การศึกษา FPGA เพื่อประยุกต์สำหรับการเข้ารหัสลับ

A STUDY OF FPGA APPLICATION FOR ENCRYPTION

นายภาณุวัฒน์ มณีวงศ์
นางสาววิภาวรรณ กฤษณ์นัน
นายสุทธิพัฒน์ แสนใจยา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2553

A STUDY OF FPGA APPLICATION FOR ENCRYPTION

PANUWAT MANEEWONG

WIPHAWAN KRITSANAN

SUTTHIPHAT SAENCHAIYA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์ การศึกษา FPGA เพื่อประยุกต์สำหรับการเข้ารหัสลับ

A STUDY OF FPGA APPLICATION FOR ENCRYPTION

นักศึกษาผู้จัดทำ นายภาณุวัฒน์ มณีวงศ์ รหัสนักศึกษา 50011172

นางสาววิภาวรรณ กฤษสนั่น รหัสนักศึกษา 50011474

นายสุทธิพัฒน์ แสนใจยา รหัสนักศึกษา 50011715

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2553

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รองศาสตราจารย์ วิริยะ กองรัตน์	

หัวข้อปริญญานิพนธ์	การศึกษา FPGA เพื่อประยุกต์สำหรับการเข้ารหัสลับ	
	A STUDY OF FPGA APPLICATION FOR ENCRYPTION	
นักศึกษาผู้จัดทำ	นายภาณุวัฒน์ มณีวงศ์	รหัสนักศึกษา 50011172
	นางสาววิภาวรรณ กฤษสนั่น	รหัสนักศึกษา 50011474
	นายสุทธิพัฒน์ แสนใจยา	รหัสนักศึกษา 50011715
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ วิริยะ กองรัตน์	
ปีการศึกษา	2553	

บทคัดย่อ

การสื่อสารข้อมูลเป็นที่ยอมรับโดยทั่วไป ว่ามีความสำคัญต่อการดำรงชีวิตประจำวันเป็นอย่างมาก แนวโน้มการแลกเปลี่ยนข้อมูลส่วนตัวมักจะส่งผ่านโครงข่ายของการสื่อสารในรูปแบบต่าง ๆ จุดนี้เองเป็นสาเหตุที่จะทำให้เกิดการรั่วไหลของข้อมูล เนื่องจากการทำการจารกรรมข้อมูลซึ่งเป็นความเสี่ยงต่อความปลอดภัยของข้อมูลและการละเมิดสิทธิส่วนบุคคล ในบทความนี้จึงได้เสนอเทคนิคการเข้ารหัสลับของข้อมูลก่อนส่งออกไปตัวกลางการสื่อสาร เพื่อป้องกันไม่ให้ข้อมูลรั่วไหล โดยเทคนิคการเข้ารหัสนี้ประกอบด้วยการประยุกต์ใช้ชิพดิจิตอลอเนกประสงค์ FPGA เป็นฮาร์ดแวร์ และใช้ภาษา VHDL เป็นซอฟแวร์ในการออกแบบการเข้ารหัสข้อมูล

Thesis Title A Study of FPGA Application for Encryption

Authors Mr. Panuwat Maneewong

Miss Wiphawan Kritsanan

Mr. Sutthihpat Saenchaiya

Thesis Advisor Assoc.Prof. Viriya Kongratana

Year 2010

ABSTRACT

Data communications are generally accepted as important to their daily life very much. In general, the exchange of personal information is often sent through the communication network in various forms. At this point, it will be cause the leakage of information. The personal information data lost will violently break out personal privacy. This article presents a technique for encrypting data before sending it out to the media of communication. To prevent the information from leakage, the encryption technique is applied before sending. Digital multi-chip FPGA using VHDL language and the hardware is implemented to encrypt the data.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปด้วยดีต้องขอขอบพระคุณท่านอาจารย์ที่ปรึกษา รองศาสตราจารย์ วิริยะ กองรัตน์ ที่ให้คำปรึกษาและคำแนะนำในการทำโครงงานนี้เป็นอย่างดีตลอดมา รวมไปถึงทุกท่านที่ได้ให้คำแนะนำและให้ความช่วยเหลือด้วยดีมาโดยตลอดซึ่งไม่ได้เอ่ยนามมา ณ ที่นี่ ที่ให้ความช่วยเหลือในด้านต่าง ๆ จนเสร็จสมบูรณ์

ขอขอบพระคุณคณาจารย์และบุคลากรภาควิชาวิศวกรรมการวัดคุม ซึ่งได้รับความกรุณาให้คำแนะนำเสนอแนะแนวทางการศึกษาค้นคว้า ให้คำปรึกษาและแก้ไขปรับปรุงข้อบกพร่องต่าง ๆ จนเป็นผลให้ปริญญานิพนธ์ฉบับนี้สมบูรณ์

สุดท้ายนี้ขอกราบขอบพระคุณ บิดา มารดา ครอบครัว ญาติพี่น้อง เพื่อน ๆ ทุกคนที่คอยเป็นห่วงและให้กำลังใจด้วยดีตลอดมา จนกระทั่งทำปริญญานิพนธ์สำเร็จลุล่วงได้ ประโยชน์อันพึงมีจากการศึกษาปริญญานิพนธ์ครั้งนี้ ขอมอบและอุทิศแด่บิดา มารดา บรรพบุรุษผู้ให้ชีวิตและทรัพย์สิน ครูอาจารย์ ผู้ประสิทธิประสาทวิชาความรู้แก่คณะผู้จัดทำตลอดจนผู้มีพระคุณทุกท่าน คณะผู้จัดทำรู้สึกซาบซึ้งเป็นอย่างยิ่งจึงใคร่ขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้

คณะผู้จัดทำ

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

ในปัจจุบันเราได้มีการพัฒนาทางด้านการสื่อสารอย่างรวดเร็ว ซึ่งการสื่อสารนับเป็นปัจจัยสำคัญอย่างหนึ่งในการพัฒนาประเทศของทุก ๆ ประเทศ โดยเมื่อประชากรเพิ่มขึ้นก็ย่อมมีการติดต่อสื่อสารกันมากขึ้นไม่ว่าจะเป็นในทางด้านธุรกิจการศึกษาและอีกหลายๆด้าน โดยการสื่อสารของผู้ใช้บริการที่เป็นประชาชนทั่วไปผ่านระบบโทรคมนาคมเหล่านี้ มักอาจจะถูกดักฟังได้ไม่ยาก หากบุคคลที่สามมีอุปกรณ์ที่ทันสมัย แต่สำหรับการสื่อสารของหน่วยงานราชการทางทหาร เพื่อเหตุผลในเรื่องความมั่นคงของประเทศมักจะมีระบบการสื่อสารที่ซับซ้อนกว่า เพราะจะต้องมีมาตรการรักษาความปลอดภัย และระบบป้องกันการดักฟังของผู้ไม่ประสงค์ดี โดยเฉพาะอย่างยิ่งในภาวะศึกสงครามการติดต่อแลกเปลี่ยนข่าวสารระหว่างหน่วยรบส่วนต่างๆกับศูนย์บัญชาการรบ จำเป็นต้องมีมาตรการที่ดีพอเพื่อป้องกันการดักฟังของฝ่ายข้าศึก อย่างไรก็ตามนับตั้งแต่ปลายทศวรรษ ค.ศ. 1970 เป็นต้นมาสถานการณ์ได้เปลี่ยนแปลงไปมากการติดต่อสื่อสารขององค์กรธุรกิจเอกชนในเชิงพาณิชย์ สถาบันการเงินหรือแม้กระทั่งการสื่อสารส่วนบุคคลเองก็เริ่มมีความต้องการระบบปกป้องข้อมูลข่าวสารของหน่วยงานหรือบุคคลให้เป็นความลับและปลอดภัยจากการจารกรรมข้อมูลของผู้ไม่ประสงค์ดี

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาแนวทางในการออกแบบชุดตัวเข้ารหัสและชุดตัวถอดรหัส โดยใช้ FPGA ในการออกแบบ
2. เพื่อศึกษาการเขียนภาษา VHDL ในการออกแบบวงจรทางลอจิก
3. เพื่อเป็นแนวทางในการพัฒนาในการออกแบบวงจรเข้ารหัส และถอดรหัสแก่ผู้ที่สนใจ
4. เพื่อนำไปใช้ในการป้องกันข้อมูลใดๆที่ต้องการให้เป็นความลับ อยู่ในความปลอดภัย

1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้นำเสนอการออกแบบการเข้ารหัสลับ โดยทำการสังเคราะห์ข้อมูลขึ้นเพื่อใช้ในการเข้ารหัสและถอดรหัส โดยในส่วนของวงจรที่เป็นดิจิตอลนั้นมีการนำ FPGA มาช่วยในการออกแบบวงจร โดยการออกแบบนั้นจะใช้ภาษา VHDL ในส่วนของการออกแบบจะใช้วิธีอธิบายพฤติกรรมการทำงานของวงจรด้วยภาษา VHDL วงจรที่เป็นดิจิตอลทั้งหมดจะทำการจำลองการทำงานรวมทั้งหมดสังเคราะห์วงจรลงบนอุปกรณ์ FPGA เพื่อใช้ทดสอบการทำงานและผลลัพธ์

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ใช้เป็นแนวทางในการออกแบบชุดตัวเข้ารหัสลับและชุดตัวถอดรหัสลับและนำมาทำการสร้างเป็นแผงวงจรแก่ผู้ที่สนใจ

2. ทำให้เข้าใจหลักการเขียนภาษา VHDL มากขึ้น

3. สามารถนำแผงวงจรตัวเข้ารหัสและตัวถอดรหัสลับที่ทำการออกแบบเชื่อมต่อ RS – 232 กับบอร์ดต่าง ๆ ของ FPGA ได้

บทที่ 2
ทฤษฎี

2.1 ระบบรหัสลับ

วัตถุประสงค์หลักของวิทยาการการเข้ารหัสลับ คือ การที่ทำให้บุคคลสองท่าน ได้แก่ นายเอ และนายบี สามารถติดต่อสื่อสารระหว่างกันผ่านช่องสัญญาณที่มีความไม่ปลอดภัยจากการดักฟัง (Insecure Channel) ได้โดยที่นายซีซึ่งเป็นบุคคลที่สามไม่อาจทราบถึงข้อความที่บุคคลทั้งสองสื่อสารกัน องค์ประกอบของระบบรหัสลับซึ่งสามารถแบ่งออกเป็นส่วนประกอบย่อย 5 ส่วน ดังนี้

1. ข้อความต้นฉบับ (Plaintext) หมายถึง ข้อความหรือข้อมูลต้นฉบับที่เราปรารถนาจะนำไปผ่านการเข้ารหัสลับ ซึ่งอาจจะบรรจุอยู่ในไฟล์ข้อมูลหรือไฟล์โปรแกรม

2. อัลกอริทึมการเข้ารหัส (Encryption Algorithm) หมายถึง กระบวนการหรือขั้นตอนที่ใช้ในการเปลี่ยนแปลงข้อความให้อยู่ในรูปแบบที่เปลี่ยนไปจากเดิม และผู้อื่นไม่สามารถเข้าใจได้ เช่น การแทนที่อักขระในข้อความต้นฉบับด้วยอักขระที่ต่างไปจากเดิม หรือการสลับตำแหน่งอักขระในข้อความต้นฉบับ เป็นต้น

3. กุญแจลับ (Key) คือ องค์ประกอบหนึ่งที่ป้อนให้กับอัลกอริทึมการเข้ารหัส เพื่อใช้ในการเข้ารหัสข้อความต้นฉบับ โดยผลลัพธ์ที่ได้จะแตกต่างกันออกไปตามกุญแจลับที่เลือกใช้ ทั้งนี้กุญแจลับเป็นค่าที่มีความเป็นอิสระไม่ขึ้นกับข้อความต้นฉบับ

4. ข้อความไซเฟอร์ (Cipher Text) เป็นผลลัพธ์ที่ได้จากการนำข้อความต้นฉบับไปผ่านกระบวนการเข้ารหัสลับโดยใช้กุญแจลัพธ์ตามที่ผู้ใช้กำหนด ข้อความไซเฟอร์ที่ได้จะเป็นข้อความที่ผู้อื่นไม่สามารถอ่านเข้าใจได้

5. อัลกอริทึมการถอดรหัส (Decryption Algorithm) หมายถึง กระบวนการหรือขั้นตอนที่ใช้ในการเปลี่ยนแปลงจากข้อความไซเฟอร์ให้กลับเป็นข้อความต้นฉบับตามเดิม โดยจะอาศัยกุญแจลับดอกเดียวกับที่ใช้ในขั้นตอนการเข้ารหัส

2.2 การจำแนกประเภทของระบบรหัสลับ

ระบบรหัสลับสามารถจำแนกออกเป็นประเภทต่าง ๆ ได้หลายรูปแบบ ตามแต่เกณฑ์ที่ใช้ในการแบ่ง ในที่นี้จะขอพิจารณาวิธีการแบ่ง 3 รูปแบบ ได้แก่ การเปลี่ยนแปลงจากข้อความต้นฉบับเป็นข้อความไซเฟอร์ จำนวนกุญแจที่ใช้ และการเข้ารหัสข้อความต้นฉบับ การเปลี่ยนแปลงจากข้อความต้นฉบับเป็นข้อความไซเฟอร์ สามารถกระทำได้ 2 รูปแบบ คือ การแทนที่ (Substitution) ซึ่งหมายถึง การเปลี่ยนแปลงส่วนประกอบในข้อความต้นฉบับ เช่น บิตตัวอักษร หรือข้อความ เป็นรูปแบบอื่นที่แตกต่างกันออกไปในข้อความไซเฟอร์ และการสลับเปลี่ยนตำแหน่ง (Permutation) ซึ่งเป็นการเปลี่ยน

ตำแหน่งของส่วนประกอบในข้อความต้นฉบับเท่านั้น โดยที่ส่วนประกอบในข้อความต้นฉบับยังคงเหมือนเดิมทุกประการ ข้อกำหนดสำคัญของการเปลี่ยนแปลงเหล่านี้ คือ ข้อมูลต้องไม่มีการสูญหาย นั่นคือ การเปลี่ยนแปลงเหล่านี้ต้องสามารถทำย้อนกลับได้

จำนวนกุญแจที่ใช้ ถ้าผู้ส่งและผู้รับข้อมูลใช้กุญแจตัวเดียวกันในการเข้ารหัสและถอดรหัส เราเรียกการเข้ารหัสประเภทนี้ว่า การเข้ารหัสแบบสมมาตร หรือการเข้ารหัสชนิดกุญแจลับ ถ้าผู้ส่งและผู้รับข้อมูลใช้กุญแจต่างกันในการเข้ารหัสและถอดรหัส เราเรียกการเข้ารหัสเช่นนี้ว่า การเข้ารหัสโดยใช้กุญแจสาธารณะ การเข้ารหัสข้อความต้นฉบับสามารถเข้ารหัสข้อความต้นฉบับได้ 2 วิธีคือ ไซเฟอร์แบบบล็อก (Block Cipher) หรือ ไซเฟอร์แบบสตรีม (Stream Cipher) ไซเฟอร์แบบบล็อก หมายถึง การแบ่งข้อความต้นฉบับที่ละบล็อก ส่วนการเข้ารหัสแบบสตรีม คือ การเข้ารหัสข้อความต้นฉบับทีละบิตนั่นเอง เนื่องจากการเข้ารหัสข้อความต้นฉบับทีละบิตจะใช้เวลานานกว่าการเข้ารหัสแบบบล็อกมาก ปัจจุบันอัลกอริทึมการเข้ารหัสที่ใช้ในคอมพิวเตอร์ส่วนใหญ่จึงเป็นการเข้ารหัสแบบบล็อกแทบทั้งสิ้น

2.3 รอม (ROM : Read Only Memory)

รอม คือ หน่วยความจำชนิดหนึ่งที่มีโปรแกรมหรือข้อมูลอยู่แล้ว และพร้อมที่จะนำมาต่อกับไมโครโปรเซสเซอร์ได้โดยตรง ซึ่งโปรแกรมหรือข้อมูลนี้จะไม่สูญหายไป แม้ว่าจะไม่มีการจ่ายไฟเลี้ยงให้กับระบบ ข้อมูลที่เก็บอยู่ในรอมจะสามารถอ่านค่าออกมาได้ แต่ไม่สามารถเขียนข้อมูลเข้าไปได้ เว้นแต่จะใช้วิธีการพิเศษซึ่งขึ้นกับชนิดของรอม ซึ่งชนิดของรอมสามารถแบ่งออกได้เป็น

2.3.1 แมนนวลรอม (Manual ROM)

ข้อมูลทั้งหมดที่อยู่ในรอมจะถูกโปรแกรมโดยผู้ผลิต (โปรแกรมมาจากโรงงาน) เราจะใช้รอมชนิดนี้เมื่อข้อมูลนั้นไม่มีการเปลี่ยนแปลง และมีความต้องการใช้งานเป็นจำนวนมาก ผู้ใช้ไม่สามารถเปลี่ยนแปลงข้อมูลภายในรอมได้ โดยรอมจะมีการใช้เทคโนโลยีที่แตกต่างกัน เช่น ไบโอโพลาร์ (Biopolar) ซีมอส (CMOS) เอ็นมอส (NMOS) และพีมอส (PMOS)

2.3.2 พีรอม (PROM : Programmable Read – Only Memory)

ข้อมูลที่ต้องการโปรแกรมจะถูกโปรแกรมโดยผู้ใช้เอง โดยป้อนพัลซ์แรงดันสูง (High Voltage Pulsed) ทำให้แผ่นโลหะหรือโพลีคริสตอลซิลิกอน (Polycrystaline Silicon) ที่อยู่ในตัวไอซีขาดออกจากกัน ทำให้เกิดเป็นลอจิก '1' หรือ '0' ตามตำแหน่งที่กำหนดในหน่วยความจำนั้น ๆ เมื่อพีรอมถูกโปรแกรมแล้ว ข้อมูลภายในจะไม่สามารถเปลี่ยนแปลงได้อีก หน่วยความจำชนิดนี้จะใช้ในงานที่ใช้ความเร็วสูง ซึ่งสูงกว่าหน่วยความจำที่โปรแกรมได้กว่าหน่วยความจำชนิดอื่น ๆ

2.3.3 อีพรอม (EPROM : Erasable Programmable Read – Only Memory)

ข้อมูลจะถูกโปรแกรมโดยผู้ใช้ โดยการให้สัญญาณที่มีแรงดันสูงผ่านเข้าไปในตัวอีพรอมซึ่งเป็นวิธีเดียวกันกับที่ใช้ในพรอม แต่ข้อมูลที่อยู่ในอีพรอมเปลี่ยนแปลงได้ โดยการลบข้อมูลที่อยู่ในอีพรอมออกก่อน แล้วค่อยโปรแกรมเข้าไปใหม่ การลบข้อมูลนี้ทำได้ด้วยการฉายแสงอุลตร้าไวโอเลตเข้าไปในตัวไอซีโดยผ่านทางกระจกใสที่อยู่บนตัวไอซีเมื่อฉากแสงครู่หนึ่ง (ประมาณ 5 – 10 นาที) ข้อมูลที่อยู่ภายในก็จะถูกลบทิ้ง ซึ่งช่วงเวลาฉายแสงนี้สามารถดูได้จากดาตาชีทมากับตัว อีพรอมและมีความเหมาะสมที่จะใช้ เมื่องานของระบบมีโอกาสที่จะปรับปรุงแก้ไขข้อมูลใหม่

2.3.4 อีเอพรอม (EAPROM : Electrically Alterable Read – Only Memory)

อีเอพรอมหรือเรียกอีกชื่อหนึ่งว่าอีอีพรอม (EEPROM : Electrical Erasable EPROM) เนื่องจากมีการใช้ไฟฟ้าในการลบข้อมูลในรอมเพื่อเขียนใหม่ซึ่งใช้เวลาสั้นกว่าของอีพรอมการลบขึ้นอยู่กับพื้นฐานการใช้เทคโนโลยีที่แตกต่างกัน ดังนั้นอีเอพรอมจะอยู่บนพื้นฐานของเทคโนโลยีแบบเอ็นมอส ข้อมูลจะถูกโปรแกรมโดยผู้ใช้เหมือนในอีพรอมแต่สิ่งที่แตกต่างก็คือ ข้อมูลของอีเอพรอมสามารถลบได้โดยทางไฟฟ้าไม่ใช่โดยการฉายแสงแบบอีพรอม

2.4 พื้นฐานการสื่อสารแบบอนุกรม

ถึงแม้ว่าการสื่อสารแบบอนุกรมในเครื่องคอมพิวเตอร์นั้น จะมีความเร็วในการสื่อสารช้ากว่าแบบขนาน ทั้งนี้ก็เพราะว่าการเคลื่อนย้ายข้อมูลแบบอนุกรมนั้นเป็นการส่งข้อมูลครั้งละ 1 บิต แต่พอร์ตขนานนั้นสามารถส่งข้อมูลได้ครั้งละหลาย ๆ บิตพร้อมกัน ส่งผลให้การสื่อสารข้อมูลแบบอนุกรมมีความเร็วต่ำกว่าแบบขนาน

แต่ว่าการส่งข้อมูลแบบอนุกรมนั้นมีข้อที่เหนือกว่าการส่งข้อมูลแบบขนาน คือ การสามารถส่งข้อมูลได้ในระยะทางที่ไกลกว่าแบบขนาน อีกทั้งสายสัญญาณที่ใช้ยังมีน้อยกว่าการส่งข้อมูลแบบขนานอีกด้วย การสื่อสารแบบอนุกรมสามารถแบ่งออกเป็น 3 รูปแบบ ดังนี้

1. Simplex สามารถส่งข้อมูลได้อย่างเดียว เป็นการสื่อสารแบบทางเดียว

2. Half – Duplex สามารถส่งข้อมูลไปยังปลายทาง และสามารถรับข้อมูลจากปลายทางได้ แต่ไม่สามารถทำการส่งและรับข้อมูลได้ในเวลาเดียวกัน

3. Full – Duplex สามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน

นอกจากนี้แล้วยังสามารถแบ่งประเภทของการสื่อสารแบบอนุกรมตามลักษณะสัญญาณในการส่งได้อีก 2 แบบคือ

1. การสื่อสารแบบซิงโครนัส(Synchronous)สำหรับการสื่อสารแบบซิงโครนัสนี้จะใช้สัญญาณนาฬิกาควบคุมการรับส่งสัญญาณ เช่น สายคีย์บอร์ดคอมพิวเตอร์ โดยจะมีสายสัญญาณเส้นหนึ่งเป็นสายสัญญาณนาฬิกา ส่วนอีกเส้นหนึ่งเป็นสายของข้อมูล (และมักจะมีสายกราวนด์ด้วย)

สำหรับการสื่อสารแบบซิงโครนัสนี้เหมาะสำหรับการทำงานในระยะใกล้ ข้อมูลที่จะส่งมีไม่มากนักเพราะถ้าระยะทางไกลขึ้นจะทำให้สัญญาณนาฬิกามีปัญหา อีกทั้งต้องมีสายหลายเส้นทำให้สิ้นเปลืองมาก

2. การสื่อสารแบบอะซิงโครนัส (Asynchronous) สำหรับการสื่อสารแบบอะซิงโครนัสนั้นจะใช้สายข้อมูลเพียงเส้นเดียว แต่จะใช้รูปแบบการส่งข้อมูลหรือ Bit Pattern เป็นตัวกำหนดว่าส่วนไหนเป็นส่วนเริ่มต้นข้อมูล, ส่วนไหนเป็นตัวข้อมูล, ส่วนไหนจะเป็นส่วนตรวจสอบความถูกต้องข้อมูล และส่วนไหนเป็นส่วนปิดท้ายของข้อมูล โดยต้องกำหนดให้สัญญาณนาฬิกาเท่ากันทั้งภาคส่งและภาครับ ซึ่งจะมีอุปกรณ์พิเศษที่ชื่อว่า UART หรือ Universal Asynchronous Receiver/Transmitter คอยควบคุมการรับ และส่งข้อมูล

แต่สำหรับมาตรฐานของการส่งข้อมูลแบบอนุกรมอีกแบบที่ได้รับความนิยมอย่างสูง ตั้งแต่อดีตถึงปัจจุบันโดยใช้งานกันอย่างแพร่หลายทั้งการสื่อสาร และการควบคุมทางอุตสาหกรรมนั่นก็คือ มาตรฐาน RS – 232C

2.5 UART

ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter หมายถึง อุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส (Asynchronous) ซึ่งเป็นส่วนหนึ่งในการสื่อสารอนุกรมแบบ Asynchronous

