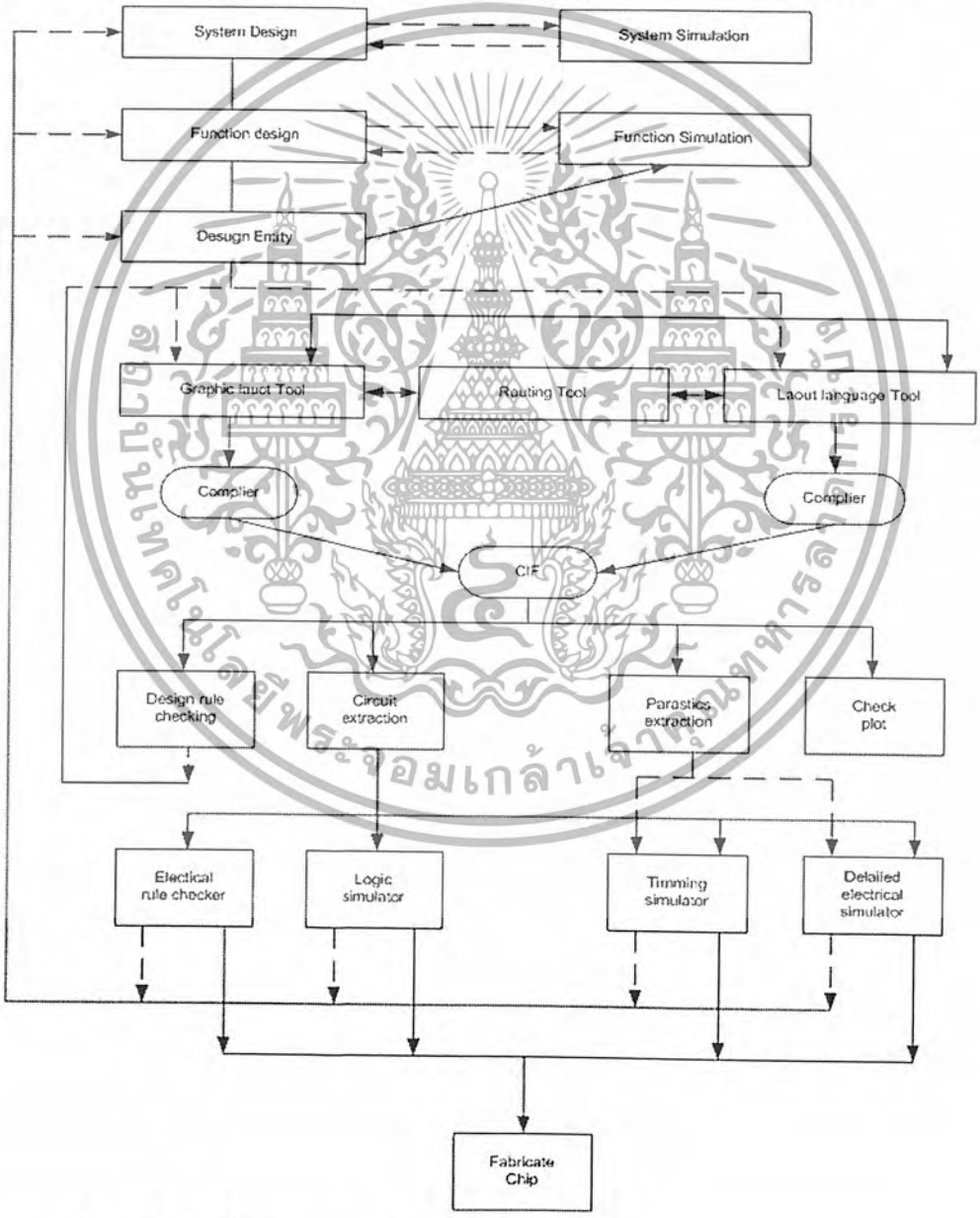
ในวงจรอิเล็กทรอนิกส์ อุปกรณ์ทุกตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ(always active) และจะมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยเสมอในทุกๆ เหตุการณ์ที่เกิดขึ้น VHDL เป็นภาษาที่ได้รับการออกแบบมาเพื่อสามารถบรรยายรูปแบบและการพ้องกันของเวลาสำหรับการทำงานของอุปกรรณ์ได้อย่างถูกต้อง การบรรยายการทำงาน ที่อยู่ภายในส่วนของสถาปัตยกรรมการบรรยายจะมีการทำงานที่พร้อมเพรียงกันเสมอหรือแม้แต่โพรเซสที่มีการทำงานภายในเป็นแบบลำดับคำสั่งก็ตาม หากมีหลายๆโพรเซสอยู่ภายในโครงสร้างเดียวกันทุกๆ โพรเซส ก็จะทำงานไปพร้อมๆ กันด้วย

3.4.8 สัญญาณและตัวแปร (Signals and Variable)

ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวคิดเบื้องต้นจนกระทั่งได้ออกมาเป็นอุปกรณ์ฮาร์ดแวร์ที่ใช้งานได้ จะต้องผ่านขั้นตอนต่างๆ มากมายและในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์สุดท้ายในแต่ละขั้น ทำการเพิ่มเติมตามความจำเป็นและเข้ากระบวนการออกแบบในขั้นต่อไป

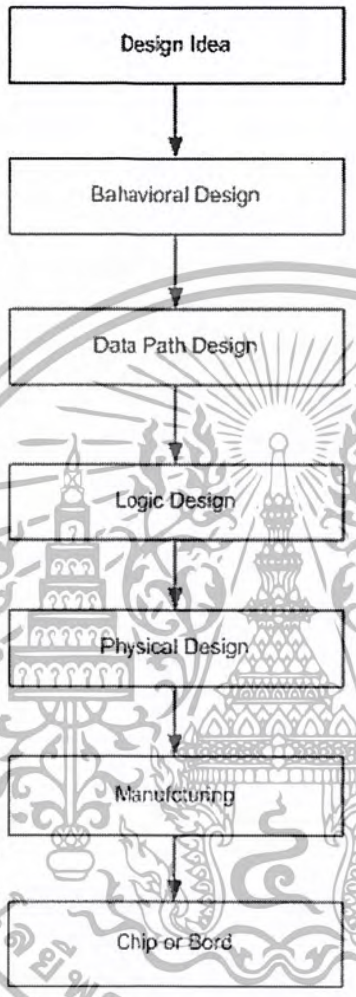
3.5 โครงสร้างของวีเอชดีแอล

ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวความคิดเบื้องต้นจนกระทั่งได้ออกมาเป็น อุปกรณ์ฮาร์ดแวร์ที่ใช้งานได้ จะต้องผ่านขั้นตอนต่างๆ มากมายและในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์สุดท้ายในแต่ละขั้น ทำการเพิ่มเติมตามความจำเป็นและเข้ากระบวนการออกแบบในขั้นต่อไป โครงสร้างการออกแบบวีเอชดีเอลสามารถเขียนเป็นแผนผังดังรูปที่3.16



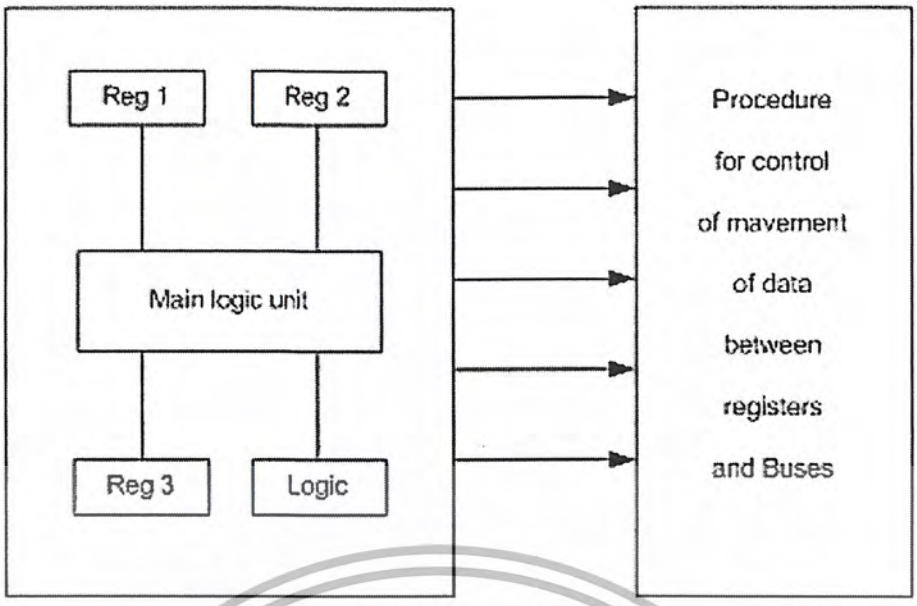
รูปที่ 3.16 รูปแสดงโครงสร้างการออกแบบวงจรรวมโดยใช้ VHDL

รูปที่ 3.17 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิตอลทั่วๆไป ซึ่งขั้นแรกผู้ออกแบบกำหนดแนวคิดในการออกแบบเสียก่อนและทำการพัฒนาให้สามารถนำมาใช้ได้อย่างสมบูรณ์ ดังนั้นในขั้นตอนนี้จึงจำเป็นที่ผู้ออกแบบจะต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบซึ่งอาจเป็นผังงาน ผังแสดงแบบหรือรหัสคำสั่งเทียม (Pseudo Code)



รูปที่ 3.17 แสดงขั้นตอนการออกแบบระบบดิจิตอล

ขั้นตอนต่อไปเป็นการออกแบบระบบเส้นทางของข้อมูล ผู้ออกแบบจะต้องกำหนดส่วนประกอบรีจีสเตอร์ ผู้ออกแบบจะกำหนดส่วนของรีจีสเตอร์(Register) และวงจรตรรกะ (Logic) ที่จำเป็นทั้งหมดที่ประกอบกันเป็นระบบที่สมบูรณ์ แต่ละองค์ประกอบสามารถเชื่อมต่อกันด้วยบัสหนึ่งหรือสองทิศทาง(Unidirectional or Bidirectional Bus) กระบวนการควบคุมในการเคลื่อนย้ายข้อมูลระหว่างรีจีสเตอร์และวงจรตรรกะจะขึ้นอยู่กับพฤติกรรมของระบบที่กำหนดไว้ ดังรูปที่ 4.15



รูปที่ 3.18 แสดงการออกแบบระบบเส้นทางของข้อมูล

การออกแบบวงจรตรรกะจะเป็นขั้นตอนต่อไป การออกแบบในขั้นนี้เกี่ยวข้องกับการใช้เกทพื้นฐานและฟลิบฟลอบเป็นส่วนของอุปกรณ์แยกต่างๆ ได้แก่ รีจีสเตอร์เก็บข้อมูลวงจรตรรกะและส่วนควบคุมฮาร์ดแวร์ ในสุดท้ายจะได้ออกมาเป็นเครือข่ายของการโยงใยระหว่างเกทและฟลิบฟลอบนั่นเอง ต่อมาเป็นขั้นตอนของการเปลี่ยนเครือข่ายการโยงใยในขั้นตอนที่แล้วให้เป็นทรานซิสเตอร์และเลย์เอาท์ (Transistor list and layout) ในขั้นตอนนี้เกี่ยวข้องกับการจัดวางเกทและฟลิบฟลอบแทนด้วยทรานซิสเตอร์หรือไลบรารีเซล ขั้นตอนสุดท้ายเป็นการส่งระบบที่ออกแบบไว้ไปทำการเจือสารที่โรงงานเพื่อผลิตออกมาเป็นวงจรรวมในที่สุด

บทที่ 4
ระบบการเข้ารหัสลับ(Cryptosystem)

4.1 หลักการเบื้องต้นของระบบเข้ารหัสลับ

ระบบการเข้ารหัสลับ คือ การนำข้อมูลข่าวสาร ไปแปลงจนทำให้มีคนเพียงจำนวนหนึ่งเท่านั้นที่สามารถเข้าใจถึงความหมายเดิมได้ เราเรียกข้อมูลเดิม, ขบวนการแปลงข่าวสารและข่าวสารที่ถูกแปลงแล้วว่า เอกสารปกติ(Plaintext) การเข้ารหัสลับ(Encryption) และ เอกสารลับ(Ciphertext) ตามลำดับ ส่วนขบวนการที่ใช้ในการแปลงเอกสารลับให้กลับเป็นเอกสารเดิมเราเรียกว่า การถอดรหัส(Decryption) การเข้ารหัสลับจะประกอบด้วยอัลกอริทึมและตัวแปรต่างๆ ที่เรามักแทนด้วยสัญลักษณ์ ที่เรียกว่า "กุญแจรหัส(Key)" ในการสื่อสารถึงแม้ว่าช่องสัญญาณจะถูกดักฟัง(Wiretap) หากผู้ดักฟังไม่มีกุญแจรหัสก็ไม่สามารถจะถอดรหัสกลับมาเป็นเอกสารปกติเดิมได้โดยง่าย

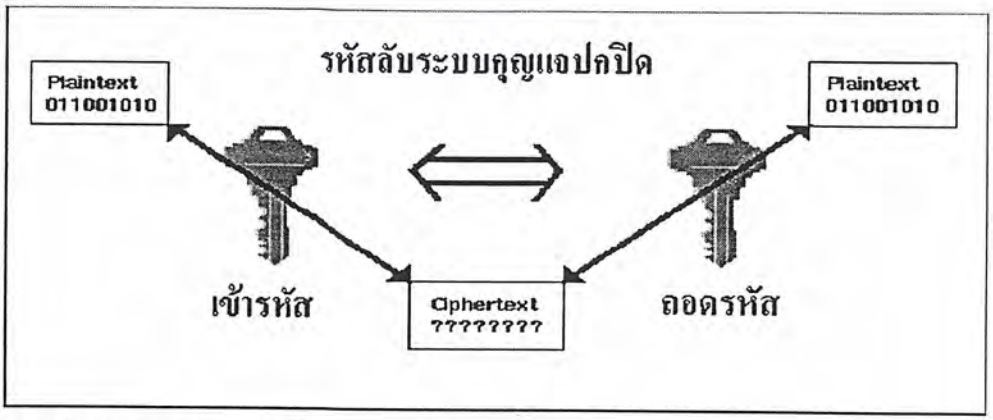
การโจมตี(Attack) เพื่อทำลายรหัส คือ การที่ผู้บุกรุกพยายามที่จะถอดรหัสลับที่ได้จากการดักฟังโดยไม่มีกุญแจรหัส แต่อาจมีข้อมูลบางส่วนที่เกี่ยวกับเอกสารที่ถูกเข้ารหัส ความยากง่ายของการทำลายรหัสนั้นขึ้นอยู่กับข้อมูลที่สามารถหาได้และวิธีการต่างๆที่ใช้ในการวิเคราะห์รหัส(Crytanalysis) ระบบการเข้ารหัสลับที่ดีนั้นความปลอดภัยจะต้องขึ้นอยู่กับกุญแจรหัสมากกว่าอัลกอริทึมที่ใช้ในการเข้ารหัส รหัสลับที่ถือว่ามีความปลอดภัยสูงจะต้องมีคุณสมบัติต่างๆ เช่น ต้องใช้กุญแจที่มีขนาดใหญ่ (เช่น จำนวนบิทมาก), ต้องมีภูมิต้านทานต่อการวิเคราะห์รหัสที่มีประสิทธิภาพชนิดต่างๆ ที่รู้จักกันในปัจจุบัน ฯลฯ อย่างไรก็ดีรหัสที่มีคุณสมบัติเหล่านี้ครบไม่จำเป็นต้องเป็นรหัสที่มีความปลอดภัยสูง รหัสลับจำนวนไม่น้อยที่ดูเหมือนจะมีความปลอดภัยสูงในครั้งแรกแต่ทำลายอย่างง่ายๆ ในภายหลัง

การเข้ารหัสลับในยุคใหม่ได้รับความสนใจอย่างสูงเมื่อการเข้ารหัสแบบ DES (Data Encryption Standard) ปรากฏตัวขึ้นการที่รัฐบาลสหรัฐยอมรับอย่างเป็นทางการให้ DES เป็นมาตรฐานการเข้ารหัสลับยิ่งเป็นการกระตุ้นให้มีการศึกษาและวิจัยเกี่ยวกับการเข้ารหัสลับอย่างมาก หลังจากนั้นการค้นพบรหัสลับระบบกุญแจสาธารณะรวมทั้งการนำรหัสลับมาใช้ในกระบวนการพิสูจน์ความเป็นเจ้าของ (Athentication) ทางอิเล็กทรอนิกส์ ยิ่งทำให้ขอบเขตการใช้งานขยายวงกว้างออกไปอย่างมาก

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

4.2 รหัสลับระบบกุญแจปกปิด

รหัสลับระบบกุญแจปกปิดเป็นรหัสลับแบบดั้งเดิมที่ใช้กันมานานแล้ว ในระบบนี้ทั้งผู้ส่งและผู้รับจะต้องมีกุญรหัสที่เหมือนกันโดยที่กุญแจนี้จะต้องถูกปกปิดเป็นความลับระหว่างผู้ส่งและผู้รับเท่านั้น ผู้ส่งจะใช้กุญแจรหัสในขบวนการเข้ารหัสส่วนผู้รับจะใช้กุญแจรหัสในการถอดรหัส



รูปที่ 4.1 รหัสลับระบบกุญแจปกปิด

4.2.1 หลักการการทำงานของรหัสลับระบบกุญแจปกปิด

การเข้ารหัสลับระบบดิจิตอลในปัจจุบันเป็นแบบที่เรียกว่าการเข้ารหัสแบบบล็อค(Block Cipher) การเข้ารหัสลับแบบบล็อคคือการเข้ารหัสลับโดยการแบ่งข้อมูลออกเป็นบล็อค โดยที่แต่ละบล็อคมีขนาดเท่ากันโดยเครื่องเข้ารหัสจะทำการเข้ารหัสด้วยอัลกอริทึมเดียวกันทีละบล็อค โดยทั่วไปแล้วยิ่งขนาดของบล็อคใหญ่เท่าไหร่ก็จะสามารถสร้างรหัสที่ปลอดภัยได้มากยิ่งขึ้น เมื่อเกือบ 50 ปีที่แล้ว C.E. Shannon ปรมาจารย์ผู้ให้กำเนิดวิชา "ทฤษฎีข่าวสาร(Information Theory)" ได้ให้นิยามหลักการสำคัญของการเข้ารหัสลับแบบบล็อคไว้ ซึ่งยังคงใช้ได้จนกระทั่งปัจจุบัน ดังต่อไปนี้

โดยทั่วไปแล้วรหัสลับจะมีฟังก์ชันการทำงานที่สำคัญ 2 อย่างคือ คอนฟิวส์ชัน(Confusion) และ ดิฟฟิวส์ชัน(Diffusion) ในการเข้ารหัส คอนฟิวส์ชันของรหัสจะทำหน้าที่ซ่อนความสำพันธ์ระหว่างส่วนประกอบสำคัญ 3 ส่วน คือเอกสารปกติ เอกสารลับ และกุญแจรหัส คอนฟิวส์ชันของรหัสที่ดีจะต้องทำให้ความสำพันธ์เหล่านี้ยุ่งยากในเชิงสถิติ จนเครื่องมือวิเคราะห์รหัสที่มีประสิทธิภาพดีเพียงใดก็ไม่สามารถทำงานได้ ส่วนหน้าที่ของดิฟฟิวส์ชัน คือกระจายผลที่แต่ละบิตของกุญแจและเอกสารปกติที่มีต่อเอกสารลับให้มากที่สุดเท่าที่จะทำได้ดิฟฟิวส์ชันยังมีผลในการซ่อนความสัมพันธ์ในเชิงสถิติซึ่งทำให้การวิเคราะห์รหัสยากขึ้นด้วย

ที่จริงแล้วการทำคอนฟิวส์ชันเพียงอย่างเดียวก็เพียงพอแล้วสำหรับรหัสที่ดี นั่นคือเราสามารถสร้างรหัสลับที่ยากต่อการวิเคราะห์โดยการใช้วิธีการเปิดตรง (Look-up Table) เพียงอย่างเดียวซึ่งก็คือการใช้หน่วยความจำในการเข้ารหัสนั่นเอง ปัญหาอยู่ที่ว่าเมื่อขนาดที่ใช้แบ่งเอกสารมีขนาดใหญ่ขึ้นขนาดของหน่วยความจำที่ต้องการใช้ก็มากขึ้นจนไม่เหมาะสมในทางปฏิบัตินั่นเอง

ตรงนี้เองที่ดิฟฟิวส์ชันถูกนำมาแก้ปัญหา โดยในการเข้ารหัสแต่ละครั้งนั้น แทนที่จะใช้ตาราง (หน่วยความจำ) ขนาดใหญ่เพียงอย่างเดียว เราจะทำการคอนฟิวส์ชันโดยใช้ตารางที่เล็กกว่าแล้วนำมาผสมกับการดิฟฟิวส์ชันหลายๆครั้งด้วยรูปแบบต่างๆกัน การดิฟฟิวส์ชันอาจทำได้ง่ายๆ โดยการสลับตำแหน่ง (Permutation) เท่านั้น การเข้ารหัสลับแบบบล็อคที่อธิบายดังกล่าวเรียกว่า Product Cipher รหัสลับระบบกุญแจปกปิดที่มีใช้กันอยู่ในปัจจุบันส่วนใหญ่จะใช้หลักการของ Product Cipher ในจำนวนนี้รหัส DES เป็นรหัสลับที่แพร่หลายมากที่สุด และยังคงได้รับความเชื่อถือจนกระทั่งปัจจุบัน

4.2.2 รหัสลับระบบกุญแจปกปิดแบบ DES

รหัส DES ก็เป็น Product Cipher ชนิดหนึ่งซึ่งถูกพัฒนาโดยบริษัทไอบีเอ็ม หลังจากนั้นจึงได้รับการยอมรับให้เป็นมาตรฐานการเข้ารหัสของรัฐบาลสหรัฐ DES มีข้อมูลขาเข้าและขาออกเท่ากันคือ 64 บิท มีกุญแจรหัสความยาว 56 บิท นอกจากหลักการของคอนฟิวส์ชันและดิฟฟิวส์ชันของ Product Cipher แล้ว DES ยังใช้หลักการพิเศษบางประการในการออกแบบหนึ่งในหลักการนั้นคือการทำขบวนการง่ายๆซ้ำไปมาหลายๆรอบเพื่อเพิ่มความแข็งแกร่งของรหัส

ลักษณะการทำซ้ำไปมาในแต่ละรอบของ DES ใช้หลักการที่เรียกว่า Fiestel Cipher ในการเข้ารหัสแบบ Fiestel Cipher นั้นในแต่ระบบบล็อคข้อมูลความยาว n บิตจะถูกแบ่งออกเป็น 2 ส่วน คือ L และ R โดยในแต่ละส่วนมีความยาว n/2 บิตเท่ากัน ความสัมพันธ์ของข้อมูลขาออกในรอบที่ i กับi-1 ดังต่อไปนี้

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

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

โดยที่ XOR คือการทำ exclusive or, K_i คือส่วนของกุญแจที่ถูกใช้ในรอบที่ i และ f คือฟังก์ชันที่ใช้ในแต่ละรอบ การเลือกฟังก์ชัน f นั้นมีผลต่อความแข็งแกร่งต่อการวิเคราะห์รหัส เพราะ f คือการคอนฟิวส์ชันนั่นเอง ส่วนการทำซ้ำคือการดิฟฟิวส์ชัน ซึ่งยิ่งทำมากรอบเท่าไหร่ความสัมพันธ์ที่แต่ละบิตของกุญแจรหัสและข้อมูลขาเข้าจะถูกกระจายไปยังข้อมูลขาออกมากยิ่งขึ้น สำหรับ DES นั้นจำนวนรอบที่ทำคือ 16 รอบ

ข้อดีของ DES ที่สำคัญที่สุดประการหนึ่งก็คือ ความง่ายของวงจรเข้าและถอดรหัส จาก (4.1) และ (2) จะเห็นว่าในแต่ละรอบมีการทำงานที่เหมือนกันจึงง่ายต่อการออกแบบวงจรให้รวดเร็วและประหยัด ยิ่งไปกว่านั้น จาก(4.1) และ (4.2) เราจะได้ว่า

