การออกแบบวงจรป้องกันการคัดลอกซอฟต์แวร์โดยใช้เอฟพีจีเอ

Hardlock by FPGA

โดย

นายโอภาส สุวิกะปกรณ์กุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบวงจรป้องกันการคัดลอกซอฟแวร์โดยใช้เอฟพีจีเอ

ผู้จัดทำ

นายโอภาส สุวิกะปกรณ์กุล รหัส 38014675

..... อาจารย์ที่ปรึกษา

(อาจารย์สมศักดิ์ มิตะถา)

การออกแบบวงจรป้องกันการคัดลอกซอฟต์แวร์โดยใช้เอฟพีจีเอ

โอภาส สุวิกะปกรณ์กุล 38014675
อาจารย์สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการออกแบบวงจรป้องกันการคัดลอกซอฟแวร์โดยใช้เอฟพีจีเอ ซึ่งเป็นวงจรที่ใช้ในการป้องกันการคัดลอกซอฟต์แวร์ โดยใช้เอฟพีจีเอ ซึ่งเป็นอุปกรณ์ที่มีความสามารถอัดข้อมูลลงไป ทำให้สามารถใช้งานได้ตามที่เราต้องการโดย ใช้ภาษาวิเฮชดีแอล ซึ่งเป็นภาษาที่บรรยายถึงลักษณะการทำงานของฮาร์ดแวร์ ซึ่งสามารถนำโปรแกรมที่เขียนได้โปรแกรมลงไปในเอฟพีจีเอ ในการออกแบบวงจรนี้ได้ทำการป้องกันการคัดลอกโดยการเข้ารหัสข้อมูลเป็นแบบดีอีเอส (DES (Data Encryption Standard)) เพื่อป้องกันมิให้ผู้ที่จะคัดลอกซอฟต์แวร์อย่างผิดกฎหมายแอบดูข้อมูลสำคัญในซอฟแวร์ที่เราป้องกันไว้ ในการเชื่อมต่อระหว่างวงจรที่ออกแบบกับคอมพิวเตอร์เราจะใช้มาตราฐานยูเอสบีในการเชื่อมต่อ ซึ่งเป็นมาตราฐานใหม่ของการติดต่อแบบอนุกรม เราคิดว่าการที่เราออกแบบวงจรนี้จะเป็นแนวทางให้กับผู้ที่ต้องการจะสร้างวงจรชนิดเดียวกันนี้ในอนาคต

Hardlock Design by FPGA

Ophat Suvikapakomkul 38014675
Somsak MitaTha Advisor
1998

ABSTRACT

This thesis is to design hardlock by FPGA, which can program anything that you want it to be by using VHDL for design structure in FPGA. VHDL is language that describe characteristic of hardware and it can program FPGA. Hardlock is circuit that can protect software from person who transgress a law such as copy license software though software owner doesn't allow. The security of data we use DES algorithm to encryption and decryption data for protect data from person such as Hacker ,Cracker etc. By the way interface between PC and hardlock use USB standard. USB standard is new version of serial communication port. We wish in the future that this design is way for next developer who want to build hardlock in commercial.

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

ปริญญานิพนธ์ฉบับนี้เสนอการออกแบบวงจรป้องกันการคัดลอกซอฟต์แวร์โดยใช้เอฟพีจีเอ ซึ่งเป็นวงจรที่ใช้ป้องกันการคัดลอกซอฟต์แวร์อย่างผิดกฎหมาย โดยการออกแบบเอฟพีจีเอนั้น ได้ใช้ ภาษาวีเฮชดีแอล ในการกำหนดลักษณะ และโครงสร้างของวงจร และการป้องกันข้อมูลเราได้ใช้วิธีการเข้าและถอดรหัสแบบดีอีเอส ซึ่งเป็นการเข้ารหัสที่นิยมใช้กันทั่วไปในการป้องกันข้อมูล การเชื่อมต่อระหว่างคอมพิวเตอร์กับวงจรเราใช้มาตราฐานยูเอสบี ในการออกแบบนั้นแบ่งออกเป็น 2 ส่วน คือ การทำงานส่วนฮาร์ดแวร์ กับการทำงานส่วนซอฟต์แวร์ ในด้านการออกแบบส่วนฮาร์ดแวร์นั้น มีในส่วนของการออกแบบดีอีเอสบนวีเฮชดีแอล และการรับส่งข้อมูลแบบมาตราฐานยูเอสบี โดยวีเฮชดีแอล เราได้ทำวงจรทดลอง แต่เนื่องจากปัญหาบางประการทำให้ไม่ได้สามารถ ทดลองการใช้งานกับเครื่องคอมพิวเตอร์ได้ ทางผู้จัดทำได้เขียนโปรแกรมวีเฮชดีแอล ที่ทำงานในส่วนต่างๆ ในภาคผนวก ในทางด้านซอฟต์แวร์นั้นเราได้แบ่งออกเป็น ส่วนที่เข้ารหัสก่อนที่จะนำไปใช้จริง กับส่วนที่นำไปใช้กับวงจรป้องกันการคัดลอกจริง ในการออกแบบนั้นสามารถทำให้ใช้งานได้จริงและสามารถทำงานได้เป็นผลดี แต่เนื่องจากประสบปัญหาที่ด้านส่วนของฮาร์ดแวร์ ทางผู้จัดทำจึงจำเป็นต้องตัดส่วนเชื่อมต่อทางยูเอสบีออกจากโปรแกรมเพราะไม่ได้สามารถทำลองใช้ได้กับวงจรที่เราจำลองขึ้นมาได้ การออกแบบนั้นยอมรับว่าการออกแบบนั้นยังไม่ได้รับผลสำเร็จในบางส่วน และในบางส่วนยังไม่ได้ทดลองใช้จริง อันเนื่องจากประสบปัญหาหลายอย่าง ซึ่งแม้ว่าจะใช้ได้จริงก็อาจจะมีจุดบกพร่อง และ ไม่สมบูรณ์ ในบางส่วน จึงทำให้เกิดความไม่สมบูรณ์ในโครงงานนี้ในบางส่วน ซึ่งผู้จัดทำก็จะขอรับคำติชมและจะปรับปรุงแก้ไขให้ดีขึ้นต่อไป หวังว่าโครงงานฉบับนี้จะเป็นประโยชน์ต่อท่านผู้ที่ต้องการออกแบบวงจร ที่ใช้เอฟพีจีเอต่อไป

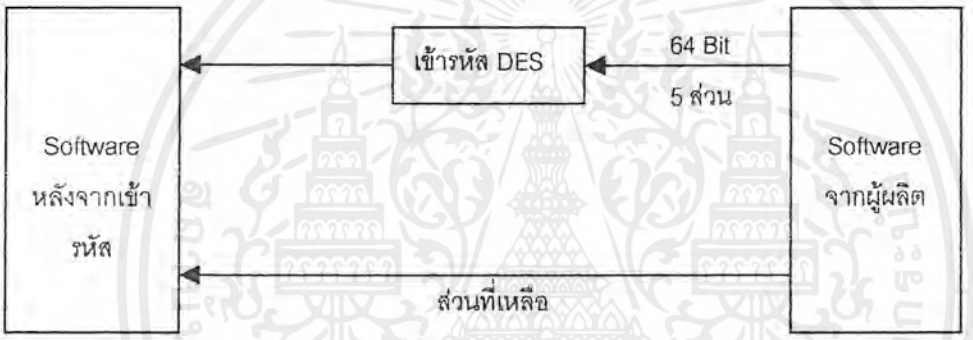
บทที่ 2
หลักการเบื้องต้น

ในงานการผลิต Software ในปัจจุบันนั้น มีความยากลำบากในการผลิตจึงทำให้ราคาในการขายมีราคาสูง เพื่อให้คุ้มทุนที่เสียไปในการผลิต ทำให้ผู้ใช้ Software ต้องเสียค่าใช้จ่ายมาก เมื่อต้องซื้อ Software ที่มีลิขสิทธิ์ถูกต้องตามกฎหมาย จึงมีผู้ไม่ประสงค์ดีทำการละเมิดลิขสิทธิ์ของทางผู้ผลิตโดยการทำการคัดลอก software นั้นออกขายในราคาถูกทำให้ผู้ผลิต Software เกิดความเสียหาย ทางผู้ผลิตจึงมีความคิดที่จะป้องกันการคัดลอกนั้นให้น้อยที่สุด โดยใช้แนวคิดในแนวต่างๆ เช่น การทำให้ Software นั้นสามารถใช้ได้ก็ต้องผ่านการลงทะเบียนโดยใช้ รหัส หรือ Serial number เพื่อให้ผู้ใช้ใช้ software ของตนที่ถูกต้องตามกฎหมายวิธีการนี้ก็สามารถใช้กับผู้ใช้ที่ไม่ค่อยมีความรู้เกี่ยวกับคอมพิวเตอร์ได้ แต่สำหรับพวกที่มีความเชี่ยวชาญแล้วพวกเขาสามารถทำการ Hack หรือ Crack software นั้นๆ ได้และบางทีก็เผยแพร่ไปยังใน Internet ด้วย ทำให้การละเมิดลิขสิทธิ์เกิดขึ้นมากขึ้นเรื่อย แนวคิดในการใช้ Hardware มาช่วยเสริมการป้องกันการคัดลอก Software จึงเกิดขึ้นซึ่งเราเรียกอุปกรณ์ที่ป้องกันการคัดลอก Software นี้ว่า Hardware lock (Hardlock)

ในตัวอย่างที่ Software ที่ใช้ Hardlock นั้นมักจะเป็น Software ที่มักจะไม่แพร่หลายในทุกๆ กลุ่มของผู้ใช้ในงานประเภทต่างๆ มักจะเป็น Software ที่ใช้กันเฉพาะงานมากกว่า เช่น Protel ที่เป็นโปรแกรมที่ใช้ในการทางอิเลคโทรนิค ในการผลิต Hardlock นั้น อาจจะทำให้ต้นทุนสูงขึ้น หรือแม้กระทั้งว่าทำให้ ราคา software สูงขึ้นด้วย แต่มันก็ช่วยให้ผู้ผลิต software ไม่เสียผลประโยชน์ที่ควรจะได้กับผู้ที่ต้องการใช้ Software ของผู้ผลิต โดยทั่วไปแล้ว Hardlock จะถูกผลิตโดยใช้ Single Chip จำพวก Microcontroller ,FPGA ,หรือ พวก Chip ที่เป็น พวก Gate Array ขึ้นอยู่กับต้นทุน และความซับซ้อนของ Hardlock ที่ผู้ผลิต software นั้นต้องการซึ่งใน Hardlock ที่เราจะผลิตนั้นเราใช้ Chip FPGA ตัวใหม่ของบริษัท Xilink โดยมีชื่อของผลิตภัณฑ์ว่า SPATRAN © ซึ่งเป็น Chip ที่มีราคาต่ำเมื่อเทียบกับ FPGA หลายเท่าตัวส่วนรูปลักษณ์ก็สามารถใช้แทนกันได้เป็นอย่างดี รายละเอียดของ Chip ตัวนี้จะอธิบายภายหลัง ในการเชื่อมต่อ Hardlock กับคอมพิวเตอร์นั้นสามารถทำได้หลายแบบอย่างเช่นการต่อเข้ากับ Communication ports, Printer ports, หรือ จะเป็น Card ISA, PCI อันนี้ขึ้นกับผู้ผลิตต้องการความเหมาะสมในงานอย่างไร ส่วนเราจะใช้การเชื่อมต่อใน Port ตัวใหม่ที่ชื่อว่า USB (Universal Serial Bus) ซึ่งมีความเร็วสูงมากในการทำงาน รายละเอียดของ USB จะอธิบายในภายหลัง

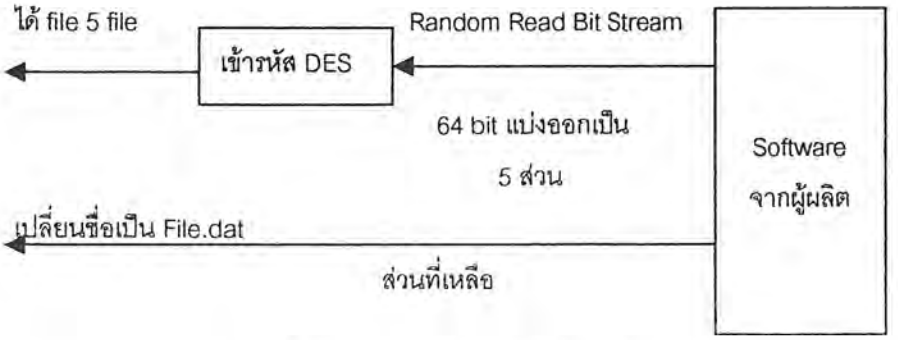
แนวคิดในการประดิษฐ์ Hardlock

แนวคิดของเรา Hardlock จะใช้ได้กับทุกชนิดของ Software โดยไม่ต้องเพิ่ม code ลงไปใน Software เลย อาจจะเรียกได้ว่าเป็น Hardlock ที่เป็นอเนกประสงค์ ใช้ได้ทุกๆ งาน โดย Software ที่ใช้นั้นทำงานอยู่บนระบบปฏิบัติการ Windows 95 หรือ Windows 98 ส่วนใน version Dos นั้นเราคิดว่ามันได้ล้าสมัยไปแล้วที่ผู้ผลิต Software จะผลิต Application ขนาดใหญ่บนระบบปฏิบัติการ Dos ในการ Lock Software นั้นเราจะ ทำการ Lock โปรแกรมหลักที่ใช้ในการรันโปรแกรม โดยเราจะเข้ารหัสโปรแกรมหลักโดยใช้ทฤษฎีของ DES เข้าช่วย ซึ่งเพื่อความรวดเร็วในการรัน Software ตอนใช้งานจริงเราจะเข้ารหัสโปรแกรมหลักเพียง 5 ส่วนส่วนละ 64 บิต ของโปรแกรมหลักส่วนที่เหลือจะไม่ทำอะไร ในส่วนนี้จะอยู่ในช่วงของผู้ผลิต Software นำ ผลิตภัณฑ์ของตนมา Lock บน Hardlock ดังรูป



