การพัฒนา SMART CARD โดยการเข้ารหัสแบบ ELLIPTIC CURVE และ ELGAMAL CRYPTOGRAPHY

DEVELOPMENT OF SMART CARD WITH ELLIPTIC CURVE AND ELGAMAL CRYPTOGRAPHY

สุรพงษ์ พงษ์ยุพินพานิช

SURAPONG PONGYUPINPANICH

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2545

ISBN 974-324-085-3

DEVELOPMENT OF SMART CARD WITH ELLIPTIC CURVE AND ELGAMAL CRYPTOGRAPHY

SURAPONG PONGYUPINPANICH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF ENGINEERING IN

ELECTRICAL ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2002

ISBN 974-324-085-3

หัวข้อวิทยานิพนธ์	การพัฒนา SMART CARD โดยการเข้ารหัสแบบ ELLIPTIC CURVE และ ELGAMAL CRYPTOGRAPHY
นักศึกษา	นายสุรพงษ์ พงษ์ยุพินพานิช
รหัสประจำตัว	42061074
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขา	วิศวกรรมไฟฟ้า
พ.ศ.	2545
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.สมศักดิ์ ชุมช่วย

บทคัดย่อ

วิทยานิพนธ์นี้ เสนอวิธีการเข้ารหัสข้อมูลแบบ ElGamal Elliptic Curve ขนาด 176 บิต ซึ่งเป็นการนำวิธีการเข้ารหัสข้อมูลแบบ ElGamal และระบบคณิตศาสตร์แบบ Elliptic Curve มาใช้งานร่วมกันเพื่อจำลองแบบสมาร์ทการ์ดที่พัฒนาบน FPGA เนื่องจากระบบคณิตศาสตร์แบบ Elliptic Curve นี้ ทำให้สามารถลดขนาดของ Public-Key โดยที่ระดับความปลอดภัยอยู่ในระดับที่สามารถใช้งานได้ เมื่อเทียบกับวิธีการของ RSA และ EDL ทั้งยังทำให้ขนาดของข้อมูลที่เข้ารหัสแล้วมีขนาดเป็น2 เท่า เมื่อเทียบกับวิธีการเข้ารหัสข้อมูลแบบ ElGamal Discrete Logarithm จึงทำให้ความจุข้อมูลบนสมาร์ทการ์ดเพิ่มมากขึ้น

Thesis Title	DEVELOPMENT OF SMART CARD WITH ELLIPTIC CURVE AND ELGAMAL CRYPTOGRAPHY
Student	Mr. Surapong Pongyupinpanich
Student ID.	42061074
Degree	Master of Engineering
Programme	Electrical Engineering
Year	2001
Thesis Advisor	Assoc.Prof.Dr.Somsak Choomchuay

ABSTRACT

This thesis proposes the implementation of a smart card with data encryption / decryption aspects. Data stored in the memory(flash ROM) is primarily encrypted/decrypted with and on-chip encrypter/decrypter. Encryption and decryption schemes are based on ElGamal Elliptic Curve Cryptography with the block length of 176 bits. This method is proved to be superior compared to RSA and ElGamal Discrete Algorithm in term of key size. The prototype was implemented using FPGA. The preliminary investigation confirmed that the designed is fully function as planed.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้เป็นอย่างดีด้วยคำแนะนำและให้คำปรึกษาอย่างดีของ รศ. ดร.สมศักดิ์ ชุมช่วย ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้ทำวิจัยรู้สึกซาบซึ้งในความกรุณาและขอกราบขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณคุณพ่อคุณแม่เป็นอย่างยิ่ง ที่ให้ทุกสิ่งทุกอย่างเพื่อการศึกษาและทำงานวิจัยด้วยดีที่สุด ขอขอบคุณสถาบันวิจัยอิเล็กทรอนิกส์แห่งชาติ และภาควิชาอิเล็กทรอนิกส์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ซึ่งเอื้อเฟื้ออุปกรณ์สำหรับทดลอง และขอบพระคุณเพื่อนๆ พี่ๆ และน้องๆ ในห้องวิจัยไมโครอิเล็กทรอนิกส์ทุกท่าน ที่มีส่วนร่วมในงานวิจัยและวิทยานิพนธ์ฉบับนี้จนสำเร็จลุล่วงไปได้ด้วยดี

สุรพงษ์ พงษ์ยุพินพานิช

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

โดยทั่วไปสมาร์ทการ์ดถูกนำไปใช้งานในรูปแบบธุรกิจทางการเงินเช่น ธุรกิจธนาคาร ต่อจากนั้น ก็เริ่มใช้ในภาคธุรกิจอุตสาหกรรม ธุรกิจข้อมูลเฉพาะบุคคล ธุรกิจประกันภัย และอื่นๆ เนื่องจากการใช้งานที่ง่ายและสะดวก มีส่วนช่วยให้สมาร์ทการ์ดเป็นที่นิยมและแพร่หลายอย่างรวดเร็ว โดยเริ่มแรกขนาดของความจุของหน่วยความจำบนสมาร์ทการ์ดมีขนาดเล็ก ทำให้การเก็บข้อมูลภายในตัวของสมาร์ทการ์ดนั้นสามารถเก็บได้น้อย ต่อมาด้วยเทคโนโลยี VLSI ทำให้สามารถสร้างหน่วยความจำที่มีขนาดความจุเพิ่มมากขึ้น แต่เนื่องจากขนาดของ Public-Key ที่ใช้งานมีขนาดใหญ่ จึงทำให้สิ้นเปลืองหน่วยความจำเพื่อใช้สำหรับเก็บค่าของ Public-Key นี้ ดังนั้นจึงมีความต้องการที่จะลดขนาดของ Public-Key ของสมาร์ทการ์ดให้มีขนาดเล็กลงเพื่อที่จะทำให้มีพื้นที่สำหรับเก็บข้อมูลเพิ่มมากขึ้น ทั้งยังทำให้ขนาดของวงจรที่ออกแบบมีขนาดเล็ก และสามารถใช้งานได้อย่างปลอดภัยตามมาตราฐานที่กำหนด ดังนั้นจึงเกิดแนวความคิดในประยุกต์ใช้งานอัลกอริธึมของ ElGamal และ Elliptic Curve Discrete Logarithm ร่วมกัน

ปกติการรักษาความปลอดภัยของข้อมูลนั้นจะต้องเข้ารหัสข้อมูลก่อนที่จะทำการเก็บลงในหน่วยความจำ ในส่วนของสมาร์ทการ์ดนั้นได้เก็บ Public-Key ซึ่งใช้สำหรับเข้ารหัส (Encryption) และถอดรหัส (Decryption) เอาไว้ในหน่วยความจำภายในสมาร์ทการ์ด โดย Public นั้นเป็นส่วนที่ผู้ส่งใช้สำหรับเข้ารหัส (Encryption) ข้อมูลและข้อมูลถูกส่งไปยังผู้รับ และผู้รับทำการถอดรหัส (Decryption) ด้วย Key จึงทำให้ทราบข้อมูลที่ผู้ส่งส่งมานั่นเอง

วิทยานิพนธ์นี้นั้นได้ทำการจำลองการทำงานของสมาร์ทการ์ด โดยใช้หลักการของคณิตศาสตร์แบบ Elliptic Curve Discrete Logarithm ร่วมกันกับอัลกอริธึมของ ElGamal Public – Key Cryptography บนคณิตศาสตร์สนามจำกัด(Finite Field) โดยในส่วนของการออกแบบตัวประมวลผลข้อมูลนั้นทำการออกแบบด้วยภาษา VHDL และใช้คอมพิวเตอร์ช่วยในการดูสัญญาณเวลาพร้อมทั้งทำการสังเคราะห์เป็นวงจรระดับเกต หลังจากนั้นจึงนำมาสร้างเป็นฮาร์ดแวร์ด้วยเทคโนโลยี VLSI เพื่อจำลองการทำงานของสมาร์ทการ์ดต่อไป

1.2 วัตถุประสงค์

1) เพื่อศึกษาการทำงานของการเข้าและถอดรหัสข้อมูล ในส่วนที่เป็น Public-Key ที่นำมาจำลองการทำงานสมาร์ทการ์ด

2) พัฒนาวิธีการเข้าและถอดรหัสที่เหมาะสม พร้อมทั้งศึกษาแนวทางการพัฒนาขั้นตอนเพื่อเป็นโครงสร้างทางฮาร์ดแวร์
3) เพื่อศึกษาการออกแบบฮาร์ดแวร์ด้วยภาษา VHDL และศึกษาถึงวิธีการตรวจสอบสัญญาณและทำการสังเคราะห์เป็นวงจรระดับเกต
4) เสนอแนะต้นแบบที่จำลองการทำงานของสมาร์ทการ์ดที่พัฒนาขึ้น

1.3 ทฤษฎีหรือแนวคิด

เนื่องจากสมาร์ทการ์ดที่ใช้งานอยู่ในปัจจุบันเป็นลักษณะของการ์ดหน่วยความจำ ซึ่งต้องทำการเข้ารหัสข้อมูลจากคอมพิวเตอร์แล้วจึงเก็บข้อมูลลงในหน่วยความจำและพร้อมกันนั้นขนาดของ Public-Key ที่ใช้งานมีขนาดใหญ่ จึงทำให้หน่วยความจำที่ใช้เก็บ Public-Key มีขนาดใหญ่

งานวิจัยนี้ ได้จำลองการทำงานของสมาร์ทการ์ดให้มีฟังก์ชั่นในการเข้าและถอดรหัสอยู่ภายใน พร้อมกันนั้นได้ทำการลดขนาดของ Public และ Private Key ให้มีขนาดเล็กลง ซึ่งเดิมใช้งานที่ 512-1024 บิตลดลงเหลือเพียง 155-201 บิต โดยที่ความปลอดภัยของระบบเท่าเดิม

1.4 ขอบเขตของวิทยานิพนธ์

งานวิจัยนี้ต้องการพัฒนาและจำลองการทำงานของสมาร์ทการ์ด โดยใช้หลักการเข้ารหัสข้อมูลตามอัลกอริธึม ElGamal Elliptic Curve Public-Key Cryptography ซึ่งภายในประกอบด้วยหน่วยความจำ ฟังก์ชันการเข้ารหัสและถอดข้อมูลอยู่ภายใน จึงทำให้สามารถเข้ารหัสและถอดรหัสข้อมูลบนชิพ FPGA (Field Programmable Gate Array) ได้โดยตรง

1.5 เนื้อหาภายในวิทยานิพนธ์

เนื้อหาวิทยานิพนธ์จะอธิบายถึงขั้นตอนและรายละเอียดต่างๆในการทดลอง โดยแบ่งออกเป็นบทต่างๆ ได้ดังนี้

บทที่ 2 อธิบายถึงการประวัติความเป็นมาและโครงสร้างการทำงานของสมาร์ทการ์ด เบื้องต้น

บทที่3 อธิบายถึงหลักการคณิตศาสตร์สนามจำกัด (Finite Field) และโอเปเรชั่นของการกระทำในคณิตศาสตร์ดังกล่าวที่ใช้สำหรับออกแบบ Public – Key Cryptography และ Elliptic Curve

บทที่ 4 อธิบายถึงการทำงานและการออกแบบฮาร์แวร์ด้วยภาษา VHDL ซึ่งประกอบด้วยความสามารถของภาษา VHDL หลักการการสร้างโมเดล โครงสร้างทางภาษา และอื่นๆ

บทที่ 5 หลักการของ Public-Key Cryptography โดยอธิบายถึงประวัติความเป็นมาของการ Cryptography ซึ่งประกอบด้วยอังกอริธึมที่ใช้งานในแบบต่างๆ

บทที่ 6 สถาปัตยกรรมและการออกแบบสมาร์ทการ์ด

บทที่ 7 อธิบายถึงการทดสอบวงจรในส่วนต่างๆ ทั้งการเข้ารหัสและการถอดรหัสข้อมูลบน FPGA พร้อมทั้งรวมแต่ละส่วนเข้าด้วยกันและแสดงผลการทดสอบ

บทที่ 8 สรุปผลการวิจัยปัญหาที่พบและข้อเสนอแนะ

บทที่ 2
สมาร์ทการ์ด

บทนำ

ในบทนี้ได้กล่าวถึงประวัติความเป็นมาของสมาร์ทการ์ด องค์ประกอบภายใน มาตราฐานการใช้งาน ชนิดของสมาร์ทการ์ดที่มีการใช้งานอยู่ในปัจจุบัน อนาคตของสมาร์ทการ์ด การใช้งานสมาร์ทการ์ด และแนวคิดในการพัฒนาสามาร์ทการ์ด ซึ่งทำให้ทราบภาพรวมของสมาร์ทการ์ด เพื่อเป็นแนวทางสำหรับศึกษาในบทต่อๆ ไป

2.1 ประวัติความเป็นมาของสมาร์ทการ์ด

การใช้งานสมาร์ทการ์ด นั้นเกิดจากการใช้งานบัตรซึ่งอยู่ในรูปของบัตรพลาสติกซึ่ง เรียกว่า พลาสติกการ์ด (Plastic Card) ซึ่งเริ่มใช้ในปี ค.ศ. 1950 โดยบริษัท Diners Club ซึ่งสมาชิกผู้ถือบัตรสามารถใช้บริการตาม ภัตตาคาร ร้านอาหาร หรือโรงแรมที่ซึ่งบริษัทนี้เป็นสมาชิกอยู่ ต่อมาในปี ค.ศ. 1951 บริษัท American Express ก็ได้นำบัตรนี้มาให้บริการแก่สมาชิกของบริษัท ซึ่งการใช้งานของทั้งสองบริษัทนั่นเป็นลักษณะของการให้สิทธิพิเศษกับสมาชิก แต่ที่จริงแล้วการใช้บัตรนี้เป็นบัตรเครดิตนั่นได้เริ่มใช้งานตั้งแต่ปี ค.ศ. 1940 ในธนาคารของสหรัฐอเมริกา ซึ่งจะให้สิทธิพิเศษแก่ลูกค้าของธนาคาร โดยที่ลูกค้าของธนาคารสามารถติดต่อกับธนาคารสาขาต่างๆ ในสหรัฐอเมริกาได้สะดวกและรวดเร็ว จึงเรียกการทำธุรกิจโดยใช้บัตรแบบนี้ว่า VISA ต่อมาในปี ค.ศ. 1966 ก็ได้มีการใช้งานบัตรนี้ในประเทศอังกฤษ โดยใช้งานร่วมกับบัตร VISA ในสหรัฐอเมริกาและปี ค.ศ. 1972 ธนาคารยักษ์ใหญ่ 4 ธนาคารคือ ธนาคาร Lloyds ธนาคาร National Westminster ธนาคาร Midland และ ธนาคาร Royal ก็ได้ร่วมมือกันเพื่อที่จะใช้บัตรนี้ทำการติดต่อธุรกิจทางการเงิน โดยใช้ชื่อของ Master card ต่อมาเพื่อเพิ่มความสะดวกสบายให้กับลูกค้าในการรับบริการถอนเงินด่วน จึงมีการใช้บัตรนี้กับตู้บริการเงินด่วน คือ Automatic Teller Machine (ATM) ซึ่งเป็นที่แพร่หลายในเวลาต่อมา

การใช้งานสมาร์ทการ์ด นั้นเริ่มใช้งานในปี ค.ศ. 1970 ที่ประเทศฝรั่งเศษ โดยเกิดจากแนวคิดที่ต้องการใช้บัตรแทนจำนวนเงินซึ่งเรียกว่า Chip – Card หลังจากนั้นก็ได้มีการพัฒนาให้มีการทำงานร่วมกับวงจรอิเลคทรอนิกส์ซึ่งเรียกว่า Integrated Circuit Card (ICC) และในทุกวันนี้มีการ ใช้งานสมาร์ทการ์ด กันอย่างแพร่หลายในวงสังคม ผลจากการใช้บัตรในการทำธุรกรรมต่างๆนั้นทำให้ชีวิตสะดวกสบายมากขึ้น รวดเร็วขึ้น ซึ่งเราสามารถนำเอาสมาร์ทการ์ด ไปใช้งานต่างๆ ได้ดังเช่น ธุรกิจทางการเงิน ข้อมูลทางสุขภาพ ข้อมูลเฉพาะบุคคล การรักษาความปลอดภัย ธุรกิจการสื่อสาร

บัตรเอทีเอ็มสำหรับเบิกเงินสด บัตรชมภาพยนต์ บัตรสำหรับซื้อตั๋วรถไฟฟ้า รถโดยสารประจำทาง และอื่นๆ อีกมากมาย ทุกวันนี้บัตรสมาร์ทการ์ดได้ถูกพัฒนามากขึ้น แต่จุดหนึ่งที่มีความสำคัญและหน้าสนใจมากที่สุดสำหรับการใช้งานบัตรสมาร์ทการ์ดคือ ทางด้านความปลอดภัยของการใช้งานทั้งระบบ ซึ่งถึงแม้ว่าจะมีการพัฒนาไปมากแล้วก็ตามแต่ก็ยังป้องกันได้ไม่เต็ม 100 เปอร์เซ็นต์ แต่ที่แน่นอนคือดีกว่าบัตรพลาสติกแถบแม่เหล็กที่ใช้กันแพร่หลายในปัจจุบันอย่างแน่นอน ซึ่งอาจกล่าวได้ว่าสมาร์ทการ์ด นั้นกำลังจะเข้ามามีส่วนสำคัญเกี่ยวกับการดำเนินชีวิตประจำวันนั่นเอง

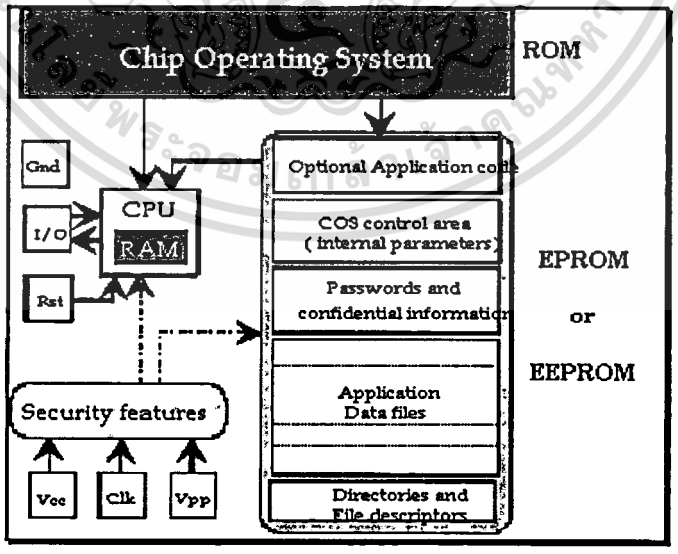
2.2 องค์ประกอบภายในสมาร์ทการ์ด

สมาร์ทการ์ด คือ อุปกรณ์เก็บข้อมูลแบบพกพาซึ่งสามารถแสดงข้อกำหนดและคุณสมบัติเฉพาะตัว พร้อมทั้งระดับความปลอดภัยได้ ซึ่งสมาร์ทการ์ด นั้นเกิดขึ้นจากการใช้งานเทคโนโลยีไมโครอิเล็กทรอนิกส์ ทำการออกแบบตัวประมวลผลสำหรับการใช้งานดังกล่าวและเนื่องจากการพัฒนาเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์ นี้ทำให้สมาร์ทการ์ด มีความจุของข้อมูลเพิ่มมากขึ้น ความเร็วในการประมวลผลสูงขึ้นนั่นเอง

2.2.1 โครงสร้างพื้นฐานของสมาร์ทการ์ด

สมาร์ทการ์ด สามารถแบ่งโครงสร้างออกเป็น 2 แบบใหญ่ๆด้วยกันคือ

- ไมโครโปรเซสเซอร์
- หน่วยความจำ



Microprocessor Chip Diagram, courtesy of Gemplus

รูปที่ 2.1 แสดงโครงสร้างพื้นฐานของสมาร์ทการ์ด

2.2.1.1 ไมโครโปรเซสเซอร์

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

ตารางที่ 2.1 แสดงสถาปัตกรรมภายในสมาร์ทการ์ด [21]

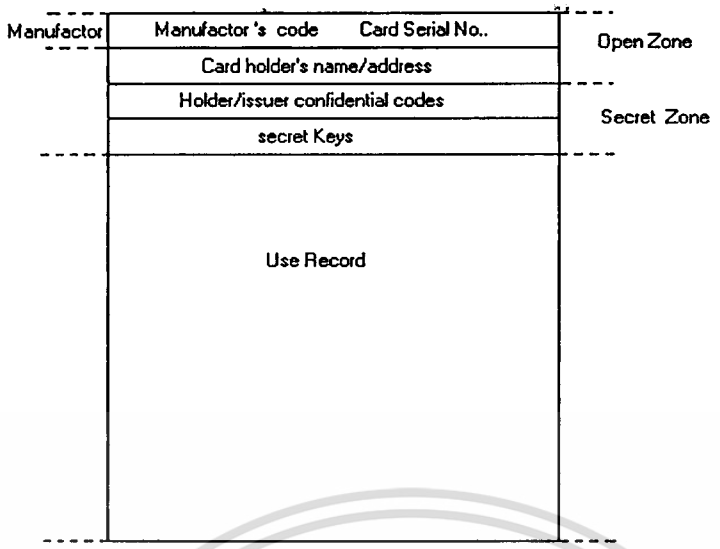
RAM	256 byte to 1 kbyte
EEPROM	1 kbyte to 16 kbyte
ROM	6 kbyte to 24 kbyte
Microprocessor	8 bit ที่ความเร็วประมาณ 5 MHZ
Interface Speed	อย่างต่ำ 9600 bps , Half - Duplex

2.2.1.2 หน่วยความจำ

ส่วนของหน่วยความจำนั้นส่วนที่เก็บข้อมูลของสมาร์ทการ์ด โดยสามารถออกแบบได้ 3 ส่วนดังนี้

- พื้นที่เปิด (Open Zone) คือ พื้นที่ที่ใช้สำหรับเก็บข้อมูลของบริษัทผู้ผลิต Serial Number ข้อมูลของผู้ถือบัตรที่สามารถเปิดเผยได้
- พื้นที่ปกปิด (Secret Zone) คือ พื้นที่ที่ใช้สำหรับเก็บข้อมูลปกปิดส่วนบุคคล
- พื้นที่ใช้งาน (User Zone) คือ พื้นที่เก็บข้อมูลที่ได้ทำการเข้ารหัส

สามารถแสดงได้ดังรูปต่อไปนี้



รูปที่ 2.2 แสดงการแบ่งหน่วยความจำ

2.3 การ์ดหน่วยความจำและการ์ดไมโครโปรเซสเซอร์

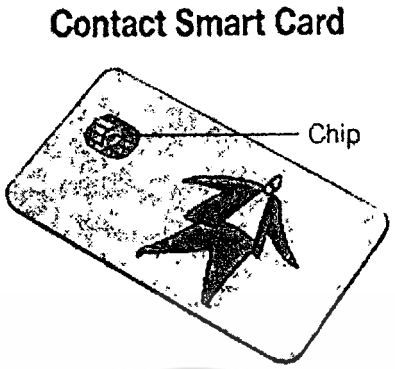
สมาร์ทการ์ดปัจจุบันมีอยู่ 2 ชนิดคือ การ์ดหน่วยความจำ (Memory Card) และการ์ดไมโครโปรเซสเซอร์ (Microprocessor Card) การ์ดหน่วยความจำนั้นทำหน้าที่ง่ายๆ เพียงเก็บข้อมูล (Store Data) และสามารถอ่านออกมาได้คล้ายๆกับแผ่นฟลอปปี้ดิสก์ขนาดเล็กอันหนึ่ง แต่มีความปลอดภัยสูงกว่า ส่วนการ์ดไมโครโปรเซสเซอร์นั้นจะมีจุดเด่นกว่าคือ สามารถเพิ่ม ลบ หรือจัดการกับหน่วยความจำภายในการ์ดได้ด้วย ดังนั้นการ์ดชนิดนี้จึงมีคุณสมบัติคล้ายกับคอมพิวเตอร์ขนาดเล็กๆ ที่เดียว ซึ่งภายในการ์ดจะมีระบบปฏิบัติการ (input/out port operation system) และหน่วยความจำ (Memory) อยู่ภายใน รวมถึงการรักษาความปลอดภัยในการใช้งานก็ถูกติดตั้งไว้ด้วย

2.4 สมาร์ทการ์ดแบบมีคอนแทกต์และไม่มีคอนแทกต์

2.4.1สมาร์ทการ์ดแบบมีคอนแทกต์

สมาร์ทการ์ดสามารถแบ่งออกเป็น 2 ประเภทใหญ่ๆคือ ชนิดมีคอนแทกต์ (Contract Smart Card) และชนิดไม่มีคอนแทกต์ (Contactless Smart Card) สำหรับสมาร์ทการ์ดแบบมีคอนแทกต์นั้นต้องใช้งาน โดยการสอดบัตรเข้ากับเครื่องอ่านบัตร (Smart Card Reader) บัตรชนิดนี้จะมีเพลตสีทองขนาดเล็ก (Small Gold Plate) ที่ด้านหน้าของบัตร ซึ่งแตกต่างจากบัตรเครดิตแบบใช้แถบแม่เหล็กที่พบเห็นได้ทั่วไปซึ่งมีแถบแม่เหล็กอยู่ทางด้านหลังบัตร เมื่อการ์ดถูกสอดเข้ากับเครื่องอ่านบัตรทำให้เกิดการสัมผัสที่คอนแทกต์ของบัตร ทำให้เกิดการเชื่อมต่อทางไฟฟ้า เพื่อใช้ใน

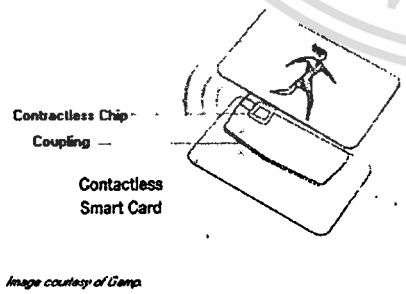
การรับส่งข้อมูลเข้าและออกจากชิปบนบัตรได้ ลักษณะของบัตรสมาร์ทการ์ดแบบมีคอนแทกต์ ดังแสดงรูปที่ 2.3



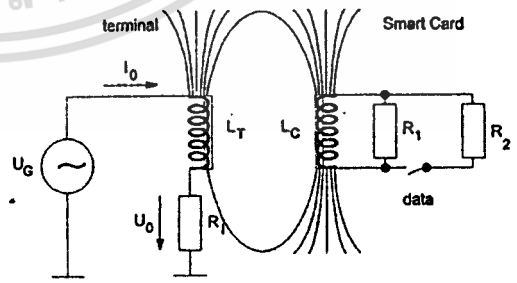
รูปที่ 2.3 สมาร์ทการ์ดแบบคอนแทกต์

2.4.1 สมาร์ทการ์ดแบบไม่มีคอนแทกต์

บัตรสมาร์ทการ์ดอีกชนิดหนึ่งคือชนิดไม่มีคอนแทกต์ (Contactless) มีความพิเศษกว่าตรงที่มีการรับส่งข้อมูลระหว่างชิปภายในบัตรกับภายนอก โดยใช้การส่งผ่านข้อมูลทางอากาศโดยการเหนี่ยวนำ (Coupling Loop) ซึ่งถูกติดตั้งอยู่ภายในบัตร ดังนั้นถ้าดูจากภายนอก ตัวบัตรจะมีลักษณะเหมือนกับบัตรเครดิตพลาสติกทั่วไป แตกต่างกันเพียงแต่ว่าสมาร์ทการ์ดแบบนี้มีไมโครชิปอิเล็กทรอนิกส์และส่วนของการสื่อสารข้อมูลโดยการเหนี่ยวนำติดตั้งอยู่ภายใน ซึ่งอุปกรณ์เหล่านี้ทำให้การ์ดสามารถติดต่อสื่อสารข้อมูลโดยปราศจากการสัมผัส บัตรสมาร์ทการ์ดแบบนี้จึงเหมาะอย่างยิ่งสำหรับงาน ที่ต้องการความรวดเร็วในการใช้งานเช่น ใช้เป็นตั๋วสำหรับระบบรถไฟฟ้าหรือการผ่านเข้าทางด่วนพิเศษที่มีผู้ใช้มากๆ เป็นต้น รูปที่ 2.4 แสดงลักษณะของบัตรสมาร์ทการ์ดแบบไม่มีคอนแทกต์



ก) Contract Less



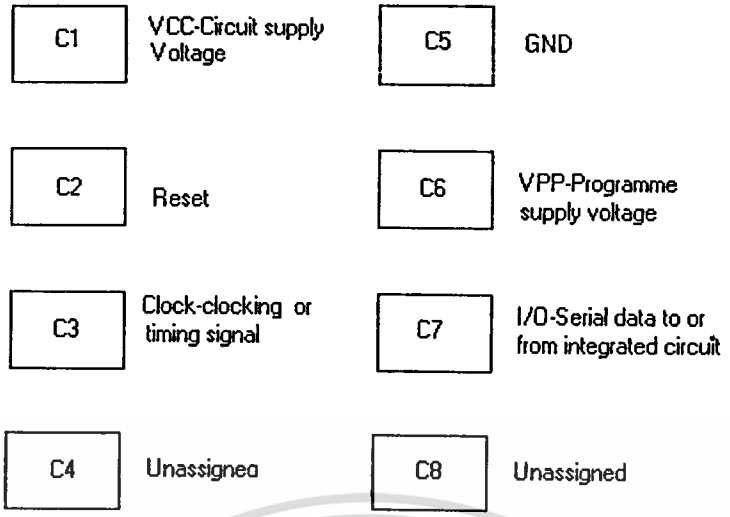
ข) Coupling loop

รูปที่ 2.4 สมาร์ทการ์ดแบบไม่มีคอนแทกต์ ก) Contract Less ข) Coupling Loop

2.5 มาตราฐานของสมาร์ทการ์ด

ในปี ค.ศ. 1981 ได้มีการกำหนดมาตราฐานการส่งผ่านข้อมูลของสมาร์ทการ์ด โดยใช้มาตราฐานของ AFNOR (Association Francais de Normalisation) และได้นำมาตราฐานนี้มากำหนดเป็นมาตราฐานของ ISO ซึ่งปัจจุบันมีมาตราฐานพื้นฐานที่เกี่ยวข้องกับสมาร์ทการ์ดชนิดของคอนแทคต์ถูกกำหนดตามมาตราฐาน ISO 7816 ซีรีส์ซึ่งมีอยู่ทั้งหมด 10 หมวด ในขณะที่บัตรชนิดที่ไม่มีคอนแทกต์จะถูกกำหนดตามมาตราฐาน ISO 1443 มาตราฐานเหล่านี้จะเป็นตัวกำหนดคุณสมบัติต่างของสมาร์ทการ์ด ที่มีการผลิตออกมาใช้งานครอบคลุมตั้งแต่ ลักษณะภายนอกโดยทั่วไป กระบวนการทางไฟฟ้า ทางกลไกและการอินเตอร์เฟส ต่อไปนี้เป็นรายละเอียดตามมาตราฐาน 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 1
  (1992) : protocol T =1 Amendment 2
  (1994) : Revision of Protocol Type Selection mendment 3
  (1998) : Introduction of 3 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 มีการนำรูปแบบและเทคนิคในการรัษาความปลอดภัยบนสมาร์ทการ์ดมาใช้งานในหลายรูปแบบ ซึ่งในที่นี้คงจะไม่ได้กล่าวถึงเทคนิคต่างๆที่นำมาใช้งาน แต่กล่าวเฉพาะรูปแบบและลักษณะหรือระดับการรักษาความาปลอดภัยที่มีใช้กันอยู่ ซึ่งการเข้าถึงข้อมูลต่างๆ บนบัตรได้นั้นถูกแบ่งออกตามลักษณะการควบคุมได้ 2 แบบคือ แบบแรกเป็นการควบคุมว่าผู้ใดจะสามารถเข้าถึงข้อมูลต่างๆในบัตรได้อย่างไรบ้างและ แบบที่ 2 คือควบคุมระดับของการเข้าถึงข้อมูลในบัตร

ในแบบแรกนั้นหมายถึงเป็นการใช้ระบบความปลอดภัยควบคุมว่าต้องการให้ใครบ้างสามารถเข้าถึงข้อมูลได้ ซึ่งแบ่งได้เป็น 3 ระดับ

- เข้าถึงได้ทุกคน คือ บัตรสมาร์ทการ์ดประเภทนี้จะไม่มีการใช้รหัสผ่าน (Password) ผู้ใดที่ถือบัตรนี้สามารถเข้าถึงข้อมูลได้ทันที แต่อาจเป็นข้อมูลบางส่วนเท่านั้นก็ได้ เช่น ข้อมูลเกี่ยวกับชื่อ มีเลือดกรุ๊ปใด เป็นต้น ซึ่งเป็นเสมือนบัตร MediCard ที่สามารถอ่าน ข้อมูลได้โดยไม่ต้องใช้รหัสผ่าน
- เข้าถึงได้เฉพาะผู้ถือบัตร คือ บัตรแบบนี้ส่วนใหญ่มีการใช้รหัสผ่านสำหรับผู้ถือบัตรร่วมด้วยซึ่งมักจะเรียกกันว่า PIN (Personal Identification Number) ที่อาจมีความยาว 4-5 หลักโดยป้อนผ่านทางคีย์แพด ระบบนี้ถ้าหากมีผู้ที่พยายามเข้าถึงข้อมูลภายในบัตรนี้ โดยพยายามป้อนรหัสผ่าน ซึ่งถ้าป้อนรหัสผิดเกิน 3 ครั้งบัตรจะทำการล๊อกตัวเอง และถ้าทำการปลดล๊อคนี้ต้องใช้รหัสผ่านตัวอื่น ซึ่งมีความซับซ้อนกว่ามาป้อนแทน จึงจะกลับมาสู่สภาวะปกติได้

- เข้าถึงได้เฉพาะบุคคลที่ 3 เท่านั้น คือ สมาร์ทการ์ดบางชนิดสามารถเข้าถึงข้อมูลได้เฉพาะบุคคลที่ 3 เท่านั้นซึ่งโดยส่วนใหญ่ก็คือ ผู้ออกบัตรให้เท่านั้น เช่นบัตรสมาร์ทการ์ดที่ใช้แทนเงินอิเล็กทรอนิกส์ สามารถโหลดข้อมูลใหม่ได้เฉพาะธนาคารที่ออกบัตรให้เท่านั้น เป็นต้น

การรักษาความปลอดภัยในแบบที่ 2 คือการจำกัดการเข้าถึงข้อมูลภายในบัตร โดยสามารถแบ่งการเข้าถึงข้อมูลภายในบัตรออกเป็นส่วนๆ ซึ่งส่วนใหญ่แบ่งออกเป็น 4 ระดับ

1. ข้อมูลที่สามารถอ่านได้เท่านั้น (Read Only) คือสามารถอ่านข้อมูลภายในบัตรได้เท่านั้น
2. ข้อมูลที่สามารถเพิ่มได้เท่านั้น (Added Only) คือ สามารถเขียนข้อมูลลงไปในบัตรได้เท่านั้น
3. ข้อมูลที่สามารถอัพเดทได้เท่านั้น (Updated Only) คือ เก็บข้อมูลใหม่ได้เท่านั้น โดยที่ข้อมูลเก่าถูกลบทิ้ง
4. ข้อมูลที่ไม่อนุญาตเข้าถึงได้เลย คือไม่สามารถอ่านหรือเขียนข้อมูลภายในบัตรได้เลย

บัตรโดยทั่วไปหรือบัตรสมาร์ทการ์ดบางประเภทควบคุมการใช้ข้อมูลที่อยู่ในบัตรโดยการใช้รหัสผ่าน ซึ่งมีแต่เพียงผู้ถือบัตรเท่านั้นที่ทราบ แต่วิธีนี้ถ้าหากข้อมูลตรงส่วนนี้จำเป็นต้องถูกส่งผ่านทางสายโทรศัพท์หรือทางคลื่นวิทยุแล้ว คงจะนับได้ว่าการรักษาความปลอดภัยเพียงเท่านี้คงไม่เพียงพอ วิธีหนึ่งที่สามารถนำมาใช้เพื่อเพิ่มความปลอดภัยขึ้นคือ การใช้กระบวนการไซเฟอร์ริ่ง (ciphering) ซึ่งจะมีการทำงานคล้ายกับการแปลภาษาจากข้อมูลหนึ่งไปเป็นภาษาอื่นที่ไม่รู้จัก บางสมาร์ทการ์ดจะถูกบรรจุระบบไซเฟอร์ริ่งและดีไซเฟอร์ริ่ง ซึ่งก็คือการเข้ารหัสและการถอดรหัสเพื่อทำการแปลข้อมูลให้กลับมาเหมือนเดิม ดังนั้นการส่งผ่านข้อมูลจึงทำได้อย่างสมบูรณ์ไม่มีอะไรผิดเพี้ยน

สมาร์ทการ์ดสามารถใช้ระบบไซเฟอร์ริ่งนี้ ทำการแปลข้อมูลที่แตกต่างกันได้มากนับหลายพันล้านรูปแบบและจะทำการสุ่มรูปแบบกันใหม่ทุกครั้งที่มีการสื่อสาร ด้วยวิธีการนี้จึงทำให้เรามั่นใจได้ว่าการใช้งานบัตรสมาร์ทการ์ดในการประยุกต์ในงานทุกรูปแบบและในบางประเภทที่ต้องการความปลอดภัยสูงเช่นการใช้บัตรแทนเงินสด

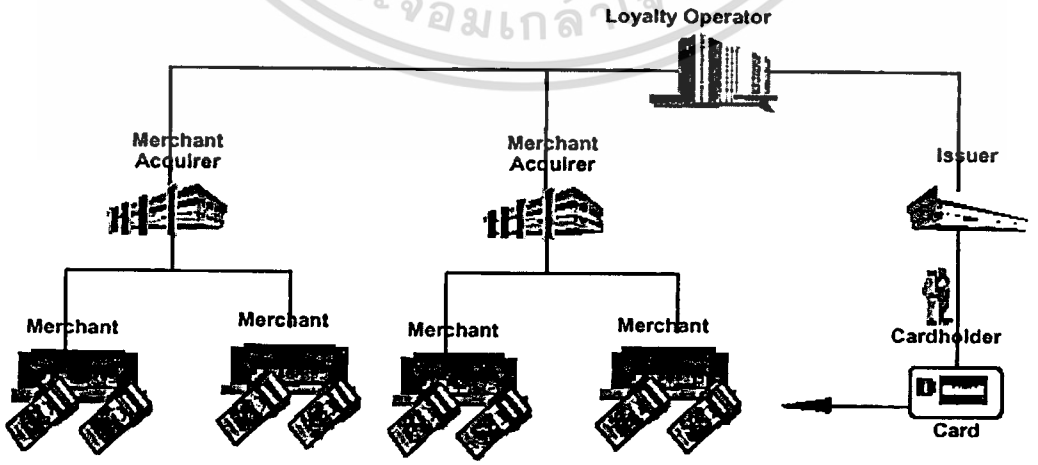
2.7 อนาคตของสมาร์ทการ์ด

คุณสมบัติที่เด่นที่สำคัญของสมาร์ทการ์ดก็คือ เป็นอุปกรณ์อิเล็กทรอนิกส์ที่สามารถพกพาได้ในกระเป๋าสตางค์ของทุกคน ซึ่งมีหน้าที่หลักคือจัดเก็บและจัดการกับข้อมูลต่างๆ ในรูปแบบอิเล็กทรอนิกส์ได้ ความฉลาดของสมาร์ตการ์ดคือ วงจรรวมอิเล็กทรอนิกส์ที่ถูกติดตั้งอยู่ภายในการ์ดพลาสติกซึ่งสามารถที่จะปกปิดข้อมูลพร้อมกับการจัดการข้อมูลภายในได้ และในอนาคตด้วย

เทคโนโลยีคล้ายกันนี้อาจนำพาไปสู่การติดตั้งในรูปแบบอื่นๆ เช่นในกุญแจ นาฬิกา แว่นตา แหวน และอุปกรณ์อื่นๆ ที่เป็นสิ่งใกล้ตัวที่เราต้องใช้อยู่ทุกๆ วัน และในทุกวันนี้ก็มีการใช้เทคโนโลยีสมาร์ทการ์ดนี้ทำเป็นสมาร์ทคีย์ (Smart Key) ในการใช้งานกับระบบเคเบิลทีวีที่มีการบอกรับสมาชิกกันแล้ว การพัฒนาอีกในด้านหนึ่งที่น่าสนใจกว่าคือการพัฒนาทางเทคโนโลยีสมาร์ทการ์ดแบบไม่มีคอนแทกต์ ซึ่งเหมาะมากสำหรับการนำไปสร้างเป็นอุปกรณ์ที่เราเรียกกันว่าแท็ก (Tag) อุปกรณ์ประเภทแท็กนั้นมีการทำงานคล้ายกับสมาร์ทการ์ดแบบไร้คอนแทกต์มากเพียงแต่ว่าอาจจะไม่ได้อยู่ในรูปแบบของบัตรพลาสติกเท่านั้น แต่อาจเป็นรูปแบบของแหวนคล้องหรือป้ายติดสินค้า ซึ่งปัจจุบันมีการนำมาใช้งานกันบ้างแล้วเช่นการนำไปติดเข้ากับถังก๊าซ รถยนต์ สัตว์ เป็นต้น ส่วนใหญ่ทำหน้าที่ในการจัดเก็บข้อมูลที่เกี่ยวข้องกับสิ่งนั้นๆไว้ และอาจเป็นไปได้ที่ผู้ควบคุมอาจทำการเปลี่ยนแปลงข้อมูลภายในโดยไม่ต้องนำมาแก้ไขทีละชิ้นก็ได้ นอกจากนั้นสมาร์ทการ์ดยังอาจนำไปใช้งานร่วมกับระบบรักษาความปลอดภัยชั้นสูง ซึ่งในปัจจุบันเราเรียกกันว่าระบบไบโอเมตริก (Biometrics) โดยการใช้นิ้วมือ ฝ่ามือ เรตินาของตา หรือใช้เสียงในการระบุและแยกแยะแต่ละบุคคล ในอนาคตอาจมีการใช้ข้อมูลทางอิเล็กทรอนิกส์ที่บรรจุอยู่ในสมาร์ทการ์ดมาร่วมประมวลผลข้อมูลอีกด้วย สมาร์ทการ์ดจึงถือได้ว่าเป็นเทคโนโลยีใหม่ที่น่าติดตามและมีความสัมพันธ์ต่อการดำรงค์ชีวิตของ ผู้คนนับหลายล้านคนซึ่งในขณะนี้ก็เริ่มเป็นที่ประจักษ์แล้วเมื่อมีการใช้สมาร์ทการ์ดในการซื้อสินค้าต่างๆในร้านค้าหรือเมื่อใช้บัตรนี้ไปพบแพทย์

2.8 การใช้งานสมาร์ทการ์ด

จากคุณลักษณะของสมาร์ทการ์ด นั้นสามารถนำสมาร์ทการ์ดไปประยุกต์ใช้งานได้หลายแบบ เช่น บัตรเครดิต บัตรเงินสด บัตรประจำตัวประชาชน และอื่นๆ รูปที่ 2.6 แสดงการประยุกต์ใช้งานบัตรสมาร์ทการ์ด ในรูปแบบของบัตรเครดิต ซึ่งสามารถนำไปใช้ตามร้านค้าต่างๆ แทนเงินสด ดังรูป



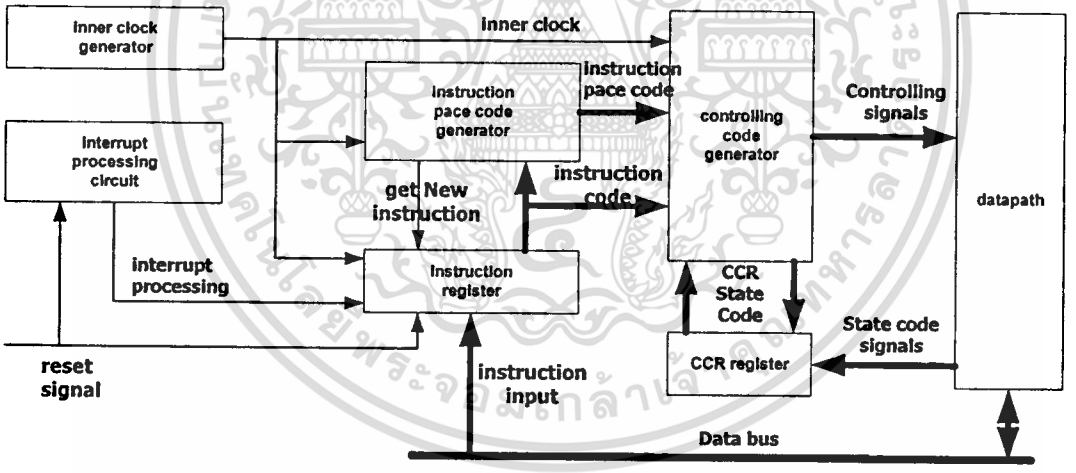
รูปที่ 2.6 แสดงการประยุกต์ใช้งานบัตรสมาร์ทการ์ด

2.9 งานวิจัยที่เกี่ยวข้องกับสมาร์ทการ์ดและการเข้ารหัส/ถอดรหัส

ในส่วนของงานวิจัยที่เกี่ยวข้องกับการทำงานของสมาร์ทการ์และการเข้ารหัส/ถอดรหัสนั้น มีงานวิจัยอยู่หลายแบบด้วยกันเช่น ด้านอัลกอริธึมคือ RSA DES DSA และอื่นๆ ด้านฮาร์แวร์เช่น การพัฒนาด้าน ตัวประมวลผล ด้านสถาปัตยกรรมต่างๆ การออกแบบรูปแบบของการติดต่อสื่อสาร (โปรโตคอล) ในส่วนนี้ได้ยกงานวิจัยที่มีส่วนคล้ายคลึงในวิทยานิพนธ์มาพิจารณา เพื่อเลือกแนวทางในการทำวิทยานิพนธ์ต่อไป

2.9.1 "Design and Implementation of Arithmetic Processor F_2155 for Elliptic Curve Cryptosystem" [16]

งานวิจัยนี้ได้ทำการออกแบบตัวประมวลผล Cryptosystem บน $GF(2^{155})$ บน Normal Basis [5] แบบ Elliptic Curve โดยประกอบด้วยโอเปอร์เรชั่นสำหรับใช้งานทั้งหมด 6 คำสั่งคือ คูณ อินเวอร์ส บวก ยกกำลัง ย้ายค่าระหว่างรีจีสเตอร์ และการกำหนดค่าในรีจีสเตอร์ เพื่อใช้ร่วมกับอัลกอริธึมแบบต่างๆทำการรับข้อมูลอินพุทเป็น Block แบบขนานขนาด 32 บิตซึ่งการออกแบบบน Normal Basis นั้นจะเน้นในส่วนของวงจรอินเวอร์สทำให้เร็วการการออกแบบแบบ Polynomial Basis[15] 10 เท่ารายละเอียดของวงจรแสดงดังรูปที่ 2.7 และภาคผนวก ง.2

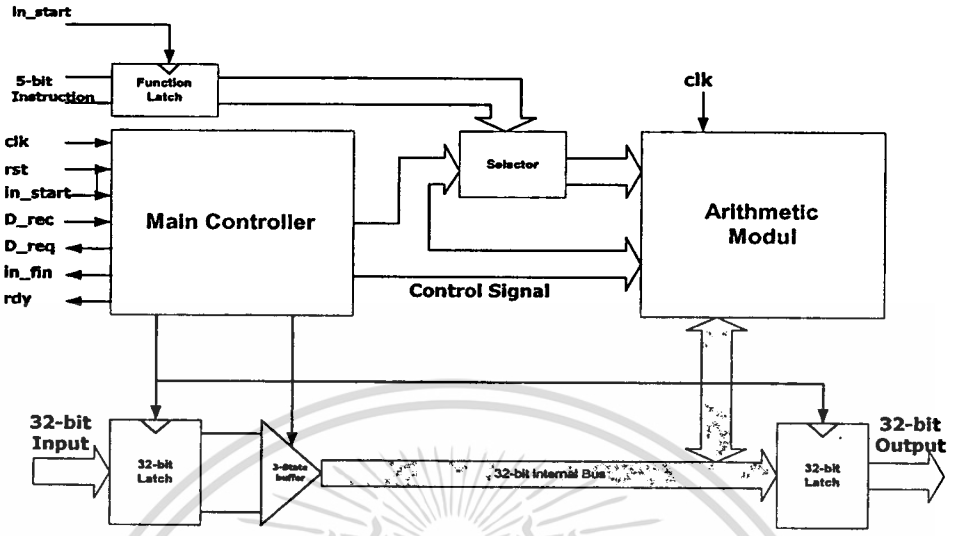


รูปที่ 2.7 แสดงสถาปัตยกรรมของงานวิจัยที่ Design and Implementation of Arithmetic Processor F_2155 for Elliptic Curve Cryptosystem

2.9.2 "Implement of An 8-bit Embedded Microprocessor For Smart Cards" [17]