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

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

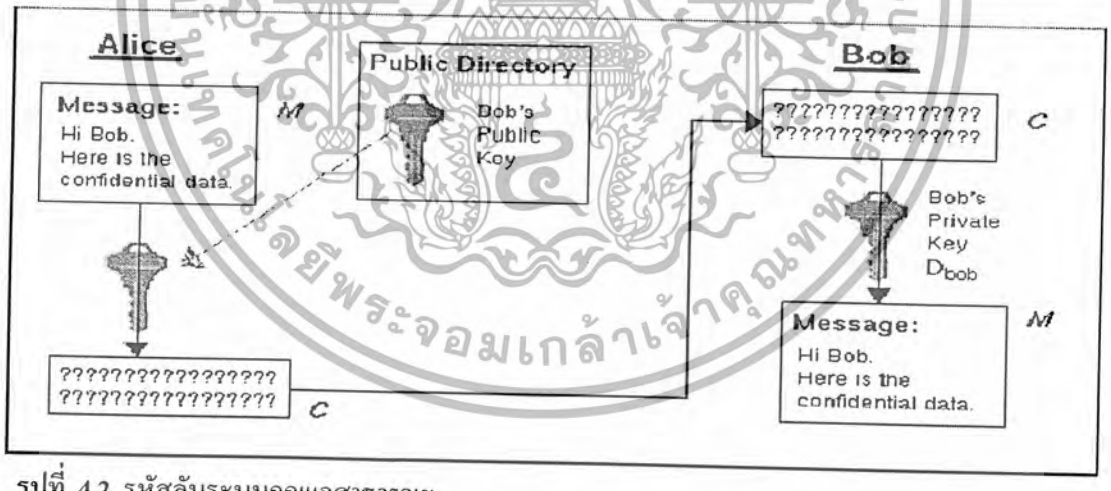
ถึงแม้ว่าอัลกอริทึมในการเข้ารหัสของ DES จะถูกเผยแพร่ต่อสาธารณะชนมากว่า 15 ปีแล้วก็ตามและได้มีการพยายามที่จะทำลายรหัส DES อย่างนับไม่ถ้วน แต่ DES ยังคงได้รับการยอมรับจนกระทั่งปัจจุบันว่าเป็นการเข้ารหัสลับที่มีความปลอดภัยสูงมากชนิดหนึ่ง ในปี 1993 ได้มีการประมาณว่าจะต้องใช้เงินประมาณ 1 ล้านดอลลาร์เพื่อเป็นค่าใช้จ่ายในการสร้างเครื่องคอมพิวเตอร์เฉพาะทางที่ใช้ในการวิเคราะห์ DES และสำหรับการส่งข้อความที่ปลอดภัยสูงมากนั้น การใช้ DES ต่อกันเป็นโครงข่าย เช่น Tripple-DES ซึ่งใช้การเข้ารหัสแบบ DES 3 รอบโดยใช้กุญแจ 3 ดอกจะยิ่งเพิ่มความแข็งแกร่งให้กับ DES เป็นอย่างมาก

ปัญหาใหญ่ของรหัสลับในระบบกุญแจปกปิด คือการที่ทั้งผู้รับและผู้ส่งจะต้องใช้กุญแจที่เหมือนกันโดยกุญแจนั้นจะต้องเก็บเป็นความลับปล่อยให้ตกอยู่ในมือของคนอื่นไม่ได้ ดังนั้นหากผู้รับและผู้ส่งอยู่ห่างจากกันในทางกายภาพ เราจำเป็นต้องให้บุคคลที่สามที่ไว้ใจได้หรือช่องสื่อสารที่ปลอดภัยในการส่งผ่านกุญแจ เพราะถ้าหากกุญแจนี้ถูกแอบลักลอบดักฟังหรือเปิดอ่านระหว่างการขนส่งแล้ว ผู้ลักลอบดักฟังก็สามารถถอดรหัสลับได้ 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 การทำงานของกุญแจรหัสลับระบบกุญแจสาธารณะสามารถอธิบายการทำงานได้ดังนี้



รูปที่ 4.2 รหัสลับระบบกุญแจสาธารณะ

จากรูปที่ 4.2 สมมติว่า Alice ต้องการส่งข้อความถึง Bob ก่อนอื่น Alice ต้องมีกุญแจสาธารณะของ Bob ซึ่งอาจจะขอจาก Bob โดยตรงหรือจากองค์กรที่ให้บริการ จากนั้น Alice ก็จะใช้กุญแจสาธารณะของ Bob ในการเข้ารหัสลับแล้วส่งไปให้ Bob จากนั้น Bob จะถอดรหัสลับโดยการใช้กุญแจส่วนตัวของตน ดังนั้นจะไม่มีใครถอดรหัสได้นอกจาก Bob ขณะเดียวกันใครๆ ก็สามารถส่งข้อความที่ถูกเข้ารหัสลับด้วยกุญแจสาธารณะของ Bob ได้โดยไม่ต้องมีกุญแจส่วนตัวของ Bob จะเห็นว่าในระบบกุญแจสาธารณะนั้นผู้ส่ง

จากรูปที่ 4.2 สมมติว่า Alice ต้องการส่งข้อความถึง Bob ก่อนอื่น Alice ต้องมีกุญแจสาธารณะของ Bob ซึ่งอาจจะขอจาก Bob โดยตรงหรือจากองค์กรที่ให้บริการ จากนั้น Alice ก็จะใช้กุญแจสาธารณะของ Bob ในการเข้ารหัสลับแล้วส่งไปให้ Bob จากนั้น Bob จะถอดรหัสลับโดยการใช้กุญแจส่วนตัวของตน ดังนั้นจะไม่มีใครถอดรหัสได้นอกจาก Bob ขณะเดียวกันใครๆ ก็สามารถส่งข้อความที่ถูกเข้ารหัสลับด้วยกุญแจสาธารณะของ Bob ได้โดยไม่ต้องมีกุญแจส่วนตัวของ Bob จะเห็นว่าในระบบกุญแจสาธารณะนั้นผู้ส่งไม่จำเป็นต้องรู้หรือมีกุญแจส่วนตัวของผู้รับ จึงไม่จำเป็นต้องมีระบบขนส่งกุญแจที่ไว้ใจได้ ทำให้ลดความยุ่งยากในระบบการจัดการกุญแจรหัสลงอย่างมาก

อย่างไรก็ดีสิ่งสำคัญที่สุดที่ควรคำนึงถึงสำหรับรหัสลับระบบกุญแจสาธารณะคือ ความปลอดภัยของมันขึ้นอยู่กับว่าการที่เราจะสามารถคำนวณกุญแจส่วนตัวจากกุญแจสาธารณะได้ยากเพียงไร รหัสลับระบบกุญแจสาธารณะที่ดีนั้นการคำนวณกุญแจส่วนตัวจากกุญแจสาธารณะจะต้องทำได้ยากมากหรือไม่อาจทำได้เลย นอกจากรหัสลับของ Diffie และ Hellman แล้วรหัสลับระบบกุญแจสาธารณะหลายแบบได้ถูกพัฒนาขึ้น แต่ที่ใช้กันแพร่หลายมากที่สุดคือ รหัส RSA เพราะนอกจากการเข้ารหัสเอกสารแล้วมันสามารถนำไปใช้ในการลงนามระบบดิจิตอลได้ด้วย RSA แตกต่างจากรหัสลับของ Diffie และ Hellman ตรงที่ กุญแจสาธารณะของรหัสที่ Diffie และ Hellman คิดขึ้นนั้นจะเปลี่ยนแปลงทุกครั้งที่มีการติดต่อสื่อสาร ดังนั้นก่อนส่ง Alice จำเป็นต้องติดต่อกับ Bob ทุกครั้งเพื่อขอกุญแจสาธารณะใหม่เสมอ ในขณะที่กุญแจสาธารณะของ RSA จะไม่มีการเปลี่ยนแปลงในการติดต่อสื่อสารแต่ละครั้ง ทำให้ลดขั้นตอนและความสิ้นเปลืองในการสื่อสารลงได้มาก แต่เนื่องจากกุญแจสาธารณะไม่มีการเปลี่ยนแปลง รหัส RSA จึงจำเป็นต้องอาศัยองค์กรที่ไว้ใจได้เป็นผู้รับรองและเผยแพร่กุญแจสาธารณะเหล่านั้น รหัส 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(Modulus) 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 นั้น (n,e) คือกุญแจสาธารณะ ส่วน d คือกุญแจส่วนตัว เมื่อได้ค่าเหล่านี้แล้ว ค่า p และ q จะต้องเก็บเป็นความลับหรือถูกทำลายในทันที

และเพราะมีแต่ Bob เท่านั้นที่รู้ค่า d จึงมีแต่ Bob เท่านั้นจึงจะถอดรหัสได้ คุณสมบัติที่สำคัญประการหนึ่งของ RSA คือจากสมการ (4.5) ในทางกลับกันถ้าเราเข้ารหัสด้วยกุญแจส่วนตัว เราก็จะสามารถถอดรหัสด้วยกุญแจสาธารณะได้ด้วยเช่นกัน คุณสมบัติข้อนี้เองที่ทำให้ RSA มีประโยชน์มาก เพราะสามารถนำไปลงลายเซ็นดิจิตอลได้

วิธีการทำลายรหัส RSA ที่รู้จักกันดีคือการหาค่า d นั่นเองจากสมการ (4.4) เราอาจหาค่า d ได้ หากรู้ค่า p และ q แต่เนื่องจาก p และ q เป็น prime number ที่ $pq = n$ ดังนั้นการทำลายรหัส RSA ขึ้นอยู่กับการแยกตัวประกอบของ n นั่นเอง แต่วิธีการแยกตัวประกอบของ n นั้นไม่ง่ายเลยถ้าหาก 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 ก็จำเป็นต้องใช้การยกกำลังในระบบโมดูโล ซึ่งการยกกำลังนั้นสามารถทำได้โดยใช้วงจรคูณระบบโมดูโลมาต่ออนุกรมกัน ดังนั้นความเร็วของทั้งการเข้าและถอดรหัสจึงขึ้นอยู่กับความเร็วของวงจรคูณโมดูโลเป็นอย่างมาก นี่เองเป็นจุดอ่อนข้อหนึ่งของ RSA เมื่อเทียบกับรหัสลับระบบกุญแจปกปิด เช่น DES เพราะปัจจุบันการคูณในระบบโมดูโลยังค่อนข้างยุ่งยากเมื่อเทียบกับการแทนค่าหรือสลับตำแหน่งใน DES ได้มีการประมาณกันว่าสำหรับ n ที่มีความยาว 512 บิท DES จะเร็วกว่า RSA ประมาณ 100 เท่าถ้าทำการเข้ารหัสด้วยซอร์ฟแวร์ และอาจเร็วกว่าถึง 1000 หรือ 10000 เท่าซึ่งแล้วแต่ลักษณะการออกแบบของวงจรหากทำการเข้ารหัสด้วยฮาร์ดแวร์ โดยทั่วไปแล้วเราต้องการให้วงจรเข้ารหัสเร็วกว่าการถอดรหัสดังนั้นเราจึงมักเลือกให้ e มีค่าน้อยกว่า d และยิ่งกว่านั้นเรามักให้ e ของสมาชิกทุกคนมีค่าเดียวกันเพื่อให้ฮาร์ดแวร์ของวงจรเข้ารหัสสำหรับสมาชิกแต่ละคนมีลักษณะคล้ายกัน

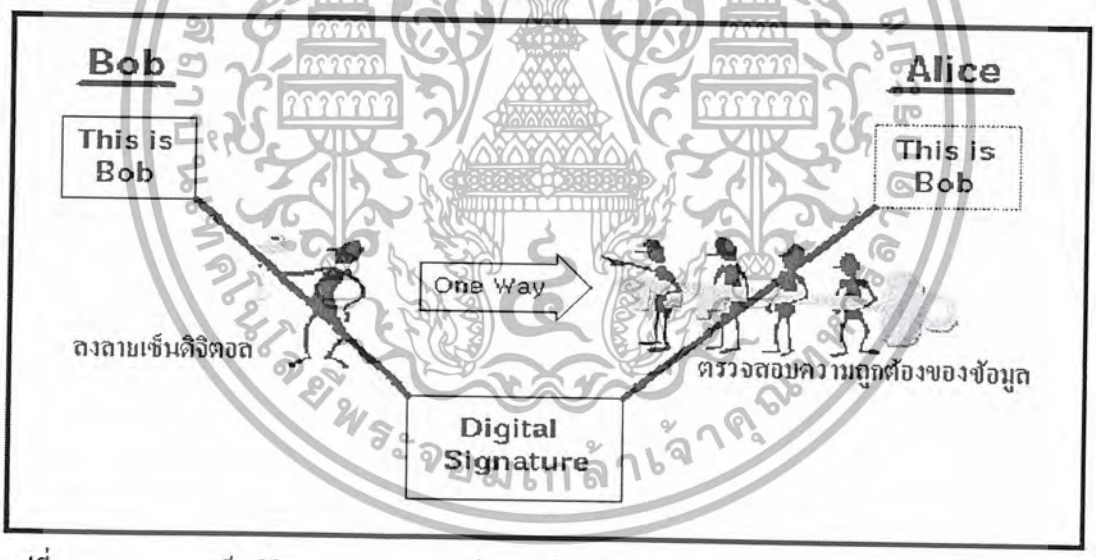
เนื่องจาก DES และ RSA มีข้อดีข้อเสียที่แตกต่างกันจึงไม่จำเป็นว่ารหัสชนิดใดจะเหมาะสมในทุกสถานการณ์ โดยทั่วไปแล้ว DES จะถูกใช้ในการเข้ารหัสข้อมูลขนาดใหญ่เพราะรวดเร็วกว่าในขณะที่ RSA จะถูกใช้ในระบบสื่อสารที่ไม่ยาวนักแต่ต้องการความปลอดภัยสูงในบางครั้ง RSA ยังถูกใช้ร่วมกับ DES เพื่อเสริมจุดเด่นซึ่งกันและกัน เช่นตัวเอกสารจริงจะถูกเข้ารหัสด้วย DES โดยที่กุญแจรหัส DES จะถูกเข้าด้วย RSA แล้วส่งไปด้วยกันหรือส่งไปก่อนแต่ในบางครั้ง DES อย่างเดียวก็พอแล้วหากการแลกเปลี่ยนกุญแจสามารถทำได้อย่างปลอดภัยพอ หรือในกรณีที่ผู้ส่งและผู้รับเป็นบุคคลคนเดียวกัน เช่น ฮาร์ดดิสก์ในคอมพิวเตอร์ส่วนตัว หรือข้อมูลส่วนตัวในสมาร์ทการ์ด

รหัส RSA เป็นรหัสลับระบบกุญแจสาธารณะที่รู้จักกันแพร่หลายมากที่สุดระบบหนึ่งในปัจจุบัน รหัส RSA ทั้งที่เป็นซอร์ฟแวร์และฮาร์ดแวร์ ถูกนำไปใช้ในองค์กรต่างๆ มากมายทั่วโลก รหัสลับในมาตรฐาน X.509 ของ CCITT และ ISO 9676 นั้นล้วนเป็นรหัสลับตระกูลเดียวกันกับรหัส RSA นอกจากนี้รหัส RSA ยังถูกกำหนดให้เป็นส่วนหนึ่งของมาตรฐานของ SWIFT(Society for Worldwide Interbank Financial Telecommunications)

4.4 การพิสูจน์ความเป็นเจ้าของ(Authentication) และ ลายเซ็นดิจิตอล(Digital Signature)

ในการสื่อสารระบบดิจิตอล การพิสูจน์ความเป็นเจ้าของคือขบวนการที่ทำให้ผู้รับมั่นใจได้ว่าข้อมูลที่ได้รับนั้นมาจากใครและผู้นั้นเป็นผู้ส่งจริงหรือเปล่า ตัวอย่างง่ายๆ ที่เห็นชัดได้แก่การป้อนรหัสลับพร้อมกับบัตร ATM ของธนาคาร หากเราคิดว่าเจ้าของเท่านั้นที่รู้รหัสลับประจำตัว การตรวจสอบว่าผู้ใช้ใส่รหัสถูกหรือไม่นั้นก็ถือเป็นการพิสูจน์ว่าผู้ใช้เป็นเจ้าของหรือไม่ ได้อย่างหนึ่ง การพิสูจน์ความเป็นเจ้าของทางอิเล็กทรอนิกส์สามารถทำได้โดยการใช้ขบวนการเข้ารหัสลับ ซึ่งอาจเป็นได้ทั้งรหัสลับระบบกุญแจปกปิด เช่น DES หรือรหัสลับระบบกุญแจสาธารณะ เช่น RSA แต่สำหรับการพิสูจน์ความเป็นเจ้าของทางอิเล็กทรอนิกส์ที่ใช้รหัสลับระบบกุญแจสาธารณะนั้นยังหมายถึงการลงลายเซ็นดิจิตอลด้วย

ในการพิสูจน์ความเป็นเจ้าของทางดิจิตอลนั้น เราอาจเปรียบเทียบได้กับการรับรองเอกสารทั่วไปก็ได้ โดยในการรับรองจะต้องมีการลงลายเซ็นจากผู้ส่งในเอกสาร ซึ่งสำหรับเอกสารทางดิจิตอลนั้นส่วนที่จะเป็นการลงลายเซ็นนั้นจะต้องเป็นข้อมูลเพิ่มเติมพิเศษที่จะต้องได้รับการยอมรับกันทั่วไปและยากต่อการปลอมแปลง ในขณะเดียวกันผู้รับหรือบุคคลที่สามก็สามารถพิสูจน์ได้ว่าเป็นของจริง ดังนั้นการรับรองที่ปลอดภัยนั้นจะต้องประกอบด้วย 2 ขบวนการใหญ่ๆ คือ การลงลายเซ็นที่ไม่สามารถปลอมแปลงได้ง่ายๆ และการพิสูจน์ความเป็นเจ้าของว่าเป็นของจริงหรือไม่ โดยที่ผู้ลงนามไม่สามารถปฏิเสธในภายหลังได้ว่าตัวเองไม่ได้ลงนาม(Non-Repudiation) การใช้รหัสลับในการส่งข้อความที่จำเป็นจะต้องมีการพิสูจน์ความเป็นเจ้าโดยอาจทำได้ดังวิธีต่อไปนี้



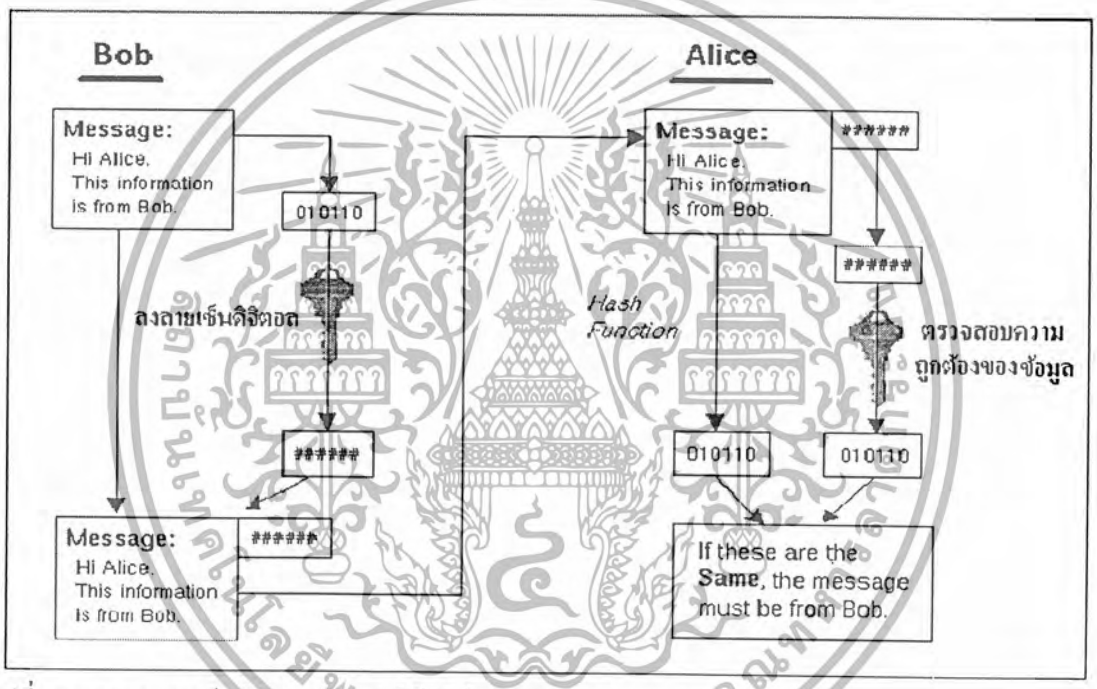
รูปที่ 4.3 การลงลายเซ็นดิจิตอลและการพิสูจน์ความเป็นเจ้าของ

ถ้า Bob ต้องการส่งเอกสารให้ Alice พร้อมกับรับรองว่าตนเองเป็นผู้เขียน นอกจากตัวเอกสารจริงแล้ว Bob จะเข้ารหัสลับเอกสารนั้นก่อนแล้วจึงส่งไปพร้อมกับเอกสารตัวจริง เมื่อ Alice ได้รับเขาจะพิสูจน์ว่าเป็นเอกสารของ Bob จริงได้โดยการถอดรหัสเอกสารรหัสลับแล้วนำมาเปรียบเทียบกับเอกสารตัวจริงที่ส่งมาด้วยว่าตรงกันหรือไม่ ถ้าตรงกันก็ถือว่าถูกต้อง

จะสังเกตได้ว่าถ้าเราใช้รหัสลับระบบกุญแจปกปิด ทั้ง Bob และ Alice จะต้องมีกุญแจที่เหมือนกันซึ่งทำให้มีปัญหาในการนำส่งกุญแจอย่างปลอดภัย หรือจำเป็นต้องให้ Bob มาพิสูจน์ด้วย

ตัวเอง แต่ถ้าเราใช้รหัสลับระบบกุญแจสาธารณะ โดยเฉพาะอย่างยิ่ง RSA ในตอนส่ง Bob ก็จะเข้ารหัสโดยใช้กุญแจส่วนตัวของตนเองในการเข้ารหัส และ Alice ก็จะใช้กุญแจสาธารณะของ Bob ในการพิสูจน์ ทำให้ลดความยุ่งยากในการนำส่งกุญแจไปได้ นอกจากนี้ใครๆ ก็สามารถพิสูจน์ลายเซ็นได้ว่าเป็นของจริงหรือไม่โดยไม่ต้องมีกุญแจส่วนตัวของ Bob โดยทั่วไปแล้วสำหรับกรณีที่ใช้รหัส RSA ในการลงลายเซ็นดิจิตอลนั้น ก็เช่นเดียวกับการเข้ารหัสลับ คือเรามักให้ e มีขนาดน้อยกว่า d และ e มีค่าเท่ากันสำหรับสมาชิกทุกคนนั่นหมายความว่าขบวนการลงนามนั้นใช้เวลามากกว่าขบวนการพิสูจน์

อย่างไรก็ดีเนื่องจากวงจรเข้ารหัสและถอดรหัสลับระบบกุญแจสาธารณะยุ่งยากมากกว่ารหัสลับระบบกุญแจปกปิดมาก ในอุปกรณ์ที่มีขนาดเล็กราคาถูก เช่น สมาร์ทการ์ด ยังคงนิยมใช้รหัสลับระบบกุญแจปกปิด เช่น DES อยู่ แต่ในอนาคตเมื่ออุปกรณ์มีราคาต่ำลงอุปกรณ์เหล่านี้มีแนวโน้มที่จะเปลี่ยนมาใช้รหัสลับระบบกุญแจสาธารณะ