รูปที่ 2.1 การทำงานของซอฟต์แวร์ตัวแรกโดยรวม

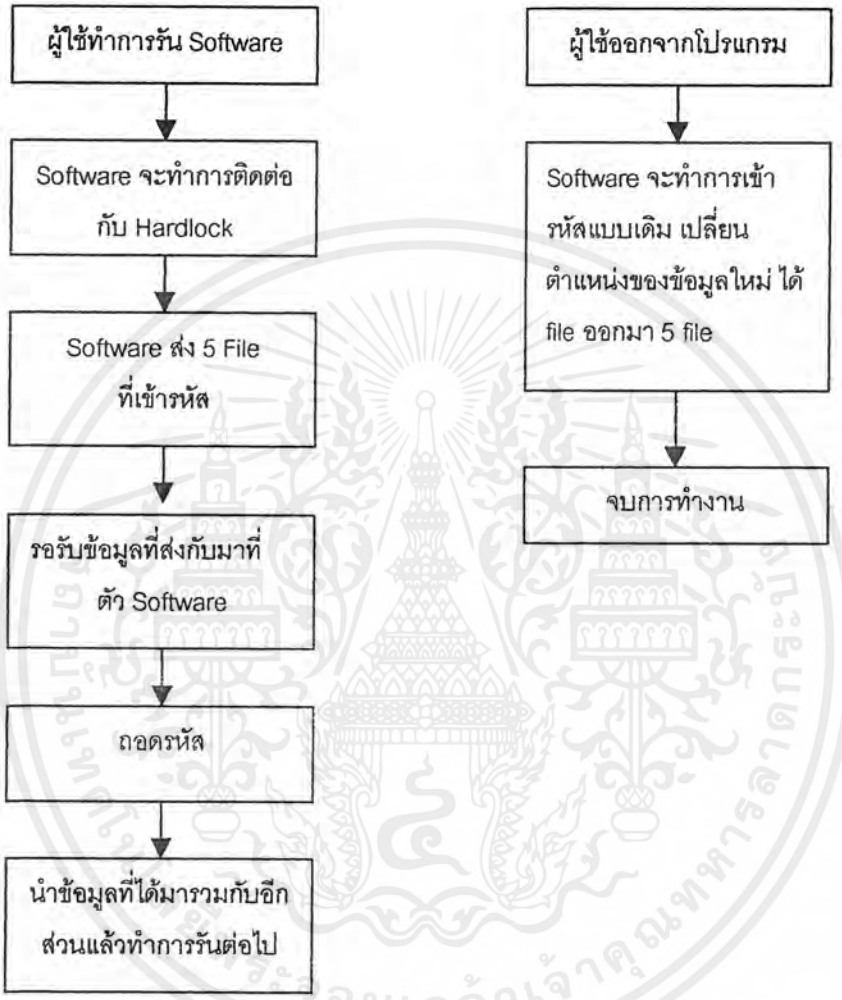
ในรูปแสดงถึงการวิธีการที่ผู้ผลิตนำ Software ของตนมาเข้าใช้ Hardlock โดยส่วนการเข้ารหัสนั้น Key ที่ใช้จะเป็นความลับของทางผู้ผลิตเอง Key นี้จะเป็น Key เดียวกับ Key ที่ใช้บน Hardlock เพื่อให้ Hardlock สามารถถอดรหัสข้อมูลเป็นข้อมูลจริงเพื่อทำการรัน Software ต่อไป แล้วส่วนนี้เราจะใช้ Software ที่เขียนขึ้นในงานนี้โดยเฉพาะ ซึ่งมีหลักการทำงานดังรูป



รูปที่ 2.2 การทำงานของซอฟต์แวร์ส่วนแรก

เมื่อเราทำเสร็จในขั้นตอนนี้เรียบร้อยแล้ว ก็ให้ทางผู้ผลิตทำการสร้างส่วนsetup โดยรวม Software ของ Hardlock ด้วยเพราะ Software จะรันได้ก็ต่อเมื่อมี Software ของเราไปด้วยกับ Hardlock อีก 1 ตัว หลักการใน

การทำงานของ Hardlock จริงๆ หลังจากที่ Install Software ลงใน Windows แล้วพอผู้ใช้ใช้โปรแกรมที่มี Hardlock ก็ต้องติดตั้ง Hardlock โดยต่อ Hardlock เข้าทาง USB ports แล้วรัน Software ของเรา โดย source code ของ Software ที่ใช้เข้ารหัสในตอนเบื้องต้น และ Software ที่ใช้รันกับโปรแกรมหลักนั้นจะอยู่ในภาคผนวก การทำงานของ Software นั้น อธิบายได้ดังรูป



รูปที่ 2.3 การทำงานของซอฟต์แวร์ส่วนสอง

บทที่ 3
การเข้ารหัส และถอดรหัส

หลักการถอดรหัสเบื้องต้น

การเข้ารหัส (Encryption) เป็นวิธีการป้องกันข้อมูลแบบหนึ่งที่มีการเปลี่ยนแปลงข้อมูลเดิมไปเป็นข้อมูลชุดใหม่ ซึ่งชุดข้อมูลที่ใช้ส่งออกจากพอร์ตเพื่อป้องกันผู้แอบดูข้อมูล ผู้เข้ารหัสจึงจำเป็นต้องทำการถอดรหัส (Decryption) เลี่ยนข้อมูลที่เข้ารหัสกลับไปเป็นข้อมูลชุดเดิม ในระบบที่ประกอบด้วยทั้งส่วนการเข้าและถอดรหัสนั้นเรียกว่า ระบบสร้างรหัส ดังรูป

รูปที่ 3.1 การเข้ารหัสและถอดรหัสเบื้องต้น

ข้อมูลปกติ (Plaintext) เขียนย่อว่า P จะเป็นการเรียงต่อกันของข้อมูลเป็นบิตๆ คือ $P = (p_1, p_2, ..., p_n)$ และข้อมูลที่ผ่านการเข้ารหัส (Ciphertext) เขียนได้เป็น $C = (c_1, c_2, ..., c_m)$ ขั้นตอนการเข้ารหัสก็คือ การแปลงระหว่างข้อมูลปกติไปเป็นข้อมูลเข้ารหัส จะเขียนแทนด้วย $C = E(P)$ ขณะที่การถอดรหัสจะเขียนสมการได้เป็น $P = D(C)$ และระบบการสร้างรหัสลับเขียนได้เป็น $P = D(E(P))$

ในการเข้าและถอดรหัสนั้น ในบางแบบอาจจะต้องมีกุญแจ (Key) มาเข้ารวมกับข้อมูลปกติเพื่อไปเป็นข้อมูลที่เข้ารหัส กรณีนี้สมการการเข้ารหัส คือ $C = E(K,P)$ โดยกุญแจ K จะเป็นค่าหรือขั้นตอนเฉพาะหนึ่ง ถ้ากุญแจนี้ใช้ทั้งการเข้าและการถอดรหัส สมการสร้างรหัสลับจะเป็น $P = D(K,E(K,P))$ แต่ในบางกรณีกุญแจที่ใช้ในการเข้าและการถอดรหัสนั้นเป็นคนละตัวกัน จะมีสมการเป็น $P = D(K_D,E(K_E,P))$ โดยที่ K_D คือกุญแจที่ใช้ในการถอดรหัส และ K_E คือกุญแจที่ใช้ในการเข้ารหัส

ขั้นตอนในการเข้ารหัสมีหลายแบบอาจแบ่งออกได้เป็น 2 แบบ ตามรูปแบบของการเข้ารหัส ตามชุดข้อมูล คือ

1. การเข้ารหัสแบบไบต์อักขระ (ASCII) มีวิธีการเช่น

   <u>การเข้ารหัสแบบ mono-alphabetic cipher หรือ Caesar cipher</u>

   วิธีการเข้ารหัสคือ การเพิ่มค่าของรหัส ASCII ของตัวอักษรที่ต้องการจะเข้ารหัส

   วิธีการถอดรหัสคือ การลดค่าที่ของรหัส ASCII เพิ่มขึ้นไป ให้กลับสู่ค่าเดิม

   <u>การเข้ารหัสแบบ poly – alphabetic cipher</u>

   วิธีการเข้ารหัสคือ เรียงตัวอักษร A-Z ออกไปเป็น row ที่ 0 และ ship Z มาที่หัวแทน A เป็น row ที่ 1 ทำอย่างนี้เรื่อยๆ จนครบถึง 25 row

   การเข้ารหัสเราส่งเอาตำแหน่งของตัวอักษร Aไปทำการ modulo กับ 26 จะได้เป็นความสัมพันธ์ของแถวที่ i ในเมทริกที่สร้างขึ้น และ j เป็นตำแหน่งของคอลัมน์ นำค่า i , j ไปดูในเมทริกซ์ได้ อักษรที่เข้ารหัสแล้ว

การเข้ารหัสแบบ Transposition cipher

วิธีการเข้ารหัสคือ นำชุดของอักษรที่ได้มาเรียงใหม่ในรูปของเมทริกซ์ เช่น Miss Piggy ถ้าจะใช้เมทริกซ์ 5 คอลัมน์จะได้ row ที่ 0 คือ Miss_ row ที่ 1 คือ Piggy (_ เป็นช่องว่าง) แล้วส่งข้อความไปเป็น column แทนที่จะส่งไปเป็น row

วิธีการถอดรหัสคือ นำชุดของข้อความมาเข้าไปในเมทริกซ์แล้วอ่านก็จะได้เหมือนเดิม

การเข้ารหัสแบบ RSA

วิธีการเข้ารหัสคือ นำอักษรที่ได้มาแทนค่าเป็นตัวเลข เช่น A=1 ,....,Z=26 แล้ว หาค่า p กับ q ซึ่งเป็นจำนวนเฉพาะ (ยิ่งมากยิ่งดี) โดย ค่าที่เข้ารหัสจะเท่ากับ ค่าที่ต้องการเข้ายกกำลัง p modulo กับ p คูณ q

วิธีการถอดรหัสคือ นำต้องหาค่า (k) ที่มายกกำลังกับ ค่าตัวเลขที่เข้ารหัส โดยหาได้จาก

$$k * p - 1 = 0 \text{ modulo } (p-1)(q-1)$$

ได้ค่า k นำมายกกำลังเลขที่เข้ารหัสมา แล้ว modulo กับ p คูณ q จะได้ค่าที่ถอดรหัสกลับเป็นค่าเดิม

2. การเข้ารหัสแบบบิต มีวิธีการเช่น

การเข้ารหัสแบบใช้ XOR

วิธีการเข้ารหัสคือ การสร้าง key เป็นเลขฐาน 2 แล้วนำไป XOR แบบบิตต่อบิตกับข้อมูลที่ต้องการเข้ารหัส

วิธีการถอดรหัส คือการนำ key เดิมมา XOR กับข้อมูลที่เข้ารหัสจะได้เป็นข้อมูลที่ถอดรหัสแล้ว

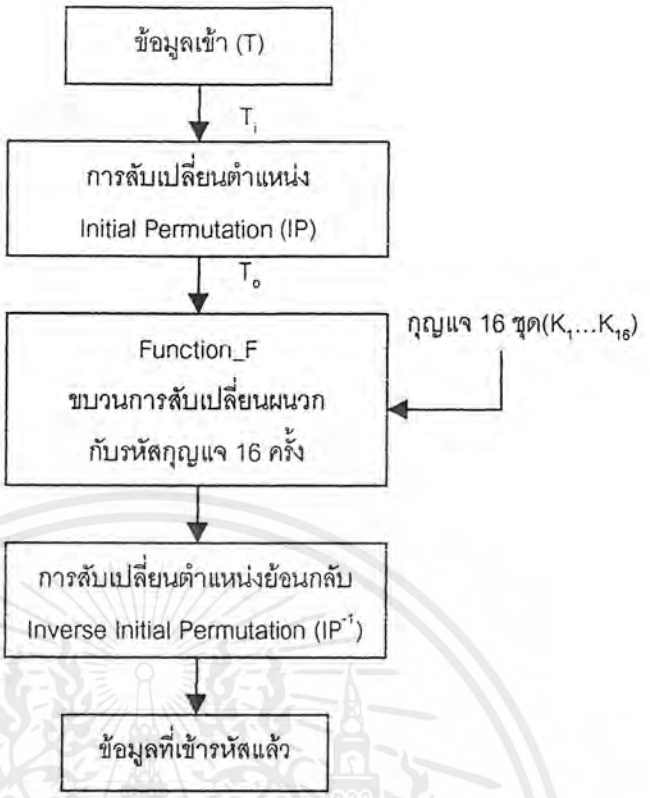
การใช้ทฤษฎี DES ในการเข้าและถอดรหัส ซึ่งจะกล่าวในหัวข้อต่อไป

ทฤษฎีของ DES

DES (Data Encryption Standard) เป็นวิธีการที่ใช้วิธีการทางคณิตศาสตร์เข้ามาสับเปลี่ยนข้อมูล โดยข้อมูลจะผ่านขบวนการการเข้ารหัสจะถูกสับเปลี่ยนเป็นข้อมูลที่ ไม่เป็นรูปแบบเรียกว่า Cipher การถอดรหัสจะนำข้อมูลที่ไม่มีรูปแบบนี้มาเข้าขบวนการถอดรหัส เปลี่ยนเป็นข้อมูลเดิม วิธีทางคณิตศาสตร์ที่จะกล่าวถึงเป็นการเข้า และถอดรหัสโดยใช้เลขฐานสอง กุญแจประกอบด้วยเลขฐานสอง 64 บิตซึ่งใช้ในการเข้าและถอดรหัส 56 บิต อีก 8 บิตเป็นบิตที่ใช้ในการตรวจสอบข้อผิดพลาด (Error Detection)