UART เป็นระบบการเชื่อมโยงที่เก่าแก่และเป็นวิธีสามัญที่สุด ข้อมูลจะถูกส่งออกจากขาส่งสัญญาณ TXD ของ UART เป็นลำดับไปด้วยความเร็วที่กำหนดค่าไว้ล่วงหน้า สัญญาณที่ส่งเลียงออกไปจะมีค่าเป็นศูนย์หรือหนึ่ง ขารับสัญญาณ RXD จะมีหน้าที่คอยตรวจสอบว่ามีสัญญาณส่งมาให้ที่ความเร็วเดียวกับขาส่งสัญญาณ ข้อมูลจะถูกส่งออกไปหนึ่งไบต์ในแต่ละครั้ง อันนี้เป็นการส่งสัญญาณทางเดี่ยว ในการส่งสัญญาณทางคู่เราจะต้องติดตั้งวงจรแบบเดียวกันที่จุดตรงข้าม การส่งและรับสัญญาณจะต้องทำบนวงจรที่แยกจากกัน และเชื่อมโยงทำงานติดต่อกันโดยอิสระไม่ขึ้นต่อกัน ในแต่ละฟากจะสามารถรับหรือส่งสัญญาณไปให้อีกฝ่ายได้พร้อมในเวลาเดียวกัน

เราเรียกค่าความเร็วสำหรับรับ/ส่งข้อมูลที่ตั้งไว้ล่วงหน้าว่า Baud Rate ค่าความเร็ว Baud Rate นี้คือตัวบ่งบอกว่าเราจะส่งข้อมูลมากได้เท่าไรในเวลาหนึ่งวินาที สำหรับค่ามาตรฐานของ Baud Rate มีหลายค่าด้วยกันดังเช่น 9600, 19200, 115200 และอื่น ๆ

เราสามารถเชื่อมโยงหน่วยประเมินผลสองตัวเข้าด้วยกันโดยตรง และใช้การติดต่อแบบ UART ทำได้โดยต่อขาสัญญาณส่งข้อมูล TXD จากหน่วยประเมินผลที่หนึ่งไปสู่ขารับสัญญาณข้อมูล RXD ของหน่วยประเมินผลที่สอง และทำการต่อขารับ/ส่งข้อมูลสลับกับที่ทำมาแล้ว เนื่องจากขาต่อรับส่งสัญญาณข้อมูลเป็นดิจิตอล ระดับค่าความต่างศักย์บนขารับส่งสัญญาณ TXD/RXD ของ UART จะวัดค่าเป็น 0V (ต่ำ) และ 3.3V หรือ 5V (สูง)

ระบบเชื่อมโยงที่ใช้ในอุตสาหกรรมทั้งหลายหรือการต่อสายสัญญาณที่ยาวมาก ค่าความต่างศักย์ 3.3V หรือแม้แต่ 5V จะไม่ทำให้การทำงานเป็นไปโดยราบรื่นมากนัก มีมาตรฐานที่ระบุวิธีการที่เราสามารถใช้สัญญาณมาตรฐาน UART มาทำการแปลงค่าความต่างศักย์ให้สูงขึ้น หรือเปลี่ยนค่าของสัญญาณเพื่อทำให้การตอบรับสัญญาณมีความเชื่อถือได้มากขึ้น

อุปกรณ์สำคัญตัวหนึ่งที่ใช้กันมากคือ RS232 เครื่องสมองกลส่วนใหญ่จะมีพอร์ตอุปกรณ์ RS232 ติดตั้งไว้ให้แล้ว การรับส่งสัญญาณด้วย RS232 โดยข้อมูลเป็น UART แต่ระดับค่าความต่างศักย์จะถูกแปลงจาก TTL (0V to 3.3V) เป็นระดับค่าความต่างศักย์ RS232 (-12V ถึง +12V)

ข้อเท็จจริงที่สำคัญอันหนึ่งเกี่ยวกับ RS232 คือว่าระดับค่าความต่างศักย์จะเป็นตรงกันข้ามกับความคิดเชิงตรรกะของเรา ไมโครชิพที่ใช้สำหรับปรับระดับสัญญาณ TTL เป็นระดับสัญญาณ RS232 มีหลายชนิดด้วยกันดังเช่น MAX232 หรือ MAX3232 เป็นต้น

ถ้าต้องการเชื่อมโยงหน่วยประมวลผลเข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้ UART เราต้องทำการปรับระดับสัญญาณ TTL บน UART ไปเป็นระดับสัญญาณ RS232 โดยการเพิ่มวงจรอุปกรณ์เพิ่มเติมสำหรับผู้เริ่มต้นหรือจะถอดอุปกรณ์เอง โดยการซื้อวงจรจรปรับระดับสัญญาณ RS 232 เป็น UART มาประกอบเองก็ได้

ในโลกของพีซีปัจจุบัน USB เป็นการสื่อสารอนุกรมที่กำลังเป็นที่นิยมมาก เครื่องสมองกลโดยเฉพาะคอมพิวเตอร์กระเป๋าหิ้ว (Laptop) จะไม่มีอุปกรณ์พอร์ตอนุกรมติดตั้งไว้แล้วโดยใช้การสื่อสาร USB แทนด้วยเหตุผลดังกล่าวทำให้ผู้ผลิตหันไปผลิตโปรเซสเซอร์ที่ทำงานแปลงค่าสัญญาณ USB เป็น UART แทนสายสัญญาณ USB<->RS232 กลายเป็นอุปกรณ์ที่นิยมใช้และมีจำหน่ายแพร่หลายในร้านขายอุปกรณ์เครื่องคอมพิวเตอร์ทั่วไป ผลิตภัณฑ์อันหนึ่งที่น่าสนใจคือ สายต่อสัญญาณแบบ USB ที่มีการสื่อสาร UART ของบริษัท FTDI สายสัญญาณนี้เป็นแบบ TTL UART ไม่ใช่ RS232 ที่ปลายสายฝั่งตรงข้ามกับข้อต่อ USB จะต่อเข้าโดยตรงกับหน่วยประมวลผลที่เป็นขาต่อรับส่งสัญญาณ TTL UART ได้ทันที อุปกรณ์ชิ้นนี้มีหมายเลขผลิตภัณฑ์คือ “TTL-232R-3V3”

สรุปข้อความทั้งหมดข้างต้นได้ว่าเราสามารถที่จะเชื่อมโยงหน่วยประมวลผลสองหน่วยเข้าหากันโดยใช้ขาต่อรับส่งข้อมูล UART ได้โดยตรง สำหรับการเชื่อมโยงหน่วยประเมินผลเข้าหาพีซีเราจะต้องทำการใช้อุปกรณ์ประกอบทำหน้าที่แปลงสัญญาณ UART ไปเป็น RS232 หรือ USB ซึ่งตัวอย่างอุปกรณ์ดังกล่าวได้แก่ MAX232 สำหรับใช้กับ RS232 และ FTDI232 สำหรับ USB NETMF สนับสนุนการใช้พอร์ตอนุกรม (UART) บางส่วนซึ่งไม่ครบวงจรเหมือน .NET framework ที่ใช้บนพีซี ในการเขียนรหัสภาษาเพื่อใช้งานพอร์ตอนุกรมเราต้องเติมโปรแกรมเพิ่มผนวก “Microsoft. SPOT.Hardware.SerialPort” เข้าไปแล้วจะต้องอ้างอิงใช้งานโดยเติมประโยคสั่งงาน “using System. IO.Ports” ไว้ที่ตอนต้นของรหัสใช้งาน

สังเกตว่าการตั้งชื่อพอร์ตอนุกรมของพีซี และของ NETMF นั้นเหมือนกันคือ COM และมีหมายเลขลำดับเริ่มต้นที่ COM1 ในระบบคอมพิวเตอร์จะไม่มี COM0 อันนี้อาจจะทำให้เกิดความ

สับสนเล็กน้อยในการเขียนโปรแกรมเพื่อเทียบเคียงพอร์ต COM ไปเป็นพอร์ต UART บนหน่วยประมวลผล เนื่องจากว่าในหน่วยประมวลผลจะใช้หมายเลขลำดับเริ่มต้นที่ UART0 ไม่ใช่ UART1 ซึ่งการเทียบเคียงที่ถูกก็คือ COM1 เท่ากับ UART0 และ COM2 เท่ากับ UART1 เป็นต้น

บนเครื่องพีซีทุกตัวจะมีโปรแกรมสถานีรับส่ง (Terminal) ที่สามารถทำงานติดต่อรับ/ส่งข้อมูลผ่านพอร์ตอนุกรมติดตั้งไว้ใช้งานให้แล้ว โปรแกรมนอกจากสามารถแสดงข้อมูลเป็นอักขระบนหน้าต่างแสดงผลของโปรแกรมแล้ว ยังสามารถส่งข้อมูลทุกชนิดออกไปทางพอร์ตอนุกรมได้อีกด้วย โปรแกรมการส่งข้อมูลแบบนี้ที่มีชื่อว่า Teraterm เพราะว่าเป็นโปรแกรมที่ได้รับความนิยมมากตัวหนึ่ง

โปรแกรมตัวอย่างต่อไปนี้จะใช้ทำการส่งค่านับจำนวน 10 ค่าทุก ๆ วินาที ข้อมูลจะถูกส่งไปด้วยความเร็ว (Baud Rate) ที่ 115200 ต้องตั้งค่าความเร็วอันเดียวกันนี้ให้สัมพันธ์กับโปรแกรมบนเครื่องพีซีด้วย โปรแกรมจะส่งข้อมูลโดยใช้ COM1 ที่อยู่บนอุปกรณ์ NETMF หมายเลขลำดับอันนี้ไม่ได้เกี่ยวข้องกับหมายเลขลำดับของ COM บนพีซีแต่อย่างใด แน่นอนว่าเราจะต้องเทียบค่าพอร์ตของ COM เป็นต้นว่าถ้าเราใช้สายเชื่อม USB ที่มีหมายเลข COM8 เราก็จะต้องเปิดใช้งาน COM8 บนพีซี สำหรับในรหัสภาษาเราจำเป็นต้องใช้ COM1 เพราะ NETMF จะใช้พอร์ตอนุกรมตัวเดียวคือ COM1 ที่ใช้ UART0

```
using System.Threading;
using System.IO.Ports;
using System.Text;

namespace MFConsoleApplication1
{
   public class Program
   {
      public static void Main()
      {
         SerialPort UART = new SerialPort("COM1", 115200);
         int counter=0;
         UART.Open();
         while (true)
         {
            // create a string
            string counter_string = "Count: " + counter.ToString() +
                                    "\r\n";
            // convert the string to bytes
            byte[] buffer = Encoding.UTF8.GetBytes(counter_string);
            // send the bytes on the serial port
            UART.Write(buffer, 0, buffer.Length);
            // increment the counter;
```

```
            counter++;
            //wait...
            Thread.Sleep(100);
        }
    }
  }
}
```

จากตัวอย่างเราจะสังเกตเห็นว่าเราใช้สายอักขระ"\r\n"เป็นคำสั่งใช้จบประโยคพร้อมกับขึ้นบรรทัดใหม่ โดยที่ "\r" จะสั่งให้โปรแกรมสถานีรับส่ง (Terminal) รู้ว่าเรา "จบประโยค" (Return) แล้ว และออกคำสั่ง "\n" ตามลำดับต่อมาให้ "ขึ้นบรรทัดใหม่"

ข้อมูลที่รับเข้ามาที่ UART จะถูกจัดการเรียงลำดับไว้โดยอัตโนมัติโดยไม่มีการสูญหาย ข้อจำกัดของการรับส่งข้อมูลแบบนี้อยู่ที่ขนาดของที่เก็บพักข้อมูลชั่วคราว (Buffer) บนระบบดังนั้นถ้าเราหยุดอ่านข้อมูลหรือไม่ได้เดินตามรหัสเพื่อตรวจสอบ (Debugging) ก็จะต้องปิดการใช้พอร์ตอนุกรมไม่เช่นนั้นจะทำให้ระบบทำงานช้าลง ทั้งยังทำให้การเดินตามรหัสเพื่อตรวจสอบจะไม่ได้ผลแน่นอน

ถ้าจะให้มีประสิทธิภาพมากขึ้นก็โดยตั้งสภาวการณ์ (Events) ทำงานเพื่อเปิดให้มีการรับข้อมูลโดยอิสระและอัตโนมัติ เราจะอธิบายเรื่องนี้ในโอกาสต่อไป ตัวอย่างต่อไปจะให้โปรแกรมทำการคอยรับข้อมูลจนกว่าจะได้รับข้อมูลหนึ่งไบต์แล้วจะแสดงข้อมูลที่ถูกพิมพ์ส่งมา (Transmit)

```
using System.Threading;
using System.IO.Ports;
using System.Text;

namespace MFConsoleApplication1
{
   public class Program
   {
      public static void Main()
      {
         SerialPort UART = new SerialPort("COM1", 115200);
         int read_count = 0;
         byte[] rx_byte = new byte[1];

         UART.Open();
         while (true)
         {

            // read one byte
```

```
            read_count = UART.Read(rx_byte, 0, 1);
            if (read_count > 0)// do we have data?
            {
               // create a string
               string counter_string =
                     "You typed: " + rx_byte[0].ToString() + "\r\n";
               // convert the string to bytes
               byte[] buffer = Encoding.UTF8.GetBytes(counter_string);
               // send the bytes on the serial port
               UART.Write(buffer, 0, buffer.Length);
               //wait...
               Thread.Sleep(10);
            }
         }
      }
   }
}
```

สำหรับตัวอย่างสุดท้ายโปรแกรมจะส่งข้อมูลวกกลับมาหาตัว(Loop–Back)แต่เราจะต้องต่อขาส่งสัญญาณ TX ไปสู่ขารับสัญญาณ RX ที่อยู่บนแผงวงจร

```
using System.Threading;
using System.IO.Ports;
using System.Text;
using Microsoft.SPOT;

namespace MFConsoleApplication1
{
   public class Program
   {
      public static void Main()
      {
         SerialPort UART = new SerialPort("COM1", 115200);
         int read_count = 0;
         byte[] tx_data;
         byte[] rx_data = new byte[10];
         tx_data = Encoding.UTF8.GetBytes("FEZ");
         UART.ReadTimeout = 0;
         UART.Open();

         while (true)
         {
            // flush all data
```

```
            UART.Flush();
            // send some data
            UART.Write(tx_data, 0, tx_data.Length);
            // wait to make sure data is transmitted
            Thread.Sleep(100);
            // read the data
            read_count = UART.Read(rx_data, 0, rx_data.Length);
            if (read_count != 3)
            {
               // we sent 3 so we should have 3 back
               Debug.Print("Wrong size: " + read_count.ToString());
            }
            else
            {
               // the count is correct so check the values
               // I am doing this the easy way so the code is more clear
               if (tx_data[0] == rx_data[0])
               {
                  if (tx_data[1] == rx_data[1])
                  {
                     if (tx_data[2] == rx_data[2])
                     {
                        Debug.Print("Perfect data!");
                     }
                  }
               }
            }
            Thread.Sleep(100);
         }
      }
   }
}
```

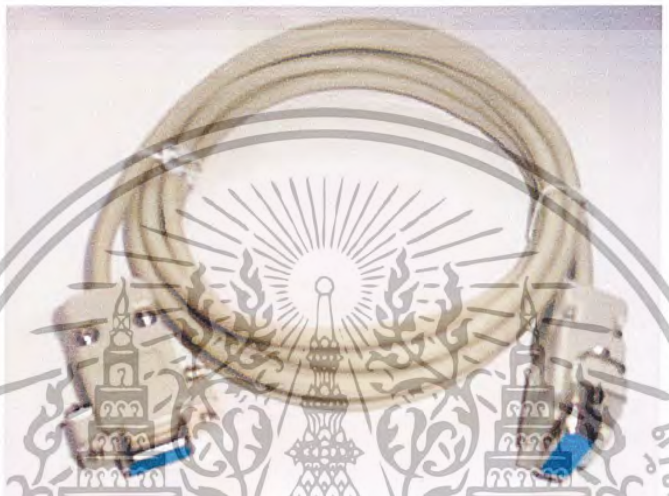

2.6 มาตรฐาน RS – 232C

มาตรฐาน RS – 232C เป็นมาตรฐานที่ได้รับการออกแบบมาเพื่อที่จะทำให้อุปกรณ์ต่อพ่วงจากผู้ผลิตต่างกันสามารถทำงานร่วมกันได้ มาตรฐานหลายชนิดได้รับการออกแบบขึ้นมา แต่มาตรฐานที่ได้รับความนิยมและใช้กันกว้างขวางมากที่สุดคือมาตรฐาน RS – 232C ซึ่งถูกประกาศใช้ในปี 1969 โดยสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ในยุคแรก ๆ การอินเตอร์เฟสแบบ RS – 232C ถูกออกแบบสำหรับเชื่อมต่อเทอร์มินอล (DTE : Data Terminal Equipment) กับโมเด็ม (DCE : Data Communication Equipment) ทั้งนี้ก็เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลบนสายเส้นเดียวกัน

มาตรฐาน RS – 232C ได้แบ่งอุปกรณ์ออกเป็น 2 ประเภท ซึ่งอุปกรณ์ทั้งสองประเภทนี้ก็คือ

1. อุปกรณ์ DTE (Data Terminal Equipment) เป็นอุปกรณ์สำหรับส่งข้อมูล (เอาต์พุต)
2. อุปกรณ์ DCE (Data Communication Equipment) เป็นอุปกรณ์สำหรับรับข้อมูล (อินพุต)

ตามมาตรฐาน RS – 232C แล้วคอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งคอนเน็กเตอร์ที่นิยมใช้กันอยู่จะเป็นชนิด D – Type แบบ 9 ขา และแบบ 25 ขา โดยจะติดตั้งอยู่หลังเครื่องคอมพิวเตอร์ ระดับแรงดันจะมีค่าระหว่าง -3 V ถึง -15 V

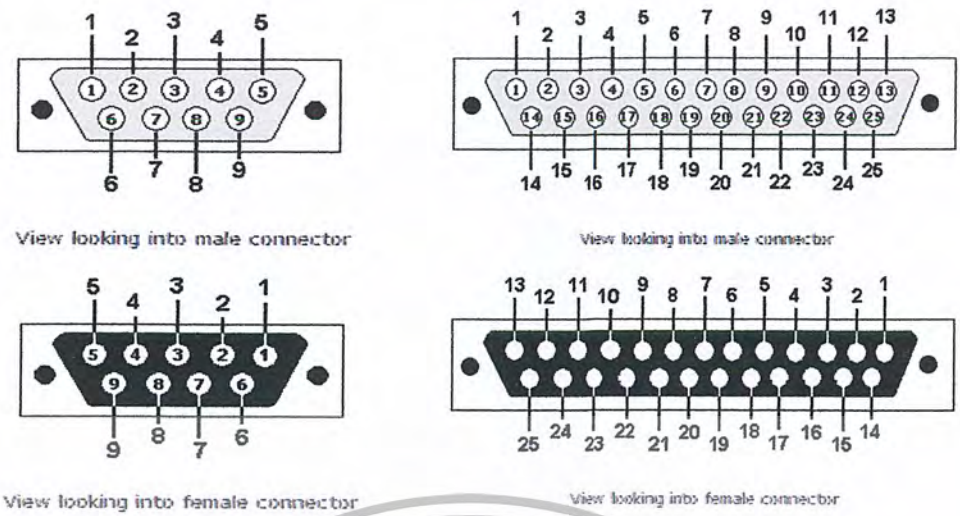


ภาพที่ 2.1 คอนเน็กเตอร์ของ RS – 232

สำหรับลอจิก High และลอจิก Low จะมีระดับแรงดันระหว่าง +3 V ถึง +15 V สามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุด 50 ฟุต หรือ 150 เมตร แต่ถ้าเราต้องการสื่อสารกับอุปกรณ์อื่นที่อยู่ห่างกันมาก ๆ เราจำเป็นต้องใช้อุปกรณ์อื่น ๆ เข้าช่วย เช่น การใช้โมเด็ม เป็นต้น

2.6.1 ลักษณะของคอนเน็กเตอร์แบบ D – Type

หัวต่อแบบ D – Type ที่ใช้ในการสื่อสารแบบอนุกรมของเครื่องคอมพิวเตอร์นั้น จะมีอยู่ 2 ลักษณะคือ แบบ 9 ขา และแบบ 25 ขา บางครั้งเราจะเรียกว่า DB9 และ DB25 ซึ่งหัวต่อทั้งสองชนิดจะมีลักษณะการทำงานของสัญญาณเหมือนกันแต่การจัดเรียงไม่เหมือนกัน



ภาพที่ 2.2 แผนผังคอนเน็กเตอร์ของ RS – 232

ตารางที่ 2.1 แสดงลักษณะการทำงานของสัญญาณ

D – Type 25 Pin	D – Type 9 Pin	สัญลักษณ์	ชื่อสัญลักษณ์
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request To Send
Pin 5	Pin 8	CTS	Clear To Send
Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator

2.6.2 รายละเอียดของสายสัญญาณ

สำหรับรายละเอียดของสายสัญญาณนั้นประกอบไปด้วย

- Transmit Data : TD ใช้สำหรับส่งข้อมูลอนุกรมออกจากคอมพิวเตอร์
- Receive Data : RD ใช้สำหรับรับข้อมูลอนุกรมเข้ามายังคอมพิวเตอร์
- Request To Send : RTS ใช้สำหรับส่งข้อมูลไปยังอุปกรณ์ปลายทาง เพื่อร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลกลับมา
- Clear To Send : CTS ใช้สำหรับตรวจสอบว่าอุปกรณ์ที่เชื่อมต่อด้วยพร้อมที่จะรับข้อมูลหรือไม่ โดยจะคอยรับสัญญาณ RTS เมื่อทุกอย่างพร้อมก็จะทำการส่งข้อมูลออกทางขา TD

- Data Set Ready : DSR ใช้สำหรับตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทางจะใช้คู่กับขา DTR

- Signal Ground : SG เป็นกราวนด์ของระบบ

- Carrier Detect : CD ขานี้จะ Active เมื่อมีการส่งสัญญาณ Carrier จากโมเด็ม

- Data Terminal Ready : DTR ใช้สำหรับบอกให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วยโดยขา DTR นี้ต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง

- Ring Indicator : RI ขานี้จะ Active เมื่อโมเด็มได้รับสัญญาณเรียกเข้าจากสายโทรศัพท์

2.6.3 องค์ประกอบของการรับส่งข้อมูลแบบอนุกรม

การสื่อสารแบบอนุกรมที่นิยมใช้กับคอมพิวเตอร์นั้น เป็นการสื่อสารข้อมูลแบบอะซิงโครนัสนั่นคือ ต้องใช้สายสัญญาณเส้นเดียวทำหน้าที่ทั้งส่งส่วนที่เป็นข้อมูลและส่วนที่ใช้ควบคุมการส่งข้อมูล ดังนั้นข้อมูลที่อ่านได้แต่ละบิตจากการส่งแบบอนุกรม จึงต้องถูกจำแนกว่าใช้สำหรับวัตถุประสงค์ใดโดยเราสามารถแบ่งได้เป็น 4 ส่วนคือ

1. Start Bit ขนาด 1 บิต
2. บิตข้อมูล (Data Character) ขนาด 7 หรือ 8 บิต
3. Parity Bit ขนาด 1 บิต
4. Stop Bit ขนาด 1 หรือ 2 บิต

แต่ละตัวอักษรที่ถูกส่งออกไปเป็นกลุ่มจะประกอบไปด้วยบิตเริ่มต้นบิตข้อมูลบิตพาริตี (จะมีหรือไม่มีก็ได้) และบิตจบ โดยเราจะสรุปหน้าที่ของแต่ละส่วนได้ดังนี้

- Start Bit หรือบิตเริ่มต้นจะใส่ที่จุดเริ่มต้นเสมอ เพื่อเตือนอุปกรณ์ฝ่ายรับว่าข้อมูลกำลังจะมาถึง

- Data Character หรือบิตข้อมูล การส่งบิตข้อมูลจะส่งเป็นกลุ่ม ๆ โดยทั่วไปจะส่งเป็น 7 หรือ 8 บิต ซึ่งเพียงพอสำหรับการส่ง Ascii Word

- Parity Bit หรือบิตพาริตี ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่ง เราจะใส่บิตพาริตีเข้าไป แต่ทั้งตัวรับและตัวส่งจะต้องรู้กันว่าใช้พาริตีแบบไหนในการส่งข้อมูลซึ่งหลักการในการกำหนดบิตพาริตีมีหลายแบบดังนี้

พาริตีคู่ (Even Parity) ค่าของบิตพาริตีนี้เมื่อรวมกับทุก ๆ บิตของข้อมูลแล้วจะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคู่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตีจะเป็น 0

พาริตีคี่ (Odd Parity) ค่าของบิตพาริตีนี้เมื่อรวมกับทุก ๆ บิตของข้อมูลแล้ว จะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคี่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตีจะเป็น 1

ไม่มีพาริตี (None) ถ้าตั้งบิตพาริตีเป็น None ทั้งภาครับและภาคส่งจะไม่มีการตรวจสอบบิตพาริตี