รูปที่ 4.4 การลงลายเซ็นดิจิตอลด้วยการใช้ฟังก์ชันแฮส

ข้อสังเกตอีกประการในการลงลายเซ็นดิจิตอลแบบที่อธิบายมาแล้วนั้น เราต้องส่งทั้งเอกสารปกติตัวจริงและเอกสารรหัสลับไปด้วยกัน เพื่อเป็นการยืนยันว่าเป็นการลงลายเซ็นดิจิตอลที่ออกให้กับเอกสารชิ้นที่ส่งไปจริง ซึ่งนอกจากจะเป็นการสูญเสียแบนด์วิดท์ในการสื่อสารแล้วยังเสียเวลามากในการถอดรหัสโดยเฉพาะอย่างยิ่งเมื่อเอกสารที่ต้องการพิสูจน์ความเป็นเจ้าของนั้นมีขนาดใหญ่มาก ปัจจุบันเราแก้ปัญหานี้ โดยการใช้"ฟังก์ชันแฮสทิศทางเดียว(One-Way Hash Function)"

ฟังก์ชันแฮสทิศทางเดียวคือฟังก์ชั่นที่ทำการเปลี่ยนเอกสารที่มีความยาวขนาดต่าง ให้เป็นข้อมูลขนาดสั้นๆ ค่าหนึ่งที่มีความยาวคงที่เรียกว่า "ไดเจสของเอกสาร(Massage Digest)" โดยที่ทิศทางเดียว

ของฟังก์ชันแฮส หมายความว่าเราไม่สามารถนำไดเจสเหล่านั้นมาตีความใดๆ ที่เกี่ยวข้องกับเอกสารเดิม หรือเพื่อแปลงกลับไปเป็นเอกสารเดิมได้เลย

ดังนั้นจากคุณสมบัติของฟังก์ชันแฮสทิศทางเดียว ในการลงลายเซ็นดิจิตอลนั้นแทนที่เราจะเข้ารหัสลับกับเอกสารโดยตรง เราสามารถเข้ารหัสไดเจสของเอกสารแทนแล้วส่งไปพร้อมกับเอกสารปกติได้ และในการพิสูจน์ลายเซ็นดิจิตอลนั้นเราทำได้โดยการนำเอาไดเจสที่ได้จากการเอาเอกสารที่ส่งมาด้วยกันมาเข้าฟังก์ชันแฮสทิศทางเดียวที่เหมือนกับที่ใช้ในตอนส่ง แล้วเปรียบเทียบกับไดเจสที่ได้จากการถอดรหัสว่าตรงกันหรือไม่ การพิสูจน์ความเป็นเจ้าของและการลงลายเซ็นระบบดิจิตอลมีความสำคัญมากขึ้นทุกทีเพราะการที่มันสามารถพิสูจน์เอกสารในระบบดิจิตอลได้ นอกจากจะถูกนำไปประยุกต์ใช้เพื่อเพิ่มความปลอดภัยในระบบรักษาความปลอดภัยของข้อมูลแล้ว ยังมีประโยชน์มหาศาลต่อระบบการเงินการธนาคารผ่านระบบโครงข่ายทางดิจิตอลที่เชื่อถือได้เป็นจริงขึ้น โดยใช้การลงลายเซ็นดิจิตอลของผู้มีอำนาจแทนการลงนามจริงในเอกสารต่างๆ เช่น การทำสัญญา การสั่งจ่ายเงินหรือธนาคารอาจใช้รับรองข้อมูลที่ใช้แทนเงินในระบบดิจิตอล เป็นต้น

4.5 การจัดการกุญแจรหัส(Key Management) และ การแลกเปลี่ยนกุญแจรหัส(Key Exchange) ในระบบดิจิตอล

ดังที่กล่าวมาแล้วระบบจัดการกุญแจรหัสที่ปลอดภัยเป็นสิ่งสำคัญมาก เพราะในทางปฏิบัติการจัดการกุญแจรหัสมักจะตกเป็นเป้าหมายในการจู่โจมเพื่อทำลายความปลอดภัยของระบบโครงข่ายข้อมูลมากกว่าการจู่โจมทำลายรหัสลับโดยตรงเสียอีก โดยทั่วไปแล้วระบบโครงข่ายข้อมูลทางดิจิตอลในองค์กรหรือหน่วยงานจะประกอบไปด้วยคอมพิวเตอร์ส่วนตัวและอุปกรณ์ปลายทางอื่นๆ ที่ต่อเข้ากับระบบสื่อสารขององค์กรที่จัดให้มีการใช้รหัสลับเพื่อรักษาความปลอดภัยของข้อมูลในองค์กร และส่วนของสมาชิกนั้นจะต้องมีระบบจัดการกุญแจซึ่งมีคุณสมบัติดังต่อไปนี้

1. กุญแจรหัสจะต้องถูกแจกจ่ายหรือแลกเปลี่ยนแก่สมาชิกใดๆ เพื่อให้ผู้รับและผู้ส่งสามารถทำการเข้ารหัสและถอดรหัสลับ หรือทำการรับรองและพิสูจน์ในขบวนการพิสูจน์ความเป็นเจ้าของในระหว่างการติดต่อสื่อสารได้ได้
2. จะต้องมีวิธีที่ปลอดภัยที่จะป้องกันไม่ให้ผู้บุกรุกดักฟัง หรือลอบนำกุญแจไปคัดลอกระหว่างการสื่อสารจนกระทั่งสามารถปลอมเป็นสมาชิก หรือแอบอ้างเป็นสมาชิกผู้หนึ่งผู้ใดได้
3. จะต้องมีวิธีปลอดภัยที่ทำให้สมาชิกสามารถตรวจสอบได้ว่ากุญแจรหัสที่ได้รับนั้นเป็นของสมาชิกที่ต้องการติดต่อสื่อสารด้วย

ระบบจัดการกุญแจที่ง่ายที่สุด(แต่ทำปลอดภัยได้ยากมากที่สุด) คือการใช้รหัสลับกุญแจปกปิดร่วมกับช่องเก็บสัญญาณซึ่งสามารถแลกเปลี่ยนกุญแจรหัสที่มีความเร้นลับและปลอดภัยพอที่จะป้องกันอาชญากรรมในข้อ 2 และ 3 อย่างไรก็ดีการจัดหาช่องสัญญาณที่ปลอดภัยอย่างแท้จริงนั้นแทบเป็นสิ่งที่เป็นไปไม่ได้ ระบบที่ยังคงใช้การจัดการกุญแจแบบนี้ในปัจจุบันนั้น มีความสามารถป้องกันความปลอดภัยได้เพียงระดับหนึ่งเท่านั้น ระบบ ATM ของธนาคารที่ใช้กันทั่วไปทุกวันนี้ก็เป็นตัวอย่างที่ดีเพราะมีผู้เชี่ยวชาญรวมทั้งคดีตัวอย่างที่เกิดขึ้นในต่างประเทศจำนวนมาก ซึ่งชี้ให้เห็นว่า มันเปราะบาง

มากต่อการถูกโจมตีด้วยการแอบดักฟังทางสายโทรศัพท์ที่ถูกใช้ในการสื่อสารระหว่างเครื่อง ATM และศูนย์คอมพิวเตอร์ของธนาคาร

การใช้รหัสลับระบบกุญแจสาธารณะสามารถแก้ปัญหาการบุกรุกในข้อ 2 ไปได้เพราะกุญแจที่ใช้ในการแลกเปลี่ยนเป็นกุญแจสาธารณะไม่ใช่กุญแจส่วนตัว จึงไม่สามารถใช้ในการถอดรหัสหรือการลงนามระบบดิจิตอลได้ อย่างไรก็ตามรหัสลับระบบกุญแจสาธารณะเพียงอย่างเดียว โดยเฉพาะ RSA ไม่อาจป้องกันการปลอมแปลงในข้อ 3 ได้

ตัวอย่างเช่น ในระบบที่ใช้รหัส RSA ถ้าผู้บุกรุกมีความชำนาญสามารถเข้ามาในระบบได้ แล้วลักลอบเปลี่ยนกุญแจของ Bob ด้วยกุญแจที่ทำขึ้นเอง หากไม่ระบบการป้องกันที่รัดกุมเพียงพอ ที่จะตรวจสอบว่ากุญแจสาธารณะที่อยู่ในฐานข้อมูลนั้นเป็นของจริงหรือไม่ ผู้บุกรุกก็สามารถแอบอ้างเป็น Bob ได้อย่างง่ายดาย ด้วยกุญแจที่ทำปลอมผู้บุกรุกสามารถที่จะอ่านเอกสารที่เป็นความลับของสมาชิกคนอื่นที่ต้องการส่งให้ Bob ได้ หรืออาจแอบอ้างการเป็น Bob โดยการใช้ลายเซ็นปลอมในการส่งเอกสารต่างๆ เช่น ใบสั่งของ หรือ โอนจ่ายเงินจากบัญชีของ Bob ได้

ปัจจุบันมาตรการที่นิยมใช้กันมากที่สุดเพื่อป้องกันการบุกรุกชนิดนี้คือจะต้องมีหน่วยงานเชื่อถือได้ทำการออกใบรับรองให้หน่วยงานที่ว่านี้ก็คือ หน่วยงานหรือองค์กรที่ได้รับสิทธิการรับรอง(Certificate Authority : CA) ส่วนใบรับรองที่องค์กรนี้จะออกให้ก็เรียกว่า ใบรับรองดิจิตอล(Digital Certificate) การรับรองของ CA ไม่ใช่ให้เลยทันที แต่ผู้ใช้จะต้องทำการสมัครขอใช้บริการ ซึ่งทาง CA ก็จะทำการตรวจสอบข้อมูลในใบสมัคร (เช่น ชื่อ, ที่อยู่, อีเมล์แอดเดรส เป็นต้น) ว่าเป็นข้อมูลจริงหรือไม่และหาข้อมูลเพื่อมายืนยันตัวบุคคลที่ใช้บริการว่ามีตัวตนอยู่หรือไม่ หากผ่านเกณฑ์การตรวจสอบ CA ก็จะออกใบรับรองดิจิตอลให้ มาตรฐานของใบรับรองที่ยอมรับกันมากที่สุดอันหนึ่งคือ X.509 ของ CCITT

บทที่ 5

Elliptic Curve Cryptosystem

5.1 ทำไม ECC จึงเหมาะสำหรับสมาร์ทการ์ด

จากบทที่ผ่านมาเราจะเห็นได้ว่าสมาร์ทการ์ดนั้นได้ถูกนำมาใช้งานอย่างแพร่หลาย ด้วยข้อดีที่มีขนาดเล็ก ราคาถูก นำมาประยุกต์ใช้งานในด้านต่างๆ มากมาย โดยที่ระบบรักษาความปลอดภัยของข้อมูลนั้นถือเป็นหัวใจสำคัญของสมาร์ทการ์ด และเนื่องด้วยขนาดที่จำกัดของสมาร์ทการ์ดทำให้การจัดการพื้นที่ในสมาร์ทการ์ดให้มีประสิทธิภาพที่สุด จึงเป็นสิ่งสำคัญที่ต้องพิจารณา ในระยะแรกนั้นการเข้ารหัสลับในสมาร์ทการ์ดนั้นใช้รหัสลับแบบกุญแจปกปิดแต่เพียงอย่างเดียว ซึ่งทำงานได้รวดเร็ว แต่ว่าความหลากหลายในการใช้งานมีน้อย จนเมื่อมีรหัสลับระบบกุญแจสาธารณะเกิดขึ้น เช่น RSA ก็ได้มีแนวความคิดที่จะนำรหัสลับระบบกุญแจสาธารณะมาใช้งานบนสมาร์ทการ์ด แต่จากบทที่แล้ว จากอัลกอลิทึมของ RSA นั้นแสดงให้เห็นว่าการออกแบบวงจรเข้ารหัสนั้น ต้องใช้การยกกำลังระบบโมดูโล ทำให้วงจรมีขนาดใหญ่มากจนไม่เหมาะที่จะนำมาสร้างบนแอพพลิเคชั่นขนาดเล็กและราคาถูก อย่างเช่นสมาร์ทการ์ด จนเมื่อปี คศ.1985 Victor Miller และ Neal ได้นำเอาการเข้ารหัสลับแบบ Elliptic Curve มาประยุกต์ใช้งานร่วมกับอัลกอลิทึมของรหัสลับระบบกุญแจสาธารณะแบบอื่นๆ เช่น ร่วมกับรหัสลับแบบ ElGamal เรียกว่า EC(Elliptic Curve) ElGamal โดยนำมาประยุกต์ใช้ในส่วนของกุญแจรหัส เนื่องจาก ECC นั้นมีขนาดกุญแจที่เล็กกว่ารหัสลับระบบกุญแจสาธารณะแบบอื่น

ตารางที่ 5.1 เปรียบเทียบขนาดของกุญแจรหัสระหว่าง RSA กับ ECC

MIPS years	RSA/DSA key size	ECC key size	RSA/ECC 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

จากตาารางที่ 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} MIPS years ซึ่งจากรูปทำให้ RSA และ DSA ควรเริ่มใช้งานที่ 1,024 บิต และ ECC ควรเริ่มใช้งานที่ 160 บิต ซึ่งเห็นได้อย่างชัดเจนว่าขนาดของกุญแจรหัสของ ECC นั้นเล็กกว่าของ RSA และ DSA มาก และเมื่อเพิ่มระดับความปลอดภัยให้สูงขึ้นจะมำให้ขนาด

ความแตกต่างของกุญแจรหัสจะเพิ่มมากขึ้น เช่น เมื่อใช้งาน ECC ที่ 600 บิต ขณะที่ RSA และ DSA นั้นต้องใช้ถึง 21,000 บิต ทำให้เห็นได้ว่าการเลือกใช้ ECC บนสมาร์ทการ์ดนั้นทำให้ลดการสิ้นเปลืองหน่วยความจำในการเก็บกุญแจรหัสลงไปได้มาก และอัลกอริทึมของ ECC นั้นใช้คณิตศาสตร์ได้หลายแบบในการสร้าง ซึ่งสามารถสร้างได้หลายวิธีแล้วแต่คณิตศาสตร์ที่เลือกใช้ ในปัจจุบันมีงานวิจัยของ ECC ออกมาอย่างต่อเนื่องทำให้ปัจจุบันสามารถสร้าง ECC ให้วงจรให้มีขนาดเล็ก สามารถทำงานได้อย่างรวดเร็วทั้งในแบบซอร์ฟแวร์ และ ฮาร์ดแวร์

ตารางที่ 5.2 แสดงกำลังที่ใช้สำหรับทำลายระบบ ECC โดยใช้วิธีการของ Pollard rho method ซึ่งเปลี่ยนแปลงตามค่าของ n

Field size (in bits)	Size of n (in bits)	$(\pi n/2)^{1/2}$	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 คณิตศาสตร์สนามจำกัด F_2m ในรูป Polynomial Basis

ให้ $f(x) = x^m + f_{m-1}x^{m-1} + \ldots\ldots\ldots + f_2x^2 + f_1x + f_0$ โดยที่ $f_i \in \{0,1\}$ สำหรับ $i = (0,1,2,\ldots\ldots,m\text{-}1)$ และ f(x) เป็น โพลิโนเมียลลดรูป(Reduction Polynomial) โดยที่โพลิโนเมียลลดรูป ดังกล่าวก็คือพหุนามที่ไม่สามารถแยกตัวประกอบได้กำลังที่ m หรือเรียกว่า โพลิโนเมียลที่ลดรูปไม่ได้(Irreducible Polynomial) ถ้ากำหนดให้ g เป็นรากของโพลิโนเมียลกล่าวคือ $P(g) = 0$ ค่ายกกำลังของ g มีค่าได้ถึง $2^m - 2$ ค่าที่แตกต่างกัน โดยที่ค่าของ g^{2m-1} จะเท่ากับ 1 ใหม่ ดังนั้น $0,g,g^2,\ldots,g^{m-1}$ จึงเป็นเซ็ตของ F_2m และเรียก F_2m ว่า Field โดยเรียกสมาชิกของ Field ว่า Field Element โดยที่แต่อิลิเมนท์สามารถขยีนได้ด้วยผลรวมของอิลิเมนท์ที่เป็น $1,g,g^2,\ldots,g^{m-1}$ การทำโอเปอร์เรชั่นของ F_2m ในรูป Polynomial Basis แสดงได้ดังนี้

5.2.1 การบวก(Addition)

ถ้า $a = (a_{m-1}\ldots a_1\ a_0)$ และ $b = (b_{m-1}\ldots b_1\ b_0)$ ซึ่งเป็นอิลิเมนท์ของ F_2m แล้ว $a + b = c = (c_{m-1}\ldots c_1\ c_0)$ โดยที่ $c_i = (a_i + b_i) \bmod 2$ ดังนั้นการบวกของอิลิเมนท์จึงอยู่ในรูปของการ exclusive or

5.2.2 การลบ(Subtraction)

การลบของ F_2m นั้นจะได้ $(a_{m-1}\ldots a_1\ a_0) + (a_{m-1}\ldots a_1\ a_0) = (00\ldots000)$ ดังนั้นการลบ และการบวกใน F_2m จึงมีการทำโอเปอร์เรชั่นที่เหมือนกัน

5.2.3 การคูณ(Mutiplication)

ถ้า a = $(a_{m-1}\ldots a_{m-1}\ a_{m-1})$ และ b = $(b_{m-1}\ldots\ldots b_1\ b_0)$ ซึ่งเป็นอิลิเมนท์ของ F_2m แล้ว a.b = c = r = $(r_{m-1}\ldots r_1\ r_0)$ โดยที่ r นั้นเป็นเศษของการคูณกันของโพลิโนเมียล$(a_{m-1}x^{m-1}+\ldots+a_2x^2+a_1x+a_0).(b_{m-1}x^{m-1}+\ldots+b_2x^2+b_1x+b_0)$ซึ่งหารด้วย f(x) over F_2

5.2.4 การหาส่วนกลับ(Inversion)

ถ้า a เป็นอิลิเมนท์ที่ไม่เท่ากับ 0 ใน F_2m แล้ว ส่วนกลับของ a จะสามารถแทนได้ในรูปของ g ยกกำลัง ซึ่งแทนด้วย $a = g^i$ จะได้ส่วนกลับของ a คือ $a^{-1} = g\hat{}\{(-i) \bmod (2^m-1)\}$

ตัวอย่างที่ 5.1 $F_2$4 อยู่ในรูปของ Polynomial Basis ให้ $f(x) = x^4 + x + 1$ โพลิโนเมียลลดรูปเราจะได้ สมาชิกจำนวน 16 อิลิเมนท์ของ $F_2$4 ดังนี้

(0000) = 0

(0001) = 1

(0010) = x

(0011) = x + 1

(0100) = x^2

(0101) = x^2 + 1

(0110) = x^2 + x

(0111) = x^2 + x + 1

(1000) = x^3

(1001) = x^3 + 1

(1010) = x^3 + x

(1011) = x^3 + x + 1

(1100) = x^3 + x^2

(1101) = x^3 + x^2 + 1

(1110) = x^3 + x^2 + x

(1111) = x^3 + x^2 + x + 1

จะเห็นว่าสมาชิกของ $F_2$4 อยู่ในรูปของ $a_3x^3 + a_2x^2 + a_1x + a_0$ โดยที่ $a_i \in \{0,1\}$

ตัวอย่างที่ 5.2 การหาผลคูณของ (1101).(1001) ของ $F_2 4$ ในรูป Polynomial Basis

$$(1101).(1001) = (x^3 + x^2 + 1).(x^3 + 1) \bmod f(x)$$
$$= x^6 + x^5 + 2x^3 + x2 + 1$$
$$= x^6 + x^5 + x^2 + 1 \bmod f(x) \text{ (สัมประสิทธิ์ลดรูปด้วยโมดูโล 2)}$$
$$= x^3 + x^2 + x + 1$$
$$= (1111)$$

โดยที่อิลิเมนท์ g = x = (0010) เป็นตัวสร้างสมาชิกอื่นของ $F^*_2 4$ ดังนั้นจำนวนอิลิเมนท์ทั้งหมดจะมีจำนวนเท่ากับ 15 อิลิเมนท์ ซึ่งแสดงได้ดังนี้

$(0010) = g^1$ $(0100) = g^2$ $(1000) = g^3$ $(0011) = g^4$ $(0110) = g^5$

$(1100) = g^6$ $(1011) = g^7$ $(0101) = g^8$ $(1010) = g^9$ $(0111) = g^{10}$

$(1110) = g^{11}$ $(1111) = g^{12}$ $(1101) = g^{13}$ $(1001) = g^{14}$ $(0001) = g^{15}$

ตัวอย่างที่ 5.3 การหาส่วนกลับของ a = (1011)

การหาส่วนกลับของ a = (1011) สามารถหาได้โดยเปลี่ยน a ให้อยู่ในรูปของ g^i ซึ่งจะได้ a $= g^7 = (1011)$ ดังนั้น $g^{(-7) \bmod 15} = g^8 = (0101)$ ซึ่งสามารถตรวจสอบความถูกต้องได้ดังนี้

$g^7 . g^{(-7)} \bmod 15 = (1011)(0101)$

$= (x3 + x + 1)(x2 + 1) \bmod f(x)$

$= x5 + x2 + x + 1 \bmod f(x)$

$= (x4 + x + 1)(x) + (1) \bmod f(x)$

$= (0001)$

5.3 Elliptic Curve Over $F_2 m$

5.3.1 สมการ Elliptic Curve

elliptic curve E over $F_2 m$ สามารถแสดงความสัมพันธ์ในรูปของสมการ Non-Supersingular ดังต่อไปนี้

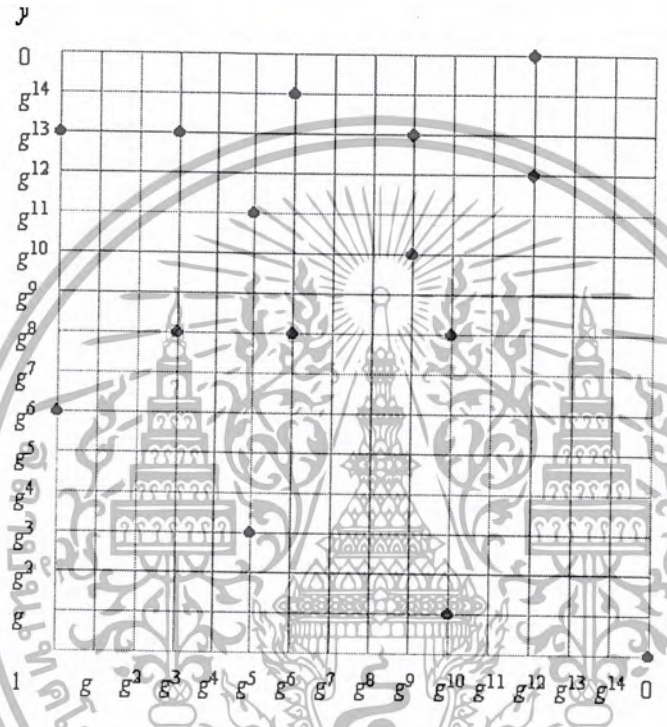
$$y^2 + xy = x^3 + ax^2 + b \bmod f(x) \qquad (5.1)$$

โดยที่ $a, b \in F_2 m$ และ $b \neq 0$ เซ็ตของ $E(F_2 m)$ จะประกอบไปด้วยจุด (x,y) โดยที่ $x \in F_2 m$, $y \in F_2 m$ ที่ทำให้สมการ(1) เป็นจริง และอีกจุดหนึ่งคือจุด O เรียกว่า Point at infinity

ตัวอย่างที่ 5.4 พิจารณา F_24 ซึ่งมี $f(x) = x^4 + x + 1$ จากสมการที่(1) ให้ $a = g^4$ และ $b = 1$ เราจะได้จุดทั้งหมดของ $E(F_24)$ และจุด O 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 ตำแหน่งของอิลิเมนท์ที่เกิดจากสมการ $y^2 + xy = x^3 + g^4x^2 + 1 \mod (x^4 + x + 1)$

5.3.2 กฎการบวกของ Elliptic curve

1. $P + O = O + P = P$ 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. ให้ $P = (x_1,y_1) \in E(F_2m)$ และ $Q = (x_2,y_2) \in E(F_2m)$ 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