ขบวนการเข้ารหัส และถอดรหัส

ขบวนการเข้า และถอดรหัสแสดงได้ดังรูป ข้อมูลเข้าเป็น Block T_i เข้าสู่ขบวนการสับเปลี่ยนตำแหน่ง IP และข้อมูลออก $T_0 = IP(T)$ ข้อมูล T_0 เข้าสู่ขบวนการสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ครั้ง ซึ่งเรียกว่า Function_F เมื่อเสร็จขบวนการจะเข้าสู่การสับเปลี่ยนตำแหน่งย้อนกลับ IP^{-1} ผลลัพธ์ที่ได้เป็นข้อมูลที่เข้ารหัสเรียบร้อยแล้ว



รูปที่ 3.2 บล็อกไดอะแกรมของการเข้ารหัสดีอีเอส

การสับเปลี่ยนตำแหน่ง

ขบวนการ DES ใช้หลักการสับเปลี่ยนตำแหน่งหลายขั้นตอนทั้งในด้านการเข้ารหัส ถอดรหัส และการคำนวณหมายเลขกุญแจ โดยการสับเปลี่ยนจะใช้ตารางที่สร้างขึ้นดังรูป

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

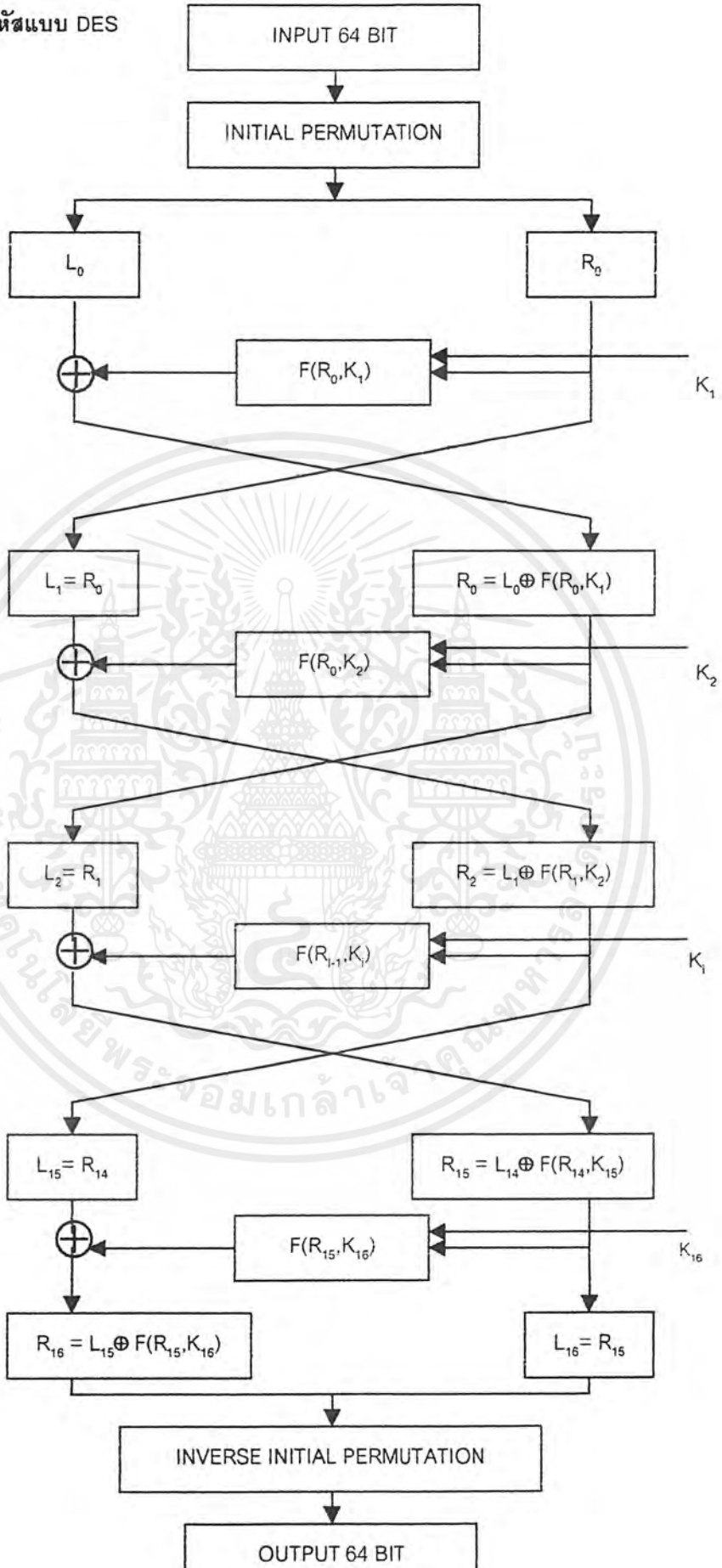
ตารางที่ 1 การสับเปลี่ยนตำแหน่ง IP

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

ตารางที่ 2 การสับเปลี่ยนย้อนกลับ IP^{-1}

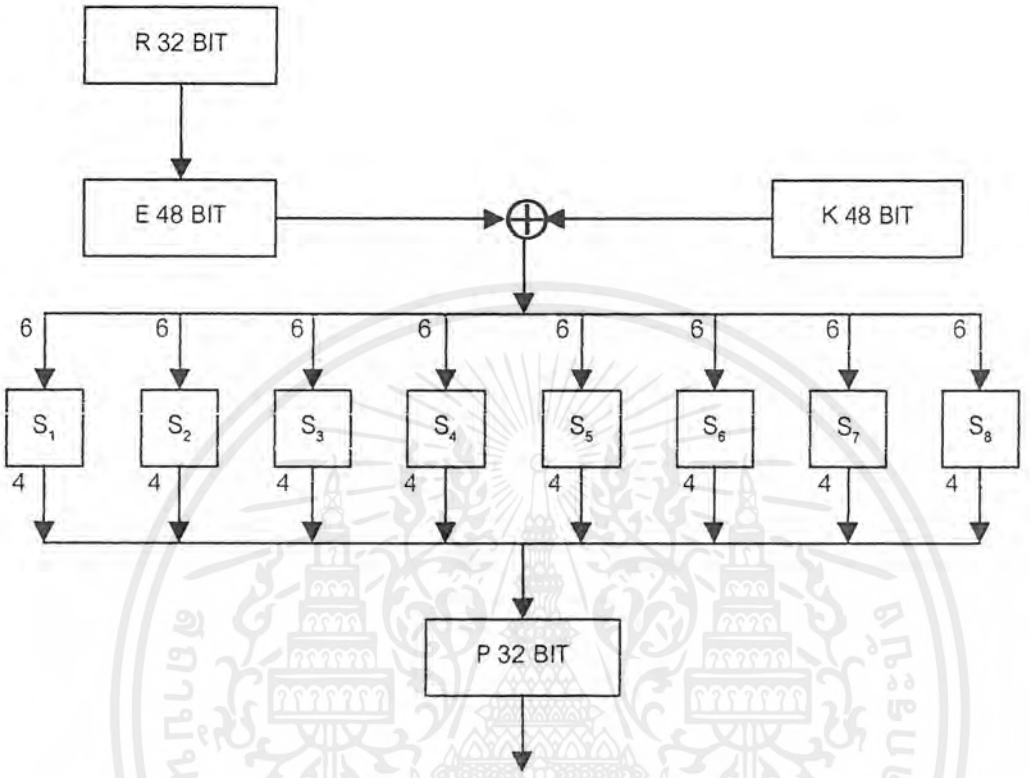
ในการใช้ตารางนั้น ตำแหน่งของตารางแทนตำแหน่งที่บิตนั้นๆ อยู่ตอนสับเปลี่ยนใหม่ เลขข้างในคือตำแหน่งที่บิตนั้นก่อนสับเปลี่ยน การ ใช้ตารางไล่จากซ้ายไปขวา แล้วลงมาอีกบรรทัด ไล่ลงไปเรื่อยๆ เช่น จากตาราง IP จะได้บิตที่สับเปลี่ยน คือ บิตที่ 58 จะมาอยู่บิตที่ 1 หลังการสับเปลี่ยน แล้วไล่ไปที่ บิตที่ 50 จะมาแทนที่บิตที่ 2 แล้วในตาราง IP-1 นั้น บิตที่ 40 จะมาอยู่ตำแหน่งที่ 1 แล้ว บิตที่ 8 จะมาอยู่ที่บิตที่ 2 สังเกตว่าจะได้ค่าเท่าเดิม เมื่อทำย้อนกลับแล้ว

รูปที่ 3.3 การเข้ารหัสแบบ DES



จากรูปข้างต้น แสดงถึง ขบวนการสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ครั้งแต่ละครั้งเรียกว่า Function_F อยู่ระหว่างการสับเปลี่ยนตำแหน่ง IP กับ IP^{-1}

Function_F **และ** S_box



รูปที่ 3.4 **การทำงานของ** Function_F

จากรูปคือการทำงานของ Function_Fโดยการทำงานเริ่มจาก Ri_{-1} ถูกสับเปลี่ยนเป็น 48 บิต โดยใช้ตาราง E ซึ่งเป็นลักษณะเดียวกับที่ใช้การทำสับเปลี่ยนตำแหน่ง IP และสับเปลี่ยนกลับ IP^{-1} มีส่วนไม่เหมือนคือ R_{i-1} ถูกใช้มากกว่า 1 ครั้ง ต่อมา แล้วนำมา XOR กับ K ผลลัพธ์ที่ได้ถูกแยกออกเป็น 8 ส่วนๆ ละ 6 บิต ส่งไปให้ S_box แล้วข้อมูลที่ออกมาจะได้ 4 บิต แล้วมารวมกันได้ เป็น 32 บิตแล้วนำไป สับเปลี่ยนกันอีกใน P โดยใช้ตาราง P ข้อมูลที่ได้จะนำไปใช้ต่อไป

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

ตารางที่ 3 P

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

ตารางที่ 4 E

วิธีการเข้า S_box อธิบายได้ดังนี้

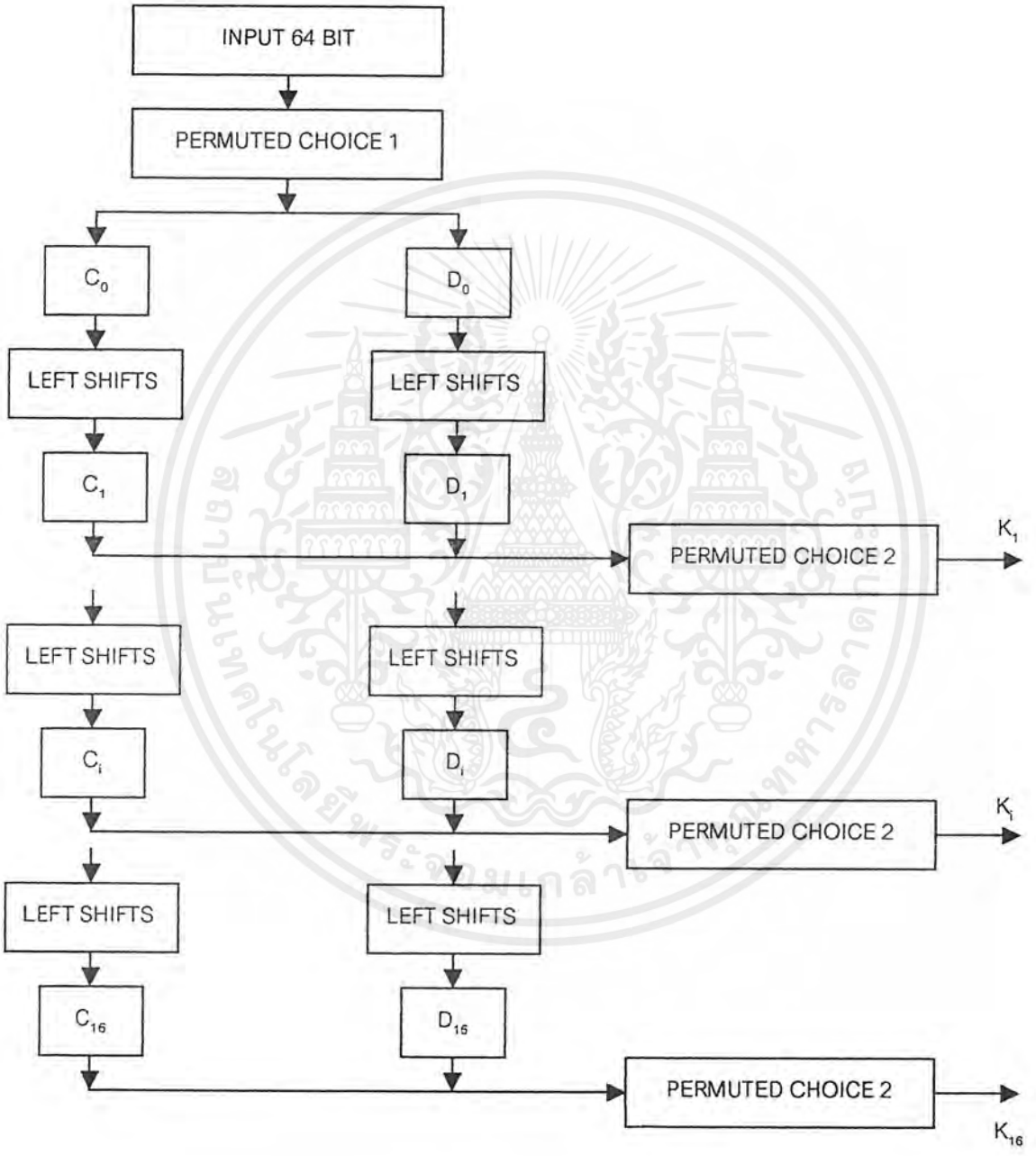
ให้ B_j เป็นข้อมูลที่เข้ามาที่ S_j หลังจากนำ E มา XOR กับ K แล้ว โดยมีขนาด 6 บิต ($b_1b_2b_3b_4b_5b_6$) ให้ j เป็นเลขจำนวนเต็มมีค่า 1 ถึง 8 เราจะแบ่ง B_j ออกเป็น 2 ส่วน เรียกว่า Row และ Column ในส่วนของ Row = b_1b_6 และส่วนของ Column = $b_2b_3b_4b_5$ นำค่า Row และ Column มาเทียบในตาราง S จะได้ผลลัพธ์ 4 บิต