- Stop Bit หรือบิตจบ เป็นบิตที่ส่งมาปิดท้ายข้อมูล

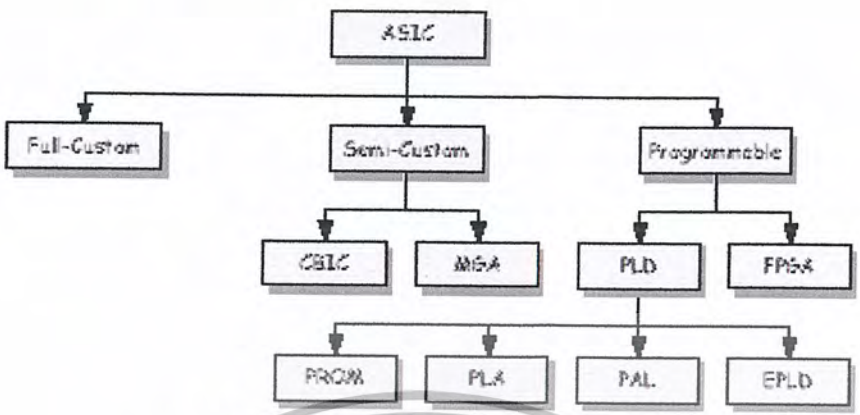
2.6.4 อัตราเร็วในการรับส่งข้อมูลแบบอนุกรม

การที่อุปกรณ์ 2 อย่างจะติดต่อสื่อสารกันได้นั้น จะต้องทำงานด้วยอัตราเร็วเท่ากัน ซึ่งอัตราเร็วในการสื่อสารแบบอะซิงโครนัสคือ ค่าบอดเรต (Baud Rate) มีหน่วยเป็นบิตต่อวินาที ซึ่งค่าอัตราเร็วในการสื่อสารแบบอนุกรมสำหรับมาตรฐาน RS – 232C นั้นมีใช้ดังนี้ 110, 150, 300, 600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที

2.7 ความเป็นมาของ FPGA (Field Programmable Gate Array)

ช่วงก่อนทศวรรษ 1970 อุตสาหกรรมเซมิคอนดักเตอร์ได้ถูกปลุกให้ตื่นตัวขึ้นหลังจากที่มีการประดิษฐ์วงจรรวมหรือไอซีตัวแรกสำเร็จ ในยุคแรกนั้นไอซีขนาดเล็กหรือ SSI (Small – Scale Integration) ประกอบไปด้วยเกทดิจิตอลจำนวนไม่มากนัก (ประมาณ 1 ถึง 10 ตัว) ต่อมาได้มีการเพิ่มปริมาณของเกทดิจิตอลและฟังก์ชันทางลอจิกให้มากขึ้นจนเป็น MSI (Medium – Scale Integration) การพัฒนาไอซีเป็นไปอย่างต่อเนื่องจนมาถึงยุคของ LSI (Large – Scale Integration) ซึ่งเป็นยุคที่มีการสร้างไมโครโปรเซสเซอร์ตัวแรกขึ้นและในปัจจุบันเป็นยุคของ VLSI (Very Large – Scale Integration)ซึ่งเทคโนโลยีในการสร้างไอซีรุดหน้าจนสามารถสร้างไมโครโปรเซสเซอร์ขนาด64บิต ที่มีหน่วยความจำแคชกับหน่วยคำนวณทางคณิตศาสตร์ของโฟล์ทติงพอยน์ (Floating – Point Arithmetic Units) รวมอยู่ภายในตัวมันและเนื่องจากการปรับปรุงเทคโนโลยีของกระบวนการสร้างซีมอสที่มีมาอย่างต่อเนื่อง ทำให้ขนาดของทรานซิสเตอร์ที่บรรจุอยู่ภายในไอซีมีขนาดเล็กลงเรื่อย ๆ จนบางคนโดยเฉพาะ ในญี่ปุ่นใช้คำว่า ULSI (Ultralarge Scale Integration) เพื่อใช้เรียกระดับของไอซีในปัจจุบันแต่คนส่วนมากยังมักนิยมเรียกเพียงแค่ VLSI จากการปรากฏตัวของ VLSI ในช่วงทศวรรษ 1980 ทำให้วิศวกรเริ่มมีการออกแบบไอซีตามความต้องการของลูกค้าซึ่งใช้ในระบบที่เจาะจงนอกเหนือจากการใช้ไอซีมาตรฐานเพียงอย่างเดียว โดยไอซีเหล่านี้มีชื่อเรียก ว่า ASIC : Application – Specific Integration Circuit (ออกเสียงว่า เอซิก) ซึ่งตัวอย่างของ ASIC ได้แก่ ชิพไอซีที่ใช้สำหรับตุ๊กตาของเล่นพูดได้ ดาวเทียม และชิพที่ภายในบรรจุด้วยไมโครโปรเซสเซอร์กับอุปกรณ์ทางลอจิกอื่น ๆ

2.7.1 ประเภทของ ASIC



ภาพที่ 2.3 ประเภทของ ASIC

2.7.1.1 Full – Custom

ASIC ประเภทนี้ลูกค้าจะเป็นผู้ออกแบบเซลล์ลอจิก (เช่น แอนด์เกท ออร์เกท มัลติ-เพล็กเซอร์ และฟลิปฟลอป) และลักษณะการจัดวางอุปกรณ์บนตัวไอซีรวมถึงหน้ากากสำหรับควบคุมการเจือและสร้างชั้นสาร (Mask) ต่าง ๆ ที่ใช้ในการทำไอซีเองดังนั้นค่าใช้จ่ายในการออกแบบและการผลิตจะสูงมาก

2.7.1.2 Semi – Custom

ASIC ประเภทนี้เซลล์ลอจิกจะถูกออกแบบเอาไว้ก่อนแล้วในรูปแบบของไลบรารี และลูกค้าจะเป็นผู้ออกแบบ Mask ต่าง ๆ เอง ตัวอย่างของไอซีประเภทนี้ได้แก่ Standard – Cell – Based ASIC และ Masked Gate – Array – Based ASIC

2.7.2 Standard – Cell – Based ASIC

ไอซีประเภทนี้จะมีพื้นที่สำหรับจัดวางเซลล์ลอจิกมาตรฐานซึ่งถูกออกแบบเอาไว้แล้ว ในบางครั้งเซลล์มาตรฐานเหล่านี้จะถูกนำมาประกอบกันเป็นเซลล์ที่มีขนาดใหญ่ขึ้นเรียกว่า Megacell สำหรับการออกแบบนั้นผู้ออกแบบจะทำเพียงแค่กำหนดตำแหน่งของเซลล์มาตรฐาน และการเชื่อมต่อภายในของแต่ละเซลล์เท่านั้นแต่อย่าไรก็ดีเซลล์ต่าง ๆ เหล่านี้สามารถวางที่ตำแหน่งใด ๆ ก็ได้บนแผ่นเวเฟอร์ซิลิกอน นั่นก็หมายความว่าชั้น Mask จะถูกจัดวางตามความต้องการของผู้ออกแบบ

2.7.3 Masked Gate – Array – Based ASIC

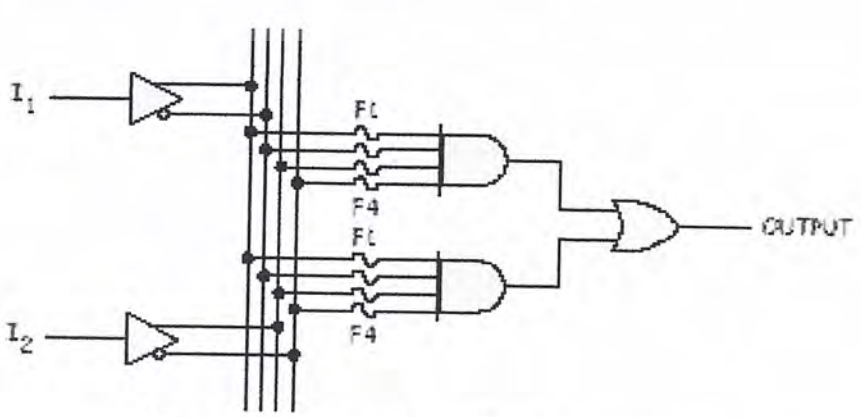
ไอซีชนิดนี้จะมีทรานซิสเตอร์หรือเกทถูกสร้างมาในลักษณะของอะเรย์ 2 มิติ แผ่นเวเฟอร์ซิลิกอนและผู้ออกแบบจะทำการออกแบบ Mask เพื่อใช้สำหรับกำหนดการต่อเชื่อมของทรานซิสเตอร์แต่ละตัว

2.7.4 Programmable

ASIC ประเภทนี้เซลล์ลอจิกจะถูกออกแบบไว้ก่อนเช่นเดียวกับ Semi – Custom แต่ชั้นของ Mask จะไม่สามารถเปลี่ยนแปลงได้ตามความต้องการของผู้ออกแบบไอซีประเภทนี้ยังแบ่งออกเป็น 2 ชนิด คือ Programmable Logic Device (PLD) และ Field Programmable Gate Array (FPGA)

2.8 Programmable Logic Device (PLD)

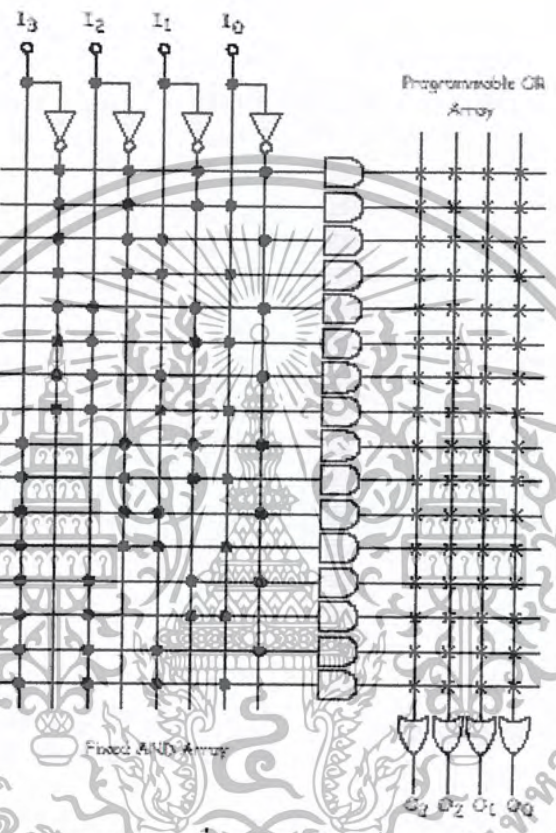
มีโครงสร้างภายในเป็นวงจรพื้นฐานทางด้านลอจิกต่อกันอยู่เป็นกลุ่ม ซึ่งมีทั้งวงจรคอมบิเนชัน (Combination) และซีเควนเชียล (Sequential) สำหรับเทคโนโลยีของวงจรที่ใช้สร้าง PLD จะมีทั้ง TTL, ECL และ CMOS ตามความเหมาะสมของแต่ละระบบไอซี PLD ทุกชนิดมีหลักการพื้นฐานของวงจรภายในที่เหมือนกันโดยมีวงจรคอมบิเนชันที่เป็นผลคูณร่วมบวก (Sum of Product) ประกอบไปด้วยชุดของแอนด์เกทต่อร่วมกับออร์เกท และในการโปรแกรมจะเป็นการเลือกว่าอินพุตภายในของแอนด์เกทกับสัญญาณอินพุทใดบ้างที่จะต้องต่อถึงกัน ซึ่งมีทั้งจากภายนอกและสัญญาณป้อนกลับจากเอาท์พุทภายในเอง เช่น การติดต่ออินพุทของออร์เกทกับเอาท์พุทของแอนด์เกทตัวต่าง ๆ สำหรับการโปรแกรมทางกายภาพนั้นอินพุตต่าง ๆ ของอุปกรณ์ทุกตัวจะถูกต่อผ่านฟิวส์เข้ากับแหล่งสัญญาณ ซึ่งถ้าไม่ต้องการใช้สัญญาณใดก็จะตัดฟิวส์ตัวนั้นทิ้งทำให้สามารถโปรแกรมได้เพียงครั้งเดียว ไอซี PLD บางชนิดใช้มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถโปรแกรมโดยใช้กระแสไฟฟ้าและสามารถลบแล้วโปรแกรมเข้าไปใหม่ได้อีก สำหรับไอซีในตระกูล PLD ได้แก่ PROM, PAL, PLA และ EPLD



ภาพที่ 2.4 วงจรพื้นฐานของ PLD ซึ่งอยู่ในรูปผลคูณร่วมบวก

2.8.1 PROM (Programmable Read Only Memory)

PROM คือหน่วยความจำประเภท ROM ซึ่งนับว่าเป็นไอซี PLD ชนิดหนึ่งซึ่งวงจรภายในของ PROM ประกอบไปด้วยอะเรย์ของแอนด์และออร์เกท (AND – OR Array) ผลลัพธ์ที่ขาดาตาเอาต์พุตสามารถแสดงได้ในสมการของฟังก์ชันผลคูณร่วมบวก (Sum of Product) ของสัญญาณอินพุตที่ขาแอดเดรส

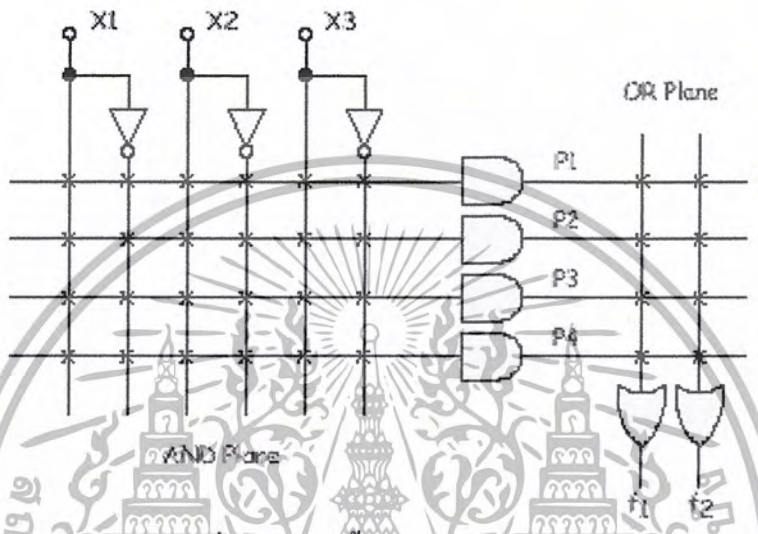


ภาพที่ 2.5 ลักษณะของ PROM เมื่อเปรียบเทียบเป็นวงจรในรูปผลคูณร่วมบวก

ภาพที่ 2.5 แสดงถึงลักษณะการเชื่อมต่อแอนด์เกทและออร์เกทของ PROM ขนาด 16x4 บิต วงจรทางด้านซ้ายบนสุดเป็นแอนด์เกทจะให้ผลคูณ(Product)ของกรณีที่อินพุทเป็น0000แอนด์เกทที่อยู่ถัดลงมาเป็นผลคูณของกรณีที่อินพุตเป็น 0001, 0010, ... จนถึงตัวล่างสุดคือผลคูณในกรณีที่อินพุตเป็น 1111 ซึ่งสำหรับ PROM ที่มีจำนวนอินพุต n ตัวจะมีค่าอินพุตที่เป็นไปได้ทั้งหมดเท่ากับ 2n และค่าอินพุทเหล่านี้จะถูกจัดวางอยู่ในส่วนอะเรย์ของ AND ซึ่งไม่สามารถแก้ไขได้ แต่ในส่วนของ OR จะเป็นส่วนที่อนุญาตให้ทำการโปรแกรมได้ และเนื่องจากการที่ด้าน AND ของ PROM มีการคอมบิเนชันของอินพุทที่เป็นไปได้ทั้งหมด ดังนั้นผู้ออกแบบจึงไม่จำเป็นต้องทำการลดรูปของฟังก์ชันลอจิกที่ออกแบบไว้เลย อย่างไรก็ดีการกระทำเช่นนี้อาจทำให้เกิดจำนวนวงจรที่ไม่มีประสิทธิภาพจำนวนมากบนตัวชิพได้

2.8.2 PLA (Programmable Logic Array)

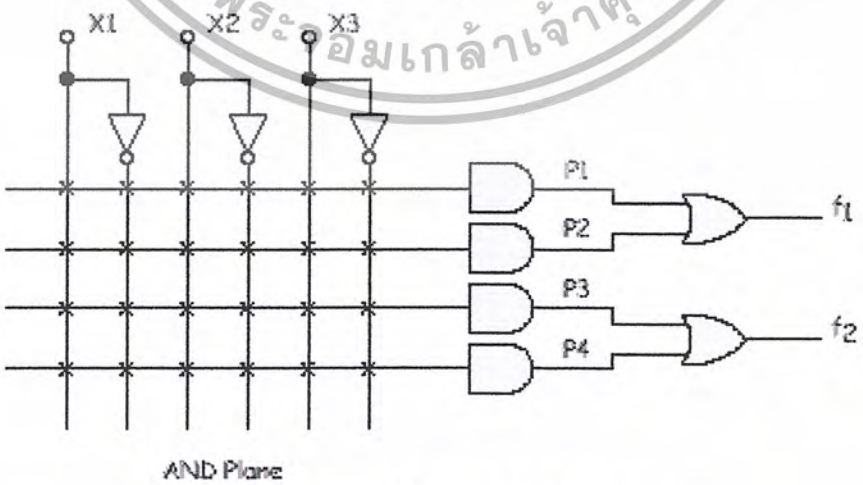
ลักษณะเด่นของ PLA คือสามารถโปรแกรมการเชื่อมต่อได้ทั้งทางด้าน AND และด้าน OR ทำให้มีความยืดหยุ่นในการใช้งานมาก แต่อย่างไรก็ดีข้อเสียที่เห็นได้อย่างชัดเจน ของ PLA คือความยุ่งยากในการสร้างและคุณสมบัติทางด้านความเร็วที่ลดลง เนื่องจากสัญญาณจะต้องวิ่งผ่านอะเรย์ของ AND และ OR



ภาพที่ 2.6 วงจรพื้นฐานภายในของ PLA

2.8.3 PAL (Programmable Array Logic)

PAL มีลักษณะโครงสร้างที่ใกล้เคียงกับ PROM และ PLA มาก แต่การโปรแกรม PAL จะสามารถทำได้เพียงด้าน AND เท่านั้น



ภาพที่ 2.7 วงจรพื้นฐานภายในของ PAL

2.8.4 EPLD (Erasable Programmable Logic Device)

EPLD เป็นอุปกรณ์ที่สามารถทำการโปรแกรมได้หลายครั้งซึ่งเหมาะสำหรับการทำวงจรต้นแบบ สำหรับเทคโนโลยีที่ใช้ในการสร้างจะเหมือนกับ CMOS EPROM คือ ใช้มอสทรานซิสเตอร์เชื่อมต่อระหว่างสัญญาณอินพุตกับจุดที่ต้องการแทนการใช้ฟิวส์แบบเดิม ทำให้สามารถโปรแกรมการต่อวงจรภายในอุปกรณ์ด้วยการจ่ายไฟฟ้าตามขนาดที่กำหนดไว้ และลบได้โดยใช้แสงอัลตราไวโอเลตฉายผ่านช่องหน้าต่างกระจกของตัวชิพ

2.9 Field – Programmable Gate Array (FPGA)

เป็นอุปกรณ์ที่มีความซับซ้อนมากกว่า PLD ไปอีกระดับหนึ่ง ซึ่งในความเป็นจริงแล้ว PLD และ FPGA แตกต่างกันน้อยมากสำหรับ FPGA แล้วนับว่าเป็นอุปกรณ์ตัวใหม่ในตระกูลของ ASIC ซึ่งมีการเจริญเติบโตอย่างรวดเร็ว และมีบทบาทที่สำคัญในการเข้ามาแทนที่ระบบอิเล็กทรอนิกส์ที่ใช้ TTL โครงสร้างภายในของ FPGA ประกอบไปด้วยอะเรย์ของลอจิกเกทต่าง ๆ มากมาย ซึ่งในปัจจุบันความจุเกทภายในตัวชิพ FPGA ได้เพิ่มขึ้นจากระดับไม่กี่พันตัวจนถึงระดับล้านตัว ซึ่งสามารถรองรับวงจรดิจิตอลที่มีความสลับซับซ้อนได้เป็นอย่างดีนอกจากนี้ในด้านของการออกแบบพัฒนาและทดสอบก็ทำได้ง่าย ซึ่งในปัจจุบันการออกแบบวงจรโดยใช้ FPGA กำลังเป็นที่นิยมและมีแนวโน้มที่จะนำมาใช้งานมากขึ้นเรื่อย

2.9.1 ทำความรู้จักกับ FPGA

ในปัจจุบันมี FPGA อยู่ 4 ชนิดที่วางขายอยู่ในท้องตลาดได้แก่ Symmetrical Array, Row-Based, Hierarchical PLD และ Sea-of-Gates ซึ่งแต่ละชนิดก็มีลักษณะการเชื่อมต่อภายในและการโปรแกรมที่แตกต่างกันไป นอกจากนี้ในการแบ่งประเภทของ FPGA อาจแบ่งได้ตามเทคโนโลยีที่ใช้ในการโปรแกรมซึ่งมีอยู่ 2 แบบคือ การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพของตัวชิพ และการโปรแกรมโดยการใช้หน่วยความจำ

2.9.1.1 การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพ

1. Fuse เป็นวิธีการโปรแกรมที่สามารถทำได้เพียงครั้งเดียว ซึ่งหลังจากที่โปรแกรมแล้วจุดเชื่อมต่อจะขาดจากกัน

2. Anti Fuse เป็นวิธีการโปรแกรมที่คล้ายกับแบบ Fuse แต่ต่างกันที่หลังจากทำการโปรแกรมแล้วจุดเชื่อมต่อจะเชื่อมถึงกัน

2.9.2 การโปรแกรมโดยใช้หน่วยความจำ

2.9.2.1 EEPROM Based FPGA

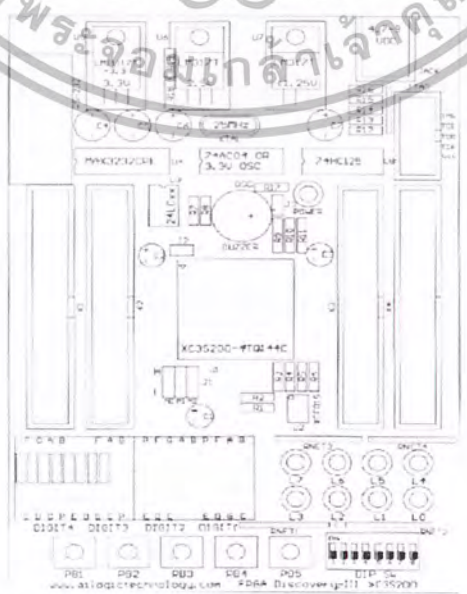
FPGA ที่ใช้การโปรแกรมแบบนี้มักเรียกว่า CPLD ซึ่งเทคโนโลยีที่ใช้จะเหมือนกับ EEPROM ทำให้มีความจำของเกทต่ำ โดยทั่วไปจะน้อยกว่า 20000 เกท แต่ข้อดีของ EEPROM Based FPGA คือ สามารถเก็บข้อมูลที่โปรแกรมลงไปได้โดยไม่จำเป็นต้องมีไฟเลี้ยง และในการโปรแกรมจะใช้ทรานซิสเตอร์ 1 ตัวต่อ 1 บิต ซึ่งการโปรแกรมสามารถทำได้ประมาณ 10000 ครั้ง

2.9.2.2 SRAM Based FPGA

FPGA แบบนี้จะใช้เทคโนโลยีในการโปรแกรมเหมือนกับ SRAM (Static RAM) ทำให้สามารถโปรแกรมซ้ำได้โดยไม่จำกัดจำนวนครั้ง นอกจากนี้ยังมีความจุของเกทในระดับปานกลางถึงสูงมาก (ประมาณ 10000 - 1000000 เกท) ซึ่งข้อดีของ SRAM Based FPGA คือใช้เวลาในการโปรแกรมน้อย (ระดับ nsec) การโปรแกรมทำได้ง่ายเทียบได้กับการเขียน SRAM ทั่วไป และเหมาะสำหรับการออกแบบวงจรที่มีความสลับซับซ้อน ส่วนข้อเสียคือไม่สามารถเก็บโปรแกรมในภาวะที่ไม่มีไฟเลี้ยงได้ ดังนั้น FPGA ชนิดนี้จึงมักใช้ควบคู่กับ ROM เพื่อเก็บโปรแกรมและทำการโหลดโปรแกรมลงในตัวชิพในขณะที่เริ่มต้นใช้งาน

2.9.3 โครงสร้างภายในของ FPGA

ลักษณะโครงสร้างภายในของ FPGA จะเป็นอะเรย์ของบล็อกลอจิกที่สามารถทำการโปรแกรมได้ดังภาพ