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

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

5.4 การเลือกใช้โพลิโนเมียลลดรูป

ถ้าโพลิโนเมียลอยู่ในรูป $x^m + x^k + 1$ โดยที่ $1 \leq k \leq m-1$ แล้ว จะเรียกโพลิโนเมียลนี้ว่า Trinomial over F_2 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 F_2 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
   - พิจรณาเฉพาะ k3 ให้มีค่าน้อยที่สุดก่อน แล้วค่อยพิจารณา k2,k1
   - พิจรณาเฉพาะ k3,k2 ให้มีค่าน้อยที่สุดก่อน แล้วค่อยพิจารณา k2
   - พิจรณาทั้ง k3,k2,k1 ให้มีค่าน้อยที่สุด

การเลือกโพลิโนเมียลลดรูปของ F_2m ในรูปของ Polynomial Basis ตามมาตรฐานของ SEC แสดงในตารางที่ 5.3

ตารางที่ 5.3 แสดงโพลิโนเมียลลดรูปดีกรีต่างๆ ตามมาตรฐานของ SEC

Field	Reduction Polynomial(s)
F_2113	$x^{113} + x^9 + 1$
F_2131	$x^{131} + x^8 + x^3 + x^2 + 1$
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	$x^{571} + x^{10} + x^5 + x^2 + 1$

5.5 อัลกอริทึม Elliptic Curve ElGamal

อัลกอริทึมเบื้องต้นของ Elliptic Curve ที่รู้จักกันดีอัลกอริทึมหนึ่งคือ อัลกอริทึม Elliptic Curve EIGamal หรือเรียกสั้นๆ ว่า EC ElGamal โดยในโครงงานนี้จะใช้อัลกอริทึม EC EIGamal เป็นอัลกอริทึมที่แสดงการเข้ารหัสและถอดรหัส เนื่องจากหลักการที่ง่าย ทำให้สามารถมองเห็นถึงการเอาหลักการของ Elliptic Curve ซึ่งจะเป็นพื้นฐานในการศึกษาอัลกอริทึมมาตรฐานอื่นๆ ในหัวข้อ 5.6 ซึ่งจะเห็นว่า จะมีการนำเอา Elliptic Curve ไปใช้ในการสร้างกุญแจแบบต่างๆ

หลักการทำงานของอัลกอริทึม EC EIGamal แสดงดังตารางที่ 5.4 โดยเบื้องต้นนั้นทุกคนที่ใช้อัลกอริทึม EC EIGamal ทุกคนต้องทำการสร้างกุญแจส่วนตัว และ กุญแจสาธารณะ จากนั้นทำการนำเอากุญแจสาธารณะไปขอใบรับรองจากองค์กรที่ทำหน้าที่รับรอง ซึ่งจะใช้กุญแจสาธารณะนี้เสมือนสิ่งยืนยันตัวของผู้ใช้ ทำให้ผู้ที่จะส่งข้อความลับไปให้จะใช้กุญแจสาธารณะของคนนั้นเข้ารหัส ดังแสดงในอัลกอริทึม เมื่อ B ต้องการส่งข้อมูลลับมาให้ A อย่างแรกที่ B จะต้องทำคือขอกุญแจสาธารณะ คือ A_T และพารามิเตอร์ต่างๆ แล้วนำมาเข้ารหัสตามตารางการทำงานดังกล่าว จากนั้นเมื่อส่งข้อความลับ (C_1, C_2) ที่ B ได้ทำการเข้ารหัสไปให้ A จากนั้น A จะทำการถอดรหัสออกมาได้โดยใช้กุญแจส่วนตัวของ a โดยทำการคูณแบบ Scalar คือ $a.C_1$ คือ Point C_1 บวกกัน a ครั้ง

ตารางที่ 5.4 แสดงอัลกอริทึมการเข้ารหัสแบบ Elliptic Curve EIGamal

A ทำการสร้างกุญแจเข้ารหัส	
ขั้นตอนที่ 1	เลือก Field(m)
ขั้นตอนที่ 2	กำหนด E ซึ่งก็คือสมการ $y^2 + xy = x^3 + ax^2 + b \bmod f(x)$ โดยเลือกค่า $a, b \in 2^m$
ขั้นตอนที่ 3	สุ่มคู่ลำดับ P ที่ทำให้ E เป็นจริง
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 $kA_T = k(aP) = (x,y)$
ขั้นตอนที่ 4	คำนวณ $C_2 = P_m + kA_T$ โดยที่เราสมมติว่า P_m เป็นเอกสารที่นำมาเข้ารหัส และ $P_m \in E$
ขั้นตอนที่ 5	ส่ง (C_1, C_2) ไปให้ A
A ทำการถอดรหัส	
ขั้นตอนที่ 1	คำนวณ $a C_1 = a(kP) = (x,y)$
ขั้นตอนที่ 2	A ถอดรหัสโดยการคำนวณ $P_m = C_2 - a C_1$

ตัวอย่างที่ 5.5 แสดงการเข้ารหัสแบบ EC Elgamal โดยที่ E : $y^2 + xy = x^3 + (1100)x^2 + (0001) \bmod (10011)$, 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 ซึ่งการสุ่มทุกครั้งทำให้รหัสที่ใช้ในการเข้ารหัสจะเปลี่ยนไปทุกครั้งแม้ว่าจะใช้กุญแจสาธารณะ ของ A ค่าเดิม ทำให้รหัสมีความปลอดภัยมากขึ้นซึ่งเป็นข้อดีของ ECC โดยสมมติว่า k = (1011)

- B ทำการคำนวณ $k.P = (1011).(1100,0101) = (1111,1111) = C_1$

- B ทำการคำนวณ $k.A = (1011).(1010,0111) = (1000,0101)$

- B ทำการคำนวณ $C_2 = P_m + kA = (0110,1000) + (1000,0101) = (1010,0111)$

- B ส่ง $(C_1, C_2) = ((1111,1111),(1010,0111))$ ไปให้ A

- A ทำการคำนวณ $a.C_1 = (0111).(1111,1111) = (1000,0101)$

- A ทำการคำนวณ $P_m = C_2 - a C_1 = (1010,0111) - (1000,0101) = (1010,0111) + (1000,1101) = (0110,1000)$ ซึ่งจะได้ข้อมูลที่ B ส่งให้มา

5.6 อัลกอริทึมมาตรฐานของ ECC

5.6.1 อัลกอริทึม Diffie-Hellman

อัลกอริทึมของ Diffie-Hellman นั้นเป็นอัลกอริทึมพื้นฐานของรหัสลับระบบกุญแจสาธารณะที่มีจุดประสงค์เพื่อใช้ทำกุญแจปกปิดร่วมกัน โดยที่ A(Alice) และ B(Bob) นั้นต้องยอมรับการใช้พารามิเตอร์ที่ใช้ในการเข้ารหัสร่วมกัน เช่น ขนาดของกุญแจรหัส, ชนิดของคณิตศาสตร์ที่ใช้, คู่ลำดับที่ใช้ เป็นต้น ซึ่งหลักการทำงานแสดงตารางที่ 5.5

ตารางที่ 5.5 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 ด้วยเช่นกัน
ขั้นตอนที่ 3	ทั้ง A และ B ต่างคำนวณกุญแจปกปิดที่ใช้ร่วมกันคือ $Q = k_Ak_BG = k_Bk_AG$

5.6.2 อัลกอริทึม ECIES

อัลกอริทึม ECIES(Elliptic Curve Integrated Encryption Scheme) นั้นเป็นการเข้ารหัสแบบ ElGamal แบบหนึ่งที่มีประสิทธิภาพมาก และ ใช้กันเป็นมาตรฐานในการเข้ารหัสและถอดรหัสลับเอกสาร โดยถ้า B ต้องการส่งเอกสาร(m_1,m_2) B จะเข้ารหัสข้อความ(m_1,m_2) ด้วยกุญแจสาธารณะของ A แล้วส่งไปให้ A เมื่อ A ได้รับเอกสารลับ A จะสามารถถอดรหัสได้ด้วยกุญแจส่วนตัวของตัวเอง การทำงานของ อัลกอริทึม อัลกอริทึม ECIES เป็นดังรูปที่ 5.6

ตารางที่ 5.6 อัลกอริทึมของ ECIES

A ทำการสร้างกุญแจเข้ารหัส	
ขั้นตอนที่ 1	เลือก Field(n)
ขั้นตอนที่ 2	กำหนด E ซึ่งก็คือสมการ $y^2 + xy = x^3 + ax^2 + b \bmod f(x)$ 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 $c_2 = m_2.y \bmod n$
ขั้นตอนที่ 5	ส่ง (c_1, c_2,P) ไปให้ A
A ทำการถอดรหัส	
ขั้นตอนที่ 1	คำนวณ aP = a(kG) = (x,y)
ขั้นตอนที่ 2	A ถอดรหัสโดยการคำนวณ $m_1 = c_1.x^{-1} \bmod n$ และ $m_2 = c_2.y^{-1} \bmod n$

5.6.3 อัลกอริทึม ECDSA

อัลกอริทึมของ ECDSA นั้นใช้ในการลงลายเซ็นดิจิตอล และ การตรวสสอบความถูกต้องของข้อมูล ซึ่งถ้า A ต้องการส่งเอกสารไปให้ B เพื่อเป็นการยืนยันว่าเป็น A จริง A ต้องส่งลายเซ็นดิจิตอลไปพร้อมกับข้อมูลด้วย โดยการนำเอกสารไปเข้าฟังก์ชันแฮส ซึ่งจะได้ไดเจสของเอกสาร จากนั้น A จะเข้ารหัสไดเจสของข่าวสารด้วยกุญแจส่วนตัว ซึ่งจะได้ลายเซ็นดิจิตอล จากนั้นจึงส่งลายเซ็นดิจิตอลไปพร้อมกับเอกสารปกติส่งไปให้ B ซึ่ง B จะทำการตรวจสอบลายเซ็นและความถูกต้องของข้อมูลด้วยกุญแจสาธาระของ A การทำงานของอัลกอริทึม ECDSA แสดงตารางที่ 5.7

ตารางที่ 5.7 อัลกอริทึมของ ECDSA

A ทำการสร้างกุญแจ	
ขั้นตอนที่ 1	เลือก Field(n)
ขั้นตอนที่ 2	กำหนด E ซึ่งก็คือสมการ $y^2 + xy = x^3 + ax^2 + b \bmod f(x)$ โดยเลือกค่า $a, b \in 2^m$
ขั้นตอนที่ 3	สุ่มคู่ลำดับ G ที่ทำให้ E เป็นจริง
ขั้นตอนที่ 4	เลือกเลขจำนวนเต็ม d ซึ่งอยู่ในช่วง [1,n -1]
ขั้นตอนที่ 5	ทำการคำนวณ $Q = dG$
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 $kG = (x_1, y_1)$
ขั้นตอนที่ 3	คำนวณ $r = x_1 \bmod n$ (โดนที่ 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	ซึ่งจะได้ลายเซ็นดิจิตอลของเอกสาร m คือ (r, s)
B ตรวจสอบความถูกต้องของข้อมูล	
ขั้นตอนที่ 1	B ขอกุญแจสาธารณะของ A คือ Q,E,n,G
ขั้นตอนที่ 2	ตรวสสอบว่า r และ s อยู่ในช่วง [1,n-1]
ขั้นตอนที่ 3	คำนวณ $w = s^{-1} \bmod n$ และ h(m)
ขั้นตอนที่ 4	คำนวณ $u_1 = h(m)w \bmod n$ และ $u_2 = rw \bmod n$
ขั้นตอนที่ 5	คำนวณ $u_1G + u_2Q = (x_0, y_0)$ และ $v = x_0 \bmod n$
ขั้นตอนที่ 6	B จะยอมรับเอกสารนี้ถ้า $v = r$

ตารางที่ 5.8 มาตรฐานที่สอดคล้องกันของ ECC

Standard	Schemes included
ANSI X9.62	ECDSA
ANSI X9.63	ECIES, ECDH, ECMQV
FIPS 186-2	ECDSA
IEEE P1363	ECDSA, ECDH, ECMQV
IEEE P1363A	ECIES
IPSec	ECDSA, ECDH
ISO 14888-3	ECDSA
ISO 15946	ECDSA, ECDH, ECMQV

บทที่ 6
การออกแบบและสถาปัตยกรรมของระบบที่ออกแบบ

6.1 การออกแบบวงจรคำนวณทางคณิตศาสตร์จำกัด (Finite Field Arithmetic)

ในระบบของ ECC ที่ใช้คณิตศาสตร์สนามจำกัดในรูปของ Binary Field ในการแทนความสัมพันธ์ทางคณิตศาสตร์ของสมาชิกใน Elliptic Curve จะมีการทำโอเปอร์เรชั่นทางคณิตศาสตร์อยู่ 3 อย่างที่สำคัญคือ การคูณ การหากำลังสอง และ การหาส่วนกลับ โดยการออกแบบโอเปอร์เรเตอร์ ของวงจรคูณ วงจรกำลังสอง และ วงจรหาส่วนกลับ นี้จะมีผลต่อขนาดและความเร็วในการทำงานของระบบเป็นอย่างมาก ซึ่งสามารถออกแบบได้ดังต่อไปนี้

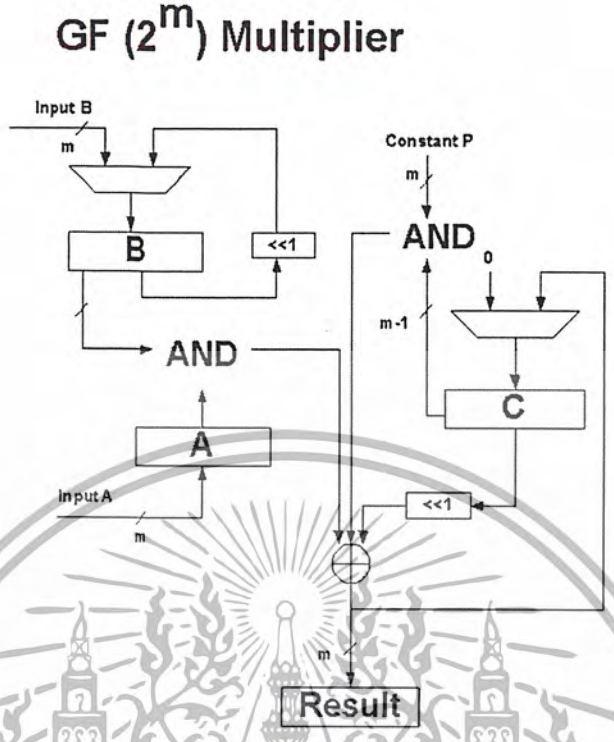
6.1.1 การออกแบบวงจรคูณ

การออกแบบวงจรคูณของ Elliptic Curve นั้นไม่เหมือนกับการออกแบบวงจรคูณทางคณิตศาสตร์ หรือ การออกแบบวงจรดิจิตอลทั่วๆ ไป เนื่องจากการทำโอเปอเรชั่น ของ Elliptic Curve นั้นทำงานอยู่บนคณิตศาสตร์สนามจำกัด ซึ่งเมื่อค่าของผลลัพธ์ที่ได้จากการคูณเกินขอบเขตฟิลด์ไป ผลลัพธ์นั้นจะถูกโมดูโล ด้วยค่าขอบเขตของฟิลด์เพื่อให้ค่าที่เกินขอบเขตของฟิลด์นั้น กลับมาอยู่ในฟิลด์ เช่น การคูณด้วยคณิตศาสตร์ธรรมดา เราจะได้ 3 x 6 = 18 แต่ถ้าเป็นคณิตศาสตร์สนามจำกัดแล้วจะต้องมีขอบเขตของสนามจำกัด ซึ่งค่าที่ได้จะกลายเป็น 3 x 6 mod 11 = 7 ซึ่งก็คือค่าของเศษที่เกิดจากผลคูณหารด้วยขอบเขตของสนาม การออกแบบวงจรคูณในโครงงานนี้ได้ใช้อัลกอริทึม GF(2^m) Multiplication ซึ่งมีการทำงานตามตารางที่ 6.1

ตารางที่ 6.1 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 6.1 นั้น m จะเป็นขนาดของบิตที่ใช้ ส่วน P(x) เป็น Polynomial Reduction ซึ่งโครงสร้างทางฮาร์ดแวร์ ของอัลกอริทึม GF(2^m) Multiplication จะแสดงได้ดังรูปที่ 6.1 โดยจะใช้ Clock ในการประมวลผลเท่ากับ m+1 ลูก



รูปที่ 6.1 แสดงโครงสร้างของวงจรคูณแบบ GF(2^m) Multiplication

6.1.2 การออกแบบวงจรกำลังสอง

จากอัลกอลิทึมของ ECC over F_2m ในรูปของ Polynomial basis การคูณกันของสองอิลิเมนท์ $A,B \in F_2m$ สามารถแสดงได้ในรูปของสมการ

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

จากสมการ (1) นั้น ที่ $a_i = 0$ และ $b_i = 0$ สำหรับ $i \geq m$ ทำให้แต่ละระดับของการคูณใน F_2 นั้นอยู่ในรูป AND โอเปอร์เรชั่นของสมการบูลีน จะเห็นได้ว่า ค่าดีกรีที่มากที่สุดที่เกิดจากผลคูณของโพลิโนเมียล $C(x) = A(x).B(x)$ เท่ากับ $2m-2$ อย่างไรก็ตาม เราสามารถลดบิตสตริงให้เท่า m ได้โดยการโมดูโลด้วย โพลิโนเมียลลดรูป f(x) ซึ่งอยู่ในรูป

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

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

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

ซึ่งทำให้

$$C(x) = \sum_{i=0}^{2m-2} c_i x^i$$

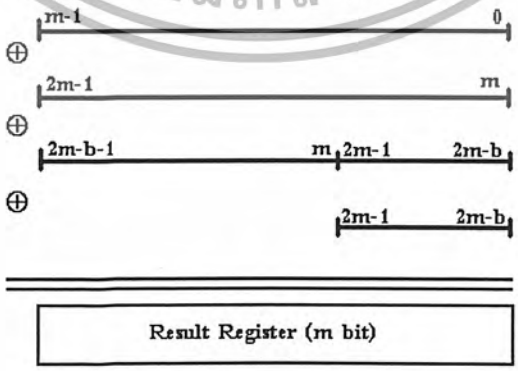
$$\equiv \sum_{i=0}^{m-1} c_i x^i + \sum_{i=m}^{2m-2} c_i (x^{b+i-m} + x^{i-m}) \bmod f(x)$$

$$= \sum_{i=0}^{m-1} c_i x^i + \sum_{i=0}^{m-1-b} c_{i+m} x^{b+i} + \sum_{i=m-b}^{m-1} c_{i+m} x^{b+i} + \sum_{i=0}^{m-1} c_{i+m} x^i$$

$$\equiv \sum_{i=0}^{m-1} c_i x^i + \sum_{i=0}^{m-1-b} c_{i+m} x^{b+i} + \sum_{i=m-b}^{m-1} c_{i+m} (x^{2b+i-m} + x^{b+i-m}) + \sum_{i=0}^{m-1} c_{i+m} x^i \bmod f(x)$$

$$= \sum_{i=0}^{m-1} c_i x^i + \sum_{i=0}^{m-1-b} c_{i+m} x^{b+i} + \sum_{i=0}^{b-1} c_{2m-b+i} x^{b+i} + \sum_{i=0}^{b-1} c_{2m-b+i} x^i + \sum_{i=0}^{m-1} c_{m+i} x^i \quad (3)$$

ผลลัพธ์จากสมการ(3) นั้นจะทำให้มีจำนวน exclusive or เท่ากับ 2m+b สำหรับหนึ่งโพลิโนเมียลลดรูป ซึ่งแสดงโครงสร้างดังรูปที่ 6.2 นั่นคือวงจรโพลิโนเมียลลดรูปสามารถนำไปใช้ในการหากำลังสองได้โดย การแทรกบิต "0" ระหว่างกลาง เช่น (1111)2 = 1010101 ก่อนนำไปลดรูปด้วยโครงสร้างดังกล่าว ซึ่งส่งผลให้เวลาในการหากำลังสองจะใช้ clock เพียงลูกเดียว



รูปที่ 6.2 แสดงโครงสร้างของการลดรูปโพลิโนเมียล

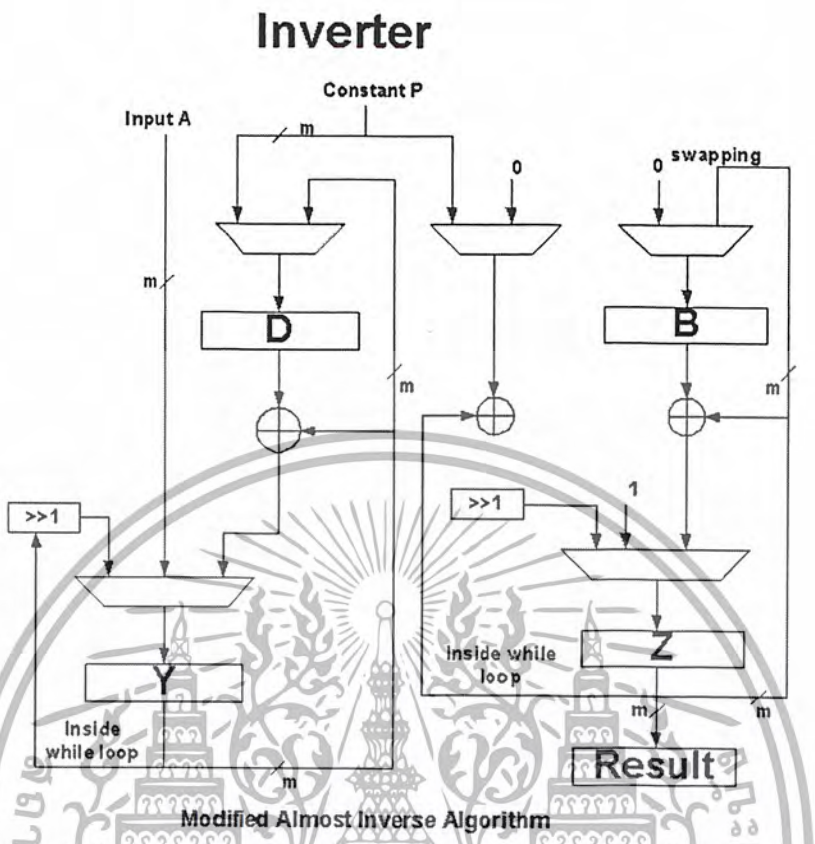
6.1.3 การออกแบบวงจรหาส่วนกลับ

การออกแบบวงจรหาส่วนกลับที่ใช้ในระบบ ECC นั้นเป็นส่วนที่สำคัญมาก เนื่องจากวงจรหาส่วนกลับนั้น เป็นส่วนที่มีขนาดใหญ่กว่าวงจรคูณและทำงานช้ากว่าวงจรคูณอยู่มาก ทำให้ความเร็วของระบบโดยรวมจึงขึ้นกับการออกแบบวงจรหาส่วนกลับว่าจะสามารถออกแบบให้ทำงานได้เร็ว และมีขนาดของวงจรเป็นอย่างไร ในโครงงานนี้ได้ใช้อัลกอริทึมแบบ Modifield Almost Inverse Algorithm (MAIA) ซึ่งมีการทำงานตามตารางที่ 6.2

ตารางที่ 6.2 อัลกอริทึม Modifield Almost Inverse Algorithm (MAIA)

Input : Binary Polynomial $A(x) \in GF(2^m)$	
Output : $C(x) = A(x)^{-1} \bmod P(x)$	
ขั้นตอนที่ 1	$Y(x) = A(x)$, $D(x) = P(x)$, $B(x) = 0$, $C(x) = 1$
ขั้นตอนที่ 2	Loop
ขั้นตอนที่ 3	While $y_0 = 0$ Do
ขั้นตอนที่ 4	$Y(x) = Y(x) >> 1$
	$X(x) = (X(x) + c_0.P(x)) >> 1$
ขั้นตอนที่ 5	End While
ขั้นตอนที่ 6	If ($Y(x) = 1$) Then
	Return $C(x)$
ขั้นตอนที่ 7	If ($D(x) > Y(x)$) Then
	$D(x) <=> Y(x)$, $B(x) <=> C(x)$
ขั้นตอนที่ 8	$Y(x) = Y(x) + D(x)$, $C(x) = C(x) + B(x)$
ขั้นตอนที่ 9	End Loop