	Column																
Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Box
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	5	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	1	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

ตารางที่ 5 S

การคำนวณหมายเลขกุญแจ (Subkey)

ในแต่ละรอบของการสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ชุด จะใช้หมายเลขกุญแจแต่ละชุดมีขนาด 48 บิต หมายเลขกุญแจ 16 ชุดได้มาจาก Key โดย Key เป็นข้อมูลขนาด 64 บิต 8 บิตในตำแหน่งบิตที่ 8, 16, ...,64 ใช้เป็นบิตตรวจสอบความผิดพลาด การทำขบวนการสับเปลี่ยน Permuted PC_1 หรือ Permuted choice 1 จะตัดในส่วนของบิตตรวจสอบความผิดพลาดออก ใช้ 56 บิตที่เหลือเท่านั้น



รูปที่ 3.5 การคำนวณหมายเลขกุญแจ

ผลลัพธ์จาก PC_1 จะถูกแบ่งออกเป็น 2 ส่วนส่วนละ 28 บิต เรียกว่า C และ D ใช้หาค่าหมายเลขกุญแจ K_i โดยกำหนดให้ Ci และ Di ซึ่งได้มาจาก C ,D ใช้หาค่า Ki มีสมการเป็น

$$C_i = LS_i(C_{i-1})$$

$$D_i = LS_i(D_{i-1})$$

LS_i คือ การทำการหมุนซ้ายเท่ากับจำนวนครั้งที่กำหนดในตาราง LS โดย C_0 และ D_0 เป็นค่าเริ่มต้น และสมการหมายเลขกุญแจ K_i คือ

$$K_i = PC\_2(C_iD_i)$$

ส่วนการสับเปลี่ยนตำแหน่ง PC_2 อยู่ในตาราง PC_2

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

ตารางที่ 6 สับเปลี่ยน PC_1

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

ตารางที่ 7 สับเปลี่ยน PC_2

ลำดับ	จำนวนครั้ง
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2

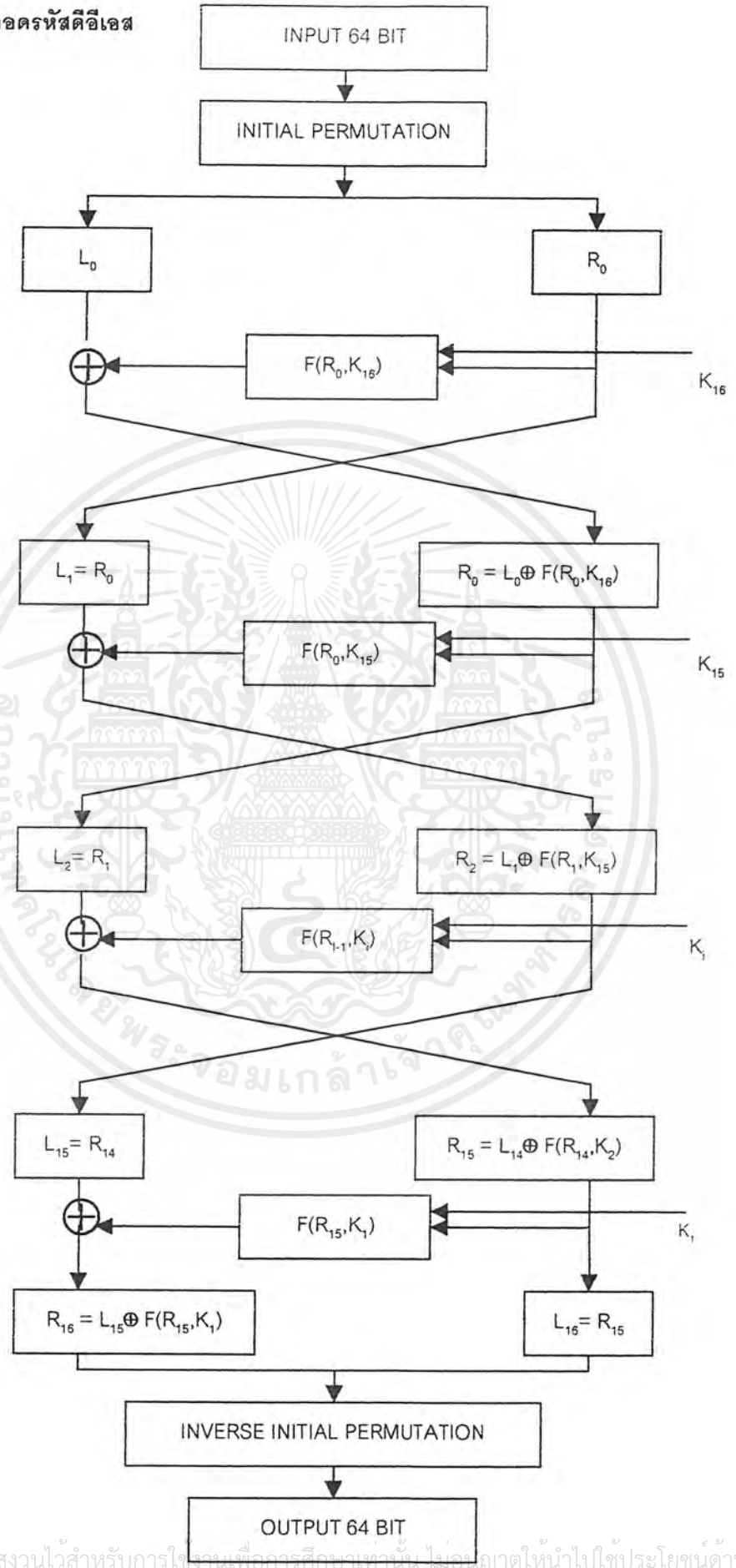
ลำดับ	จำนวนครั้ง
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

ตารางที่ 8 แสดงจำนวนครั้งในการหมุนซ้าย (LS)

การถอดรหัส

การทำการถอดรหัส ทำโดยใช้วิธีการเดิมแต่หมายเลขกุญแจจะใช้ K_{16}, K_{15}, ..., K_1 โดยจะย้อนกลับแค่กุญแจที่ใช้ ส่วนวิธีการอื่นทำเหมือนกับการเข้ารหัส ดังรูป

รูปที่ 3.6 การถอดรหัสดีอีเอส



บทที่ 4
ภาษาวิเฮชดีแอล(VHDL)

วีเฮชดีแอล (VHDL) ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC ย่อมาจาก Very High Speed Integrated Circuit) เป็นภาษาคอมพิวเตอร์ระดับสูง (High Level Language) ซึ่งใช้อธิบายการทำงานของระบบดิจิตอลฮาร์แวร์ สามารถอธิบายการทำงานของฟังก์ชันได้หลายระดับ ตั้งแต่ระดับบล็อก จนถึงระดับเกท ความซับซ้อนของระบบสามารถจะเขียนได้ตั้งแต่ระดับเกท ประกอบกันจนเป็นระบบที่สมบูรณ์ รูปแบบของภาษาวีเฮชดีแอลนั้น จะประกอบไปด้วยสองส่วนใหญ่ๆ ได้แก่ ส่วนของภาษาซีเควนเชียล(Sequential Language) และภาษาคอนเคอร์เรนท์ (Concurrent Language) การโปรแกรมด้วยภาษาวีเฮชดีแอลสามารถเขียนได้ทั้งสองรูปแบบรวมกัน เพราะในการทำงานของระบบใดๆ ย่อมมีการทำงานในระบบ ซีเควนเชียล และคอนเคอร์เรนท์ อยู่รวมกัน นอกจากนี้ภาษาวีเฮชดีแอลยังสามารถอธิบายการเชื่อมต่อระหว่างระบบย่อยๆ เข้าด้วยกันเพื่อนเป็นระบบใหญ่ได้ ตัวภาษาวีเฮชดีแอลนอกจากจะกำหนดในรูปของไวยากรณ์ของตัวภาษาแล้ว ยังมีการตรวจสอบว่าจะซิมูเลชั่นได้หรือไม่ เพราะภาษาวีเฮชดีแอลต้องผ่านการซิมมูเลชั่นเพื่อตรวจสอบการทำงาน ฉะนั้นการคอมไพล์ จะมีการตรวจสอบทั้งไวยากรณ์และซิมูเลชั่นซีแมนติค แย่างไรก็ตามแม้ตัวภาษาจะมความซับซ้อนในรูปแบบและกฎเกณฑ์ของภาษา แต่การเรียนรู้เพียงบางส่วนของภาษาก็สามารถนำไปใช้โดยไม่จำเป็นต้องศึกษารายละเอียดทั้งหมด เนื่องจากตัวภาษาวีเฮชดีแอลออกแบบมาเพื่อให้ใช้สำหรับการออกแบบตั้งแต่วงจรที่มีขนาดเล็กจนถึงขนาดใหญ่ และซับซ้อน

4.1 ความสามารถของวีเฮชดีแอล

1 ตัวภาษาวีเฮชดีแอลสามารถใช้เป็นสื่อกลางของการในการแลกเปลี่ยนระหว่างผู้ผลิตชิปกับผู้ออกแบบ

2 ใช้เป็นตัวสื่อกลางในการแลกเปลี่ยนสื่อสารระหว่างซีเออี และซีเอดีทูล เช่นตัวภาษาซ้อสโค้ดของวีเฮชดีแอล สามารถคอมไพล์โดยใช้คอมไพล์เลอร์ และซิมูเลเตอร์หลายๆ ตัวต่างกันได้

3 ภาษาวีเฮชดีแอลสนับสนุนการออกแบบแบบท็อปดาวน์ และการออกแบบแบบบัททอมอัป หรือผสมกันทั้งสองแบบ

4 ตัวภาษาวีเฮชดีแอลเป็นแบบทั่วไป คือไม่อิงเทคโนโลยีอันใดอันหนึ่งสามารถอิงเทคโนโลยีอะไรก็ได้ และในขณะเดียวกันสามารถสนับสนุนหลายๆ เทคโนโลยี

5 สามารถสนับสนุนการออกแบบทั้งระบบซิงโครนัส และอะซิงโครนัส

6 สนับสนุนการออกแบบระบบดิจิตอล ในหลายๆ เทคนิค เช่นไฟไนท์สเตทแมชชีน ,อัลกอลิทะมิก หรือสมการบูลีน

7 ภาษาวีเฮชดีแอลสามารถอ่านและทำความเข้าใจได้โดยมนุษย์

8 ภาษาวีเฮชดีแอลเป็นมาตราฐานรับรองโดย IEEE และ ANSI ทำให้มีโมเดลที่ออกแบบโดยภาษาวีเฮชดีแอล สามารถเคลื่อนย้ายไปยังระบบใดๆ ก็ได้และสามารถนำกลับมาใช้ใหม่ได้

9 ภาษาวีเฮชดีแอลสนับสนุนรูปแบบการเขียนถึงสามรูปแบบ ได้แก้ แบบบีเฮฟเวอร์ ,แบบสทรักเตอร์ , แบบดาต้าโฟลว์ หรือสามารถเขียนรวมกันได้ถึงสามรูปแบบ

10 สนับสนุนการออกแบบขนาดใหญ่โดยใช้ความสามารถของ ส่วนประกอบ ,ฟังก์ชันโปรซีเดอร์ และแพคเกต

11 ไม่จำเป็นต้องศึกษาซอฟท์แวร์ซิมูเลเตอร์เพราะซิมูเลชันโมเดลสามารถเขียนได้โดยใช้ภาษาวิเฮชดีแอลด้วยเช่นกัน

12 สามารถเขียนโมเดลได้ในขนาดไม่จำกัด ไม่มีข้อจำกัดเรื่องขนาดของโมเดล อาจจะขึ้นอยู่กับซอฟต์แวร์

13 สามารถอธิบายตัวแปรที่เกี่ยวข้องกับฟังก์ชันด้านเวลา เช่น Propagation Delay ,Min-Max Delay ,Setup ,Holding Time ,Spike Detection สามารถอธิบายได้ในตัวภาษา

14 เจเนริกซ์ ช่วยสามารถสร้างตัวแปรของรูปแบบ

15 โมเดลที่สร้างด้วยวีเอชดีแอลนั้น ไม่เพียงแต่จะอธิบายฟังก์ชันการทำงานเท่านั้นแต่ยังสามารถอธิบายถึงรายละเอียดของโมเดล เช่น Total Area และ Speed ของโมเดล

16 โมเดลที่สร้างขึ้นสามารถจำลองการทำงานได้ เพราะว่าตัวแปลภาษาได้ตรวจสอบไวยากรณ์ทางด้านซิมูเลชันซิแมนติคไว้ด้วย

17 การอธิบายโมเดลด้วยแบบบีเฮฟเวอร์สามารถซินเทซิส ไปเป็นระดับเกท-เลเวล ได้ ถ้าทำตามกฎของ ซินเทซิส ไกด์ไลน์

18 มีความสามารถที่ให้เราออกแบบข้อมูลชนิดใหม่ๆ ได้ทำให้วีเฮชดีแอลโมเดล เป็นการออกแบบในระดับสูง ที่ไม่ต้องคำนึงถึงว่าจะสร้างตัวโมเดลนั้นขึ้นมาได้อย่างไร