ภาพที่ 2.8 โครงสร้างภายในของ FPGA รุ่น Discovery – III XC3S200 Board Manual

2.9.4 การออกแบบโดยใช้ภาษาอธิบายพฤติกรรมของฮาร์ดแวร์

ในการออกแบบวงจรดิจิตอลนั้นสามารถทำได้โดยการวาดวงจร (Schematic) หรือใช้ภาษาอธิบายพฤติกรรม (Hardware Description Language) ของฮาร์ดแวร์ ในกรณีของการออกแบบวงจรด้วย ASIC ชนิด Full Custom ผู้ออกแบบจะต้องเขียนวงจรด้วย Schematic จากนั้นจะนำวงจรที่ออกแบบไว้ไปทำการจำลองการทำงาน (Simulate) ซึ่งหากผลออกมาเป็นที่พอใจก็จะต้อง Layout เป็นชั้นสารและในการออกแบบ ASIC ชนิดนี้ผู้ออกแบบจำเป็นจะต้องทราบถึงเทคโนโลยีที่ใช้ในการสร้างด้วยหลังจากได้ Layout ที่สมบูรณ์แล้วจึงจะส่งไปเข้ากระบวนการสร้างไอซี หรือ Fabrication เพื่อสร้างเป็นชิพไอซีออกมาแต่ในการออกแบบวงจรด้วย FPGA โดยการใช้ Schematic หรือใช้ภาษาอธิบายการทำงานของวงจรจะทำได้สะดวกกว่า เนื่องจากวิธีการนี้ผู้ออกแบบไม่จำเป็น ต้องคำนึงถึงเทคโนโลยีที่จะใช้สร้างไอซีและที่สำคัญ การออกแบบโดยวิธีนี้สามารถแก้ไขโมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่าเพราะไม่ต้องวาดวงจรใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยีสำหรับภาษาที่ใช้ สำหรับอธิบายพฤติ-กรรมของฮาร์ดแวร์ที่ใช้กันก็มี VHDL, AHDL และ Verilog เป็นต้น ส่วนรายละเอียดของขั้นตอนในการออกแบบสามารถอธิบายได้ดังนี้

2.9.4.1 การสังเคราะห์วงจร (Logic synthesis)

ในขั้นตอนนี้จะใช้ซอฟต์แวร์ในการสังเคราะห์วงจร (Synthesis Tools) ทำการสังเคราะห์พฤติกรรมของวงจรที่ได้จากการออกแบบด้วย Schematic หรือ VHDL ซึ่งต้องทำการตรวจสอบด้วยว่าซอฟต์แวร์นั้นสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการหรือไม่ ตัวอย่างเช่น FPGA ของบริษัท XILINX และบริษัท ALTERA จะมีซอฟต์แวร์หลายตัวที่สามารถใช้ได้ เช่น Max Plus II ในขั้นตอนนี้ซอฟต์แวร์สังเคราะห์วงจรจะทำการแปลงโค้ด VHDL และทำการ Optimize เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ในการสังเคราะห์วงจรนั้นวงจรระดับเกท(GateLevel)จะไม่เหมาะสมกับโครงสร้างที่มีอยู่ในอุปกรณ์ FPGA ดังนั้นในการ Optimize ซอฟต์แวร์สังเคราะห์วงจรจะต้องทำการ Optimize ให้ได้เป็นวงจรที่ประกอบด้วยกลุ่มของลอจิกที่เหมาะสมกับอุปกรณ์ FPGA นั้น ๆ จึงทำให้ผลที่ได้มีประสิทธิภาพและในขั้นตอนการสังเคราะห์วงจรนี้ ผู้ออกแบบสามารถกำหนดข้อบังคับสำหรับโมเดลแต่ละตัวได้ เช่น ข้อบังคับในเรื่องเวลา (Timing Constraints) หรือข้อบังคับในเรื่องของพื้นที่ (Area) หรือกำหนดชนิดและตำแหน่งของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอน Optimize เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนดส่วนสำคัญในการ Optimize คือการเทียบ (Mapping) โมเดลให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้างและสถาปัตยกรรมภายในอุปกรณ์ FPGA เมื่อทำการสังเคราะห์วงจรเสร็จแล้วซอฟแวร์การสังเคราะห์วงจรก็จะมีการรายงานผลว่าโมเดลที่ออกแบบไปนั้นเป็นอย่างไรเช่นมีค่าความหน่วง(Delay)เท่าใด

ใช้ทรัพยากรต่าง ๆ ใน FPGA อะไรบ้าง เมื่อมาถึงขั้นตอนนี้ผู้ออกแบบก็จะทราบว่าโมเดลเป็นไปตามข้อบังคับหรือไม่ถ้าไม่ก็สังเคราะห์ใหม่จนกว่าจะเป็นไปตามที่กำหนด

2.9.4.2 การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นการแบ่งวงจรที่ได้จากการสังเคราะห์ เป็นส่วนย่อย ๆ สำหรับลงใน CLBs, IOBs หรือองค์ประกอบอื่น ๆ ภายในอุปกรณ์ FPGA สำหรับเกณฑ์ที่ใช้ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกัน มีจำนวนสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้เพื่อลดความหนาแน่นในตอนทำการเชื่อมต่อสัญญาณ (Routing) ในขั้นตอนนี้จะใช้ซอฟต์แวร์ทำโดยซอฟต์แวร์จะเทียบส่วนประกอบของวงจร เช่น เกท (Gate), ฟลิป – ฟลอป (Flip – Flop) ลงในทรัพยากรต่างๆที่มีอยู่ในอุปกรณ์FPGAหลังจากทำขั้นตอนนี้เสร็จแล้วผู้ออกแบบสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ไปเท่าไร ส่วนข้อมูลทางเวลานั้นผู้ออกแบบจะทราบเฉพาะความหน่วงภายในแต่ละส่วนเท่านั้น หรือที่เรียกว่าความหน่วงลอจิก (Logic Delay) ส่วนซอฟต์แวร์จะรวมเอาซอฟต์แวร์ย่อยอื่น ๆ อีก เพื่อให้การทำ PPR (Partitioning Placement & Routing) เป็นไปอย่างต่อเนื่อง

2.9.4.3 การวางอุปกรณ์ (Placement)

ในขั้นตอนนี้จะเป็นการเลือกทำเลที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (Partitioning) มาแล้วว่าควรจะอยู่ ณ ตำแหน่งไหนในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ดีที่สุด เช่น วงจรส่วนไหนควรอยู่ใกล้กันเพื่อจะได้ค้นหาเส้นทางได้ (Route) ง่ายหรือช่วยลดความหน่วง จะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือ Router ทำการค้นหาเส้นทางสัญญาณได้ไม่หมด การวางอุปกรณ์ที่ดีควรวางส่วนต่าง ๆ ให้อยู่ใกล้กัน โดยเฉพาะส่วนที่มีการเชื่อมต่อสัญญาณด้วยกัน นอกจากนั้นการกำหนดตำแหน่งขา I/O (I/O pin) ตามตำแหน่งขา I/O ของ FPGA บนแผ่น PCB ก็จะมีผลโดยตรงเลยคือซอฟต์แวร์จะวาง I/O ลงในตำแหน่งที่ผู้ออกแบบกำหนด ซึ่งบางครั้งตำแหน่งที่กำหนดไปไม่เหมาะสม ดังนั้นการกำหนดขา I/O ควรกำหนดตำแหน่งให้เหมาะสม หรือไม่ก็ให้ซอฟต์แวร์จัดการเอง

2.9.4.4 การเชื่อมต่อสัญญาณ (Routing)

ในขั้นตอนนี้เป็นการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่าง ๆ ภายในอุปกรณ์ FPGA ขั้นตอนนี้จะทำต่อเนื่องจากการวางอุปกรณ์ ในกรณีที่ทำการวางอุปกรณ์ไว้ไม่ดีซอฟต์แวร์ก็จะทำการเชื่อมต่อสัญญาณได้ไม่หมด (เนื่องจากจำนวนทรัพยากรสำหรับเชื่อมต่อสัญญาณนั้นมีอยู่จำกัด) หรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับ ผู้ออกแบบสามารถทำขั้นตอนนี้ได้โดยใช้

ซอฟต์แวร์หรือผู้ออกแบบจะทำการเชื่อมต่อสัญญาณด้วยตนเองก็ได้ แต่ทางที่ดีควรใช้ซอฟต์แวร์ทำดีกว่า นอกจากนั้นการกำหนดข้อบังคับทางเวลาจะช่วยให้ผลที่ได้จากการเชื่อมต่อสัญญาณดีขึ้นได้

2.9.4.5 ความหน่วงด้านเวลา (Delay)

ในการทำ FPGA นั้นความหน่วงที่เกิดขึ้นเป็นความหน่วงที่เกิดจากการวางตำแหน่ง (Layout) ของอุปกรณ์ ซึ่งผู้ออกแบบไม่สามารถเข้าไปแก้ไขได้ แต่สามารถทำให้มีความหน่วงน้อยที่สุดได้ สำหรับความหน่วงที่เกิดขึ้นนั้นแยกได้เป็น 2 ประเภท คือ

1. ความหน่วงลอจิก (Logic Delay) เป็นความหน่วงภายในองค์ประกอบของอุปกรณ์ FPGA เอง

2. ความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณ (Routing Delay) เป็นความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณระหว่างองค์ประกอบภายในอุปกรณ์ FPGA โดยปกติแล้วค่าความหน่วงลอจิกไม่ควรเกิน 50% ของค่าความหน่วงที่ยอมรับได้ เพราะความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณมักจะมีค่ามากกว่าค่าความหน่วงลอจิก ดังนั้นในการวางอุปกรณ์และเชื่อมต่อสัญญาณ ผู้ออกแบบควรกำหนดข้อบังคับทางเวลาเพื่อให้ซอฟต์แวร์ได้ทำงานอย่างมีประสิทธิภาพเพิ่มขึ้น และเพื่อให้ได้ผลลัพธ์ที่ดีขึ้นค่าความหน่วงที่ได้หลังจากการวางอุปกรณ์ และเชื่อมต่อสัญญาณแล้วจะมีค่าความหน่วงที่ค่อนข้างแน่นอน ซึ่งผู้ออกแบบสามารถทราบได้ว่าโมเดลที่ออกแบบนั้นเป็นไปตามข้อกำหนดหรือไม่

2.9.4.6 การจำลองการทำงานของวงจร (Simulation)

ในขั้นตอนนี้เป็นขั้นตอนที่สำคัญอีกขั้นตอนหนึ่ง เพราะเป็นขั้นตอนที่ผู้ออกแบบตรวจสอบฟังก์ชันการทำงานของโมเดลว่าถูกต้องหรือไม่ มีข้อผิดพลาดตรงไหนเพื่อจะได้ทำการแก้ไขให้ถูกต้อง ในขั้นตอนนี้จะมีซอฟต์แวร์ที่ใช้สำหรับทำการจำลองการทำงานของวงจรที่ใช้อยู่ เช่น Model Sim ของบริษัท Model Technology หรือ Max Plus II ของบริษัท Altera ในการจำลองการทำงานของวงจรควรทำทุกครั้งหลังจากที่มีการทำแต่ละขั้นตอนหลักเสร็จแล้ว เพื่อจะได้ทราบว่าข้อผิดพลาดของโมเดลเกิดขึ้นตอนไหน จะได้แก้ไขข้อผิดพลาดตรงขั้นตอนนั้น ๆ ได้เลย ไม่ต้องมาคอยตรวจหาขั้นตอนที่ทำให้เกิดข้อผิดพลาด นั่นคือการทำการจำลองการทำงานของวงจรต้องทำทั้งหลังการเขียนโค้ด, การสังเคราะห์วงจร และการทำ PPR การจำลองการทำงานของวงจรหลังจากที่เขียนโค้ดเสร็จแล้วนั้น ผู้ออกแบบสามารถทราบได้แค่โมเดลทำงานถูกต้องหรือไม่เท่านั้น (Functional Test) ยังไม่สามารถตรวจสอบการทำงานในเชิงเวลาได้ถูกต้อง ในการจำลองการทำงานของวงจรหลังจากที่สังเคราะห์เป็นวงจรแล้ว เพื่อตรวจสอบว่าฟังก์ชันการทำงานยังคงถูกต้องหรือไม่ และค่าความหน่วงที่เกิดขึ้นเป็นไปตามข้อบังคับหรือไม่ มีข้อผิดพลาดเกิดขึ้นหรือไม่ถ้ามีจะแก้ไขให้ถูกต้อง ในการจำลองการทำงานของวงจรหลังจากที่ทำการวางอุปกรณ์การเชื่อมต่อสัญญาณ (Post

Layout Simulation) แล้วก็มีความสำคัญเช่นกันเพราะผลที่ได้จากการจำลองการทำงานของวงจรในตอนนี้จะเป็นผลลัพธ์ของโมเดลเลย ซึ่งผู้ออกแบบนอกจากจะตรวจสอบฟังก์ชันการทำงานแล้วยังต้องตรวจสอบคุณสมบัติอื่น ๆ เช่น ความหน่วงที่ได้จากการทำ PPR ในรูปแบบค่าความหน่วงมาตรฐาน (Standard Delay Format : SDF) ว่าตรงตามที่กำหนดหรือไม่ หรือตรวจสอบว่าวงจรรวมสามารถใช้งานที่ความถี่สูงสุดเท่าไรนั่นเอง ในการจำลองการทำงานของวงจรควรใช้ซอฟต์แวร์ตัวเดียวกันตลอดเพื่อจะได้เปรียบเทียบผลที่ได้จากขั้นตอนต่าง ๆ

2.9.5 การโปรแกรมอุปกรณ์ FPGA (Configuration)

หลังจากที่โมเดลผ่านขั้นตอนต่าง ๆ จนกระทั่งผ่านการทำ PPR แล้วนั้น ถึงตอนนี้ก็สามารถที่จะดาวน์โหลด (Download) ลงในอุปกรณ์ FPGA ได้แล้ว ในการดาวน์โหลดนี้ก่อนอื่นต้องแปลงแบบวงจรรวมที่ได้เป็นข้อมูลวงจร (Configuration Data) ซึ่งอยู่ในรูปของบิตสตรีม (Bit Stream) ก่อนแล้วจึงดาวน์โหลดลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชันการทำงานตามโมเดลที่ผู้ออกแบบต้องการ ซึ่งในขั้นตอนนี้จะใช้วิธีที่แตกต่างกันออกไปสำหรับอุปกรณ์ FPGA ของแต่ละบริษัทผู้ผลิตคือ ในกรณีที่เป็นอุปกรณ์ FPGA ชนิดที่ต้องโปรแกรมโดยวิธี SRAM นั้น ในการใช้งานผู้ออกแบบจะต้องเก็บข้อมูลวงจรไว้ในหน่วยความจำประเภท EPROM หรือ Serial PROM ด้วย เพื่อจะใช้งานสะดวกขึ้นคือ ในการใช้งานโมเดลครั้งต่อไปไม่ต้องดาวน์โหลดข้อมูลวงจรจากเครื่องคอมพิวเตอร์อีก เพราะมีข้อมูลวงจรเก็บอยู่ในหน่วยความจำอยู่แล้ว แต่กรณีที่อุปกรณ์ FPGA เป็นชนิดที่โปรแกรมโดยใช้วิธี EPROM หรือ Anti Fuse ที่ไม่จำเป็นต้องมีหน่วยความจำสำหรับเก็บข้อมูลวงจร เพราะว่าอุปกรณ์ FPGA ชนิดนี้เมื่อดาวน์โหลดข้อมูลวงจรลงไปข้อมูลที่ดาวน์โหลดลงไปก็ยังคงอยู่ในอุปกรณ์ FPGA และครั้งต่อไปก็ใช้งานโมเดลที่ออกแบบไว้ได้เลย

2.9.6 เครื่องมือสำหรับการออกแบบ FPGA

จะเห็นได้ว่าการออกแบบเพื่อทำ FPGA นั้นทำได้สะดวกกว่า ASIC มากเพราะใช้เวลาน้อยกว่ามากด้วย ส่วนสำคัญที่ใช้ในการทำ FPGA คือ ซอฟต์แวร์ที่ใช้ตั้งแต่เขียนโค้ดอธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลดลงในอุปกรณ์FPGAซึ่งซอฟต์แวร์ที่ใช้ต้องเป็นซอฟต์แวร์ที่ทำงานต่อเนื่องกันได้ สำหรับซอฟต์แวร์ที่ใช้ทำการจำลองการทำงานของวงจรนั้น ต้องสามารถใช้งานต่อเนื่องกับซอฟต์-แวร์ที่ใช้ทั้งระบบ เพราะโมเดลที่ได้จากการทำขั้นตอนต่าง ๆ (ด้วยซอฟต์แวร์ต่าง ๆ ต้องเอามาจำลองการทำงานได้ และในการจำลองการทำงานของวงจรควรใช้ซอฟต์แวร์ตัวเดียวกันตลอด ทั้งระบบเพื่อจะได้เปรียบเทียบผลได้ง่าย ในอดีตซอฟต์แวร์ส่วนใหญ่จะใช้งานอยู่บนคอมพิวเตอร์สมรรถนะสูงอย่างเวิร์คสเตชัน (Workstation) ในปัจจุบันมีการพัฒนาซอฟต์แวร์ที่ใช้บนพีซี (PC) มากขึ้นซึ่งสามารถลดค่าใช้จ่ายในด้านอุปกรณ์คอมพิวเตอร์

2.10 ภาษา VHDL

2.10.1 ประวัติความเป็นมาของภาษา VHDL

VHDL ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC : Very High Speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (High Level Language) ที่ใช้สำหรับการออกแบบฮาร์ดแวร์ในระบบดิจิตอลตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (Hierarchy) และสามารถเขียนได้หลายรูปแบบ ด้วยเหตุผลนี้จึงทำให้ภาษา VHDL เป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุดคือ แนวความคิดที่จะแก้ปัญหาลงไปทีละขั้นจนถึงขั้นตอนของการสร้างวงจรจริง และตัวภาษาก็เปิดโอกาสให้วิศวกรได้พัฒนาและจำลองการทำงานของรูปแบบฟังก์ชันการทำงานของวงจรอย่างสังเขป โดยไม่ต้องคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้น VHDL ยังเป็นภาษาที่สนับสนุนลักษณะต่าง ๆ ของระบบดิจิตอลที่มีความซับซ้อนได้ทั้งหมด ดังนั้น VHDL จึงเป็นภาษาที่น่าสนใจในการศึกษาและนำไปใช้งานเป็นอย่างยิ่ง วิวัฒนาการของภาษา VHDL เริ่มต้นประมาณปี ค.ศ. 1981 เมื่อกระทรวงกลาโหมสหรัฐอเมริกาหรือ DoD (Department of Defense) ได้พยายามปรับปรุงอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทางทหารให้มีความทันสมัยมากขึ้น ประกอบกับเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์มีการพัฒนาไปอย่างรวดเร็วดังจะเห็นได้ จากการนำวงจรดิจิตอลหลาย ๆ วงจรมาทำการผลิตอยู่บนแผ่นซิลิกอนที่มีพื้นที่เพียง1–2ตารางเซนติเมตรเท่านั้นซึ่งเป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้นตลอดจนความน่าเชื่อถือ ในการทำงานและความคงทนต่อสภาพแวดล้อมสูง แต่เนื่องจากในขณะนั้นขั้นตอนของการออกแบบการผลิต และการตรวจสอบวงจรต้นแบบเป็นกระบวนการที่ต้องใช้วิศวกรและเวลาในดำเนินการมาก ฉะนั้นทาง DoD จึงจัดตั้งโครงการขึ้นมาเพื่อศึกษาวิธีการที่ช่วยในการพัฒนาวงจรอิเล็กทรอนิกส์ โดยเฉพาะวงจรระบบดิจิตอลให้สามารถนำไปผลิตได้เร็วขึ้น ซึ่งโครงการดังกล่าวมีชื่อว่า "Very High Speed Integrated Circuits" หรือ VHSIC โดยในระยะแรกนั้นโครงการนี้ถือเป็นความลับทางด้านความมั่งคงของประเทศ และอยู่ภายใต้ความควบคุมดูแลของ United States International Traffic and Arms Regulations (ITAR) สำหรับมาตรฐานของภาษาที่ใช้บรรยายพฤติกรรมวงจรหรือฮาร์ดแวร์ของระบบ สำหรับโครงการ VHSIC ที่ DoD ได้ให้ไว้สามารถสรุปได้ดังนี้

- ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถเข้าใจได้ทั้งมนุษย์และเครื่องคอมพิวเตอร์โดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก
- สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้
- ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

เพราะฉะนั้นภาษาดังกล่าวนี้จัดเป็นภาษาโปรแกรมระดับสูงเช่นเดียวกับภาษาปาสคาล หรือภาษาซี ซึ่งในทางวิศวกรรมภาษาที่ใช้ในการออกแบบฮาร์ดแวร์นี้เรียกว่า "Hardware Description Language" หรือ HDL

ในตอนเริ่มแรกนั้น DoD ได้มอบหมายให้บริษัทไอบีเอ็ม เท็กซัสอินสตูเมนท์ และอินเตอร์เมทริกซ์ เป็นผู้ศึกษาและพัฒนาโครงการ ซึ่งการดำเนินงานเป็นไปอย่างต่อเนื่องจนกระทั่งในปี ค.ศ. 1985 ทาง ITAR ได้ยกเลิกข้อจำกัดในการถ่ายทอดเทคโนโลยีทางทหารออกจากโครงการนี้ ดังนั้นภาษา VHDL จึงเริ่มเป็นที่รู้จักกันโดยทั่วไป และประมาณปี ค.ศ. 1987 IEEE ได้ทำการกำหนดมาตรฐานของภาษานี้เป็น IEEE 1076-1987 และมีชื่อเรียกว่า VHDL ซึ่งมาตรฐานนี้ได้รับ การปรับปรุงจนเป็นมาตรฐาน IEEE 1076-1993 หรือ VHDL 1993 เนื่องจากในขณะนั้น DoD เป็นลูกค้ารายใหญ่ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์ ดังนั้นจึงมีผู้รับโครงการต่าง ๆ จาก DoD ไปดำเนินการวิจัยและพัฒนาเป็นจำนวนมาก และเพื่อให้ทุกโครงการอยู่ในมาตรฐานเดียวกันหมดดังนั้นทาง DoD จึงได้กำหนดว่าทุกๆ โครงการต้องเขียนอยู่ในรูปของภาษา VHDL เท่านั้น ซึ่งทำให้ DoD สามารถนำโครงการเหล่านี้ไปจำลองกับเครื่องคอมพิวเตอร์ได้หลาย ๆ ระบบ

2.10.2 ข้อกำหนด

DoD ได้ตั้งข้อกำหนดสำหรับภาษา VHDL ในเดือนมกราคม ปี ค.ศ.1983 ไว้ดังนี้

2.10.2.1 ลักษณะทั่วไป

DoD ได้กำหนดให้ VHDL เป็นภาษาสำหรับการออกแบบและบรรยายของฮาร์ด-แวร์ ซึ่งหมายถึงความสามารถในการอธิบายและออกแบบในระดับสูง การจำลอง (Simulation) การสังเคราะห์ (Synthesis) และการทดสอบ (Testing) นอกจากนั้น VHDL ยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์ตั้งแต่ระดับบนซึ่งก็คือระบบจนถึงระดับเกทอีกด้วย เนื่องจากในการทำงานของระบบดิจิตอลนั้นทุก ๆ องค์ประกอบภายในระบบไม่ว่าเล็กหรือใหญ่จะทำงานไปพร้อม ๆ กัน ซึ่งในเรื่องของความพร้อมเพรียงในการทำงานนี้ก็ถือเป็นข้อกำหนดที่สำคัญอย่างหนึ่งของ VHDL ด้วยเช่นกัน (สำหรับในภาษาที่ใช้ในการบรรยายฮาร์ดแวร์นั้นความพร้อมเพรียงจะหมายถึงทุก ๆ คำสั่งองค์ประกอบเกทหรือวงจรลอจิกจะถูกนำมาปฏิบัติทั้งหมด ดังนั้นในที่สุดแล้วก็จะดูเหมือนว่าได้มีการปฏิบัติไปพร้อม ๆ กัน)

2.10.2.2 สนับสนุนการออกแบบแบบลำดับขั้น

การออกแบบแบบลำดับขั้นเป็นลักษณะที่สำคัญอย่างหนึ่ง ซึ่งสำหรับการออกแบบระบบที่มีหลาย ๆ ระดับ โดยในการออกแบบจะประกอบด้วยส่วนการบรรยายการเชื่อมต่อและส่วนการบรรยายหน้าที่การทำงาน ซึ่งหน้าที่การทำงานของระบบสามารถกำหนดได้ด้วยตัวเอง หรืออาจ