จากตารางที่ 6.2 นั้น อัลกอริทึมของ MAIA สามารถมองเป็นโครงสร้าง ดังรูปที่ 6.3 โดยความเร็วในการทำงานนั้นจะไม่แน่นอน ขึ้นอยู่กับค่า A(x) แต่โดยเฉลี่ยแล้วจะใช้เวลาโดยประมาณเท่ากับ 3-4 เท่าของความเร็วของวงจรคูณ เช่นเดียวกับขนาดของวงจร ก็จะมีขนาดใหญ่กว่าวงจรคูณอยู่ 3-4 เท่าเช่นเดียวกัน ซึ่งถือว่าไม่ใหญ่ จนเกินไปเมื่อเทียบกับ อัลกอริทึมของวงจรหาส่วนกลับอีกหลายๆ แบบ แต่ข้อเสียของอัลกอริทึม MAIA ที่เห็นได้ชัดคือ แม้ว่าเวลาการทำงานจะเร็วเพียง 3 – 4 เท่าของวงจรคูณ แต่ว่าเนื่องจากไม่มีเวลาที่แน่นอนในการประมวลผล บางครั้งอาจเร็ว บางครั้งอาจช้า ทำให้ประสิทธิภาพที่แท้จริงของระบบไม่แน่นอน



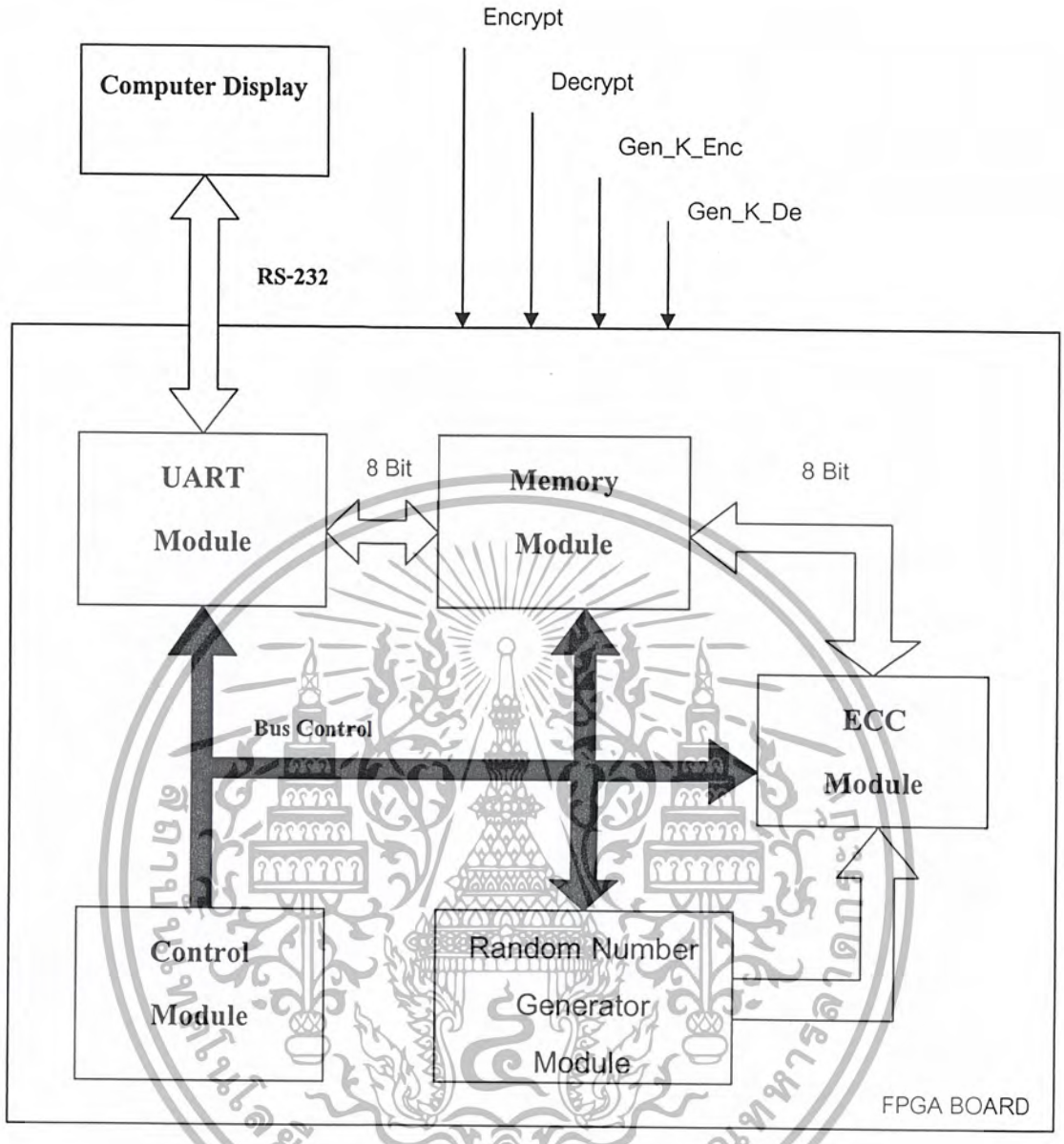
รูปที่ 6.3 แสดงโครงสร้างของวงจรหาส่วนกลับแบบ Modifield Almost Inverse Algorithm

6.2 การออกแบบระบบ ECC

6.2.1 สถาปัตยกรรมของระบบ ECC

สถาปัตยกรรมของระบบที่ออกแบบประกอบไปด้วยส่วนประกอบต่างๆ คือ ส่วนของการสื่อสารข้อมูล (UART Module) ส่วนของหน่วยความจำ (Memory Module) ส่วนประมวลผลของ Elliptic Curve (ECC Module) ส่วนของการสุ่มเลขเทียม (Random Number Generator Module) และ ส่วนของการควบคุม (Control Module) สามารถแสดงได้ดังรูปที่ 6.4

ในการออกแบบได้ทำการออกแบบให้สมาร์ทการ์ดรับและส่งข้อมูลดิจิตอลจากคอมพิวเตอร์ผ่านทางช่องสื่อสารอนุกรม ตามมาตรฐาน RS-232 โดยใช้ออสซิเลเตอร์ขนาด 6 MHz ทำหน้าที่เป็นสัญญาณนาฬิกาของสมาร์ทการ์ด จากรูปที่ 6.4 ข้อมูลจากคอมพิวเตอร์ที่ส่งเข้ามา จะเข้าไปยังส่วนของการสื่อสารข้อมูล (UART Module) เพื่อทำการแปลงข้อมูลตามโปรโตคอลของ RS-232 ให้เป็นข้อมูลดิจิตอลขนาด 8 บิต แล้วจึงเก็บไว้ยังส่วนของหน่วยความจำ (Memory Module) เพื่อที่จะรอคำสั่งจากหน่วยควบคุม (Control Module) ทำการส่งข้อมูลจากหน่วยความจำไปที่ส่วนประมวลผลของ Elliptic Curve จากนั้นผลลัพธ์ที่ได้จะถูกส่งกลับไปเก็บไว้ยังส่วนของหน่วยความจำ ก่อนที่หน่วยของการสื่อสารข้อมูลจะส่งข้อมูลออกไปยังคอมพิวเตอร์ตามโปรโตคอลของ RS-232



รูปที่ 6.4 แสดงส่วนประกอบต่างๆของสถาปัตยกรรมเพื่อจำลองการทำงานของสมาร์ทการ์ด

ฟังก์ชันการทำงานของระบบนั้น ได้ออกแบบให้มีการทำงานอยู่ 4 ฟังก์ชันคือ ฟังชันการเข้ารหัสข้อมูล (Encrypt) ฟังก์ชันถอดรหัสข้อมูล(Decrypt) ฟังก์ชั่นสร้างกุญแจเข้ารหัส(Gen_K_En) และฟังก์ชันสร้างกุญแจถอดรหัสข้อมูล(Gen_K_De) ซึ่งการที่จะทำงานฟังก์ชันใดส่วนการควบคุมจะเป็นตัวกำหนด

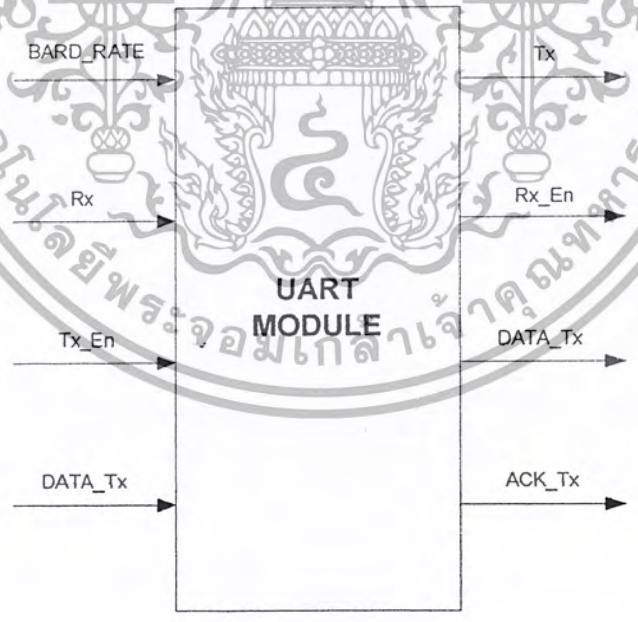
บทที่ 7

การทดสอบและผลการทดสอบ

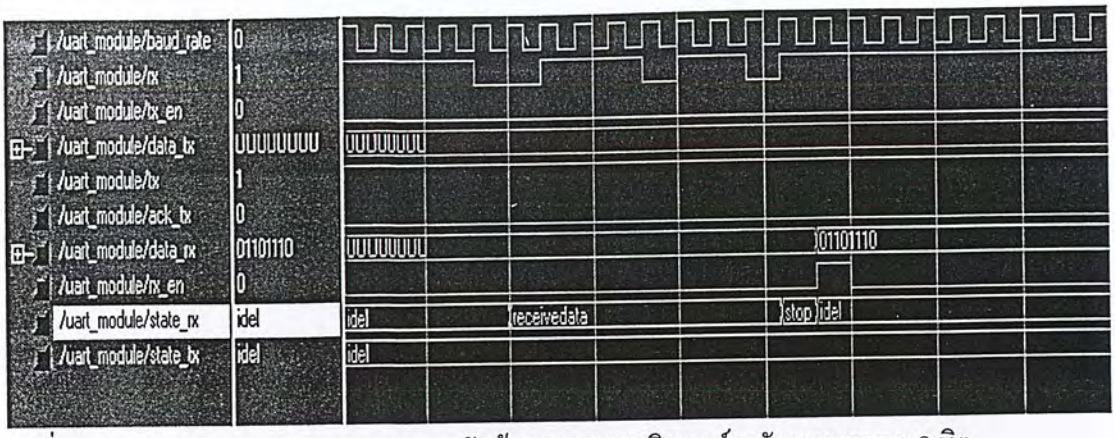
บทนี้ได้กล่าวถึงการทดสอบและผลการทดสอบส่วนประกอบต่างๆ ของสถาปัตยกรรม 5 ส่วน โดยทดสอบกับซอร์ฟแวร์จำลองการทำงาน และ ทำการทดสอบการทำงานจริงบนอุปกรณ์เอฟพีจีเอเบอร์ XC2S200-PQ208 โดยส่วนของประมวลผล ECC นั้นจะทดสอบการทำงานของระบบเข้ารหัสตามตัวอย่างที่ 5.5 ในบทที่ 5

7.1 การทดสอบส่วนการสื่อสารข้อมูล

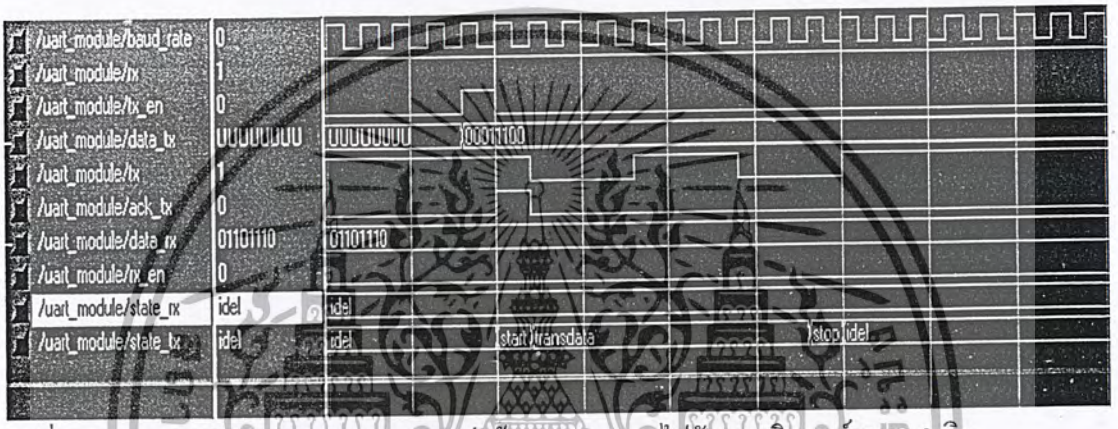
ส่วนของการสื่อสารข้อมูล จะเป็นการสื่อสารแบบอนุกรม ตามมาตรฐาน RS-232 ระหว่างระบบที่ออกแบบกับ คอมพิวเตอร์ โดยใช้ Baud Rate ที่ 9,600 บิตต่อวินาที ตามมาตรฐานของสมาร์ทการ์ด โดยส่วนรับข้อมูลจะรับข้อมูลจากคอมพิวเตอร์ทางขาสัญญาณ RX จากนั้นเมื่อรับข้อมูลจนครบ 8 บิตจะทำการส่งข้อมูลออกไปยังส่วน Control Module ทางขาสัญญาณ DATA_RX ซึ่งมีขนาด 8 บิต โดยผลการจำลองแสดงดังรูปที่ 7.2 และส่วนที่ทำการส่งข้อมูลจะรับข้อมูลจาก Control Module ทางขาสัญญาณ DATA_TX ซึ่งมีขนาด 8 บิต โดยจะมีสัญญาณ TX_EN จาก Control Module เป็นสัญญาณแจ้งให้รับข้อมูล จากนั้น UART Module จะทำการตอบรับ Control Module ทางขาสัญญาณ Ack Tx และส่งข้อมูลไปยังคอมพิวเตอร์ทางขาสัญญาณ TX โดยผลการจำลองแสดงดังรูปที่ 7.3 ซึ่งบล็อคไดอะแกรมที่ออกแบบแสดงดังรูปที่ 7.1



รูปที่ 7.1 แสดงบล็อกไดอะแกรมของส่วนการสื่อสารข้อมูล



รูปที่ 7.2 แสดง Timming Diagram ของการรับข้อมูลจากคอมพิวเตอร์มายังระบบขนาด 8 บิต



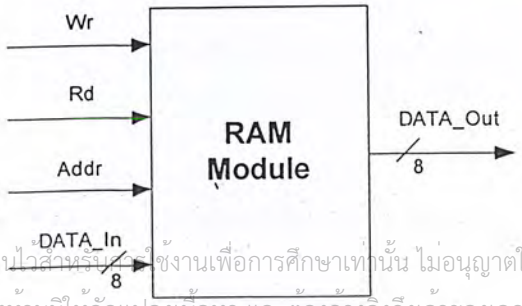
รูปที่ 7.3 แสดง Timming Diagram ของการส่งข้อมูลจากระบบไปยังคอมพิวเตอร์ขนาด 8 บิต

7.2 การทดสอบส่วนของหน่วยความจำ

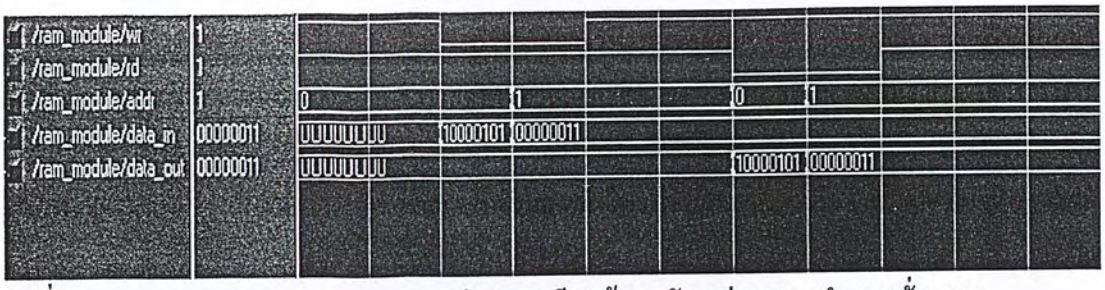
ส่วนของหน่วยความจำนั้นจะประกอบไปด้วยหน่วยความจำ 2 แบบด้วยกันคือ หน่วยความจำแบบชั่วคราว และ หน่วยความจำแบบถาวร

7.2.1 การทดสอบหน่วยความแบบชั่วคราว

ส่วนของหน่วยความจำแบบชั่วคราวนั้นจะใช้ในการเก็บค่าของกุญแจที่ส่วน ECC Module ประมวลผลในขั้นตอนสร้างกุญแจเข้ารหัส และ สร้างกุญแจถอดรหัส เพื่อรอข้อมูลที่จะส่งมายังคอมพิวเตอร์เพื่อทำการเข้ารหัสและถอดรหัสต่อไป โดยการทำงานจะถูกควบคุมโดย Control Module โดยบล๊อคไดอะแกรมที่ออกแบบแสดงดังรูปที่ 7.4 และผลการจำลองการอ่านและการเขียนข้อมูลกับหน่วยความจำแบบชั่วคราวแสดงดังรูปที่ 7.5



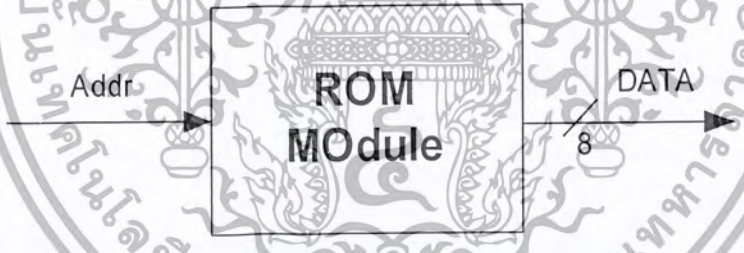
รูปที่ 7.4 แสดงบล็อกไดอะแกรมของส่วนหน่วยความจำแบบชั่วคราว



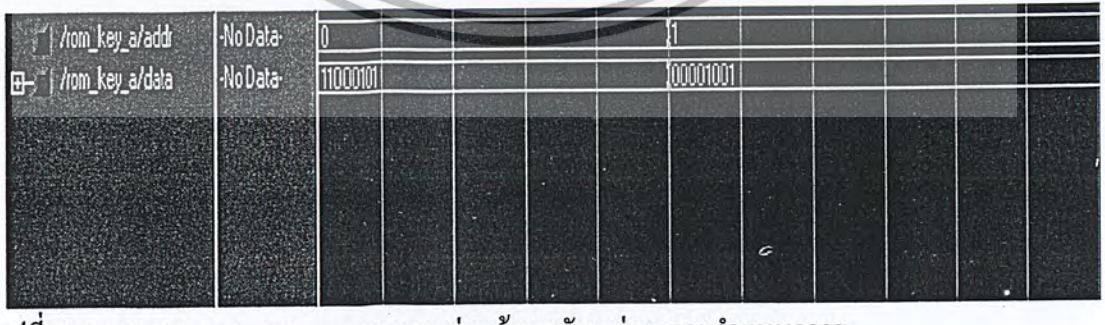
รูปที่ 7.5 แสดง Timming Diagram ของการอ่านและเขียนข้อมูลกับหน่วยความจำแบบชั่วคราว

7.2.2 การทดสอบหน่วยความแบบถาวร

ส่วนของหน่วยความจำแบบถาวรนั้นจะใช้ในการเก็บค่าของกุญแจส่วนตัวผู้ใช้สมาร์ทการ์ด โดยที่บัตรสมาร์ทการ์ดแต่ละใบนั้นจะมีกุญแจที่ต่างกัน และเก็บค่าจุดหลักบน Elliptic Curve คือ P(x,y) ที่ใช้ในการคำนวณกุญแจรหัส คือ k.P(x,y) โดยการอ่านข้อมูลส่วน Control Module จะทำการส่งสัญญาณระบุตำแหน่งของหน่วยความจำทางขาสัญญาณ Addr และหน่วยความจำแบบถาวรจะส่งข้อมูลออกทาง ขาสัญญาณ Data_Out ขนาด 8 บิต โดยบล็อคไดอะแกรมที่ออกแบบแสดงดังรูปที่ 7.6 และผลการจำลองการอ่านข้อมูลกับหน่วยความจำแบบถาวรแสดงดังรูปที่ 7.7



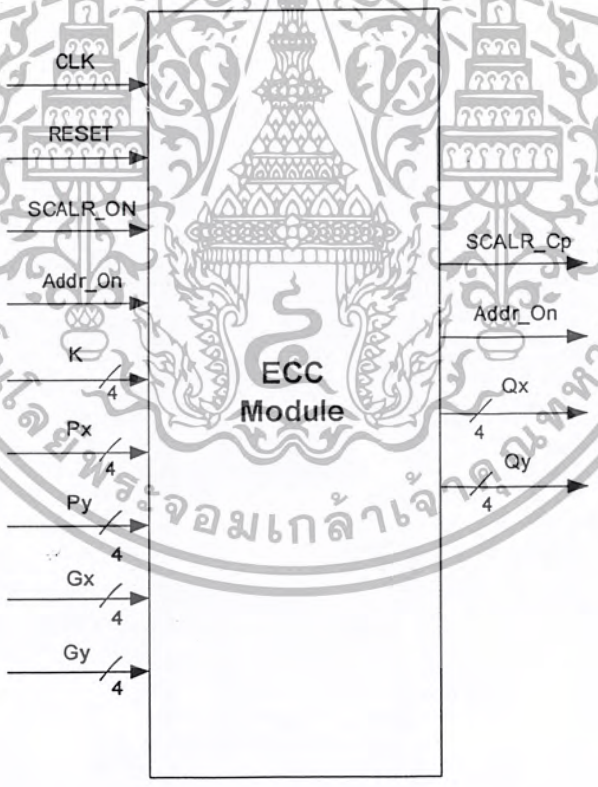
รูปที่ 7.6 แสดงบล็อกไดอะแกรมของส่วนหน่วยความจำแบบถาวร



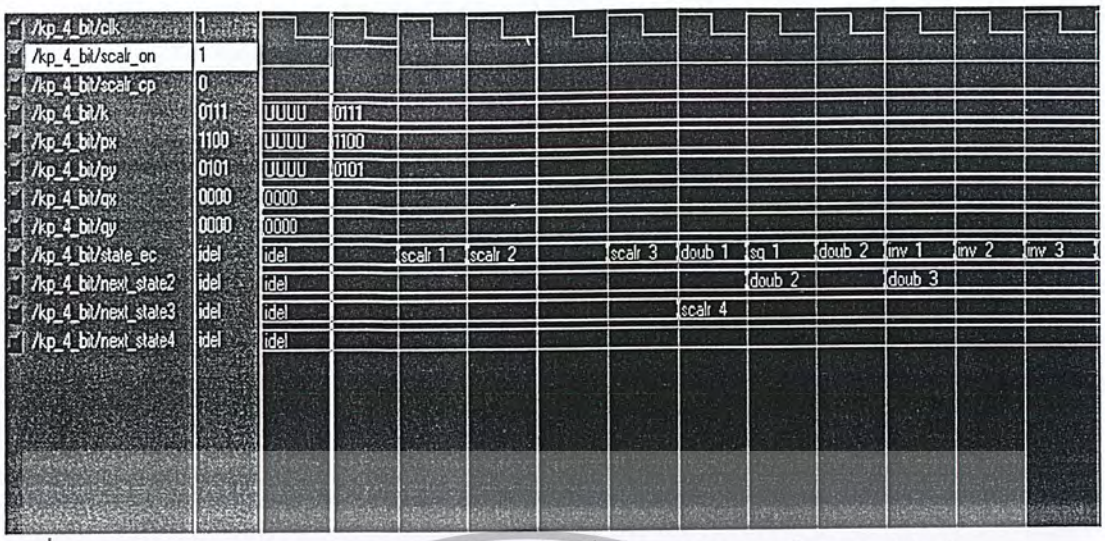
รูปที่ 7.7 แสดง Timming Diagram ของการอ่านข้อมูลกับหน่วยความจำแบบถาวร