4.2 รูปแบบพื้นฐานของภาษา

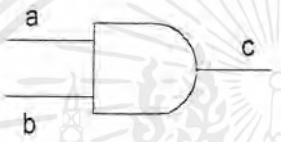
ระหว่างการออกแบบนักออกแบบจะพยายามย่อยวงจรออกเป็นส่วนย่อยๆ เพื่อที่จะจัดการได้ง่ายขึ้น ภาษาวีเฮชดีแอลสนับสนุนการแยกส่วนฮาร์ดแวร์ และทำให้ง่ายที่จะเขียนพฤติกรรมการทำงานของแต่ละส่วนย่อยๆ นั้นบางทีส่วนย่อยๆ สามารถใช้ได้ในหลายๆ ส่วนของวงจรที่เรอจะออกแบบ หรือแม้กระทั้งใช้ในรูปแบบอื่นๆ รูปแบบพื้นฐานของภาษาวีเฮชดีแอล ในการสร้างฮาร์ดแวร์โมเดล ก็คือ Design Entity ซึ่งสามารถแทน เซล ,ชิป ,บอร์ด หรือ ระบบย่อยได้

Design Entity ประกอบด้วย 2 ส่วนใหญ่คือ ส่วนของ Entity Declaration และส่วนของ Architecture Body

Entity Declaration และ Architecture Body เป็นสองส่วนหลักที่สำคัญในภาษา VHDL Language Library คือส่วนของ Hardware Description หรือโมเดลซึ่งสามารถจะอยู่ใน Design File ใดๆ หรือถูกคอมไพล์อยู่ใน Design File หลายๆ File ที่แยกจากกัน ความสามารถนี้ทำให้นักออกแบบสามารถจัดวางวงจรให้เป็น Module เขียนอธิบายแต่ละ Module จากนั้นก็คอมไพล์แต่ละ Entity แยกจากกัน โดยมี Architecture Body ของแต่ละ Entity ที่แตกต่างกันออกไป การประกาศ Entity Declaration เป็นการกำหนดการเชื่อมต่อระหว่าง Design Entity กับวงจรส่วนอื่นๆ ภายนอก โครงสร้างของ Entity Declaration แสดงตัวอย่างต่อไปนี้

```
Entity identifier  Is
        Entity_header
        --(generic and/or port clause)
        Entity_declarative_part
        --(Declaration for subprograms,type,signal,..)
begin
        Entity_Statement_part
End identifier;
```

Entity identifier คือชื่อของ Entity ที่เรากำหนดขึ้น แต่ละ design entity รับข้อมูลจากภายนอกโดย Port Mode In และส่งข้อมูลออกไปภายนอกโดย Port Mode Out ดังตัวอย่างในรูปที่ 4.1




```
ENTITY and2 IS
        PORT(a,b:IN bit;
             C:OUT bit);
END  and2;
```

รูปที่ 4.1 **แสดง** port in **และ** out **ของ** AND **เกท**

Architecture Body อธิบายความสัมพันธ์ระหว่างอินพุทและเอาท์พุทของ Design entity โครงสร้างของ Architecture แสดงได้ดังนี้

```
Architecture identifier of Entity_name is
        Architecture_declaretive_part
Begin
        Architecture_statement_part
End identifier;
```

Identifier และ Entity name คือ words ที่ต้องเขียนไว้ใน ภาษาวีเฮชดีแอลโค้ด สิ่งที่สำคัญที่จะต้องคำนึงถึงคือ ชื่อของ Entity name ใน Architecture body จะต้องสัมพันธ์กันดังรูปที่ 4.2

```
ENTITY and2 IS
        PORT(a,b:IN bit;
                C:OUT bit);
END and2

ARCHITECTURE example OF and2 IS
        --declarations here
BEGIN
        --statement here
END example;
```

รูปที่ 4.2 การใช้ชื่อ Entity ใน Entity Declaration และ Architecture body

4.3 การเขียนอธิบายในรูปแบบต่างๆ

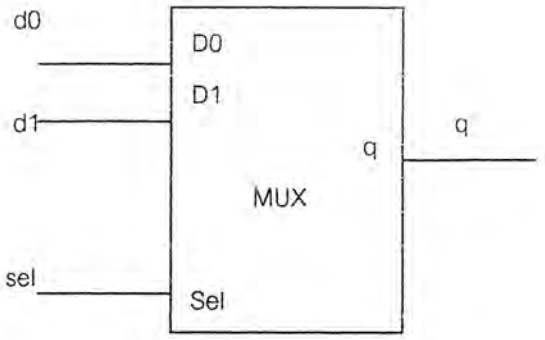
ภาษาวีเฮชดีแอล เป็นวิธีการอธิบายการทำงานของฮาร์ดแวร์ ในลักษณะของ Texual Format แทนที่จะใช้ schematic design เหมือนเมื่อก่อน หัวห้องต่างๆ ดังนี้คือวิธีการเขียนอธิบาย ภาษาวีเฮชดีแอลในหลายๆวิธีเพื่ออธิบาย Hardware Architecture

- Structural Description Method ใช้อธิบายตัวรูปแบบ ในรูปแบบของการเชื่อมต่อ component ต่างๆ เข้าด้วยกัน
- วิธีการบรรยายบีเฮฟเวอร์อธิบายฟังก์ชันการทำงานของรูปแบบ Hardware ในรูปแบบของ Circuit , Signal ที่ตอบสนองกับสัญญาณที่รับเข้ามาจากภายนอก พฤติกรรมการทำงาน (Hardware Behavior) จะถูกอธิบายด้วย Algorithm โดยไม่ว่าจะสร้างขึ้นมาอย่างไร
- Data Flow Description Method มีความคล้ายคลึงกับ Register-transfer Language วิธีการนี้อธิบายฟังก์ชันการทำงานของรูปแบบ โดยการกำหนดการไหล (Flow) ของข้อมูลจากอินพุท หรือรีจิสเตอร์ ไปยังตัวเอาท์พุท หรือรีจิสเตอร์ อีกตัววิธีการทั้ง 3แบบที่ใช้อธิบาย Architecture ของ Hardware สามารถอธิบายร่วมกันโดยใช้ทั้ง 3 แบบ ต่อ 1 รูปก็ได้

4.3.1 Structural Description

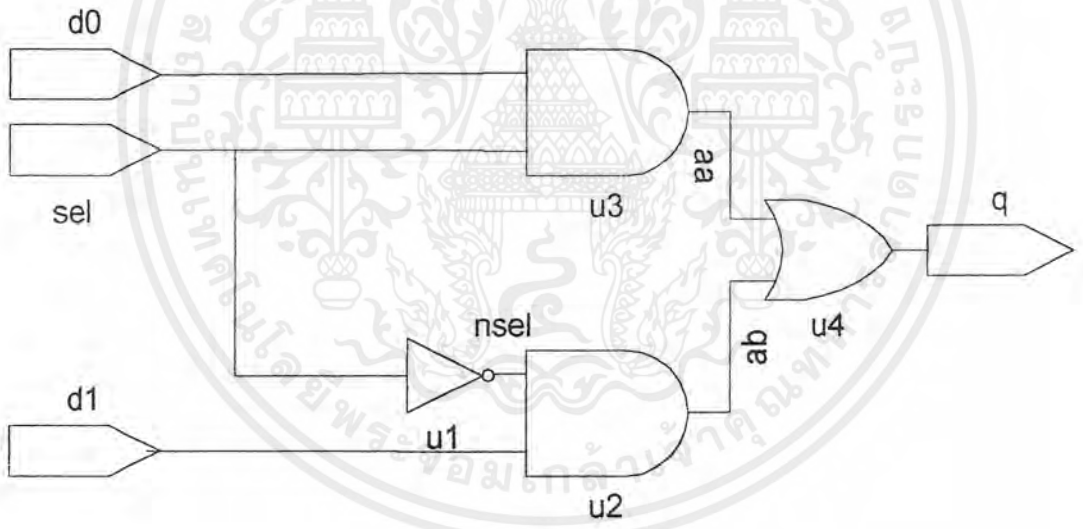
ในส่วนนี้จะอธิบายในส่วนของภาษาเพียงบางส่วนในการที่จะแสดง VHDL Structural Description โดยใช้ตัวอย่างคือ มัลติเพลกเซอร์ 2 อินพุท จะไม่ได้อธิบายโครงสร้างภายในส่วนนี้ทั้งหมด เพียงต้องการยกตัวอย่างให้เห็นเท่านั้น รายละเอียดให้ดูได้ที่ VHDL Reference Manual

VHDL Structural Description เป็นวิธีการอธิบายตัวรูปแบบฮาร์ดแวร์ ที่คล้ายกับแสดงโดยใช้ Schematic Diagram เพราะว่าแสดงให้เห็นถึงการเชื่อมต่อของส่วนประกอย ดังตัวอย่างที่จะแสดงให้เห็นในส่วนต่อไปนี้ เป็นการเปรียบเทียบง่ายๆ 1 วงจร ซึ่งแทนด้วย VHDL และแทนด้วย Schematic Diagram ว่าเป็นอย่างไร



รูปที่ 4.3 สัญลักษณ์แสดงสองอินพุทมัลติเพลกเซอร์

จากรูปที่ 4.4 แสดงสัญลักษณ์ของมัลติเพลกเซอร์ 2 อินพุท วงจร Mux นี้เป็นการออกแบบในลักษณะของ Hierarchical Design ซึ่งวงจรในระดับล่างสุดเป็น Schematic Diagram หรืออยู่ในรูปแบบของการอธิบายถึงการเชื่อมต่อภายใจ ดังรูปที่ 4.5



รูปที่ 4.4 แสดงระดับเกทของสองอินพุทมัลติเพลกเซอร์

รูปที่ 4.6 แสดง VHDL Structural Description ของมัลติเพลกเซอร์ 2 อินพุท โดย VHDL Code สามารถเขียนส่วนประกอบ โดยการเขียน Double dash (--) ข้อความหรืออักษรใดที่อยู่หลังเครื่องหมายจะถูกมองว่าเป็น comment โดย compiler จะไม่สนใจ comment นั้นจะทำให้ง่ายต่อการเข้าใจ

```
ENTITY mux IS                                   -- Entity Declaration
        PORT(d0,d1,sel:IN bit;q:OUT bit);       -- Port Clause
END mux;
ARCHITECTURE struct OF mux IS                   -- Architecture Body
COMPONENT and2
        PORT(a,b:IN bit;c:OUT bit);
END COMPONENT;
COMPONENT or2
        PORT(a,b:IN,bit;c:OUT bit);
END COMPONENT;
COMPONENT inv
        PORT(a:IN bit;c:OUT bit);
END COMPONENT;

SIGNAL aa,ab,nsel:bit;                          -- Signal Declaration
FOR u1:inv USE ENTITY WORK.invrt(behav);        -- config
FOR u2,u3:and2 USE ENITITY WORK.and_gt(dFlow);  -- Specif
FOR u4:or2 USE ENTITY WORK.or_gt(archl);
BEGIN
        u1:inv PORT MAP(sel,nsel);              -- Architecture Statement part
        u2:and2 PORT MAP(nsel,d1,ab);
        u3:and2 PORT MAP(d0,sel,aa);
        u4:or2 PORT MAP(aa,bb,q);
END struct;
```

รูปที่ 4.5 ตัวอย่างโปรแกรมของ Structural Description สำหรับมัลติเพลกเซอร์

มัลติเพลกเซอร์ 2 อินพุท แสดงในรูปที่ 4.5 เป็นวงจรพื้นฐาน Entity Declaration อยู่ที่ด้านบน บรรทัด 1 ถึง 3 กำหนดการ interface design entity และสภาพแวดล้อม หรือวงจรอื่นภายนอก

Entiiy Declaration มี port Clause ซึ่งบอกช่องสัญญาณอินพุทและเอาท์พุท สัญญาณกำหนดเป็น bit มีค่าได้แค่ 0 หรือ 1

Architecture Body ในรูปที่ 4.6 อธิบายถึงความสัมพันธ์ระหว่าง Design Entity Input และ Output ในลักษณะโครงสร้าง Architecture นี้สามารถทำงานได้เหมือนกับ Schematic Component หลายๆตัว (AND2 ,OR2 ,INV) ที่ประกอบรวมกันจนเป็น MUX Design Entity ในรูปที่ 4.5 ถูกประกาศไว้ในส่วนของ Architecture Declaration Signal(aa,ab,nsel) ถูกประกาศไว้ใน Architecture body ด้วยเช่นกัน เพื่อที่จะแทนเอาท์พุทของ AND เกท 2 ตัวและ inverter

Configuration Specification เชื่อมเอาส่วนประกอบแต่ละตัวที่ประกาศไว้เข้ามายัง Design Entity เพื่อบอก Design Entity ว่าส่วนประกอบแต่ละตัวทำงานอย่างไร เช่น u1 ถูกกระโดดมายัง Architecture Body ที่ชื่อ behav สำหรับ Design Entity ที่เรียกว่า Invert

Architecture Statement Part อธิบายการเชื่อมต่อระหว่างส่วนประกอบที่ประกอบด้วย Design Unit ในส่วนนี้จะมีการประกาศใช้ Component

4.3.2 Behavioral Description

ในส่วนนี้จะเป็นการอธิบายถึงส่วนสำคัญของภาษาส่วนหนึ่ง นั่นคือ Behavioral Description โดยใช้วงจรซึ่งได้ยกตัวอย่างมาแล้วก่อนหน้านี้คือ MUX หลังจากได้อธิบายส่วนของ Structural Description เราสามารถเปรียบเทียบกันได้ระหว่างการอธิบายด้วย Behavioral Description กับ Structural Method ว่าแตกต่างกันอย่างไร

VHDL Behavioral Description แทนฟังก์ชันการทำงานของรูปแบบ ในรูปแบบของวงจร และสัญญาณที่ตอบสนองการกระตุ้นการจากภายนอก แสดงในรูปที่ 4.3 ถึง 4.6 พฤติกรรมการทำงานของ MUX ถูกกำหนดด้วยการเชื่อมต่อระหว่าง Inverter ,And gate ,Or gate ซึ่งฟังก์ชันของแต่ละเกทนี้เป็นที่เข้าใจกันดีอยู่แล้ว ในรูปแบบที่มีความซับซ้อนมากขึ้นส่วนประกอบ u1 ถึง u5 ในรูปที่ 4.5 ต้องสามารถแทนด้วย Entity ที่มีฟังก์ชันการทำงานแต่ละส่วนประกอบ ด้วย Behavioral Description