งานวิจัยนี้ได้ทำการออกแบบออกแบบตัวประมวลผลข้อมูลจำลองการทำงานบนสมาร์ทการ์ด โดยเน้นทางด้านสถาปัตยกรรมของสมาร์ตการ์ด ได้ทดสอบการทำงานการออกแบบดังกล่าวบน FPGA และสร้างต้นแบบทางฟิสิกส์คอล (Physical)ด้วยเทคโนโลยี CMOS รวมเป็นต้นแบบ

ขนาด 0.9 x 0.9 mm ซึ่งในส่วนของของชุดคำสั่งสามารถโหลดชุดคำสั่งผ่านทางซอร์ฟแวร์ได้ รายละเอียดของวงจรแสดงดังรูปที่ 2.8 และในภาคผนวก ง.3



รูปที่ 2.8 แสดงสถาปัตยกรรมของงานวิจัยที่ Implement of An 8-bit Embedded Microprocessor For Smart Cards

2.9.3 สรุปงานวิจัย

เนื่องจากงานวิจัยที่เกี่ยวเนื่องทั้งสองชุดนั้นมีความแตกต่างกัน โดยในชุดแรกนั้นออกแบบโอเปอร์เรชั่นของ Elliptic Curve เพียงอย่างเดียว ซึ่งทำให้ในบางอัลกอริธึมไม่ได้มีการใช้ชุดคำสั่งบางคำสั่งบนตัวประมวลผลนั้น ทำให้เมื่อรวมเป็นชิพแล้วทำให้สิ้นเปลืองฮาร์แวร์ ชุดที่สองออกแบบสถาปัตยกรรมของของสมาร์ทการ์ดในทางฟิสิกส์คอล โดยสามารถโปรแกรมอัลกอริธึมได้ซึ่งแบบนี้เหมาะสำหรับการเป็นชุดทดสอบอัลกอริธึมต่างๆ ดังนั้นจึงได้เกิดแนวคิดในการออกแบบฮาร์แวร์โดยเน้นอัลกอริธึมที่ทำใช้งานเฉพาะอย่าง และทำให้ฮาร์แวร์มีขนาดเล็ก โดยเลือกโอเปอร์เรชั่นให้เหมาะกับการใช้งานบนสมาร์ตการ์ดนั่นเอง

2.10 แนวคิดในการพัฒนาสมาร์ทการ์ด

ในปัจจุบันมีการใช้งานสมาร์ทการ์ดกันอย่างแพร่หลาย อีกทั้งความต้องการใช้งานสมาร์ทการ์ดเพื่อที่จะเก็บข้อมูลเฉพาะบุคคลมีเพิ่มมากขึ้นเช่น ข้อมูลบัตรผู้ป่วยในโรงพยาบาล จึงส่งผลให้ความต้องการในเพิ่มความจุในการเก็บข้อมูลบนสมาร์ทการ์ดเพิ่มมากขึ้น ดังนั้นจึงเกิดแนวคิดที่จะเพิ่มขนาดของหน่วยความจำบนบัตรนั่นเอง เนื่องจากขนาดของ Public-Key ที่ใช้งานในปัจจุบันมีขนาดใหญ่มาก จึงทำให้เกิดทำให้สิ้นเปลืองเนื้อที่บนหน่วยความจำบนสมาร์ทการ์ดซึ่งใช้สำหรับเก็บ Public-Key และยังทำให้วงจรที่ออกแบบมีขนาดใหญ่ ดังนั้นจึงเกิดแนวคิดที่จะทำการลดขนาดของ Public-Key โดยที่ระดับความปลอดภัยยังคงใช้งานได้ดี พร้อมกับวิธีการเข้ารหัสข้อมูลเพื่อทำให้ความจุของข้อมูลบนบัตรสมาร์ทการ์ดเพิ่มมากขึ้น

บทที่ 3
คณิตศาสตร์สนามจำกัด (Finite Field)

บทนำ

ในบทนี้ได้แนะนำคุณสมบัติเป็นฮาร์แวร์ขั้นพื้นฐานที่สำคัญของคณิตศาสตร์สนามจำกัด (Finite Field) พร้อมทั้งการออกแบบโอเปอร์เรชั่น ซึ่งแบ่งออกเป็นวงจรคูณแบบโมดูโล วงจรคูณด้วยค่าคงที่แบบโมดูโล และวงจรบวกแบบโมดูโล ซึ่งแสดงคุณลักษณะของวงจรเพื่อใช้สำหรับการออกแบบเป็นตัวประมวลผลของสมาร์ทการ์ดต่อไป

3.1 กาลัวฟิลด์ (Galois Field Arithmetic)

ในระบบการสื่อสารข้อมูล (Data Communication)นั้น ข้อมูลต่างๆแสดงออกมาเป็นสัญญาณดิจิตอล ซึ่งมีสัญลักษณ์เพียง "0" และ "1" โดยการกระทำกับสัญญาณเหล่านั้นต้องมีการกระทำโอเปอร์เรชั่นซึ่งประกอบด้วย การบวก การลบ การคูณ(Multiplication)และการหาส่วนกลับ(Inverse) ดังนั้น จึงมีคณิตศาสตร์เพื่อใช้สำหรับการทำโอเปอร์เรชั่นดังกล่าวและเรียกคณิตศาสตร์นั้นว่า Finite Field (Galois Field)

ตารางที่ 3.1 การบวกและการคูณแบบคณิตศาสตร์สนามจำกัด

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

$\oplus$ XOR $\otimes$ AND

การบวกและการคูณดังกล่าวเรียกว่าการบวกและการคูณแบบโมดูโล 2 โดยที่ผลลัพธ์ที่ได้จากการทำโอเปอร์เรชั่นบวก ลบ คูณ และ อินเวอร์ส ต้องถูกโมดูโลด้วย 2 จึงได้ผลลัพธ์ก็คือการเศษที่เหลือจากการหารด้วย 2 นั้นเอง

$GF(2^m)$ เป็นเซตซึ่งมีสมาชิกอยู่จำนวน 2^m ตัวที่ไม่ซ้ำกัน โดยที่ m เป็นจำนวนเต็ม และเนื่องจากมีเซตที่ไม่ซ้ำกันจำนวน 2^m นี้เองจึงทำให้สามารถนำ $GF(2^m)$ ไปประยุกต์ใช้ในระบบการสื่อสารแบบต่างๆมากมาย เช่นการการแก้ไขข้อมูลผิดพลาด แบบ BCH Coding (Bose—Chaudhuri-Hocquenghem codes) แบบ Reed Solomon Coding และอื่นๆ ดังนั้นสมาชิกจำนวน 2^m ตัวของ $GF(2^m)$ สามารถหาได้ตามวิธีการต่อไปนี้ อันดับแรกเริ่มจากการกำหนดค่าของ Primitive Polynomial p(x) อันดับที่ m โดยที่ Primitive Polynomial ดังกล่าวก็คือพหุนามที่ไม่สามารถแยกตัวประกอบออกได้กำลังที่ m หรือเรียกว่า Irreducible Polynomial นั่นเองถ้ากำหนดให้ α เป็นรากของโพลิโนเมียล กล่าวคือ $P(\alpha)=0$ ค่ายกกำลังของ α มีค่าได้ถึง 2^m-2 ค่าที่แตกต่างกัน โดยที่ค่ายกกำลังของ α^{2m-1} จะเท่ากับ 1 ใหม่ ดังนั้น 0,1, $\alpha, \alpha^2, \ldots, \alpha^{2^m-2}$ จึงเป็นเซตของ $GF(2^m)$ และเรียก $GF(2^m)$ ว่า Field โดยเรียกสมาชิกของ Field ว่า Field Element โดยแต่ละอิลิเมนท์สามารถเขียนได้ด้วยผลรวมของอิลิเมนท์ที่เป็น $1, \alpha, \alpha^2, \ldots \alpha^{m-1}$ ตัวอย่างสำหรับ $m=4$, $p(x) = x^4 + x + 1$ เป็น Irreducible Polynomial ของ Field $GF(2^4)$ และให้ α เป็นรากของโพลิโนเมียล $p(x) = x^4 + x + 1$ ดังนั้น $\alpha^4 = \alpha + 1$,α สามารถยกกำลังที่ให้ค่าแตกต่างกัน 2^4-2 = 14 ค่า ดังนั้นการกระจายเทอมกำลังต่างๆของ α ทำได้ดังตารางที่ 3.2

ตารางที่ 3.2 กาลัวส์ฟิลด์ 2^4 อีลีเมนท์ ($GF(2^4)$) ซึ่ง $p(x) = x^4 + x + 1$

ฟิลด์อีลีเมนท์ ($GF(2^4)$)	เลขฐานสอง	เลขฐานสิบ
0	0000	0
1	0001	1
α	0010	2
α^2	0100	4
α^3	1000	8
$\alpha^4 = \alpha + 1$	0011	3
$\alpha^5 = \alpha(\alpha+1) = \alpha^2 + \alpha$	0110	6
$\alpha^6 = \alpha(\alpha^2+\alpha) = \alpha^3 + \alpha^2$	1100	12
$\alpha^7 = \alpha(\alpha^3+\alpha^2) = \alpha^4 + \alpha^3 = \alpha^3 + \alpha^2 + 1$	1101	13
$\alpha^8 = \alpha(\alpha^3+\alpha^2+1) = \alpha^4 + \alpha^3 + \alpha = \alpha^2 + 1$	0101	5
$\alpha^9 = \alpha(\alpha^2+1) = \alpha^3 + 1$	1001	9
$\alpha^{10} = \alpha(\alpha^3+\alpha) = \alpha^4 + \alpha^2 = \alpha^2 + \alpha + 1$	0111	7
$\alpha^{11} = \alpha(\alpha^2+\alpha+1) = \alpha^3 + \alpha^2 + \alpha$	1110	14

ตารางที่ 3.2 กาลัวส์ฟิลด์ 2^4 อีลีเมนท์ (GF(2^4)) ซึ่ง $p(x) = x^4 + x + 1$ (ต่อ)

$\alpha^{12} = \alpha(\alpha^3 + \alpha^2 + \alpha) = \alpha^4 + \alpha^3 + \alpha^2 = \alpha^3 + \alpha^2 + \alpha + 1$	1111	15
$\alpha^{13} = \alpha(\alpha^3 + \alpha^2 + \alpha + 1) = \alpha^4 + \alpha^3 + \alpha^2 + \alpha = \alpha^3 + \alpha^2 + 1$	1101	13
$\alpha^{14} = \alpha(\alpha^3 + \alpha^2 + 1) = \alpha^4 + \alpha^3 + \alpha = \alpha^3 + 1$	1001	9
$\alpha^{15} = \alpha(\alpha^3 + 1) = \alpha^4 + \alpha = 1$	0001	1
$\alpha^1 = \alpha$	0010	2

อีลีเม้นท์ α นี้จะเรียกว่าไพรมีทีฟอีลีเม้นท์ (Primitive Element) ของ GF(2^m) ได้โดยอีลีเม้นท์ใดๆของ GF(2^m) ที่ค่ายกกำลังของอีลีเม้นท์นั้นไม่เท่ากับศูนย์บน GF(2^m) แล้วจะเรียกอีลีเม้นท์ดังกล่าวว่าไพรมีทีฟ (Primitive) ตัวอย่าง ค่ายกกำลังของ α ที่ α^4 บน GF(2^4) ในตารางที่ 3.3 คือ

ตารางที่ 3.3 ตัวอย่างของกำลังของ α^4 ของ GF(2^4) ในตารางที่ 3.2 [22]

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

พบว่าทั้ง 15 อีลีเม้นท์ มีค่าไม่เท่ากับ 0 ของ GF(2^4) ดังนั้น α^4 จึงเป็นไพรมีทีฟอิลีเม้นท์ของ GF(2^4) ส่วนอีลีเม้นท์ α^3 ไม่เป็นไพรมีทีฟอิลีเม้นท์เพราะค่าของ $(\alpha^3)^5$ มีค่าเท่ากับ 0 ดังนั้นโพลิโนเมียล p(x) ที่มีอันดับเท่ากับ m ที่สร้างเซตซึ่งมีสมาชิกเท่ากับ 2^m ตัวจะถูกเรียกว่าไพรมีทีฟโพลีโนเมียล จากวิธีการหาค่าไพรมีทีฟโพลิโนเมียลจึงพิสูจน์ได้ว่าแต่ละจำนวนเต็มบวก m จะมีอย่างน้อยที่สุด 1 ไพรมีทีฟโพลิโนเมียล ไพรมีทีฟโพลีโนเมียลต่างๆ พอสรุปมาได้ดังตารางที่ 3.3

ตารางที่ 3.4 ไพรมีทีฟโพลิโนเมียลอันดับต่างๆ [22]

อันดับที่ m	ไพรมีทีฟโพลิโนเมียล
3	$x^3 + x + 1$
4	$x^4 + x + 1$

ตารางที่ 3.4 ไพรมีทีฟโพลิโนเมียลอันดับต่างๆ (ต่อ)

อันดับที่ m	ไพรมีทีฟโพลิโนเมียล
5	x^5+x^2+1
6	x^6+x+1
7	x^7+x^3+1
8	$x^8+x^4+x^3+x^2+1$
9	x^9+x^4+1
10	$x^{10}+x^3+1$
11	$x^{11}+x^2+1$
12	$x^{12}+x^6+x^4+x+1$
13	$x^{13}+x^4+x^3+x+1$
14	$x^{14}+x^{10}+x^6+x+1$
15	$x^{15}+x+1$
16	$x^{16}+x^{12}+x^3+x+1$
17	$x^{17}+x^3+1$
18	$x^{18}+x^7+1$
19	$x^{19}+x^5+x^2+x+1$
20	$x^{20}+x^3+1$
21	$x^{21}+x^2+1$
22	$x^{22}+x+1$
23	$x^{23}+x^5+1$
24	$x^{24}+x^7+x^2+x+1$

3.2 โอเปอร์เรชั่นของกาลัวฟิลด์

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

3.2.1 โอเปอร์เรชั่นการบวก

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

$B(x)=(b_{m-1}x^{m-1}+b_{m-2}x^{m-2}+\ldots+b_1x+b_0)=\sum_{k=0}^{m-1}b_kx^k$ โดยที่ $a,b\in GF(2^m)$

สามารถแสดงโอเปอร์เรชั่นการบวกได้ดังนี้

$$A(x)+B(x)=C(x)=\sum_{i=0}^{m-1}c_ix^i \tag{3.1}$$

โดยที่ $c_i=(a_i+b_i)\bmod 2$

3.2.2 โอเปอร์เรชั่นการคูณ

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

$B(x)=(b_{m-1}x^{m-1}+b_{m-2}x^{m-2}+\ldots+b_1x+b_0)=\sum_{k=0}^{m-1}b_kx^k$ โดยที่ $a,b\in GF(2^m)$

แสดงโอเปอร์เรชั่นการบวกได้ดังนี้

$$A(x)\cdot B(x)=R(x)\bmod p(x)=\sum_{i=0}^{m-1}r_ix^i \tag{3.2}$$

3.2.3 โอเปอร์เรชั่นอินเวอร์ส

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

$$A(x)^{-1}=\frac{1}{C(x)} \tag{3.3}$$

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

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

ตัวอย่าง3.1 แสดงโอเปอร์ชั่นของกาลัวฟิวด์

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

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

กำหนดให้ $A(x)=x^3+x^2+1$ และ $B(x)=x^3+1$ จากสมการที่ (3.1) สามารถแสดงโอเปอร์เรชั่นการบวกได้ดังนี้

$$A(x)+B(x)=C(x)$$

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

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

กำหนดให้ $A(x)=x^3+x^2+1$ และ $B(x)=x^3+1$ จากสมการที่ (3.2) สามารถแสดงโอเปอร์เรชั่นการคูณได้ดังนี้

$$A(x)\cdot B(x)=R(x) \bmod p(x)$$

$$\begin{aligned}(x^3+x^2+1)\cdot(x^3+1) &= x^6+x^5+x^2+1 \\ &= (x^6+x^5+x^2+1) \bmod (x^4+x+1) \\ &= (x^3+x^2+x+1)\end{aligned}$$

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

กำหนดให้ $A(x)=x^3+x^2+1$ จากสมการที่ (3.3) สามารถแสดงโอเปอร์เรชั่นอินเวอร์สได้ดังนี้

$$\begin{aligned}A(x)^{-1} &= \frac{1}{C(x)} \\ &= \frac{1}{x^3+x^2+1} \\ &= x^2\end{aligned}$$

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

3.3 การประยุกต์ใช้งานกาลัวฟิลด์

3.3.1 การออกแบบวงจรคูณ

การคูณในสนามจำกัด(Finite Field) จะมีความยุ่งยากกว่าการคูณโดยทั่วไป เนื่องจากเป็นการคูณแบบโมดูโล และเมื่อนำไปออกแบบเป็นวงจรแล้วยังต้องคำนึงถึงจุดประสงค์ของการประยุกต์ใช้งานและคอมเพล็กซิตี้ของชิฟอีกด้วย สามารถแสดงการออกแบบวงจรคูณแบบต่างๆ ได้ดังนี้

3.3.1.1 การออกแบบวงจรคูณค่าคงที่

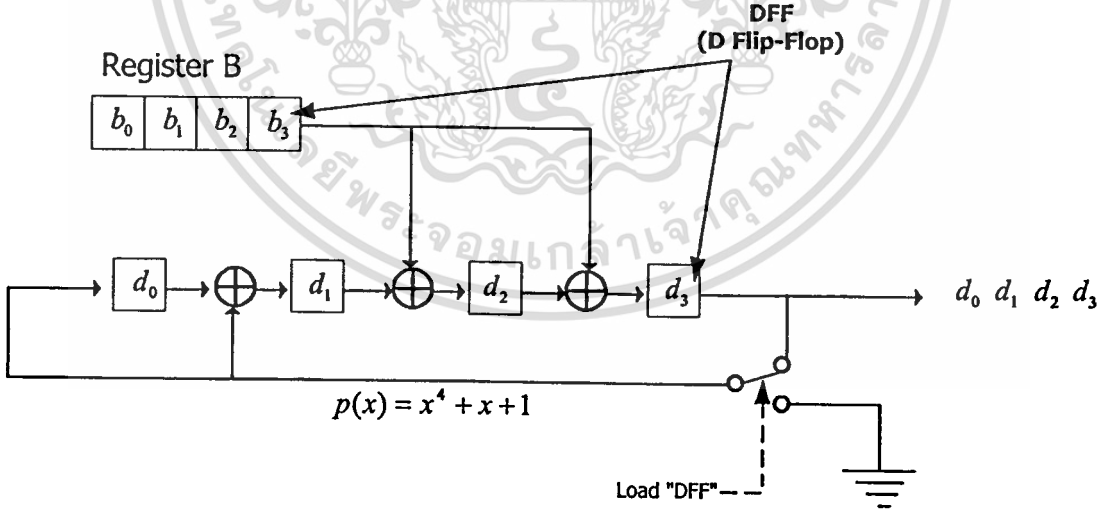
การออกแบบวงจรคูณด้วยค่าคงที่สามารถออกแบบตามชนิดของอินพุทได้ 2 แบบ คือ อินพุทแบบอนุกรมและอินพุทแบบขนาน

3.3.1.1.1 การออกแบบวงจรคูณค่าคงที่แบบอนุกรม

วงจรคูณชนิดนี้จะมีอินพุทแบบอนุกรมซึ่งจะเข้ามาทีละบิตสามารถแสดงเป็นสมการได้ดังนี้

$$D(x) = A(x)B(x) \mod p(x) \tag{3.4}$$

เมื่อ $A(x)$,$B(x)$ และ $D(x)$ เป็นสมาชิกของ $GF(2^n)$ $p(x)$ เป็นพหุนามลดทอนไม่ได้ (Irreducible Polynomial) ของ $GF(2^n)$ ตัวอย่างแสดงการออกแบบใน $GF(2^4)$ เมื่อ $p(x) = x^4 + x + 1$ และค่าคงที่ $A(x) = x^3 + x^2$ และเป็นอินพุทโพลิโนเมียล $B(x) = \sum_{k=0}^{k=3} b_k x^k$ สามารถออกแบบเป็นวงจรดังสมการที่ 3.1 ค่าของ $B(x)$ ถูกชิพเข้าไปในรีจีสเตอร์ $D(x)$ ทีละบิต จนครบ 4 บิต โดยผลลัพธ์นั้นจะต้องใช้ Clock จำนวน 4 Clock Cycle เพื่อชิพค่าออกไป โดยที่ขา Load ถูกต่อลงกราว์ด



รูปที่ 3.1 แสดงวงจรคูณค่าคงที่แบบอนุกรม $GF(2^4)$, $d(x) = (x^3 + x^2) \cdot b(x)$

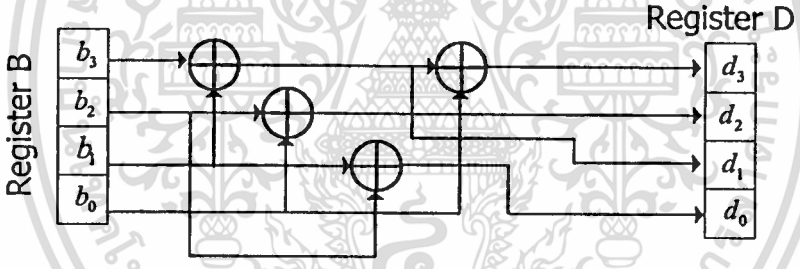
3.3.1.1.2 การออกแบบวงจรคูณค่าคงที่แบบขนาน

วงจรคูณชนิดนี้จะมีอินพุทแบบขนานขนาด n บิตและแสดงดังสมการที่ 3.4 ตัวอย่างการออกแบบวงจรใน $GF(2^4)$ เมื่อ $A(x)$ $B(x)$ และ $p(X)$ เป็นโพลิโนเมียลตามหัวข้อที่ 3.3.1.1.1 แสดงรูปแบบการคูณได้ดังนี้

$$
\begin{aligned}
D(x) &= A(x)B(x) \mod p(x) \\
&= b_0A(x) + b_1A(x) + b_2A(x) + b_3A(x) \\
&= b_0(x^3 + x^2) + b_1x(x^3 + x^2) + b_2x^2(x^3 + x^2) + b_3x^3(x^3 + x^2) \\
&= (b_0 + b_2) + (b_1 + b_3)x + (b_0 + b_2)x^2 + (b_0 + b_1 + b_3)x^3
\end{aligned} \tag{3.5}
$$

จากสมการที่ 3.5 สามารถแสดงในรูปของเมตริก $[D] = [T][B]$ เมื่อ $[T]$ เป็นเมตริก Tranformation แสดงและแสดงเป็นวงจร ดังรูปที่ 3.2 ได้ดังนี้

$$
[d_0, d_1, d_2, d_3] = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} [b_0, b_1, b_2, b_3] \tag{3.6}
$$



รูปที่ 3.2 แสดงวงจรคูณค่าคงที่แบบขนาน GF(2^4), $d(x) = (x^3 + x^2) \cdot b(x)$

3.3.1.2 การออกแบบวงจรคูณแบบปกติ (General Propose)

การออกแบบวงจรแบบนี้เป็นที่นิยมมากที่สุด สำหรับออกแบบเป็นตัวคูณในวงจรซึ่งสามารถแบ่งออกเป็น 2 ชนิดตามชนิดของอินพุทที่เข้ามาคือ อินพุทแบบอนุกรมและอินพุทแบบขนาน

3.3.1.2.1 การออกแบบวงจรคูณแบบปกติแบบอนุกรม

ถ้ากำหนดให้ $A(x)$ และ $B(x)$ เป็นโพลิโนเมียล สามารถนิยามได้ดังนี้

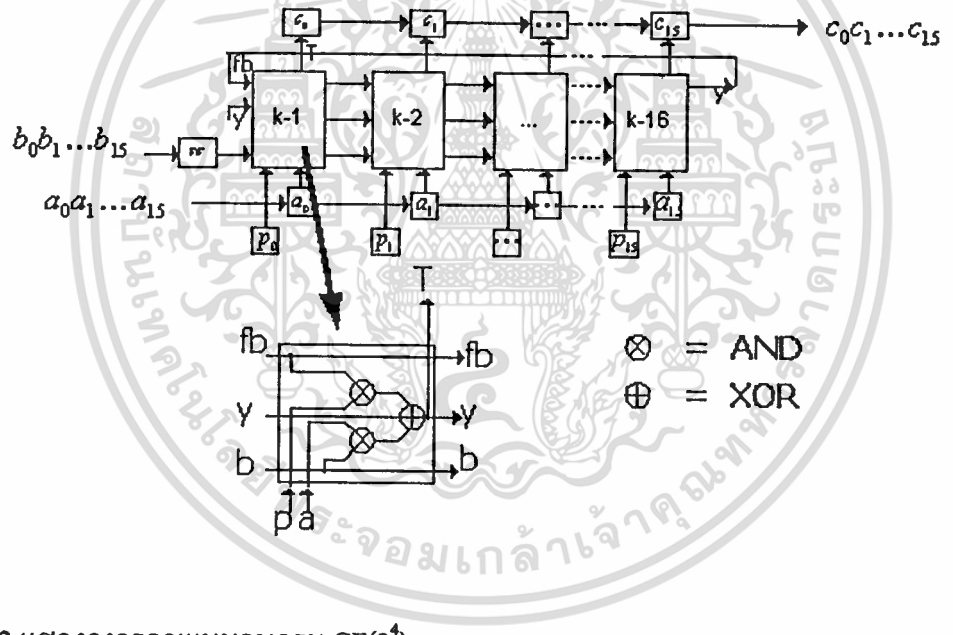
$$
A(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_2x^2 + a_1x + a_0 \tag{3.7}
$$

$$B(x) = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \ldots + b_2x^2 + b_1x + b_0$$

ซึ่งการคูณของ $A(x)$ และ $B(x)$ สามารถแสดงได้ตามสมการที่ 3.4 ดังนี้

$$\begin{aligned} D(x) &= A(x)B(x) \mod p(x) \\ &= A(x)\{b_{m-1}x^{m-1} + \ldots + b_2x^2 + b_1x + b_0\} \mod p(x) \\ &= \{A(x)b_{m-1}x^{m-1} \mod p(x) + \ldots + A(x)b_2x^2 \mod p(x) \\ &\quad + A(x)b_1x \mod p(x) + A(x)b_0 \mod p(x)\} \mod p(x) \end{aligned} \tag{3.8}$$

จากสมการที่ 3.8 จะพบว่าการคูณนั้นจะกระทำซ้ำๆกัน m ครั้งเป็นกลุ่ม ผลคูณของแต่ละกลุ่มจะเป็นอิสระต่อกัน ซึ่งจะเป็นการคูณแบบ Cascade ดังนั้นได้ผลลัพท์เมื่อผ่านไปจำนวน m Clock cycles สามารถแสดงวงจรได้ดังรูปที่ 3.3



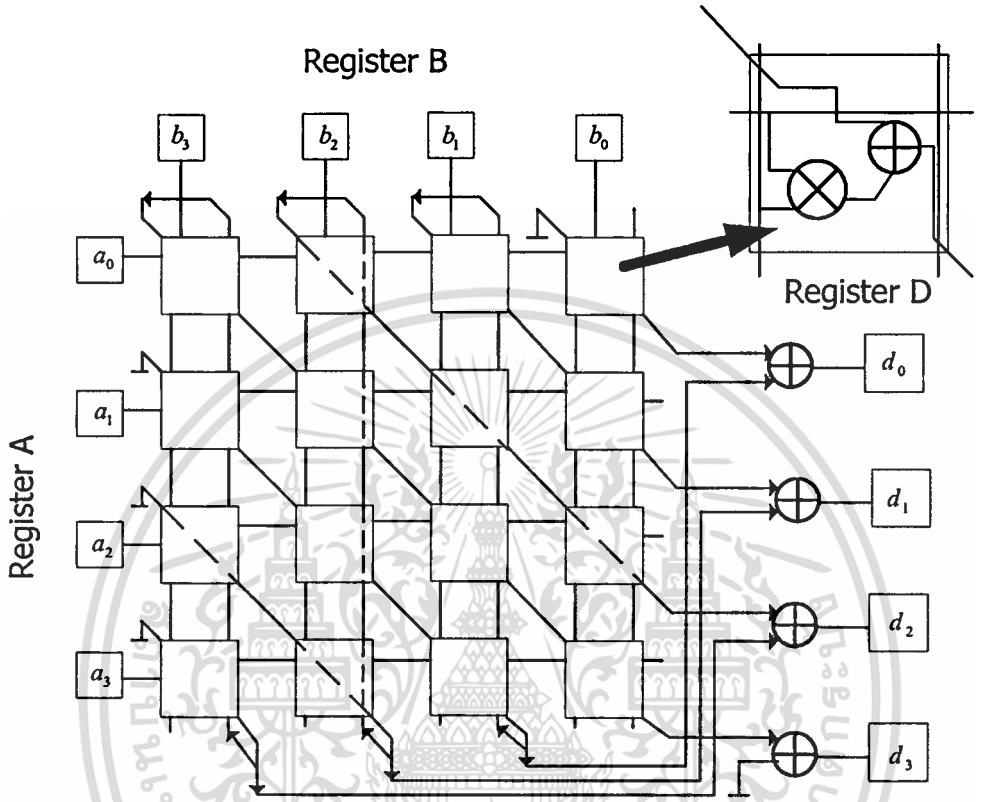
รูปที่ 3.3 แสดงวงจรคูณแบบอนุกรม GF(2^4)

3.3.1.2.2 การออกแบบวงจรคูณแบบปกติแบบขนาน

วงจรคูณแบบนี้จะมีอินพุททั้งสองเป็นโพลิโนเมียลคูณกันดังสมการที่ 3.4 แสดงการออกแบบวงจรคูณได้ดังนี้ ให้ $A(x)$ และ $B(x)$ เป็นดังสมการที่ 3.7 และสมการการคูณแสดงในสมการที่ 3.4 ดังนั้นสามารถแสดงการคูณแบบขนานบน GF(2^4) ได้ดังนี้

$$\begin{aligned} D(x) &= A(x)B(x) \mod p(x) \\ &= (a_1b_3 + a_2b_2 + a_3b_2 + a_0b_0) \end{aligned} \tag{3.9}$$

$$+(a_1b_3 + a_2b_2 + a_3b_1 + a_0b_1 + a_1b_0 + a_2b_3 + a_3b_2)x$$
$$+(a_2b_3 + a_3b_2 + a_0b_1 + a_1b_1 + a_2b_0 + a_3b_3)x^2$$
$$+(a_3b_3 + a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0)x^3$$



รูปที่ 3.4 แสดงวงจรคูณแบบขนาน $GF(2^4)$

3.3.2 การออกแบบวงจรบวก

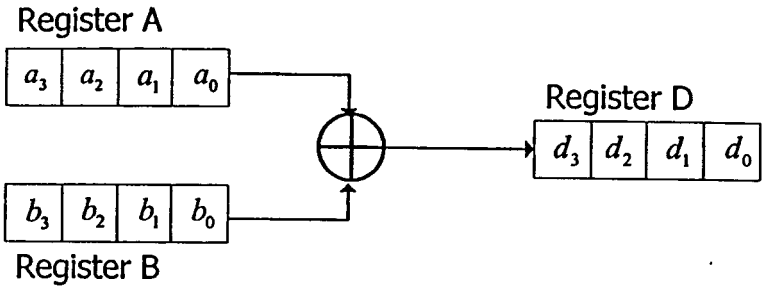
การออกแบบวงจรบวกใน Finite Field $GF(2^m)$ จะเป็นวงจรบวกที่ไม่มีตัวทด (Carry) และตัวยืม (Borrow) ซึ่งจะสมารถแทนได้ด้วย *XOR* เกต

$$C = A \oplus B \tag{3.10}$$

ซึ่งวงจรการบวกนี้สามารถแสดงได้ทั้งวงจรการบวกแบบอนุกรมและแบบขนาน ดังนี้

3.3.2.1 การออกแบบวงจรบวกแบบอนุกรม

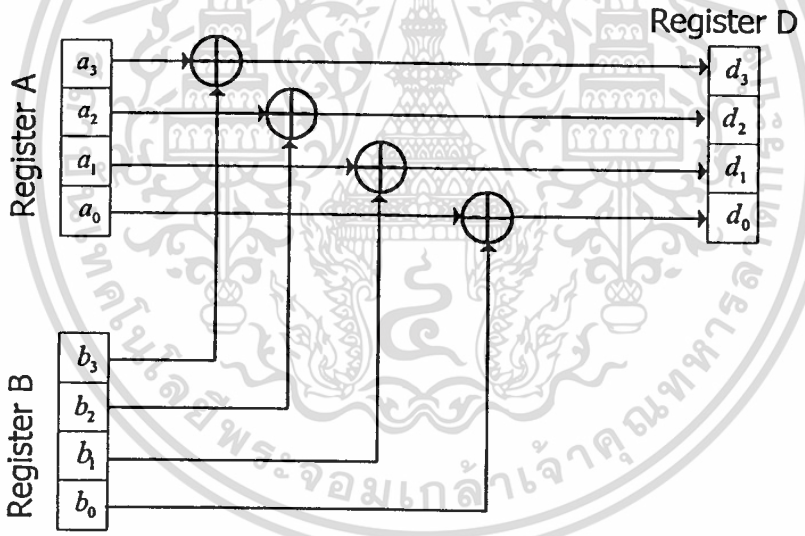
จากสมการที่ 3.10 สามารถแสดงวงจรบวกแบบอนุกรมซึ่งข้อมูลจะถูกส่งเข้ามาทีละบิต แล้วทำการบวกกันแบบ XOR ด้วยวิธีการนี้จะใช้เวลาในการคำนวณ m Clock cycles เมื่อ m เป็นจำนวนบิตของข้อมูล สามารถแสดงตัวอย่างวงจรบวกแบบอนุกรม 4 บิต ได้ดังรูปที่ 3.5



รูปที่ 3.5 แสดงวงจรบวกแบบอนุกรม GF(2^4)

3.3.2.2 การออกแบบวงจรบวกแบบขนาน

จากสมการที่ 3.10 สามารถแสดงวงจรบวกแบบขนาน ข้อมูลจะถูกส่งเข้ามาทีละ m บิต แล้วทำการบวกกันแบบ XOR ซึ่งจะใช้เวลาในการคำนวณ 1 Clock cycles สามารถแสดงตัวอย่างวงจรบวกแบบขนาน 4 บิตได้ดังรูปที่ 3.6



รูปที่ 3.6 แสดงวงจรบวกแบบขนาน GF(2^4)

ในการคูณสัญลักษณ์เหล่านี้สามารถทำได้โดยการบวกกำลังของสัญลักษณ์ ดังใน GF(2^4) ถ้า $\alpha^{15}=1$ แล้ว $\alpha^5\cdot\alpha^7=\alpha^{12}$ และ $\alpha^{12}\cdot\alpha^7=\alpha^{19}=\alpha^4$ และในการหารก็คล้ายคลึงกัน กล่าวคือ $\alpha^{12}/\alpha^5=\alpha^7$ และ $\alpha^4/\alpha^{12}=\alpha^{19}/\alpha^{12}=\alpha^7$ ส่วนในการบวกของสัญลักษณ์ใช้คุณสมบัติจากตารางที่กำหนดไว้เช่น

$$\alpha^5+\alpha^7=(\alpha^2+\alpha)+(\alpha^3+\alpha+1)=\alpha^{13}$$

$$1+\alpha^5+\alpha^{10}=1+(\alpha^2+\alpha)+(\alpha^2+\alpha+1)=0$$

โดยที่ -1 = 1 ดังนั้นการลบจึงมีค่าเหมือนกับการบวก

ตัวอย่างที่ 3.2 การคำนวณสำหรับสมการเชิงเส้น

$$x + \alpha^7 y = \alpha^2 \tag{3.11}$$

$$\alpha^{12}x + \alpha^8 y = \alpha^4 \tag{3.12}$$

คูณสมการที่ 3.2 ด้วย α^3 จะได้

$$\alpha^{15}x + \alpha^{11}y = \alpha^7$$

หรือ $$x + \alpha^{11}y = \alpha^7 \qquad (\text{โดยที่ } \alpha^{15} = 1) \tag{3.13}$$

ในที่สุดจะได้ว่า

$$x + \alpha^7 y = \alpha^2 \tag{3.14}$$

$$x + \alpha^{11} y = \alpha^7 \tag{3.15}$$

บวกสมการที่ 3.14 และ 3.15 เข้าด้วยกัน

$$(\alpha^7 + \alpha^{11})y = \alpha^2 + \alpha^7$$

$$(\alpha^2+1)y = \alpha^3+\alpha^2+\alpha+1$$

$$\alpha^8 y = \alpha^{12}$$

$$y = \alpha^4$$

แทนค่า $y=\alpha^4$ ลงในสมการที่ (3.11)

$$x + \alpha^7\alpha^4 = \alpha^2$$

$$x = \alpha^2+\alpha^{11} = \alpha^2+(\alpha^3+\alpha^2+\alpha)$$

$$x = \alpha^9$$

ดังนั้นคำตอบของสมการที่ (3.11) และ (3.12) คือ $x = \alpha^9$ และ $y = \alpha^4$

3.4 สรุป

โอเปอร์เรชั่นของคณิตศาสตร์แบบ Finite Field ซึ่งประกอบด้วย การบวก การคูณ และการอินเวอร์ส สามารถออกแบบเป็นวงจรทางด้านฮาร์แวร์ได้ ทั้งในรูปของวงจรแบบอนุกรม และแบบขนาน โดยที่แต่ละแบบนั้น มีคุณลักษณะในส่วนของ Clock Cycle Gate Count และ Delay แตกต่างกัน ซึ่งสามารถสรุปได้ดังตารางที่ 3.5 เพื่อเป็นแนวทางสำหรับการออกแบบต่อไป

ตารางที่ 3.5 แสดงการเปรียบเทียบวงจรโอเปอร์เรชั่นของ Finite Field แบบต่างๆ [5].

Type	T	S	T
Serial Adder	m	S_{xor}	mt_{xor}
Parallel Adder	1	MS_{xor}	t_{xor}
Constant Bit Serial Multiplier	m	$m(2S_{dff})+S_{xor}$	$m(t_{dff}+t_{and}+t_{xor})$
General Bit Serial Multiplier	m	$m(2S_{dff}+2S_{and}+S_{xor})$	$m(t_{dff}+t_{and}+t_{xor})$
Constant Bit Parallel Multiplier	1	$m(2S_{dff})+S_{xor}$	mt_{xor}
General Bit Parallel Multiplier	1	$m^2(S_{and}+S_{xor})+mS_{xor}$	mt_{xor}
Inversion	m-1	$mS_{dff}+S_{mul}+0.5S_{xor}$	$m\text{-}1(2t_{dff}+t_{mul})$

T = Clock Cycle

S = Gate Count

t = Delay Time

m = bits

จากตารางที่ 3.5 พบว่าแต่ละวงจรมีคุณลักษณะสำหรับพิจารณา 3 แบบด้วยกันคือ Clock Cycle, Gate Count และ Delay Time ซึ่งในวิทยานิพนธ์นี้ต้องการประมวลผลข้อมูลอินพุทขนาด 155-201 บิต ไม่ต้องการความเร็วสูงมากในการคำนวณ ต้องใช้กำลังงานต่ำ และได้วงจรที่มีขนาดเล็ก จึงทำให้ต้องออกแบบวงจรเป็นแบบอนุกรมซึ่งก็คือ วงจรแบบ Constant Bit Serial Multiplier เนื่องจากขีดจำกัดทางด้านวงจรที่เลือกสำหรับการออกแบบ จึงทำให้เกิดแนวคิดในการปรับปรุงอัลกอริธึมเพื่อให้เหมาะกับความต้องการของวงจรดังกล่าว และในส่วนของอัลกอริธึมนั้นจะได้ศึกษาในบทที่ 5 ต่อไป

บทที่ 4
ภาษาวีเอชดีแอล (VHDL)

บทนำ

วีเอชดีแอล (VHDL) ย่อมาจากคำว่า VHSIC Hardware Description Language (VSHIC : Very High Speed Integrated Circuit) เป็นภาษาคอมพิวเตอร์ระดับสูง (High Level Language) VHDL เป็นภาษาที่ใช้ในการบรรยายหรืออธิบายรูปแบบการทำงานและความสัมพันธ์ของอุปกรณ์ฮาร์ดแวร์ทั้งในระดับเกทจนถึงระดับดิจิตอลที่ซับซ้อน ทั้งนี้เนื่องจาก VHDL เป็นภาษาที่ที่เหมาะในการนำมาใช้ในการเลียนแบบการทำงานของอุปกรณ์ อีกทั้งยังมีความยืดหยุ่นและไม่ถูกจำกัดโดยความสามารถทางเทคโนโลยีใดๆ ทำให้ประหยัดเวลาและค่าใช้จ่ายในการออกแบบ จึงได้มีการนำมาใช้กันอย่างกว้างขวางในวงการอุตสาหกรรม รูปแบบของภาษาวีเอชดีแอลประกอบด้วย 2 ส่วนใหญ่ๆ ได้แก่ ส่วนของภาษาซีเควนเชียล(Sequential Language) และภาษาคอนเคอร์เรนท์ (Concurrent Language) การโปรแกรมด้วยภาษาวีเอชดีแอลสามารถเขียนได้ทั้งสองรูปแบบรวมกัน นอกจากนี้ตัวภาษายังสามารถอธิบายถึงการเชื่อมต่อระหว่างระบบย่อยเข้าด้วยกันเพื่อให้เป็นระบบใหญ่ได้และสามารถกำหนดรูปแบบไวยากรณ์(Syntex) อีกทั้งยังมีการตรวจสอบความหมายของภาษาว่าจะซิมูเลท(Simulate) ได้หรือไม่ เพราะโปรแกรมที่เขียนโดยวีเอชดีแอลต้องผ่านการซิมูเลทเพื่อตรวจสอบการทำงาน ฉะนั้นในการคอมไพล์(Compile) จะมีการตรวจสอบทั้งวงจรและซีมูเลชั่นซีแมนติก(Semantic) อย่างไรก็ตามแม้ตัวภาษาจะมีความซับซ้อนในรูปแบบและกฎเกณฑ์ของภาษา แต่การเรียนรู้เพียงบางส่วนของภาษาก็สามารถนำไปใช้งานโดยไม่จำเป็นต้องศึกษารายละเอียดทั้งหมด

4.1 แนะนำวีเอชดีแอล (Introduction to VHDL)

ในช่วงฤดูร้อนของปี 1981 สถาบันเพื่อป้องกัน (The Institute for Defence Analysis) ในสหรัฐอเมริกาได้จัดตั้งคณะทำงานขึ้นคณะหนึ่ง เพื่อทำการพัฒนาภาษาที่ใช้ในการบรรยายหรืออธิบายรูปแบบการทำงานและความสัมพันธ์ ของอุปกรณ์ฮาร์ดแวร์แบบใหม่ขึ้น ผลการทำงานของคณะทำงานชุดนี้ได้ก่อให้เกิดภาษาการบรรยายฮาร์ดแวร์ขึ้น เรียกว่า VHDL (VHSIC Hardware Description Language) โดย VHISC เป็นชื่อย่อของแผนกหนึ่งของสถาบันที่ทำงานเกี่ยวกับวงจรรวมที่มีความเร็วสูงมาก (VERY High Speed Intergrated Circuit) ต่อมาในปี 1985 IEEE ได้ทำการผลักดันให้ VHDL กลายเป็นภาษาที่เป็นมาตรฐานและมีการยอมรับกันอย่างกว้างขวางในวงการอุตสาหกรรมคอมพิวเตอร์ ด้วยความสามารถของ VHDL ในด้านของการกำหนดพฤติกรรม

การทำงานของวงจร ทำให้นักออกแบบสามารถกำหนดรูปแบบพฤติกรรมการทำงานได้ทั้งวงจรของดิจิตอลทั่วๆ ไป และในระบบที่แตกต่างกันออกไป เช่น พฤติกรรมการทำงานของระบบเรดาร์หรือพฤติกรรมการทำงานของระบบเครือข่ายใยประสาทในสมองมนุษย์ได้ ข้อดีหลักที่สำคัญของ VHDL ก็คือภาษานี้จะสามารถถูกใช้ได้ตลอดในทุกๆ ระดับขั้นของการออกแบบที่ต่างกันได้ นั่นคือในกระบวนการออกแบบตั้งแต่ระดับสูง(System Level) จนถึงระดับที่ต่ำกว่า(Lower hardware level) สามารถใช้ภาษาเดียวกันได้โดยตลอด ทำให้แพิ่มประสิทธิภาพในการติดต่อระหว่างกลุ่มที่ทำงานร่วมกันได้เป็นอย่างดี

4.1.1 ข้อกำหนด (VHDL Requirement)

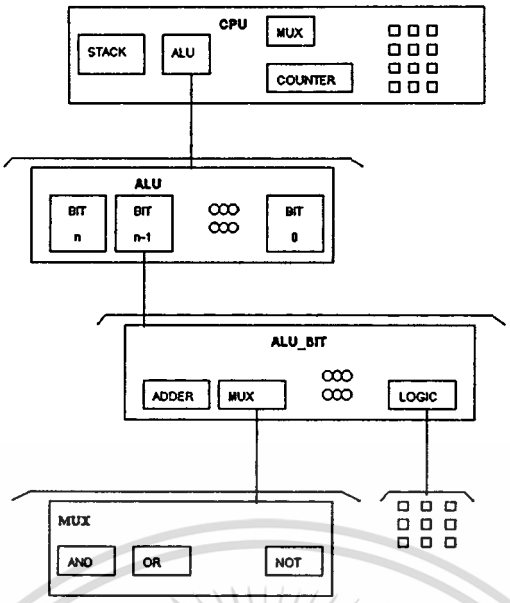
ในเอกสารของ DoD (Department of Defense Requriment for Hardware Description Language) ซึ่งออกมาในเดือนมกราคมปี 1983 ได้ตั้งข้อกำหนดสำหรับภาษา VHDL ไว้ดังนี้

4.1.1.1 ลักษณะทั่วไป (Generation Features)

เอกสารของ DoD กำหนดไว้ว่า VHDL เป็นภาษาสำหรับการออกแบบและบรรยายของฮาร์ดแวร์ ซึ่งหมายถึงความสามารถในการอธิบายและออกแบบในระดับสูง ความสามารถในการเลียนแบบ(Simulation) การสังเคราะห์(Synthesis) และการทดสอบ(Testing) นอกจากนั้น VHDL ยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์คือ ระบบจนถึงระดับเกทอีกด้วย เนื่องจากในการทำงานของระบบดิจิตอลจริงๆ ทุกๆ องค์ประกอบภายในระบบไม่ว่าเล็กหรือใหญ่จะทำงานไปพร้อมๆ กัน ซึ่งในเรื่องของความพร้อมเพรียงในการทำงานนี้ถือว่าเป็นข้อกำหนดที่สำคัญอย่างหนึ่งใน VHDL ด้วยเช่นกัน (สำหรับในภาษาที่ใช้ในการบรรยายฮาร์ดแวร์แล้ว ความพร้อมเพรียงจะหมายถึงทุกๆ คำสั่ง องค์ประกอบเกท หรือวงจรต่างๆ จะถูกนำมาปฏิบัติทั้งหมด ดังนั้นในตอนท้ายแล้วก็จะดูเหมือนว่า ได้มีการปฏิบัติไปพร้อมๆ กัน)

4.1.1.2 สนับสนุนการออกแบบแบบลำดับขั้น (Support for Design Hierarchy)

การออกแบบลำดับขั้น เป็นลักษณะที่สำคัญอย่างหนึ่งสำหรับการออกแบบที่มีหลายๆ ระดับในการออกแบบจะประกอบด้วยส่วนการบรรยายการเชื่อมต่อและส่วนการบรรยายหน้าที่การทำงาน หน้าที่การทำงานของระบบก็สามารถกำหนดได้ด้วยตนเองหรือถูกกำหนดโดยโครงสร้างที่ประกอบด้วยองค์ประกอบย่อยๆ ลงไปได้เช่นกัน แต่ที่ระดับล่างสุดองค์ประกอบต้องถูกบรรยายหน้าที่การทำงานด้วยตัวมันเองและไม่สามารถกำหนดการทำงานโดยลักษณะแบบโครงสร้างได้ ดังรูปที่ 4.1



รูปที่ 4.1 แสดงตัวอย่างการออกแบบแบบลำดับขั้น

4.1.1.3 ไลบรารี (Library Support)

VHDL ได้สนับสนุนการมีไลบราลีเพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษณะและการทำงานของอุปกรณ์พื้นฐานไว้ในระบบไลบรารี หรือจะใช้ไลบรารีที่ระบบได้จัดเตรียมไว้แล้วก็ได้ โมเดลและการบรรยายที่ถูกต้องควรจะถูกเก็บไว้ในไลบรารี หลังจากที่ได้ผ่านการคอมไพล์เรียบร้อยแล้ว เพื่อให้ผู้ออกแบบคนอื่นๆ สามารถนำไปใช้ได้ด้วย

4.1.1.4 ลำดับคำสั่ง (Sequential Statement)