7.3 การทดสอบส่วนประมวลผลของ Elliptic Curve

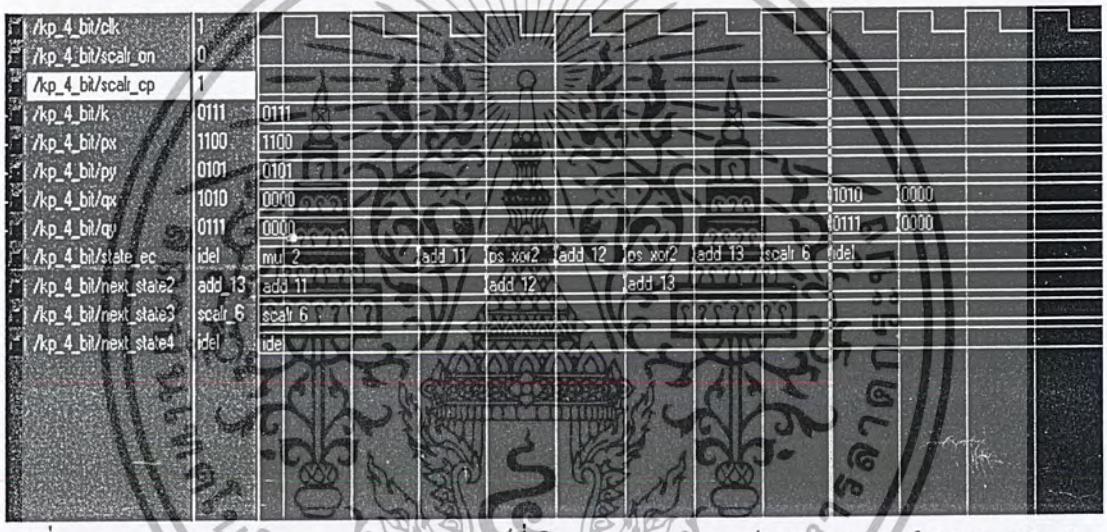
ส่วนของการประมวลผลของ Elliptic Curve ที่ใช้ในอัลกอริทึม EC ElGamal จะมีอยู่ 2 อย่างคือ การบวกจุดบน Elliptic Curve ซึ่งก็คือ Point Addition ที่ใช้ในการบวกรหัสกุญแจ กับ ข้อความที่จะนำมาเข้ารหัส กับ อย่างที่สองคือ Scalar Multiplication เป็นการคำนวณจำนวนจุดว่าจุดที่ทำการบวกกันเป็นจำนวน k ครั้งมีค่าเท่าไหร่ เช่น k.P(x,y) คือ P(x,y) + P(x,y) + ... k ครั้ง ซึ่งการบวกกัน k ครั้ง จะเกิดจากการทำการบวกจุดของ Elliptic Curve ที่มีอยู่ 2 อย่างตามที่ได้กล่าวในบทการเข้ารหัสแบบ ECC คือ Point Addition และ Point Double ในทำนองเดียวกันการบวกจุดทั้งสองแบบต่างก็อาศัยการทำโอเปอร์เรชั่น 3 แบบ คือ การคูณ การหาค่าลังสอง และ การหาส่วนกลับ ดังนั้นเมื่อเราทำการออกแบบส่วนคำนวณ Scalar Multiplication จะทำให้เราได้ โอเปอร์เรเตอร์อื่นๆ ด้วย ในผลการทดสอบจะแสดงผลการทำงานจาก Timming Diagram ตามตัวอย่างที่ 5.5 เพื่อทดสอบผลการทำงานของระบบที่ออกแบบว่าเป็นไปตามทฤษฎี หรือไม่ ซึ่งค่าที่ได้จากการออกแบบนั้นเป็นไปตามทฤษฎีทุกประการ โดยรูปที่ 7.8 เป็นการแสดงบล็อคไดอะแกรมที่ทำการออกแบบ



รูปที่ 7.8 แสดงบล็อกไดอะแกรมของส่วนการประมวลผล Elliptic Curve

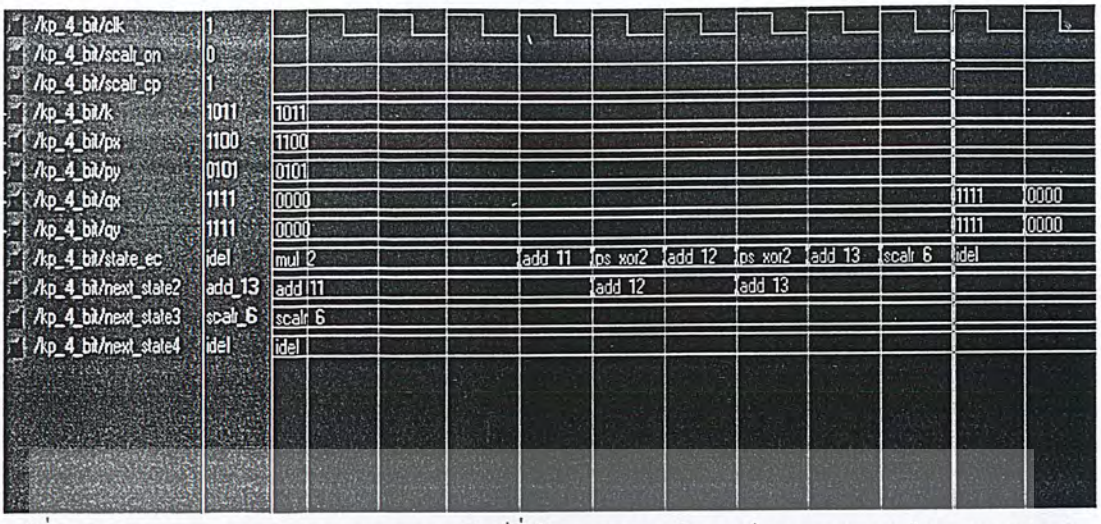


รูปที่ 7.9 แสดง Timming Diagram ของการกำหนดค่าให้ของระบบของการคำนวณ a.P(x,y)

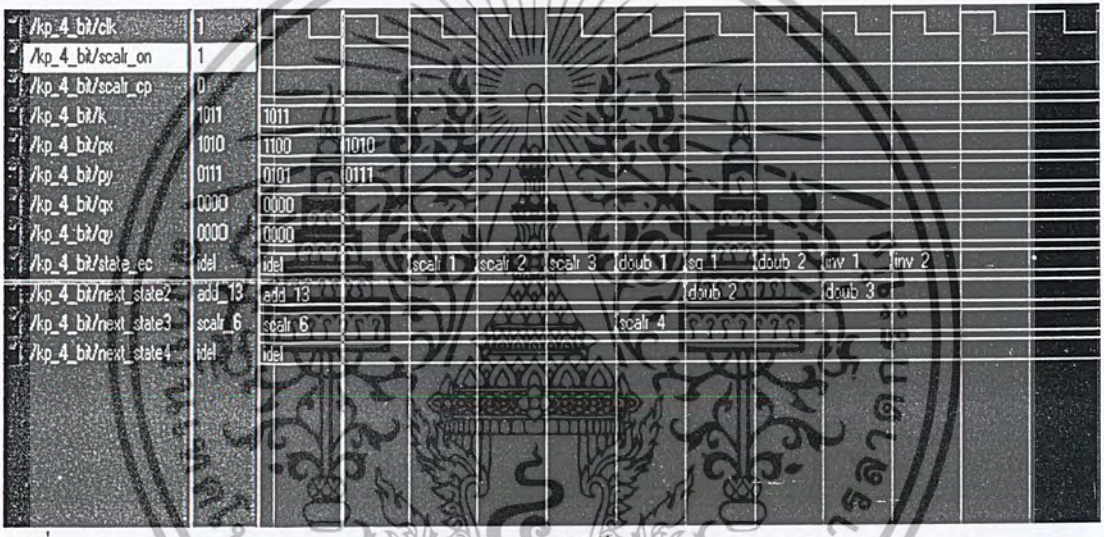


รูปที่ 7.10 แสดง Timming Diagram ของผลลัพธ์ที่เกิดจากระบบทำการประมวลผลการคำนวณ a.P(x,y)

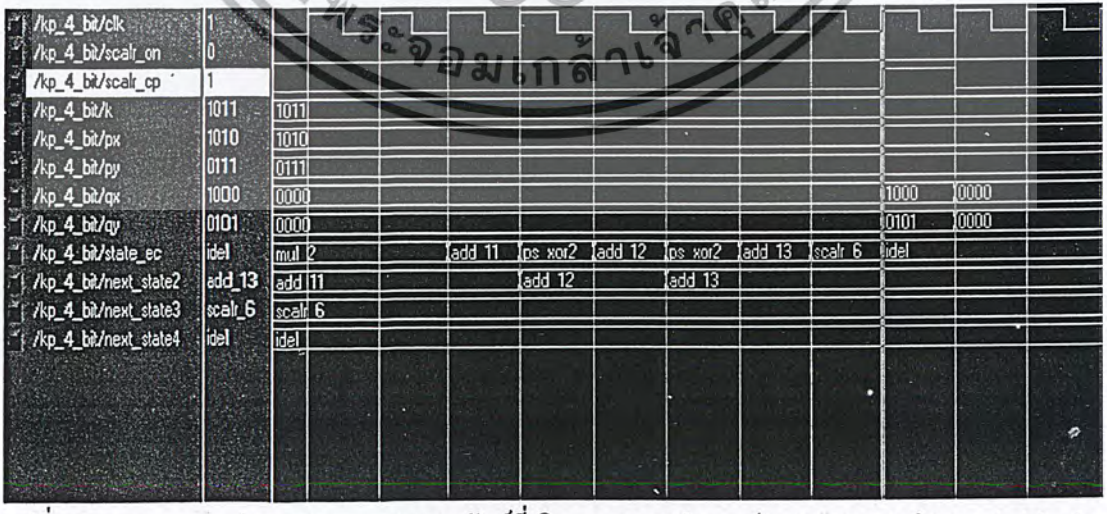
รูปที่ 7.11 แสดง Timming Diagram ของการกำหนดค่าให้ของระบบของการคำนวณ k.P(x,y)



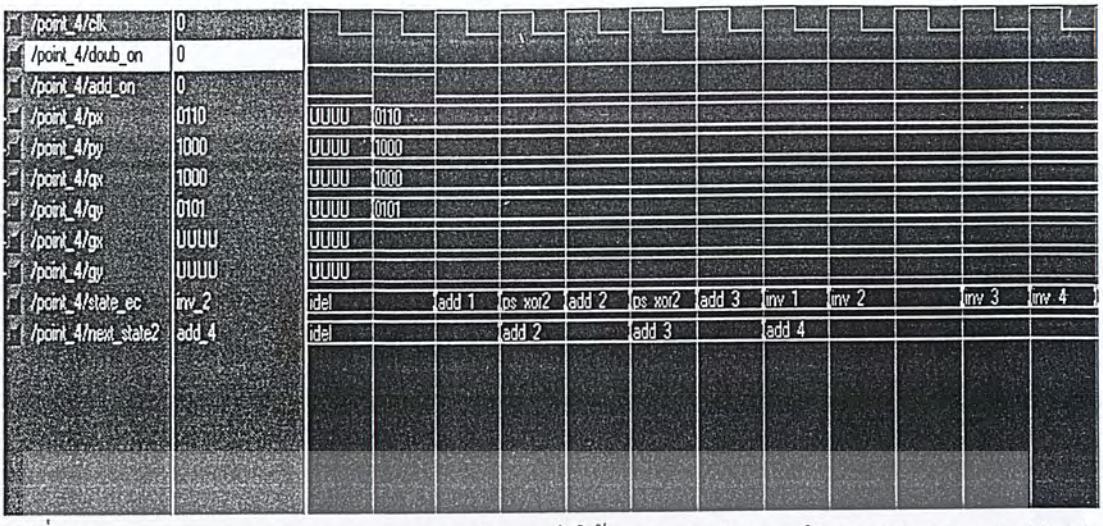
รูปที่ 7.12 แสดง Timming Diagram ของผลลัพธ์ที่เกิดจากระบบทำการประมวลผลการคำนวณ k.P(x,y)



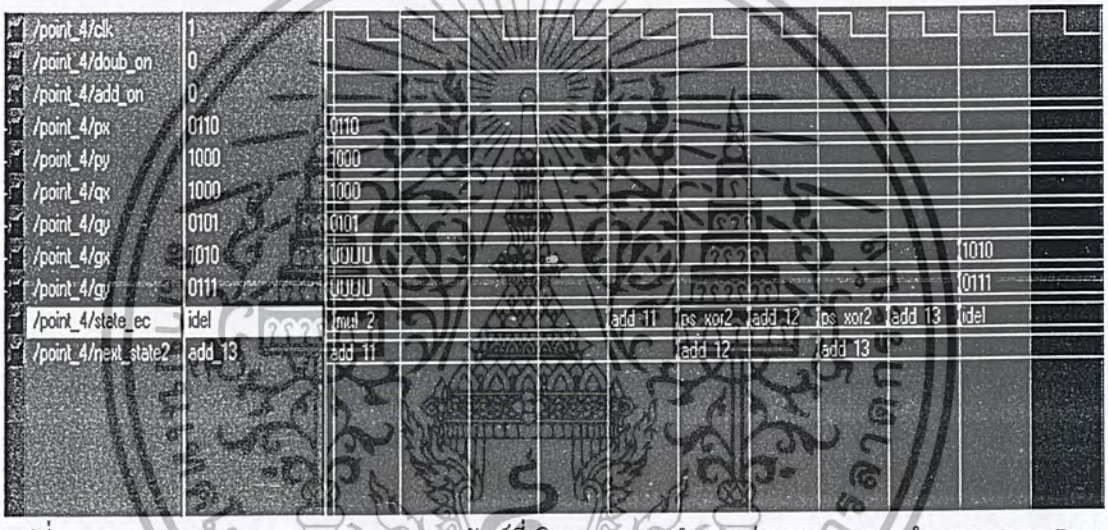
รูปที่ 7.13 แสดง Timming Diagram ของการกำหนดค่าให้ของระบบของการคำนวณ k.A(x,y)



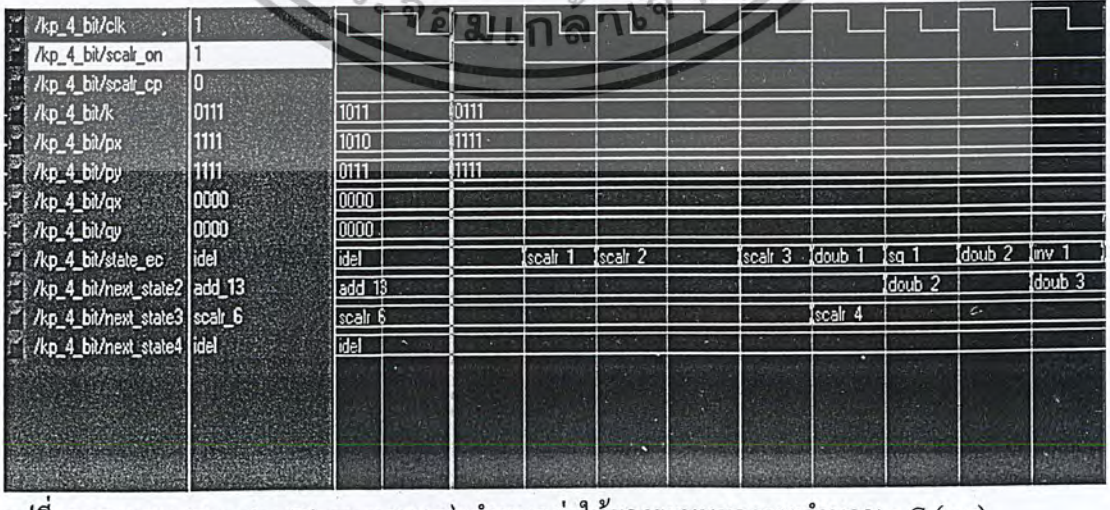
รูปที่ 7.14 แสดง Timming Diagram ของผลลัพธ์ที่เกิดจากระบบทำการประมวลผลการคำนวณ k.A(x,y)



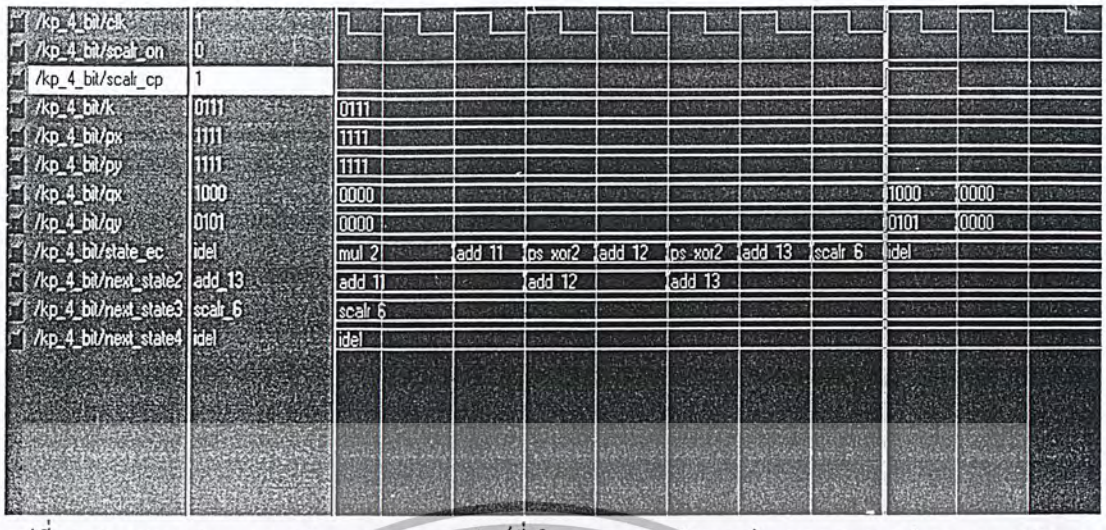
รูปที่ 7.15 แสดง Timming Diagram ของการกำหนดค่าให้ของระบบของการคำนวณ $C_2 = P_m + k.A(x,y)$



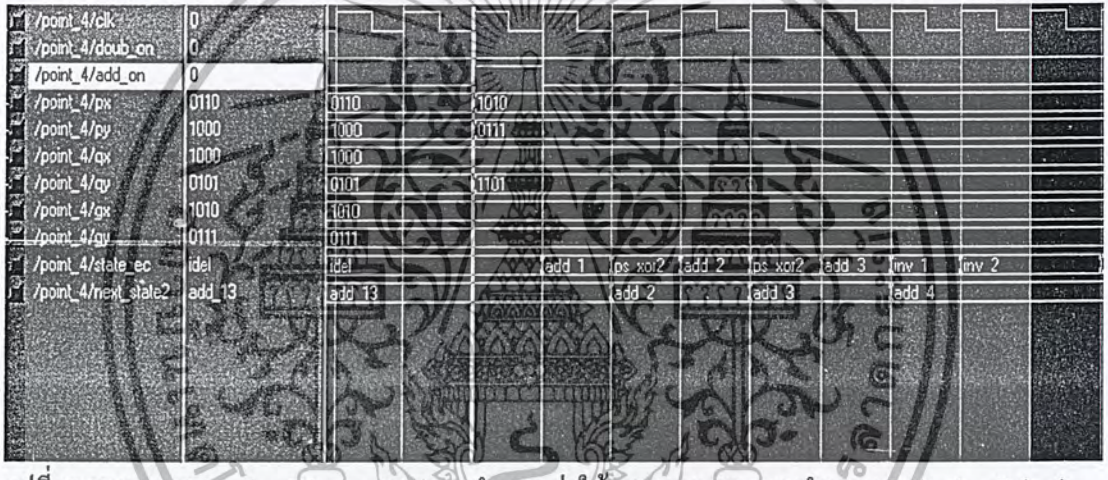
รูปที่ 7.16 แสดง Timming Diagram ของผลลัพธ์ที่เกิดจากระบบทำการประมวลผลการคำนวณ $C_2 = P_m + k.A(x,y)$



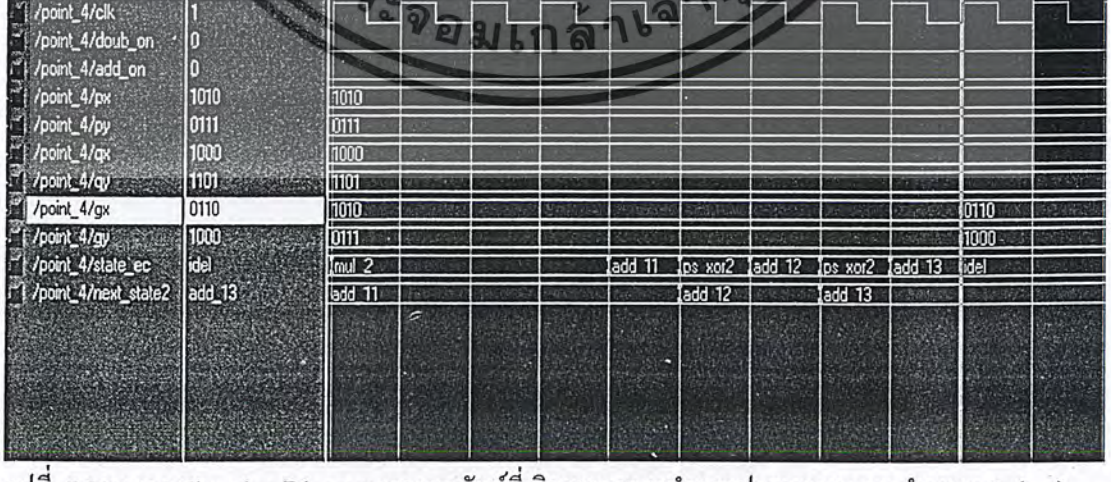
รูปที่ 7.17 แสดง Timming Diagram ของการกำหนดค่าให้ของระบบของการคำนวณ $a.C_1(x,y)$



รูปที่ 7.18 แสดง Timming Diagram ของผลลัพธ์ที่เกิดจากระบบทำการประมวลผลการคำนวณ $a.C_1(x,y)$



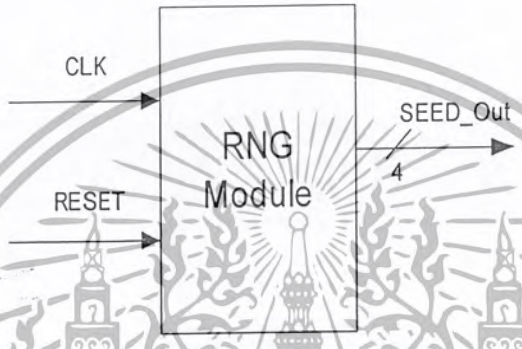
รูปที่ 7.19 แสดง Timming Diagram ของการกำหนดค่าให้ของระบบของการคำนวณ $P_m(x,y) = C_2(x,y) - a.C_1(x,y)$



รูปที่ 7.20แสดง Timming Diagram ของผลลัพธ์ที่เกิดจากระบบทำการประมวลผลการคำนวณ $P_m(x,y) = C_2(x,y) - a.C_1(x,y)$