Behavioral Description แทนที่จะเป็นแบบ Structural Description เหมือนหัวข้อก่อนหน้าที่ ในครั้งนี้เราสามารถวาง MUX Symbol ลงใน Schematic แต่ในที่นี้เราใช้ Behavioral ของส่วนประกอบระหว่างการทำ circuit ซิมูเลชั่น รูปที่ 4.6 แสดงภาษาวิเฮชดีแอลโค้ด ซึ่งกำหนดการทำงานของ MUX ในรูปแบบของ Behavioral

Behavioral Description ในรูป 4.6 แบะ Structural Description ในรูปที่ 4.5 ทั้งคู่มี Entity Description และ Architecture body เช่นกัน ในทางปฏิบัติเราไม่จำเป็นจะต้องเขียน 2 Architecture body ไว้ในไฟล์เดียวกัน เราควรเขียน Entity Declaration ไว้ที่ ไฟล์หนึ่ง แล้วเขียน Behavioral Description ไว้อีกไฟล์หนึ่ง และ Structural Description ไว้อีกไฟล์เช่นกัน ในการออกแบบจริงๆ หลังจากการเขียน และ คอมไพล์ Entity Description เสร็จเรียบร้อยแล้ว ขั้นต่อไปควรเขียน Behavioral Architecture เป็นขั้นตอนต่อไปในการที่จะทดลองการทำงานวงจรทั้งหมดโดยรวมทั้งหมด หลังจากนั้นเราจึงทำการเขียน simulate และ refine ฟังก์ชันการทำงานของโมเดลจนกว่าจะทำงานได้ถูกต้อง จึงจะทำการเขียน Structural Architectureจากนั้นก็เปลี่ยน Structural Description เข้าไปแทนที่ Behavioral Description เพื่อที่จะทำการ Simulate ต่อไป

```
ENTITY mux IS
        PORT(d0,d1,sel:IN bit;q:OUT bit);
END mux;
ARCHITECTURE behav OF mux IS
BEGIN
        f1:
        PROCESS(d0,d1,sel);
BEGIN
IF sel='0' THEN
Q<=d1;
ELSE
Q<=d0;
END IF;
END PROCESS f1;
END behav;
```

รูปที่ 4.6 ตัวอย่างโปรแกรมของ Behavioral Description **สำหรับมัลติเพลกเซอร์**

Behavioral Description Mode มีประโยชน์ต่อการกำเนิดสัญญาณอินพุทขึ้นมาทดสอบ VHDL โมเดลในส่วนอื่นๆ เมื่อต้องการทซิมูเลชั่น เช่น เราต้องการออกแบบ Traffic Light Controller โดยใช้ Structural Description และเราต้องการจะทดสอบโมเดลนั้นโดยที่ Traffic Light Controller นั้น อินพุทจะต้องรับจาก Sensor ที่ต่อเข้ามา สำหรับการซิมูเลชัน เราต้องเขียน Behavior Description โมเดลขึ้นมาเพื่อทำการจำลองสัญญาณที่ออกจาก Sensor

ข้อแตกต่างที่เห็นได้ชัดระหว่าง Structural และ Behavioral Description ของ MUX คือ Architecture body ดังรูปที่ 4.6 มี process statement ซึ่งแทนการทำงานที่เป็นอิสระกำหนดพฤติกรรมการทำงานของฮาร์ดแวร์ หรือส่วนใดส่วนหนึ่งของรูปแบบ รูปแบบของการเขียน Process Statement แสดงดังต่อไปนี้

```
Process Statement.....label;
        Process(sensitivity_list)
                Process_declaration_part
        Begin
                Process_Statement_part
        End Process label;
```

Process Statement ในรูปที่ 4.6 Process label f1 ตามด้วย ':' Process label เป็น optonal แต่มีประโยชน์ในการที่ช่วยเห็นความแตกต่าง process หลายๆ ตัวให้รูปแบบใหญ่ๆ

ในวงเล็บที่ต่อจาก process คือ Option sensitivity List ดังรูปที่ 4.6 ประกอบไปด้วยสัญญาณ d0 ,d1 ,sel ระหว่างการทำซิมูเลชั่น ถ้าสัญญาณใดสัญญาณหนึ่งใน sensitivity List เกิดการเปลี่ยนแปลง State, process จะทำการ Execute และ state ของ เอาท์พุท ก็จะเป็นไปตามเช่นกัน แต่ละ process ในการออกแบบวิเฮซดีแอล จะ Execute หนึ่งครั้งระหว่างการทำ Initialize VHDL Hardware โมเดล

หัวใจสำคัญของ Process Statement ในรูปที่ 4.6 คือ IF Statement ที่อยู่ภายใน Process Statement Part รูปแบบพื้นฐานของการเขียน IF Statement แสดงดังข้างล่าง

```
if Statement.....if condition then
                        sequence_of_Statements
                else if condition then
                        sequence_of_Statements
                    else
                        sequence_of_Statements
                end if;
```

if statement ในภาษาวีเฮซดีแอล จะถูกตีความคล้ายๆ กับประโยคในภาษาอังกฤษ เมื่อใดก็ตามที่สภาวะภายใน if condition หรือ else condition ในตัวอย่าง ได้รับสภาพความตรงตามเงื่อนไข หรือสภาวะตรงข้าม target signal q จะถูกเปลี่ยนแปลง ตามความเหมาะสมซึ่งขึ้นอยู่กับ Signal Assignment Statement รูปแบบพื้นฐานของ Signal Assignment มีดังนี้

```
Signal Assignment Statement:target(=transport Waveform)
(Note transport, Optional)
```

Signal Assignment Statement ประโยคแรกในรูปที่ 4.6 คือ

```
q<=d1;
```

ประโยคนี้จะทำการกำหนดสัญญาณ d1 ให้แก่สัญญาณ q (Optional transport) ไม่ได้ใช้ในตัวอย่างนี้ Signal Assignment delimeter ประกอบด้วยตัวอักษรพิเศษ 2 ตัวคือ บางครั้งเรียกว่า compound delimeter() ประโยค Signal assignment Statement อันที่สองคือ

```
q<=d0;
```

จะทำการกำหนด waveform ให้กับสัญญาณ q การใช้งานอีกอย่างของ compound delimeter คือใช้เป็น relational operator ใยประโยค condition เช่น

```
If Z<='1' Then
```

4.3.3 Data Flow Description

VHDL Data Flow Description และ Register Transfer Language มีความคล้ายคลึงกับคือทั้งคู่อธิบายการทำงานขอรูปแบบ โดยการกำหนดการไหลของข้อมูลจากอินพุทหนึ่ง หรือรีจิสเตอร์หนึ่งไปยังเอาท์พุท

หรืออีก รีจีสเตอร์หนึ่ง Data Flow และ Behavioral Description มีความคล้ายคลึงกันคือ ทั้งคู่ใช้ Process เพื่อที่จะอธิบายฟังก์ชันการทำงานของวงจร Behavioral Description ใช้จำนวน Process น้อยกว่า โดยภายในแต่ละ Process ใช้ลักษณะของซิเควนเชียล Signal Assignment หลายๆ คำสั่งภายใน Process ในทางตรงกันข้าม Data Flow Description ใช้จำนวนของคอมเคอร์เรนท์ Signal Assignment มาก คอนเคอร์เรนท์ Statement ใช้ใน Data Flow Description ประกอบด้วย

- Block Statement
- คอนเคอร์เรนท์ Procedure Call
- คอนเคอร์เรนท์ Assertation Statement
- คอนเคอร์เรนท์ Signal Assignment Statement

นอกจากนี้ Process Statement ,Generate Statement และ Component Instantiation Statement เป็น คอนเคอร์เรนท์ statement ด้วยเหมือนกัน ซึ่งโครงสร้างเหล่านี้ไม่ค่อยพบในการอธิบาย Data Flow Description

คอนเคอร์เรนท์ statement กำหนดการเชื่อมต่อ Process Blocks ซึ่งทั้งหมดรวมๆ กันจะอธิบายการทำงานของ Design ตอนเคอร์เรนท์ Statement ทำการ Execute แบบ Asynchronous ร่วมไปกัน คอนเคอร์เรนท์ Statement อื่นๆ

รูปที่ 4.7 แสดงให้เห็นรูปแบบการเขียนอธิบาย MUX โดยวิธีการ Data Flow Description ซึ่งก่อนหน้านี้ใช้ Behavioral และ Structural Description เขียนอธิบายมาแล้ว ซึ่งตัวอย่างนี้ดูง่ายเกินไป ที่จะเห็นประโยชน์ของ Data Flow Description เพราะว่ามีความคล้ายคลึงกับ Behavioral Description ในรูปที่ 4.6 มาก โดยทั้งสองตัวอย่างใช้ Process Statement แสดงด้วย คอนเคอร์เรนท์ Statement ดังรูปที่ 4.7 เพื่อกำหนด Signal Behavior

```
ENTITY mux IS
        PORT(d0,d1,sel:IN bit;q:OUT bit);
END mux;
ARCHITECTURE data_flow OF mux IS
BEGIN
        cls:
        q<=d1 WHEN sel='0' ELSE                    --condition signal assignment
        d0;
END data_flow;
```

รูปที่ 4.7 ตัวอย่างโปรแกรมของ Data Flow Description ของมัลติเพลกเซอร์

Data Flow Description ประกอบไปด้วย Entity Declaration เหมือนกับสองแบบที่กล่าวมา ส่วน Architecture body ประกอบไปด้วย คอนเคอร์เรนท์ Signal Assignment Statement ซึ่งทำหน้าที่เทียบเท่ากับ process Statement รูปแบบของการเขียนคอนเคอร์เรนท์ Signal Assignment แสดงดังต่อไปนี้

Concurrent Signal Label: Conditional_signal_assignment

Assignment Statement.....

Label:seleced)signal_assignment

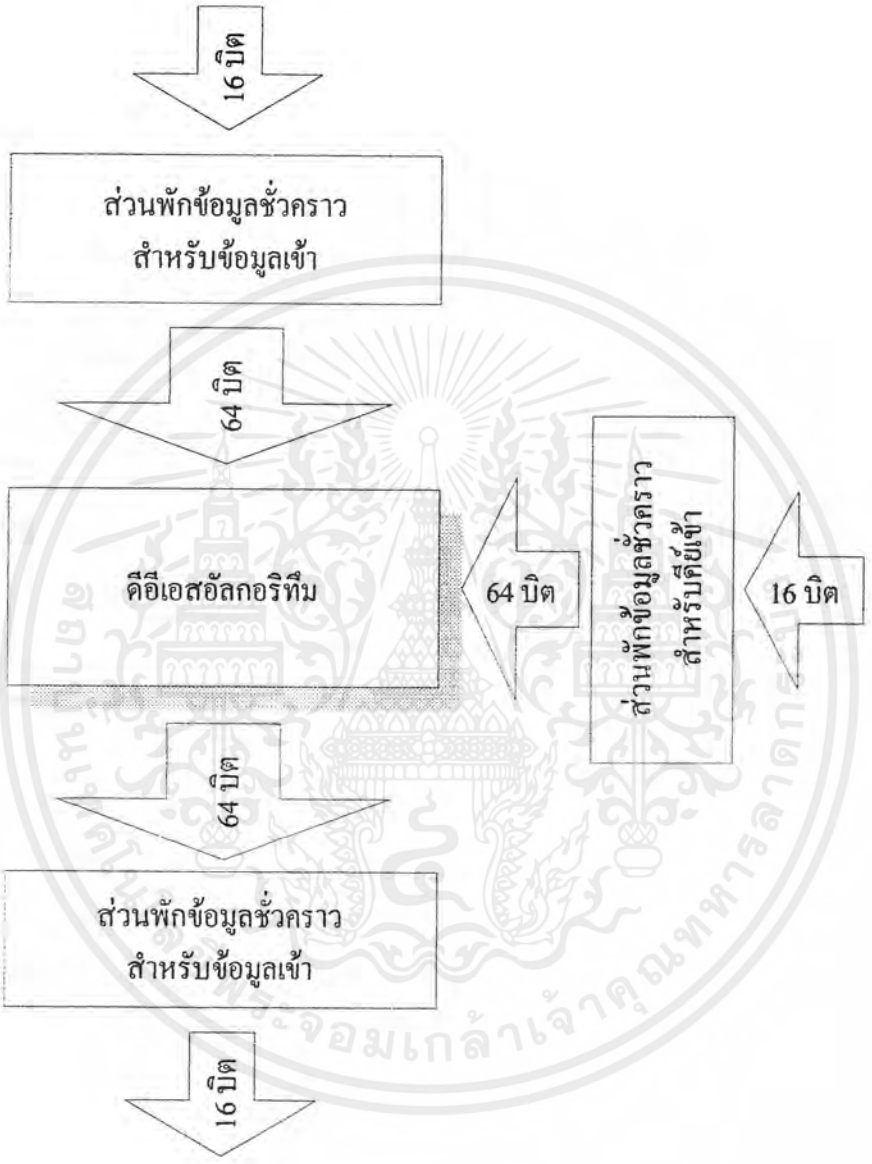
ในรูปที่ 4.7 Condition Signal Assignment ทำหน้าที่เป็น Signal Assignment (q<d1 หรือ q<d0) ขึ้นอยู่กับสถานะที่กำหนดใน condition Waveform รูปแบบของการทำ Conditional Waveform มีดังนี้

Conditional Signal Assignment	target<=options conditional_waveform;
Conditional Waveforms	waveform when condition else
	→;
	waveform when condition else
	waveform

condition signal Assignment แทนการทำงานของ Process Statement ที่ใช้ IF Statement ในการเปลี่ยนแปลงลักษณะของสัญญาณ (Optional Guarded Transport) ไม่ได้แสดงให้เห็นในตัวอย่าง