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการโดยพร้อมเพรียงกันจะเป็นคุณสมบัติที่สำคัญของ VHDL ก็ตาม ตัวภาษาเองยังได้มีการจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วย เมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ประกอบที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้ว ผู้ออกแบบก็ยังสามารถบรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดภายในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรมที่ประกอบด้วยโครงสร้างแบบ case if-then-eles และ loop ทั่วๆ ไปได้ การบรรยายแบบลำดับคำสั่งทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้สะดวกและง่ายขึ้น อย่างไรก็ตามโครงสร้างทั้งหมดของ VHDL ก็ยังคงเป็นการทำงานพร้อมเพรียงกันเช่นเดิม

4.1.1.5 การกำหนดคุณสมบัติ (Generic Design)

นอกจากการกำหนดอินพุทและเอาท์พุทแล้ว เงื่อนไขอื่นๆ ก็มีผลต่อการปฏิบัติหน้าที่ของอุปกรณ์ฮาร์ดแวร์ด้วยเช่นกัน สิ่งนี้ก็รวมถึงสภาพแวดล้อมและลักษณะทางกายภาพของอุปกรณ์นั้นๆ ภาษาสำหรับการออกแบบที่ดีควรจะสามารถให้ผู้ออกแบบกำหนดคุณสมบัติของอุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนดขนาด ลักษณะทางกายภาพ เวลา โหลดและเงื่อนไขทางสภาพแวดล้อมอื่นๆ ความสามารถในการกำหนดคุณสมบัติก็เป็นส่วนหนึ่งที่มีอยู่ในภาษา VHDL ด้วยเช่นกัน

4.1.1.6 ชนิดของข้อมูล (Type Declaration and usage)

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

4.1.1.7 โปรแกรมย่อย (Use of Subprogram)

ความสามารถในการใช้ฟังก์ชันและโพรซีเจ อร์(Procedure) เป็นข้อกำหนดอีกอย่างหนึ่งในVHDL เราสามารถที่จะใช้โปรแกรมย่อยในการเปลี่ยนแปลงชนิดของข้อมูล การกำหนดหน่วยของลอจิก(Logic) การกำหนดตัวกระทำต่างๆ ทั้งเก่าและใหม่หรืออะไรก็ตามได้เช่นเดียวกับการเขียนโปรแกรมทั่วไป

4.1.1.8 การควบคุมเวลา (Timing Control)

VHDL อนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ตามต้องการการตรวจสอบ การออกแบบเกต หรือการหน่วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอยเหตุการณ์(Event) นอกจากนี้ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

4.1.1.9 การกำหนดแบบโครงสร้าง (Structural specification)

การกำหนดโครงสร้างขององค์ประกอบสามารถกระทำได้ในทุกๆ ระดับของการออกแบบ การกำหนดโครงสร้างขององค์ประกอบรวม ที่เกิดจากองค์ประกอบย่อยที่ต่างกันหรือเหมือนกันก็เป็นข้อกำหนดมาตรฐานอย่างหนึ่งเช่นกัน

4.2 ความสามารถของภาษาวีเอชดีแอล (Capability)

- ตัวภาษา VHDL สามารถใช้เป็นสื่อกลางในการแลกเปลี่ยนระหว่างผู้ผลิตชิพกับผู้ออกแบบ (CAD Tools)

- ใช้เป็นสื่อกลางในการแลกเปลี่ยนสื่อสารระหว่างซีเออี(CAE) และซีเอดีทูล(CAD Tools) เช่นตัวภาษาซ้อสโค้ด(SOURCE CODE) ของ VHDL สามารถคอมไพล์โดยใช้คอมไพล์เลอร์ (Compiler) และซิมูเลเตอร์(Simulator) ได้หลายตัวแตกต่างกัน

- ภาษา VHDL สนับสนุนการออกแบบ แบบท๊อปดาวน์(Top Down Design) และแบบบัททอมอัพ(Bottom Up Design) หรือผสมกันทั้งสองแบบ

- ตัวภาษา VHDL เป็นแบบทั่วไป(Generic)ไม่อิงเทคโนโลยีอันใดอันหนึ่ง ในขณะเดียวกันก็สนับสนุนหลายๆ เทคโนโลยี

- ตัวภาษา VHDL สามารถอ่านและทำความเข้าใจได้โดยมนุษย์

-สนับสนุนการออกแบบทั้งระบบซิงโครนัส(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

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

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

-ภาษา VHDL เป็นมาตรฐานที่ใช้โดยบริษัทและผู้ออกแบบหลายๆ แห่ง ฉะนั้นจึงง่ายที่จะทำความเข้าใจถึงแม้ว่าจะมาจากแหล่งต่างๆ

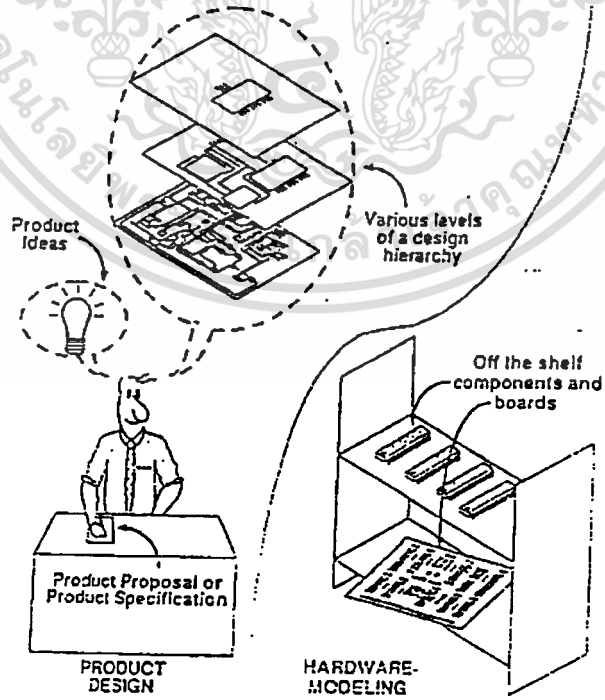
-โมเดลที่สร้างขึ้นสามารถจำลองการทำงานได้ เพราะว่าตัวแปรภาษาได้ตรวจสอบตัวแปรทางซิมูเลชั่นซีแมนติกไว้ด้วย

4.3 หลักการสร้างโมเดลโดยใช้ภาษาวีเอชดีแอล

(General VHDL Modelling Principles)

วีเอชดีแอลเป็นภาษาที่ใช้สำหรับอธิบายการทำงานของฮาร์ดแวร์ในรูปแบบฟอร์มที่อ่านเข้าใจได้ ซึ่งจะช่วยในการสร้างและออกแบบวงจรรวมดิจิตอลและส่วนประกอบต่างๆ อาจใช้อธิบายระบบทั้งระบบหรืออธิบายเพียงบางส่วน ซึ่งอยู่ในรูปของ (Component Block) จากนั้นก็ทำการจำลองการทำงาน(Simulate) โดยที่รูปแบบนั้นยังไม่ได้สร้างขึ้นจริงหรือเพียงแต่อยู่ในรูปของคำอธิบายเท่านั้น(Textual Format) หลังจากจำลองการทำงานจนได้ตามที่ต้องการจึงนำไปทำการ Synthesis เพื่อให้ได้วงจรเกทเลเวลต่อไป ประโยชน์จริงของการใช้วงจร วีเอชดีแอลเป็น Design Tools แทนการสร้างต้นแบบ (Prototype) ขึ้นมาจริง คือเราสามารถอธิบาย Product Idea , Product Proposal, Product Specification ในรูปของ Text จากนั้นก็นำคอมไพล์เพื่อดู Timing การทำงานแล้วแก้ไข (Refine) จนกว่าจะได้ Specification ตามต้องการ เมื่อ product ได้ผลตามที่ต้องการแล้วจึงนำไปสู่การสังเคราะห์(Synthesis) เพื่อให้ได้เกทเลเวล Schematic เพื่อนำไปสร้างเป็นต้นแบบจริงต่อไป ซึ่งต้นแบบที่สร้างนั้นทำงานได้จริงเพราะได้ทำการ Simulate เรียบร้อยแล้ว เป็นการลดเวลาและค่าใช้จ่ายในการสร้างต้นแบบได้มาก

ตัวภาษา VHDL สนับสนุนหลักการต่างๆ ให้เขียนแก้ไขและบำรุงรักษาวงจรดิจิตอลที่มีความซับซ้อนให้เป็นไปอย่างรวดเร็วและมีประสิทธิภาพโดยมีหลักการดังนี้



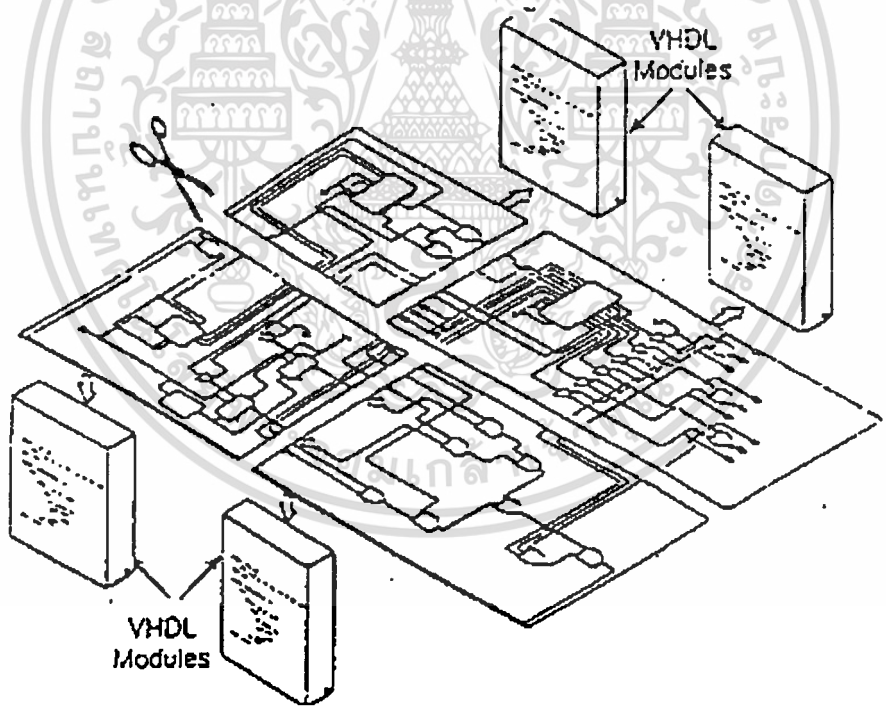
รูปที่ 4.2 สิ่งต่างๆ ที่สามารถอธิบายได้ด้วย VHDL

4.3.1 Top Down Design

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

4.3.2 Modularity

Modularity คือ หลักการในการแยกส่วน (Partitioning) ฮาร์ดแวร์ ออกเป็นส่วนย่อยเล็กลงไป ซึ่งปกติการทำงานของฮาร์ดแวร์ใหญ่ๆ ต้องประกอบด้วยฮาร์ดแวร์ย่อยๆ ลงไปดังรูปที่ 4.3

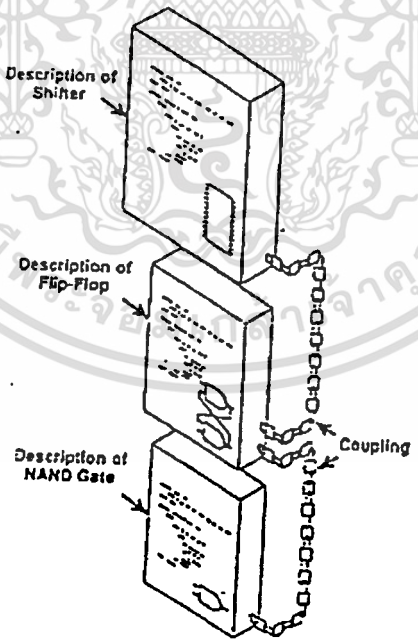


รูปที่ 4.3 การแบ่งย่อยในระดับราบของการออกแบบฮาร์ดแวร์

จากรูปได้แสดงวงจรทั้งหมดในรูปเดียว (Flatten Design) หลังจากนั้นตัดเป็นส่วนย่อยๆ เล็กลงมา เมื่อเราออกแบบโดยใช้ภาษา VHDL หน้าที่การทำงานของวงแต่ละส่วนต้องสามารถ

อธิบายได้โดยโมดูลของโค้ด(คล้ายฟังก์ชั่นหรือโพรซีเดอร์) ซึ่งแสดงการทำงานของส่วนย่อยนั้นอย่างชัดเจน ซึ่งการแยกรูปใหญ่ๆ ออกเป็นส่วนย่อยๆ นี้ ทำให้ง่ายต่อการจัดการและง่ายต่อการทำความเข้าใจ

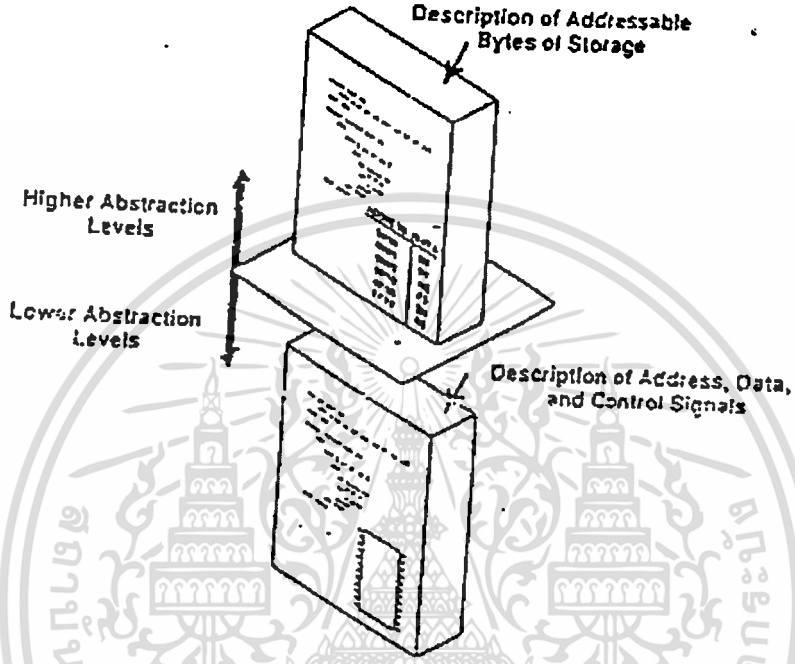
รูปที่ 4.4 แสดง Hierarchy Method โดยการแยกส่วนรูปแบบออกเป็นส่วนย่อยๆ ส่วนบนสุดอธิบายการทำงานของ Shifter ส่วนล่างๆ ลงมา คือการแยกส่วนของ Shifter ออกเป็นฟลิปฟลอป จากฟลิปฟลอปแยกเป็น NAND เกท ภายใน Shifter ได้อธิบายการทำงานโดยใช้การต่อกันของฟลิปฟลอป ในระดับที่ต่ำลงมา ฟลิปฟลอปก็เกิดจาก NAND เกท ต่อกัน 2 ตัวในระดับที่ต่ำลงมาอีกก็เป็น NAND เกท ซึ่งมีการอธิบายการทำงานอยู่ภายใน โดยแต่ละโมดูลจะมีคำอธิบายการทำงานในตัวของมันเองอยู่แล้วคำอธิบายในแต่ละโมดูลมีไว้เพื่อให้สามารถใช้ฟลิปฟลอปโมดูลได้ ส่วนฟลิปฟลอปโมดูลก็อธิบายการเชื่อมต่อไว้อย่างดีทำให้สามารถเชื่อมต่อกับ NAND เกท ในระดับล่างสุดได้ ประโยชน์อย่างหนึ่งของการแยกส่วนฟลิปฟลอปและ NAND เกท ออกจากกัน เนื่องจากทำให้ง่ายในการใช้ NAND เกท ตัวนี้ในรูปแบบไฮเลเวลตัวอื่นๆ ทำให้นำออกไปใช้งานได้อีกและลดความซับซ้อนในการใช้อุปกรณ์เพื่อแก้ไขการทำงานของ Shifter ง่ายขึ้น โดยปราศจากการแก้ไขฟลิปฟลอปและ NAND เกท ประโยชน์ที่ได้จากการทำ Modularity นี้ทำให้รูปแบบที่ออกแบบง่ายต่อการเข้าใจและแก้ไขได้เสมอ



รูปที่ 4.4 การแบ่งแบบ Hierarchy ของ VHDL Shifter Description

4.3.3 Abstraction

คำนิยามของรูปแบบ จะอธิบายการทำงานของตัวรูปแบบมากกว่าจะอธิบายว่าพัฒนาตัวรูปแบบนั้นได้อย่างไร หลักการนี้มีความสำคัญอย่างใกล้ชิดกับหลักการของ Modularity ในรูปที่ 4.4 ฟลิปฟลอป เป็นการนิยามในการใช้ NAND เกท และ Shifter เป็นนิยามในการใช้ฟลิปฟลอป

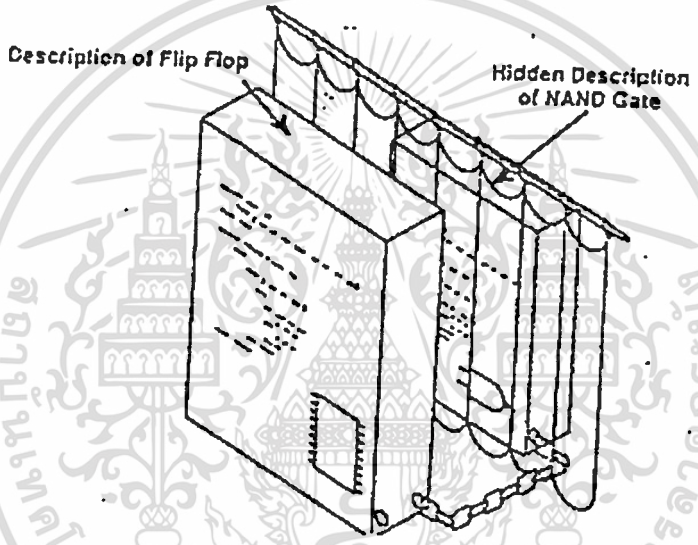


รูปที่ 4.5 Applying Abstraction to a ROM Description

รูปที่ 4.5 แสดงถึงการอธิบายการทำงานของรูปแบบโดยใช้ VHDL ในหลายๆ ระดับของการนิยาม ROM (Read Only Memory) อธิบายโดยใช้ภาษาระดับสูง แสดงถึงตำแหน่งต่างๆ ซึ่งเก็บข้อมูลไว้ ในตำแหน่งนั้นๆ ที่ระดับนี้ไม่ต้องสนใจถึง Address Line , Data Line และ Control Line เราสามารถเพ่งจุดสนใจไปที่ขนาดของข้อมูลโดยไม่ต้องคำนึงถึงสัญญานควบคุมต่างๆ ภายในเพราะว่าส่วนนั้นจะถูกจัดการเองในระดับที่ต่ำลงมา ในระดับล่างลงมาเราสามารถอธิบายการทำงานของสัญญาณแต่ละเส้นภายใน ROM ในการจัดการสัญญาณภายในทุกเส้นภายใน การที่จะอ่านข้อมูลหรือโปรแกรมข้อมูลใน ROM ถ้าต้องการเปลี่ยนค่าข้อมูลภายใน ROM ควรแก้ไขในระดับที่สูงขึ้นมาจะง่ายกว่าการควบคุมสัญญาณภายใน จะเห็นว่าแต่ละระดับมีความเหมาะสมแตกต่างกันออกไปทำให้รูปแบบที่เราออกแบบไปง่ายต่อการแก้ไขโดยการใช้ประโยชน์ของ Abstuction

4.3.4 Information Hiding

เมื่อทำการเขียน VHDL Code ขึ้นมาเพื่ออธิบายการทำงานของฮาร์ดแวร์ตัวหนึ่ง บางครั้งอาจต้องการที่จะซ่อนรายละเอียดการพัฒนาโมดูล นั้นโดยไม่ต้องการให้ส่วนโมดูลอื่นๆ รู้การทำงานภายใน Information Hiding มีประโยชน์คือ ทำให้รูปแบบภาษา VHDL นั้นสามารถจัดการและอ่านเข้าใจได้ง่าย หลักการนี้จะสนับสนุนหลักการ Abstraction คือสนใจรายละเอียดในการใช้งานมากกว่าจะสนใจว่ารูปแบบนั้นจะถูกสร้างขึ้นมาอย่างไร เป็นต้น การซ่อนรายละเอียดภายในโมดูลทำให้ความสนใจของผู้ออกแบบนั้นสนใจไปในส่วนที่สำคัญมากกว่า ในส่วนที่ไม่น่าสนใจจะซ่อนไว้และเข้าถึงไม่ได้ ดังรูปที่ 4.6



รูปที่ 4.6 การซ่อนรายละเอียดที่ไม่จำเป็นของระดับ NAND เกท

อธิบายการทำงานของฟลิปฟลอปไม่ต้องสนใจว่า NAND เกท จะทำงานอย่างไร จะต่อกันภายในอย่างไร โดย NAND เกท สามารถเขียนขึ้นมาแล้วคอมไพล์เก็บไว้ในไลบรารี ผู้ที่ออกแบบฟลิปฟลอประดับสูงขึ้นมา เพียงแต่ต้องรู้ว่า จะเชื่อมต่ออินพุท/เอาท์พุทของ NAND เกท มาใช้งานได้อย่างไร โดยไม่ต้องสนใจว่า NAND เกท จะถูกสร้างและพัฒนามาอย่างไร ประโยชน์อีกอย่างหนึ่งคือ ป้องกันข้อมูลภายใน ในกรณีที่แจกจ่าย VHDL โมเดลไปยังที่อื่นทำให้เราป้องกันทรัพย์สินทางปัญญาได้อีกในระดับหนึ่ง

4.3.5 Uniforminity

Uniforminity เป็นหลักการอีกอย่างหนึ่งที่ช่วยในการอธิบายฮาร์ดแวร์ด้วยภาษา VHDL หมายถึง การสร้างโมดูลของรหัส ในลักษณะคล้ายกัน โดยใช้ตัวภาษา VHDL Building Block ทำให้เกิดการเขียนรหัสที่ดีอย่างเช่น มีการใช้ย่อหน้า มีการใช้คำอธิบาย(Comment) เป็นต้น ทำให้การพัฒนาโมดูลทำความเข้าใจง่าย

4.4 องค์ประกอบพื้นฐานในวีเอชดีแอล (Basic concept in VHDL)

รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบใน VHDL ประกอบด้วยส่วนกำหนดการเชื่อมต่อ(Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม(Architecture) ดังแสดงในรูปที่ 4.7 การบรรยายการเชื่อมต่อจะขึ้นต้นด้วยคำ ENTITY ตามด้วยชื่อขององค์ประกอบและคำ IS ภายในบรรยายถึงพอร์ตการติดต่อ อินพุท/เอาท์พุท พอร์ตขององค์ประกอบ ส่วนลักษณะกายภาพภายนอกอื่นๆ เช่น เวลา อุณหภูมิ ก็สามารถรวมเข้าไปในส่วนนี้ได้เช่นกัน ในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้บรรยายหน้าที่การทำงานขององค์ประกอบ หน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณ อินพุท/เอาท์พุท และพารามิเตอร์อื่นๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูปที่ 4.7 การบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังคำว่า BEGIN เป็นต้นไป

```
ENTITY component_name IS
        Input and output ports.
        Physical and other parameters.
END component_name;
```

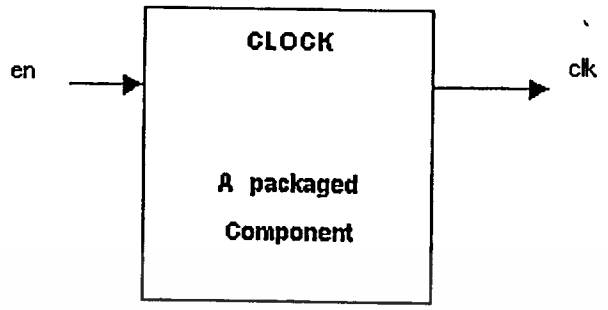
```
ARCHITECTURE identifier OF compionent_name IS
Declaratuons.
BEGIN
        Specification of functionality of the component
        In terms of its input lines and as influenced
        By physical and other parameters.
END identifier;
```

รูปที่ 4.7 แสดงการกำหนดการเชื่อมต่อและสถาปัตยกรรม

4.4.1 การกำหนดการเชื่อมต่อ (Interface Description)

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ ในระดับนี้จะต้องกำหนดพอร์ต สำหรับการติดต่อกับองค์ประกอบภายนอกอื่นๆ ดังตัวอย่างในรูปที่ 4.8 บรรทัดแรกเป็นการกำหนดชื่อขององค์ประกอบซึ่งกำหนดให้เป็นชื่อ clock component ตามด้วยคำว่า PORT และชื่อ

ของพอร์ตอยู่ในวงเล็บ IN และ OUT กำหนดโหมดของสัญญาณเป็นอินพุทหรือเอาท์พุท BIT แสดงชนิดของข้อมูล




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

รูปที่ 4.8 แสดงบล็อคไดอะแกรมและการบรรยายการเชื่อมต่อของ clock component

4.4.2 การกำหนดรูปแบบการบรรยาย (Architecture Description)

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้การบรรยายสามารถกำหนดค่าของสัญญานเอาท์พุทในเทอมของ clock component ในรูปที่ 4.9 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมมี en เป็นอินพุทและมี clk เป็นเอาท์พุท PROCESS เป็นค่าเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม ภายในโพรเซสกำหนดให้ periodic เป็นตัวแปรที่มีค่าเริ่มต้นเป็น "0" ถ้าสัญญาน en มีค่าเป็น "1" ค่าของ periodic จะถูกคอมพลีเมนต์และส่งค่าให้กับ clk ซึ่งเป็นสัญญานเอาท์พุท คำสั่ง WAIT กำหนดให้สัญญานมีคาบเป็นเวลา 1 ไมโครวินาที

```
ARCHITECTURE behavioral OF clock_component IS
BEGIN
        PROCESS
                VARIABLE periodic : BIT := '0';
BEGIN
  IF en = '1' THEN
                periodic := NOT periodic;
  END IF;
                ck <= periodic;
                WAIT FOR 1US;
  END PROCESS;
END behavioral;
```

รูปที่ 4.9 แสดงการบรรยายเชิงพฤติกรรมของ Clock_component

4.4.3 โปรแกรมย่อย (Subprogram)

การใช้ฟังก์ชันและโพรซีเดอร์ใน VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไป ค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่น ถ้าเราให้ฟังก์ชั่นแทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรตรรกะจริงๆ ในขณะที่เราใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูลหรือในการคำนวนค่าหน่วงเวลา แล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์

รูปที่ 4.10 แสดงการใช้ฟังก์ชั่นโพรซีเดอร์เพื่อเปลี่ยนข้อมูลชนิด 8 บิต เป็นค่าจำนวนเต็ม รูปที่ 4.11 แสดงการใช้ฟังก์ชันโดยกำหนดให้ X เป็นตัวแปรชนิดบิตแทนการกระทำในสมการบูลีน

```
TYPE byte IS ARRAY (7 DOWN TO  ) OF BIT
...
PROCEDURE byte_to_integer (ib: IN byte; oi: OUT INTEGER) IS
        VARIABLE result : INTERGER :=0;
BEGIN
        FOR I IN 0 TO 7 LOOP
                IF ib(I) = "1" THEN
                        Result := result+2**I;
                END IF;
END LOOP;
Oi := result;
END byte_to_interger;
```