ถูกกำหนดโดยโครงสร้างที่ประกอบด้วยองค์ประกอบย่อย ๆ ลงไปได้เช่นกัน แต่ที่ระดับล่างสุด องค์ประกอบต้องถูกบรรยายหน้าที่การทำงานด้วยตัวมันเอง และไม่สามารถกำหนดการทำงานโดยลักษณะแบบโครงสร้างได้

2.10.2.3 ไลบรารี

VHDL ได้สนับสนุนการมีไลบรารีเพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษณะและการทำงานของอุปกรณ์พื้นฐานไว้ในระบบไลบรารี หรือจะใช้ไลบรารีที่ระบบได้จัดเตรียมไว้แล้วก็ได้ โมเดลและการบรรยายที่ถูกต้องควรจัดเก็บไว้ในไลบรารีหลังจากที่ได้ผ่านการคอมไพล์เรียบร้อยแล้วเพื่อให้ผู้ออกแบบคนอื่น ๆ สามารถนำไปใช้ได้ด้วย

2.10.2.4 ลำดับคำสั่ง

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการโดยพร้อมเพรียงกัน เป็นคุณสมบัติที่สำคัญของ VHDL ก็ตามตัวภาษาเองก็ยังมีการจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วย เมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ประกอบที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้ว ผู้ออกแบบยังสามารถบรรยายหน้าที่การทำงาน ซึ่งเป็นรายละเอียดภายในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรมที่ประกอบด้วยโครงสร้างแบบ case, if - then - else และ loop ทั่ว ๆ ไปได้ การบรรยายแบบลำดับคำสั่งทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้สะดวกและง่ายขึ้น อย่างไรก็ตามโครงสร้างทั้งหมดของ VHDL ก็ยังคงเป็นการทำงานแบบพร้อมเพรียงกันเช่นเดิม

2.10.2.5 การกำหนดคุณสมบัติ

นอกจากการกำหนดอินพุตและเอาต์พุตแล้วเงื่อนไขอื่น ๆ ก็มีผลต่อการปฏิบัติหน้าที่ของอุปกรณ์ฮาร์ดแวร์ด้วยเช่นกัน โดยสิ่งนี้จะรวมถึงสภาพแวดล้อมและลักษณะทางกายภาพของอุปกรณ์นั้น ๆ ด้วย ซึ่งภาษาสำหรับการออกแบบที่ดีควรให้ผู้ออกแบบกำหนดคุณสมบัติของอุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนดขนาดลักษณะทางกายภาพเวลาโหลด และเงื่อนไขทางสภาพแวดล้อมอื่น ๆ ซึ่งความสามารถในการกำหนดคุณสมบัตินี้ก็เป็นส่วนหนึ่งที่มีอยู่ในภาษา VHDL ด้วยเช่นกัน

2.10.2.6 ชนิดของข้อมูล

VHDL สามารถกำหนดชนิดของข้อมูลไม่เพียงแต่ชนิด BIT และ BOOLEAN เท่านั้น แต่ยังสามารถกำหนดชนิดของข้อมูลเป็นจำนวนเต็ม จำนวนจริง จุดทศนิยม และชนิดลำดับการนับ (Enumerate Type) หรือแม้แต่ชนิดของข้อมูลที่ผู้ออกแบบกำหนดขึ้นมาเองก็ได้

2.10.2.7 โปรแกรมย่อย

ความสามารถในการใช้ฟังก์ชันและโพรซีเจอร์ (Procedure) ก็เป็นข้อกำหนดอีกอย่างหนึ่งใน VHDL ซึ่งผู้ออกแบบสามารถนำโปรแกรมย่อยมาใช้ในการเปลี่ยนแปลงชนิดของข้อมูล การกำหนดหน่วยของลอจิก การกำหนดตัวกระทำต่าง ๆ หรือหน้าที่อื่น ๆ ตามที่ต้องการได้เช่นเดียวกับการเขียนโปรแกรมทั่วไป

2.10.2.8 การควบคุมเวลา

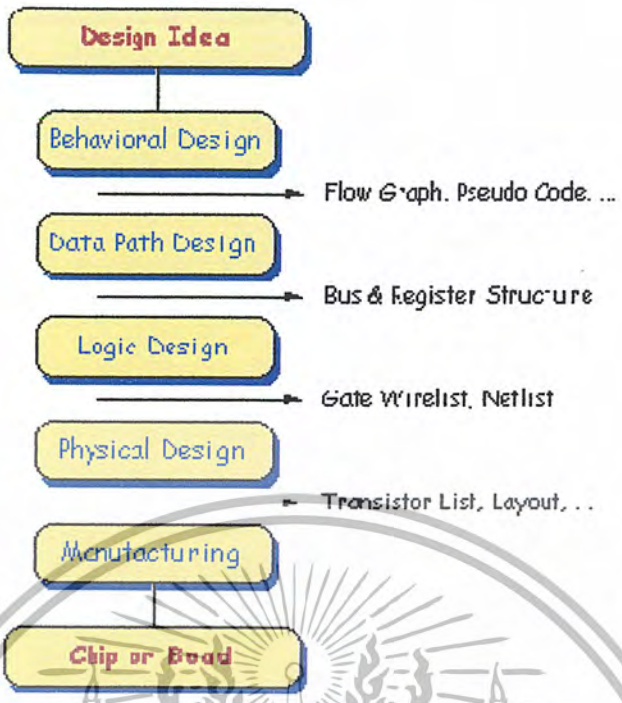
VHDL อนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูล หรือสัญญาณได้ตามต้องการ การตรวจสอบการออกแบบเกทหรือการหน่วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอย เหตุการณ์ (Event) นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

2.10.2.9 การกำหนดแบบโครงสร้าง

การกำหนดโครงสร้างขององค์ประกอบต่าง ๆ สามารถกระทำได้ในทุกระดับของการออกแบบ โดยการกำหนดโครงสร้างขององค์ประกอบร่วมที่เกิดจากองค์ประกอบย่อย ซึ่งแตกต่างกันหรือเหมือนกันก็เป็นข้อกำหนดอย่างหนึ่งของ VHDL เช่นกัน

2.10.3 การออกแบบระบบดิจิตอล

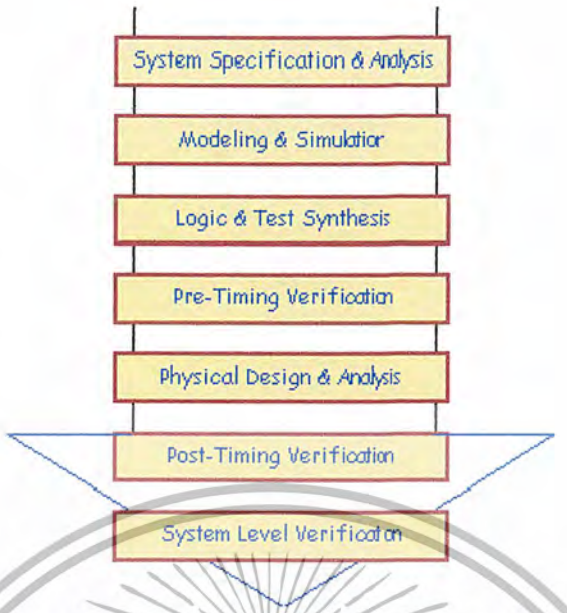
ความซับซ้อนและขนาดของระบบดิจิตอลในปัจจุบันได้เพิ่มมากขึ้นทุกขณะ ส่งผลให้มีการนำคอมพิวเตอร์เพื่อช่วยในการออกแบบหรือ CAD มาใช้ในขบวนการออกแบบฮาร์ดแวร์เพิ่มขึ้นเช่นกันอีกทั้งอุปกรณ์และวิธีการออกแบบใหม่ๆก็ถูกพัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้กับนักออกแบบมากขึ้นด้วยสำหรับภาษาบรรยายอุปกรณ์ฮาร์ดแวร์(HDL:HardwareDescriptionLanguage) ก็เป็นเครื่องมืออย่างหนึ่งที่ได้รับการพัฒนามาอย่างต่อเนื่อง เพื่อช่วยให้การปรับปรุงกระบวนการออกแบบระบบดิจิตอลเป็นไปอย่างมีประสิทธิภาพ ในการออกแบบระบบดิจิตอลเริ่มตั้งแต่กำหนดแนวความคิดเบื้องต้น จนกระทั่งได้ออกมาเป็นอุปกรณ์ฮาร์ดแวร์ที่ใช้งานได้ต้องผ่านขั้นตอนต่าง ๆ มากมาย และในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์ในแต่ละขั้นก่อนเข้าสู่กระบวนการออกแบบในขั้นต่อไป



ภาพที่ 2.9 แสดงขั้นตอนการออกแบบระบบดิจิตอล

ภาพที่ 2.9 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิตอลทั่วไป ขั้นแรกผู้ออกแบบจะกำหนดแนวความคิดในการออกแบบแล้วทำการพัฒนาให้สามารถนำมาใช้ได้อย่างสมบูรณ์ ภายในขั้นตอนนี้ผู้ออกแบบจำเป็นต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบ ซึ่งอาจเป็นผังงานแสดงแบบหรือรหัสคำสั่งเทียม (Pseudo Code) ได้

ขั้นตอนต่อไปเป็นการออกแบบระบบเส้นทางของข้อมูล ผู้ออกแบบจะต้องกำหนดส่วนประกอบของรีจิสเตอร์และวงจรลอจิกที่จำเป็นทั้งหมดเพื่อนำมาประกอบเป็นระบบที่สมบูรณ์ โดยแต่ละองค์ประกอบสามารถเชื่อมต่อกันด้วยบัสหนึ่งหรือสองทิศทาง(Unidirectional or Bidirectional Bus) ส่วนกระบวนการในการควบคุมการเคลื่อนย้ายข้อมูลระหว่างรีจิสเตอร์และวงจรลอจิกจะขึ้นอยู่กับพฤติกรรมของระบบที่กำหนดไว้ดังภาพ



ภาพที่ 2.10 ขั้นตอนการออกแบบจากบนลงล่าง

จากภาพที่ 2.10 แสดงให้เห็นถึงขั้นตอนการออกแบบจากบนลงล่างทั้งนี้ ในทางปฏิบัติอาจจะมีข้อแตกต่างไปจากนี้บ้างเล็กน้อย โดยขั้นตอนของการออกแบบจากบนลงล่างมีรายละเอียดดังนี้

1. ขั้นตอนการสร้างข้อกำหนดของความต้องการและวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ในการแก้ปัญหา

2. ขั้นตอนการเขียนรูปแบบของระบบที่ต้องการออกแบบ โดยใช้ภาษาวีเอชดีแอลสำหรับบรรยายพฤติกรรมการทำงานพร้อมทั้งจำลองการทำงานเพื่อทำการเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

3. ขั้นตอนการสังเคราะห์ต้องทำการกำหนดเทคโนโลยีที่จะมารองรับวงจรที่ออกแบบและระบบช่วยออกแบบจะทำการสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้น โดยให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจรในระดับเกท (Gate Level) และการเชื่อมต่อกับของอุปกรณ์เหล่านั้น หรือไม่ก็อยู่ในรูปของเน็ทลิสต์ (Net List) ที่สามารถนำไปผลิตลงบนอุปกรณ์อื่นได้

4. หลังจากการสังเคราะห์วงจรให้อยู่ในระดับเกทหรือเน็ทลิสต์แล้ว ข้อมูลที่ได้นอกจากจะเป็นข้อมูลสำหรับจำลองการทำงานในเรื่องของความถูกต้องของฟังก์ชันแล้ว ยังมีข้อมูลที่เกี่ยวกับเวลาอีกด้วย ซึ่งจากความจริงที่ว่าอุปกรณ์อิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการเคลื่อนผ่าน (Propagation Delay Time) เสมอ ถึงแม้ว่าจะเป็นเวลาที่น้อยมากในระดับนาโนวินาที แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกทของฟังก์ชันต่าง ๆ จำนวน 10000 เกทขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้นอาจทำให้การทำงานของวงจรทั้งหมดผิดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาสูง ๆ ได้

5. ขั้นตอนของการผลิตเป็นวงจรจริง (Technology and Device Mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจจะอยู่ในรูปของอุปกรณ์เอฟพีจีเอหรือวงจรรวม

6. หลังจากที่ได้วงจรจริงมาแล้ว ยังต้องมีความจำเป็นที่จะต้องตรวจสอบการทำงานที่คำนึงถึงเวลาด้วย เพื่อความถูกต้องของวงจรครั้งสุดท้ายก่อนที่จะนำไปรวมเข้ากับอุปกรณ์อื่น ๆ ให้เป็นระบบ เพราะในขั้นตอนนี้วงจรที่ออกแบบจะประกอบด้วยอินพุตและเอาต์พุตแพด (Pad) ซึ่งเป็นจุดต่อสำหรับรับและส่งสัญญาณกับภายนอก

7. หลังจากที่นำวงจรที่ออกแบบรวมเข้ากับอุปกรณ์อื่น ๆ ให้เป็นระบบแล้วนั้น จะต้องทดสอบการทำงานรวมทั้งระบบร่วมกับอุปกรณ์อื่น ๆ อีกครั้ง ซึ่งเป็นการทดสอบการทำงานจริงขั้นสุดท้าย

2.10.4 องค์ประกอบของภาษาวีเอชดีแอล

ส่วนประกอบที่สำคัญและเป็นพื้นฐานของการเขียนมี 4 หน่วย คือ

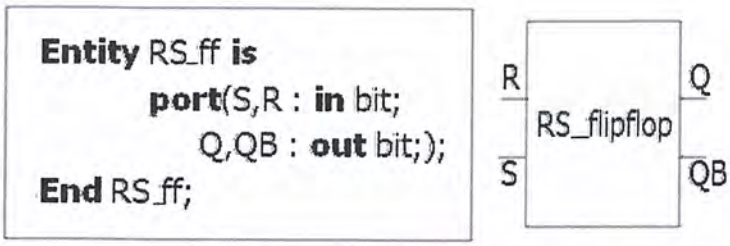
2.10.4.1 หน่วยการออกแบบเอนทิตี (Entity Design Unit)

หน่วยการออกแบบนี้เป็นส่วนที่ใช้สำหรับติดต่อระหว่างอุปกรณ์ภายนอก กับรูปแบบที่เขียนขึ้นรวมทั้งการส่งผ่านค่าพารามิเตอร์บางอย่างระหว่างวงจรกับอุปกรณ์ภายนอก โดยเป็นการกำหนดจุดเชื่อมต่อของรูปแบบกำหนดทิศทางการไหลของสัญญาณ ประเภทของค่าที่สามารถกำหนดให้กับสัญญาณตามจุดต่าง ๆ ของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้น ภาพที่ 2.9 แสดงให้เห็นถึงโครงสร้างของหน่วยการออกแบบเอนทิตี

```
Entity component_name is
        Input and output ports
        Physical and other parameters
End component_name;
```

ภาพที่ 2.11 โครงสร้างโดยทั่วไปของหน่วยการออกแบบเอนทิตี

ส่วนนี้จะขึ้นต้นด้วยคำว่า Entity และ is ระหว่างคำทั้งสองคำเป็นส่วนสำหรับชื่อของรูปแบบที่ต้องการจะเขียน (component_name) หลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทางเข้าและออกของข้อมูลรวมทั้งพารามิเตอร์อื่น ๆ และที่สำคัญคือ หน่วยการออกแบบเอนทิตีต้องปิดท้ายด้วยคำว่า End และเครื่องหมายอัฒภาคเสมอ



ภาพที่ 2.12 รูปแบบของอาร์เอสฟลิปฟลอป

ในภาพที่ 2.12 เป็นหน่วยการออกแบบเอนทิตีที่บรรยายอุปกรณ์ชื่อ อาร์เอสฟลิปฟลอป ในส่วนหัวของเอนทิตีมีการกำหนดจุดต่อ 4 จุด ภายใต้ชุดคำสั่ง port โดยที่ 2 จุดแรกเป็นจุดให้ข้อมูลไหลผ่านเข้าได้แก่ R, S ซึ่งกำหนดด้วยทิศทางการติดต่อกับโลกภายนอกเป็นการไหลเข้าของข้อมูล ส่วนจุดเอาต์พุตเป็นจุดให้ข้อมูลไหลออก ได้แก่ Q, QB ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลออก ส่วนประเภทของข้อมูลที่จะไหลเข้าและออกนั้นเป็นประเภทบิตที่สามารถมีค่าได้เพียงสองค่าเท่านั้น คือ "0" และ "1" เท่านั้น ประเภทของพอร์ตที่สามารถประกาศใช้ในเอนทิตีมี 4 ประเภทดังนี้

1. พอร์ตอินพุตเข้าเป็นพอร์ตทิศทางเดียว ที่นำค่าสัญญาณจากอุปกรณ์ภายนอกเข้ามาภายในวงจร สามารถนำมาป้อนให้กับสัญญาณอื่น หรืออ่านค่าได้แต่ไม่สามารถถูกเขียนจากภายในวงจรได้

2. พอร์ตเอาต์พุตออกเป็นพอร์ตทิศทางเดียว ที่นำค่าสัญญาณจากวงจรส่งออกไปยังอุปกรณ์ภายนอก สามารถเขียนจากภายในวงจรได้ แต่ไม่สามารถอ่านจากภายในวงจรได้

3. พอร์ตอินพุตและเอาต์พุต (Bidirectional) เป็นพอร์ต 2 ทิศทาง ที่สามารถส่งถูกเขียนและอ่านได้จากภายในวงจร

4. พอร์ตบัฟเฟอร์ (Output with Internal Feedback) เป็นพอร์ตเอาต์พุตประเภทหนึ่งที่สามารถอ่านค่ากลับ (Feedback) เข้ามาภายในวงจรได้

2.10.4.2 หน่วยการออกแบบสถาปัตยกรรม (Architecture Design Unit)

หน่วยการออกแบบสถาปัตยกรรม คือ ส่วนที่ใช้เขียนบรรยายพฤติกรรมของรูปแบบในมุมมองของการจำลองการทำงาน พฤติกรรมต่าง ๆ ที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่านเข้าและออกตรงช่องทาง ตลอดจนพารามิเตอร์ต่าง ๆ ที่กำหนดในหน่วยการออกแบบเอนทิตี ภาพที่ 2.13 แสดงให้เห็นถึงโครงสร้างของหน่วยการออกแบบสถาปัตยกรรม

```
ARCHITECTURE identifier OF component_name IS
        [declartion]
BEGIN
        Specification of the functionality of the component
        In terms of its input lines and as influenced
        By physical and other parmeters
END [identifier];
```

ภาพที่ 2.13 โครงสร้างโดยทั่วไปของหน่วยการออกแบบสถาปัตยกรรม

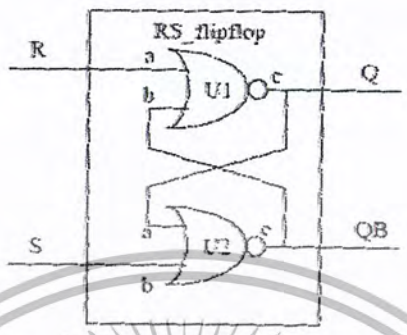
ส่วนของหน่วยการออกแบบสถาปัตยกรรมเริ่มต้นด้วยคำว่า Architecture และตามด้วยชื่อ (identifier) สิ่งที่ต้องการกำหนดลงไปได้แก่ สิ่งที่แสดงให้เห็นว่าหน่วยการออกแบบสถาปัตยกรรมนั้นใช้บรรยายหน่วยการออกแบบเอนทิตีใด ๆ (of <entity design unit> is) ส่วนที่อยู่ระหว่าง Architecture และ Begin เป็นพื้นที่ส่วนประกาศหน่วยของสถาปัตยกรรมกำหนด (Architecture declaration area) ที่เป็นส่วนเผื่อเลือกในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่าง ๆ ที่จะนำไปใช้ภายในสถาปัตยกรรมนั้นได้อาทิเช่น สัญญาณ ค่าคงที่ โปรแกรมย่อย และอุปกรณ์ ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้าและไหลออกของรูปแบบ(สัญญาณที่กำหนดในชุดคำสั่งพอร์ต) นั้นจะถูกบรรยายในบริเวณเนื้อที่ระหว่างคำว่า Begin กับ End ของหน่วยการออกแบบสถาปัตยกรรม และนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ภายในบริเวณนี้ จะเป็นชุดคำสั่งแบบแข่งขนาน (Concurrent statement) เท่านั้น คือทุก ๆ สถานะจะทำงานพร้อมกันลำดับก่อนหลังจะไม่มีผลต่อการทำงานของรูปแบบ หน่วยการออกแบบสถาปัตยกรรมจะต้องปิดท้ายด้วยคำสั่ง End และชื่อของสถาปัตยกรรมนั้น ๆ โดยทั่วไปการเขียนรูปแบบโมเดล (Modeling styles) ระบบเชิงเลขด้วยภาษาวีเอชดีแอลสามารถเขียนได้ในลักษณะต่าง ๆ ดังนี้

- ลักษณะการไหลของข้อมูล (Dataflow Style) [RTL Descriptions]
- ลักษณะพฤติกรรม (Behavioral Style) [Algorithm Descriptions]
- ลักษณะโครงสร้าง (Structural Style) [Netlist Description]
- ลักษณะผสม (Mixed Model Style)

```
Architecture dataflow of RS_ff is
    Begin
            Q <= not(QB or R);
            QB <= not(Q or S);
End dataflow;
```

ภาพที่ 2.14 หน่วยการออกแบบสถาปัตยกรรมของอาร์เอสฟลิปฟลอปตามฟังก์ชันบูลีน

ภาพที่ 2.14 ส่วนที่บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า (R, S) กับข้อมูลที่ไหลออก (Q, QB) ประกอบด้วยชุดคำสั่งแบบแข่งขนาน 2 ชุด ซึ่งเขียนเป็นประเภทการไหลของข้อมูล หรือเรียกว่าระดับการถ่ายโอนข้อมูลระหว่างรีจิสเตอร์ (RTL : Register Transfer Level)



ภาพที่ 2.15 โครงสร้างภายในสถาปัตยกรรมของอาร์เอสฟลิปฟลอป

ภาพที่ 2.15 เป็นหน่วยการออกแบบสถาปัตยกรรมของอาร์เอสฟลิปฟลอปในลักษณะโครงสร้าง ซึ่งเปรียบเสมือนการนำอุปกรณ์ที่มีอยู่ในไลบรารีมาต่อเป็นวงจรตามต้องการ โดยใช้นอร์เกท 2 อินพุต (nor2) จำนวนสองตัวมาสร้างตามฟังก์ชันบูลีน

```
Architecture struc of RS_ff is
        component nor2
                port(a,b : in bit;
                     c : out bit);
        end component;

Begin
        U1 : nor2 port map(R,QB,Q);
        U2 : nor2 port map(S,Q,QB);
End struc;
```

ภาพที่ 2.16 หน่วยการออกแบบสถาปัตยกรรมของอาร์เอสฟลิปฟลอปในลักษณะโครงสร้าง

```
Architecture behave of RS_ff is
        Begin
                process(R,S)
                        begin
                                if R='0' and S='1' then
                                        Q <='1';
                                        QB <='0';
                                else R='1' and S='0' then
                                        Q <='1';
                                        QB <='0';
                                end if;

                end process
End behave;
```

ภาพที่ 2.17 หน่วยการออกแบบสถาปัตยกรรมของอาร์เอสฟลิปฟลอปในลักษณะพฤติกรรม

ภาพที่ 2.17 เป็นการเขียนบรรยายการทำงานของรูปแบบในลักษณะพฤติกรรม ซึ่งจะเห็นได้ว่ามีลักษณะที่เหมือนกับการเขียนโปรแกรมทั่วไป โดยจะต้องมีการใช้งานส่วนที่เรียกว่า Process และการทำงานของรูปแบบจะขึ้นอยู่กับการเปลี่ยนแปลงของสิ่งที่อยู่ภายในProcess (อินพุตR,S) ซึ่งเรียกว่า เซนซิติวิตีลิสต์ (Sensitivity List) การเขียนในลักษณะนี้ลำดับก่อนหลังของชุดคำสั่งจะมีผลต่อการทำงานของรูปแบบที่เขียนขึ้น

\

```
Architecture mixed of RS_ff is
                component nor2
                        port(a,b : in bit;
                                c : out bit;
                end component;

        Begin
                U1 : nor2 port map(R,QB,Q);
                QB <= not(Q or S);
End mixed;
```