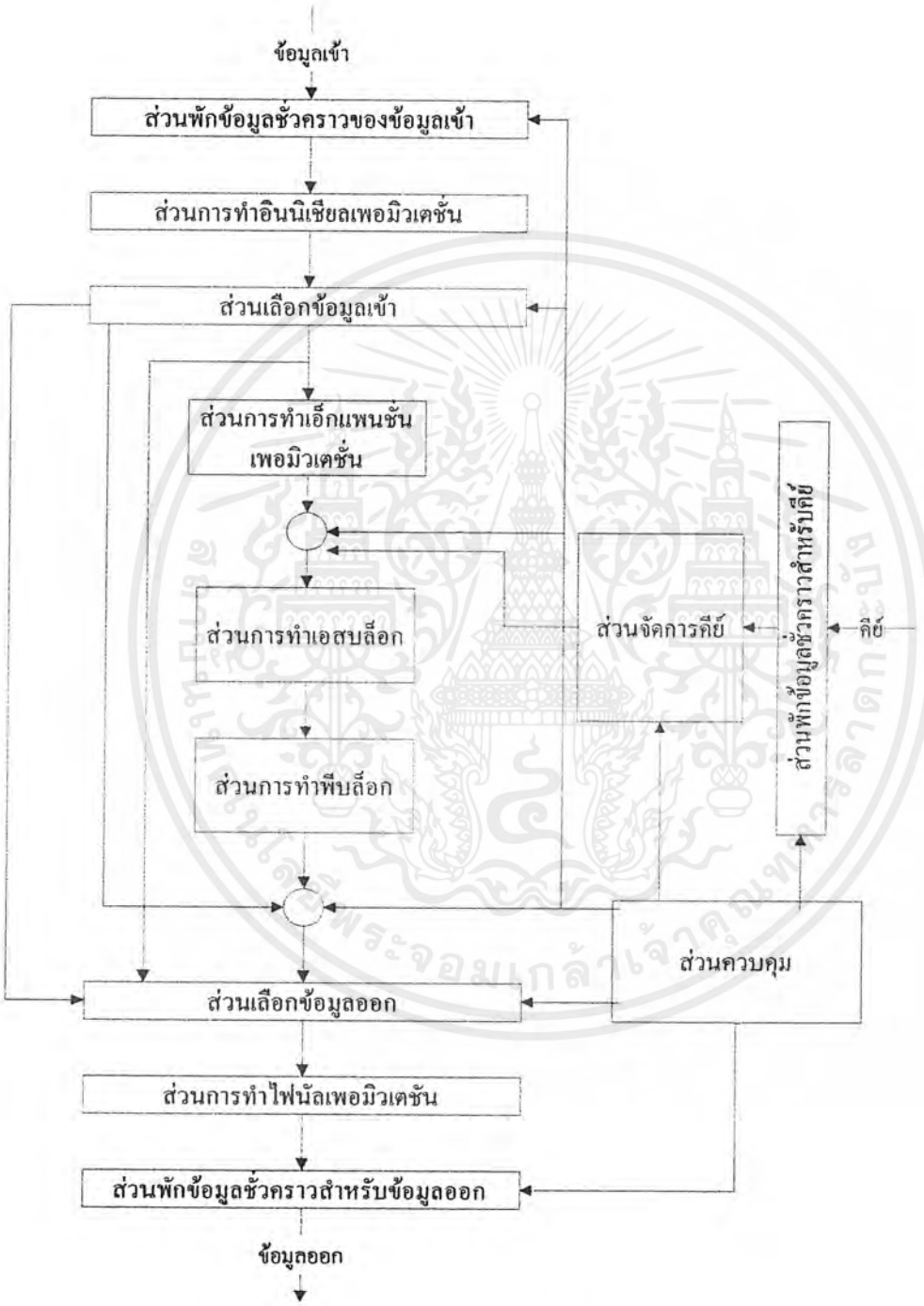
บทที่ 5
การออกแบบการเข้ารหัสและถอดรหัสดีอีเอสโดยใช้วีเฮชดีแอล

การเข้าและถอดรหัสดีอีเอส แบ่งออกได้เป็นโมดูลดังรูปที่ 5.1



รูปที่5.1 Block diagram **ของโมดูลในส่วนของดีอีเอสอัลกอริทึม**

จากรูป5.1 เราสามารถแบ่งย่อยได้อีกในส่วนของดีอีเอสอัลกอริทึม ได้ออกมาเป็นรูปที่ 5.2



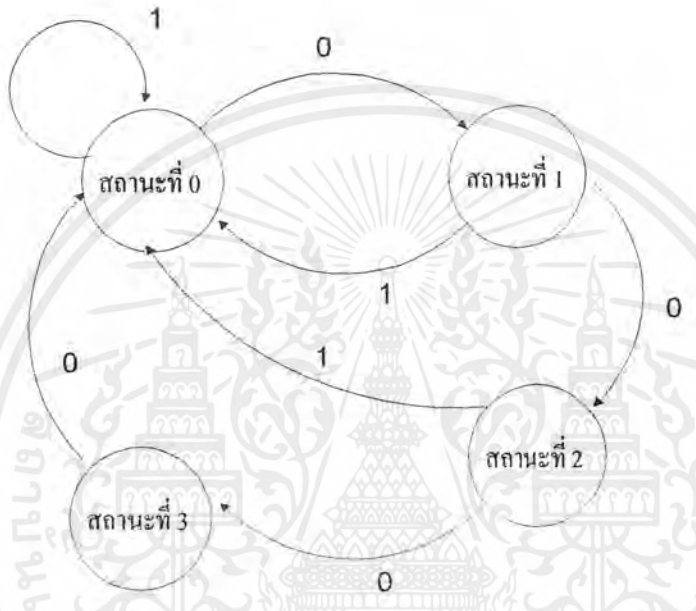
รูปที่5.2 แผนภาพแสดงส่วนประกอบทั้งหมดของดีอีเอสอัลกอริทึม

ในการออกแบบเราต้องพิจารณาการไหลของข้อมูล เมื่อมีข้อมูลเข้ามาเนื่องจากเราได้ลด ขั้นตอนที่ซ้ำลงไปเช่นการผนวกคีย์กับข้อมูล 16 ครั้งซึ่งเราสามารถลดขั้นตอนให้เป็นส่วนเดียวแล้วทำการเปลี่ยนคีย์ ไปเรื่อยๆ จบครบ 16 ครั้ง

นำแต่ละส่วนมาแปลงเป็นภาษาวีเฮชดีแอล

1 ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า (โมดูล Input Buffer)

ส่วนนี้ทำหน้าที่รับข้อมูลเข้าทีละ 16 บิตเป็นจังหวะตามสัญญาณควบคุมจากส่วนควบคุม โดยสามารถแทนส่วนนี้ได้ด้วยไฟไนสเตดแมชชีน ดังรูป 5.3



รูปที่ 5.3 ไฟไนสเตดแมชชีนของ Input Buffer module

มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นมีค่าเท่ากับ 1 และสัญญาณควบคุมเป็นขอบขาลง จะทำการตั้งสถานะของตนเองให้เป็นสถานะเริ่มต้นหรือสถานะที่ 0
- ถ้าสัญญาณอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าจำนวน 16บิตแล้วส่งเป็นบิตข้อมูลที่ 0 ถึง 15 จากนั้นเปลี่ยนสถานะจาก 0 ไปเป็นที่ สถานะ 1
- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลจำนวน 16 บิตแล้วส่งออกเป็นข้อมูลที่ 16 ถึง 31 จากนั้นเปลี่ยนสถานะจาก 1 ไปเป็นที่สถานะ 2
- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลจำนวน 16 บิตแล้วส่งออกเป็นข้อมูลที่ 32 ถึง 47 จากนั้นเปลี่ยนสถานะจาก 2 ไปเป็นที่สถานะ 3

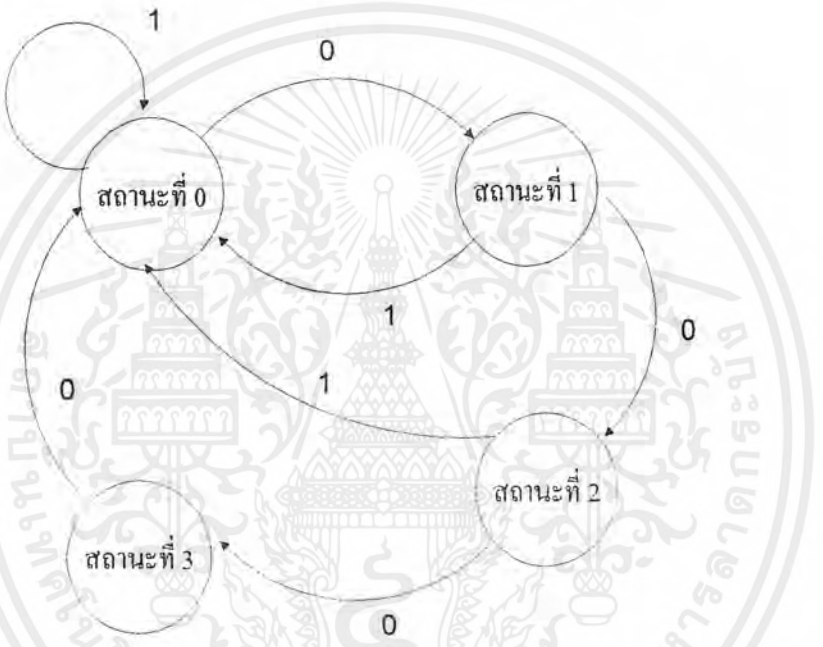
- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลจำนวน 16 บิตแล้วส่งออกเป็นข้อมูลที่ 48 ถึง 63 จากนั้นเปลี่ยนสถานะจาก 3 ไปเป็นที่สถานะ 0

2 ส่วนพักข้อมูลชั่วคราวสำหรับคีย์ (โมดูล KeyBuffer)

ส่วนนี้จะรับคีย์เข้ามาทีละ 16 บิต เป็นจังหวะตามสัญญาณควบคุม โดยสามารถแทนโดยใช้ไฟไนสเตดแมชชีนเหมือนกับ Input Buffer

3 ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออก (โมดูล output buffer)

ส่วนนี้จะรับข้อมูลเข้าจำนวน 64 บิตแล้วส่งออกทีละ 16 บิตจำนวน 4 ครั้ง โดยมีจังหวะตามสัญญาณควบคุมจากส่วนควบคุม ซึ่งสามารถแสดงการทำงานได้โดยใช้ไฟไนสเตดแมชชีน ดังรูปที่ 5.4



รูปที่ 5.4 ไฟไนสเตดแมชชีนของ output buffer module

ที่มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณควบคุมเป็นขอบขา จะทำการตั้งสถานะตนเองให้เป็นสถานะเริ่มต้น หรือ สถานะ 0
- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าที่บิตที่ 0 ถึง 15 แล้วส่งเป็นข้อมูลออก จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 1
- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าที่บิตที่ 16 ถึง 31 แล้วส่งเป็นข้อมูลออก จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 2
- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าที่บิตที่ 32 ถึง 47 แล้วส่งเป็นข้อมูลออก จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 3

- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าที่บิตที่ 48 ถึง 63 แล้วส่งเป็นข้อมูลออก จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 0

4 ส่วนการทำอินนิเชียลเพอร์มิวเทชั่น

ส่วนนี้จะทำการสลับบิตตามอัลกอริทึมดีอีเอส สามารถแสดงส่วนนี้โดยการใช้คุณสมบัติของภาษาวีเฮชดีแอล โดยเขียนแบบ structural

5 ส่วนการทำไฟนัลเพอร์มิวเทชั่น

ส่วนนี้จะทำการสลับบิตตามอัลกอริทึมดีอีเอส สามารถแสดงส่วนนี้โดยการใช้คุณสมบัติของภาษาวีเฮชดีแอล โดยเขียนแบบ structural

6 ส่วนการทำคีเพอร์มิวเทชั่น

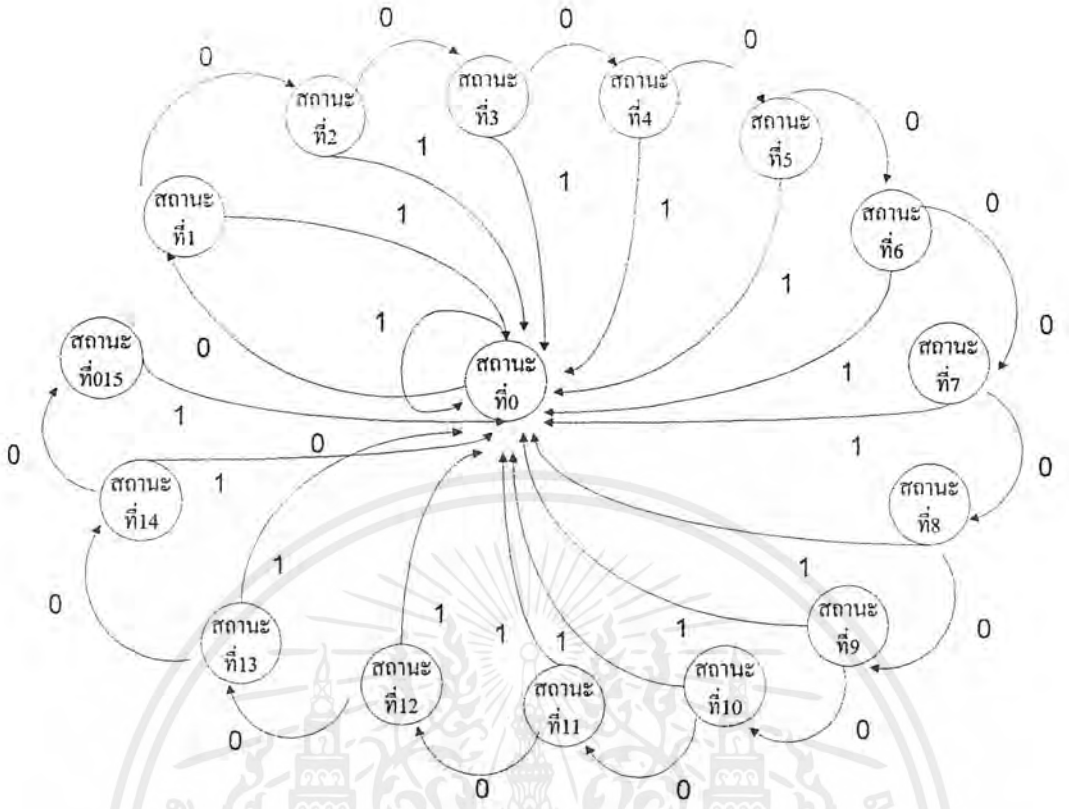
ส่วนนี้จะทำการสลับบิตตามอัลกอริทึมดีอีเอส สามารถแสดงส่วนนี้โดยการใช้คุณสมบัติของภาษาวีเฮชดีแอล โดยเขียนแบบ structural