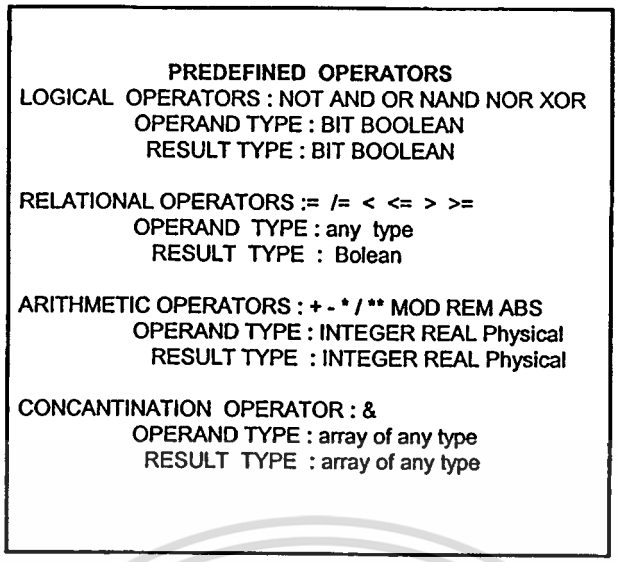
รูปที่ 4.10 แสดงการใช้โพรซีเดอร์

```
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 4.11 แสดงการใช้ฟังก์ชัน

4.4.4 โอเปอร์เรเตอร์ (VHDL OPERATORS)

การบรรยายเชิงพฤติกรรมใน VHDL ก็มีตัวกระทำทางลอจิกและคณิตศาสตร์เช่นเดียวกันกับภาษาซอฟแวร์ทั่วไปดังรูปที่ 4.12



รูปที่ 4.12 แสดงตัวกระทำใน VHDL

4.4.5 เวลาและความพร้อมเพรียง (Timing and Concurrency)

ในวงจรอิเล็กทรอนิกส์ อุปกรณ์ทุกตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ(always active) และจะมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยเสมอในทุกๆ เหตุการณ์ที่เกิดขึ้น VHDL เป็นภาษาที่ได้รับการออกแบบมาเพื่อสามารถบรรยายรูปแบบและการพ้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงาน ที่อยู่ภายในส่วนของสถาปัตยกรรมการบรรยายจะมีการทำงานที่พร้อมเพรียงกันเสมอหรือแม้แต่โพรเซสที่มีการทำงานภายในเป็นแบบลำดับคำสั่งก็ตาม หากมีหลายๆโพรเซสอยู่ภายในโครงการสร้างเดียวกันทุกๆ โพรเซสก็จะทำงานไปพร้อมๆกันด้วย

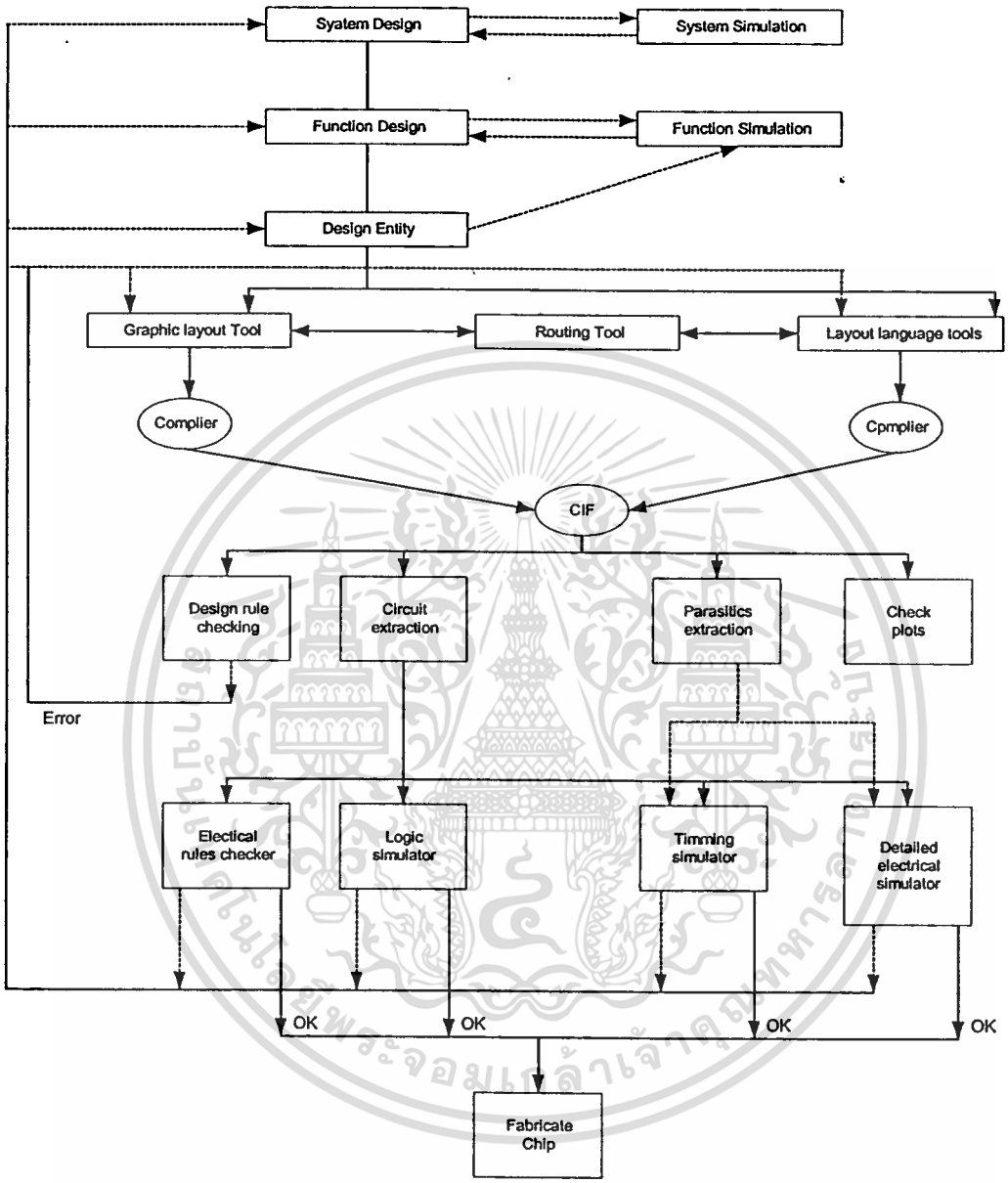
4.4.6 สัญญาณและตัวแปร (Signals and Variable)

สัญญาณที่เป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูลและมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วย การกำหนดค่าให้กันสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการส่งผ่านค่าของสัญญาณ เช่น W <= AFTER 12 ns หมายถึง กำหนดค่าของสัญญาณ a ให้กับ w หลังจากเวลาผ่านไป 12 ns.

4.5 โครงสร้างของวีเอชดีแอล

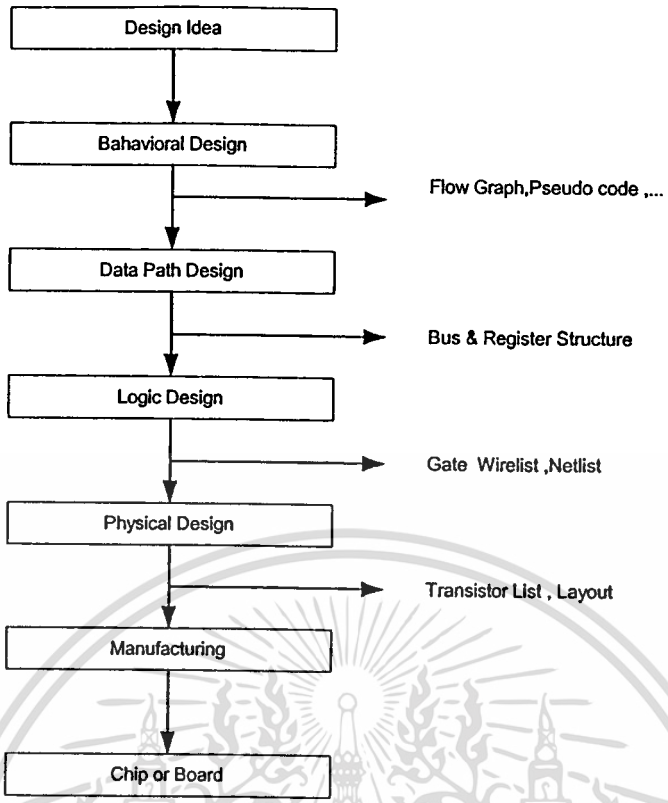
ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวความคิดเบื้องต้นจนกระทั่งได้ออกมาเป็น อุปกรณ์ฮาร์ดแวร์ที่ใช้งานได้ จะต้องผ่านขั้นตอนต่างๆ มากมายและในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์สุดท้ายในแต่ละขั้น ทำการเพิ่มเติมตามความจำเป็นและ

เข้ากระบวนการออกแบบในขั้นต่อไป โครงสร้างการออกแบบ VHDL สามารถเขียนเป็นแผนผังได้ดังรูปที่ 4.13



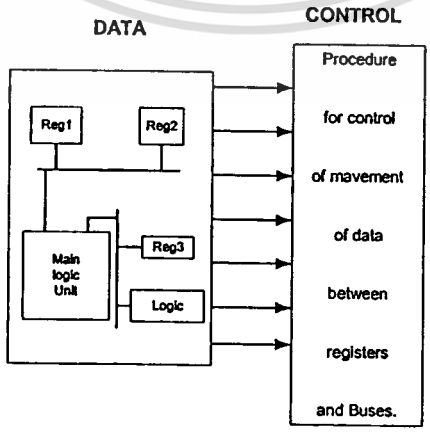
รูปที่ 4.13 รูปแสดงโครงสร้างการออกแบบวงจรรวมโดยใช้ VHDL

รูปที่ 4.14 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิตอลทั่วๆ ไป ซึ่งขั้นแรกผู้ออกแบบกำหนดแนวความคิดในการออกแบบเสียก่อนและทำการพัฒนาให้สามารถนำมาใช้ได้อย่างสมบูรณ์ ดังนั้นในขั้นตอนนี้จึงมีความจำเป็นที่ผู้ออกแบบจะต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบ ซึ่งอาจเป็นผังงาน ผังแสดงแบบหรือรหัสคำสั่งเทียม (Pseudo Code)



รูปที่ 4.14 แสดงขั้นตอนการออกแบบระบบดิจิตอล

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



รูปที่ 4.15 แสดงการออกแบบระบบเส้นทางของข้อมูล

การออกแบบวงจรตรรกะจะเป็นขั้นตอนต่อไป การออกแบบในขั้นนี้เกี่ยวข้องกับการใช้เกทพื้นฐานและฟลิปฟลอปเป็นส่วนของอุปกรณ์แยกต่างๆ ได้แก่ รีจีสเตอร์เก็บข้อมูลจงจรตรรกะและส่วนควบคุมฮาร์ดแวร์ ซึ่งในขั้นสุดท้ายจะได้ออกมาเป็นเครือข่ายของการโยงใยระหว่างเกทและ ฟลิปฟลอปนั่นเอง

ต่อมาเป็นขั้นตอนของการเปลี่ยนเครือข่ายการโยงใยในขั้นตอนที่แล้วให้เป็นทรานซิสเตอร์และเลย์เอาท์(Transistor list and layout) ในขั้นตอนนี้เกี่ยวข้องกับการจัดวางเกทและฟลิปฟลอปแทนด้วยทรานซิสเตอร์หรือไลบรารีเซล ขั้นตอนสุดท้ายเป็นการส่งระบบที่ออกแบบไว้ไปทำการเจือสารที่โรงงานเพื่อผลิตออกมาเป็นวงจรรวมในที่สุด

บทที่ 5
ทฤษฎี Cryptography

บทนำ

ในบทนี้ได้กล่าวถึงประวัติความเป็นมาและระบบคณิตศาสตร์ที่ทำให้เกิดความปลอดภัยของ Cryptography ซึ่งมี 3 ระบบด้วยกัน พร้อมทั้งแสดงการเปรียบเทียบระดับความปลอดภัยของการเข้ารหัสข้อมูลด้วยระบบคณิตศาสตร์ทั้งสามและกล่าวถึงการนำหลักการของ ElGamal Cryptography มาใช้ร่วมกับระบบคณิตศาสตร์แบบ Elliptic Curve พร้อมทั้งนำหลักการดังกล่าวไปจำลองการทำงานเป็นสมาร์ทการ์ด

5.1 ประวัติและความเป็นมาของการ Cryptography

ในปี ค.ศ. 1960 คอมพิวเตอร์และระบบการสื่อสารข้อมูลนั้นเป็นที่แพร่หลายเป็นอย่างมาก ซึ่งทำให้เกิดแนวความคิดที่ทำให้ข้อมูลข่าวสารนั้นเป็นข้อมูลข่าวสารส่วนตัวเกิดขึ้นนั้น หมายถึงระบบความปลอดภัยของข้อมูลต้องเกิดขึ้นด้วย ดังนั้นบริษัท IBM ซึ่งเป็นบริษัทยักษ์ใหญ่ที่ผลิตคอมพิวเตอร์และชิ้นส่วนคอมพิวเตอร์ได้สร้างทีมงานค้นคว้าและวิจัยจากแนวความคิดดังกล่าวขึ้น โดยเริ่มค้นคว้าและวิจัยตั้งแต่ปี ค.ศ.1970 และสำเร็จในปี ค.ศ.1977 จากนั้นบริษัท Federal Information Processing Standard ในสหรัฐอเมริกา ก็ได้นำระบบดังกล่าวมาใช้งานเรียกระบบนั้นว่า Data Encryption Standard(DES) อาจกล่าวได้ว่าเป็น Cryptography ระบบแรกที่เกิดขึ้น

เนื่องจาก DES เป็นระบบกุญแจเดี่ยว(Symmetric Key) ในปี ค.ศ. 1976 Diffe และ Hellman ก็ทำการสร้าง Public-Key อีกระบบหนึ่งขึ้นมาซึ่งเป็นแบบกุญแจคู่ (Asymmetric Key) ซึ่งเรียกว่า Public – Key Cryptography ซึ่งใช้คณิตศาสตร์พื้นฐานของ Discrete Logarithm Problem (DLP) เพื่อทำให้ระบบมีความปลอดภัยยิ่งขึ้น ในปี ค.ศ. 1978 Rivest Shamir และ Adleman ได้วิจัยและพัฒนาหลักการของ Public-Key Cryptography โดยเรียกว่า RSA ซึ่ง RSA นั้นใช้หลักการการแยกตัวประกอบของคณิตศาสตร์ เพื่อสร้าง Factoring Large Integer ต่อมาในปี ค.ศ.1985 ElGamal ได้พัฒนาหลักการของ Public–Key Cryptography ขึ้นมาอีก คือ ElGamal Cryptography โดยยังคงใช้คณิตศาสตร์พื้นฐานของ Discrete Logarithm Problem และต่อมาก็ได้มีการวิจัยและค้นคว้าหลักการของ Public-Key Cryptography ต่างๆ อีกมากมาย

ปัจจุบันนี้เป็นยุคของข้อมูลข่าวสาร การติดต่อสื่อสารต่างๆ ต้องการความปลอดภัยและเป็นส่วนตัว ดังนั้นจึงมีการใช้งาน Public-Key Cryptography ในเชิงธุรกิจและพาณิชย์ต่างๆ

มากมาย เช่น ระบบการเงินการธนาคาร ระบบการค้าทางอิเลคทรอนิกส์ ระบบโทรศัพท์ ระบบข้อมูลส่วนบุคคลแบบพกพา เป็นต้น

5.2 ทฤษฎีและหลักการของ CRYPTOGRAPHY

5.2.1 มาตรฐานการเข้ารหัสข้อมูล

การเข้ารหัสข้อมูล คือ การนำข้อมูลปกติที่อยู่ในรูปตัวอักษรธรรมดา มาผ่านการเข้ารหัสทำให้ได้ข้อมูลมาอีกในรูปแบบหนึ่ง ซึ่งไม่มีความหมาย อ่านไม่ออกและแปลกลับไปหาข้อมูลเดิมไม่ได้ หากไม่มีรหัสลับที่ถูกต้อง ดังนั้นถึงแม้จะมีใครแอบคัดลอกข้อมูลที่เข้ารหัสนี้ไปได้ แต่ไม่รู้รหัสลับก็ไม่สามารถใช้ประโยชน์จากข้อมูลนี้ได้ ดังนั้นการเข้ารหัสข้อมูลจึงเป็นมาตราการสำหรับการรักษาความปลอดภัยของข้อมูลที่เพิ่มขึ้นจากระบบความปลอดภัยของคอมพิวเตอร์ ซึ่งการเข้ารหัสข้อมูลจะแบ่งออกเป็น 2 ประเภท ระบบกุญแจเดี่ยว (Symmetric Encryption) และ ระบบกุญแจคู่ (Asymmetric Encryption)

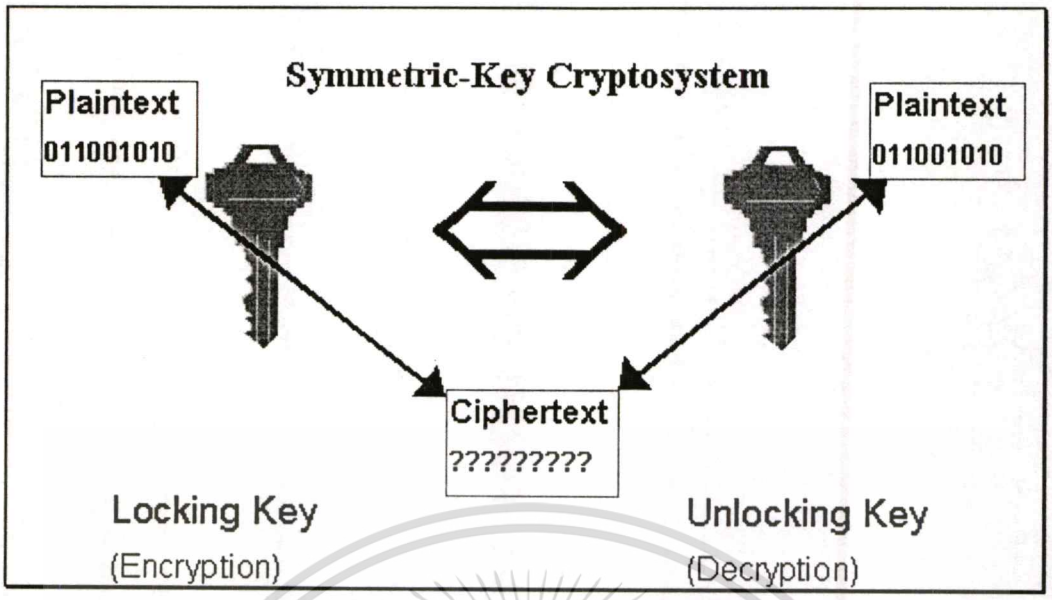
5.2.1.1 ระบบกุญแจเดี่ยว (Symmetric Encryption)

เป็นการเข้ารหัสข้อมูลขั้นพื้นฐาน โดยนำข้อมูลตัวอักษรและรหัสลับที่กำหนดขึ้นมาเข้ารหัสทางคณิตศาสตร์ เพื่อให้ได้ผลลัพธ์ออกมาเป็นข้อมูลที่เข้ารหัส ซึ่งสามารถทำให้อยู่ในรูปของฟังก์ชันทางคณิตศาสตร์ได้ดังนี้

ข้อมูลที่เข้ารหัสแล้ว = ฟังก์ชันการเข้ารหัส (ข้อมูลดิบ ,รหัสลับ)

และการถอดรหัสข้อมูลให้กลับคืนมาอยู่ในรูปเดิม ก็ต้องใช้รหัสลับตัวเดิมมาทำเป็นกระบวนการย้อนกลับกับการเข้ารหัส โดยมีฟังก์ชันทางคณิตศาสตร์เป็นดังนี้

ข้อมูลดิบ = ฟังก์ชันการถอดรหัส (ข้อมูลที่เข้ารหัสแล้ว,รหัสลับ)



รูปที่ 5.1 แสดงการเข้ารหัสและการถอดรหัสของ Symmetric Encryption

ถึงแม้ว่าฟังก์ชันการเข้ารหัสและถอดรหัสจะมีกระบวนการที่ชัดเจน ซึ่งทราบดีว่ามีขั้นตอนรายละเอียดอย่างไร ก็จะถอดรหัสข้อมูลไม่ได้ ถ้าไม่ทราบรหัสลับที่ใช้ในการเข้ารหัสข้อมูลนั้นๆ ดังแสดงในรูปที่ 5.1 ข้อดีของ Symmetric Encryption คือ สามารถทำการเข้ารหัสและทำการถอดรหัสข้อมูลได้อย่างรวดเร็ว แต่มีข้อเสียคือ การเก็บรักษารหัสลับที่ใช้สำหรับการเข้ารหัสและการถอดรหัสนั้นทำได้ยาก โดยเฉพาะในระบบใหญ่ๆที่มีผู้ใช้งานเป็นจำนวนมาก หากมีใครรู้รหัสลับที่ใช้ก็จะสามารถถอดรหัสมาอ่านข้อความและแก้ไขข้อความนั้นได้โดยไม่มีใครรู้ นอกจากนี้การที่ผู้ส่งกับผู้รับใช้รหัสลับในการเข้ารหัสและถอดรหัสตัวเดียวกัน การที่ผู้ส่งจะแจ้งรหัสลับไปให้ผู้รับได้ทราบโดยที่ไม่ให้คนอื่นรู้นั้น เป็นสิ่งที่ลำบากที่สุดของการเข้ารหัสข้อมูลแบบ Symmetric Encryption โดยทั่วไปก็จะส่งรหัสลับกันทางไปรษณีย์หรือใช้วิธีโทรศัพท์แจ้งให้ผู้รับทราบ จะเห็นได้ว่าแต่ละวิธีนั้นก็มีความเสี่ยงในการรั่วไหลของรหัสลับได้ทั้งนั้น

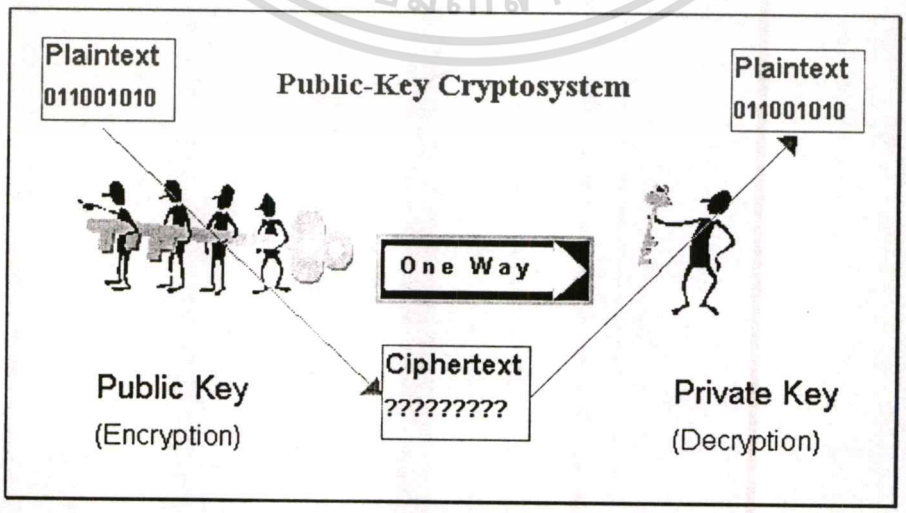
การเข้ารหัสข้อมูลแบบ Symmetric Encryption มีชื่อเรียกอีกอย่างหนึ่งว่า "Secret Key Encryption" คือ รหัสลับที่ใช้ในการเข้ารหัสและถอดรหัสนั้นเป็นรหัสลับตัวเดียวกัน โดยที่ผู้ส่งและผู้รับจะต้องทราบรหัสลับนี้ร่วมกัน จึงสามารถรับส่งข้อมูลที่เข้ารหัสนี้ได้อย่างถูกต้อง ซึ่งการเข้ารหัสข้อมูลแบบนี้ สามารถทำได้ทีละกลุ่มเรียกว่า Block Cipher คือจะอ่านข้อมูลดิบออกมาทีละกลุ่ม โดยมีความยาวของข้อมูลที่คงที่แล้วทำการเข้ารหัสของข้อมูลกลุ่มนั้น เมื่อได้ผลลัพธ์ออกมาเป็นข้อมูลที่ทำการเข้ารหัสแล้วและความยาวคงที่เช่นเดียวกัน ส่วนอีกแบบหนึ่งเป็นการเข้ารหัสข้อมูลแบบต่อเนื่องหรือ Stream Cipher จะอ่านข้อมูลดิบเข้ามาและทำการเข้ารหัสข้อมูลทีละบิตต่อเนื่องกันไป โดยจะทำการเข้ารหัสข้อมูลไปเรื่อยๆจนกว่าจะจบข้อมูล การเข้ารหัสข้อมูลแบบ

Symmetric Encryption ที่นิยมใช้กันนั้นมีอยู่หลายวิธี เช่น Data Encryption Standard(DES) International Data Encryption Algorithm(IDEA) RC2 และ RC4 เป็นต้น

5.2.1.2 ระบบกุญแจคู่ (Asymmetric Encryption)

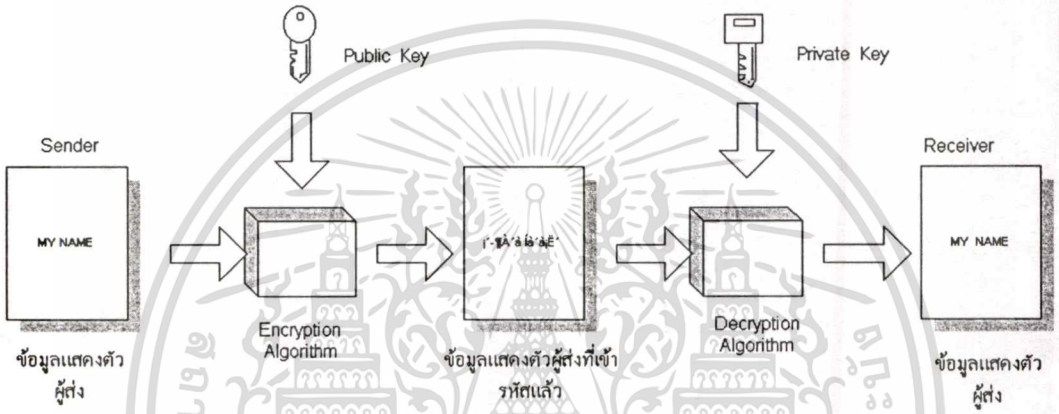
Asymmetric Encryption ถูกคิดค้นขึ้นในปี ค.ศ. 1976 โดย Whitfield Diffe และ Martin Hellman เพื่อทำการแก้ไขการเข้ารหัสแบบ Symmetric Encryption ที่ใช้รหัสลับตัวเดียวกันในการเข้ารหัสและถอดรหัสข้อมูล โดย Asymmetric Encryption เป็นการเข้ารหัสข้อมูลโดยมีรหัสที่เกี่ยวข้องอยู่ 2 รหัสด้วยกันคือ Public และ Key ซึ่งรหัสนี้ต้องเข้าคู่กันเสมอ โดยจับคู่เป็นคู่ของใครของมัน การเข้ารหัสข้อมูลทำได้โดยนำข้อมูลดิบมาเข้ารหัสโดยใช้ Public เป็นรหัสลับ เมื่อได้ข้อมูลที่เข้ารหัสแล้วผู้รับสามารถถอดรหัสโดยใช้ Key ซึ่งไม่สามารถใช้ Public-Key ของคู่อื่นมาถอดรหัสเนื่องจากคู่ของ Public-Key แต่ละคู่นั้นเกิดจากความสัมพันธ์ทางคณิตศาสตร์ จึงทำให้คู่ของ Public-Key แต่ละคู่ไม่ซ้ำกับคู่ของ Public-Key อื่น ดังนั้นในการเข้ารหัสข้อมูลผู้รับสามารถเปิดเผย Public ให้ผู้ส่งทราบ เพื่อนำไปใช้ในการเข้ารหัสข้อมูล

สิ่งที่น่าสนใจของ Asymmetric Encryption คือ Public-Key และ Private-Key สามารถใช้งานสลับกันได้ หมายถึง ถ้ามีรหัสลับ Public-Key และ Private-Key หนึ่งคู่เราอาจใช้ Private-Key เป็นรหัสลับในการเข้ารหัสข้อมูลและใช้ Public-Key มาทำการถอดรหัสข้อมูลได้ และคุณสมบัติอีกอย่างหนึ่งของ Asymmetric Encryption คือ แม้ว่าจะทราบ Public-Key ที่ใช้ในการเข้ารหัสข้อมูล แต่ก็เป็นการยากที่จะนำ Public นั้นมาคำนวณย้อนกลับไปหา Private-Key ที่เป็นคู่ของ Public-Key ดังนั้นแม้ว่าจะทราบค่า Public และมีข้อมูลที่เข้ารหัสแล้วอยู่ แต่ก็ไม่สามารถ ย้อนกลับไปคำนวณหา Private-Key และไม่สามารถถอดรหัสนั้นได้ ในการใช้งานจึงสามารถเปิดเผย Public-Key ที่ใช้สำหรับเข้ารหัสได้แก่ผู้ส่งและมีเพียงผู้รับข้อมูลเท่านั้นที่ทราบ Private-Key เพื่อใช้ในการถอดรหัสข้อมูลดังรูปที่ 5.2



รูปที่ 5.2 แสดงการเข้ารหัสของ Asymmetric Encryption

เนื่องจากคุณสมบัติที่สามารถสลับรหัสลับที่ใช้ในการเข้ารหัส คือ ใช้ Private เป็นรหัสลับในการเข้ารหัสข้อมูล และใช้ Key เป็นตัวถอดรหัส จึงทำให้สามารถทราบได้ทันทีว่าใครเป็นผู้ส่งข้อมูลนี้มา เนื่องจากมีเพียงผู้ส่งเท่านั้นที่จะทราบ Key ของตัวเอง ดังนั้นหากได้รับข้อมูลจากใครคนหนึ่ง แล้วใช้ Public ของเขาถอดรหัสข้อมูลออกมาได้ ก็แสดงว่าผู้ที่ส่งข้อมูลมาให้เรานั้นก็คือ เจ้าของ Key นั่นเอง ซึ่งคนอื่นๆที่รู้ Public ก็สามารถถอดรหัสข้อมูลนี้ได้เช่นกัน แต่จะไม่ใช่ผู้เข้ารหัสข้อมูลนี้อย่างแน่นอน และโดยการกำหนดตัวผู้ส่งข้อมูลได้อย่างแน่นอนนี้เองจึงเป็นหลักการใช้งานของ Digital Signature นั่นเองดังรูปที่ 5.3



รูปที่ 5.3 แสดงการทำงานของ Digital Signature Algorithm (DSA)

ข้อเสียของการใช้ Asymmetric Encryption คือ การเข้ารหัสและการถอดรหัสจะยาวนานกว่าการเข้ารหัสแบบ Symmetric Encryption เนื่องจากมีความซับซ้อนสูงกว่า และหากเลือกใช้รหัสลับที่มีความยาวของรหัสลับน้อยเกินไป อาจมีผู้นำ Public และข้อมูลที่เข้ารหัสแล้วไปคำนวณย้อนกลับหา Key ได้ การเข้ารหัสข้อมูลแบบ Asymmetric Encryption ที่ใช้กันอย่างแพร่หลาย ได้แก่ RSA Digital Signature Standard(DSS), Secure Electronic Transaction(SET) เป็นต้น

5.2.2 อัลกอริธึมของ RSA Cryptography

เป็นอัลกอริธึมของ Public-Key Cryptography แบบ Asymmetric ที่ถูกวิจัยและพัฒนาขึ้นในปี ค.ศ.1978 โดย Rivest Shamir และ Adleman โดยการนำเอาคณิตศาสตร์ในรูปของ Integer Factorization Problem (IFP) มาประยุกต์ใช้งานซึ่งเมื่อมีการวิจัยและพัฒนาสำเร็จจึงเรียกชื่อว่า RSA Public-Key Cryptography ซึ่งสามารถแสดงอัลกอริธึมของ RSA Cryptography ได้ดังต่อไปนี้

อัลกอริธึมการสร้าง Key ของ RSA (Key Generation for RSA)

สามารถสร้าง Public-Key ของ RSA Cryptography สามารถทำได้ดังต่อไปนี้

1. ทำการเลือกค่าของจำนวนเฉพาะที่มีขนาดใหญ่มากๆ (Prime Number) คือ p และ q
2. คำนวณค่าของ n โดยที่ $n = pxq$ และ $\phi = (p-1)(q-1)$
3. ทำการสุ่มของจำนวนเต็ม e, $1 < e < \phi$ โดยที่ $\gcd(e,\phi) = 1$
4. ใช้อัลกอริธึมของ Euclidean ทำการหาค่าของ d โดย $1 < d < \phi$ ซึ่ง $ed \equiv 1(\mathrm{mod}\,\phi)$
5. ได้ Public มีค่าเท่ากับ (n,e) และ Key เท่ากับ d

การเข้ารหัสและการถอดรหัสของ RSA (RSA Public-Key Encryption)

ผู้ส่งทำการเข้ารหัส (Encryption) ข้อมูล (m) เพื่อส่งไปยังผู้รับโดยผู้รับจะทำการถอดรหัส (Decryption)

1. การเข้ารหัส (Encryption) ซึ่งผู้ส่งทำการเข้ารหัสดังนี้
   - นำ Public-Key ของผู้ส่งคือ (n,e)
   - ทำการตรวจสอบข้อมูล (m) ให้อยู่ในช่วงของ $[0,n-1]$
   - คำนวณ $c = m^e \bmod n$
   - ทำการส่งข้อมูลที่มีการเข้ารหัส (Cliphertext) c ไปยังผู้รับ
2. การถอดรหัส (Decryption) ทำการคำนวณหาข้อมูล (m) จาก Key (d) ซึ่งผู้รับทำดังนี้
   - ใช้ Key d เพื่อคำนวณข้อมูล (m) ดังนี้ $m = c^d \bmod n$

<u>ตัวอย่าง 5.1</u> เมื่อผู้รับทำการเลือกจำนวนเฉพาะ $p = 2357, q = 2551$ ทำการคำนวณค่าของ $n = pq = 6012707$ และ $\phi = (p-1)(q-1) = 6007800$ ทำการเลือกค่าของ $e = 3674911$ และใช้อัลกอริธึมของ Euclidean หาค่าของ $d = 422191$ ซึ่ง $ed \equiv 1(\mathrm{mod}\,\phi)$ คู่ของ Public ของผู้ส่งเท่ากับ $(n = 6012707, e = 3674911)$ และ Private ของผู้รับเป็น $d = 422191$

<u>การเข้ารหัส (Encryption)</u> ข้อมูล (m) มีค่าเท่ากับ 5234673 ผู้ส่งทำการเข้ารหัสข้อมูลโดยใช้ Exponentiation สามารถคำนวณได้ดังนี้

$$\begin{aligned} c &= m^e \bmod n \\ &= 5234673^{3674911} \bmod 6012707 \\ &= 3650502 \end{aligned}$$

ซึ่งผู้ส่งทำการส่งข้อมูล 3650502 ไปยังผู้ใช้รับ

<u>การถอดรหัส (Decryption)</u> ผู้รับทำการคำนวณหาค่าของ c ซึ่งสามารถคำนวณได้ดังนี้

$$m = c^d \bmod n$$
$$= 3650502^{422191} \bmod 6012707$$
$$= 5234673$$

ซึ่งผู้รับได้รับข้อมูล m เท่ากับ 5234673

5.2.3 อัลกอริธึมของ ElGamal Cryptography (ElGamal Cryptography Algorithm)

ElGamal Public-Key Cryptography จะทำการสร้าง Public-Key และ Private-Key ในแบบ Asymmetric ที่ถูกวิจัยและพัฒนาขึ้นในปี ค.ศ. 1985 โดย ElGamal โดยการนำคณิตศาสตร์ในรูปของ Discrete Logarithm Problem(DLP) มาเป็นคณิตศาสตร์พื้นฐานสำหรับการเข้ารหัส (Encryption) และถอดรหัส (Decryption) ซึ่งมีนิยามดังนี้

อัลกอริธึมการสร้าง Key ของ ElGamal Public-Key Cryptography

(Key Generation for ElGamal Public-Key Cryptography)

ในการสร้าง Private และ Key ของ ElGamal Public-Key Cryptography สามารถทำได้ดังนี้

1. ทำการเลือกค่าของจำนวนเฉพาะ p ที่มีขนาดใหญ่มากๆ (Prime Number) และเลือกค่าของ α โดยที่ α เป็นสมาชิกที่อยู่ในเซตของจำนวนเต็มโมดูโล p (Z_p)
2. เลือกค่าของจำนวนเต็ม $a, 1 \le a \le p-2$ และคำนวณ $\beta = \alpha^a \bmod p$
3. ผู้ส่งสามารถใช้ Public คือ (p, α, β) และ Key เท่ากับ a

การเข้ารหัสและการถอดรหัสของ ElGamal (ElGamal Public-Key Encryption)

ผู้ส่งทำการเข้ารหัส (Encryption) ข้อมูล (m) เพื่อส่งไปยังผู้รับโดยผู้รับทำการถอดรหัส (Decryption)

1. การเข้ารหัส (Encryption) ซึ่งผู้ส่งสามารถทำการเข้ารหัสดังนี้
   - ใช้ Public ของผู้รับคือ (p, α, β)
   - ทำการตรวจสอบข้อมูล (m) ให้อยู่ในช่วงของ $[0,1,.., p-1]$
   - ทำการเลือกค่าของจำนวนเต็ม k โดยที่ $1 \le k \le p-2$
   - คำนวณค่าของ $\gamma = \alpha^k \bmod p, \delta = m \cdot (\alpha^a)^k \bmod p$
   - ทำการส่งข้อมูลที่มีการเข้ารหัส (Cliphertext) $c = (\gamma, \delta)$ ไปยังผู้รับ
2. การถอดรหัส (Decryption) ทำการคำนวณหาข้อมูล (m) จาก d ซึ่งผู้รับทำดังนี้
   - ใช้ Key a เพื่อหาค่า γ^{-a} ดังนี้ $\gamma^{p-1-a} \bmod p$

- คำนวณข้อมูล (m) ดังนี้ $\gamma^{-a} \cdot \delta \bmod p$

<u>ตัวอย่าง 5.2</u> เมื่อผู้รับเลือกจำนวนเฉพาะ $p = 2357$ $\alpha = 2$ ซึ่งเป็นสมาชิกของ Z_{2357} ผู้รับเลือกค่าของ Key ที่มีค่า $a = 1751$ และคำนวณค่าดังนี้

$$\beta = \alpha^{a} \bmod p = 2^{1751} \bmod 2357$$

ได้ค่าของ Public สำหรับผู้ส่งเป็น $p = 2357, \alpha = 2, \beta = 1185$

<u>การเข้ารหัส (Encryption)</u> ข้อมูล (m) มีค่าเท่ากับ 2035 ผู้ส่งทำการเลือกค่าของ K = 1520 และคำนวณได้ดังนี้

$$\gamma = 2^{1520} \bmod 2357 = 1430$$

และ

$$\delta = 2035 \cdot 1185^{1520} \bmod 2357 = 697$$

ผู้ใช้ส่งทำการส่งข้อมูล $\gamma = 1430, \delta = 697$ ไปยังผู้รับ

<u>การถอดรหัส (Decryption)</u> ผู้รับได้รับข้อมูล (γ, δ) พร้อมทำการคำนวณหาดังนี้

$$\gamma^{p-1-a} = 1430^{605} \bmod 2357$$
$$= 872$$

ผู้รับทำการคำนวณหาค่าของ m ซึ่งสามารถคำนวณได้ดังนี้

$$m = 872 \cdot 697 \bmod 2357$$
$$= 2035$$

ข้อมูลที่ผู้รับมีค่าเท่ากับ 2035

5.2.4 อัลกอริธึมของ Data Encryption Standard (DES)

DES เป็นมาตราฐานการเข้ารหัสข้อมูลชนิดหนึ่งของ Symmetric Encryption ที่คิดค้นโดยบริษัท IBM เมื่อปี ค.ศ.1977 ต่อมาได้ทำการปรับปรุงโดยกระทรวงกลาโหมของสหรัฐและกำหนดเป็นมาตรฐาน ANSI X3.92 และ X3.106 โดย DES ถูกออกแบบมาให้ทำงานโดยใช้ Hardware เป็น

ตัวเข้ารหัสและถอดรหัสหรือจะใช้ Software ทำงานลักษณะเดียวกันก็ได้ การทำงานของ DES จะเป็นแบบ Block Cipher คือ เข้ารหัสข้อมูลทีละกลุ่ม กลุ่มละ 64 บิต โดยใช้รหัสลับขนาด 56 บิตซึ่ง การเข้ารหัสข้อมูล DES นี้ได้ผ่านการทดสอบจนเป็นที่ยอมรับว่าเป็นการเข้ารหัสข้อมูลที่ปลอดภัยมากที่สุดชนิดหนึ่ง

การเข้ารหัสของ DES สามารถเข้ารหัสข้อมูลได้ 2 วิธี Electronic Code Block (ECB) ซึ่งเข้ารหัสข้อมูลทีละครั้ง ครั้งละ 64 บิต โดยใช้รหัสลับขนาด 56 บิต ดังนั้นข้อมูล 64 บิต ที่นำมาเข้ารหัสจะไม่เกี่ยวข้องกับข้อมูลส่วนอื่นๆที่เหลือเลย ส่วนอีกวิธีหนึ่งคือ Cipher Block Chaining (CBC) จะทำการเข้ารหัสข้อมูลครั้ง 64 บิตเช่นเดียวกัน และใช้รหัสลับขนาด 56 บิต ต่างกันตรงที่ว่าข้อมูลที่นำมาเข้ารหัสจะถูก Exclusive OR (XOR) กับข้อมูล 64 บิตก่อนหน้านั้นเสียก่อนแล้วจึงนำมาเข้ารหัส ทำให้ข้อมูล 64 บิตมีข้อความเหมือนกัน ถูกเข้ารหัสข้อมูลแล้วได้ข้อความไม่เหมือนกัน เป็นผลให้การเดาถอดรหัสข้อมูลนั้นยากยิ่งขึ้น

จากจำนวนรหัสลับขนาด 56 บิตนี้จะมีจำนวนรหัสที่เป็นไปได้ทั้งหมดประมาณ 72,000 ล้านล้านรหัส ซึ่งในการเข้ารหัสนั้นเราจะสุ่มรหัสใดรหัสหนึ่งมาเป็น Key สำหรับเข้ารหัสของ DES ทำให้การเดารหัสทำได้ยากมากและนำรหัสลับที่เลือกไว้มาประมวลผลการเข้ารหัสข้อมูลกลับไปกลับมาถึง 16 ครั้ง จึงถือได้ว่าการเข้ารหัสของ DES เป็นการเข้ารหัสที่มีความปลอดภัยสูง ยากแก่การเดาการถอดรหัส หากมีผู้ต้องการถอดรหัสก็จะต้องเดารหัสหลายหมื่นล้านล้านรหัสกว่าจะพบรหัสที่ถูกต้อง สำหรับงานเข้ารหัสข้อมูลที่ต้องการความปลอดภัยสูงจะใช้การเข้ารหัสข้อมูลที่เรียกว่า Triple – DES โดยมีหลักการทำงานคือ ครั้งที่ 1 ทำการเข้ารหัสข้อมูลตามปกติ และนำผลลัพธ์ที่ได้มาเข้ารหัสตามมาตรฐาน DES ครั้งที่ 2 ใช้รหัสลับลำดับที่ 2 ในการเข้ารหัส จากนั้นจึงนำผลลัพธ์ที่ได้มาเข้ารหัส DES อีกครั้งหนึ่ง ครั้งที่ 3 โดยใช้รหัสลับลำดับที่ 3 ในการเข้ารหัส ซึ่งรหัสลับทั้ง 3 ตัวนี้ไม่มีส่วนเกี่ยวข้องกันเลย ซึ่งทำให้ข้อมูลมีความปลอดภัยสูงขึ้นไปอีก ส่วนทางผู้รับก็จะถอดรหัสข้อมูลนี้ทีละขั้น ย้อนกลับไปจนได้ข้อมูลดิบออกมา

5.3 ระบบความปลอดภัยของ Cryptography

เนื่องจากระบบความปลอดภัยของข้อมูลขึ้นอยู่กับความซับซ้อนของระบบคณิตศาสตร์ที่ใช้สำหรับการเข้ารหัสข้อมูล ซึ่งระบบคณิตศาสตร์ที่ใช้กับ Cryptography แบ่งออกเป็น 3 ระบบด้วยกันคือ Integer Factorization System, Discrete Logarithm System และ Elliptic Curve System [ECC] ซึ่งสามารถอธิบายทั้ง 3 ระบบได้ดังนี้

5.3.1 Integer Factorization System

เป็นระบบคณิตศาสตร์ที่ใช้สำหรับเข้ารหัสข้อมูลที่ถูกคิดค้นโดยสถาบัน MIT ในสหรัฐอเมริกา และได้เรียกการเข้ารหัสข้อมูลโดยวิธีดังกล่าวว่า RSA ตามชื่อผู้คิดค้นคือ Ron Rivest, Adi Shamir และ Len Adleman

RSA นั้นเป็นวิธีการ Cryptography ที่รู้จักกันอย่างแพร่หลายซึ่งระดับความปลอดภัยนั้นจะขึ้นอยู่กับความยากง่ายในการแยกตัวประกอบของจำนวนเต็ม สามารถแสดงได้ดังนี้ ถ้าให้ p เป็นจำนวนเฉพาะซึ่งจะมี 1 และ p เท่านั้นที่หารลงตัวและกำหนดให้ n เป็นผลคูณของจำนวนเฉพาะที่มีค่าสูงมากๆ ดังนั้นสามารถแสดงการแยกตัวประกอบได้ดังนี้

$$n = p \times q \tag{5.1}$$

ด้วยวิธีการดังกล่าว RSA จะประกอบด้วยคู่ของ (n,e) ซึ่ง e เป็นจำนวนซึ่งอยู่ระหว่าง 1 และ n-1 โดยที่ n เป็นผลคูณของจำวนวนเฉพาะ ดังนั้นในการสร้างคู่ของ Public-Key ของ RSA จะต้องเลือกค่าของจำนวนเฉพาะ ให้มีขนาดใหญ่เพียงพอที่จะทำให้เกิดความปลอดภัย โดยทั่วไปแล้ว RSA จะใช้งานในช่วง 512-1024 บิต

5.3.2 Discrete Logarithm System (DL)

ระบบคณิตศาสตร์แบบนี้จะอยู่ในรูปของการโมดูโลจำนวนเฉพาะ p ซึ่งถูกกำหนดไว้คงที่ และจะมีค่าของ g ซึ่งเป็นจำนวนเต็มอยู่ในช่วง 0 ถึง p-1 และ y เป็นผลลัพธ์ที่เกิดจากการยกกำลังสองของ g แสดงได้ดังนี้

$$y = g^x \pmod{p} \tag{5.2}$$

เมื่อ x เป็นจำนวนเต็มซึ่งทำให้เกิดคู่ของ g และ y จะเห็นว่าระบบ Discrete Logarithm คล้ายกับระบบ Integer Factorization แต่จะต่างกันตรงที่ Discrete Logarithm นั้นจะมีการ modulo ด้วย p ต่อมา Taher ElGamal ได้นำระบบคณิตศาสตร์นี้มาทำการเข้ารหัสแบบ ElGamal ซึ่งใช้งานในรูปแบบของ Digital Signature Algorithm (DSA) และจำนวนเฉพาะ p ที่จะนำมาทำการโมดูโลนั้นจะต้องมากกว่า 760 บิต

5.3.3 Elliptic Curve System

ระบบคณิตศาสตร์แบบ Elliptic Curve จะทำการสร้างคู่ (x,y) ซึ่งเป็นจุดบน Elliptic Curve โดยที่คู่ลำดับนั้นจะถูกโมดูโลด้วย p สามารถแสดงเป็นสมการได้ดังนี้

$$y^2 = x^3 + ax + b \pmod{p} \tag{5.3}$$