ภาพที่ 2.18 แสดงหน่วยการออกแบบสถาปัตยกรรมของอาร์เอสฟลิปฟลอปในลักษณะผสม

ไม่ว่าจะเขียนบรรยายส่วนของสถาปัตยกรรมของอาร์เอสฟลิปฟลอป ในลักษณะของพฤติกรรมการไหลของข้อมูลโครงสร้าง หรือผสมที่นำเอาแต่ละลักษณะมาเขียนไว้ในส่วนของสถาปัตยกรรมก็ตามต่างก็มีพฤติกรรมเดียวกัน และจะให้ผลลัพธ์จากการจำลองการทำงานที่เหมือนกัน ซึ่งถือว่าเป็นข้อดีของภาษาวีเอชดีแอลการออกแบบวงจรดิจิตอลใด ๆ จะต้องออกแบบหน่วยการออกแบบพื้นฐานก็คือ ส่วนของเอนทิตีและส่วนของหน่วยของสถาปัตยกรรมที่มีความสัมพันธ์กัน ซึ่งมีด้วยกันหลายรูปแบบสำหรับการออกแบบวงจรเดียวกัน

2.10.4.3 หน่วยการออกแบบแพ็กเก็ต (Package Design Unit)

ข้อมูลต่าง ๆ ตลอดจนโปรแกรมย่อย เป็นประโยชน์ต่อการเขียนรูปแบบบรรยายระบบเชิงเลขสามารถเก็บไว้ในส่วนของแพ็กเก็ตได้ ข้อมูลเหล่านี้สามารถเรียกไปใช้ได้โดยหน่วยการออกแบบเอนทิตี หน่วยการออกแบบสถาปัตยกรรม หรือจากหน่วยการออกแบบแพ็กเก็ตอื่น ๆ โดยปกติแล้วแพ็กเก็ตจะแบ่งออกเป็น 2 ส่วน คือ การประกาศแพ็กเก็ต (Package Declaration) และส่วนของบอดี้แพ็กเก็ต (Package Body) เนื่องจากแพ็กเก็ตถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่จะนำแพ็กเก็ตไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษาวีเอชดีแอลสามารถกระทำได้ด้วยชุดคำสั่ง USE โดยสิ่งที่สามารถสร้างไว้ใน Package ได้แก่

- Subprogram
- Types
- Constants
- Signals
- Aliases
- Attributes
- Component
- Disconnection Specification

ส่วนการประกอบแพ็กเก็ตเป็นส่วนที่มีความสำคัญที่สุดของแพ็กเก็ต (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ ส่วนประกาศแพ็กเก็ตเพราะจะเป็นส่วนที่กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเก็ตสำหรับนำไปใช้ภายนอกตัวของแพ็กเก็ตเองสิ่งใด ๆ ที่ถูกประกาศไว้ในส่วนของบอดี้แพ็กเก็ต แต่ไม่ได้ถูกประกาศไว้ในส่วนการประกาศแพ็กเก็ตจะไม่สามารถถูกนำค่าและพฤติกรรมไปใช้ส่วนนอกได้ ซึ่งสามารถเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศเอนทิตีคือ จุดเชื่อมต่อหรือพอร์ตที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเก็ตสามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ (Package Body) และยังสามารถถูกนำไปใช้จากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศชนิดของข้อมูล (Type) หรือสัญญาณ (Signal) ในทางกลับกันกับส่วนบอดี้แพ็กเก็ตที่ไม่จำเป็นต้องมีส่วนของการประกาศแพ็กเก็ต แต่แพ็กเก็ตนั้นจะไม่สามารถถูกนำไปใช้จากรูปแบบอื่นได้

```
PACKAGE package_name IS
        Package_declarative_part
END package_name;
```

ภาพที่ 2.19 โครงสร้างโดยทั่วไปของส่วนการประกาศแพ็กเก็ต

บอดี้แพ็กเก็ตจะเป็นโครงสร้างที่ประกอบด้วยคำสั่งต่างๆ ในรูปของคำสั่งลำดับ (Sequential Statement) ที่ใช้บรรยายฟังก์ชั่นการทำงานของโปรแกรมย่อยทั้งหลายที่ชื่อของโปรแกรมย่อยนั้น ๆ ที่ถูกประกาศไปในส่วนของการประกาศแพ็กเก็ตแล้วจะถูกเก็บไว้ในส่วนบอดี้แพ็กเก็ต ทั้งนี้รวมทั้งการกำหนดค่าคงที่ต่าง ๆ อันได้แก่ ค่าคงที่ที่ถูกประกาศชื่อก่อนในส่วนของการประกาศแพ็กเก็ต แต่ถูกกำหนดค่าในส่วนของบอดี้แพ็กเก็ต ฉะนั้นส่วนบอดี้แพ็กเก็ตจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเก็ตไม่มีการประกาศชื่อที่เป็นโปรแกรมย่อยหรือค่าคงที่ การเขียนบอดี้แพ็กเก็ตนั้นเป็นไปตามกฎเกณฑ์ที่แสดงในภาพที่ 2.20

```
PACKAGE BODY package_name IS
        declarative part
END package_name;
```

ภาพที่ 2.20 โครงสร้างโดยทั่วไปของบอดี้แพ็กเก็ต

2.10.4.4 หน่วยการออกแบบโครงแบบ (Configuration Design Unit)

ดังที่ทราบกันแล้วว่ารูปแบบหนึ่งของระบบดิจิตอลไม่ว่าจะเป็นอะไร จะมีหน่วยออกแบบเอนทิตีได้เพียงหนึ่งหน่วยเท่านั้น แต่ในขณะที่หน่วยการออกแบบเอนทิตีหนึ่งหน่วยจะมีสถาปัตยกรรมที่เป็นรองได้หลายหน่วย ดังนั้นจะต้องมีหน่วยการออกแบบโครงแบบมาเพื่อกำหนดการใช้โครงแบบประกอบเอนทิตีกับหน่วยการออกแบบสถาปัตยกรรมหน่วยไหนเข้า

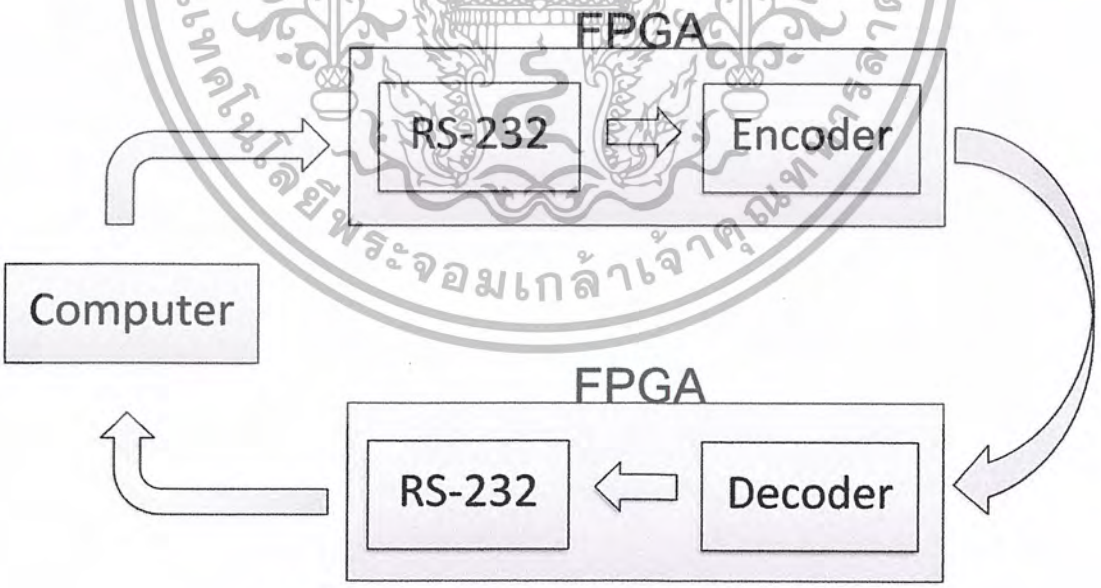
```
CONFIGURATION identifier OF entity_name IS
        Configuration_declarative_part
END;
```

ภาพที่ 2.21 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

บทที่ 3
วิธีการดำเนินงาน

อุปกรณ์เอฟพีจีเอ เป็นอุปกรณ์ที่ใช้ในการโปรแกรมวงจรที่ได้ออกแบบลงไปเพื่อให้อุปกรณ์เอฟพีจีเอมีฟังก์ชันการทำงานตามที่ได้ออกแบบไว้ ในการทำเอฟพีจีเอซึ่งเป็นวิธีการออกแบบไอซี (IC : Integrated Circuit) แบบเซมิคัสทอม (Semicustom) อีกวิธีหนึ่ง เมื่อเทียบกับการทำวงจรรวมแล้วนั้นก็มีทั้งดีและข้อเสีย คือ การทำเอฟพีจีเอจะมีข้อจำกัดในด้านขนาดของวงจรเพราะภายในอุปกรณ์เอฟพีจีเอจะมีจำนวนเกท (Gate) ให้ใช้จำนวนจำกัดและการทำเอฟพีจีเอก็เหมาะสำหรับการทำผลิตภัณฑ์ ต้นแบบหรือเพื่อผลิตในปริมาณต่ำ ส่วนข้อดีของการทำเอฟพีจีเอก็คือระยะเวลาที่ใช้ในการทำตั้งแต่ เขียนรหัส (Code) อธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลดนั้นน้อยกว่าการทำวงจรรวมมากและการตรวจหรือแก้ไขการออกแบบก็ทำได้สะดวก นอกจากนั้นอุปกรณ์เอฟพีจีเอของแต่ละผู้ผลิตก็มีโครงสร้างและความสามารถที่แตกต่างกันบางส่วน ในการใช้งานนั้นอุปกรณ์เอฟพีจีเอสามารถนำไปประยุกต์ใช้งานได้

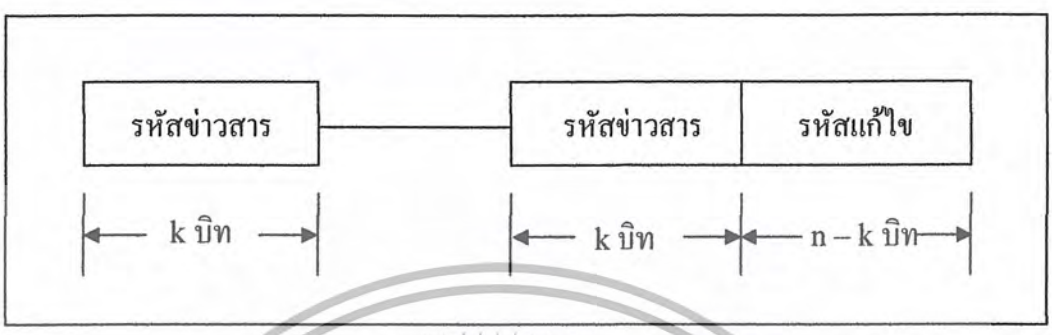
3.1 การออกแบบการเข้ารหัสลับ



ภาพที่ 3.1 แสดงหลักการทำงานของการเข้ารหัสลับ

3.1.1 การสังเคราะห์ข้อมูลข่าวสาร

โดยเราจะให้ u = m + G หรือ m - G เป็นสมการที่เราสังเคราะห์ขึ้นมาเพื่อให้นำข้อมูลที่ได้ไปทำการเข้ารหัสและถอดรหัสโดยใช้เอฟพีจีเอ



ภาพที่ 3.2 แสดงรูปรหัสคำ

โดยที่ u คือ ข้อมูลที่ส่งกลับมาแสดงผลที่คอมพิวเตอร์

$$u = (m_1 + v_1) + (m_2 + v_2) + \ldots\ldots + (m_k + v_k) \quad (3.1)$$

$$u = (m_1 - v_1) + (m_2 - v_2) + \ldots\ldots + (m_k - v_k) \quad (3.2)$$

m คือ ข้อมูลข่าวสารที่ใช้ในการส่ง

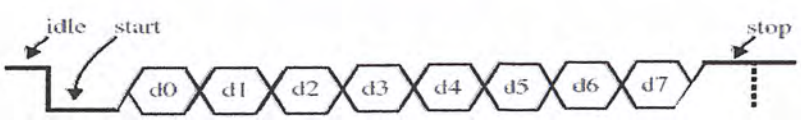
G คือ กุญแจลับที่เรานำมาที่ใช้ในการเข้ารหัสและถอดรหัส

$$G = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \ldots & v_{1n} \\ v_{21} & v_{22} & \ldots & v_{2n} \\ \vdots & & & \\ v_{k1} & v_{k2} & \ldots & v_{kn} \end{bmatrix}$$

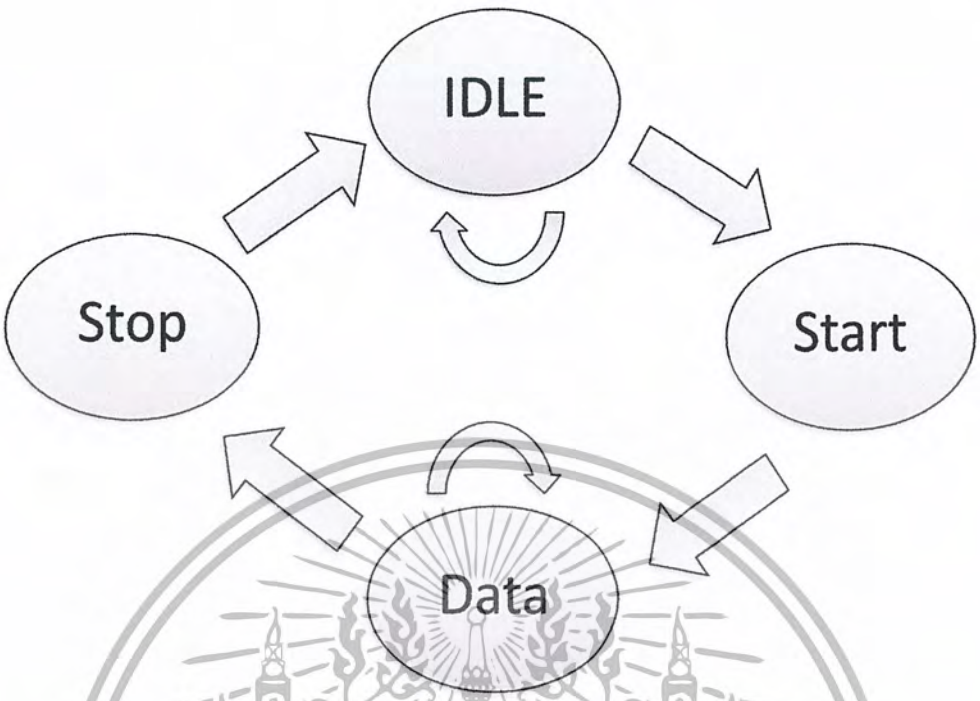
v คือ เมตริกซ์ที่เซตค่าไว้ในเอฟพีจีเอ

3.1.2 Transmission of a byte

สถานะก่อนที่ข้อมูลจะส่งเข้ามานั้นเป็นสถานะ idle ซึ่งมีค่าเป็น 1 จากนั้น เมื่อมีการส่งข้อมูลจะให้บิต start เป็น 0 แล้วเป็นข้อมูลไปอีก 8 บิต เมื่อครบแล้วจะให้กลับมามีค่าเป็น 1 ซึ่งถือว่าเป็นบิต stop ในการตรวจสอบค่าข้อมูล 1 บิตนั้นจะแบ่งการแซมปิ้งออกเป็น 16 ครั้ง เมื่อเราต้องการค่าข้อมูลจะเริ่มตรวจสอบค่าในครั้งที่ 7 เป็นครั้งแรก (โดยเริ่มนับที่ 0) และครั้งต่อ ๆ ไปเพิ่มค่าทีละ 16 จนจบ



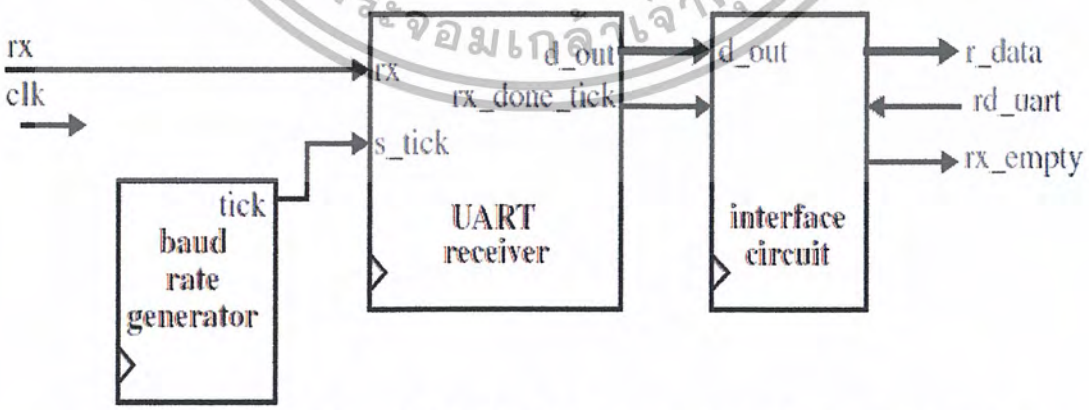
ภาพที่ 3.3 แสดงการส่งข้อมูล 8 บิตและบิตหยุด 1



ภาพที่ 3.4 แสดง Finite State Machine ของ Transmission of a byte

3.1.3 Conceptual Block Diagram of a UART Receiving Subsystem

แผนภาพของวงจรการรับ UART ที่แสดงนี้ประกอบไปด้วย 3 ส่วน คือ UART Receiver ซึ่งเป็นวงจรที่รับข้อมูลจากการแซมปิ้งเข้ามา ส่วนที่สองคือ Baud rate Generator เป็นวงจรที่กำหนดจำนวนครั้งของการแซมปิง ส่วนสุดท้าย Interface Circuit เป็นวงจรที่ให้ค่าบัฟเฟอร์และสถานะแก่ UART Receiver และระบบที่ใช้ UART

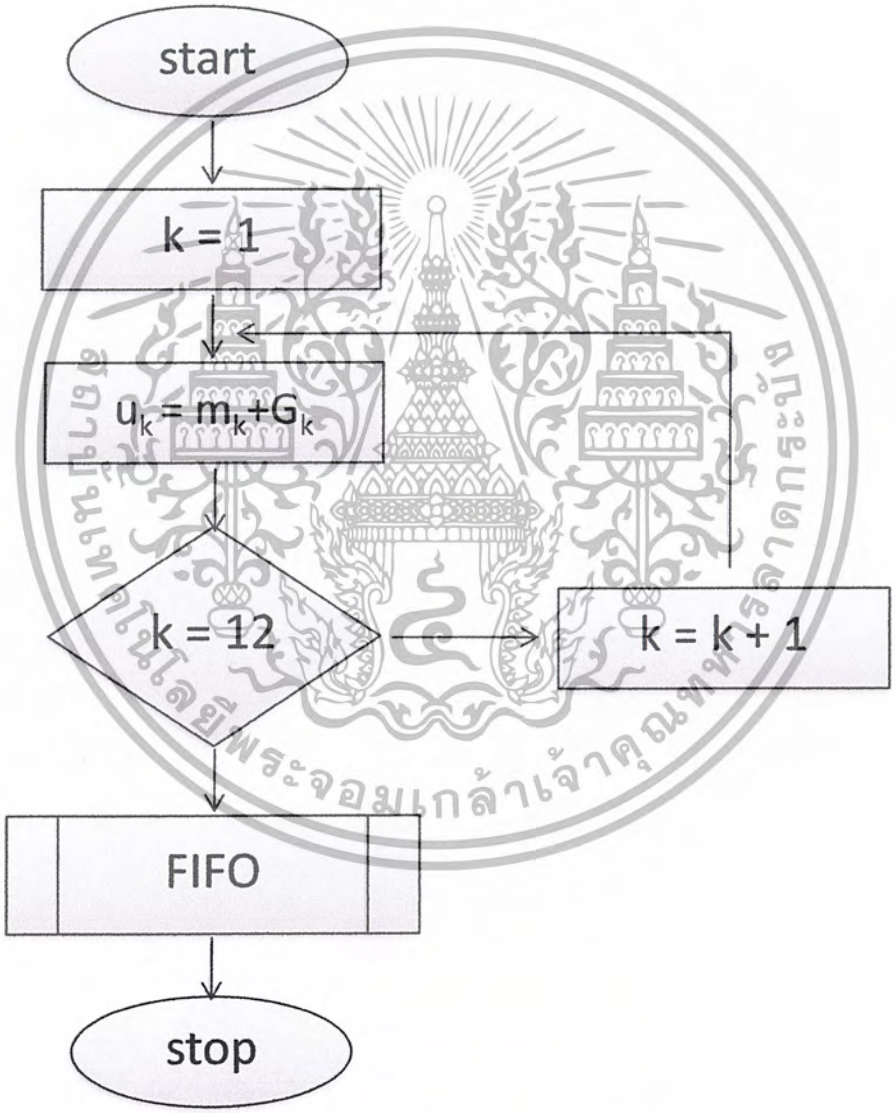


ภาพที่ 3.5 แสดงวงจรอินเตอร์เฟซให้บัฟเฟอร์และสถานะระหว่าง UART และ FPGA

3.1.4 หลักของการเข้ารหัส

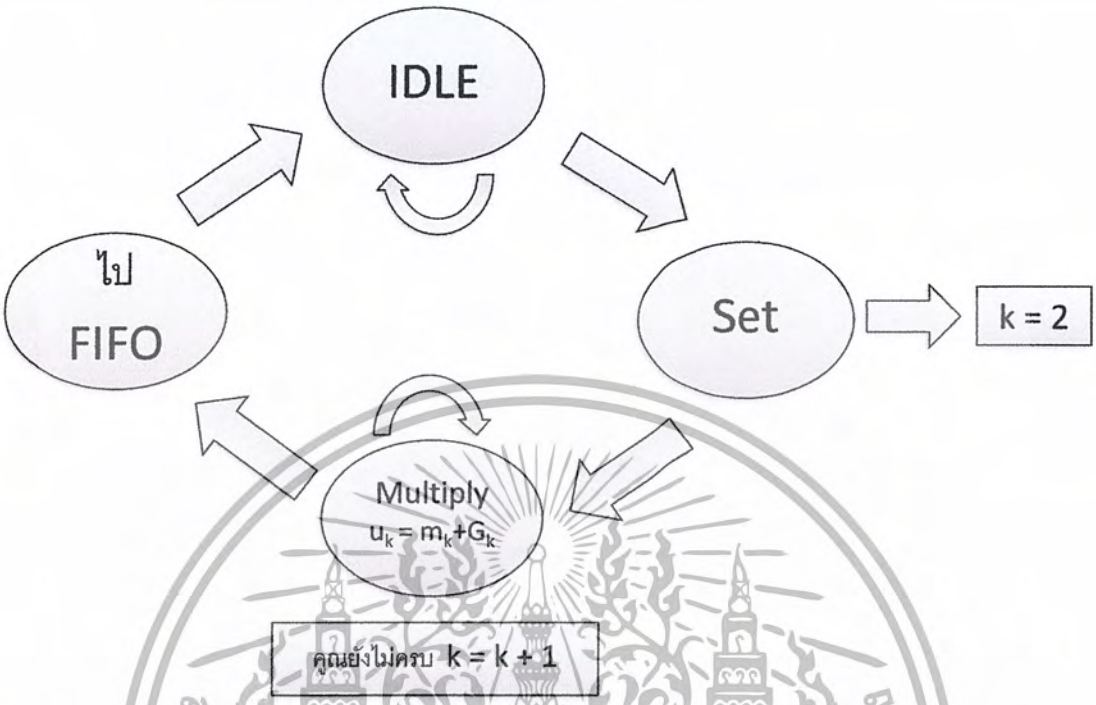
3.1.4.1 Flow Chart ของการเข้ารหัส

เริ่มต้นการเข้ารหัสจะกำหนดให้ค่า k เท่ากับ 1 จากนั้นจะใช้สูตร $u_k = m_k + G_k$ จากนั้นตรวจสอบว่า k มีค่าเท่ากับ 12 หรือไม่ หากไม่เท่ากับ 12 ให้ทำการบวกค่า k เข้าไป 1 แล้วเข้าสูตร $u_k = m_k + G_k$ อีกครั้ง จนเมื่อ k มีค่าเท่ากับ 12 จะนำค่าไปเก็บไว้ในตัวแปรและนำค่าออกไปใช้แบบ First In First Out