7.4 การทดสอบส่วนของการสุ่มเลขเทียม

ส่วนของการสุ่มเลขเทียม จะใช้วิธีการสุ่มแบบ LFSR (Linear Feedback Shift Register) ซึ่งเป็นวิธีการที่สร้างได้ง่ายโดย กระบวนการสุ่มที่ทำการออกแบบจะทำการสุ่มค่าเลขอยู่ในช่วง 2 ถึง 15 ซึ่งนำไปใช้ในการทำโอเปอร์เรชั่น k.P(x,y) โดยรูปที่ 7.21 จะแสดงบล็อกไดอะแกรมที่ออกแบบ และ รูปที่ 7.22 นั้นจะแสดงผลการสุ่มดังจะเห็นว่าระบบจะทำการสุ่มไปเรื่อยๆ โดยการเลือกค่า k จะแล้วแต่ว่าการประมวลผล k.P(x,y) จะมีการทำงานเมื่อไหร่ ซึ่งการเริ่มประมวลผล k.P(x,y) จะเกิดจากเมื่อผู้ใช้ระบบต้องการเข้ารหัสข้อความซึ่งมีเวลาไม่แน่นอน ทำให้การนำค่าช่วงนั้นๆ เข้าไปจึงเป็นการสุ่มนั่นเอง



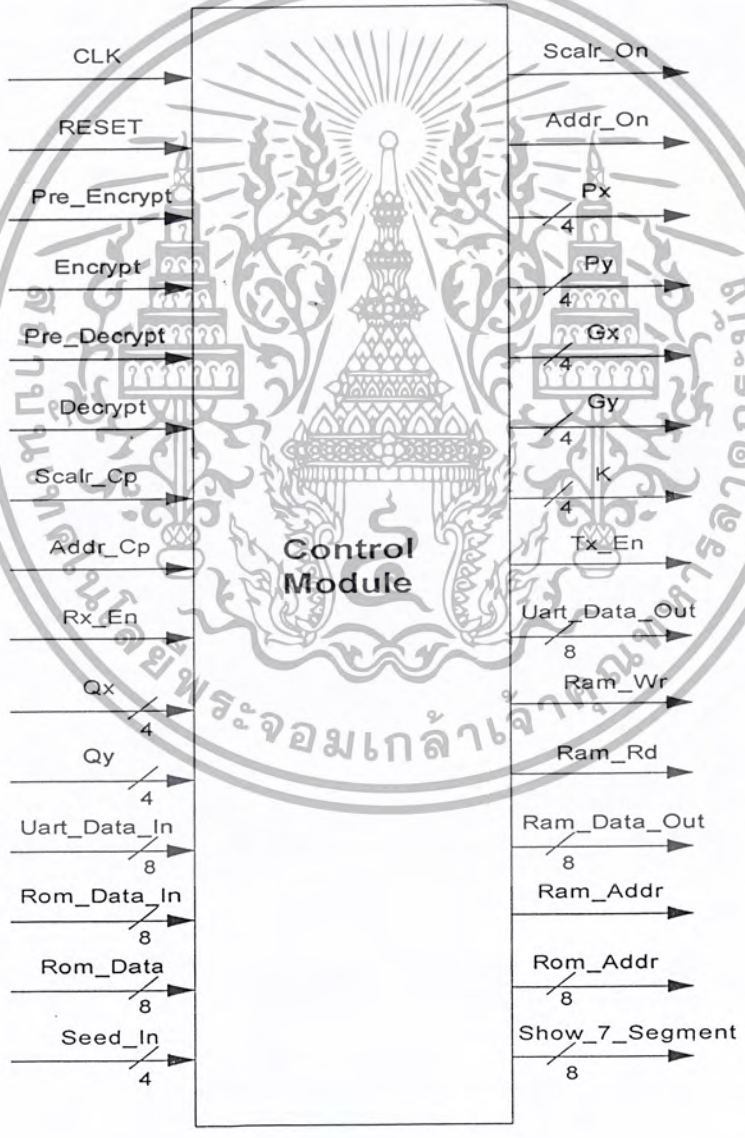
รูปที่ 7.21 แสดงบล็อกไดอะแกรมส่วนของการสุ่มเลขเทียม

รูปที่ 7.22แสดง Timming Diagram ของการกำหนดค่าเริ่มต้นให้วงจรสุ่มเลขแบบ LFSR ทำงาน และ ผลที่เกิดจากการสุ่มเลข

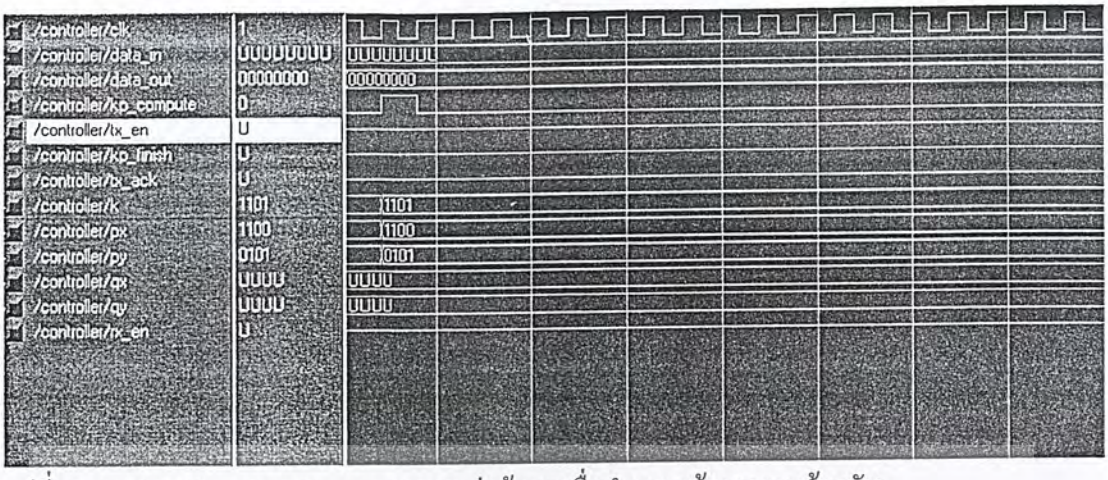
7.5 การทดสอบส่วนของการควบคุม

ส่วนของการควบคุม จะทำการควบคุมการทำงานของระบบทั้งหมด โดย รูปที่ 7.23 จะแสดงบล็อกไดอะแกรมที่ออกแบบ ซึ่งจะมีขาสัญญาณ Reset เป็นตัวเริ่มต้นทำงานใหม่ และจะมีขาสัญญาณอีก 4 ขาที่ต่อเข้ากับสวิทช์ซึ่งก็คือ 1.ขาสัญญาณ Pre_Encrypt ทำหน้าที่สร้างกุญแจเข้ารหัสแล้วเก็บค่าของ

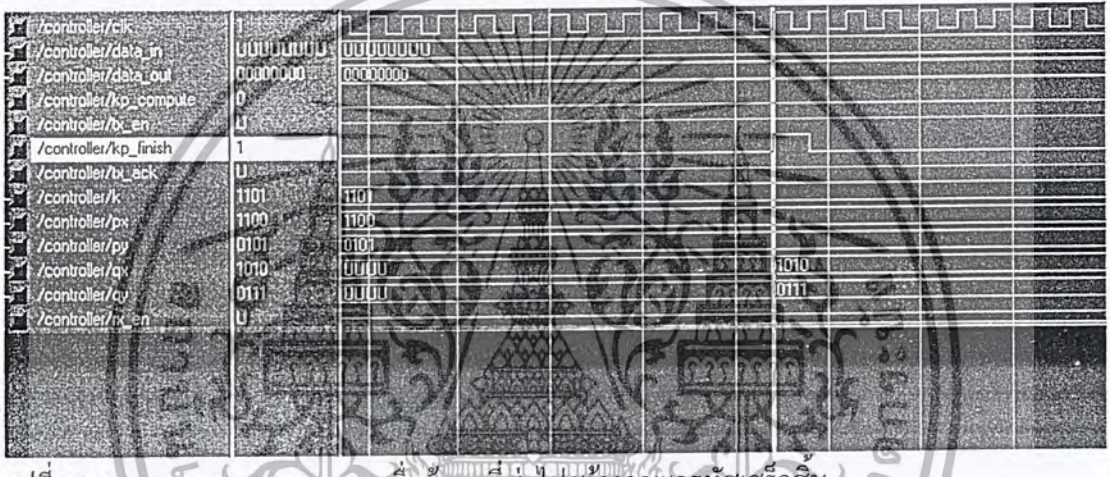
กุญแจไว้ที่หน่วยความจำแบบชั่วคราว 2.ขาสัญญาณ Encrypt ทำหน้าที่เข้ารหัสข้อมูลโดยรับข้อมูลจากส่วนการสื่อสารข้อมูลและนำไปเข้ารหัสลับกับกุญแจที่เก็บไว้ที่หน่วยความจำแบบชั่วคราว 3.ขาสัญญาณ Pre_Decrypt จะทำการสร้างกุญแจถอดรหัสและนำไปเข้ารหัสลับกับกุญแจที่เก็บไว้ที่หน่วยความจำแบบชั่วคราว 4.ขาสัญญาณ Decrypt ทำหน้าที่ถอดรหัสข้อมูลโดยรับข้อมูลจากส่วนการสื่อสารข้อมูลและนำไปถอดรหัสลับกับกุญแจที่เก็บไว้ที่หน่วยความจำแบบชั่วคราว โดยจะมีขาสัญญาณส่วนที่เชื่อมต่อกับหน่วยประมวลผลของ Elliptic Curve โดยที่ขาสัญญาณ k , Px , Py ,Gx ,Gy เป็นขาสัญญาณที่ใช้ในการส่งข้อมูลเข้าไปประมวลผล ในส่วนของ Elliptic Curve และ จะมีขาสัญญาณ Scalr_Cp,Add_Cp เป็นขาตอบกลับของหน่วยประมวลผล Elliptic Curve ในการแจ้งว่าสร้างกุญแจรหัสและเข้ารหัสลับเสร็จสิ้นตามลำดับ โดยจะส่งข้อมูลกลับมาทางขาสัญญาณ Qx,Qy โดยมีขา Tx_En เป็นขาสัญญาณที่แจ้งให้หน่วยสื่อสารข้อมูลส่งข้อมูลออกไปยังคอมพิวเตอร์



รูปที่ 7.23 แสดงบล็อกไดอะแกรมส่วนของการควบคุม



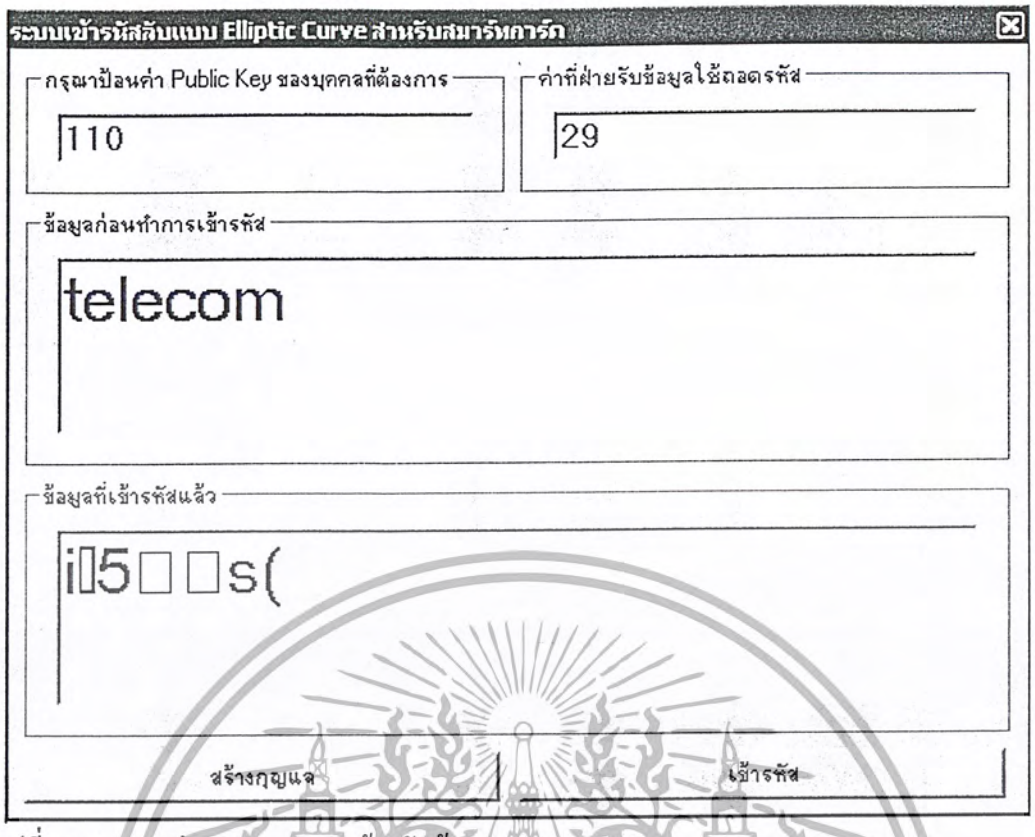
รูปที่ 7.24 แสดง Timming Diagram ของการส่งข้อมูลเพื่อทำการสร้างกุญแจเข้ารหัส



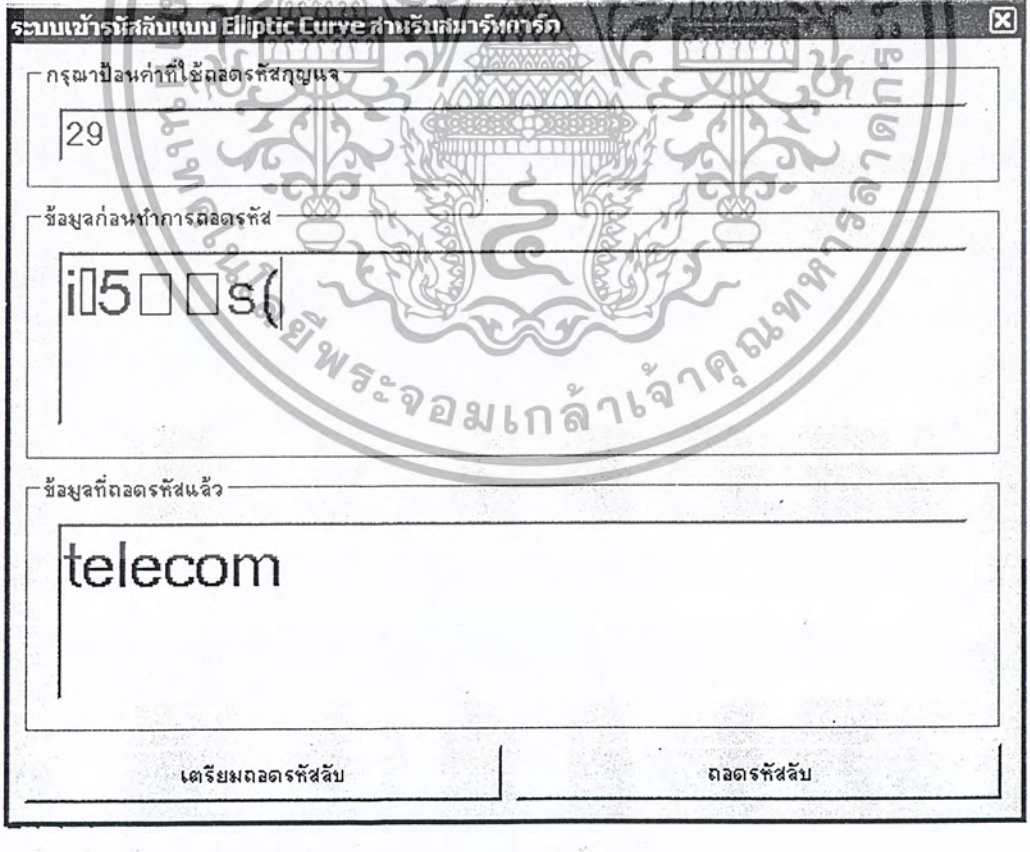
รูปที่ 7.25 แสดง Timming Diagram เมื่อข้อมูลที่ส่งไปสร้างกุญแจรหัสเสร็จสิ้น

7.6 การทดสอบส่วนการทำงานจริงบนอุปกรณ์เอฟพีจีเอ

การทดสอบจะทำการ โปรแกรมวงจรที่ออกแบบลงบนอุปกรณ์เอฟพีจีเอ โดยจะมีโปรแกรมแสดงผลบนคอมพิวเตอร์เพื่อรับข้อมูลในการเข้ารหัสและถอดรหัสลับ ซึ่งจากรูปที่ 7.26 แสดงการทำสอบส่วนสร้างกุญแจเข้ารหัสและส่วนเข้ารหัสข้อมูล โดยการสร้างกุญแจเข้ารหัสจะได้ผลลัพธ์อยู่ 2 ค่า คือ กุญแจเข้ารหัสและค่าที่ฝ่ายรับใช้ในการถอดรหัสเพื่อเอาค่ากุญแจรหัสออกมา จากรูปค่า “110” คือกุญแจสาธารณะของเจ้าของบัตร และ “29” เป็นค่าของ k.P(x,y) โดยทำการเข้ารหัสคำว่า “telecom” ซึ่งได้ผลลัพธ์ออกมาเป็นข้อความ “i□5□□ร(“ ซึ่งเป็นข้อความที่ไม่มีใครเข้าใจได้ จากนั้นจึงทดสอบส่วนการถอดรหัสโดยป้อนค่า “29” ซึ่งใช้ในการถอดรหัสกุญแจออกมา จากนั้นจึงนำข้อความที่เข้ารหัสแล้วไปถอดรหัสผลปรากฏว่าได้ข้อความ “telecom” กลับมาซึ่งได้ผลลัพธ์ถูกต้องทุกประการ



รูปที่ 7.26 แสดงส่วนแสดงผลการเข้ารหัสข้อความ



รูปที่ 7.27 แสดงส่วนแสดงผลการถอดรหัสข้อความ

บทที่ 8
บทสรุปและวิจารณ์

ปัจจุบันมีการใช้งานสมาร์ทการ์ดกันอย่างแพร่หลาย โดยที่ระบบรักษาความปลอดภัยถือว่าเป็นหัวใจสำคัญของสมาร์ทการ์ด แต่ด้วยโครงสร้างของสมาร์ทการ์ดซึ่งมีขนาดเล็กและกำลังในการทำงานต่ำ จึงได้เกิดแนวคิดที่สร้างระบบเข้ารหัสลับที่เหมาะสมสำหรับสมร์ทการ์ด

ในการออกแบบวงจร ได้ทำการออกแบบวงจรด้วยภาษาวีเอชดีแอล ซึ่งเป็นภาษาที่ใช้ในการบรรยายหรืออธิบายรูปแบบการทำงานและความสัมพันธ์ของอุปกรณ์ฮาร์ดแวร์ทั้งในระดับเกทจนถึงระดับดิจิตอลที่ซับซ้อน ทั้งนี้เนื่องจากภาษาวีเอชดีแอลเป็นภาษาที่เหมาะในการนำมาใช้ในการเลียนแบบการทำงานของอุปกรณ์ อีกทั้งยังไม่ขึ้นกับเทคโนโลยี ทำให้ประหยัดเวลาและค่าใช้จ่ายในการออกแบบ จึงได้นำมาใช้กันอย่างกว้างขวางในวงการอุตสาหกรรม รูปแบบของภาษาวีเอชดีแอลประกอบด้วย 2 ส่วนใหญ่ๆ ได้แก่ ส่วนของภาษาวีเควนเชียล(Sequential Language) และภาษาคอนเคอร์เรนท์(Concurrent Language) อีกทั้งการออกแบบสามารถใช้ได้ทั้งสองแบบรวมกัน นอกจากนี้ตัวภาษายังสามารถอธิบายถึงการเชื่อมต่อระหว่างระบบย่อยเข้าด้วยกันเพื่อให้เป็นระบบใหญ่ได้และสามารถกำหนดรูปแบบไวยกรณ์(Syntex) อีกทั้งยังมีการตรวจสอบความหมายของภาษาว่าจะซิมูเลท(Simulate) ได้หรือไม่ เพราะโปรแกรมที่เขียนโดยภาษาวีเอชดีแอลต้องผ่านการซิมูเลทเพื่อตรวจสอบการทำงาน ฉะนั้นในการคอมไพล์(Compile) จะมีการตรวจสอบทั้งวงจรและซิมูเลชั่นซีแมนติก(Semantic) จึงทำให้สะดวกในการใช้งาน

ส่วนของระบบเข้ารหัสลับนั้นสามารถแบ่งการเข้ารหัสข้อมูลออกเป็น 2 แบบคือแบบรหัสลับระบบกุญแจปกปิด และรหัสลับระบบกุญแจสาธารณะ ซึ่งนำไปใช้งานในรูปแบบที่ต่างกัน ดังนั้นในโครงงานนี้จึงได้ใช้ระบบการเข้ารหัสแบบ Elliptic Curve Cryptosystem(ECC) ซึ่งเป็นรหัสลับระบบกุญแจสาธารณะ เนื่องจากความซับซ้อนของคณิตศาสตร์แบบ Elliptic Curve ทำให้ขนาดของกุญแจรหัสที่ใช้ในการเข้ารหัสมีขนาดเล็กเมื่อเทียบรหัสลับระบบกุญแจสาธารณะแบบอื่น โดยที่ระดับความปลอดภัยยังสามารถใช้งานได้ในระดับที่กำหนด อีกทั้งยังทำให้ขนาดของวงจรมีขนาดเล็ก ต้องการกำลังในการคำนวณต่ำและทำงานได้รวดเร็ว ซึ่งมีความเหมาะสมที่จะนำมาใช้สำหรับสมาร์ทการ์ด ซึ่งมีขนาดเล็กและราคาถูก และยังส่งผลให้ลดหน่วยความจำของสมาร์ทการ์ดในการเก็บกุญแจรหัสอีกด้วย

ในการออกแบบและจำลองการทำงาน ได้ทำการออกแบบวงจรคูณขนาด 193 บิต ซึ่งเป็นการคูณแบบคณิตศาสตร์สนามจำกัด โดยได้ทำการทดลองสร้างวงจรคูณขนาด 4 บิตก่อน เพื่อตรวจสอบความถูกต้องของวิธีการออกแบบที่ใช้ว่าเป็นไปตามทฤษฎีหรือไม่ จากนั้นจึงทำการออกแบบวงจรคูณขนาด 193 บิตและทำการทดสอบผลการทำงาน ก่อนที่นำสร้างเป็นระบบเข้ารหัสลับแบบ ECC ซึ่งต้องใช้โอเปอร์เรชั่นหลักๆ คือวงจรคูณ และวงจรอินเวอร์ส ตามอัลกอริทึมของ ECIES ซึ่งมีประสิทธิภาพที่ดี

บรรณานุกรม

1. Doulgas R. Stinson "CRYTOGRAPHY'Theory and Practice' CRC Press,In 200.
2. ECC White Paper "THE ELLIPTIC CURVE CRYPTOSYSTEM FOR SMART CERDS." Certicom ,May 1998,www.certicom.com
3. ECC White Paper. "Introduction to Information Security." Certicom ,March 1997
4. ECC White Paper. "REMARK ON THE SECURETY OF THE ELLIPTIC CURVE CRYPTOSYSTEM" Certicom , July 2000
5. Don Johnson, Alfred Menezes and Scott Vanstone. "The Elliptic Curve Digital Singnature Algorithm (ECDSA)" Certicom.
6. M.Ernst, M Jung, F.Madlener, S.Huss and R. Blumel. "A Reconfigurable System On Chip Implementation for Elliptic Curve Cryptography over GF(2^n)" Integrated Circuits and Systems Lab.,Computer Science Department,Darmstadt University of Technology,Germany
7. YongJe Choi,HuWon Kim and MooSeop Kim. "Implementation of Elliptic Curve Cryptographic Coprocessor over GF(2^{163}) for ECC protocals" Electronics and Telecommunications Institute ,,KOREA.
8. Dan Boneh. "Review of SEC 1: Elliptic Curve Cryptography" Standards for Efficient Cryptography.
9. Certicom Research."SEC 1: Elliptic Curve Cryptography" Standards for Efficient Cryptography.
10. Certicom Research."SEC : Reccommended Elliptic Curve Domian Parameters" Standards for Efficient Cryptography.