โดยที่ค่าของ a,b เป็นค่าของจำนวนเต็มใดๆ ปกติแล้วสามารถสร้างคู่ลำดับ (x,y) ในรูปของคณิตศาสตร์สนามจำกัด(Finite Field) ได้คืออยู่ในรูปของ $GF(2^n)$ ดังนั้นจากความซับซ้อนในการสร้างคู่ (x,y) บน Elliptic Curve จะทำให้ค่าของจำนวนเฉพาะสามารถใช้งานได้ในช่วง 155-201 บิต

5.4 แสดงการเปรียบเทียบระบบ Cryptosystem

โดยทั่วไปสามารถเปรียบเทียบระบบของ Cryptography ได้ใน 2 ลักษณะคือ ความปลอดภัย และประสิทธิภาพ ซึ่งสามารถแสดงได้ดังนี้

5.4.1 ความปลอดภัย

ในหัวข้อนี้เน้นในส่วนของความปลอดภัยในทางทฤษฎี คือ การทำลายระบบ Public-Key โดยทั่วๆไป ซึ่งเมื่อมีการตรวจสอบระบบของ Public–Key Cryptography แล้วอัลกอริธึมที่ใช้สำหรับทดสอบความปลอดภัยของระบบ Integer Factorization และ Discrete Logarithm ใช้อัลกอริธึมของ Sub-exponential Time ซึ่งแสดงได้ดังนี้

$$O(\exp((c+o(1)))(\ln n)^{1/3}(\ln \ln n)^{2/3}) \tag{5.4}$$

เมื่อ c เป็นค่าคงที่ n เป็นขนาดของบิตที่ใช้งาน

ส่วนอังกอริธึมที่ใช้สำหรับทดสอบความปลอดภัยของระบบ Elliptic Curve ในรูปของ Exponential Time สามารถแสดงได้คือ

$$O(\sqrt{p}) \tag{5.5}$$

เมื่อ p เป็นตัวโมดูโลของระบบ

ในรูปที่ 5.4 แสดงการเปรียบเทียบเวลาที่ใช้ในการทำลายระบบ ECC, RSA และ DSA โดยเปลี่ยนแปลงตามขนาดของบิตที่ใช้งาน ซึ่งค่าในการคำนวณจะแสดงในรูป MIPS Years ซึ่งก็คือเวลาที่ใช้ในการคำนวณใน 1 ปี บนขีดความสามารถในการประมวลผลของของเครื่องคำนวณที่ 1 ล้านคำสั่งต่อ 1วินาที (One Million instructions per second) ค่าที่ยอมรับเป็นมาตรฐานโดยทั่วไปคือที่ 10^{12} MIPS years จากรูปเป็นเหตุผลที่ทำให้ RSA และ DSA ความใช้งานที่ 1024 บิต และ ECC ควรใช้งานที่ 160 บิตและสังเกตได้ว่าขนาดของบิตที่ใช้งานของ ECC จะมีขนาดเล็กกว่า RSA และ DSA เมื่อเพิ่มระดับความปลอดภัยให้สูงขึ้นจะทำให้ความแต่ต่างของขนาดของบิตเพิ่มมากขึ้น เช่น เมื่อ RSA และ DSA ใช้งานที่ 2000 บิตแต่ ECC จะใช้งานที่ 300 บิต ที่ระดับความปลอดภัยเดียวกัน[2]

ตามเอกสารของ Certicom ECC Challenge[2] ซึ่งใช้ CPU Pentium 100 ซึ่งมีความสามารถในการทำงานที่ 16,000 คำสั่งต่อ 1 วินาทีสำหรับคำนวณ ECC บน Finite Field $F_{2^{89}}$ ซึ่งจะใช้เวลาในการถอดรหัส 15,550 วันสำหรับคอมพิวเตอร์ 1 เครื่องทำงาน 24 ชั่วโมงและเมื่อนำคอมพิวเตอร์มาคำนวณแบบ Network จะใช้เวลาในการคำนวณประมาณ 5 วัน

ตารางที่ 5.1 แสดงกำลังที่ใช้สำหรับทำลายระบบ ECC โดยมใช้วิธีการของ Pollard rho-method ซึ่งเปลี่ยนแปลงตามค่าของ n [2]

Field size (in bits)	Size of n (in bits)	$\sqrt{\pi n}/2$	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}

ตารางที่ 5.2 แสดงกำลังที่ใช้สำหรับทำลายระบบ Integer Factorization [2]

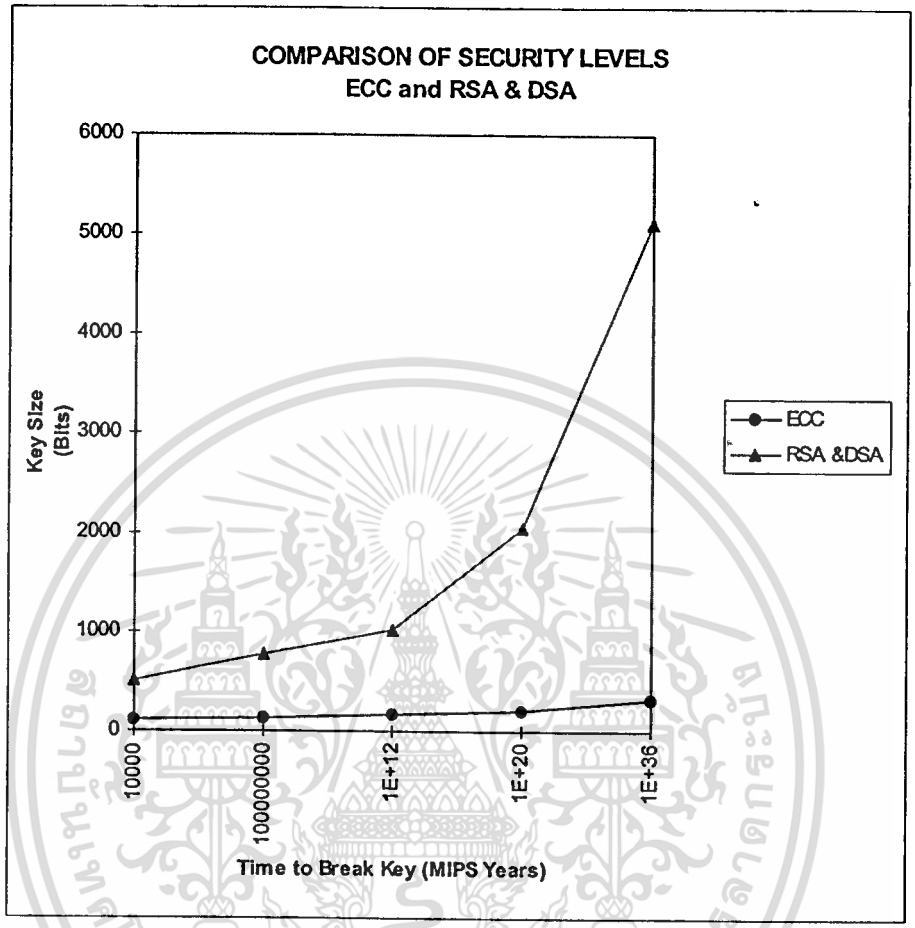
Size of integer To be factored (in bits)	MIPS years
512	3×10^{4}
768	2×10^{8}
1024	3×10^{11}

5.4.2 ประสิทธิภาพ

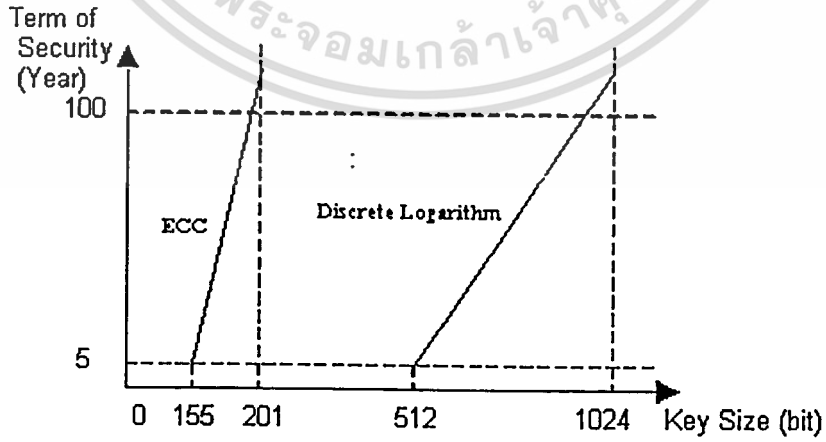
ในการทดสอบประสิทธิภาพของระบบ Public-Key Cryptography นั้นสามารถพิจารณาได้ชัดเจนใน 3 หัวข้อด้วยกันคือ

- Computational Overheads คือ จำนวนครั้งในการคำนวณ การเปลี่ยนแปลง Public Key และ Private Key
- Key Size คือ ขนาดของ Public Key และ Private Key ที่ใช้งาน
- Bandwidth คือ ขนาดของบิตข้อมูลที่ใช้ในการเข้ารหัส

ซึ่งในการเปรียบเทียบระบบนั้น จะทำการเปรียบเทียบที่ระดับความปลอดภัยมาตราฐานโดยทั่วไป ดังได้กล่าวไว้ในหัวข้อที่ 5.4.1คือที่ 160 บิตสำหรับ ECC กับ 1024 บิตสำหรับ RSA และ DSA



รูปที่ 5.4 แสดงการเปรียบเทียบระดับความปลอดภัยของระบบ[2]



รูปที่ 5.5 เปรียบเทียบประสิทธิภาพของอังกอริธึม[3]

5.4.2.1 Computation Overhead

ในแต่ละระบบนั้น สิ่งที่สำคัญก็คือการทำอย่างไรการคำนวณจะประหยัดที่สุด ใน RSA การ Exponent ของ Public Key และ Private Key ($e^{Public}, e^{Private}$) ที่มีขนาดเล็กนั้นจะทำให้จำนวนครั้งในการคำนวณน้อยลงซึ่งจะทำให้การเข้ารหัสข้อมูลเร็วขึ้น และเมื่อทำการเปลี่ยนแปลงฐานของ Finite Field (F_{2m}) เช่น ฐาน 10 เป็น ฐาน 2 ก็จะทำให้ส่วนคำนวณ (Arithmetic Module) เกิดการเปลี่ยนแปลง และความเร็วจะเปลี่ยนแปลงตามส่วนของการคำนวณ (Arithmetic) ด้วย

5.4.2.2 Key Size

ในตารางที่ 5.1 แสดงการเปรียบเทียบตัวแปรของระบบและคู่ของ Key ในแต่ละระบบ [2] ซึ่งประกอบไปด้วย ตัวแปรของระบบ (System parameter) คู่ของ Key ซึ่งใน ECC จะมีขนาดเล็กกว่า RSA และ DSA

ตารางที่ 5.3 แสดงขนาดและ Key ของตัวแปรในแต่ละระบบ [2]

	System parameters (bits)	Public key (bits)	Private key (bits)
RSA	n/a	1088	2048
DSA	2208	1024	160
ECC	481	161	160

5.4.2.3 Bandwidth

จากระบบคณิตศาสตร์ทั้ง 3 ที่กล่าวมา สามารถนำไปเข้ารหัสข้อมูลที่ยาวมากๆได้ แต่ในกรณีที่ข้อมูลสั้น แล้วการนำระบบคณิตศาสตร์บางระบบมาใช้งานอาจทำให้ไม่สามารถใช้งานได้ ดังนั้นระบบคณิตศาสตร์แต่ละระบบก็มีข้อมูลจำกัดในช่วงของการใช้งานซึ่งแสดงดังตารางที่ 5.4 และ 5.5 ซึ่งเปรียบเทียบความยาวของข้อมูลที่ใช้สำหรับการเข้ารหัสที่ 2000 บิตและ 100 บิต

ตารางที่ 5.4 แสดงความยาวของข้อมูลที่ใช้สำหรับ Signature (e.g. 2000-bit) [2]

	Signature size (bits)
RSA	1024
DSA	320
ECC	320

ตารางที่ 5.5 แสดงความยาวของข้อมูลที่ใช้สำหรับการเข้ารหัส (encryption 100-bit) [2]

	Encrypted message (bits)
RSA	1024
ElGamal	2048
ECC	321

จากการพิจารณา Bandwidth ของระบบจะเห็นว่า ECC จะสามารถใช้งานใน Bandwidth ที่มีขนาดแคบได้ดังนั้นเมื่อใช้กับ ข้อมูลที่สั้นๆ ECC ก็สามารถใช้งานได้ดี จากที่กล่าวมาระบบคณิตศาสตร์ทั้ง 3 ที่ใช้สำหรับ Cryptography ECC นั้นจะมีประสิทธิภาพสูงสุดทั้งในส่วนของ computational Overhead, Key Size และ Bandwidth ซึ่งเมื่อนำไปประยุกต์ใช้งานจะทำให้ ระบบมีความเร็วสูง กำลังในการทำงานต่ำ และทำให้ขนาดของ Code เล็กลง

5.5 อัลกอริธึมของ Elliptic Curve

Victor Miller และ Neal Koblite [4] ได้นำเอาวิธีการ Elliptic Curve Algorithm มาประยุกต์ใช้งานร่วมกับอัลกอริธึมของ Public-Key Cryptography อื่นๆ เช่นใช้งานร่วมกับ ElGamal Public-key Cryptography ซึ่งจะเรียกเป็น ElGamal Elliptic Curve หรือ ใช้งานร่วมกับ Menezs-Venstone Public-key Cryptography ซึ่งเรียกเป็น Menezs-Venstone Elliptic Curve เพื่อที่จะช่วยลดขนาดของ Public และ Private Key ที่ใช้เก็บบนหน่วยความจำนั้นเล็กลงนั่นเอง โดยทั่วไปแล้ว Elliptic Curve แบ่งออกได้ 2 ชนิดคือ Supersingular Curve และ Nonsupersigular Curve แสดงเป็นนิยามได้ดังนี้

นิยาม 5.1 ถ้า Z_p เป็นเซตของคณิตศาสตร์สนามจำกัด $\alpha > 2$ (GF (α))บนคณิตศาสตร์สนามจำกัด(Finite Field) สมการของ Supersingular Curve สามารถเขียนได้ดังนี้

$$y^2 = x^3 + ax + b \tag{5.6}$$

เมื่อ $a, b \in Z_p, b \neq 0$

<u>นิยาม 5.2</u> ถ้า Z_p เป็นเซตของคณิตศาสตร์สนามจำกัด $\alpha = 2$ (GF(α))บนคณิตศาสตร์สนามจำกัด(Finite Field) สมการของ NonSupersingular Curve สามารถเขียนได้ดังนี้

$$y^2 + xy = x^3 + ax^2 + b \tag{5.7}$$

เมื่อ $a, b \in Z_p, b \neq 0$

จากนิยามข้างต้นเราสามารถบวกจุดเหล่านี้แบบ abelian group

1) เราสามารถนิยามการบวกของ Elliptic Curve E ได้ดังนี้

<u>นิยาม</u> ถ้า $P = (x_1, y_1) \in E$ เป็นจุดบนสมการ (5.6) แล้ว $-P = (x_1, y_1 + a)$, $P + O = O + P = P$ สำหรับทุกค่าของ $P \in E$ ถ้า $Q(x_2, y_2) \in E$ และ $Q = -P$ แล้ว $P + Q = (x_3, y_3)$ เมื่อ

$$x_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2} \right)^2 + x_1 + x_2 & P \neq Q \\ \dfrac{x_1^4 + b^2}{a^2} & P = Q \end{cases} \tag{5.8}$$

และ

$$y_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2} \right)(x_1 + x_3) + y_1 + a & P \neq Q \\ \dfrac{x_1^2 + b}{a}(x_1 + x_3) + y_1 + a & P = Q \end{cases} \tag{5.9}$$

2) เราสามารถนิยามการบวกของ Elliptic Curve E ได้ดังนี้

<u>นิยาม</u> $P = (x_1, y_1) \in E$ เป็นจุดบนสมการที่ (5.7) แล้ว $-P = (x_1, y_1 + x_1)$ สำหรับทุกค่าของ $P \in E, O + P = P + O = P$ ถ้า $Q = (x_2, y_2) \in E$ และ $Q \neq -P$ แล้ว $P + Q = (x_3, y_3)$ เมื่อ

$$x_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)^2 + \dfrac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 & P \neq Q \\ \dfrac{x_1^2 + b}{a^2} & P = Q \end{cases} \quad (5.10)$$

และ

$$y_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + y_1 + x_3 & P \neq Q \\ x_1^2 + \left(x_1 + \dfrac{y_1}{x_1}\right)x_3 + x_3 & P = Q \end{cases} \quad (5.11)$$

สังเกตว่า การคำนวณในการบวกของจุดบน E ต้องการ Operation การคูณ การบวก และอินเวอร์ส ภายใต้คณิตศาสตร์สนามจำกัด F_{2^m}

5.6 การเลือก Elliptic Curve

การเลือก ECC จะเป็นการเลือกตามมาตราฐาน IEEE P1363

ตารางที่ 5.6 แสดงการทดสอบตามมาตราฐาน IEEE ของ Testing random number

K	T
160	34
161-163	33
164-166	32
167-169	31
170-173	30
174-177	29
178-181	28
182-185	27

เมื่อ K คือขนาดของบิตของ Public-Key และ T คือ เวลาตามการทดสอบของ Miller-Rabin [2]

5.7 การประยุกต์ใช้งาน ElGamal Elliptic Curve Public-Key Cryptography

5.7.1 NonSupersingular Curve

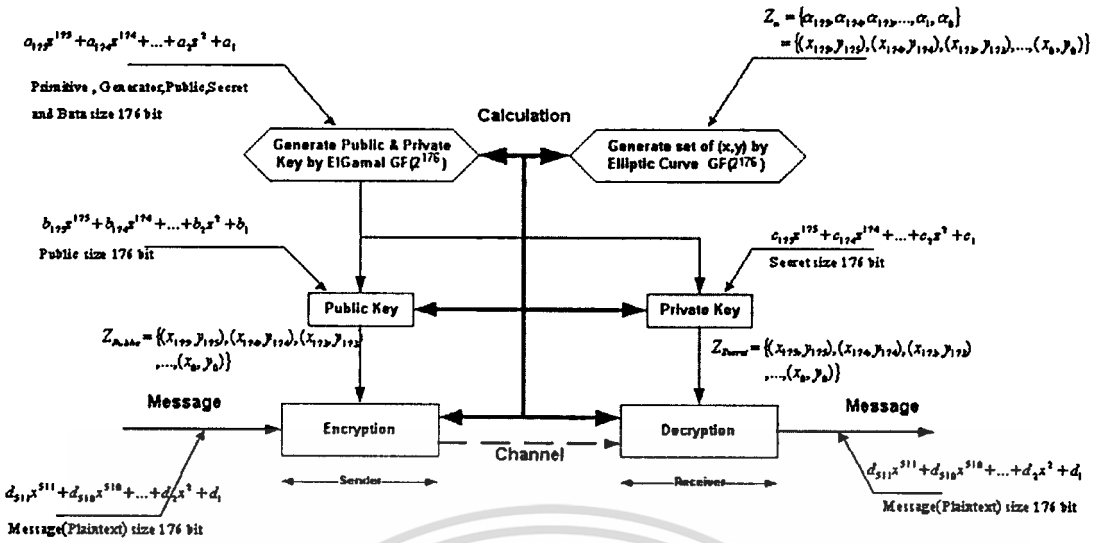
การประยุกต์ใช้งานสมการ Nonsupersingular และ Supersingular ของ ECC นั้น สามารถประยุกต์ในด้าน Hardware หรือ Software ก็ได้ [3] โดยในวิทยานิพนธ์ฉบับนี้ได้ประยุกต์ใช้งานในด้าน Hardware ซึ่งใช้ ECC แบบ NonSupersingular ดังนั้นสามารถแสดงสมการแบบ NonSupersingular ดังสมการที่ 5.7

5.7.2 ElGamal Elliptic Curve Encryption

ที่ผ่านมาได้อธิบายหลักการของ ElGamal และ NonSupersingular Curve ของ Elliptic Curve แล้ว ซึ่งถ้า $E(GF(2^n))$ เป็นจุดของ ECC แบบ NonSupersingular และ $P \in E(GF(2^n))$ ดังนั้นผู้ใช้จะทำการเลือกค่า a ซึ่งใช้คำนวณค่าของ Private Key ได้ดังนี้ $R = aP$ ผู้ส่ง(Sender) จะส่งคู่ของข้อมูล (Plain text) M_1, M_2 แล้วทำการเลือกค่าของจำนวนเต็ม k และนำจุด P ที่ถูกกำหนดไว้มาทำการคำนวณ $Q = kP$

ทั้ง ElGamal และ Elliptic Curve ต่างก็ใช้คณิตศาสตร์ Discrete Logarithm Problem และสร้างเซตบนคณิตศาสตร์สนามจำกัด (GF(2)) ได้ แต่เมื่อทำการเปรียบเทียบขนาดของ Public-Key ที่จะนำมาใช้งานแล้วนั้น ถ้าใช้ ElGamal เพียงอย่างเดียวขนาดของ Public-Key ที่ใช้งานจะอยู่ในช่วง 512-1024 บิต ส่วนElliptic Curve ที่ใช้งานจะอยู่ในช่วง 155-201 บิต [2] ดังนั้นเมื่อนำมาใช้งานร่วมกันจึงเรียกใหม่ได้เป็น ElGamal Elliptic Curve Public-Key Cryptography ซึ่งขนาดของ Public-Key ที่ใช้งานจะอยู่ในช่วง 155-201 บิต

แต่ในงานวิจัยนี้เลือกขนาด Public-Key ที่ 176 บิตคือ GF(2^{176}) ซึ่งอยู่ในช่วงที่สามารถใช้งานได้ตามมาตรฐาน IEEE P1363 และใช้สมการแบบ NonSupersingular Curve ในการสร้างคู่ลำดับของ Elliptic Curve



รูปที่ 5.6 แสดงผังการทำงานของ ElGamal Elliptic Curve Cryptography Algorithm

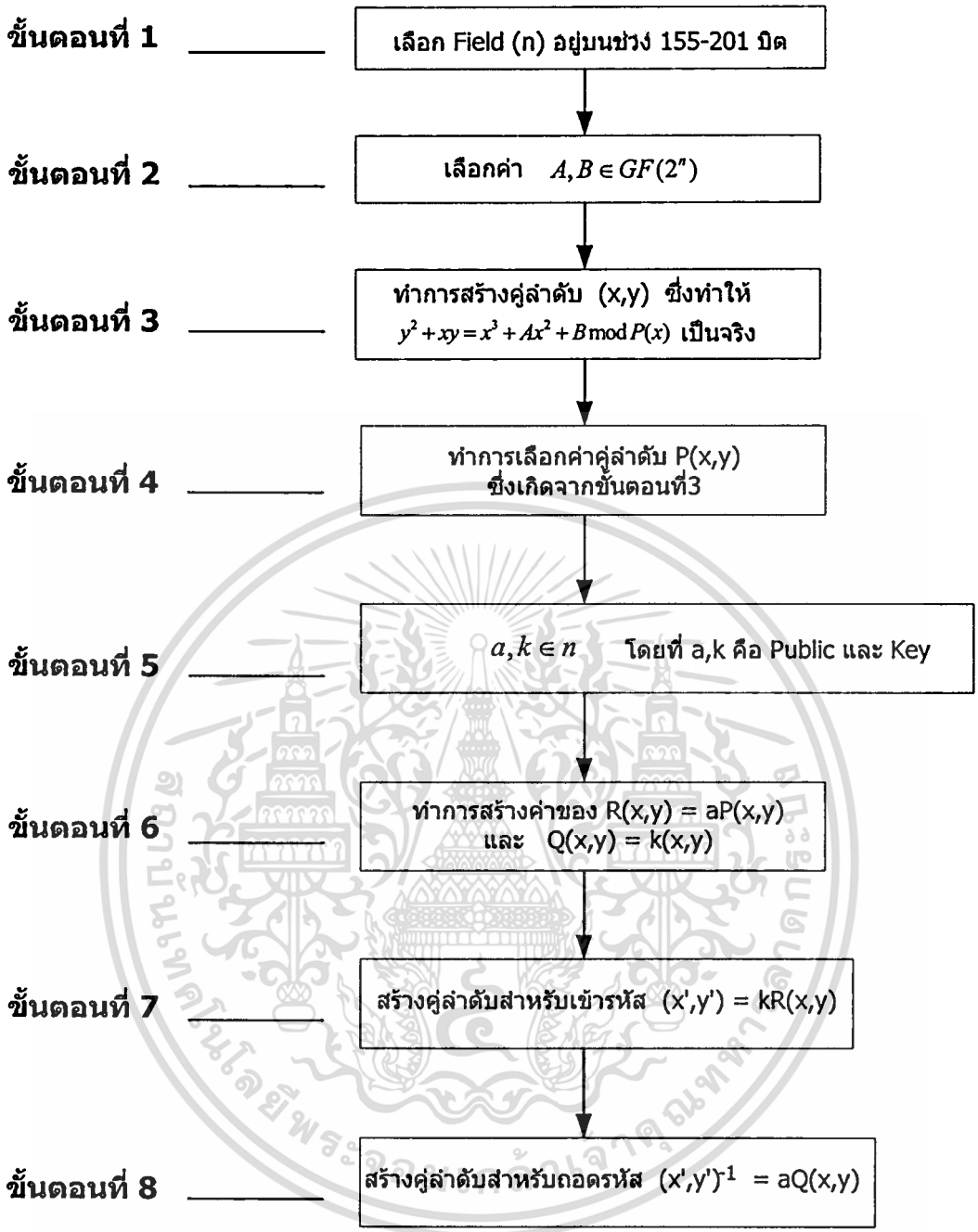
ตารางที่ 5.7 แสดงขนาดและ Key ของตัวแปรของ ElGamal และ ElGamal ECC

	Size of (n)	Public key (bits)	Private key (bits)
ElGamal	1024	1024	1024
ElGamal ECC	176	176 x 2	176 x 2

5.8 ผลการทดลองการสร้างคู่ลำดับ (X,Y) บน Elliptic Curve

5.8.1 การสร้างคู่ (X,Y) บน ECC

สามารถแสดงขั้นตอนการสร้างคู่ลำดับได้ดังรูปที่ 5.7 พร้อมทั้งแสดงการสร้างเซตคู่ลำดับ (X,Y) ของ Elliptic Curve บน GF(2^{176}) เพื่อใช้สำหรับการออกแบบวงจรทางด้าน Hardware บน FPGA ในบทที่ 7 แสดงดังนี้



รูปที่ 5.7 แสดงขั้นตอนการสร้างคู่ลำดับ (x , y) สำหรับการเข้ารหัสและถอดรหัส

ตัวอย่างที่ 5.1

Field (n) = 176 **ขั้นตอนที่ 1**

Elliptic Curve Public - Key Cryptography on GF(2^176):

y^2+xy = x^3+Ax^2+B และ $p(x) = x^{176} + x^{11} + x^3 + x^2 + 1$

แสดงขั้นตอนการเข้ารหัสและถอดรหัส $GF(2^{176})$

A = 6f4e a37c cb7c 6443 44de 2368 dba3 c524 bd4a e585 bb6c **ขั้นตอนที่ 2**

B = 8a38 e3e4 1595 f58f a35e c0a2 dd61 d80a 8984 bc08 4f91

Primitive = 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 080d

Random point of ECC ($P(x,y)$): **ขั้นตอนที่ 3,4**

$P(x,y)$

---

x = 9e15 2cda 8c22 e6e8 23d7 674e 72e5 a195 44b2 35e7 c3d1

y = 8bf3 e7ed dbe0 3681 aa7b c488 1815 dba6 c633 569a 43b3

เมื่อ A,B เป็นสัมประสิทธิ์ของ ECC , p(x) เป็นค่าของ Irreducible Polynomial ของ $GF(2^{176})$

5.8.2 การเลือกค่า Public-Key ของ Elgamal Algorithm

ค่าของ Public และ Key นั้นสามารถสุ่มเลือกได้ตั้งแต่ $0-(2^{176}-1)$ เพื่อใช้สำหรับสร้างคู่ของ Elliptic Curve ต่อไปจากตัวอย่างที่ 5.1 เมื่อ

$a = 75$ **ขั้นตอนที่ 5**

$k = 100$

$R(x, y) = a \cdot P(x, y)$

---

x = c548 8f45 e2a3 7a25 8347 6ef3 ef67 11cc 5859 c389 ae17

y = dec6 0dec 866d 09ba 2d63 779a a8f3 c836 255e 473c 4d9b

$Q(x, y) = k \cdot P(x, y)$

---

x = e4ea 83c4 fbda eef0 88f8 1414 5978 0c95 c76c e64a 6223

y = 5bd7 13e9 0ce6 0212 cca5 3649 b952 433a 021a 9578 4b4a

Encryption

$k \cdot R = (x', y')$: **ขั้นตอนที่ 6,7**

---

x' = f874 8697 a7d1 73d5 da23 9933 615f 89d1 aef7 93af c1e8

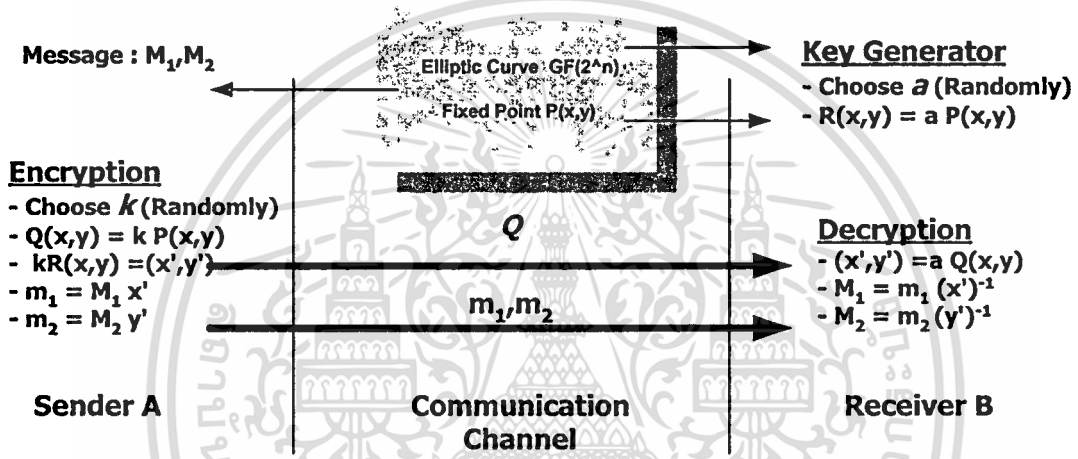
y' = 34ab ecbb a95f 327f ffb2 50f2 c7a8 65d5 dd5a 471e 8422

Decryption

$a \cdot Q = (x', y')$ **ขั้นตอนที่ 6,8**

$(x')^{-1}$ = 78c4 f832 64fe 4305 7a4c 2f3e 9cd2 88bd 8d1c e7ee be32

$(y')^{-1}$ = f64c b612 cb58 5156 292d f8d1 0e60 6c8a a4c9 677a a0b9



รูปที่ 5.8 แสดงการประยุกต์ใช้งานของ ElGamal ECC

5.9 แสดงการเปรียบเทียบ Time ในการเข้ารหัสและถอรหัสของ RSA ElGamal Discrete Logarithm (EDL) และ ElGamal Elliptic Curve(EEC)

ในการพิจารณาเวลาของอัลกอริธึม RSA EDL และ EEC สามารถพิจารณาได้ในรูปแบบของโอเปอร์เรชั่นที่เกิดขึ้นภายในอัลกอริธึมเหล่านั้น สามารถแสดงได้ดังนี้

ตารางที่ 5.7 แสดงความซับซ้อนของอัลกอริธึมของ RSA

	RSA
Public	(n, e)
Key	d

ตารางที่ 5.8 แสดงความซับซ้อนของอัลกอริธึมของ RSA(ต่อ)

Key Generator	$n = p \times q$ $\phi = (p - 1) \times (q - 1)$ $e \leftarrow$ Random $\gcd(e, \phi) = 1$ $d = e^{-1} \bmod \phi$
Encryption	$c = m^{e} \pmod n$
Decryption	$m = c^{d} \pmod n$

จากตารางที่ 5.8 ซึ่งแสดงถึงความซับซ้อนของอัลกอริธึมของ RSA ในการสร้างค่าของ Public-Key พบว่าค่าของ Public-Key เกิดขึ้นจากค่าของ n และค่าของ n เกิดจากการแยกตัวประกอบออกเป็นจำนวนเฉพาะ p และ q ซึ่งถ้าทราบค่าของ p และ q แล้วจะทำให้สามารถทราบค่าของ Public-Key ได้ ดังนั้นเวลาในการสร้างค่าของ Public-Key จะแปรผันโดยตรงกับ การหาค่า e^{-1} ($\tau_{Key} \propto \tau_{e^{-1}}$) ส่วนเวลาที่ใช้สำหรับการเข้ารหัสและถอดรหัสจะเป็น $e \cdot \tau_{NMUL}$ และ $d \cdot \tau_{NMUL}$

ตารางที่ 5.9 แสดงความซับซ้อนของอัลกอริธึมของ EDL

	EDL
Public	$p(x), \alpha, \beta$
Key	a
Key Generator	$p(x) =$ Primetive Polynomial $\alpha \leftarrow$ Random $\in GF(2^{n})$ $a \leftarrow$ Random $\in n$ $\beta = \alpha^{a} \bmod \phi$
Encryption	$\gamma = \alpha^{k} \bmod p(x)$ $\delta = m \cdot \beta^{k} \bmod p(x)$
Decryption	$m = \gamma^{-a} \cdot \delta \bmod p(x)$

จากตารางที่ 5.9 ซึ่งแสดงถึงความซับซ้อนของอัลกอริธึมของ EDL ในการสร้างค่าของ Public-Key พบว่าค่าของ Public-Key เกิดขึ้นจากการเลือกค่าของ a และ α ซึ่งเป็นจำนวนเต็ม (n) และ $GF(2^n)$ ดังนั้นเวลาในการสร้างค่าของ Public-Key จะแปรผันโดยตรงกับ การหาค่า α^a ($\tau_{Key} \propto \tau_{\alpha^a}$) ส่วนเวลาที่ใช้สำหรับการเข้ารหัสและถอดรหัสจะเป็น $2 \cdot k \cdot \tau_{MUL} + \tau_{MUL}$ และ $\tau_{MUL} + a \cdot \tau_{INV}$

ตารางที่ 5.10 แสดงความซับซ้อนของอัลกอริธึมของ EEC

	EEC
Public	(x', y')
Key	$(x', y')^{-1}$
Key Generator	$p(x) =$ Primetive Polynomial $A, B \leftarrow$ Random $\in GF(2^n)$ Gen $\rightarrow$ #EC(x , y) $y^2 + xy = x^3 + Ax^2 + B \bmod P(x)$ Select $\rightarrow$ P(x, y) $\in$# EC(x, y) Select $\rightarrow$ a, k $\in$ n $R(x, y) = a \cdot P(x, y)$ $Q(x, y) = k \cdot P(x, y)$ $(x', y') = k \cdot R(x, y)$ $(x', y')^{-1} = a \cdot Q(x, y)$
Encryption	$m_1 = M_1 y' \pmod n$ $m_2 = M_2 y' \pmod n$
Decryption	$M_1 = m_1 (x')^{-1} \pmod n$ $M_2 = M_2 (y')^{-1} \pmod n$

จากตารางที่ 5.10 ซึ่งแสดงถึงความซับซ้อนของอัลกอริธึมของ EEC เวลาสำหรับการสร้างค่าของ Public-Key ขึ้นอยู่กับการคำนวณแปรผันโดยตรงกับการคำนวณ 3 ส่วนคือ 1. ส่วนของการสร้างเซตของคู่ลำดับบน EC (#EC(x,y)) 2. ส่วนของการคำนวณค่าของ R(x,y) และ Q(x,y) 3. ส่วนของการคำนวณค่าของ (x',y') และ $(x',y')^{-1}$ ดังนั้นเวลาในการสร้างค่าของ Public-Key นั่น

เท่ากับ $\tau_{Key} \propto \tau_{GEN\_\#EC(x,y)} + 4\tau_{OEC}$ ส่วนเวลาที่ใช้สำหรับการเข้ารหัสและถอดรหัสจะเป็น $2 \cdot \tau_{MUL}$ และ $2 \cdot \tau_{NMUL} + 2 \cdot \tau_{INV}$

เมื่อ τ_{Nmul} คือ เวลาที่ใช้ในการคูณของจำนวนเต็ม

$\tau_{GEN\_\#EC(x,y)}$ คือ เวลาทีใช้สำหรับสร้างเซตของคู่ลำดับ (x,y) ที่ทำให้สมการ EC เป็นจริง

τ_{mul} คือ เวลาที่ใช้ในการคูณโพลิโนเมียลดีกรี n

τ_{INV} คือ เวลาที่ใช้ในอิเวอร์สโพลิโนเมียลดีกรี n

τ_{OEC} คือ เวลาที่ใช้สำหรับคำนวณค่าของการ Point Double(PD) และPoint Add (PA) บน EC ($\tau_{PD} + \tau_{PA}$)

$\tau_{e^{-1}}$ คือ เวลาที่ใช้สำหรับการการอินเวอร์สของ RSA

τ_{α^a} คือ เวลาที่ใช้สำหรับการคูณ α จำนวน a ครั้งบน Discrete Logrithm (DL)

τ_{Key} คือ เวลาในการกำเหนิดค่าของ Public-Key

ตารางที่ 5.11 แสดงการเปรียบเทียบเวลาสำหรับการสร้างค่าของ Public-Key

	Time
RSA	$\tau_{Key} \propto \tau_{e^{-1}}$
EDL	$\tau_{Key} \propto \tau_{\alpha^a}$
EEC	$\tau_{Key} \propto \tau_{GEN\_\#EC(x,y)} + 4\tau_{OEC}$

ตารางที่ 5.12 แสดงการเปรียบเทียบเวลาที่ใช้สำหรับการเข้ารหัสและถอดรหัสของ RSA และ EEC [15]

RSA		**EEC**	
Encryption	Decryption	Encryption	Decryption
$e \cdot \tau_{NMUL}$	$d \cdot \tau_{NMUL}$	$2 \cdot \tau_{MUL}$	$2 \cdot \tau_{MUL} + 2 \cdot \tau_{INV}$

ตารางที่ 5.13 แสดงการเปรียบเทียบเวลาที่ใช้สำหรับการเข้ารหัสและถอดรหัสของ EDL และ EEC15]

EDL		EEC	
Encryption	Decryption	Encryption	Decryption
$2 \cdot k \cdot \tau_{MUL} + \tau_{MUL}$	$\tau_{MUL} + a \cdot \tau_{INV}$	$2 \cdot \tau_{MUL}$	$2 \cdot \tau_{MUL} + 2 \cdot \tau_{INV}$

เนื่องความซับซ้อนของอัลกอริธึมที่ใช้ในการกำเหนิดค่าของ Public-Key ต่างกันจึงทำให้เวลาสำหรับการกำเหนิดค่าของ Public-Key แตกต่างกันด้วย ซึ่งจะแปรผันตามความซับซ้อนของอัลกอริธึมนั่นเอง ซึ่งเรียงตามลำดับดังนี้ RSA < EDL < EC ซึ่งทำให้ RSA(τ_{KEY}) < EDL(τ_{KEY}) < EC(τ_{KEY}) จากตารางที่ 5.10 และ 5.11 พบว่าเวลาที่ใช้สำหรับเข้ารหัส (Encryption) และถอดรหัส (Decryption) ของ RSA จะขึ้นอยู่กับขนาดของ e และ d ของ EDL ขึ้นอยู่กับขนาดของ k และ a และของ EEC จะมีค่าคงที่

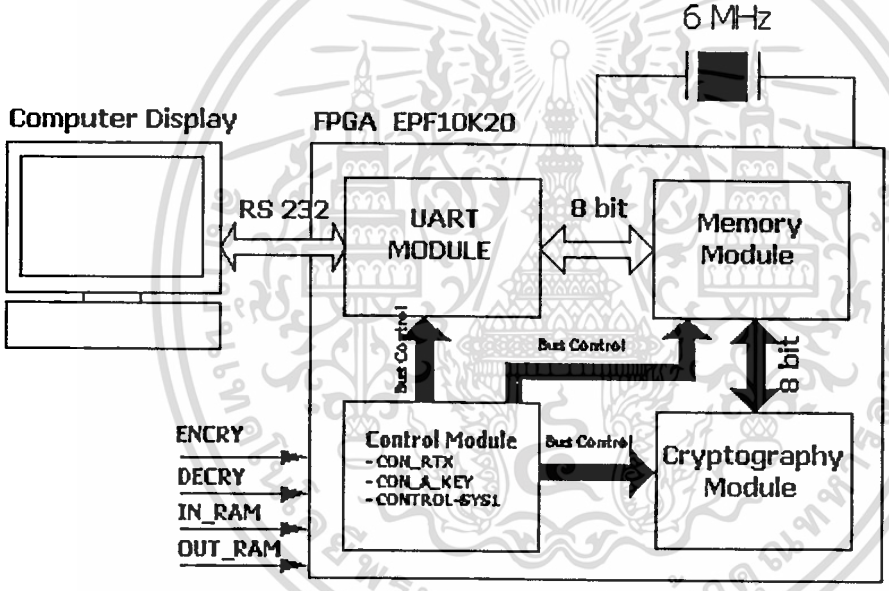
5.9 สรุป

เมื่อทำการเปรียบเทียบระบบของ Cryptography ทั้ง 3 ระบบจะเห็นได้ว่า ECC มีประสิทธิภาพที่ดีและมีความเหมาะสำหรับเข้ารหัสข้อมูลที่มีขนาดสั้น พร้อมทั้งขนาดของ Public-Key ที่มีขนาดเล็กกว่าเมื่อเทียบกับ RSA และ EDL ในส่วนของเวลาที่ใช้สำหรับเข้ารหัส (Encryption) และถอดรหัส (Decryption) ของ RSA จะขึ้นอยู่กับขนาดของ e และ d ของ EDL ขึ้นอยู่กับขนาดของ k และ a และของ EEC จะมีค่าคงที่ ดังนั้นในงานวิจัยนี้จึงเลือกใช้ระบบคณิตศาสตร์แบบ ECC และใช้ Public-Key ขนาด 176 บิต เพื่อทดสอบการทำงาน ตามมาตรฐาน IEEE P1363 ซึ่งเป็นช่วงที่สามารถใช้งานได้ดีสำหรับ ECC เพื่อทำการออกแบบตัวเข้ารหัสและถอดรหัสข้อมูลในบทที่ 6 และ 7 ต่อไป

บทที่ 6
โครงสร้างและสถาปัตยกรรมของสมาร์ทการ์ดที่ออกแบบ

บทนำ

ในบทนี้ได้กล่าวถึงโครงสร้างและสถาปัตยกรรมของสมาร์ทการ์ด ที่ได้ทำการออกแบบประกอบด้วยส่วนประกอบต่างๆ คือ ส่วนของการสื่อสารข้อมูล (UART Module) ส่วนของหน่วยความจำ (Memory Module) ส่วนของการเข้ารหัสและถอดรหัส (Cryptography Module) และส่วนของการควบคุม (Control Module) สามารถแสดงส่วนประกอบทั้งหมดได้ดังรูปที่ 6.1 โดยทั้งหมดได้ถูกรวมอยู่ภายในชิพ FPGA เพียงตัวเดียว



รูปที่ 6.1 แสดงส่วนประกอบต่างๆ เพื่อจำลองการทำงานของสมาร์ทการ์ด

ในการออกแบบได้ทำการออกแบบให้สมาร์ทการ์ดรับและส่งข้อมูลดิจิตอลจากคอมพิวเตอร์ผ่าน RS232 โดยใช้ออสซิเลเตอร์ขนาด 6 MHz ทำหน้าที่เป็นสัญญาณนาฬิกาของสมาร์ทการ์ด จากรูปที่ 6.1 ข้อมูลจากคอมพิวเตอร์ที่ถูกส่งออกมา จะเข้าไปยังส่วนของการสื่อสารข้อมูล (UART Module) เพื่อทำการแปลงข้อมูลซึ่งเป็นโปรโตรคอลของ RS232 ให้เป็นข้อมูลดิจิตอลแบบขนานขนาด 8 บิต แล้วจึงส่งไปเก็บไว้ยังหน่วยความจำบน FPGA เพื่อที่จะทำการเข้ารหัสและถอดรหัสข้อมูลต่อไป

ฟังก์ชันการทำงานของระบบนั้น ได้ออกแบบให้มีการทำงานอยู่ 4 ฟังก์ชันคือ ฟังก์ชันการเข้ารหัสข้อมูล(ENCRY) ฟังก์ชันการถอดรหัสข้อมูล(DECRY) ฟังก์ชันการรับข้อมูล(IN_RAM) และฟังก์ชันการส่งข้อมูล(OUT_RAM)

ในการส่งข้อมูลไปยังคอมพิวเตอร์นั้น ส่วนของการสื่อสารข้อมูล(UART Module) ได้แปลงข้อมูลดิจิตอลขนาด 8 บิต ให้เป็นโปโตรคอลของ RS232 แล้วจึงส่งไปยังคอมพิวเตอร์เพื่อแสดงผลต่อไป

6.1 ส่วนของการสื่อสารข้อมูล(UART Module)

เนื่องจากระบบที่ทำการออกแบบเป็น ระบบการสื่อสารข้อมูลอนุกรมแบบ Asynchronous ระหว่างคอมพิวเตอร์และระบบ ซึ่งสามารถกำหนด Baud Rate ที่ใช้งานได้ โดยที่คอมพิวเตอร์ได้เตรียมส่วนของการติดต่อแบบนี้ไว้แล้วเรียกว่า UART (Universal Asynchronous Receiver Transmitter) ดังนั้นจึงได้ทำการออกแบบระบบในส่วนนี้บน FPGA แบบ UART เพื่อใช้ติดต่อระหว่างคอมพิวเตอร์และระบบโดยกำหนด Baud Rate ที่ใช้งานอยู่ที่ 9,600 บิตต่อวินาที โดยภายใน UART ได้แบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ ส่วนของการรับและส่วนของการส่งข้อมูล โปรโตคอลของการส่งและการรับประกอบด้วย Start Bit (1), Data Bit (8), Parity Bit (1), Stop Bit (1) แสดงได้ดังรูปที่ 6.1

รูปที่ 6.2 แสดงโปรโตคอลของการสื่อสารข้อมูล UART

เนื่องจากระบบกำหนดการทำงานของ Baud Rate ไว้ที่ 9,600 บิตต่อวินาที ที่ความถี่ 6 เม็กกะเฮิร์ต ดังนั้นต้องลดความถี่ที่ใช้สำหรับรับและส่งข้อมูลเพื่อให้สัมพันธ์กับข้อมูลที่มาจากคอมพิวเตอร์ สามารถคำนวณความถี่ของการรับและส่งข้อมูลได้ดังสมการที่ 6.1

$$Round = \frac{f(Hz)}{12 \times Buad \quad Rate} \tag{6.1}$$

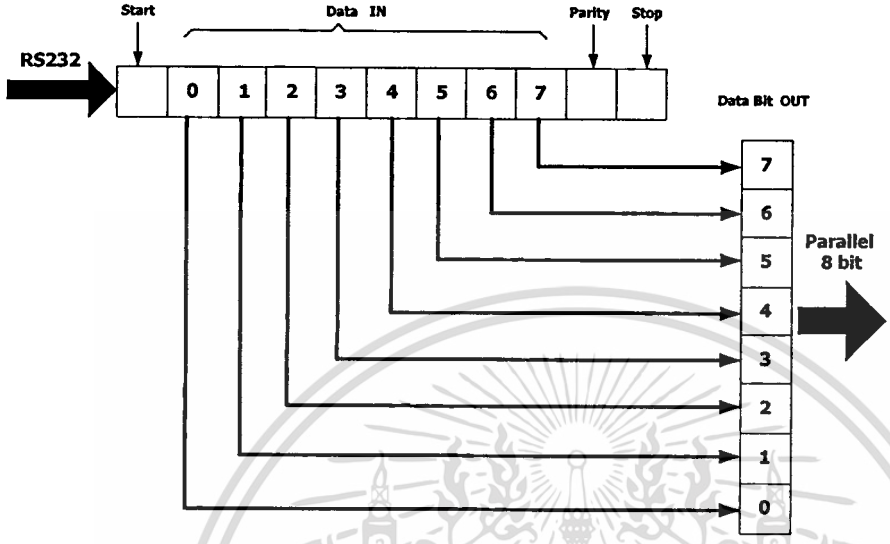
$Round$ = ความถี่ของการรับส่งข้อมูล

$f(Hz)$ = ความถี่ใช้งาน(เฮิร์ต)

$Buad \quad Rate$ = จำนวนบิตต่อวินาที

6.1.1 ส่วนของการรับข้อมูล

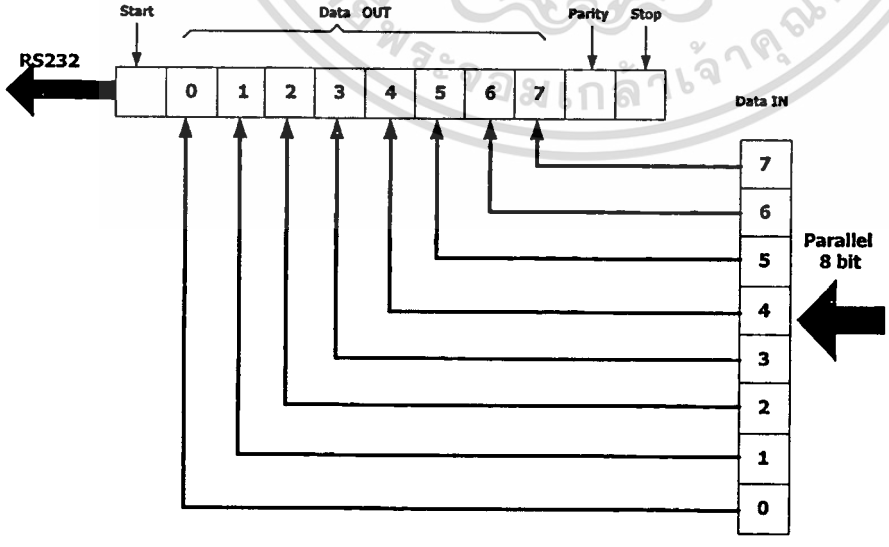
ส่วนนี้ต้องทำการรับข้อมูลซึ่งเป็นโปรโตคอลของ RS232 แล้วทำการดึงข้อมูลในส่วน ขนาด 8 บิต ออกมาใช้งาน สามารถแสดงได้ดังรูปที่ 6.3



รูปที่ 6.3 แสดงการดึงข้อมูลขนาด 8 บิต จากโปรโตคอล RS232