ภาพที่ 3.6 แสดง Flow chart ของการเข้ารหัส

3.1.4.2 Finite State Machine ของการเข้ารหัส

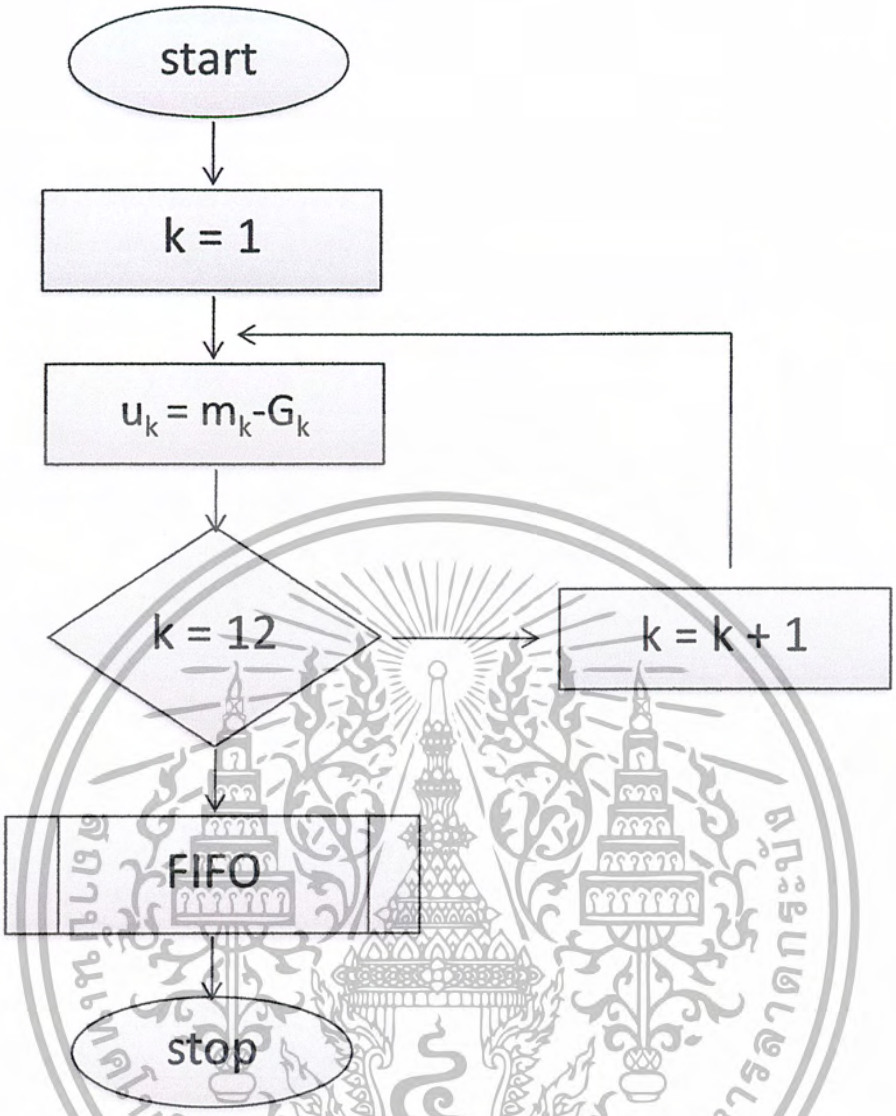


ภาพที่ 3.7 แสดง Finite State Machine ของการเข้ารหัส

3.1.5 หลักของการถอดรหัส

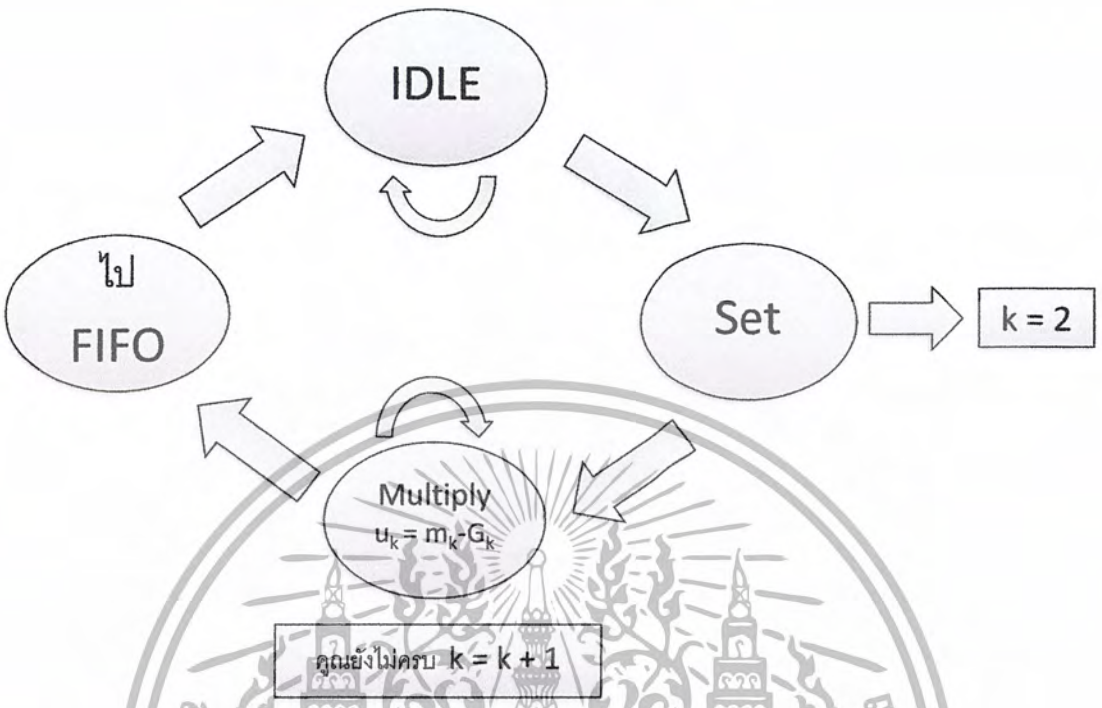
3.1.5.1 Flow Chart ของการถอดรหัส

ใช้วิธีการเช่นเดียวกับการเข้ารหัสแต่เปลี่ยนสูตรจาก $u_k = m_k + G_k$ เป็น $u_k = m_k - G_k$ คือ เริ่มต้นการเข้ารหัสจะกำหนดให้ค่า k เท่ากับ 1 จากนั้นจะใช้สูตร $u_k = m_k - G_k$ จากนั้นตรวจสอบว่า k มีค่าเท่ากับ 12 หรือไม่ หากไม่เท่ากับ 12 ให้ทำการบวกค่า k เข้าไป 1 แล้วเข้าสูตร $u_k = m_k - G_k$ อีกครั้ง จนเมื่อ k มีค่าเท่ากับ 12 จะนำค่าไปเก็บไว้ในตัวแปรและนำค่าออกไปใช้แบบ First In First Out



ภาพที่ 3.8 แสดง Flow chart ของการถอดรหัส

3.1.5.2 Finite State Machine ของการถอดรหัส



ภาพที่ 3.9 แสดง Finite State Machine ของการถอดรหัส

3.2 การจัดเก็บในส่วนของข้อมูลจาก FPGA

1. เขียนโค้ดในส่วนของวงจรการเข้ารหัสโดยใช้ภาษา VHDL
2. เขียนโค้ดในส่วนของวงจรการรับ – ส่งอนุกรมโดยใช้ภาษา Visual Basic
3. เขียนโค้ดในส่วนของวงจรหน่วยความจำโดยใช้ภาษา VHDL
4. เขียนโค้ดในส่วนของวงจรแปลงข้อมูลอนุกรม
5. ทดสอบโค้ดโดยใช้ Modelsim (โปรแกรม ISE WebPACK) ในการจำลองการทำงานของโค้ด VHDL

บทที่ 4

ผลการทดลอง

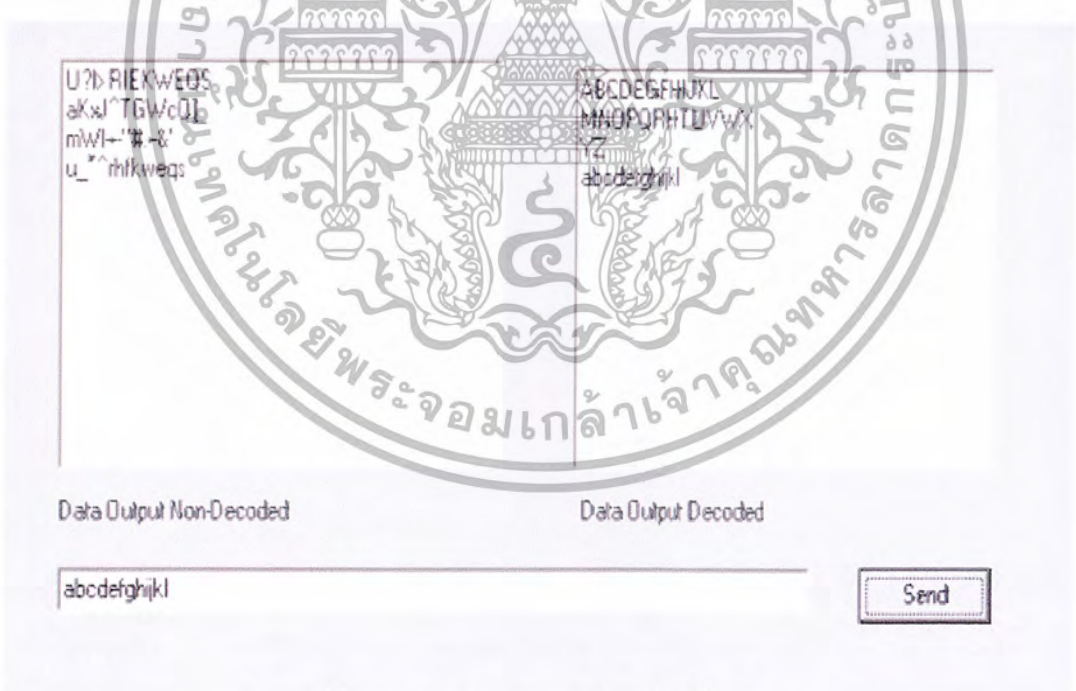
4.1 แสดงผลการเข้ารหัสลับ

จากการทำการทดลองการนำสัญญาณดิจิตอลจากคอมพิวเตอร์ ที่อยู่ในรูปแบบรหัสแอสกี นำมาทำการเข้ารหัสโดยส่งสัญญาณผ่านพอร์ต RS – 232

จากการทดลองในขั้นแรกการเชื่อมต่อกันระหว่างสาย RS – 232 กับคอมพิวเตอร์สามารถเชื่อมต่อกันได้โดยทราบผลจากการแสดงสัญญาณไฟในตัวบอร์ดว่าเมื่อมีสัญญาณเข้ามาจะแสดงไฟติด และถ้าไม่มีสัญญาณหรือไม่ได้รับสัญญาณไฟที่หลอด LED จะดับ

4.1.1 การเข้ารหัสลับข้อความ

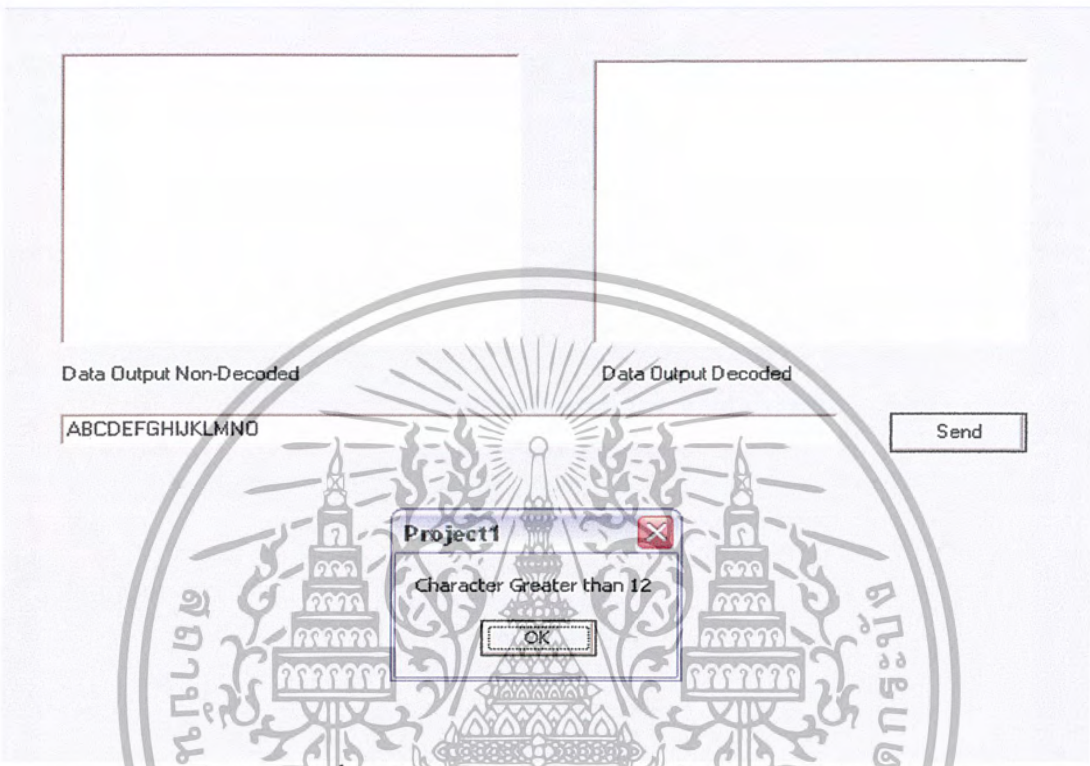
เมื่อทำการทดลองการเข้ารหัสเป็นชุด โดยจะเข้าทีละ 12 word พบว่าถ้าทำการส่งข้อมูลที่ละ 12 word FPGA จะทำการเข้ารหัสที่ละชุดทำให้ได้รหัสที่ผิดเพี้ยนออกมาดังภาพที่ 4.1



ภาพที่ 4.1 แสดงรูปแบบการเข้ารหัสครั้งละ 12 word

4.1.2 การส่งรหัสลับข้อความมากกว่า 12 word

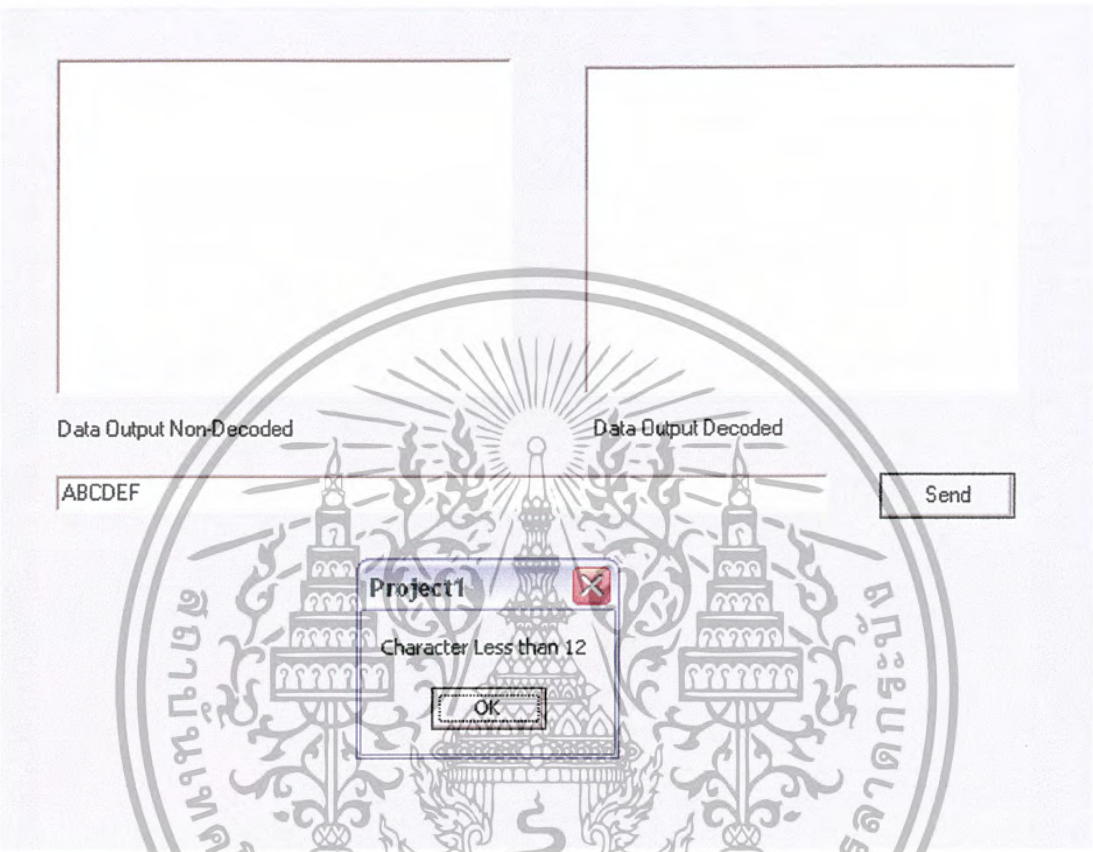
หากทำการส่งข้อมูลในแต่ละชุดซึ่งจำนวนมากกว่า 12 word ทางตัว FPGA จะไม่ดำเนินการต่อ และจะส่งข้อความเตือน Character Greater 12 ดังภาพที่ 4.2



ภาพที่ 4.2 แสดงการส่งสัญญาณมากกว่า 12 word

4.1.3 การส่งรหัสลับข้อความน้อยกว่า 12 word

หากทำการส่งข้อมูลในแต่ละชุดซึ่งจำนวนน้อยกว่า 12 word ทางตัว FPGA จะไม่ดำเนินการต่อ และจะส่งข้อความเตือน Character Less than 12 ดังภาพที่ 4.3



ภาพที่ 4.3 แสดงการส่งสัญญาณน้อยกว่า 12 word

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

จะเห็นได้ว่า เราสามารถนำสัญญาณจากคอมพิวเตอร์ ในรูปแบบของรหัสแอสกีมาทำการเข้ารหัส โดยผ่าน FPGA ได้และส่งข้อมูลทั้งที่เข้ารหัสและถอดรหัสกลับไปยังคอมพิวเตอร์ โดยผ่านทาง RS – 232 ได้

ในส่วนของการเข้ารหัสลับข้อมูลและการถอดรหัสลับข้อมูลโดยใช้อัลกอริทึมซึ่งจะแบ่งการศึกษาเป็นสองส่วนคือ ส่วนแรกเป็นการศึกษาและจำลองโปรแกรมโดยใช้ภาษา VHDL และส่วนที่สองเป็นการทดลองโปรแกรมลงบนอุปกรณ์ FPGA เพื่อใช้งาน จากการศึกษาและทดลองสามารถสรุปผลได้ดังนี้

5.1 การศึกษาและจำลองโปรแกรมโดยใช้ VHDL

สามารถใช้อัลกอริทึมเข้ารหัสและถอดรหัสข้อมูลประเภทข้อความได้

5.2 การทดลองโปรแกรมลงบนอุปกรณ์ FPGA

1. สามารถโปรแกรมวงจรที่เป็นส่วนประกอบของวงจรรวมที่สังเคราะห์จากภาษา VHDL ลงบนอุปกรณ์ FPGA ได้
2. สามารถวัดผลของวงจรที่เป็นส่วนประกอบของวงจรรวมที่โปรแกรมลงอุปกรณ์ FPGA ได้

5.3 ข้อเสนอแนะ

ถ้าโปรแกรมวงจรรวมที่สังเคราะห์ขึ้น มีการใช้ทรัพยากรของอุปกรณ์ FPGA มากเกินไปจะทำให้ไม่สามารถสื่อสารระหว่างอุปกรณ์ FPGA และคอมพิวเตอร์ได้

บรรณานุกรม

[1] คเชนทร์ แจ่มกมล. *การเข้ารหัสลับโดยใช้ซีนโครมของบล็อคโค้ดเชิงเส้นและการสลับบิทโดยอาศัยการกำเนิดแบบสุ่ม*. วิทยานิพนธ์วิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, กรุงเทพฯ, 2539.

[2] ชำนาญ ปัญญาใส, วัชรากร หนูทอง,ภาษา. *VHDL สำหรับออกแบบวงจรดิจิตอล*. กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2547

[3] ณรงค์ ทองฉิม และ เจริญ วงษ์ชุ่มเย็น. *ออกแบบไอซีดิจิตอลด้วย FPGA และ CPLD ภาคปฏิบัติโดยใช้ภาษา VHDL ซอฟต์แวร์ทูล ISE WebPACK*. พิมพ์ครั้งที่ 1. กรุงเทพฯ : บริษัท วี.พริ้นท์ (1991) จำกัด

[4] ลัญฉกร วุฒิสิทธิกุลกิจ, ธงชัย โรจน์กังสดาล, วรากร ศรีเชวงทรัพย์, นภดล พรมภักษร, สุวิทย์ นาคพีระยุทธ. *วิทยาการรหัสลับเบื้องต้น*. กรุงเทพฯ : สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2548

[5] อภิชาติ ภู่พลับ. *เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาร์ดแวร์ด้วย Visual Basic*. พิมพ์ครั้งที่ 1. นนทบุรี : Infopress Developer Book, 2546

[6] Stephen Brown and Zvonko Vranesic. "Fundamentals of Digital Logic with VHDL Design." McGraw-Hill Company.

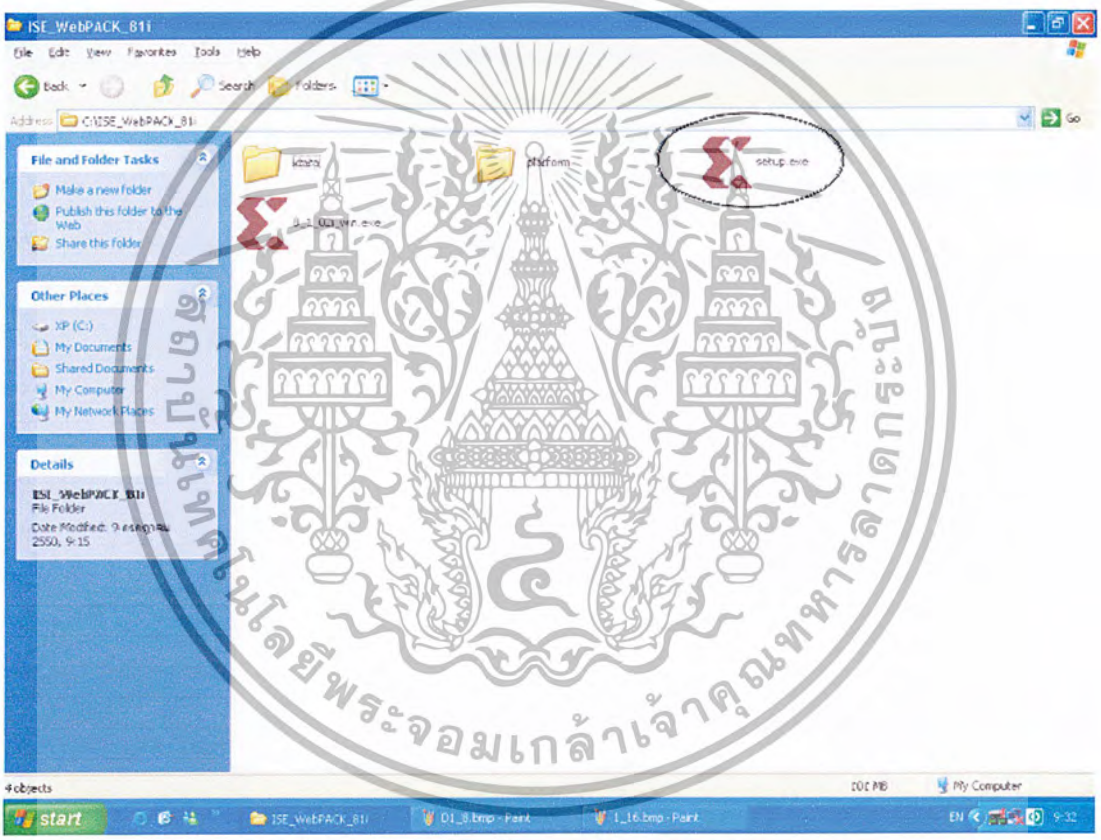
[7] http://www.xilinx.com

ภาคผนวก

ภาคผนวก ก
การติดตั้งโปรแกรม

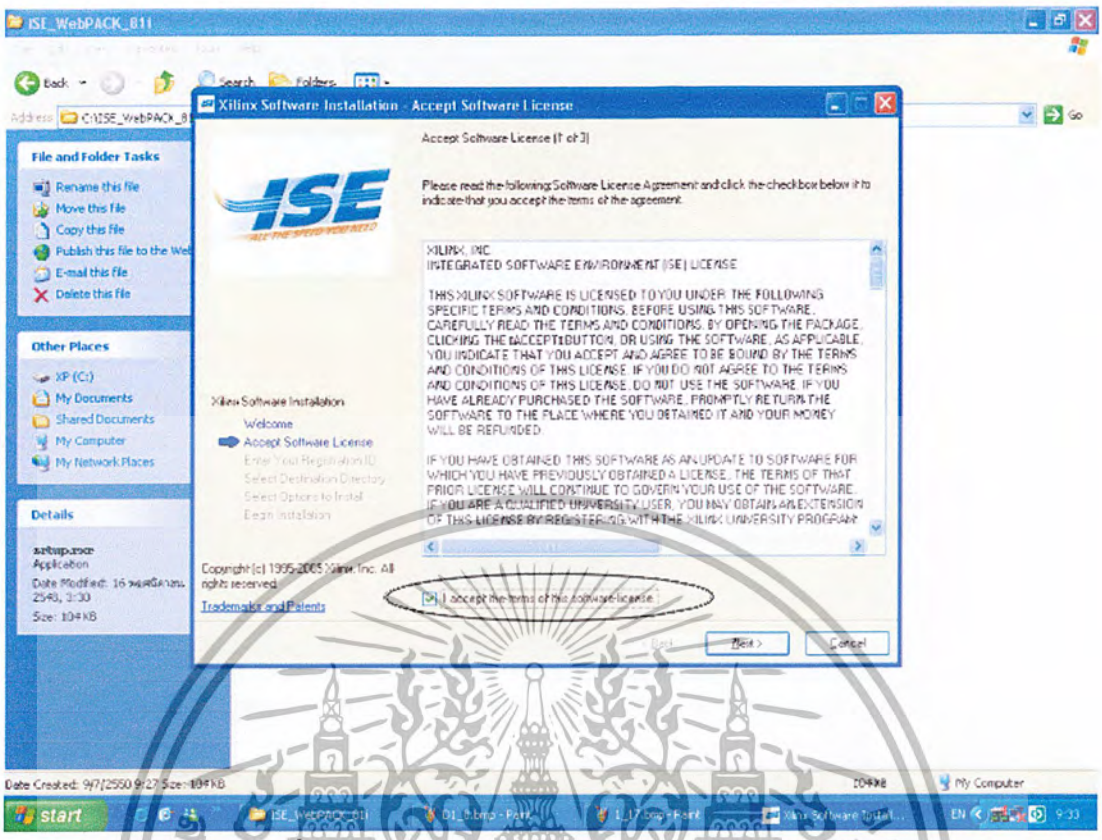
1. วิธีการติดตั้ง ISE WebPACK 8.1i

1. Copy ไฟล์ใน Folder ชื่อ ISE_WebPACK_81i จากแผ่น DVD ไปไว้ในไดรฟ์ C เมื่อดับเบิลคลิก Folder ชื่อ ISE_WebPACK_81i (ในไดรฟ์ C) แล้วจะได้ดังภาพที่ ก.1 แต่ถ้าดาวน์โหลดไฟล์ของซอฟต์แวร์ทูลจาก Xilinx โดยตรงให้ดับเบิลคลิกไฟล์ชื่อ WebPACK_81i_SFD.exe