ภาคผนวก

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity ECC_Module is
   Port ( clk : in std_logic;
                Scalr_On,Reset,Add_On : in std_logic;
                Scalr_Cp,Add_Cp : out std_logic;
                k : in std_logic_vector(3 downto 0);
             Px,Py,Gx,Gy : in std_logic_vector(3 downto 0);
                Qx,Qy : out std_logic_vector(3 downto 0));
end ECC_Module;
architecture Behavioral of ECC_Module is
type State_Type_EC is
(Idel,Add_Complete,Scalr_1,Scalr_2,Scalr_3,Scalr_4,Scalr_5,Scalr_6,Add_1,Add_2,Add_3,Ad
d_4,Add_5,Add_6,Add_7,Add_8,Add_9,Add_10,Add_11,Add_12,Add_13,Doub_1,Doub_2,Do
ub_3,Doub_4,Doub_5,Doub_6,Doub_7,Doub_8,Doub_9,Doub_10,Doub_11,MUL_1,MUL_2,SQ
_1,INV_1,INV_2,INV_3,INV_4,Ps_Xor2);
signal State_EC,Next_State2,Next_State3,Next_State4 : State_Type_EC;
constant a : std_logic_vector(3 downto 0) := "0011";
constant b : std_logic_vector(3 downto 0) := "0001";
begin
--------------------------------------------------------------------------------
process(clk,Scalr_On)
        variable reg_A,reg_B,reg_C,reg_D,reg_E : std_logic_vector(4 downto 0);
        variable reg_Px,reg_Py,reg_Qx,reg_Qy : std_logic_vector(3 downto 0);
        variable reg_F : std_logic_vector(6 downto 0);
        variable count,count1 : integer range 0 to 6;
begin
if (Reset = '1') then
        Qx <= (others => '0');
        Qy <= (others => '0');
        Scalr_Cp <= '0';
        Add_Cp <= '0';
        State_EC <= Idel;
```

```
case State_EC is
        when Idel =>
                if (Scalr_On = '1') then
                        State_EC <= Scalr_1;
                elsif (Add_On = '1') then
                        reg_Px := Px;
                        reg_Py := Py;
                        reg_Qx := Gx;
                        reg_Qy := Gy;
                        if (reg_Px = reg_Qx and reg_Py = reg_Qy) then
                                Next_State3 <= Add_Complete;
                                State_EC <= Doub_1;
                        else
                                Next_State3 <= Add_Complete;
                                State_EC <= Add_1;
                        end if;
                else
                        State_EC <= Idel;
                        Scalr_Cp <= '0';
                        Add_Cp <= '0';
                end if;
-------------------------------------------------------------------------------
        when Add_Complete =>
                        Qx <= reg_Qx;
                        Qy <= reg_Qy;
                        Add_Cp <= '1';
                        State_EC <= Idel;
        when Scalr_1 =>
                        reg_Qx := (others => '0');
                        reg_Qy := (others => '0');
                        State_EC <= Scalr_2;
                        Count1 := 3;
        when Scalr_2 =>
                if (k(Count1) = '1') then
```

```
                if (Count1 = 0) then
                        reg_Qx := Px;
                        reg_Qy := Py;
                        State_EC <= Next_State4;
                else
                        reg_Qx := Px;
                        reg_Qy := Py;
                        State_EC <= Scalr_3;
                end if;
        else
                Count1 := Count1 - 1;
        end if;
when Scalr_3 =>
        if (Count1 = 0) then
                State_EC <= Idel;
                Qx <= reg_Qx;
                Qy <= reg_Qy;
                Scalr_Cp <= '1';
        else
                Count1 := Count1 - 1;
                reg_Px := reg_Qx;
                reg_Py := reg_Qy;
                Next_State3 <= Scalr_4;
                State_EC <= Doub_1;
        end if;
when Scalr_4 =>
        if (k(Count1) = '1') then
                State_EC <= Scalr_5;
        else
                State_EC <= Scalr_3;
        end if;
when Scalr_5 =>
        reg_Px := Px;
        reg_Py := Py;
        Next_State3 <= Scalr_6;
```

```
                State_EC <= Add_1;
        when Scalr_6 =>
                if (Count1 = 0) then
                        State_EC <= Idel;
                        Scalr_Cp <= '1';
                        Qx <= reg_Qx;
                        Qy <= reg_Qy;
                else
                        Count1 := Count1 - 1;
                        reg_Px := reg_Qx;
                        reg_Py := reg_Qy;
                        Next_State3 <= Scalr_4;
                        State_EC <= Doub_1;
                end if;
-------------------------------------------------------------------------------
        when Add_1 =>
                reg_D(3 downto 0) := reg_Qx;
                reg_E(3 downto 0) := Px;
                Next_State2 <= Add_2;
                State_EC <= Ps_Xor2;
        when Add_2 =>
                reg_Qx := reg_D(3 downto 0);
                reg_D(3 downto 0) := reg_Qy;
                reg_E(3 downto 0) := reg_Py;
                Next_State2 <= Add_3;
                State_EC <= Ps_Xor2;
        when Add_3 =>
                reg_Qy := reg_D(3 downto 0);
                reg_A := '0' & reg_Qx;
                Next_State2 <= Add_4;
                State_EC <= INV_1;
        when Add_4 =>
                reg_A := '0' & reg_C(3 downto 0);
                reg_B := '0' & reg_Qy;
                Next_State2 <= Add_5;
```

```
            State_EC <= MUL_1;
        when Add_5 =>
            reg_Qy := reg_C(3 downto 0);
            reg_D(3 downto 0) := reg_Qx;
            reg_E(3 downto 0) := a;   -------a
            Next_State2 <= Add_6;
            State_EC <= Ps_Xor2;
        when Add_6 =>
            reg_Qx := reg_D(3 downto 0);
            reg_A := '0' & reg_Qy;
            Next_State2 <= Add_7;
            State_EC <= SQ_1;
        when Add_7 =>
            reg_D(3 downto 0) := reg_C(3 downto 0);
            reg_E(3 downto 0) := reg_Qx;
            Next_State2 <= Add_8;
            State_EC <= Ps_Xor2;
        when Add_8 =>
            reg_Qx(3 downto 0) := reg_D(3 downto 0);
            reg_D(3 downto 0) := reg_Qy;
            reg_E(3 downto 0) := reg_Qx;
            Next_State2 <= Add_9;
            State_EC <= Ps_Xor2;
        when Add_9 =>
            reg_Qx(3 downto 0) := reg_D(3 downto 0);    --X3 = reg_Qx
            reg_D(3 downto 0) := reg_Px;
            reg_E(3 downto 0) := reg_Qx;
            Next_State2 <= Add_10;
            State_EC <= Ps_Xor2;
        when Add_10 =>
            reg_Px(3 downto 0) := reg_D(3 downto 0);
            reg_A := '0' & reg_Px;
            reg_B := '0' & reg_Qy;
            Next_State2 <= Add_11;
            State_EC <= MUL_1;
```

```
when Add_11 =>
        reg_Px(3 downto 0) := reg_C(3 downto 0);
        reg_D(3 downto 0) := reg_Px;
        reg_E(3 downto 0) := reg_Qx;
        Next_State2 <= Add_12;
        State_EC <= Ps_Xor2;
when Add_12 =>
        reg_Px(3 downto 0) := reg_D(3 downto 0);
        reg_D(3 downto 0) := reg_Px;
        reg_E(3 downto 0) := reg_Py;
        Next_State2 <= Add_13;
        State_EC <= Ps_Xor2;
when Add_13 =>
        reg_Qy(3 downto 0) := reg_D(3 downto 0);
        State_EC <= Next_State3;
when Doub_1 =>
        reg_A := '0' & reg_Px;
        Next_State2 <= Doub_2;
        State_EC <= SQ_1;
when Doub_2 =>
        reg_Qy := reg_C(3 downto 0);
        reg_A := '0' & reg_Qy;
        Next_State2 <= Doub_3;
        State_EC <= INV_1;
when Doub_3 =>
        reg_A := '0' & reg_C(3 downto 0);
        reg_B(3 downto 0) := b;
        Next_State2 <= Doub_4;
        State_EC <= MUL_1;
when Doub_4 =>
        reg_D(3 downto 0) := reg_C(3 downto 0);
        reg_E(3 downto 0) := reg_Qy;
        Next_State2 <= Doub_5;
        State_EC <= Ps_Xor2;
when Doub_5 =>
```

```
            reg_Qx := reg_D(3 downto 0);
            reg_A := '0' & reg_Px;
            Next_State2 <= Doub_6;
            State_EC <= INV_1;
        when Doub_6 =>
            reg_A := '0' & reg_C(3 downto 0);
            reg_B := '0' & reg_Py;
            Next_State2 <= Doub_7;
            State_EC <= MUL_1;
        when Doub_7 =>
            reg_D(3 downto 0) := reg_Px;
            reg_E(3 downto 0) := reg_C(3 downto 0);
            Next_State2 <= Doub_8;
            State_EC <= Ps_Xor2;
        when Doub_8 =>
            reg_Px := reg_D(3 downto 0);
            reg_A := '0' & reg_Px;
            reg_B := '0' & reg_Qx;
            Next_State2 <= Doub_9;
            State_EC <= MUL_1;
        when Doub_9 =>
            reg_Px(3 downto 0) := reg_C(3 downto 0);
            reg_D(3 downto 0) := reg_Px;
            reg_E(3 downto 0) := reg_Qx;
            Next_State2 <= Doub_10;
            State_EC <= Ps_Xor2;
        when Doub_10 =>
            reg_Px(3 downto 0) := reg_D(3 downto 0);
            reg_D(3 downto 0) := reg_Px;
            reg_E(3 downto 0) := reg_Qy;
            Next_State2 <= Doub_11;
            State_EC <= Ps_Xor2;
        when Doub_11 =>
            reg_Qy(3 downto 0) := reg_D(3 downto 0);
            State_EC <= Next_State3;
```

```
when Mul_1 =>
        reg_C := (others => '0');
        count := 3;
        State_EC <= Mul_2;
when MUL_2 =>
        if (Count < 4) then
                reg_C := reg_C(3 downto 0 ) & '0';
                if (reg_B(Count) = '1') then
                        reg_C := reg_C xor reg_A;
                end if;
                if (reg_C(4) = '1') then
                        reg_C(4) := reg_C(4) xor '1';
                        reg_C(1) := reg_C(1) xor '1';
                        reg_C(0) := reg_C(0) xor '1';
                end if;
                if (Count = 0) then
                        State_EC <= Next_State2;
                else
                        Count := Count - 1;
                end if;
        end if;
when SQ_1 =>
        reg_C := (others => '0');insert '0'
        reg_F := reg_A(3) & '0' & reg_A(2) & '0' & reg_A(1) & '0' & reg_A(0);
        reg_C(3 downto 0) := reg_F(3 downto 0) xor '0' & reg_F(6 downto 4) xor
reg_F(6 downto 4) & '0';
        State_EC <= Next_State2;

when INV_1 =>
        reg_B := (others => '0');
        reg_B(4) := '1';
        reg_B(1) := '1';
        reg_B(0) := '1';
        reg_D := (others => '0');
```

```
                reg_C := "00001";
                if (reg_A = "00000") then
                        State_EC <= Next_State2;
                        reg_C := "00000";
                else
                        State_EC <= INV_2;
                end if;
        when INV_2 =>
                if (reg_A(0) = '0') then
                        reg_A := '0' & reg_A(4 downto 1);
                        if (reg_C(0) = '1') then
                                reg_C(4) := reg_C(4) xor '1';
                                reg_C(1) := reg_C(1) xor '1';
                                reg_C(0) := reg_C(0) xor '1';
                        end if;
                        reg_C := '0' & reg_C(4 downto 1);
                else
                        State_EC <= INV_3;
                end if;
        when INV_3 =>
                if (reg_A = "00001") then
                        State_EC <= Next_State2;
                else
                        State_EC <= INV_4;
                end if;
        when INV_4 =>
                if (reg_B > reg_A) then
                        reg_E := reg_B;
                        reg_B := reg_A;
                        reg_A := reg_E;
                        reg_E := reg_D;
                        reg_D := reg_C;
                reg_C := reg_E;
                end if;
                reg_A := reg_A xor reg_B;
```

```
                reg_C := reg_C xor reg_D;
                State_EC <= INV_2;
            when Ps_Xor2 =>
                reg_D := reg_D xor reg_E;
                State_EC <= Next_State2;
            when Others =>
                State_EC <= Idel;
                Scalr_Cp <= '0';
                Add_Cp <= '0';
                Count := 0;
        end case;
end if;
end process;
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Control_Module is
    Port ( Rom_Addr,Ram_Addr : out integer range 0 to 1;
        Ram_Wr,Ram_Rd : out std_logic;
        Pre_Encrypt,Pre_Decrypt,Encrypt,Decrypt,Ack_Tx,Add_Cp,Scalr_Cp,Reset,clk,Rx_En : in
std_logic;
            Add_On,Scalr_On,Tx_En : out std_logic;
        Qx,Qy,Seed_In : in std_logic_vector(3 downto 0);
            Px,Py,Gx,Gy,k : out std_logic_vector(3 downto 0);
        UART_Data_In,Ram_Data_In,Rom_Data : in std_logic_vector(7 downto 0);
        UART_Data_Out,Ram_Data_Out,Show_7_Segment : out std_logic_vector(7 downto 0));
end Control_Module;
architecture Behavioral of Control_Module is
type State_Type_Control is
(Idel,Pre_En_1,Pre_En_2,Pre_En_3,Pre_En_4,Pre_En_5,Pre_En_6,Pre_En_7,En_1,En_2,En_3,Pre_De_
1,Pre_De_2,De_1,De_2,De_3);
signal State_Control : State_Type_Control := Idel;
begin
process(clk)
        variable Buffer_Data,Key_Data : std_logic_vector(7 downto 0);
```

```
        variable reg_Qx,reg_Qy : std_logic_vector(3 downto 0);
        variable RNG : std_logic_vector(3 downto 0);
begin
if (Reset = '1') then
        Ram_Wr <= '1';
        Ram_Rd <= '1';
        Add_On <= '0';
        Scalr_On <= '0';
        Tx_En <= '0';
        Show_7_Segment <= "00000000";
        State_Control <= Idel;
elsif (clk'Event and clk = '1') then
        case State_Control is
                when Idel =>
                        if (Rx_En = '1') then
                                if (Pre_Encrypt = '1') then
                                        Buffer_Data := UART_Data_In;   --Receive Public Key
                                        State_Control <= Pre_En_1;
                                elsif (Encrypt = '1') then
                                        Buffer_Data := UART_Data_In;
                                        Ram_Rd <= '0';
                                        Ram_Addr <= 0;
                                        State_Control <= En_1;
                                elsif (Pre_Decrypt = '1') then
                                        Buffer_Data := UART_Data_In;   --Receive kP
                                        Rom_Addr <= 1;
                                        State_Control <= Pre_De_1;
                                elsif (Decrypt = '1') then
                                        Buffer_Data := UART_Data_In;
                                        Ram_Rd <= '0';
                                        Ram_Addr <= 0;
                                        State_Control <= De_1;
                                end if;
                        else
                                State_Control <= Idel;
```

```
            end if;
        when Pre_En_1 =>
            if (Rx_En = '0') then
                RNG := Seed_In; --Random Number Generator
                k <= RNG;
                Px <= Buffer_Data(7 downto 4);
                Py <= Buffer_Data(3 downto 0);
                Scalr_On <= '1';  --Compute kP
                State_Control <= Pre_En_2;
            else
                State_Control <= Pre_En_1;
            end if;
        when Pre_En_2 =>
            Scalr_On <= '0';
            if (Scalr_Cp = '1') then      --Wait ECC Module Processed
                Ram_Wr <= '0';
                Ram_Addr <= 0;   --Storage k.aP at Ram Address '0'
                reg_Qx := Qx;
                reg_Qy := Qy;
                Ram_Data_Out <= reg_Qx & reg_Qy;
                State_Control <= Pre_En_3;
            else
                State_Control <= Pre_En_2;
            end if;
        when Pre_En_3 =>
            Ram_Wr <= '0';
            Ram_Addr <= 1;   --Ram Address 1 Storage k
            Ram_Data_Out <= "0000" & RNG;
            State_Control <= Pre_En_4;
        when Pre_En_4 =>
            Rom_Addr <= 0;   --Base Point
            State_Control <= Pre_En_5;
        when Pre_En_5 =>
            Buffer_Data := Rom_Data;
            k <= RNG;
```

```
            Px <= Buffer_Data(7 downto 4);
            Py <= Buffer_Data(3 downto 0);
            Scalr_On <= '1';
            State_Control <= Pre_En_6;
        when Pre_En_6 =>
            Scalr_On <= '0';
            if (Scalr_Cp = '1') then
                Tx_En <= '1';
                reg_Qx := Qx;
                reg_Qy := Qy;
                UART_Data_Out <= reg_Qx & reg_Qy;
                State_Control <= Pre_En_7;
            else
                State_Control <= Pre_En_6;
            end if;
        when Pre_En_7 =>
            if (Ack_Tx = '1') then
                Tx_En <= '0';
                Show_7_Segment <= "00000001";
                State_Control <= Idel;
            else
                State_Control <= Pre_En_7;
            end if;
        when En_1 =>
            if (Rx_En = '0') then
                Key_Data := Ram_Data_In;
                Px <= Buffer_Data(7 downto 4);
                Py <= Buffer_Data(3 downto 0);
                Gx <= Key_Data(7 downto 4);
                Gy <= Key_Data(3 downto 0);
                Add_On <= '1';  --Compute kP
                State_Control <= En_2;
            else
                State_Control <= En_1;
            end if;
```

```
when En_2 =>
        Add_On <= '0';
        if (Add_Cp = '1') then        --Wait ECC Module Processed
                Tx_En <= '1';
                Ram_Rd <= '1';
                reg_Qx := Qx;
                reg_Qy := Qy;
                UART_Data_Out <= reg_Qx & reg_Qy;
                State_Control <= En_3;
        else
                State_Control <= En_2;
        end if;
when En_3 =>
        if (Ack_Tx = '1') then
                Tx_En <= '0';
                Show_7_Segment <= "00000011";
                State_Control <= Idel;
        else
                State_Control <= En_3;
        end if;
-------------------------------------------------------------------------------
when Pre_De_1 =>
        if (Rx_En = '0') then
                Key_Data := Rom_Data;
                k <= Key_Data(3 downto 0);
                Px <= Buffer_Data(7 downto 4);
                Py <= Buffer_Data(3 downto 0);
                Scalr_On <= '1';  --Compute kP
                State_Control <= Pre_De_2;
        else
                State_Control <= Pre_De_1;
        end if;
when Pre_De_2 =>
        Scalr_On <= '0';
```

```
            if (Scalr_Cp = '1') then
                    Show_7_Segment <= "00000111";
                    reg_Qx := Qx;
                    reg_Qy := Qy;
                    Ram_Data_Out <= reg_Qx & reg_Qy;
                    Ram_Wr <= '0';
                    Ram_Addr <= 0;
                    State_Control <= Idel;
            else
                    State_Control <= Pre_De_2;
            end if;
-------------------------------------------------------------------------------
        when De_1 =>
            if (Rx_En = '0') then
                    Key_Data := Ram_Data_In;
                    Ram_Rd <= '1';
                    Px <= Buffer_Data(7 downto 4);
                    Py <= Buffer_Data(3 downto 0);
                    Gx <= Key_Data(7 downto 4);
                    Gy <= Key_Data(3 downto 0);
                    Add_On <= '1'; --Compute kP
                    State_Control <= De_2;
            else
                    State_Control <= De_1;
            end if;
        when De_2 =>
                Add_On <= '0';
                if (Add_Cp = '1') then        --Wait ECC Module Processed
                        Tx_En <= '1';
                        reg_Qx := Qx;
                        reg_Qy := Qy;
                        UART_Data_Out <= reg_Qx & reg_Qy;
                        State_Control <= De_3;
                else
                        State_Control <= De_2;
```

```
                end if;
            when De_3 =>
                if (Ack_Tx = '1') then
                    Tx_En <= '0';
                    Show_7_Segment <= "00001111";

                    State_Control <= Idel;
                else
                    State_Control <= De_3;
                end if;
            when others =>
                State_Control <= Idel;
        end case;
end if;
end process;
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Ram_Module is
    port(   Wr,Rd : in std_logic;
            Addr : in integer range 0 to 1;
            Data_In : in std_logic_vector(7 downto 0);
            Data_Out : out std_logic_vector(7 downto 0));
end Ram_Module;
architecture Behavioral of Ram_Module is
begin
process(Addr,Wr,Rd)
    type ram_array is array (0 to 1) of std_logic_vector(7 downto 0);
    variable mem : ram_array;
begin
    if (Wr = '0') then
        mem(Addr) := Data_In;
    elsif (Rd = '0')  then
        Data_Out <= mem(Addr);
```

```
        end if;
end process;
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Rom_key_A is
    Port ( Addr : in integer range 0 to 1;
        Data : out std_logic_vector(7 downto 0));
end Rom_key_A;

architecture Behavioral of Rom_key_A is

begin
process(Addr)

type rom_array is array (0 to 1) of std_logic_vector (7 downto 0);
constant rom : rom_array := (
"11000101",-- Base Point
"00001001");--Private key of Alice
begin
        Data <= rom(Addr);
end process;
end Behavioral;
```

```
use ieee.std_logic_1164.all;
entity UART_Module 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,Ack_Tx : out std_logic;
        DATA_RX : out std_logic_vector(7 downto 0);
        RX_EN : out std_logic
            );
```

```
end UART_Module;
architecture rtl of UART_Module 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;
                                                Ack_Tx <= '1';
                                                State_TX <= Start;
                                        else
                                                TX_Data_Count := 0;
                                                Ack_Tx <= '0';
                                                State_TX <= Idel;
                                        end if;
                                when Start =>
                                        Ack_Tx <= '0';
                                        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';
                Ack_Tx <= '0';

                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 rtl;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RNG_Module is
    Port ( clk,Reset : in std_logic;
           Seed_Out : out std_logic_vector(3 downto 0));
end RNG_Module;

architecture Behavioral of RNG_Module is

begin
process(clk)
variable rng_valid : std_logic_vector(3 downto 0);
```

```
variable r : std_logic_vector(4 downto 0);

begin
        if (clk'event and clk = '1') then
                if (Reset = '1') then
                        r := "11111";
                        Seed_Out <= "0010";
                else
                        r(0) := r(1) xor r(4);
                        r(4 downto 1) := r(3 downto 0);
                        mg_valid := r(3 downto 0);
                        if (mg_valid = "0001" or mg_valid = "1111" or mg_valid = "0000") then
                                Seed_Out <= "1000" xor mg_valid;
                        else
                                Seed_Out <= '0' & mg_valid(2 downto 0);
                        end if;
                end if;
        end if;
end process;
end Behavioral;
```

```
library IEEE;
LIBRARY IEEE;
use ieee.std_logic_1164.all;
entity One_Pulse is
        port(   Clk : in std_logic;
                SW : in std_logic;
                SW_Single_Pulse : out std_logic);
end One_Pulse;
architecture Behavioral of One_Pulse is
        signal sw_debounce_delay : std_logic;
        signal power_on : std_logic;
begin
process(Clk)
begin
if (Clk'Event) and (Clk = '1') then
```

```
        if power_on = '0' then
                sw_single_pulse <= '0';
                sw_debounce_delay <= '1';
                power_on <= '1';
        else
                if (sw = '1') and (sw_debounce_delay = '0') then
                        sw_single_pulse <='1';
                else
                        sw_single_pulse <='0';
                end if;
                sw_debounce_delay <= sw;
        end if;
end if;
end process;
end Behavioral;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity Show_7_Segment is
port(data_in : in std_logic_vector(7 downto 0);
        data_out : out std_logic_vector(7 downto 0) );
end Show_7_Segment;
architecture Behavioral of Show_7_Segment is
begin
process(Data_in)
begin
        data_out <= data_in;
end process;
end Behavioral;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity CLk_Div is
        port(   Clk_in : in std_logic;
                        Clk_out : out std_logic );
end CLk_Div;
architecture Behavioral of CLk_Div is
```

```
begin
process(Clk_in)
        variable Clk_temp : std_logic := '0';
        variable count : integer range 0 to 312 := 0;
begin
if Clk_in'Event and Clk_in = '1' then
        if count < 312 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 Behavioral;
```