6.1.2 ส่วนของการส่งข้อมูล

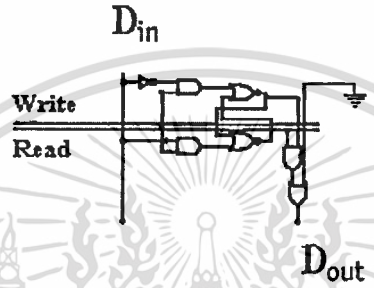
ส่วนนี้ต้องทำการแปลงข้อมูลขนาด 8 บิต ให้เป็นโปรโตคอลของ RS232 เพื่อส่งไปยังคอมพิวเตอร์โดยต้องเพิ่มส่วนของ Star Bit Parity Bit และ Stop Bit สามารถแสดงได้ดังรูปที่ 6.4



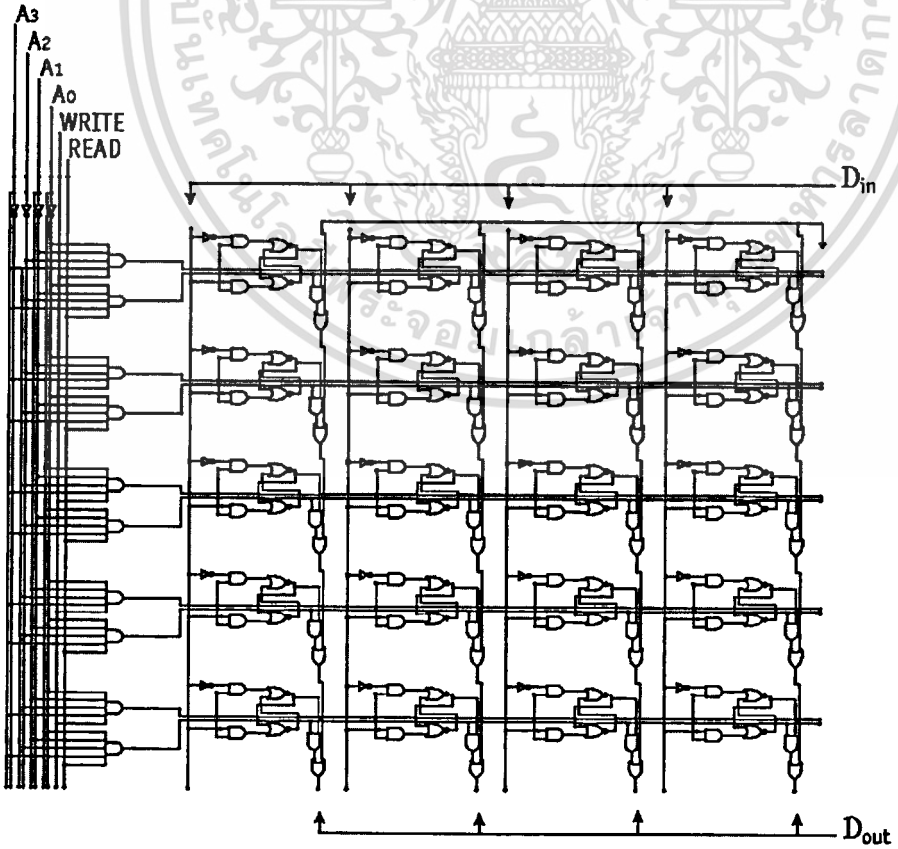
รูปที่ 6.4 แสดงการแปลงข้อมูลขนาด 8 บิต เป็นโปรโตคอล RS232

6.2 ส่วนของหน่วยความจำ(Memory Module)

ในการออกแบบหน่วยความจำ สามารถออกแบบหน่วยความจำได้ 2 แบบ คือ ROM และ RAM ซึ่งแต่ละแบบมีการใช้งานที่แตกต่างกัน ในวิทยานิพนธ์ฉบับนี้ได้ทำการออกแบบหน่วยความจำเป็นแบบ RAM เพื่อใช้สำหรับอ่านและเขียนข้อมูลในการเข้ารหัสและถอดรหัส ซึ่งโครงสร้างของหน่วยความจำต่อหนึ่งหน่วย แสดงได้ดังรูปที่ 6.5 และแสดงตัวอย่างการออกแบบหน่วยความจำขนาด 16 x 4 บิต ได้ดังรูปที่ 6.6 ได้ดังนี้



รูปที่ 6.5 แสดงวงจรของหน่วยความจำต่อหนึ่งเซลล์



รูปที่ 6.6 แสดงวงจรหน่วยความจำขนาด 16 x 4 บิต

เนื่องจาก FPGA เบอร์ EPF10K20RC มีจำนวนเกตที่ใช้สำหรับการออกแบบทั้งหมด 20,000 เกตและสามารถกำหนดให้เป็นหน่วยความจำแบบ RAM ภายในได้ไม่เกิน 12,288 เกตโดยแต่ละเกตจะเท่ากับหน่วยความจำขนาด 1 บิต ดังนั้นระบบนี้ต้องการออกแบบหน่วยความจำที่มีขนาด 256 ไบท์ เมื่อทำการสังเคราะห์(Synthesizable) เป็นวงจรระดับเกตจะได้จำนวนเกตเท่ากับ 2048 เกต

6.3 ส่วนของการเข้ารหัสและถอดรหัส(Cryptography module)

ส่วนของการเข้ารหัสและถอดรหัสนั้น ได้ทำการออกแบบโดยการนำคู่ลำดับ(x,y) ซึ่งมีขนาด m บิต มาทำการออกแบบเป็นวงจรคูณแบบค่าคงที่อนุกรม(Serial Constant Multiplier) ในงานวิจัยนี้ได้ทำการออกแบบวงจรเข้ารหัสและวงจรถอดรหัสข้อมูลขนาด 176 บิต จากสมการที่ 3.4 สามารถแสดงได้ในรูปของสมการดังนี้

$$\sum_{i=0}^{175} d_i x^i = \sum_{i=0}^{175} a_i x^i \cdot \sum_{i=0}^{175} b_i x^i \bmod p(x) \qquad (6.2)$$

โดยที่ $a_i, b_i, d_i \in GF(2)$

$$A(x) = \sum_{i=0}^{175} a_i x^i$$

$$B(x) = \sum_{i=0}^{175} b_i x^i$$

$$D(x) = \sum_{i=0}^{175} d_i x^i$$

ดังนั้นเมื่อทำการออกแบบเป็นส่วนของการเข้ารหัสและถอดรหัสจึงกำหนดให้

$A(x)$ เป็นค่าของ Public หรือ Private Key ขนาด 176 บิต

$B(x)$ เป็นข้อมูลขนาด 176 บิต

$D(x)$ เป็นข้อมูลที่ได้เข้ารหัสเรียบร้อยแล้ว

ค่าของ Public และ Private Key ที่ใช้สำหรับทำการออกแบบนั้นได้สร้างขึ้นจากหัวข้อ 5.8.2 ได้ค่าของ (x,y) ในรูปของเลขฐานสองดังนี้

โพลิโนเมียลปฐม (Primitive Polynomial) $p(x)$

Polynimial : $X^{176} + x^{11} + x^3 + x^2 + 1$

Hexcimal : 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 080d

Binary : 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
1000 0000 1101

Public

x

Hexcimal : f874 8697 a7d1 73d5 da23 9933 615f 89d1 aef7 93af c1e8

Binary : 1111 1000 0111 0100 1000 0110 1001 0111 1010 0111 1101 0001 0111 0011
1101 0101 1101 1010 0010 0011 1001 1001 0011 0011 0110 0001 0101 1111
1000 1001 1101 0001 1010 1110 1111 0111 1001 0011 1010 1111 1100 0001
1110 1000

y

Hexcimal : 34ab ecbb a95f 327f ffb2 50f2 c7a8 65d5 dd5a 471e 8422

Binary : 0011 0100 1010 1011 1110 1100 1011 1011 1010 1001 0101 1111 0011 0010
0111 1110 1111 1111 1011 0010 0101 0000 1111 0010 1100 0111 1010 1000
0110 1010 1101 0101 1101 1101 0101 1010 0100 0111 0001 1110 1000 0100
0010 0010

Key

x′

Hexcimal : 78c4 f832 64fe 4305 7a4c 2f3e 9cd2 88bd 8d1c e7ee be32

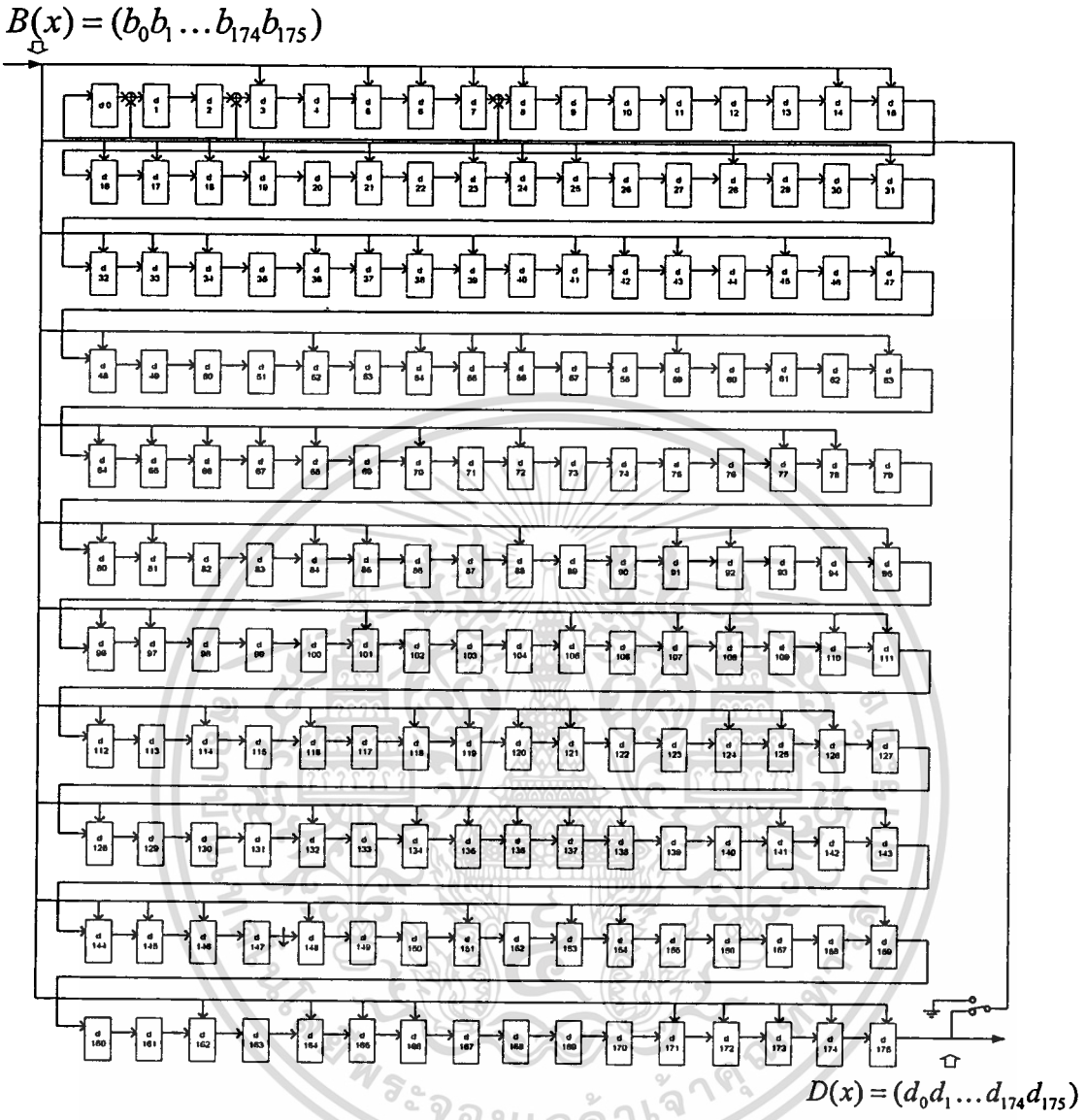
Binary : 0111 1000 1100 0100 1111 1000 0011 0010 0110 0100 1111 1110 0100 0011
0000 0101 0111 1010 0100 1100 0010 1111 0011 1110 1001 1100 1101 0010
1000 1000 1011 1101 1000 1101 0001 1100 1110 0111 1110 1110 1011 1110
0011 0010

y′

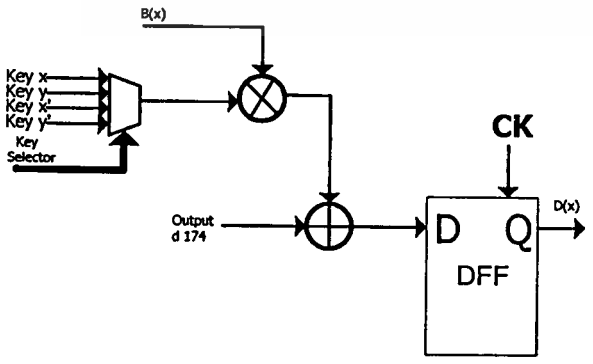
Hexcimal : f64c b612 cb58 5156 292d f8d1 0e60 6c8a a4c9 677a a0b9

Binary : 1111 0110 0100 1100 1011 0110 0001 0010 1100 1011 0101 1000 0101 0001
0101 0110 0010 1001 0010 1101 1111 1000 1101 0001 0000 1110 0110 0000
0110 1100 1000 1010 1010 0100 1100 1001 0110 0111 0111 1010 1010 0000
1011 1001

ดังนั้นสามารถนำค่า Public Key x ดังกล่าวมาออกแบบเป็นวงจรเข้ารหัสข้อมูล ดังรูปที่ 6.7



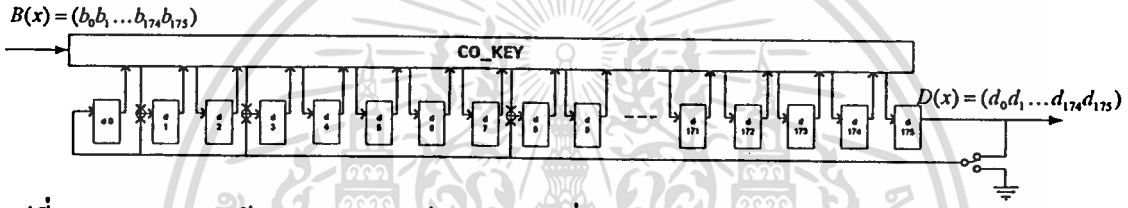
รูปที่ 6.7 แสดงวงจรเข้ารหัสข้อมูลโดยค่าของ Public- Key (x)



รูปที่ 6.8 แสดงวงจรเลือกค่าของ Public-Key x , y , x' , y'

ข้อมูลได้ถูกนำเข้าไปเก็บไว้ในรีจีสเตอร์โดยใช้ Clock จำนวน 176 ลูก เมื่อครบแล้วจะนำข้อมูลออกจากรีจีสเตอร์ โดยใช้ Clock อีกจำนวน 176 ลูก ดังนั้นจึงต้องใช้ Clock จำนวน 176 x 2 ลูกเพื่อใช้สำหรับเข้ารหัสและถอดรหัสข้อมูลจำนวน 176 บิตนั่นเอง

เนื่องจากการเข้ารหัสและถอดรหัสข้อมูลโดย (x,y) จะใช้ไม่พร้อมกันโดยใน 176 บิตแรกจะทำการใช้ Key x ก่อน และใน 176 หลังจะทำการใช้ Key y สลับกันไป ดังนั้นจึงออกแบบให้วงจรสามารถเปลี่ยนค่าของ Key ได้ โดยใช้บล็อก CON_KEY ควบคุมการทำงาน ทำให้สามารถรวมวงจรการเข้ารหัสและถอดรหัสให้เป็นชุดเดียวกันได้ ส่งผลให้จำนวนเกตที่ใช้งานลดลงและคอมเพล็กซิตี้ของวงจรยังลดลงอีกด้วย แสดงการออกแบบได้ดังรูปที่ 6.7 และวงจรการทำงานดังภาคผนวก ข.5 รูปที่ 6.8 แสดงวงจรเลือกค่าของ Public-Key x , y , x' , y' ที่อยู่ในแต่ละส่วนของ D Filp-Flop



รูปที่ 6.9 แสดงวงจรเข้าและถอดรหัสข้อมูลแบบเปลี่ยนค่าได้

รูปที่ 6.7 แสดงวงจรเข้ารหัสและถอดรหัสข้อมูลโดยมีวงจรในรูปที่ 6.8 เป็นวงจรสำหรับต่อร่วมกับ D Flip-Flop ทุกตัว เพื่อทำการเลือกค่าของ Public-Key x , y , x' , y' ในรูปที่ 6.7 ส่วนในรูปที่ 6.9 แสดงการต่อร่วมกันของวงจรในรูปที่ 6.7 และ 6.8

6.4 ส่วนของการควบคุม(Control Module)

ส่วนของการการควบคุมทำหน้าที่ในการสร้าง State Machine เพื่อให้ส่วนประกอบแต่ละส่วนทำงานได้สอดคล้องกันแบ่งออกเป็น ส่วนของการควบคุมการอ่านและเขียนข้อมูลจาก UART ไปยังหน่วยความจำ ส่วนของการควบคุมการเข้ารหัสและถอดรหัสข้อมูล และส่วนของการควบคุมการทำงานของระบบทั้งหมด ซึ่งแสดงรายระเอียดของส่วนต่างในบทที่ 7

6.4.1 ส่วนของการควบคุมการอ่านและเขียนข้อมูลจาก UART ไปยังหน่วยความจำ

เมื่อทำการทำการสังเคราะห์ (Synthesizable) เป็นวงจรระดับเกต ส่วนของการสื่อสารข้อมูลบน FPGA พบว่าได้ใช้จำนวนเกตทั้งหมด 752 เกต

6.4.2 ส่วนของการควบคุมการเข้ารหัสและถอดรหัสข้อมูล

เป็นส่วนที่ควบคุมการอ่านและเขียนข้อมูลจากหน่วยความจำเพื่อทำการเข้ารหัสและถอดรหัสข้อมูล พร้อมทั้งกำหนดค่าของ Public-Key เพื่อใช้สำหรับเข้ารหัสและถอดรหัสข้อมูลที่เข้ามา

ใน CRYPTOGRAPHY MODULE และทำการ Enable CRYPTOGRAPHY MODULE เมื่อทำการทำการสังเคราะห์ (Synthesizable) เป็นวงจรระดับเกตบน FPGA พบว่าได้ใช้จำนวนเกตทั้งหมด 1,800 เกต

6.4.3 แสดงส่วนควบคุมการทำงานของระบบทั้งหมด

ในส่วนนี้ทำการควบคุมส่วนประกอบแต่ละส่วนให้ทำงานตามฟังก์ชันซึ่งมีทั้งหมด 4 ฟังก์ชันดังนี้

1. ฟังก์ชันการรับข้อมูลจากคอมพิวเตอร์ (IN_RAM)
2. ฟังก์ชันการส่งข้อมูลไปยังคอมพิวเตอร์ (OUT_RAM)
3. ฟังก์ชันการเข้ารหัสข้อมูล (ENCRY)
4. ฟังก์ชันการถอดรหัสข้อมูล (DECRY)

เมื่อทำการทำการสังเคราะห์ (Synthesizable) เป็นวงจรระดับเกตบน FPGA พบว่าได้ใช้จำนวนเกตทั้งหมด 200 เกต

6.5 สรุป

การออกแบบและจำลองการทำงานของสมาร์ทการ์ด ได้สร้างวงจรรวมภายใน FPGA พื่อให้ได้วงจรรวมที่มีขนาดเล็ก สามารถเข้ารหัสและถอดรหัสข้อมูลได้ดี และระดับความปลอดภัยอยู่ในระดับที่ใช้งานได้ ในวิทยานิพนธ์นี้ เมื่อได้ทำการสังเคราะห์เป็นวงจรระดับเกต แต่ละส่วนได้ใช้จำนวนเกตดังนี้ ส่วนของการสื่อสารข้อมูล(UART Module) 1,600 เกต ส่วนของหน่วยความจำ (Memory Module) 2,048 เกต ส่วนของการเข้ารหัสและถอดรหัส(Cryptography Module) 5,000 เกต และส่วนของการควบคุม(Control Module) 2,752 เกต ซึ่งรายระเอียดของ FPGA เบอร์ EPF10K20 แสดงในภาคผนวก ค.

บทที่ 7

การทดสอบและผลการทดสอบ

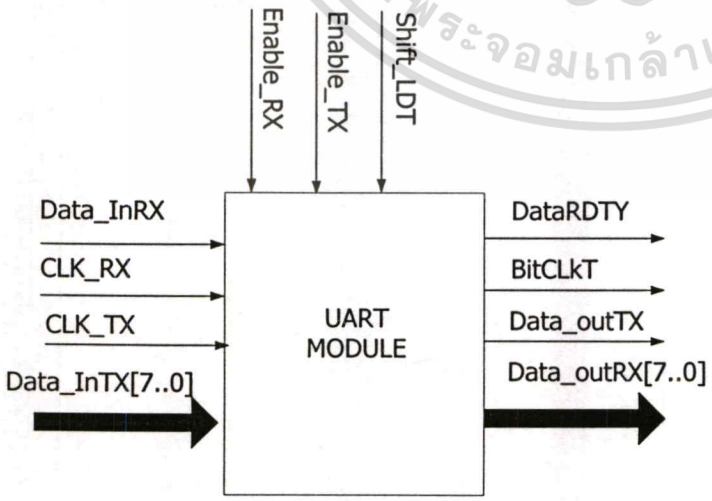
บทนี้ได้กล่าวถึงการทดสอบและผลการทดสอบของระบบการเข้ารหัสข้อมูล โดยการทดสอบกับซอฟท์แวร์จำลองการทำงาน และทดสอบบนอุปกรณ์ FPGA เบอร์ EPF10K20 ที่เวลาจริง โดยการทดสอบนั้นได้ทดสอบการเข้ารหัสข้อมูลทั้งข้อมูลตัวอักษรและข้อมูลรูปภาพ

บทนำ

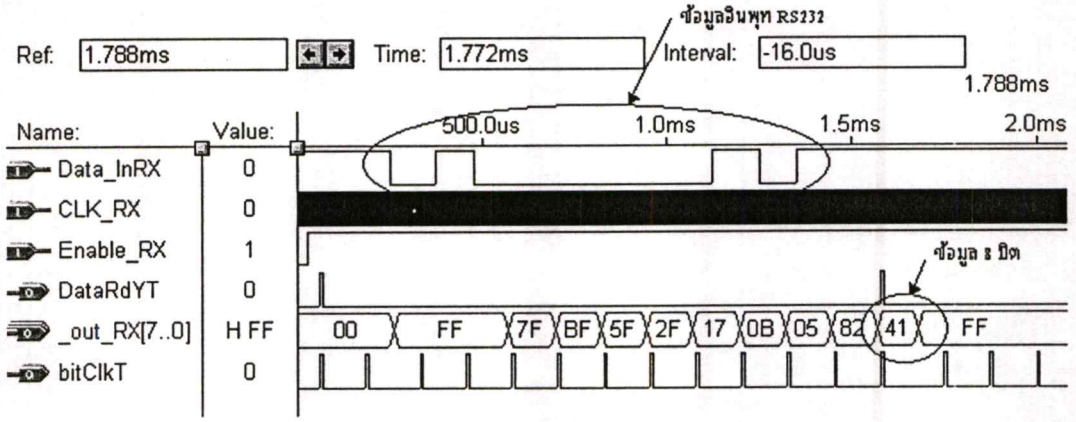
แสดงส่วนประกอบของระบบโดยจำลองการทำงานของสมาร์ทการ์ดบน FPGA เบอร์ EPF10K20 ภายในสามารถแบ่งออกเป็น 4 ส่วนใหญ่ๆ ด้วยกัน คือ ส่วนการสื่อสารข้อมูล(UART Module) ส่วนของหน่วยความจำ(Memory Module) ส่วนของการเข้ารหัสและถอดรหัส (Cryptography Module) และส่วนของการควบคุม(Control Module) การทดสอบในแต่ละส่วนนั้นได้ใช้ซอร์ฟแวร์ช่วยในการทดสอบโดยแสดงผลของการทดสอบในรูปของเวลา พร้อมทั้งได้ทำการรวมแต่ละส่วนเข้าด้วยกัน เพื่อจำลองการทำงานของสมาร์ตการ์ดและแสดงผลการทดสอบในรูปแบบของตัวอักษรและภาพต่อไป

7.1 การทดสอบส่วนของการสื่อสารข้อมูล (UART Module)

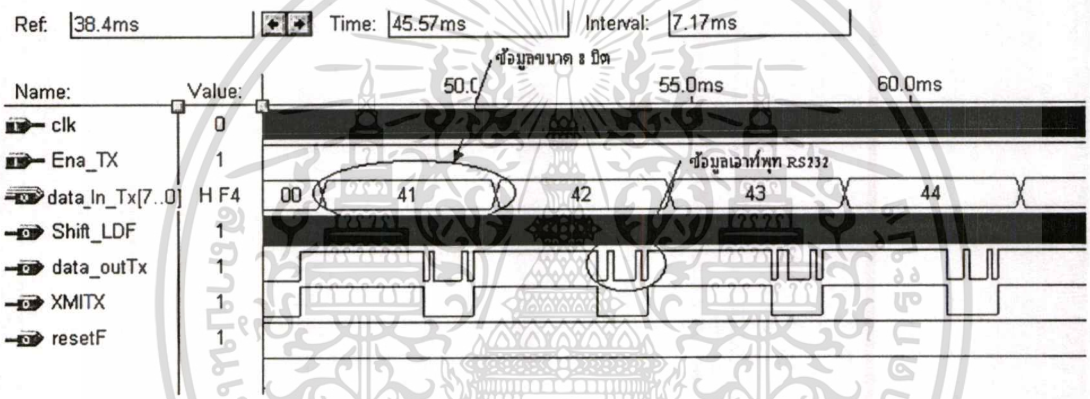
แสดงบล็อคของ UART ได้ดังรูปที่ 7.1 และเวลาที่ใช้สำหรับการรับ-ส่งข้อมูลได้ดังรูปที่ 7.2 และ7.3 พร้อมทั้งแสดงขั้นตอนในการทดสอบดังรูปที่ 7.4



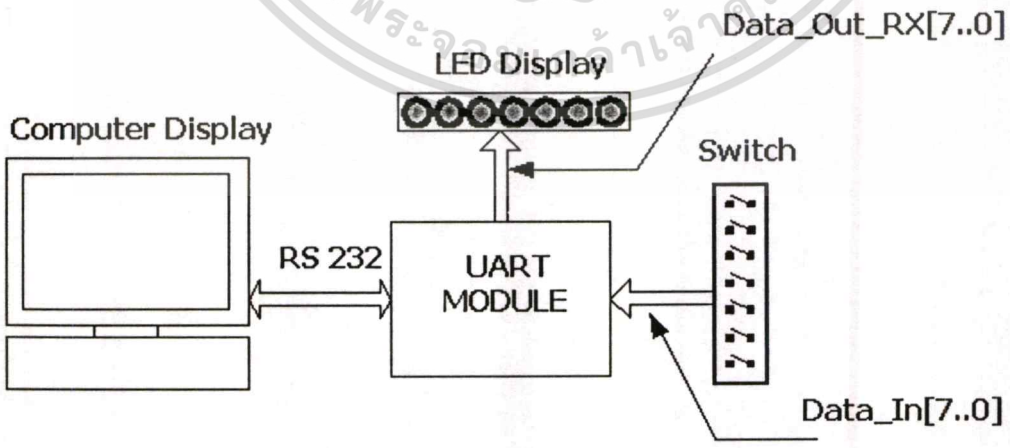
รูปที่ 7.1 แสดงบล็อคการทำงานของ UART



รูปที่ 7.2 แสดง Timing Diagram การรับข้อมูลของ UART



รูปที่ 7.3 แสดง Timing Diagram การส่งข้อมูลของ UART

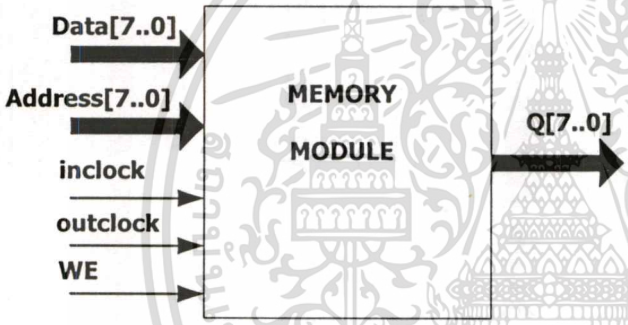


รูปที่ 7.4 แสดงขั้นตอนการทดสอบส่วนของการสื่อสารข้อมูล(UART Module)

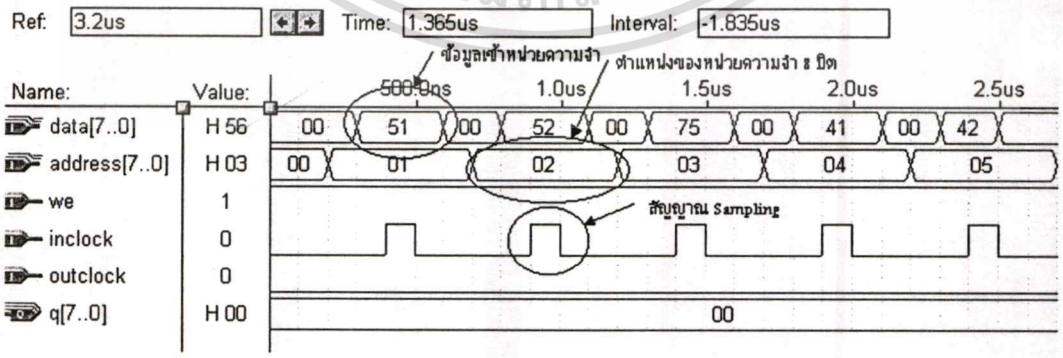
จากรูปที่ 7.2 พบว่าข้อมูลอินพุทที่เข้ามาประกอบด้วยส่วนของ Start Bit(1) Data Bit(8) Parity Bit (1) และ Stop Bit(1) ซึ่งค่าที่รับเข้ามาคือ ค่า " 41" ใช้ clock ในการรับข้อมูลทั้งหมด 11 ลูก ผลลัพธ์ที่ได้จะเกิดขึ้นใน clock ลูกที่ 11 จากรูปที่ 7.4 พบว่าข้อมูลอินพุท ที่ทำการแปลงเป็นโปรโตคอลของ RS232 มีค่าเท่ากับ "41" "42" "43" และ "44" และสัญญาณที่ขาของ XMITX จะแสดงคาบของการส่งข้อมูลซึ่งมีอัตราการส่ง 9,600 บิตต่อวินาที ซึ่งข้อมูลจะถูกเพิ่มส่วนของ Start Bit, Parity Bit และ Stop Bit ผลลัพธ์ที่ได้จะส่งออกไปทางขา data_outTX และส่งไปแสดงผลยังคอมพิวเตอร์ต่อไป เมื่อนำส่วนของการสื่อสารข้อมูล (UART Module) ไปทำการทดสอบดังรูปที่ 7.4 เพื่อทำการรับและส่งข้อมูล

7.2 การทดสอบส่วนของหน่วยความจำ (Memory Module)

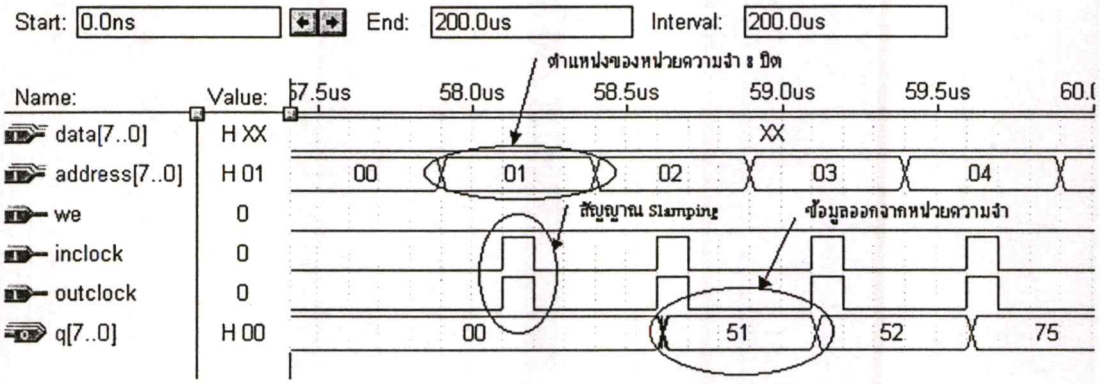
แสดงบล็อคของหน่วยความจำและ Timing diagram สำหรับการอ่านและเขียนข้อมูล พร้อมทั้งแสดงขั้นตอนในการทดสอบส่วนของหน่วยความจำดังรูปที่ 7.5, 7.6, 7.7 และ 7.8



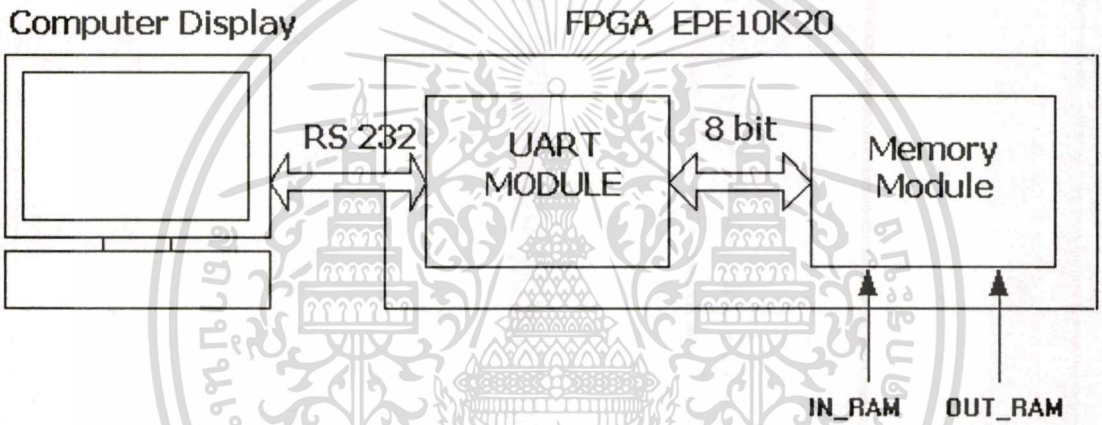
รูปที่ 7.5 แสดงบล็อคของหน่วยความจำ



รูปที่ 7.6 แสดง Timing Diagam ของการเขียนข้อมูลบนหน่วยความจำ



รูปที่ 7.7 แสดง Timing Diagam ของการอ่านข้อมูลจากหน่วยความจำ



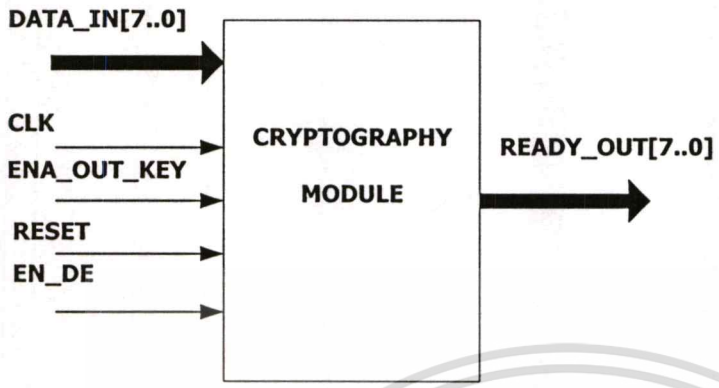
รูปที่ 7.8 แสดงขั้นตอนการทดสอบส่วนของหน่วยความจำ(Memory Module)

จากรูปที่ 7.6 แสดงการเขียนข้อมูลลงบนหน่วยความจำ ในการเขียนข้อมูลใช้สัญญาณจากขา "inclock" เป็นสัญญาณ Slamping เพื่อนำข้อมูลไปเก็บไว้ในตำแหน่งที่ปรากฏบน Address ขนาด 8 บิต จากรูปที่ 7.7 แสดงการอ่านข้อมูลออกจากหน่วยความจำ ในการอ่านนั้นได้ใช้สัญญาณจากขา "inclock" และ "outclock" เป็นสัญญาณ Slamping เพื่อนำข้อมูลขนาด 8 บิตออกจากตำแหน่งของหน่วยความจำที่ปรากฏ และเมื่อทำการทดสอบส่วนของหน่วยความจำดังรูปที่ 7.8 ปรากฏว่าสามารถอ่านและเขียนข้อมูลบนหน่วยความจำบน FPGA ได้

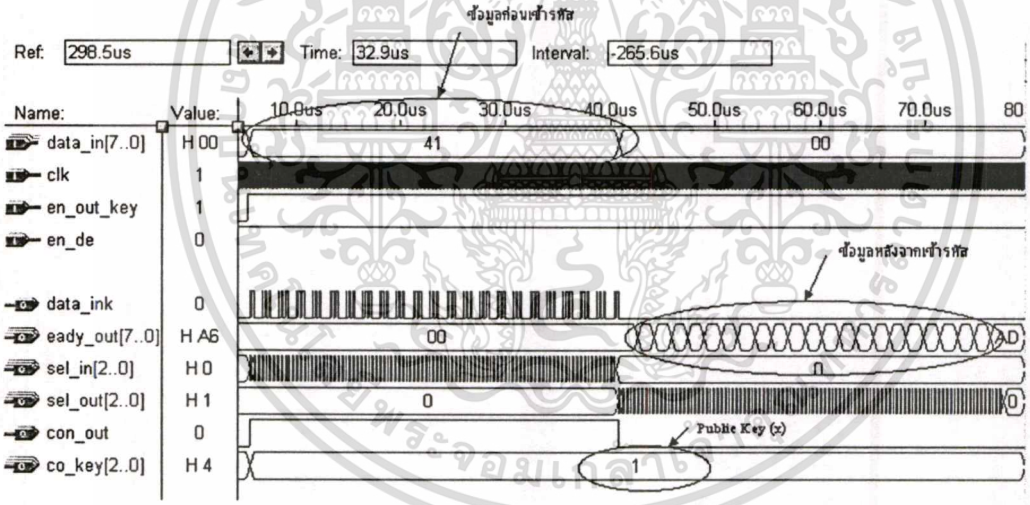
7.3 การทดสอบส่วนของการเข้ารหัสและถอดรหัส(Cryptography Module)

จากการออกแบบวงจรการเข้ารหัสและถอดรหัสในหัวข้อที่ 6.3 สามารถแสดงบล็อคของการเข้ารหัสและถอดรหัสข้อมูล Timing Diagram และขั้นตอนในการทดสอบส่วนของการเข้ารหัสและถอดรหัสได้ดังรูปที่ 7.9, 7.10, 7.11, 7.12, 7.13 และ 7.14 พร้อมทั้งแสดงการทดสอบการเข้ารหัส

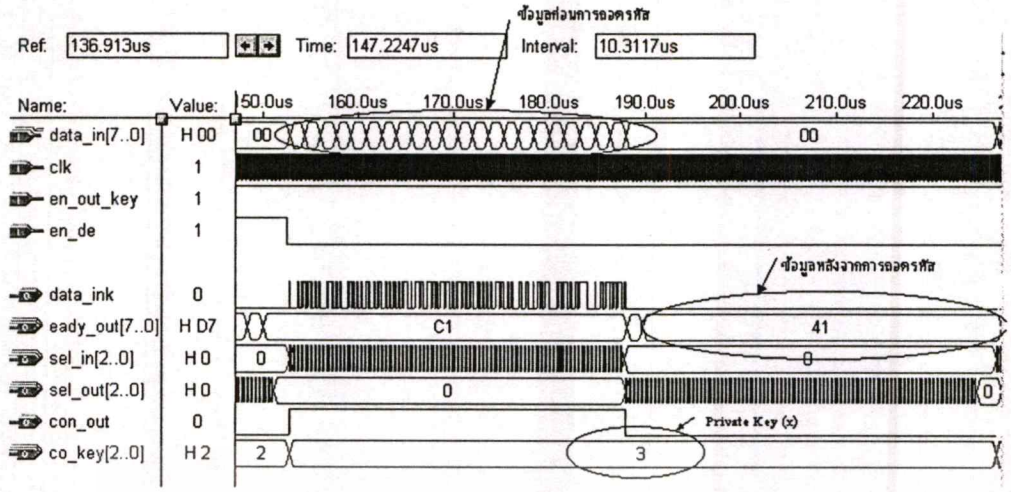
และการถอดรหัสข้อมูล M(X,Y) โดยที่ X และ Y มีขนาด 176 บิต โดยแสดงผลการทดสอบใน รูปของ ACSII โค้ด และกราฟฟิกได้ดังหัวข้อที่ 7.3.1 และ 7.3.2



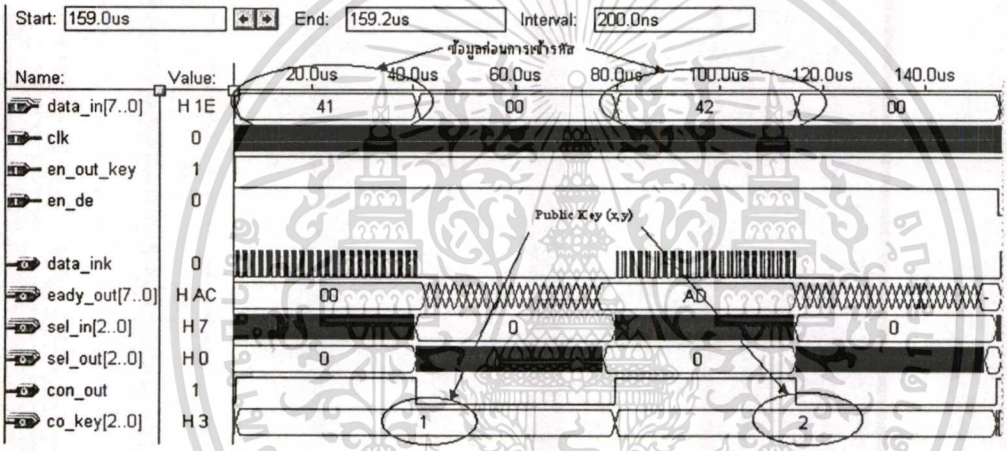
รูปที่ 7.9 แสดงบล็อคของการเข้ารหัสและถอดรหัสข้อมูล



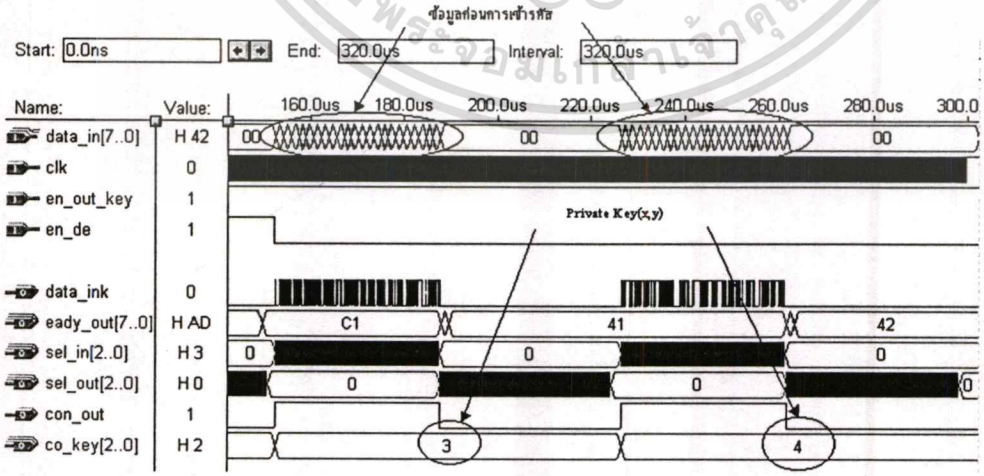
รูปที่ 7.10 แสดง Timing Diagram ของการเข้ารหัสข้อมูล



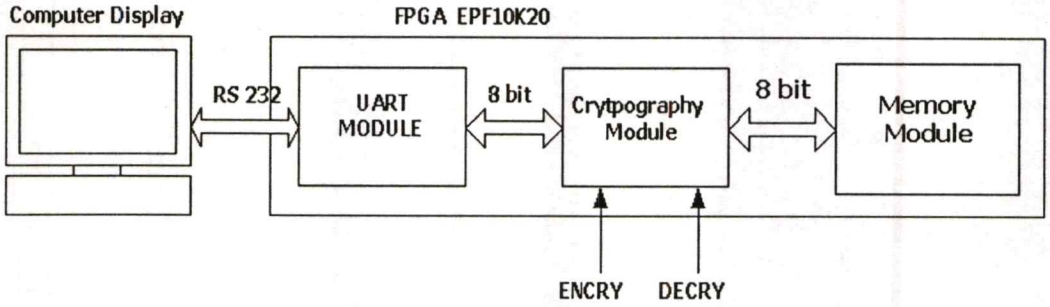
รูปที่ 7.11 แสดง Timing Diagram ของการถอดรหัสข้อมูล



รูปที่ 7.12 แสดง Timing Diagram ของการเข้ารหัสข้อมูลโดยคู่ลำดับ (X,Y)



รูปที่ 7.13 แสดง Timing Diagram ของการถอดรหัสข้อมูลโดยคู่ลำดับ (X,Y)



รูปที่ 7.14 แสดงขั้นตอนการทดสอบส่วนการเข้ารหัสและถอดรหัส(Cryptography Module)

7.3.1 การทดสอบการเข้ารหัสข้อมูล

ข้อมูลที่ทำการเข้ารหัสแสดงในรูปของ ACSII โค้ด

M(X,Y) = (4343 4343 4343 4343 4343 4343 4343 4343 4343 4343 4343 4343 ,
4242 4242 4242 4242 4242 4242 4242 4242 4242 4242 4242 4242)

Plaintext M(X) = 4343 4343 4343 4343 4343 4343 4343 4343 4343 4343 4343 4343 ,

Public Key $P_{K(x)}$ = F874 8697 A7D1 73D5 DA23 9933 615F 89D1 AEF7 93AF C1E8

Cliphertext m(x) = 58AC 1267 4666 DFCC 97FF B061 0CC1 70EE F136 27B2 9B93

Plaintext M(Y) = 4242 4242 4242 4242 4242 4242 4242 4242 4242 4242 4242 4242

Public Key $P_{K(y)}$ = 34AB ECBB A95F 327F FFB2 50F2 C7A8 65D5 DD5A 471E 8422

Cliphertext m(y) = DF6E B777 EF29 9101 7C07 FA6F 7BFB 25EE 4274 0EA7 8FAC

ดังนั้นข้อมูลที่ทำการเข้ารหัสแล้วเท่ากับ

m(x,y) = (7D1D 0EDA 4B83 6300 2FB8 2EC4 B35A D67B 0D1A F82A C915 ,
DF6E B777 EF29 9101 7C07 FA6F 7BFB 25EE 4274 0EA7 8FAC)

เมื่อค่าของ '43' และ '42' ของ ASCII โค้ด มีค่าเท่ากับ 'C' และ 'B' สามารถแสดงผลข้อมูลในรูปตัวอักษรดังในรูปที่ 7.16, 7.17, 7.18, 7.19 และ 7.20 ผลการทดสอบแสดงได้ดังนี้

CCCCCCCCCCCCCCCCCCCCCC

รูปที่ 7.15 แสดงตัวอักษรเพื่อใช้สำหรับการเข้ารหัสขนาด 176 บิตกับ Key x

X¬□gFfßÌ—ÿ°a□Ápîñ6'²›“

รูปที่ 7.16 แสดงผลการทดสอบการเข้ารหัสจากรูปที่ 7.15
ในรูปแบบตัวอักษรของ Times New Roman

Xฌ□gFfฺB₩—□฽ฐa□มp์๑6'฻“

รูปที่ 7.17 แสดงผลการทดสอบการเข้ารหัสจากรูปที่ 7.15
ในรูปแบบตัวอักษรของ AngsanaUPC

BBBBBBBBBBBBBBBBBBBBB

รูปที่ 7.18 แสดงตัวอักษรเพื่อใช้สำหรับการเข้ารหัสขนาด 176 บิตกับ Key y

ßn·wï)‘□|□úo{û%îBt□§□¬

รูปที่ 7.19 แสดงผลการทดสอบการเข้ารหัสจากรูปที่ 7.18
ในรูปแบบตัวอักษรของ Times New Roman

฿n₸w⊙)‘□|□ฬo{c~%Bt□ง์ฌ

รูปที่ 7.20 แสดงผลการทดสอบการเข้ารหัสจากรูปที่ 7.18
ในรูปแบบตัวอักษรของ AngsanaUPC

7.3.2 การทดสอบการถอดรหัสข้อมูล

ข้อมูลที่ทำการถอดรหัสแสดงในรูปของ ACSII โค้ด

m(x,y) = (7D1D 0EDA 4B83 6300 2FB8 2EC4 B35A D67B 0D1A F82A C915 ,
DF6E B777 EF29 9101 7C07 FA6F 7BFB 25EE 4274 0EA7 8FAC)

Cliphertext m(x) = 58AC 1267 4666 DFCC 97FF B061 0CC1 70EE F136 27B2 9B93

Public Key $P_{K(x)}$ = 78C4 F832 64FE 4305 7A4C 2F3E 9CD2 88BD 8D1C E7EE BE32

Plaintext M(X) = 4343 4343 4343 4343 4343 4343 4343 4343 4343 4343 4343

Cliphertext m(y) = DF6E B777 EF29 9101 7C07 FA6F 7BFB 25EE 4274 0EA7 8FAC

Public Key $P_{K(y)}$ = F64C B612 CB58 5156 292D F8D1 0E60 6C8A A4C9 677A A0B9

Plaintext M(Y) = 4242 4242 4242 4242 4242 4242 4242 4242 4242 4242 4242

ดังนั้นข้อมูลที่ทำการถอดรหัสข้อมูลเท่ากับ

M(X,Y) = (4343 4343 4343 4343 4343 4343 4343 4343 4343 4343 4343 ,
4242 4242 4242 4242 4242 4242 4242 4242 4242 4242 4242)

สามารถแสดงผลของการถอดรหัสในรูปตัวอักษรได้ดังนี้

m(x,y) =

x = X¬□gFfß3Ì—ÿºa□Ápîñ6²›“

y = ßn·wï)‘□|□úo{û%îBt□§□¬

M(X,Y) =

X = CCCCCCCCCCCCCCCCCCCCCCCC

Y = BBBBBBBBBBBBBBBBBBBBBBBB

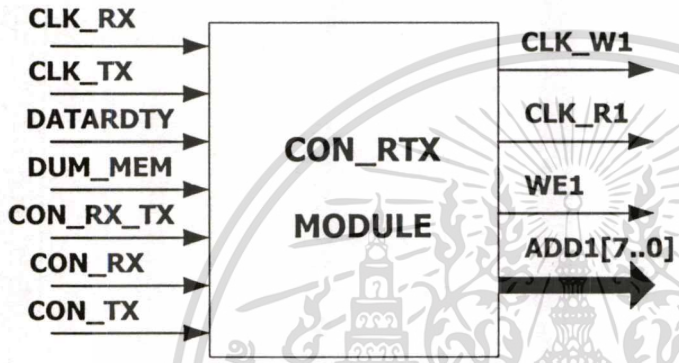
จาก Timing Diagram ของการเข้ารหัสข้อมูล สามารถเข้ารหัสข้อมูลได้ทีละ 176 บิตโดยใช้ Key ที่ 1 ซึ่งข้อมูลถูกส่งเข้าไปในวงจรโดยใช้ Clock จำนวน 176 ลูกและผลลัพธ์ที่ได้จะเกิดใน clock 176 ลูกถัดไป สามารถแสดงได้ดังรูปที่ 7.10 ต่อจากนั้นสามารถเข้ารหัสข้อมูลได้อีก 176 บิต โดยใช้ Key ที่ 2 และผลลัพธ์ที่เกิดขึ้นจะเกิดขึ้นดังแสดงในรูปที่ 7.12 สลับกันไปซึ่งในส่วนของการถอดรหัสก็ทำเช่นเดียวกันดังแสดงในรูปที่ 7.13 และ7.14 เมื่อทำการทดสอบส่วนของการเข้ารหัสและถอดรหัสดังแสดงในรูปที่ 7.15 บน FPGA สามารถแสดงผลของการทดสอบ ในหัวข้อที่ 7.3.1 และ 7.3.2 ก็สามารถเข้ารหัสและถอดรหัสข้อมูลได้

7.4 ส่วนของการควบคุม (Control Module)

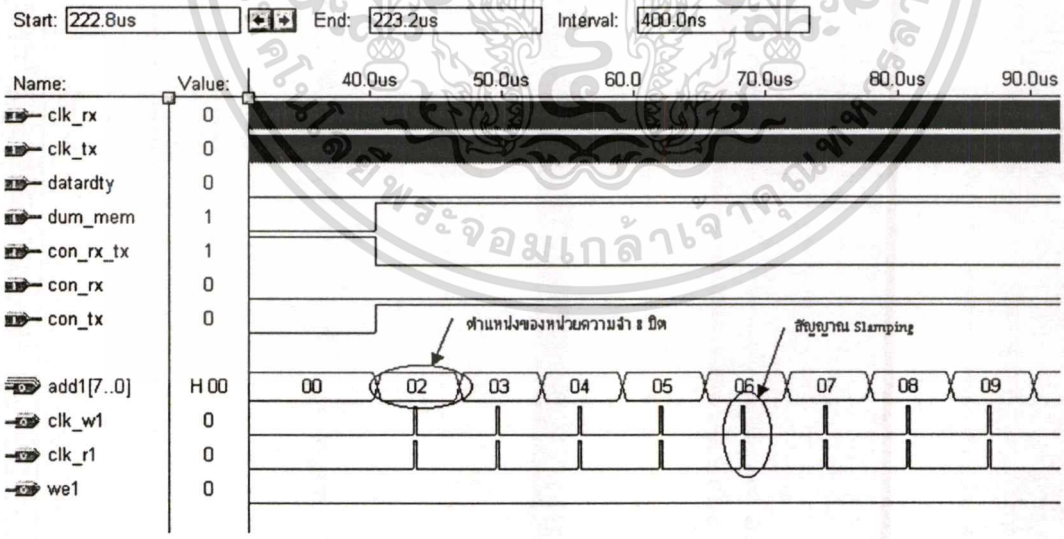
แบ่งออกเป็น 4 ส่วนของการควบคุมการอ่านและเขียนข้อมูลจาก UART ไปยังหน่วยความจำ ส่วนของการควบคุมการเข้ารหัสและถอดรหัสข้อมูลและส่วนของการควบคุมการทำงานของระบบทั้งหมด สามารถแบ่งส่วนต่างๆได้ดังนี้

7.4.1 ส่วนของการควบคุมการอ่านและเขียนข้อมูลจาก UART ไปยังหน่วยความจำ

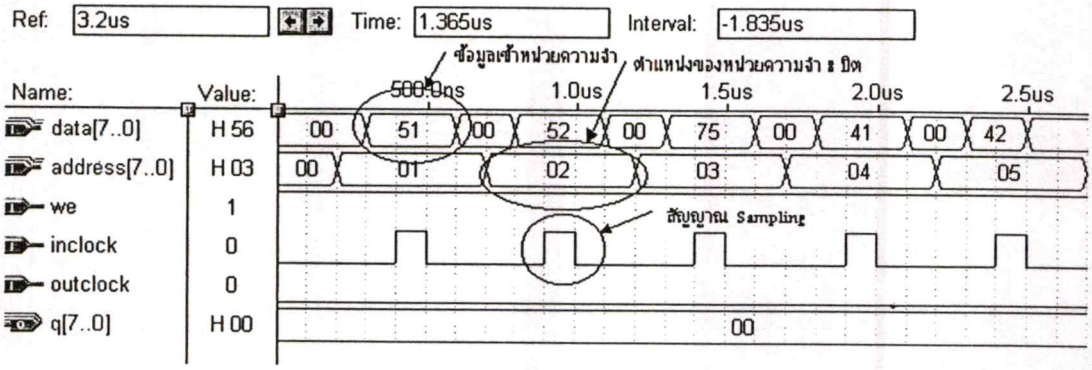
สามารถแสดงบล็อคของการควบคุมและ Timing Diagram ได้ดังรูปที่ 7.21, 7.22 และ 7.23



รูปที่ 7.21 แสดงบล็อคของการควบคุมการอ่านและเขียนข้อมูลจาก UART



รูปที่ 7.22 แสดง Timing Diagram ของการอ่านข้อมูลออกจากหน่วยความจำไปยัง UART_TX

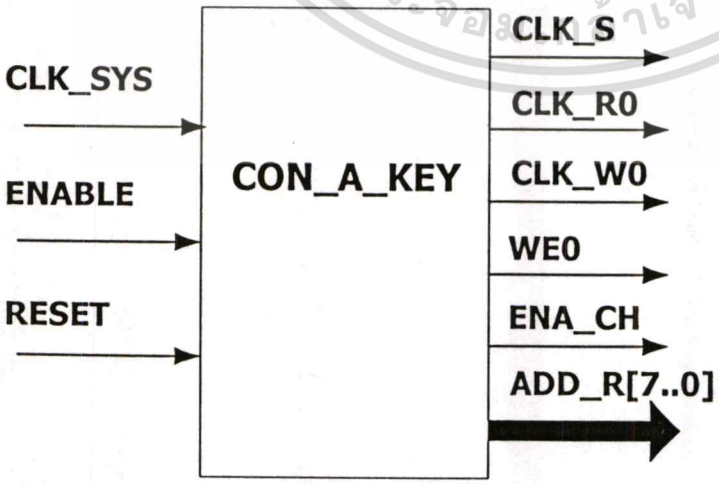


รูปที่ 7.23 แสดง Timing Diagram ของการเขียนข้อมูล UART_RX ลงบนหน่วยความจำ

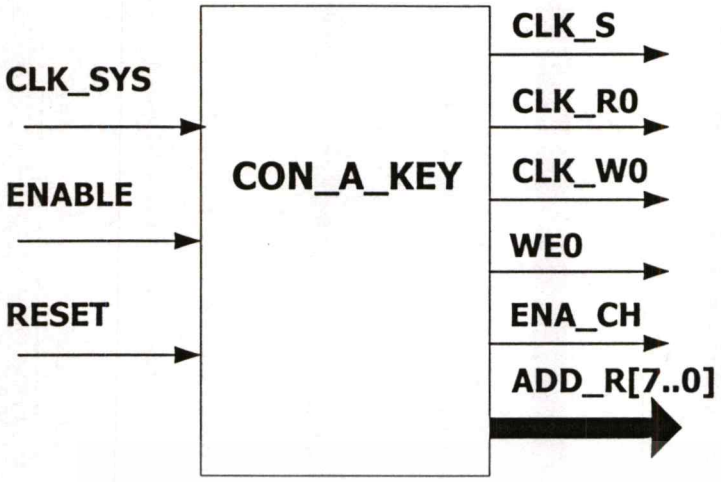
จากรูปที่ 7.22 แสดงส่วนของการควบคุมการอ่านข้อมูลจากหน่วยความจำไปยังส่วนของการสื่อสารข้อมูล(UART Module) ซึ่งได้ส่งค่าตำแหน่งของหน่วยความจำขนาด 8 บิตและสัญญาณ Slampling ออกทางขา "clk_w1" และ "clk_r1" เพื่อควบคุมการอ่านข้อมูลจากหน่วยความจำ จากรูปที่ 7.24แสดงส่วนของการควบคุมการเขียนข้อมูลจากหน่วยความจำไปยังส่วนของการสื่อสารข้อมูล (UART Module) ซึ่งได้ทำการส่งค่าตำแหน่งของหน่วยความจำขนาด 8 บิตและสัญญาณ Sampling ออกทางขา "clk_w1" เพื่อควบคุมการเขียนข้อมูลลงบนหน่วยความจำ

7.4.2 ส่วนของการควบคุมการเข้ารหัสและถอดรหัสข้อมูล

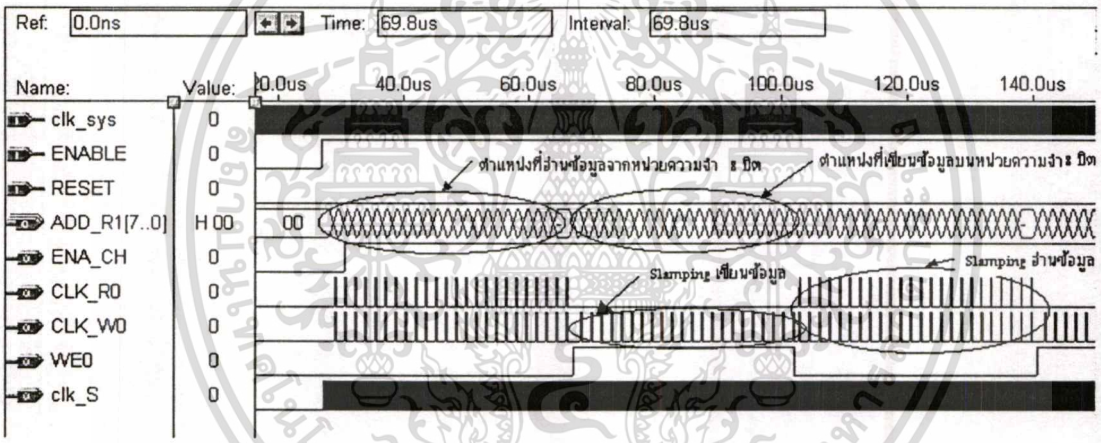
แสดงบล็อคการทำงานของส่วนของการควบคุมการเข้ารหัสและถอดรหัสข้อมูล พร้อมทั้งแสดง Timing Diagram ของ CON_A_KEY ดังรูปที่ 7.24 และ 7.25



รูปที่ 7.24 แสดงบล็อคการทำงานของส่วนของการควบคุมการเข้ารหัส



รูปที่ 7.24 แสดงบล็อคการทำงานของส่วนของการควบคุมการเข้ารหัสและถอดรหัสข้อมูล(CON_A_KEY)

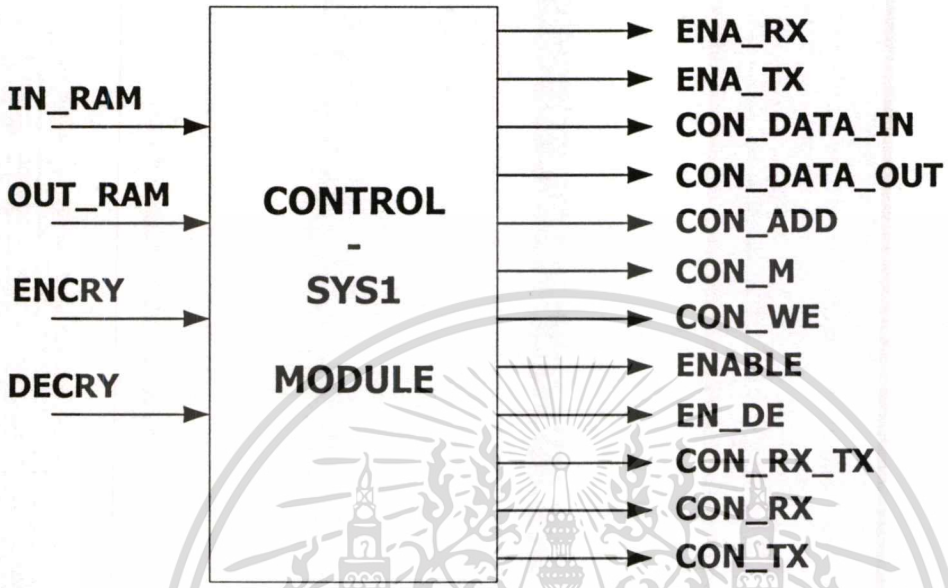


รูปที่ 7.25 แสดง Timing Diagram ของส่วนของการควบคุมการเข้ารหัสและถอดรหัสข้อมูล(CON_A_KEY)

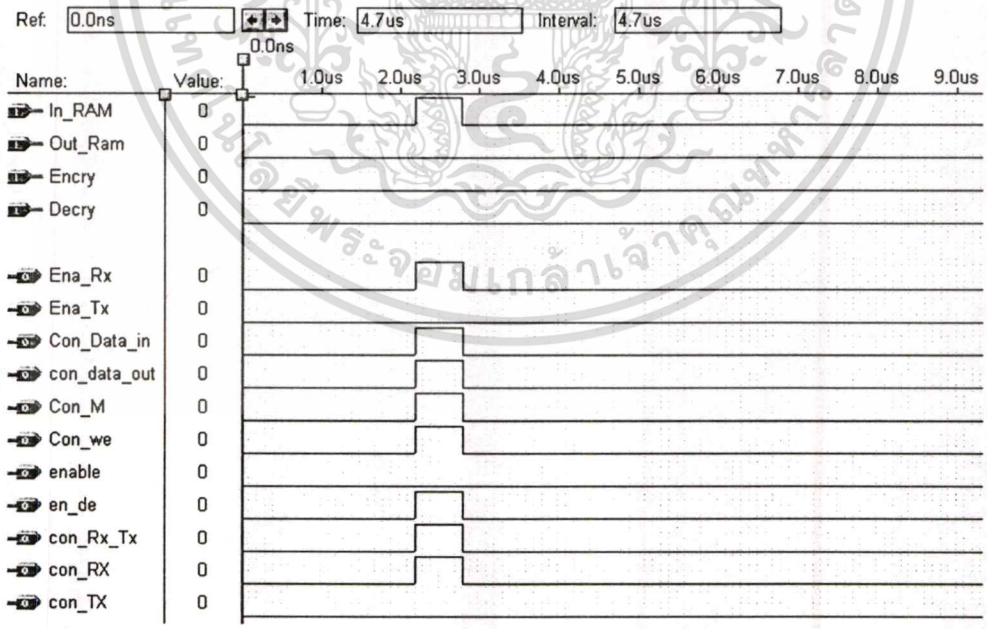
รูปที่ 7.25 แสดงส่วนของการควบคุมการเข้ารหัสและถอดรหัสข้อมูล(CON_A_KEY) โดยส่งค่าตำแหน่งของหน่วยความจำขนาด 8 บิต จำนวน 2 ชุดโดยชุดแรกใช้ร่วมกับสัญญาณ Slamping "CLK_R0" และ "CLK_W0" สำหรับอ่านข้อมูลจากหน่วยความจำเพื่อทำการเข้ารหัสหรือถอดรหัสข้อมูล และชุดที่สองจะใช้ร่วมกับสัญญาณ "CLK_W0" เพื่อที่จะนำข้อมูลที่ได้ทำการเข้ารหัสหรือถอดรหัสแล้วไปเก็บไว้ยังตำแหน่งเดิม

7.4.3 แสดงส่วนควบคุมการทำงานของระบบทั้งหมด

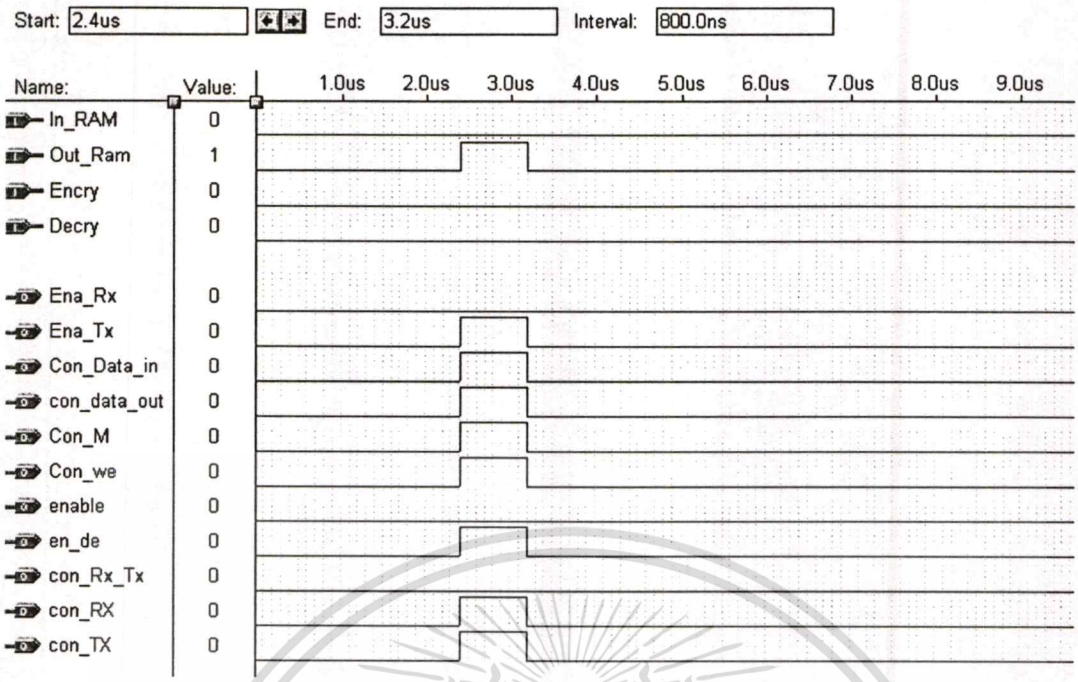
สามารถแสดงบล็อคของการทำงานได้ดังรูปที่ 7.26 และแสดง Timing Diagram ของแต่ละฟังก์ชันการทำงานได้ดังในรูปที่ 7.27, 7.28, 7.29 และ 7.30



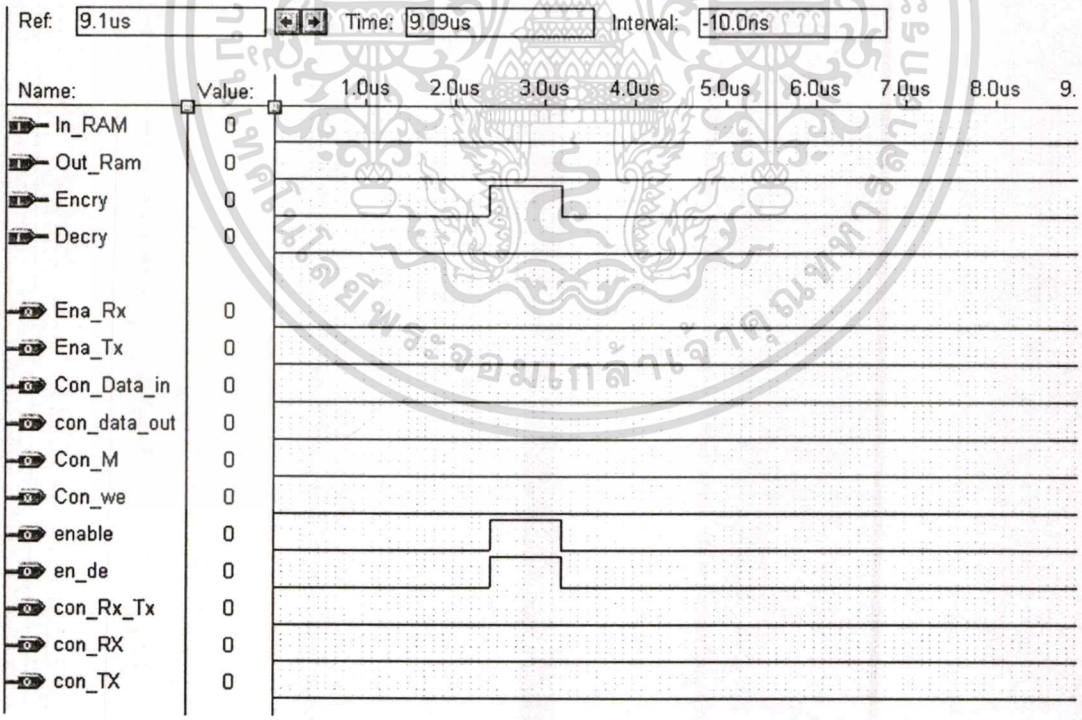
รูปที่ 7.26 แสดงบล็อคการทำงานของ Control_SYS



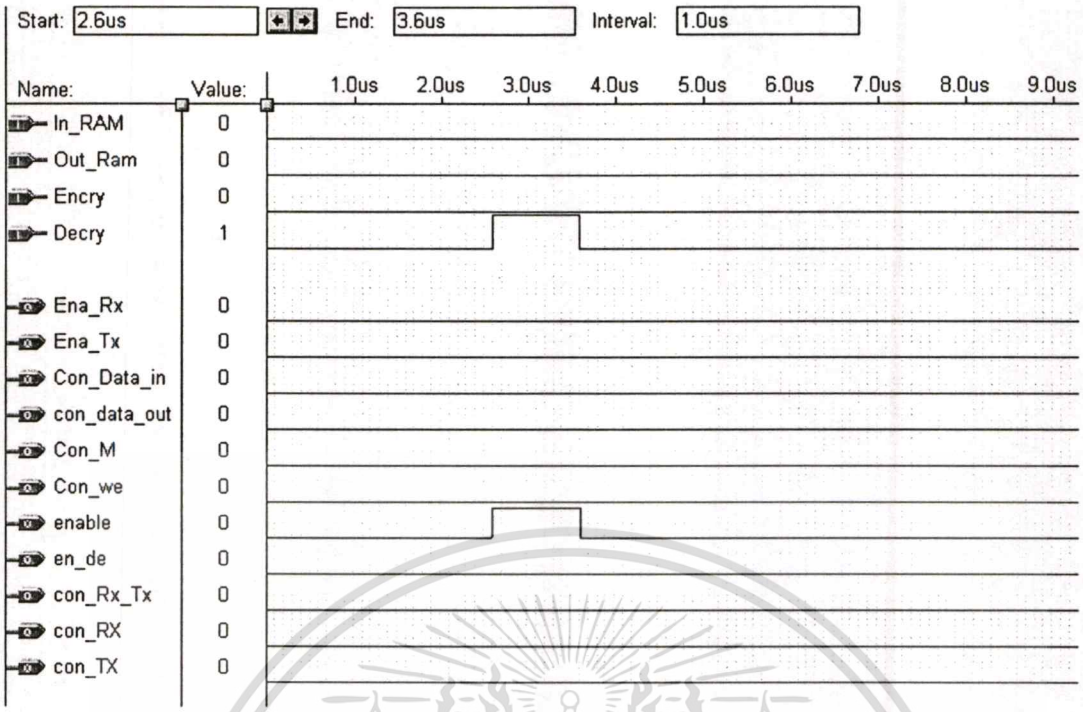
รูปที่ 7.27 แสดง Timing Diagram ของฟังก์ชันการรับข้อมูลจากคอมพิวเตอร์ (IN_RAM)



รูปที่ 7.28 แสดง Timing Diagram ของฟังก์ชันการส่งข้อมูลไปยังคอมพิวเตอร์ (OUT_RAM)



รูปที่ 7.29 แสดง Timing Diagram ของฟังก์ชันการเข้ารหัสข้อมูล (ENCRY)

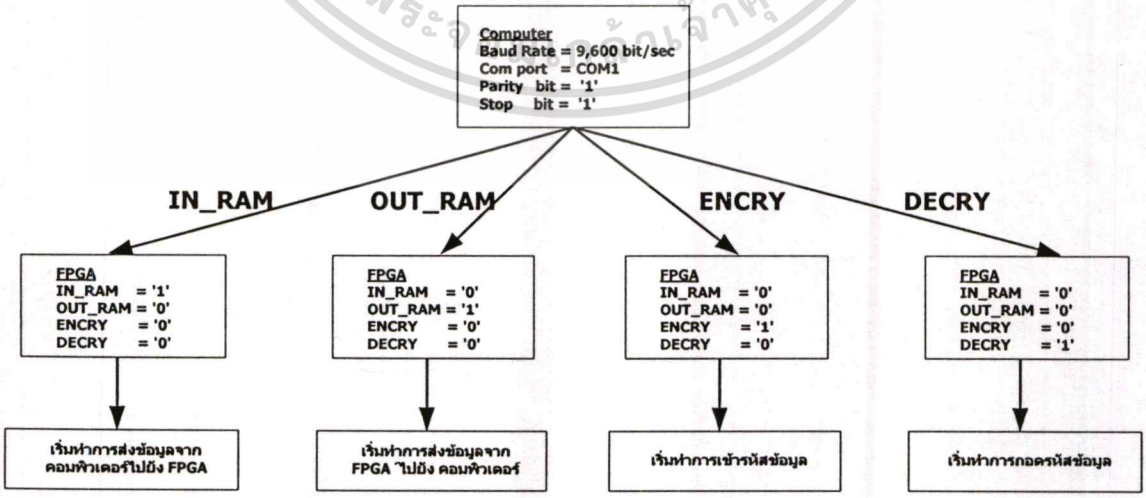


รูปที่ 7.30 แสดง Timing Diagram ของฟังก์ชันการถอดรหัสข้อมูล (DECRY)

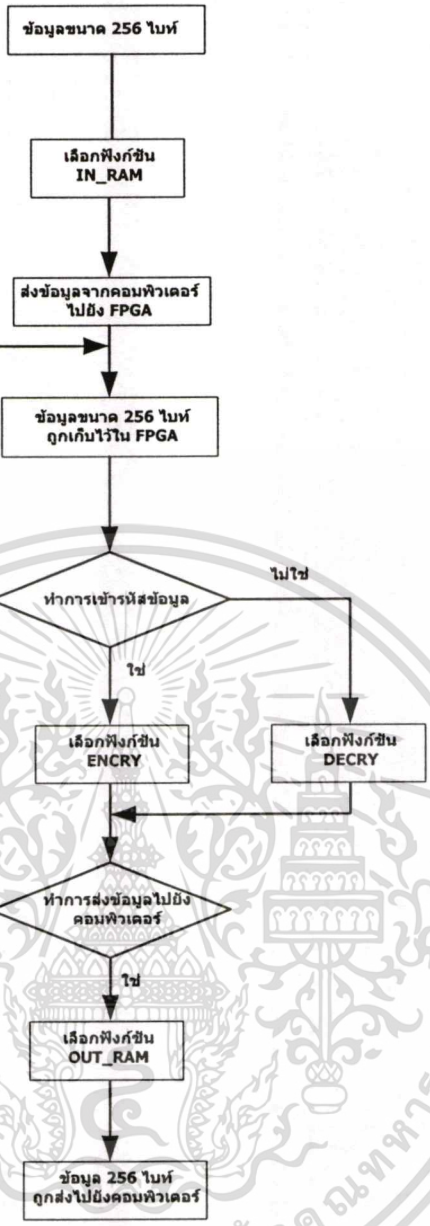
จากรูปที่ 7.27, 7.28, 7.29 และ 7.30 แสดงสัญญาณควบคุมเพื่อควบคุมให้สมาร์ทการ์ดสามารถทำงานตามฟังก์ชันดังกล่าวได้ตามลำดับ ซึ่งสามารถแสดงผลของการทดสอบได้ดังในหัวข้อที่ 7.5

7.5 การทดสอบการรวมส่วนประกอบแต่ละส่วนเป็นสมาร์ทการ์ด

เมื่อทำการรวมแต่ละส่วนประกอบที่ได้ทดสอบเป็นที่เรียบร้อยแล้วดังรูปที่ 6.1 สามารถแสดงขั้นตอนในการทำงานได้ดังรูปที่ 7.31



รูปที่ 7.31 แสดงการตั้งค่าบนคอมพิวเตอร์และบน FPGA ในแต่ละฟังก์ชัน



รูปที่ 7.32 แสดงขั้นตอนการทำงานของระบบการเข้ารหัสและถอดรหัสข้อมูลขนาด 256 ไบต์

จากขั้นตอนการทำงานของระบบในรูปที่ 7.32 สามารถแสดงผลการทดสอบพร้อมทั้งผลการเข้ารหัสและถอดรหัสข้อมูลขนาด 256 ไบต์ได้ดังนี้

The Computational cost of encryption is a barrier to wider appli cation of a variety of data security protocols. Virtually all resear ch on Elliptic Curve Cryptography(ECC) provides evidence to s uggest that ECC can provide family of encryption

รูปที่ 7.33 แสดงข้อมูลที่ใช้สำหรับการเข้ารหัส

ãª‡□*D ,ø*Üäå¯V□”Å□□ÄÙ□□+ª%H£ƒ□O□!hw-¦Ñ-’”’Ab
ßì—Vµèç¼□jÚ¹Gô8uÔU¡□□£□é□□□ìúÅøˆ□ ¨□□IsÀ…Ce©
ïa4□»dz□□□¾ó□UßPm°RŠ—□›ï_#ª-‡é«□Õ{ëqÂ˜zÅÔX?□„
□³_éšx□yÊ¶ºcœåŸš°”(„P— sI¤-ý9ÛÚ¢Ï□Û–q÷{.µçF*ÁD
□¹wÝ!Vu×[Ð6Uû□×□c(□¹ŠöHØƒÍY7·°{Ðãê□øFàÂpð·JK□î.
□pC=²Â□

รูปที่ 7.34 แสดงข้อมูลซึ่งได้เข้ารหัสเรียบร้อยแล้ว

จากการทดสอบรูปที่ 6.1 ได้เข้ารหัสข้อมูล โดยส่งข้อมูลขนาด 256 ไบต์ จากคอมพิวเตอร์ผ่าน RS 232 ไปยังสมาร์ทการ์ด สามารถเข้ารหัสและถอดรหัสได้ดี ซึ่งผลของการเข้ารหัสและถอดรหัสสามารถแสดงได้ดังรูปที่ 7.33 และ 7.34

ก)

ข)

รูปที่ 7.35 แสดงการทดสอบการเข้ารหัสข้อมูลรูปภาพขนาด 128 x 128 พิกเซล

ก) ข้อมูลรูปภาพต้นแบบขนาด 64 กิโลไบต์

ข) ข้อมูลรูปภาพต้นแบบขนาด 64 กิโลไบต์

ก)

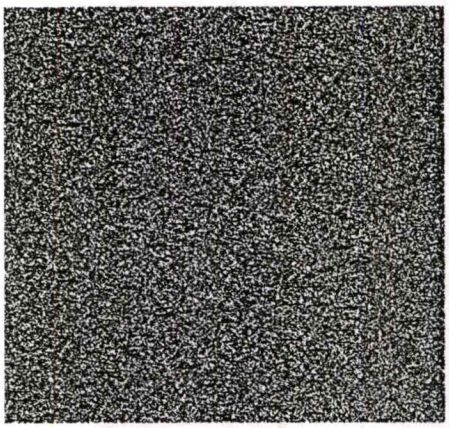
ข)

รูปที่ 7.36 แสดงการทดสอบการเข้ารหัสข้อมูลรูปภาพขนาด 256 x 256 พิกเซล

ก) ข้อมูลรูปภาพต้นแบบขนาด 263 กิโลไบต์

ข) ข้อมูลรูปภาพต้นแบบขนาด 263 กิโลไบต์

ก)



ข)

รูปที่ 7.37 แสดงการทดสอบการเข้ารหัสข้อมูลรูปภาพขนาด 512 x 512 พิกเซล
ก) ข้อมูลรูปภาพต้นแบบขนาด 1 เม็กกะไบต์
ข) ข้อมูลรูปภาพต้นแบบขนาด 1 เม็กกะไบต์

รูปที่ 7.35 7.36 และ 7.37 แสดงการทดสอบการนำข้อมูลรูปภาพมาทำการเข้ารหัส โดยทำการแปลงข้อมูลรูปภาพดังกล่าว เป็นไบนารีแล้วจึงนำไปเข้ารหัสบนสมาร์ตการ์ดทีละ 256 ไบต์ซึ่งเต็มความจุของฮาร์แวร์ ซึ่งข้อมูลในรูปที่ 7.35 และ 7.36 เป็นการแบ่งส่วนของข้อมูลรูปภาพในรูปที่ 7.37 โดยแต่ละรูปมีขนาด 64 กิโลไบต์ 263 กิโลไบต์ และ 1 เมกกะไบต์ตามลำดับ โดยจากการทดสอบพบว่าเมื่อนำข้อมูลทั้งหมดที่ได้จากการเข้ารหัสมาแสดงเป็นรูปภาพแล้ว ภาพที่ได้มีลักษณะผิดเพี้ยนไปจากภาพต้นแบบ ซึ่งจะไม่แสดงเค้าโครงภาพต้นแบบได้ ดังนั้นจึงไม่สามารถคาดเดาลักษณะของภาพต้นแบบได้

7.6 สรุป