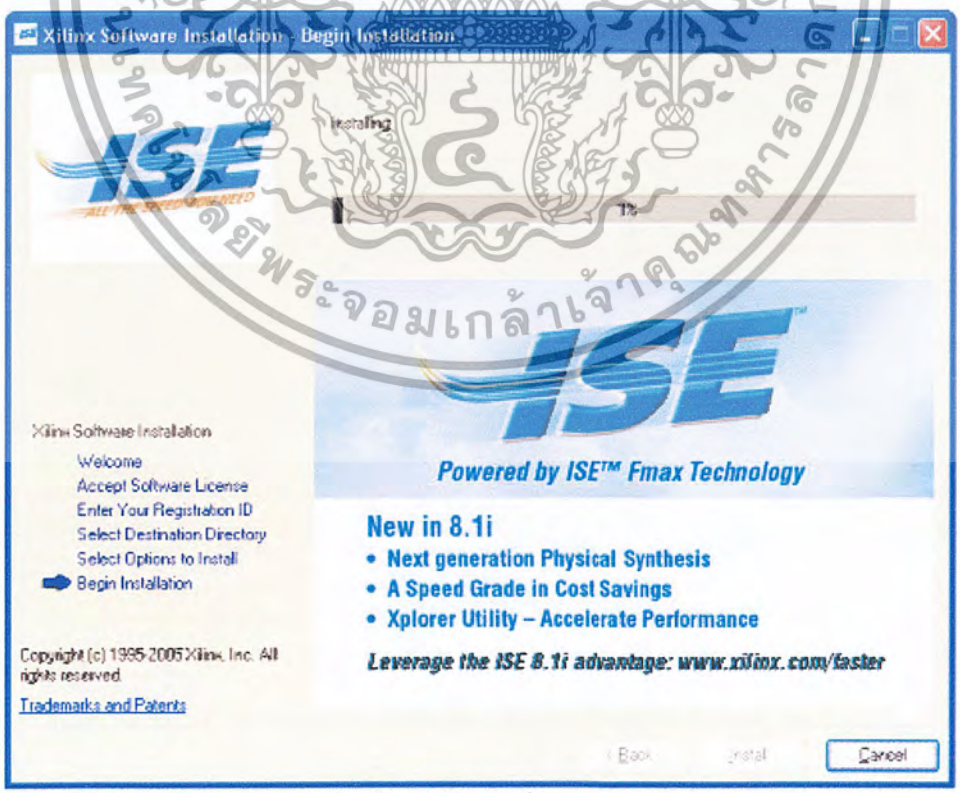


ภาพที่ ก.1 แสดงภาพเมื่อรวมไฟล์ ISE WebPACK 8.1i

2. ดับเบิลคลิกที่ไฟล์ setup.exe แล้วจะได้หน้าต่าง Accept Software License ให้คลิก "✓" ที่หน้า "I accept the term of this software license" ดังภาพที่ ก.2 แล้วคลิกปุ่ม Next แล้วจะได้หน้าต่างถัดไป (ทำซ้ำจนครบ 3 ครั้ง) เสร็จแล้วคลิกปุ่ม Next ไปอีก 2 ครั้งและคลิกปุ่ม Install แล้วโปรแกรมจะเริ่มติดตั้งไฟล์ต่างๆ ลงในคอมพิวเตอร์ ดังภาพที่ ก.3

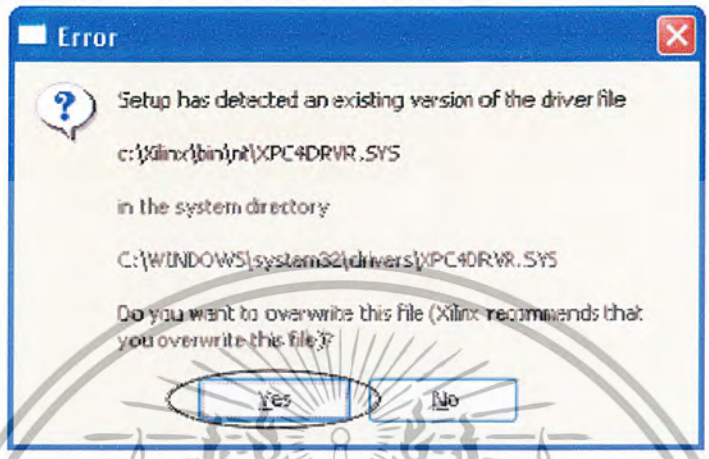


ภาพที่ ก.2 หน้าต่าง Accept Software License (มี 3 หน้า)



ภาพที่ ก.3 เริ่มติดตั้งซอฟต์แวร์

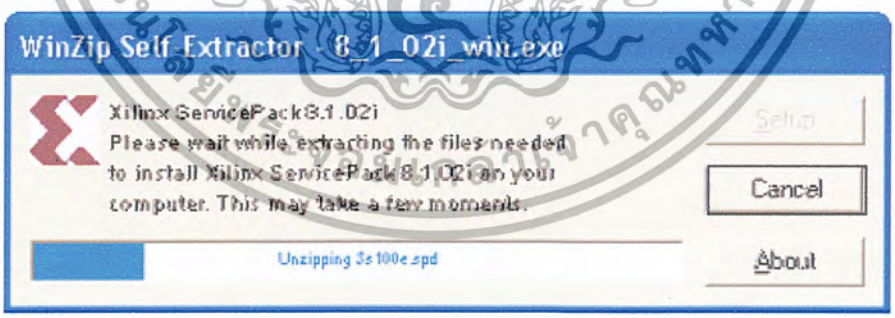
3. ในกรณีที่คอมพิวเตอร์เคยติดตั้งซอฟต์แวร์เวอร์ชั่นก่อนหน้านั้น ในระหว่างติดตั้งอาจมีหน้าต่าง Error ปรากฏขึ้นดังภาพที่ ก.4 ให้คลิก Yes แล้วจะได้หน้าต่างถัดไปเพื่อทำการ Setup เมื่อ Setup แล้วเสร็จให้คลิก OK ก็จะถือว่าติดตั้งซอฟต์แวร์แล้วเสร็จ



ภาพที่ ก.4 ยกเลิกไฟล์ Driver ของซอฟต์แวร์เวอร์ชั่นเก่า

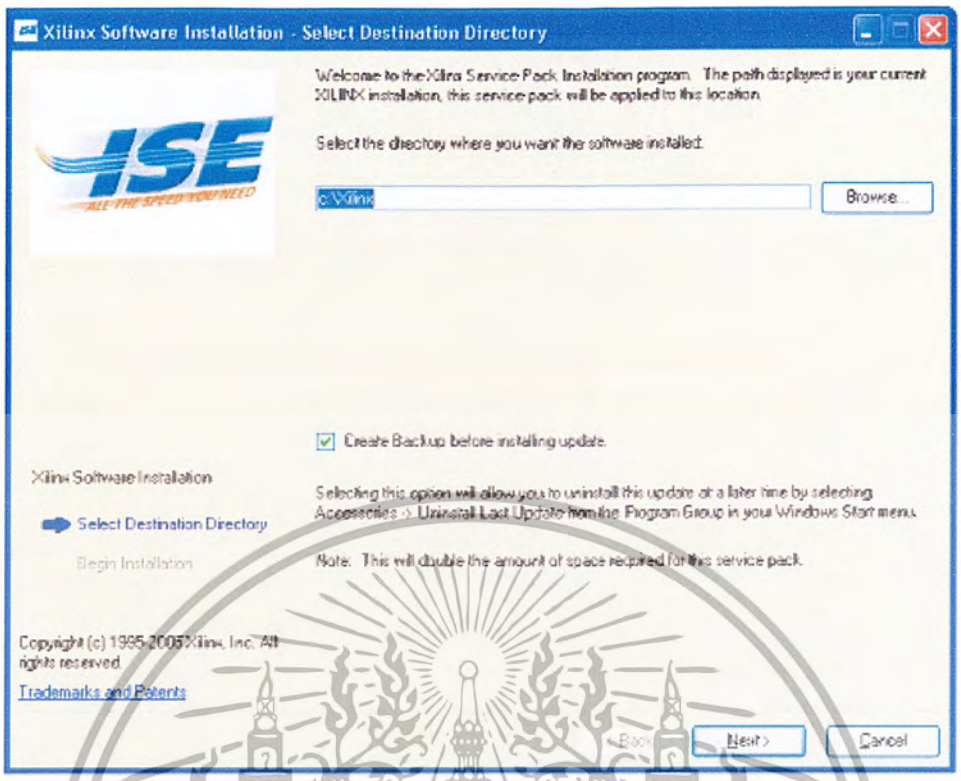
2. วิธีการติดตั้ง Service Pack 2

1. ในภาพที่ ก.1 จะมีไฟล์ Service Pack 2 (ชื่อ 8_1_02i_win.exe) อยู่ จากนั้นดับเบิลคลิกไฟล์ 8_1_02i_win.exe แล้วจะได้ดังภาพที่ ก.5 แล้วโปรแกรมจะทำการแตกไฟล์ .exe ออกโดยอัตโนมัติ เมื่อแตกไฟล์เสร็จแล้วจะได้ดังภาพที่ ก.6

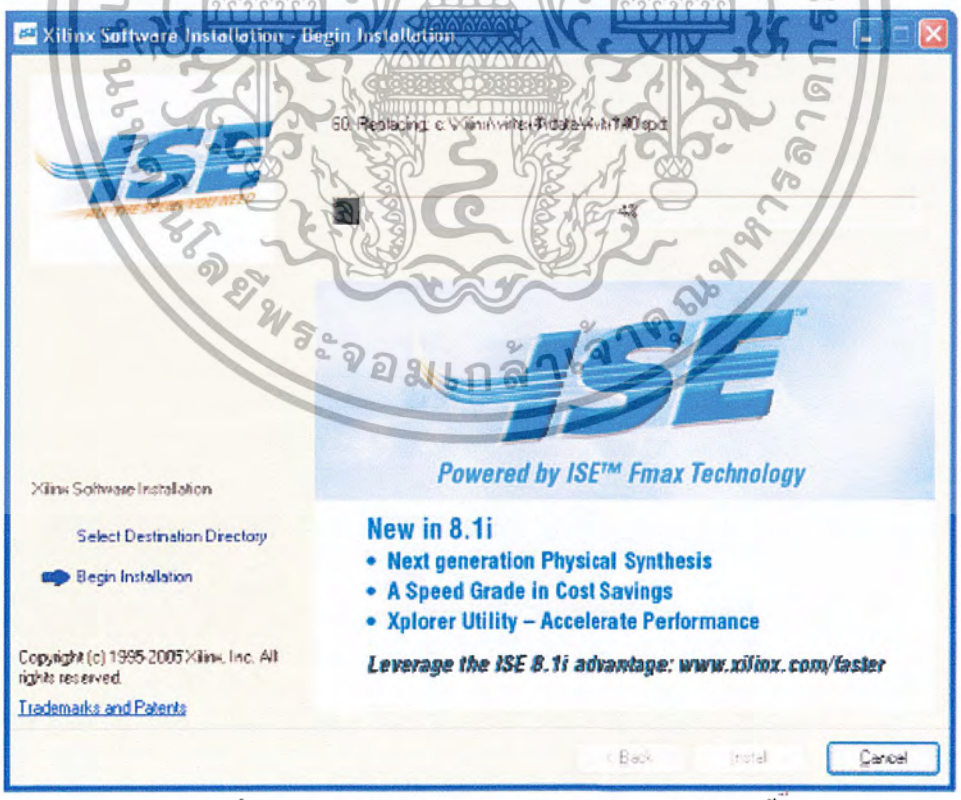


ภาพที่ ก.5 โปรแกรมจะแตกไฟล์ 8_1_02i_win.exe

2. คลิก Next ในภาพที่ ก.6 เพื่อเริ่มแบ็คอัพไฟล์ก่อนทำการติดตั้ง เมื่อแล้วเสร็จให้คลิก OK จากนั้นคลิก Install เพื่อเริ่มติดตั้งซอฟต์แวร์ดังภาพที่ ก.7 ไปจนแล้วเสร็จ คลิก OK ก็จะถือว่าการติดตั้งซอฟต์แวร์แล้วเสร็จสมบูรณ์



ภาพที่ ก.6 เลือก Directory และ Create Backup before installing update



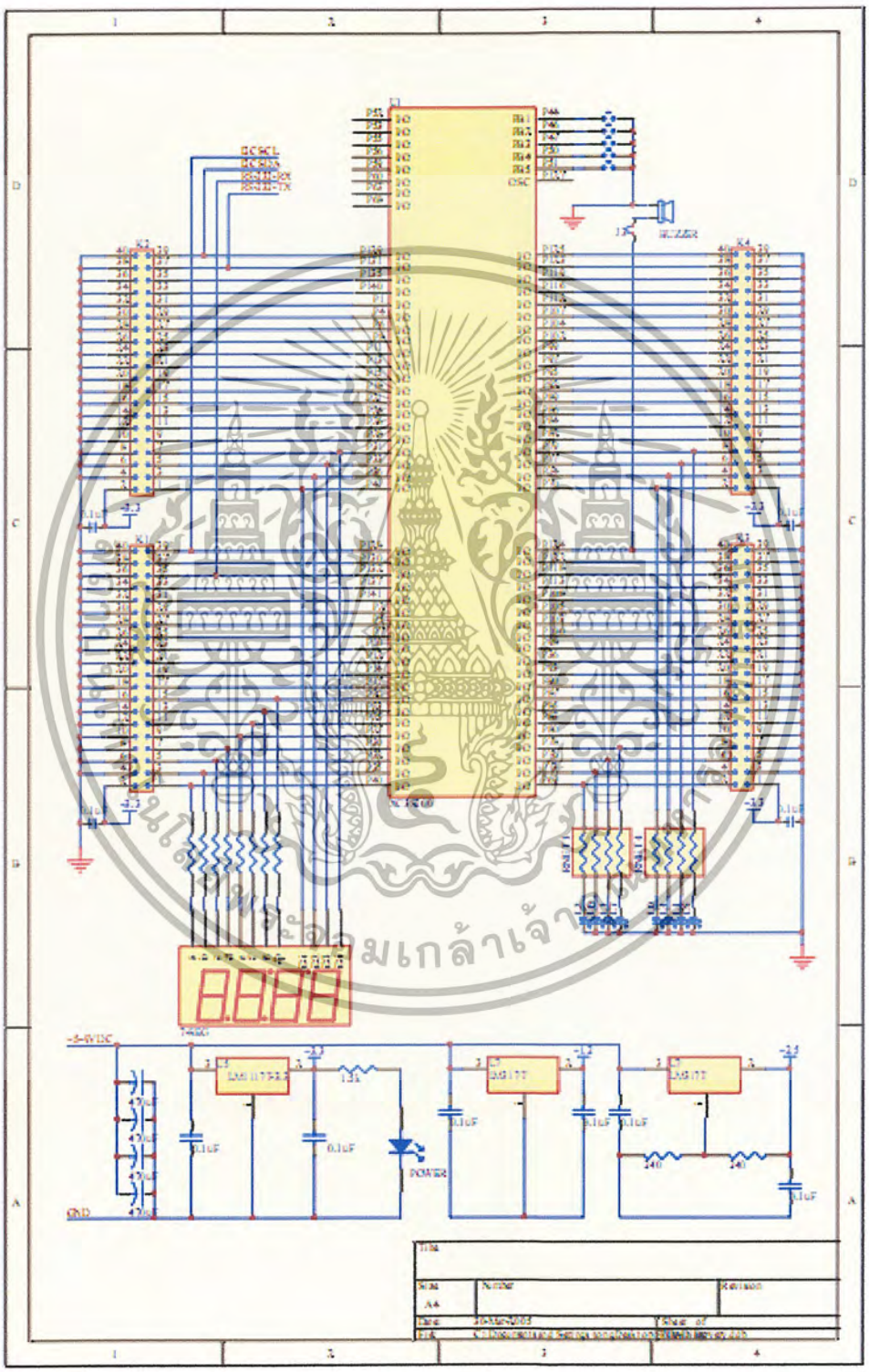
ภาพที่ ก.7 หน้าต่าง Begin Installation ขณะกำลังติดตั้ง

หมายเหตุ

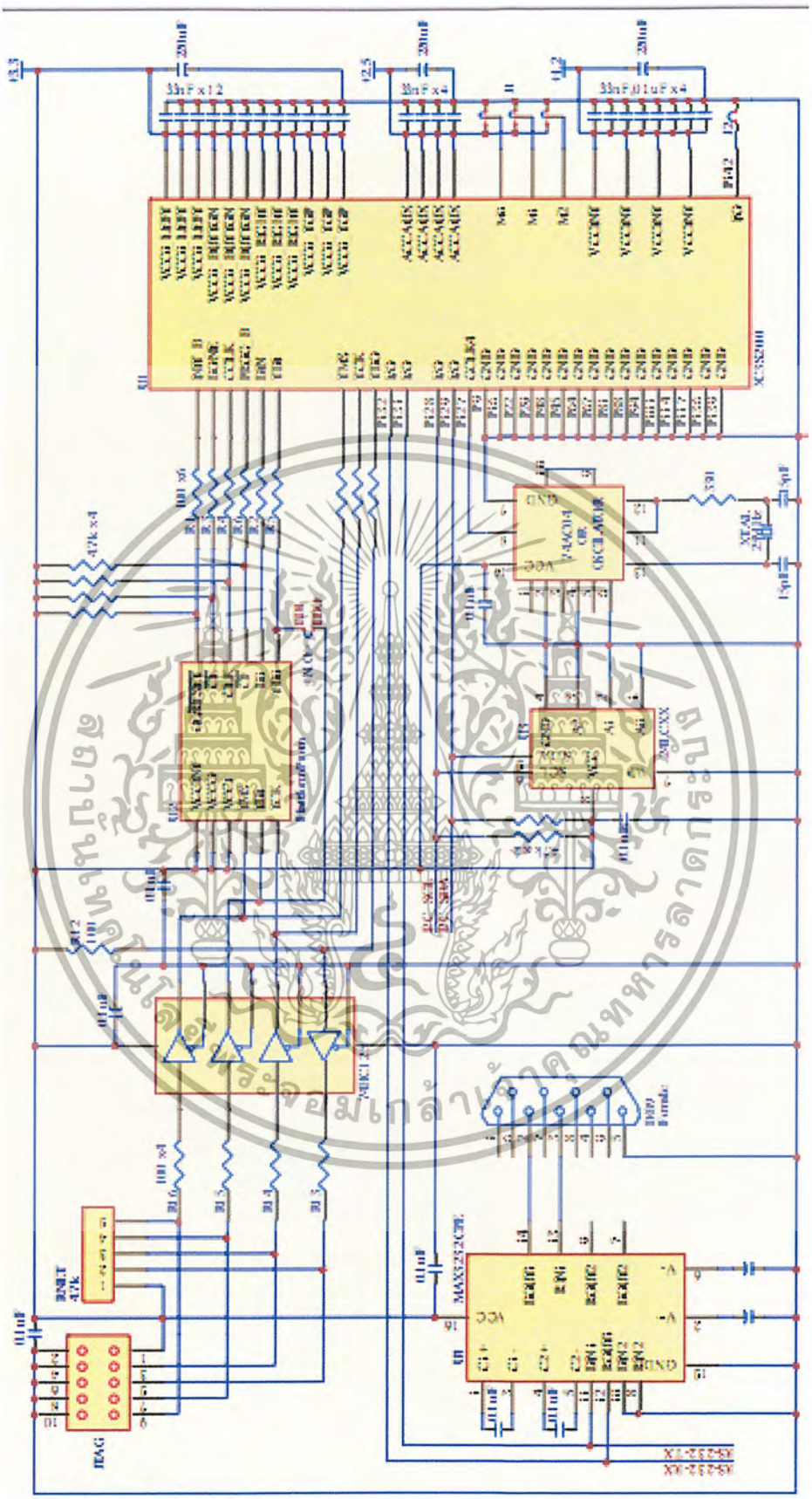
การใช้ซอฟต์แวร์ ISE WebPACK 8.1i นั้นมีความจำเป็นต้องติดตั้งซอฟต์แวร์ชื่อ Adobe Reader 7 ขึ้นไปควบคู่กันไปด้วยเพื่อใช้สำหรับอ่านไฟล์ .pdf ดังนั้นหากยังไม่ได้ติดตั้งก็ให้ผู้ใช้จะติดตั้งซอฟต์แวร์ชื่อ Adobe Reader นี้ด้วย

ภาคผนวก ข

ไดอะแกรมผังวงจร



ภาพที่ ข.1 รายละเอียดไดอะแกรมผังวงจรบอร์ดทดลองอเนกประสงค์รุ่น FPGA Discovery – III XC3S200



ภาพที่ ข.2 รายละเอียดไดอะแกรมผังวงจรบอร์ดทดลองอเนกประสงค์รุ่น FPGA Discovery – III XC3S200

ภาคผนวก ค

รหัสแอสกี

ตารางที่ ค.1 ตารางรหัสแอสกี

				b7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b6	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b3	b2	b1	b0																	
0	0	0	0					@	P	`	p			ฐ	ภ	ะ	เ	๐
0	0	0	1					!	A	Q	a	q		ก	ฑ	ม	ั	แ	๑
0	0	1	0					"	B	R	b	r		ข	ฒ	ย	า	โ	๒
0	0	1	1					#	C	S	c	s		ฃ	ณ	ร	ำ	ใ	๓
0	1	0	0					$	D	T	d	t		ค	ด	ฤ	ิ	ไ	๔
0	1	0	1					%	E	U	e	u		ฅ	ต	ล	ี	ๅ	๕
0	1	1	0					&	F	V	f	v		ฆ	ถ	ฦ	ึ	ๆ	๖
0	1	1	1					'	G	W	g	w		ง	ท	ว	ื	็	๗
1	0	0	0					(	H	X	h	x		จ	ธ	ศ	ุ	่	๘
1	0	0	1					)	I	Y	i	y		ฉ	น	ษ	ู	้	๙
1	0	1	0					*	J	Z	j	z		ช	บ	ส	ฺ	๊	๚
1	0	1	1					+	K	[	k	{		ซ	ป	ห		๋	๛
1	1	0	0					,	L	\	l	|		ฌ	ผ	ฬ		์	
1	1	0	1					-	M	]	m	}		ญ	ฝ	อ		๎	
1	1	1	0					.	N	^	n	~		ฎ	พ	ฮ		๏	
1	1	1	1					/	O	_	o			ฏ	ฟ	ฯ	฿	๐	