จากการทดสอบนั้น ได้ทำการทดสอบส่วนประกอบทั้ง 4 ส่วนของวงจรคือ ส่วนการสื่อสารข้อมูล(UART Module) ส่วนของหน่วยความจำ(Memory Module) ส่วนของการเข้ารหัสและถอดรหัส (Cryptography Module) และส่วนของการควบคุม(Control Module) โดยในการทดสอบนั้นได้ทำการทดสอบร่วมกันทั้งบนซอร์ฟแวร์และฮาร์ดแวร์ ซึ่งแสดงผลในรูปของ Timing Diagram และทดสอบจริงบน FPGA พบว่า แต่ละส่วนประกอบสามารถทำงานได้ตรงกับความต้องการในการออกแบบทุกประการ หลังจากนั้นได้ทำการรวมแต่ละส่วนเข้าด้วยกัน เพื่อจำลองการทำงานของสมาร์ตการ์ด ตามฟังก์ชั่นที่ได้ทำการออกแบบไว้คือ ENCRY,DECRY,IN_RAM และ OUT_RAM และแสดงผลการทดสอบในรูปตัวอักษร จากการทดสอบข้อมูลขนาด 22 ตัวอักษร พบว่าข้อมูลที่ได้ ไม่สามารถแสดงในรูปข้อความเดิม และเมื่อทำการถอดรหัสข้อมูล สามารถได้ข้อมูลเดิมก่อนการเข้ารหัสกลับมา และเมื่อทำการเข้ารหัสข้อมูลขนาด 256 ไบท์ พบว่าข้อมูลที่ได้ไม่สามารถแสดง

ในรูปข้อความเดิม และเมื่อทำการถอดรหัสข้อมูลสามารถได้ข้อมูลเดิมก่อนการเข้ารหัสเช่นกัน และได้ทดสอบโดยทำการเข้ารหัสข้อมูลรูปภาพซึ่งแต่ละรูปมีขนาด 64 กิโลไบต์ 263 กิโลไบต์ และ 1 เม็กกะไบต์ตามลำดับ จากการทดสอบพบว่าเมื่อนำข้อมูลทั้งหมดที่ได้จากการเข้ารหัสมาแสดงเป็นรูปภาพแล้ว ภาพที่ได้มีลักษณะผิดเพี้ยนไปจากภาพต้นแบบ ซึ่งจะไม่แสดงเค้าโครงภาพต้นแบบได้ ดังนั้นจึงไม่สามารถคาดเดาลักษณะของภาพต้นแบบได้

บทที่ 8
สรุปผลการวิจัยและข้อเสนอแนะ

8.1 สรุป

การใช้งานสมาร์ทการ์ดกันอย่างแพร่หลาย ได้เกิดแนวที่ทำให้ความจุของข้อมูลเพิ่มมากขึ้น โดยที่ขนาดของฮาร์แวร์มีจำกัด เมื่อทำการพิจารณาพบว่า Public และ Private Key สามารถลดขนาดลงได้ และเมื่อทำการลดขนาดดังกล่าวลง เป็นผลให้สามารถใช้วิธีการเข้ารหัสข้อมูลแบบอื่นเพื่อทำให้ขนาดของข้อมูลที่เก็บบนสมาร์ทการ์ดเพิ่มมากขึ้นนั่นเอง

การออแบบวงจร ได้ทำการออกแบบวงจรด้วยภาษา VHDL ซึ่งเป็นภาษาที่ใช้ในการบรรยายหรืออธิบายรูปแบบการทำงานและความสัมพันธ์ของอุปกรณ์ฮาร์ดแวร์ทั้งในระดับเกทจนถึงระดับดิจิตอลที่ซับซ้อน ทั้งนี้เนื่องจาก VHDL เป็นภาษาที่เหมาะในการนำมาใช้ในการเลียนแบบการทำงานของอุปกรณ์ อีกทั้งยังมีความยืดหยุ่นและไม่ถูกจำกัดโดยความสามารถทางเทคโนโลยีใดๆ ทำให้ประหยัดเวลาและค่าใช้จ่ายในการออกแบบ จึงได้นำมาใช้กันอย่างกว้างขวางในวงการอุตสาหกรรม รูปแบบของภาษา VHDL ประกอบด้วย 2 ส่วนใหญ่ๆ ได้แก่ ส่วนของภาษาซีเควนเชียล (Sequential Language) และภาษาคอนเคอร์เรนท์(Concurrent Language) อีกทั้งการโปรแกรมด้วยภาษา VHDL สามารถเขียนได้ทั้งสองรูปแบบรวมกัน นอกจากนี้ตัวภาษายังสามารถอธิบายถึงการเชื่อมต่อระหว่างระบบย่อยเข้าด้วยกันเพื่อให้เป็นระบบใหญ่ได้และสามารถกำหนดรูปแบบไวยากรณ์(Syntex) อีกทั้งยังมีการตรวจสอบความหมายของภาษาว่าจะซิมูเลท(Simulate) ได้หรือไม่ เพราะโปรแกรมที่เขียนโดยวีเอชดีแอลต้องผ่านการซิมูเลทเพื่อตรวจสอบการทำงาน ฉะนั้นในการคอมไพล์(Compile) จะมีการตรวจสอบทั้งวงจรและซีมูเลชั่นซีแมนติก(Semantic) จึงทำให้สะดวกต่อการใช้งาน

ส่วนของ CRYPTOGRAPHY นั้นสามารถแบ่งการเข้ารหัสข้อมูลจะแบ่งออกเป็น 2 ประเภท คือ แบบSymmetric Encryption และแบบAsymmetric Encryption เนื่องจากระบบคณิตศาสตร์ของการ CRYPTOGRAPHY ได้แบ่งออกเป็น 3 ระบบด้วยกันคือ Integer Factorization System, Discrete Logarithm System และ Elliptic Curve System [ECC] โดยที่แต่ละระบบมีการใช้งานและระดับความปลอดภัยที่แตกต่างกัน ดังนั้นในวิทยานิพนธ์นี้ได้ใช้การเข้ารหัสข้อมูลแบบ Asymmetric Encryption เลือกระบบคณิตศาสตร์แบบ Elliptic Curve และส่วนของวิธีการในการเข้ารหัสข้อมูลได้เลือกวิธีการเข้ารหัสข้อมูลแบบ ElGamal ซึ่งเมื่อนำมาใช้รวมกับระบบคณิตศาสตร์ดังกล่าว จึงเรียกรวมกันว่า ElGamal Elliptic Curve และด้วยวิธีการนี้ทำให้ขนาดของ Public และ Private Key ลดลงเพราะความซับซ้อนของระบบคณิตศาสตร์แบบ Elliptic Curve ทำให้ความจุของข้อมูลเพิ่มมากขึ้น อีกทั้งยังสามารถใช้งานได้ในระดับความความปลอดภัยที่กำหนดเมื่อเทียบกับวิธีการของ RSA

ส่วนของคณิตศาสตร์ที่ใช้งาน ได้นำคณิตศาสตร์กาลัวฟิลด์มาใช้งาน โดยทำการออกแบบโอเปอร์เรชั่นทางคณิตศาสตร์ดังกล่าวในรูปของวงจร เพื่อใช้เป็นวงจรสำหรับการเข้ารหัสข้อมูลและถอดรหัสข้อมูล ประกอบด้วย วงจรคูณ และวงจรบวกแบบต่างๆ

ในการออกแบบและจำลองการทำงานของสมาร์ทการ์ด ได้ทำการออกแบบประกอบด้วยส่วนประกอบต่างๆ คือ ส่วนของการสื่อสารข้อมูล (UART Module) ส่วนของหน่วยความจำ (Memory Module) ส่วนของการเข้ารหัสและถอดรหัส (Cryptography Module) และส่วนของการควบคุม (Control Module)ภายใน FPGA พร้อมทั้งทำการทดสอบการทำงานแต่ละส่วนเพื่อให้ได้วงจรรวมที่มีขนาดเล็ก สามารถเข้ารหัสและถอดรหัสข้อมูลได้ดี และระดับความปลอดภัยอยู่ในระดับที่ใช้งานได้ ในวิทยานิพนธ์นี้ เมื่อได้ทำการสังเคราะห์เป็นวงจรระดับเกต แต่ละส่วนได้ใช้จำนวนเกตดังนี้ ส่วนของการสื่อสารข้อมูล(UART Module) 1,600 เกต ส่วนของหน่วยความจำ (Memory Module) 2,048 เกต ส่วนของการเข้ารหัสและถอดรหัส (Cryptography Module) 5,000 เกต และส่วนของการควบคุม (Control Module) 2,752 เกต

จากการทดสอบนั้น ได้ทำการทดสอบส่วนประกอบทั้ง 4 ส่วนของวงจรคือ ส่วนการสื่อสารข้อมูล(UART Module) ส่วนของหน่วยความจำ(Memory Module) ส่วนของการเข้ารหัสและถอดรหัส (Cryptography Module) และส่วนของการควบคุม(Control Module)โดยในการทดสอบนั้นได้ทำการทดสอบทั้งบนซอร์ฟแวร์ ซึ่งแสดงผลในรูปของ Timing Diagram และทดสอบจริงบน FPGA พบว่าแต่ละส่วนประกอบสามารถทำงานได้ตรงกับความต้องการในการออกแบบทุกประการ หลังจากนั้นได้ทำการรวมแต่ละส่วนเข้าด้วยกัน เพื่อจำลองการทำงานของสมาร์ตการ์ดและแสดงผลการทดสอบในรูปของตัวอักษร Front แบบต่างๆพบว่า สามารถแสดงการเข้ารหัสและถอดรหัสข้อมูลได้ถูกต้อง ตามฟังก์ชั่นที่ได้ทำการออกแบบไว้คือ ENCRY, DECRY, IN_RAM และ OUT_RAM พร้อมทั้งได้ทดสอบโดยทำการเข้ารหัสข้อมูลรูปภาพซึ่งแต่ละรูปมีขนาด 64 กิโลไบต์ 263 กิโลไบต์ และ 1 เม็กกะไบต์ตามลำดับ โดยทำการแปลงข้อมูลรูปภาพดังกล่าวเป็นไบนารี แล้วจึงนำไปทำการเข้ารหัสทีละ 256 ไบต์ จากการทดสอบพบว่าเมื่อนำข้อมูลทั้งหมดที่ได้จากการเข้ารหัสมาแสดงเป็นรูปภาพแล้ว ภาพที่ได้มีลักษณะผิดเพี้ยนไปจากภาพต้นแบบ ซึ่งจะไม่แสดงเค้าโครงภาพต้นแบบได้ ดังนั้นจึงไม่สามารถคาดเดาลักษณะของภาพต้นแบบได้

การทำงานวิจัยนี้มีข้อจำกัดหลายประการทั้งในด้านอุปกรณ์และเครื่องมือในการวิเคราะห์วงจรดิจิตอลขนาดใหญ่ที่รองรับการทำงานกับข้อมูลปริมาณมาก จึงทำให้ต้องหาวิธีการทดสอบแบบต่างๆ ดังกล่าวในบทที่ 7 การเข้ารหัสข้อมูล (Cryptography) ที่เลือกใช้ในงานวิจัยนี้ ทำให้ขนาดของ Public และ Private Key ที่ใช้งานมีขนาดลดลงโดยที่ระดับความปลอดภัยยังคงเท่าเดิม พร้อมทั้งยังทำให้ขนาดของข้อมูลที่ทำการเข้ารหัสแล้วมีขนาดเพิ่มขึ้นเป็น 2 เท่าเมื่อเทียบกับการเข้ารหัสข้อมูลของ ElGamal ดังนั้นในอนาคตอาจจะต้องมีการออกแบบฮาร์ดแวร์หรือซอร์ฟแวร์เพื่อใช้

สำหรับการบีบอัดข้อมูลก่อนการเข้ารหัสเพื่อทำให้ความจุของข้อมูลเพิ่มขึ้น พร้อมกับเพิ่มวงจรตรวจสอบความผิดพลาดของข้อมูลเพื่อให้งานวิจัยมีประสิทธิภาพมากยิ่งขึ้น

ดังนั้นงานวิจัยนี้ เป็นการออกแบบอุปกรณ์เข้ารหัสข้อมูล ซึ่งสร้างชิ้นงานขึ้นด้วยอุปกรณ์แยกส่วน มาเป็นระบบการเข้ารหัสข้อมูลที่สร้างบน FPGA โดยจำลองการทำงานเป็นสมาร์ทการ์ด เพื่อเป็นอุปกรณ์เก็บข้อมูลแบบพกพาที่เป็นความลับ เนื่องจากส่วนประกอบที่กำหนดโครงสร้างลอจิกบล็อคภายใน FPGA เป็นหน่วยความจำแรมแก้ไขได้ไม่จำกัดจำนวนครั้ง

ตารางที่ 8.1 แสดงคุณสมบัติของต้นแบบสมาร์ทการ์ด

Supply (Voltage)	12 Vdc
Clock Speed	6 MHZ
Interface Speed	9,600 bit /sec, Half -Duplex
Capacity	256 ไบท์
Input/Output	Serial Communication (RS232)

ตารางที่ 8.2 แสดงการเปรียบเทียบฮาร์ดแวร์ต้นแบบของสมาร์ทการ์ดในวิทยานิพนธ์กับต้นแบบของสมาร์ตการ์ด แบบอื่น[16,17]

Items	Design 1	Design 2	Design 3
Data Bus(bit)	8	8	32
Address Bus(bit)	8-10	16	No
Memory(Byte)	256-1K	64K	No
Interface	RS232	8	32
Frequency(MHZ)	6	10	10-15
Device (Gate)	11,400	No	17,000

จากตารางแสดงการเปรียบเทียบฮาร์แวร์ของต้นแบบสมาร์ทการ์ดที่ได้ทำการออกแบบ ซึ่ง Design 1 ได้ใช้ชิพ Altera FPGA EPF10K20RC240-4 โดยมีจำนวน 20,000 เกต Design 2 ได้ใช้ชิพ Xilinx FPGA Virtex XCV300 (PQ240) โดยมีจำนวน 322,970 เกต Design 3 ได้ใช้ชิพ Xilinx FPGA XC4020 โดยมีจำนวน 40,000 เกต เป็นการออกแบบในส่วนของ ALU บน GF(2^{155}) โดยไม่มีส่วนของหน่วยความจำเลย พบว่าขนาดของหน่วยความความจำที่ออกแบบแปรผันโดยตรงกับขนาดของหน่วยความจำบน FPGA ที่ใช้สำหรับออกแบบ การออกแบบ Data Bus และ Address Bus ขึ้นอยู่กับความเร็วที่ต้องการในระบบ ส่วนของการ Interface ขึ้นอยู่กับการออกแบบให้ติดต่อกับอุปกรณ์

ภายนอก โดยสามารถออกแบบเป็น Serial, Parallel ส่วน Frequency นั้น ถ้าใช้พลังงานต่ำและไม่ต้องการความเร็วในการประมวลผลสูงมากสามารถออกแบบเป็นความถี่ต่ำได้

Design 1 เป็นการออกแบบในวิทยานิพนธ์แสดงดังภาคผนวกที่ ง รูปที่ ง.1

Design 2 แสดงดังภาคผนวกที่ ง รูปที่ ง.2 [16]

Design 3 แสดงดังภาคผนวกที่ ง รูปที่ ง.3 [17]

8.2 ปัญหาที่พบในการทำวิทยานิพนธ์

ปัญหาส่วนใหญ่ที่พบในงานวิจัยคือ เครื่องมือในการออกแบบและสร้างวงจรบน FPGA ยังคงมีราคาสูง การจัดซื้อและจัดหาภายในประเทศยังยากลำบาก

8.3 วิธีการแก้ปัญหา

เครื่องมือในการวิจัย ได้รับการอนุเคราะห์จากภาควิชาอิเล็กทรอนิกส์และสถาบันวิจัยและพัฒนาอิเลคทรอนิกส์แห่งชาติ ซึ่งให้อุปกรณ์ FPGA มาทำการทดสอบ

8.4 ข้อเสนอแนะในการพัฒนา

งานวิจัยนี้ เป็นการปกปิดข้อมูลและนำข้อมูลมาเก็บไว้ในหน่วยความจำแบบพกพา ได้ทำการทดสอบโดยการส่งข้อมูลจากคอมพิวเตอร์ผ่าน RS232 ไปยังส่วนของการรับข้อมูลบน FPGA ในระยะทางที่กำหนดตามมาตราฐานของ RS232 ดังนั้นในกรณีที่ต้องส่งข้อมูลไปตามสายส่ง ที่มีระยะทางไกลมากหรือส่งข้อมูลในบริเวณที่มีสัญญาณรบกวนสูง ควรเพิ่มเติมส่วนของการตรวจสอบและแก้ไขข้อมูล เพื่อทำการตรวจสอบและแก้ไขข้อมูล ให้ได้ข้อมูลที่ครบถ้วนและถูกต้อง จึงทำให้การเข้ารหัสและถอดรหัสข้อมูลมีความถูกต้องแม่นยำมากยิ่งขึ้น

เนื่องจากฮาร์แวร์มีขนาดจำกัดจึงทำให้การออกแบบในแต่ละส่วนต้องมีคอมเพล็กซิตี้ต่ำสุด เพื่อที่จะทำให้สามารถเพิ่มขนาดของหน่วยความจำได้ ดังนั้นจึงต้องทำปรับปรุงส่วนหน่วยความจำโดยทำการเลือกโมเดลที่ใช้ในการออกแบบให้เล็กที่สุด และเมื่อทำการปรับปรุงทางด้านฮาร์แวร์แล้วจึงทำการปรับปรุงการบีบอัดข้อมูลจากคอมพิวเตอร์ก่อนการเข้ารหัสหรือถอดรหัส ซึ่งส่วนนี้นั้นสามารถปรับปรุงโดยทางฮาร์แวร์หรือทางซอร์แวร์ก็ได้

เอกสารอ้างอิง

1. Douglas R. Stinson. " Cryptography Theory and Practice " CRC Press, Inc 1995
2. Certicom. " The Elliptic Curve Cryptography ." 5520 Explorer Drive 4th Floor Mississauga, Ontario, Canada L4W 5L1, 1997,www.Certicom.com
3. MEMG YUAN HUANG. "Investigation of the Efficiency of Elliptic Curve Cryptography For Multi-application Smart Card. " 1998 Second International Conference on Knowledge-Based Intelligent Electronic System,21-23 April 1998, pp 318-322
4. S. Sutikno, A. Surya and R. Effendi, "An Implementation of ElGamal Elliptic Curves Cryptosystems", Int. Symposium on Intelligent Signal Processing and Communication Systems (ISPACS '99), Phuket, Thailand, Dec.1999.
5. Somsak Choomchuay. " On the Implement of Finite Field Operation " Ladkrabang Engineering Journal, Vol.11, No1, June 1994
6. ฟูศักดิ์ ชีวสุวิทย์. "การแก้รหัสผิด." คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง พ.ศ. 2528
7. ElsayedMohammed,A.E Emarah and Hh. El-Shennawy. "Elliptic Curve Cryptography on Smart Cards." Senior Member, IEEE Arab Academy for science and Technology, Air Defense College. 2001 IEEE, pp.213-222.
8. Adam D. Woodbury "Elliptic Curve Cryptography on Smart Cards without Coprocessors." Electric and Computer Engineering Department.
9. E.R. Berlekamp, "Bit-serial Reed-Solomon Encoder", IEEE Trans. on Information Theory, Vol. IT-28, Nov. 1982, pp. 869-874.
10. S. Choomchuay, "On the Implementation of Finite Field Basis Conversions", EECON-17, King Mongkut's Institute of Technology, North Bangkok, 1994, pp. 482-486.
11. S. Pongyupinpanich and S.Choomchuay, "Composite Field Bit Serial-Parallel Multiplier, Proc. of Information and Computer Engineering Workshop 2002 (ICEP2002), Prince of Songkla University, Jan. 2002, pp.65-69.
12. Christof Paar, "A New Architecture for a Parallel Finite Field Multiplier With Low Complexity Based on Composite Field", IEEE Trans. On Computers, Vol. 45, No. 7, July 1996.

เอกสารอ้างอิง(ต่อ)

13. G. Orlando and C. Paar, "A Super-serial Galois Field Multiplier for FPGAs and its Application to Public-Key Algorithms", 7th Annual IEEE Symp. on Field-Programmable Custom Computing Machines, Napa, CA, Apr. 1999.
14. J. Guajardo and C. Paar, "Fast Inverse Inversion in Composite Galois Filed $(2^n)^m$), ISIT 1998, Cambridge USA.
15. Zhi Li,Jhon Higgins, Mark Clement, "Performance of Finite Field Arithmetic in an Elliptic Curve Cryptography", Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on , 2001.
16. Sarwono Sutikno, Ronny Effendi, Andy Surya, "Design and Implementation of Arithmetic Processor F_{2155} for Elliptic Curve Cryptosystem", IEEE 1998.
17. Mao Jinliang, Shan Yu, Yang Xingzi, " Implement of An 8-bit Embedded Microprocessor For Smart Cards", IEEE 2001.
18. S. Choomchuay, S. Pongyupinpanich and K. Hardkhuntod, " An On-Chip Data Stroage with ElGamal Elliptic Curve Cryptosystem", ISCIT 2002 Thailand.
19. Douglas R. Stinson, " CRYPTOGRAPHY Theory and Practice ",Computer Science and Engineering Department and center for communication and Information Science University of Nebraska, Lincoln.
20. A.Menezes, P.van Oorscot and S. Vanstone, "Handbook of Applied Cryptography" ,CRC Press, 1996.
21. W. Rankl, W. Effing, " Smart card handbook ", translated by Chanterelle Translations .
22. R.E. Blahut, Theory and practice of Error Control Codes, Addision-Wesley, Reading, MA, 1983.

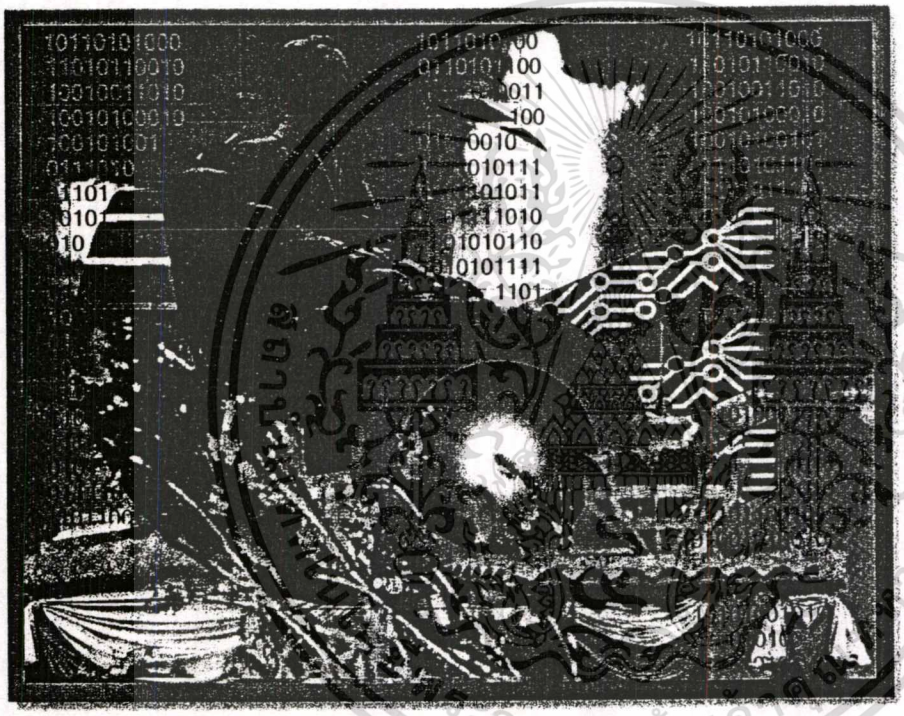
ภาคผนวก ก
ผลงานที่ได้รับการตีพิมพ์

1. S. Pongyupinpanich and S. Choomchuay . "Composite Field Bit Serial-Parallel Multiplier", **Proceedings Information and Computer Engineering Postgraduate Workshop 2002. (ICEP),** pp 65-69

Proceedings

Information and Computer Engineering Postgraduate Workshop 2002

ICEP2002



January 17-18, 2002

Department of Computer Engineering, Faculty of Engineering,
Prince of Songkla University, Hatyai, Songkhla, THAILAND 90112

Organised and sponsored by
Department of Computer Engineering,
Faculty of Engineering,
Prince of Songkla University, Thailand
http://www.coe.psu.ac.th

In cooperation with
IEEE ComSoc, Thailand Chapter
ISBN: 974-644-238-4

Composite Field Bit Serial – Parallel Multiplier

*S.Pongyupinpanich * and S.Choomchuay***

* Facility of Engineering, King Mongkut's Institute of Technology Ladkrabang
** Research Center for Communications and Information technology (ReCCIT), King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand
Email: kchsomsa@kmitl.ac.th, p_surapong2000@yahoo.com

ABSTRACT

An architecture for bit serial–parallel multiplication which operates in a composite field $GF(2^{176}) = GF((2^{11})^{16})$ is presented in this paper. The modularity designed of such a proposed structure makes it suitable and efficient for VLSI implementation. This architecture trades a reduction in resources with an increase in the number of clock cycles.

KEYWORDS

Composite Field , Finite Field , Multiplication.

1. INTRODUCTION

The application of finite field mathematics has increased its role in many modern digital communication systems. Typical areas of applications are cryptography schemes [1] and error correction codes such as Reed-Solomon code and other codes [2]. According to the development of VLSI design simplicity, many complicated signal processing algorithms have been mapped into hardware. In general to speed up the operation, reduction in both physical transistor size, algorithm complexity and circuit dimension. In most hardware implementation, a multiplier is considered to occupy major chip area. However, application that the size of the circuit is limited but not the operating speed, a serial multiplier is more preferable.

The general description of a multiplier in $GF(2^k)$ for any arbitrary k is firstly given in section 2. Its bit serial implementation is outlined in section 3 before the composite field multiplication and its implementation is detailed in section 4. For particular application, a multiplier in $GF(2^{11})^{16}$ is taken as an example case. Section 5 discusses the complexity of the architecture suggested in section 4. The generalization to other field size is also given before the paper is concluded.

2. GENERAL MULTIPLIER IN $GF(2^K)$

In the standard basis of element representation in $GF(2^k)$, let the polynomial $A(x)$ and $B(x)$ be defined as:

$$A(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1} \quad \text{and}$$

$$B(x) = b_0 + b_1 x + \ldots + b_{k-1} x^{k-1}$$

where a_i and b_i are field elements. The product of $A(x)$ and $B(x)$ denoted by $C(x)$ can be similarly defined $C(x) = c_0 + c_1 x + \ldots + c_{k-1} x^{k-1}$ where c_i are also field elements. $P(x)$ is the irreducible polynomial in such a field. The multiplication operation can be further explained as follows:

$$\begin{aligned} C(x) &= A(x) \cdot B(x) \bmod P(x) \\ &= A(x)\{b_0 + b_1 x + b_2 x^2 \ldots + b_{k-1} x^{k-1}\} \bmod P(x) \\ &= A(x) b_0 \bmod P(x) + A(x) b_1 x \bmod P(x) + \\ &\quad \ldots + A(x) b_{k-1} x^{k-1} \bmod P(x) \end{aligned} \quad \text{(1)}$$

From Equation (1), it can be noted that the multiplication is performed recursively for k steps (clock cycles) since the elements a_i and b_i are an k-bit signal. Each step gives an intermediate product.The hardware implementation results in k computation blocks cascaded to each other as shown in Fig.1.[7]

A basic computation cell comprises of 2 AND gates and a 3 input XOR gate. There are k cells which all are identical. Each is allocated for each coefficient. The delay time in each computing cell can be approximated to be the time that signal propagates through an AND gate and a XOR gate. Define T as computation period, S as a complexity and τ as a signal propagation time. Excluding of latches, the bit-serial multiplier yields,

$$T_{bit-serial} = k$$

$$S_{bit-serial} = k(2S_{AND} + S_{XOR})$$

$$\tau_{bit-serial} = k(\tau_{AND} + \tau_{XOR})$$

where τ denotes the gate propagation delay times.

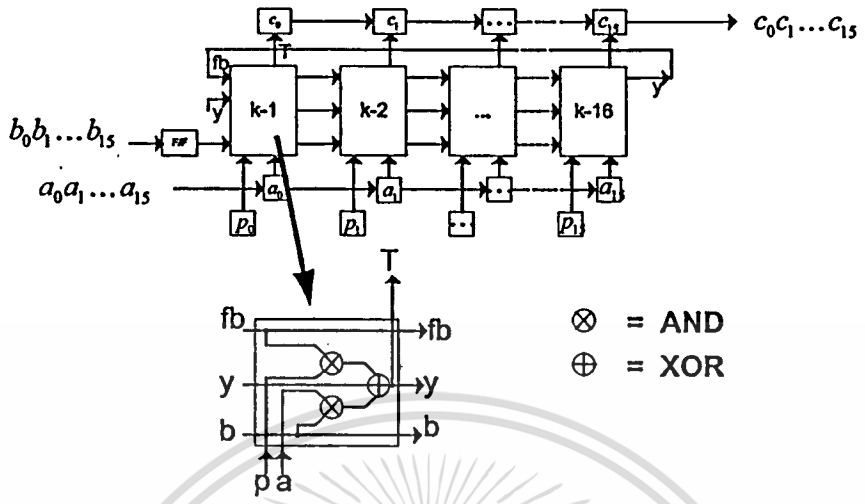


Fig.1 A General Purpose GF(2^{16}) standard basis multiplier

3. CONSTANT MULTIPLICATION IN GF (2^K)

A constant or fix coefficient multiplier is much simpler than a general purpose multiplier described in section 1 since many unnecessary terms can be made vanish. Consider an example given in a field GF(2^{16}) where $P(x) = x^{16} + x^{11} + x^6 + x^5 + 1$ is given as an irreducible polynomial. Let the arbitrary constant in such a field be $A(x) = \omega^{21} = x^{10} + x^5 + 1$. The multiplication of $A(x)$ and $B(x)$ denoted by $C(x)$ can be written as:

$$C(x) = A(x)B(x) \bmod P(x) = \omega^{21} B(x) \bmod (x^{16} + x^{11} + x^6 + x^5 + 1) \quad \text{—(3)}$$

Register B holds the value of $B(x)$ that will fed serially into the multiplier. The multiplier itself is a shift register type that the feed back path has been wired according to the generator polynomial, $P(x)$. The position where b_i fed into is determined by the constant term, which is $X^{10} + X^5 + 1$ in the above case. After k clock cycles, the multiplication result becomes available in register C.

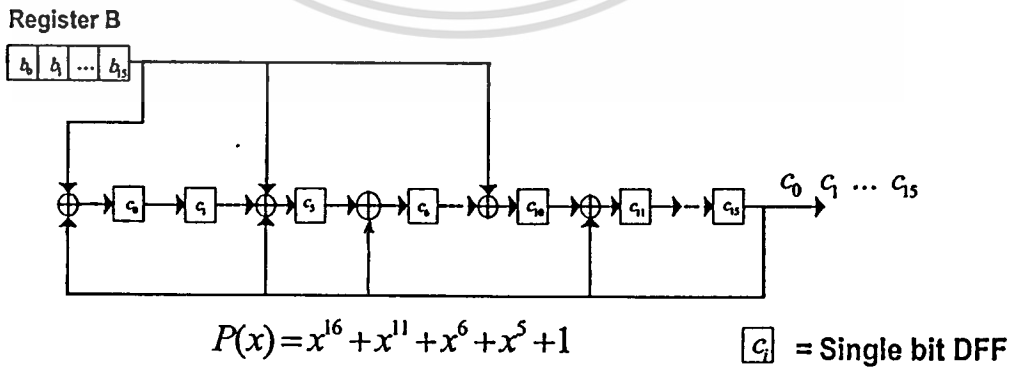


Fig.2 A constant multiplier, $A(x) = \omega^{21} = x^{10} + x^5 + 1$

4. THE COMPOSITE FIELD SERIAL - PARALLEL MULTIPLIER

Consider a Galois fields $GF(2^k)$ with k>1, an element A of a composite field $GF((2^k)^m)$ can be represented in the standard basis over $GF(2^k)$ as a polynomial with maximum degree of *m-1*. Similar to element in $GF(2^k)$, in polynomial form, let *A(y), B(y)* and *C(y)* be defined, ie.

$$A(y) = a_0 + a_1 y + a_2 y^2 + \ldots + a_{m-1} y^{m-1},$$

$$B(y) = b_0 + b_1 y + b_2 y^2 + \ldots + b_{m-1} y^{m-1} \quad \text{and}$$

$$C(y) = c_0 + c_1 y + c_2 y^2 + \ldots + c_{m-1} y^{m-1}.$$

Also be noted that $a_i, b_i, c_i \in GF(2^k)$ where

$$a_i = \sum_{p=0}^{k-1} \alpha_p u^p, \quad b_i = \sum_{p=0}^{k-1} \beta_p v^p \quad \text{and}$$

$$c_i = \sum_{p=0}^{k-1} \gamma_p w^p \; ; \; \alpha, \beta, \gamma \in 0,1.$$

Having Q(y) be an irreducible polynomial in $GF(2^m)$, the multiplication in a composite field $GF((2^k)^m)$ then can be written using the follow expression:

$$C(y) = B(y) \cdot A(y) \bmod Q(y)$$

$$C(y) = \{b_0 A(y) + y(b_1 A(y) + y(b_2 A(y) + y(b_3 A(y) + \ldots . y(b_{m-1} A(y)))..)\} \bmod Q(y) \quad \text{\_\_\_(4)}$$

In a particular case of GF(176) which can be written in the form of $GF(2^{11})^{16}$ and let $Q(y) = y^{16} + y^{11} + y^{10} + y^6 + 1$ be an irreducible polynomial with degree 16. Such a polynomial can be written in the way that

$$Q(y) = y^{16} + y^{11} + y^6 + y^5 + (y^{10} + y^5 + 1)$$
$$= y^{16} + y^{11} + y^6 + y^5 + \omega^{21}.$$

The multiplication resulted polynomial C(y) then becomes

$$C(y) = \{b_0 A(y) + y(b_1 A(y) + y(b_2 A(y) + y(b_3 A(y) + \ldots . y(b_{15} A(y)))..)\} \bmod Q(y) \quad \text{\_\_\_(5)}$$

$$C(y) = \{((..((b_{15} A(y))y + b_{14} A(y))y + (b_{13} A(y))y + (b_{12} A(y))y + \ldots . + (b_1 A(y))y + b_0 A(y)\} \bmod Q(y), \quad \text{\_\_\_(6)}$$

and we have

$$\langle A(y) y \rangle_{Q(y)} = D(y) = a_0 y + a_1 y^2 + \ldots + a_{15}(y^{11} + y^6 + y^5 + \omega^{21}) \quad \text{\_\_(7)}$$

The circuit that implements Equations (6) and (7) can be drawn as shown in Figure 3 below.

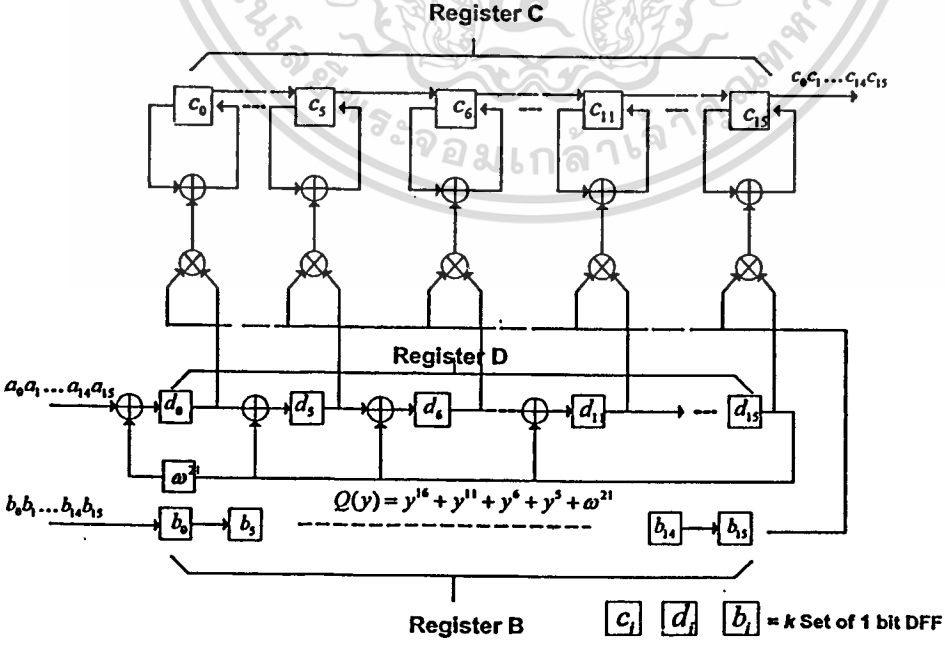


Fig.3 A Serial-Parallel multiplier in $GF(2^{11})^{16}$

Both *A(y)* and *B(y)* are serially shifted in to the multiplier unit, higher degree first. After 176 clock cycles, register B is loaded with B(y) while register D holds the partial product of $\langle y \square A(y) \rangle_{Q(y)}$ according to Equation (7). The following 176 clock cycles, the Equation (6) is fully implemented and the register C can be shifted out serially either ascending or descending order. However, since resulted data bits are available once the last clock is over. Register C can be also shitted out in a parallel manner. Note that circuit shown in Fig 2 has been incorporated in Fig 3 as a block with ω^{21} label.

Each of 16 element of register D is detailed in Fig 4. Each comprises of 11 flip-flops, feed back path is wired according to the irreducible polynomial $P(x) = x^{11} + x^2 + 1$.

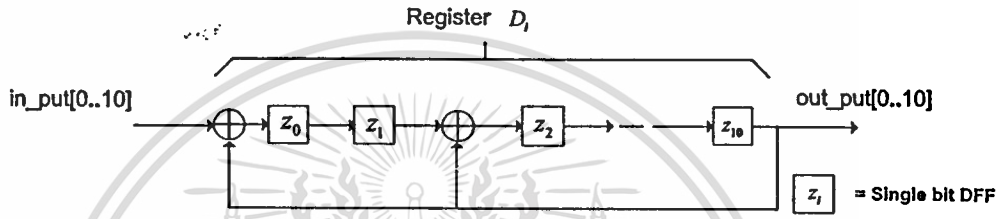


Fig 4. Detail diagram of each element of register D

5. COMPLEXITY

A composite filed multiplier denoted by equations (6) and (7) and its implementation given in Fig 3 and 4 can be elaborated in terms of complexity and latency below.

Circuit Complexity: The multiplier requires 3 registers, each of 176 bits. Each register is Divided into *m* sets, each of *k* bits. Minor hardware pieces such as XOR and AND gates can be assumed to 16 sets (*m* sets). Constant addition block can comprise of a single *m*-bit register and few XORs which could be neglected. Roughly estimation:

$$S = 3km_{DFF} + m_{XOR} + m_{AND} + m_{DFF}.$$

Latency: The multiplier requires 2×176 clock cycles, to multiply 2 elements defined in $GF((2^{11})^{16})$. To be generalized: $L = 2km$. The minimum clock period seems to be determined by the propagation delay time of a D flip-flop.

Let $N = k \times m$, the proposed serial-parallel multiplier detailed in section 4 above can be applied to other composite field. Based on the above estimation, the complexity can be tabulated as below.

Table 1 Complexity and latency of a serial-parallel multiplier derived for composite field $GF((2^k)^m)$; m =16, k = 3 up to 17

Composite Field			*Irreducible Polynomial, p(x)*	*Hardware Complexity*			*L*
N	*k*	*m*		*DFF*	*AND*	*XOR*	
48	3	16	$x^3 + x + 1$	160	16	16	96
80	5	16	$x^5 + x^2 + 1$	256	16	16	160
112	7	16	$x^7 + x^3 + 1$	352	16	16	224
176	11	16	$x^{11} + x^2 + 1$	544	16	16	352
208	13	16	$x^{13} + x^4 + x^3 + x + 1$	640	16	16	416
272	17	16	$x^{17} + x^3 + 1$	832	16	16	544

6. CONCLUSION

A bit serial-parallel multiplier that operates in a composite field GF(2^N) where $N = k \times m$ is proposed in this paper. Of its modularity designed, such a multiplier can have good impact in VLSI implementation either gate array or full custom layout. Compared to a bit parallel approach, a bit serial technique usually requires less complexity but suffering from long time execution. The low complexity of the bit serial-parallel multiplier is particularly useful for the application that requires low speed, low power and limited silicon area. Upon the current VLSI technology, the speed drawback can be much improved because of the smaller device feature size. In this implementation, the multiplier takes $2km$ clock cycle to deliver the output. Obviously no matter what the choice of k or m is. The computation time remains the same. On the theory side, the choice of k and m effect the irreducible in GF(2^k) and GF(2^m). On the implementation side, number of XOR and AND gates are depending on m while the number of DFF varies with $3mk + m$ or can be estimated to $3mk$ if $3k >> 1$. Careful selection of irreducible polynomial can effect in lower wiring complexity. Swapping of m and k can result in the slightly change of area complexity. For our application we have chosen m as an even for particular purpose. To verify the proposed mathematics and its corresponding hardware, the prototype design has been performed using VHDL and its simulation facility.

References

[1] A. Bouridane, M. Nibouche, O. Nibouche, D. Crookes and B. Albesher, "A Low Latency Bi Directional Serial-Parallel Multiplier Architecture", ISCAS 2000-IEEE International Symposium on Citcuit and Systems, May 28-31,2000,Geneva, witczerland

[2] Christof Paar, "A new Architecture for a Parallel Finite Field Multiplier with Low Complexity Based on Composite Field",IEEE Transactions on Computers, Vol. 45,No.7, July 1996

[3] PEKMESTIZI, K.Z., and CARISCOS, C.G: "A class of systolic serial-parallel multipliers", Int. J. Electronics 1994, Vol.76, No.3 pp. 463-468

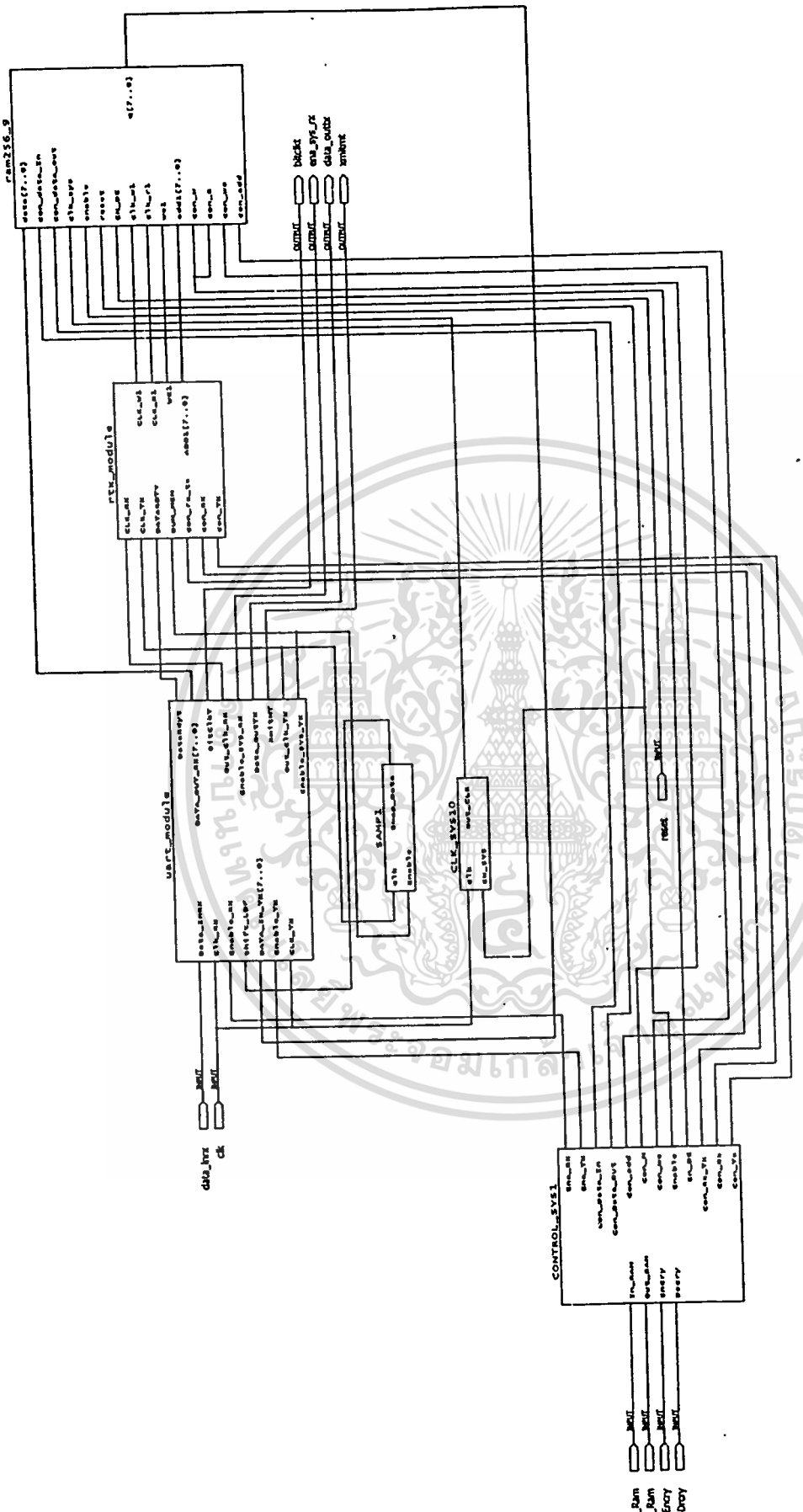
[4] CARAISCOS, C.G, AND PEKMESTZI, K. Z.; "Low-Latency Bit-Parallel Systolic VLSI Omplementation of FIR Digital Filter", IEEE Trans. Circuits Syst., 1996, Vol 43.,No. 7, pp.529-534

[5] AL-BESHER, B.,BOURIDANE, A.,ASUR, A.S., AND CROOKES D.: "A Hybrid Low-Latency Serial-Parallel Multiplier Architecture",Electronics Letter, 1997, Vol.34, No.2, pp.141-143

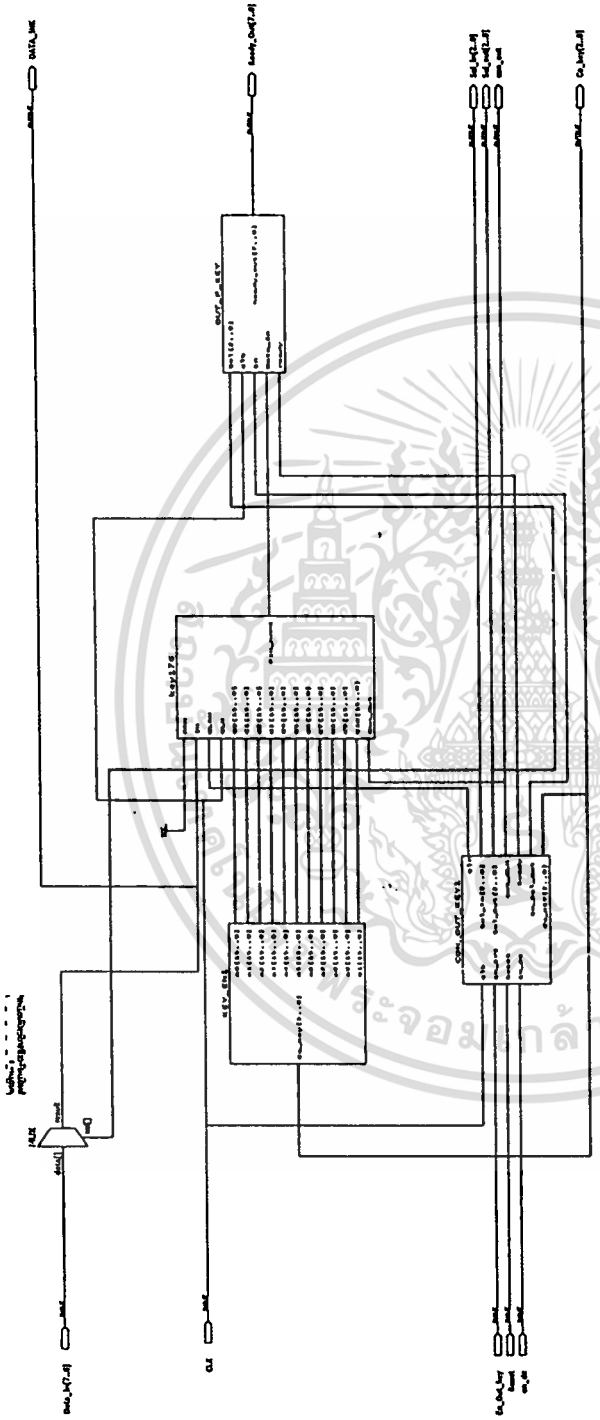
[6] E. Mastrovito, "VLSI Architecture for Computation in Galois Fields ", PhD thesis, Linkoping Univ. Dept. of Electrical Eng., Linkoping, Sweden, 1991

[7] Somsak Choomchuay, " On Implementation of Finite Field Operation ",Ladkrabang Engineering Journal, Vol. 11 No.1, June 1994 Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang.

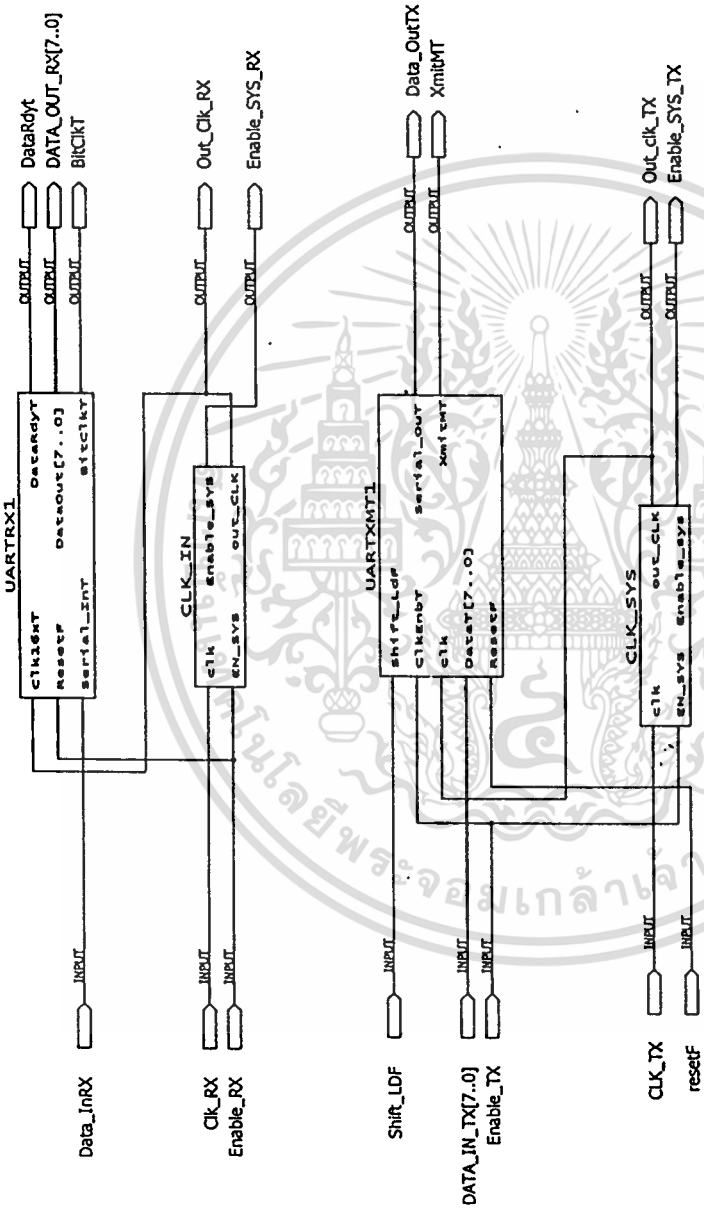
ภาคผนวก ข
วงจร FPGA และวงจรประกอบการทดลอง



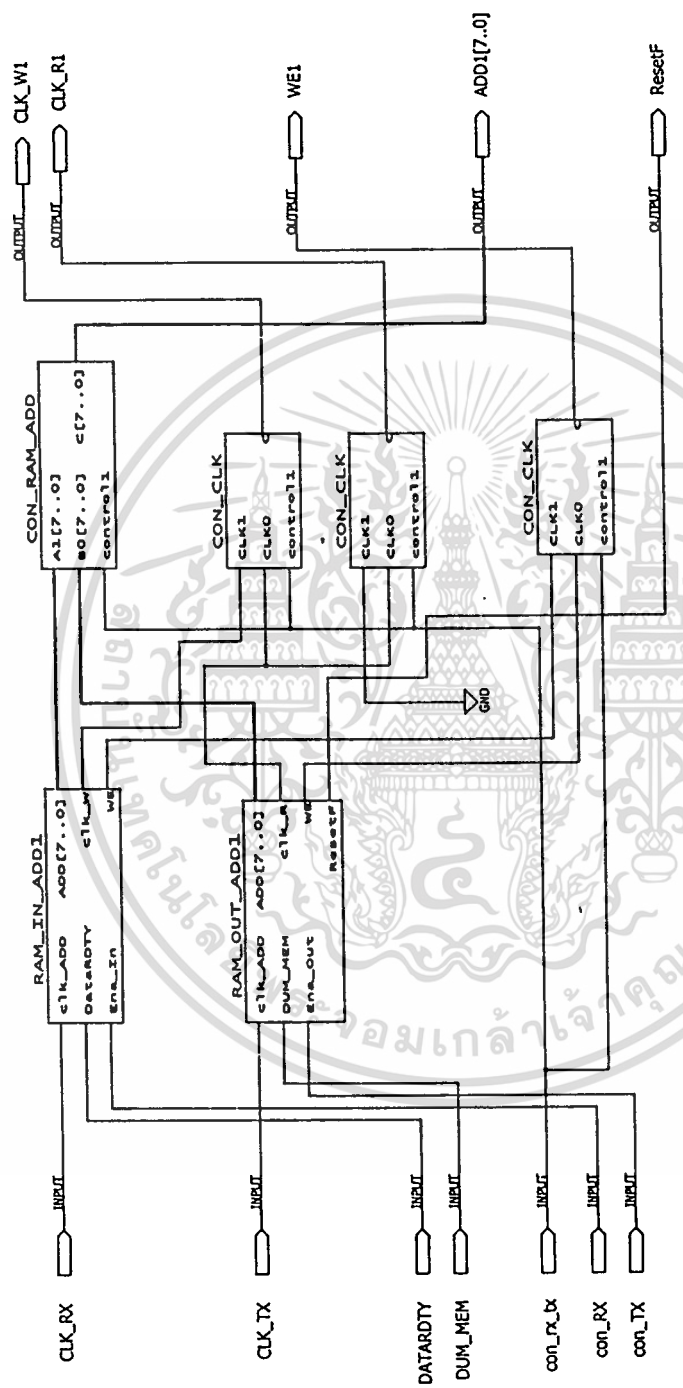
รูปที่ ข.1 แสดงการรวมวงจรแต่ละส่วนของสมาร์ตการ์ด



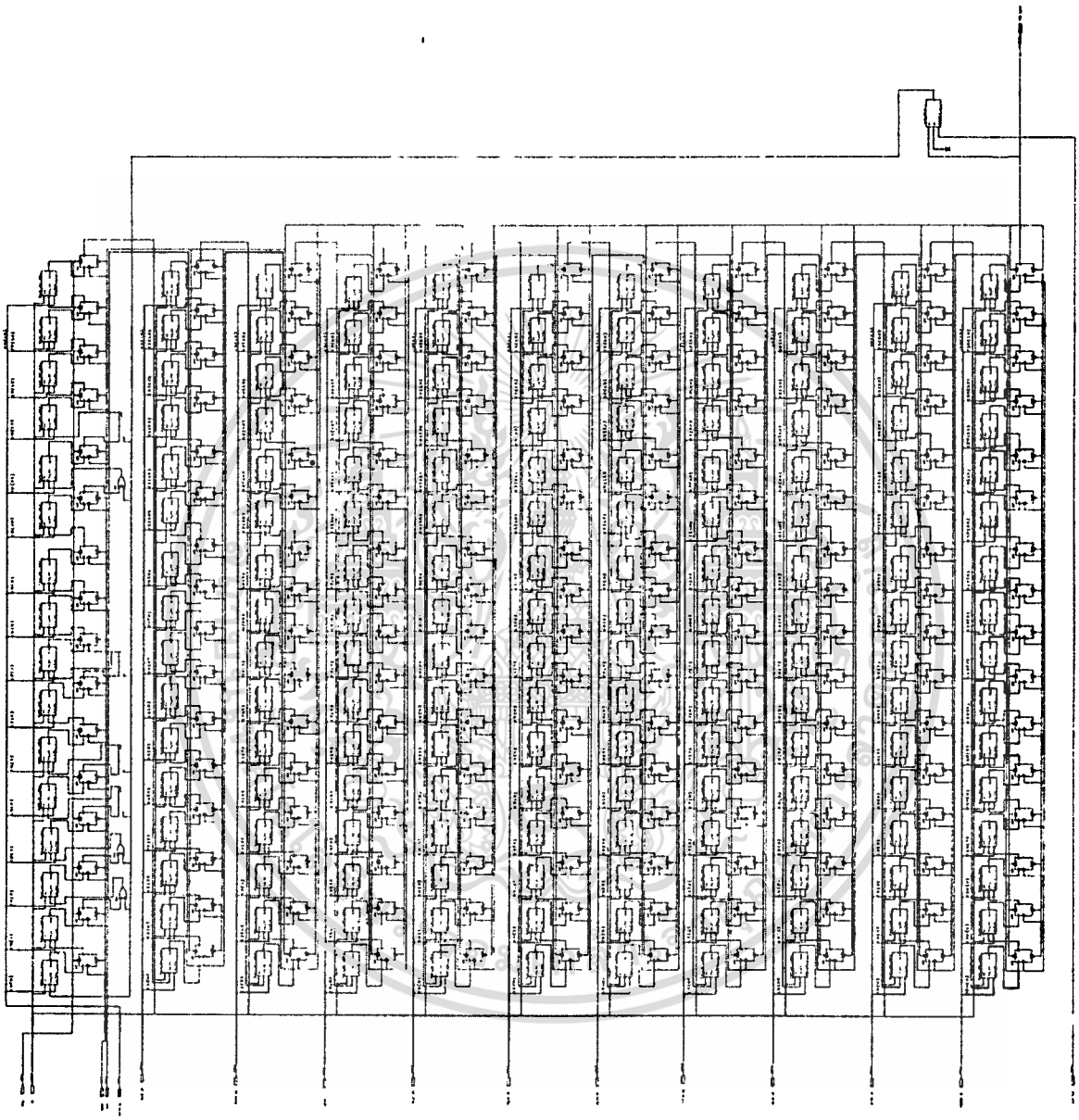
รูปที่ ข.2 แสดงวงจรการทำงานของ CRYPTOGRAPHY MODULE



รูปที่ ข.3 แสดงวงจรการทำงานของ UART MODULE



รูปที่ ข.4 แสดงวงรควบคุมการการอ่านข้อมูลจากหน่วยความจำ



รูปที่ ข.5 แสดงวงจรการทำงานของ Public-Key ขนาด 176 บิต

ภาคผนวก ค

อุปกรณ์ FPGA เบอร์ EPF10K20

Includes
FLEX 10KA

FLEX 10K

Embedded Programmable Logic Device Family

March 2001, ver. 4.1 Data Sheet

Features...

- The industry's first embedded programmable logic device (PLD) family, providing System-on-a-Programmable-Chip (SOPC) integration
  - Embedded array for implementing megafunctions, such as efficient memory and specialized logic functions
  - Logic array for general logic functions
- High density
  - 10,000 to 250,000 typical gates (see Table 1 and 2)
  - Up to 40,960 RAM bits; 2,048 bits per embedded array block (EAB), all of which can be used without reducing logic capacity
- System-level features
  - MultiVolt™ I/O interface support
  - 5.0-V tolerant input pins in FLEX® 10KA devices
  - Low power consumption (typical specification less than 0.5 mA in standby mode for most devices)
  - FLEX 10K and FLEX 10KA devices support peripheral component interconnect Special Interest Group (PCI SIG) *PCI Local Bus Specification, Revision 2.2*
  - FLEX 10KA devices include pull-up clamping diode, selectable on a pin-by-pin basis for 3.3-V PCI compliance
  - Select FLEX 10KA devices support 5.0-V PCI buses with eight or fewer loads
  - Built-in Joint Test Action Group (JTAG) boundary-scan test (BST) circuitry compliant with IEEE Std. 1149.1-1990, available without consuming any device logic

Table 1. FLEX 10K Device Features

Feature	EPF10K10 EPF10K10A	EPF10K20	EPF10K30 EPF10K30A	EPF10K40	EPF10K50 EPF10K50V
Typical gates (logic and RAM) *(1)*	10,000	20,000	30,000	40,000	50,000
Maximum system gates	31,000	63,000	69,000	93,000	116,000
Logic elements (LEs)	576	1,152	1,728	2,304	2,880
Logic array blocks (LABs)	72	144	216	288	360
Embedded array blocks (EABs)	3	6	6	8	10
Total RAM bits	6,144	12,288	12,288	16,384	20,480
Maximum user I/O pins	150	189	246	189	310

Table 2. FLEX 10K Device Features

Feature	EPF10K70	EPF10K100 EPF10K100A	EPF10K130V	EPF10K250A
Typical gates (logic and RAM) *(1)*	70,000	100,000	130,000	250,000
Maximum system gates	118,000	158,000	211,000	310,000
LEs	3,744	4,992	6,656	12,160
LABs	468	624	832	1,520
EABs	9	12	16	20
Total RAM bits	18,432	24,576	32,768	40,960
Maximum user I/O pins	358	406	470	470

Note to tables:

1) The embedded IEEE Std. 1149.1 JTAG circuitry adds up to 31,250 gates in addition to the listed typical or maximum system gates.

..and More Features

- Devices are fabricated on advanced processes and operate with a 3.3-V or 5.0-V supply voltage (see Table 3)
- In-circuit reconfigurability (ICR) via external configuration device, intelligent controller, or JTAG port
- ClockLock™ and ClockBoost™ options for reduced clock delay/skew and clock multiplication
- Built-in low-skew clock distribution trees
- 100% functional testing of all devices; test vectors or scan chains are not required

Table 3. Supply Voltages for FLEX 10K & FLEX 10KA Devices

5.0-V Devices	3.3-V Devices
EPF10K10 EPF10K20 EPF10K30 EPF10K40 EPF10K50 EPF10K70 EPF10K100	EPF10K10A EPF10K30A EPF10K50V EPF10K100A EPF10K130V EPF10K250A

- Flexible interconnect
  - FastTrack® Interconnect continuous routing structure for fast, predictable interconnect delays
  - Dedicated carry chain that implements arithmetic functions such as fast adders, counters, and comparators (automatically used by software tools and megafunctions)
  - Dedicated cascade chain that implements high-speed, high-fan-in logic functions (automatically used by software tools and megafunctions)
  - Tri-state emulation that implements internal tri-state buses
  - Up to six global clock signals and four global clear signals
- Powerful I/O pins
  - Individual tri-state output enable control for each pin
  - Open-drain option on each I/O pin
  - Programmable output slew-rate control to reduce switching noise
  - FLEX 10KA devices support hot-socketing
- Peripheral register for fast setup and clock-to-output delay
- Flexible package options
  - Available in a variety of packages with 84 to 600 pins (see Tables 4 and 5)
  - Pin-compatibility with other FLEX 10K devices in the same package
  - FineLine BGA™ packages maximize board space efficiency
- Software design support and automatic place-and-route provided by Altera development systems for Windows-based PCs and Sun SPARCstation, HP 9000 Series 700/800 workstations
- Additional design entry and simulation support provided by EDIF 2 0 0 and 3 0 0 netlist files, library of parameterized modules (LPM), DesignWare components, Verilog HDL, VHDL, and other interfaces to popular EDA tools from manufacturers such as Cadence, Exemplar Logic, Mentor Graphics, OrCAD, Synopsys, Synplicity, VeriBest, and Viewlogic

Table 4. FLEX 10K Package Options & I/O Pin Count *Note (1)*

Device	84-Pin PLCC	100-Pin TQFP	144-Pin TQFP	208-Pin PQFP RQFP	240-Pin PQFP RQFP
EPF10K10	59		102	134	
EPF10K10A		66	102	134	
EPF10K20			102	147	189
EPF10K30				147	189
EPF10K30A			102	147	189
EPF10K40				147	189
EPF10K50					189
EPF10K50V					189
EPF10K70					189
EPF10K100					
EPF10K100A					189
EPF10K130V					
EPF10K250A					

Table 5. FLEX 10K Package Options & I/O Pin Count (Continued) *Note (1)*

Device	503-Pin PGA	599-Pin PGA	256-Pin FineLine BGA	356-Pin BGA	484-Pin FineLine BGA	600-Pin BGA	403-Pin PGA
EPF10K10							
EPF10K10A			150		150 *(2)*		
EPF10K20							
EPF10K30				246			
EPF10K30A			191	246	246		
EPF10K40							
EPF10K50				274			310
EPF10K50V				274			
EPF10K70	358						
EPF10K100	406						
EPF10K100A				274	369	406	
EPF10K130V		470				470	
EPF10K250A		470				470	

lotes to tables:

) FLEX 10K and FLEX 10KA device package types include plastic J-lead chip carrier (PLCC), thin quad flat pack (TQFP), plastic quad flat pack (PQFP), power quad flat pack (RQFP), ball-grid array (BGA), pin-grid array (PGA), and FineLine BGA™ packages.

) This option is supported with a 256-pin FineLine BGA package. By using SameFrame pin migration, all FineLine BGA packages are pin compatible. For example, a board can be designed to support both 256-pin and 484-pin FineLine BGA packages. The Altera software automatically avoids conflicting pins when future migration is set.

ieneral)escription

Altera's FLEX 10K devices are the industry's first embedded PLDs. Based on reconfigurable CMOS SRAM elements, the Flexible Logic Element MatriX (FLEX) architecture incorporates all features necessary to implement common gate array megafunctions. With up to 250,000 gates, the FLEX 10K family provides the density, speed, and features to integrate entire systems, including multiple 32-bit buses, into a single device.

FLEX 10K devices are reconfigurable, which allows 100% testing prior to shipment. As a result, the designer is not required to generate test vectors for fault coverage purposes. Additionally, the designer does not need to manage inventories of different ASIC designs; FLEX 10K devices can be configured on the board for the specific functionality required.

Table 6 shows FLEX 10K performance for some common designs. All performance values were obtained with Synopsys DesignWare or LPM functions. No special design technique was required to implement the applications; the designer simply inferred or instantiated a function in a Verilog HDL, VHDL, Altera Hardware Description Language (AHDL), or schematic design file.

Table 6. FLEX 10K & FLEX 10KA Performance

Application	Resources Used		Performance				Units
	LEs	EABs	-1 Speed Grade	-2 Speed Grade	-3 Speed Grade	-4 Speed Grade	
16-bit loadable counter *(1)*	16	0	204	166	125	95	MHz
16-bit accumulator *(1)*	16	0	204	166	125	95	MHz
16-to-1 multiplexer *(2)*	10	0	4.2	5.8	6.0	7.0	ns
256 × 8 RAM read cycle speed *(3)*	0	1	172	145	108	84	MHz
256 × 8 RAM write cycle speed *(3)*	0	1	106	89	68	63	MHz

Notes:

(1) The speed grade of this application is limited because of clock high and low specifications.
(2) This application uses combinatorial inputs and outputs.
(3) This application uses registered inputs and outputs.

The FLEX 10K architecture is similar to that of embedded gate arrays, the fastest-growing segment of the gate array market. As with standard gate arrays, embedded gate arrays implement general logic in a conventional "sea-of-gates" architecture. In addition, embedded gate arrays have dedicated die areas for implementing large, specialized functions. By embedding functions in silicon, embedded gate arrays provide reduced die area and increased speed compared to standard gate arrays. However, embedded megafunctions typically cannot be customized, limiting the designer's options. In contrast, FLEX 10K devices are programmable, providing the designer with full control over embedded megafunctions and general logic while facilitating iterative design changes during debugging.

Each FLEX 10K device contains an embedded array and a logic array. The embedded array is used to implement a variety of memory functions or complex logic functions, such as digital signal processing (DSP), microcontroller, wide-data-path manipulation, and data-transformation functions. The logic array performs the same function as the sea-of-gates in the gate array; it is used to implement general logic, such as counters, adders, state machines, and multiplexers. The combination of embedded and logic arrays provides the high performance and high density of embedded gate arrays, enabling designers to implement an entire system on a single device.

FLEX 10K devices are configured at system power-up with data stored in an Altera serial configuration device or provided by a system controller. Altera offers the EPC1, EPC2, EPC16, and EPC1441 configuration devices, which configure FLEX 10K devices via a serial data stream. Configuration data can also be downloaded from system RAM or from Altera's BitBlaster™ serial download cable or ByteBlasterMV™ parallel port download cable. After a FLEX 10K device has been configured, it can be reconfigured in-circuit by resetting the device and loading new data. Because reconfiguration requires less than 320 ms, real-time changes can be made during system operation.

FLEX 10K devices contain an optimized interface that permits microprocessors to configure FLEX 10K devices serially or in parallel, and synchronously or asynchronously. The interface also enables microprocessors to treat a FLEX 10K device as memory and configure the device by writing to a virtual memory location, making it very easy for the designer to reconfigure the device.

For more information, see the following documents:

- *Configuration Devices for APEX & FLEX Devices Data Sheet*
- *BitBlaster Serial Download Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheet*
- *Application Note 116 (Configuring APEX 20K, FLEX 10K & FLEX 6000 Devices)*

FLEX 10K devices are supported by Altera development systems; single, integrated packages that offer schematic, text (including AHDL), and waveform design entry, compilation and logic synthesis, full simulation and worst-case timing analysis, and device configuration. The Altera software provides EDIF 2 0 0 and 3 0 0, LPM, VHDL, Verilog HDL, and other interfaces for additional design entry and simulation support from other industry-standard PC- and UNIX workstation-based EDA tools.

The Altera software works easily with common gate array EDA tools for synthesis and simulation. For example, the Altera software can generate Verilog HDL files for simulation with tools such as Cadence Verilog-XL. Additionally, the Altera software contains EDA libraries that use device-specific features such as carry chains which are used for fast counter and arithmetic functions. For instance, the Synopsys Design Compiler library supplied with the Altera development systems include DesignWare functions that are optimized for the FLEX 10K architecture.

The Altera development systems run on Windows-based PCs and Sun SPARCstation, and HP 9000 Series 700/800 workstations.

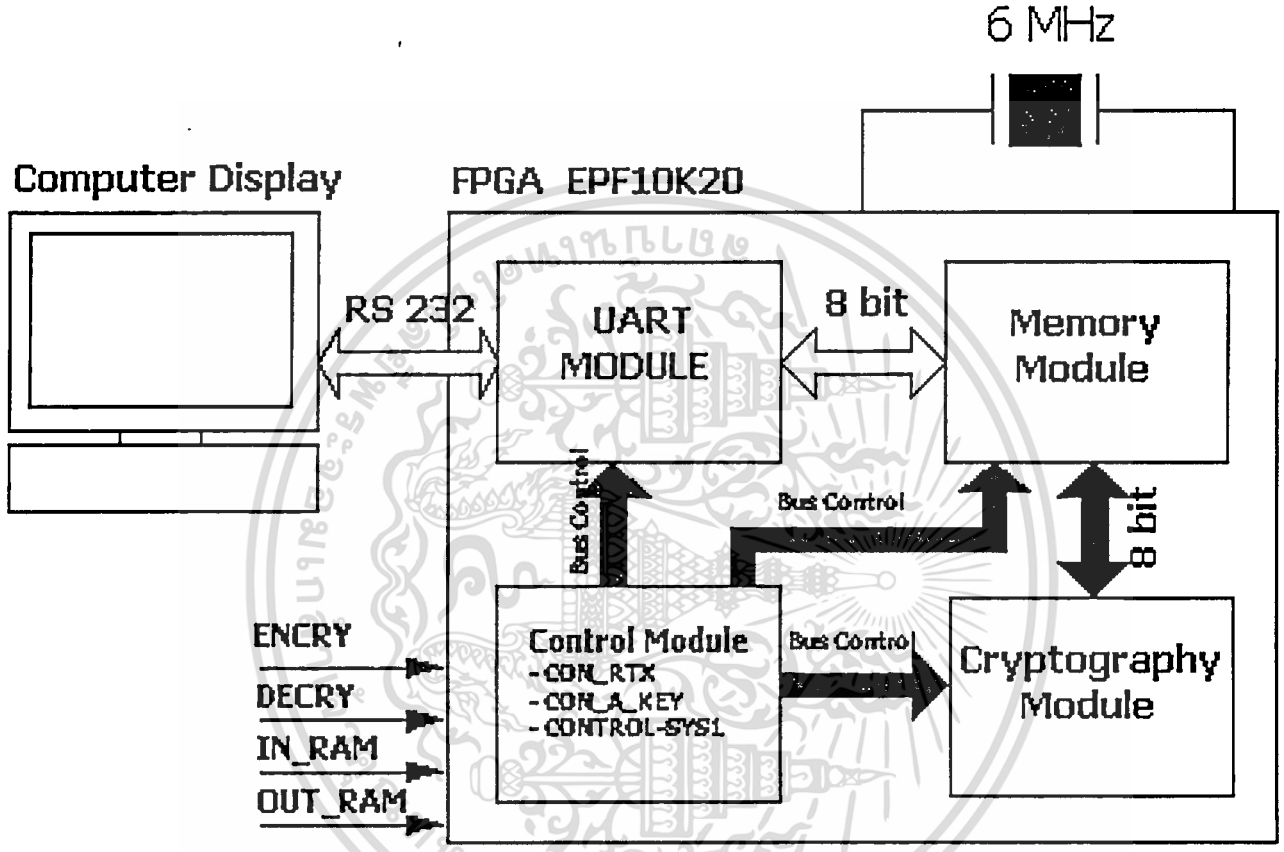
See the *MAX+PLUS II Programmable Logic Development System & Software Data Sheet* for more information.

nctional scription

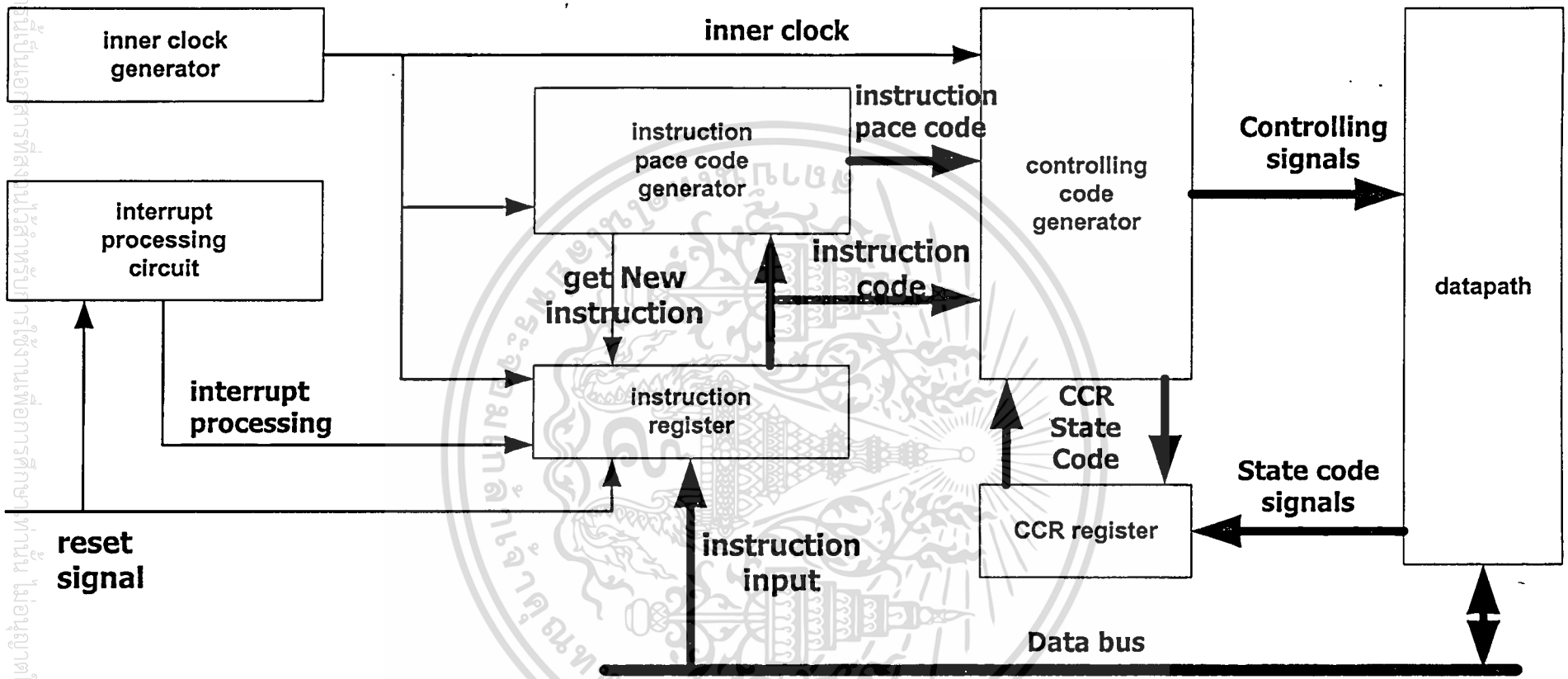
Each FLEX 10K device contains an embedded array to implement memory and specialized logic functions, and a logic array to implement general logic.

The embedded array consists of a series of EABs. When implementing memory functions, each EAB provides 2,048 bits, which can be used to create RAM, ROM, dual-port RAM, or first-in first-out (FIFO) functions. When implementing logic, each EAB can contribute 100 to 600 gates towards complex logic functions, such as multipliers, microcontrollers, state machines, and DSP functions. EABs can be used independently, or multiple EABs can be combined to implement larger functions.

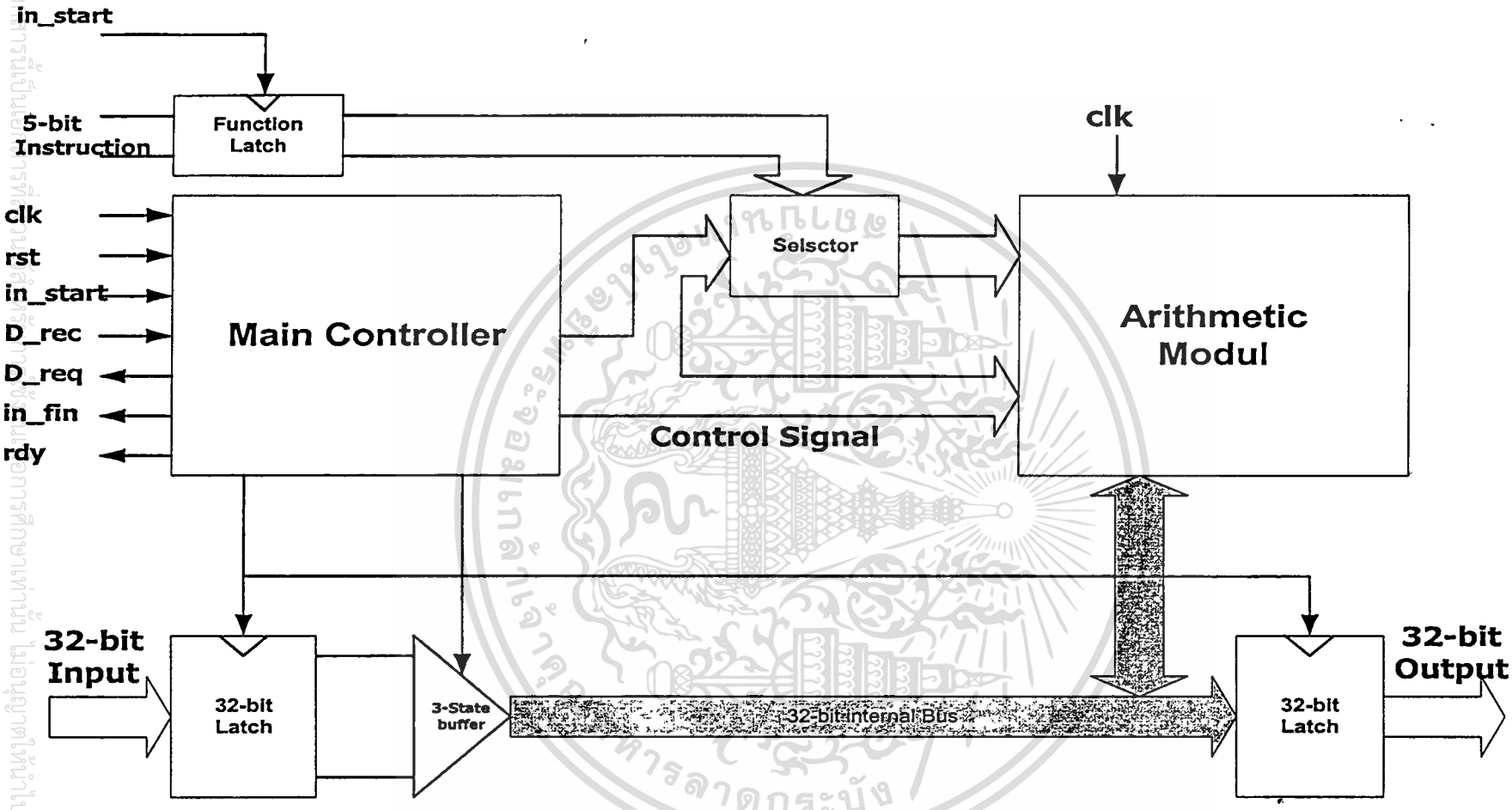
ภาคผนวก ง
แสดงวงจรของสมาร์ทการ์ด



รูปที่ ง.1 แสดงต้นแบบสถาปัตยกรรมของต้นแบบสมาร์ทการ์ดที่ได้ทำการออกแบบในวิทยานิพนธ์



รูปที่ ง.2 แสดงต้นแบบสถาปัตยกรรมของสมาร์ทการ์ดของงานวิจัย Design and Implementation of Arithmetic Processor F_2155 for Elliptic Curve Cryptosystem



รูปที่ ง.3 แสดงต้นแบบสถาปัตยกรรมของสมาร์ทการ์ดของงานวิจัย Implement of An 8-bit Embedded Microprocessor For Smart Cards

ประวัติผู้เขียน

นายสุรพงษ์ พงษ์ยุพินพานิช เกิดเมื่อวันที่ 17 มิถุนายน พ.ศ. 2519 ที่จังหวัด นครนายก สำเร็จการศึกษาวิศวกรรมศาสตรบัณฑิต(เทคโนโลยีการวัดคุมทางอุตสาหกรรม) จากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2541 และประกาศนียบัตรวิชาชีพชั้นสูง(ไฟฟ้ากำลัง) ปีการศึกษา 2539 จากสถาบันเทคโนโลยีราชมงคลวิทยาเขตนนทบุรี