7 ส่วนจัดการคีย์ (โมดูล key)

ส่วนนี้ทำหน้าที่เปลี่ยนตำแหน่งบิตของคีย์ตามวิธีการดีอีเอส โดยคีย์ที่ได้จะแตกต่างกันในแต่ละรอบของการเข้ารหัสและถอดรหัส ซึ่งในส่วนนี้ได้แยกย่อยออกเป็น 3ส่วนย่อยๆ คือ

7.1 ส่วนจัดการกับคีย์ที่ใช้ในการเข้ารหัส (โมดูล key for data encryption)

ในส่วนนี้จะทำการสลับบิตของคีย์ซึ่งแต่ละรอบแตกต่างกัน โดยอาศัยสัญญาณควบคุม จากส่วนควบคุมเป็นตัวกำหนดรอบของการเข้ารหัส สามารถแทนได้โดยใช้ไฟไนสเตตแมชชีน ดังรูปที่ 5.5



รูปที่ 5.5 **ไฟไนสเตดแมชชีนของ** Key for data encryption

มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นวงจรมีค่าเท่ากับ1 และสัญญาณควบคุมเป็นขอบขาลง จะทำการตั้งสถานะให้เป็นสถานะเริ่มต้นหรือสถานะที่0
- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่1 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 1
- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 2 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 2
- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 3 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 3
- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 4 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 4
- ถ้าอยู่ในสถานะที่ 4 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 5 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 5
- ถ้าอยู่ในสถานะที่ 5 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 6 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 6

- ถ้าอยู่ในสถานะที่ 6 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 7 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 7
- ถ้าอยู่ในสถานะที่ 7 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 8 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 8
- ถ้าอยู่ในสถานะที่ 8 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 9 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 9
- ถ้าอยู่ในสถานะที่ 9 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 10 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 10
- ถ้าอยู่ในสถานะที่ 10 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 11 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 11
- ถ้าอยู่ในสถานะที่ 11 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 12 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 12
- ถ้าอยู่ในสถานะที่ 12 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 13 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 13
- ถ้าอยู่ในสถานะที่ 13 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 14 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 14
- ถ้าอยู่ในสถานะที่ 14 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 15 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 15
- ถ้าอยู่ในสถานะที่ 15 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามวิธีการดีอีเอส ในรอบที่ 0 จากนั้นเปลี่ยนสถานะให้เป็นสถานะที่ 0

7.2 ส่วนจัดการกับคีย์ที่ใช้ในการถอดรหัส (โมดูล Key for data Decryption)

เหมือนกับส่วนจัดการคีย์ที่ใช้ในการเข้ารหัสแต่การสลับคีย์จะทำตามการถอดรหัสดีอีเอส

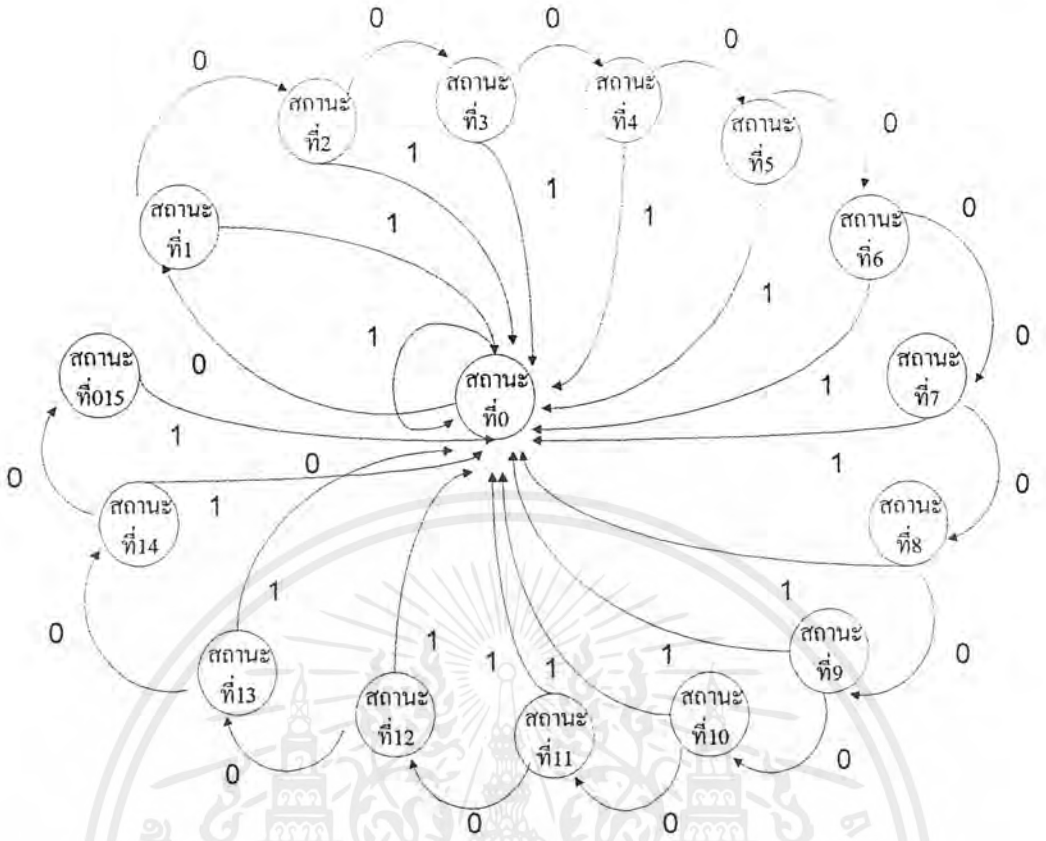
7.3 ส่วนเลือกว่าจะใช้คีย์เพื่อการเข้ารหัสและถอดรหัส (โมดูล Key selector)

ส่วนนี้จะรับข้อมูลทั้งการเข้าและถอดรหัส แต่จะเลือกเอาเพียง 1 ชุด โดยใช้สัญญาณเข้ารหัส หรือถอดรหัสเป็นตัวกำหนด สามารถแสดงส่วนนี้ได้ด้วยวงจรซิเควนเชียล ซึ่งแสดงการทำงานได้ดังนี้

- ถ้าสัญญาณแสดงการเข้าหรือถอดรหัสเป็น1ให้เลือกส่วนการเข้ารหัส นอกเหนือจากนั้นจะเป็นส่วนการถอดรหัส

8 ส่วนการเลือกข้อมูลเข้า (โมดูล Input buffer)

ส่วนนี้จะทำหน้าที่เลือกข้อมูลเพื่อเข้ากระทำวิธีการดีอีเอส จะเลือกระหว่างข้อมูลจากภายนอกกับข้อมูลที่ได้จากการกระทำรอบที่แล้ว โดยอาศัยสัญญาณจากส่วนควบคุมสามารถแสดงด้วยไฟไนสเตดแมชชีน ได้ดังรูปที่ 5.6



รูปที่ 5.6 **ไฟไนสเตดแมชชีนของ** input Buffer

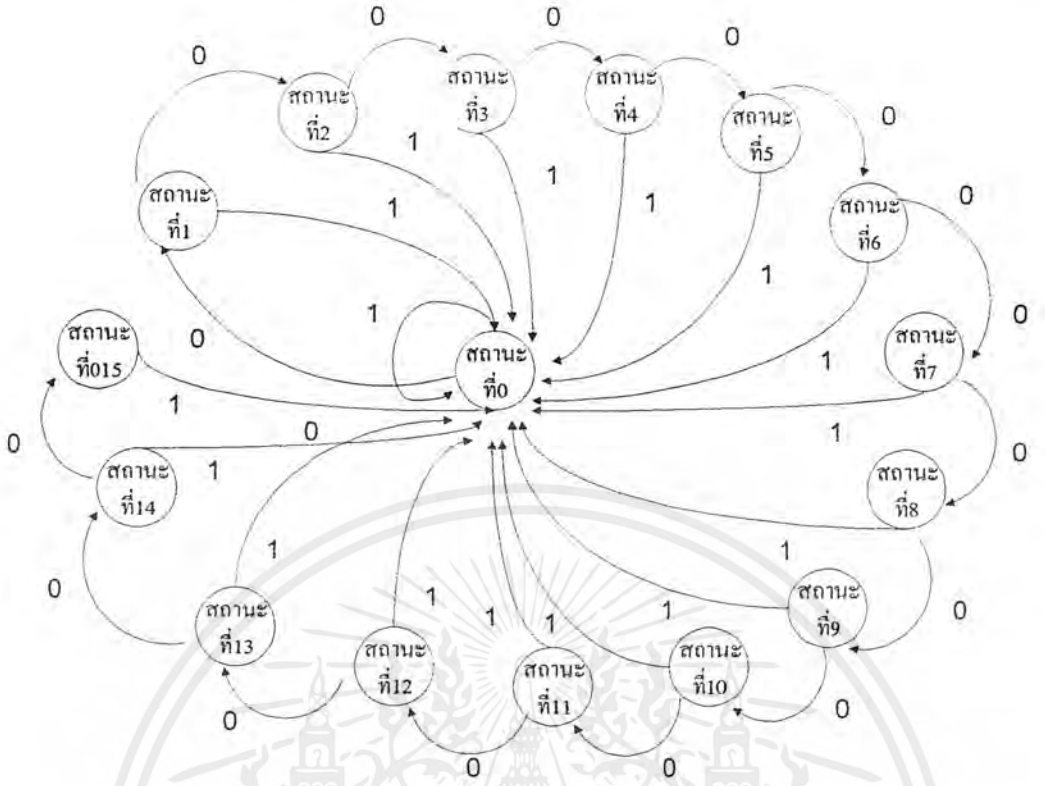
มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณควบคุมเป็นขอบขาลง จะทำการตั้งสถานะของตนเองไปเป็นสถานะเริ่มต้น หรือสถานะที่ 0
- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 1
- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 2
- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 3
- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 4
- ถ้าอยู่ในสถานะที่ 4 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 5
- ถ้าอยู่ในสถานะที่ 5 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 6

- ถ้าอยู่ในสถานะที่ 6 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 7
- ถ้าอยู่ในสถานะที่ 7 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 8
- ถ้าอยู่ในสถานะที่ 8 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 9
- ถ้าอยู่ในสถานะที่ 9 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 10
- ถ้าอยู่ในสถานะที่ 10 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 11
- ถ้าอยู่ในสถานะที่ 11 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 12
- ถ้าอยู่ในสถานะที่ 12 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 13
- ถ้าอยู่ในสถานะที่ 13 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 14
- ถ้าอยู่ในสถานะที่ 14 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 15
- ถ้าอยู่ในสถานะที่ 15 สัญญาณเริ่มต้นของวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นเปลี่ยนไปเป็นสถานะที่ 0

9 ส่วนเลือกข้อมูลออก (โมดูล Output buffer)

ส่วนนี้จะทำการเลือกข้อมูลเพื่อส่งไปยังรอบต่อไป หรือส่งข้อมูลออกเมื่อทำครบ 16 รอบ โดยอาศัยสัญญาณจากส่วนควบคุม สามารถแสดงได้โดยไฟไนสเตตแมชชีน ได้ดังรูปที่ 5.7



รูปที่ 5.7 ไฟไนสเตดแมชชีนของ output buffer

มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณควบคุมเป็นขอบขาลง จะทำการตั้งสถานะของตัวเองไปเป็นสถานะเริ่มต้น หรือสถานะ 0
- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 1
- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 2
- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 3
- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 4

- ถ้าอยู่ในสถานะที่ 4 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 5
- ถ้าอยู่ในสถานะที่ 5 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 6
- ถ้าอยู่ในสถานะที่ 6 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 7
- ถ้าอยู่ในสถานะที่ 7 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 8
- ถ้าอยู่ในสถานะที่ 9 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 10
- ถ้าอยู่ในสถานะที่ 10 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 11
- ถ้าอยู่ในสถานะที่ 11 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 12
- ถ้าอยู่ในสถานะที่ 12 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 13
- ถ้าอยู่ในสถานะที่ 13 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 14
- ถ้าอยู่ในสถานะที่ 14 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 15
- ถ้าอยู่ในสถานะที่ 15 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อทำรอบต่อไป จากนั้นเปลี่ยนสถานะเป็นสถานะที่ 0

10 ส่วนการทำเอ็กแพนชั่นเพอมิวเทชั่น

ส่วนนี้จะทำการขยายจำนวนบิตตามวิธีการดีอีเอส สามารถแสดงส่วนนี้ได้โดยการใช้คุณสมบัติของภาษาวีเฮชดีแอล โดยเขียนแบบ structural

11 ส่วนการทำเอ็กครูซีฟออร์ 48 บิต (โมดูล XOR 48)

ส่วนนี้จะทำการเอ็กครูซีฟออร์แบบบิตต่อบิตกับข้อมูล 48 บิต โดยมีสัญญาณจากส่วนควบคุมเป็นตัวกำหนดให้เริ่มการกระทำเอ็กครูซีฟออร์ ซึ่งสามารถแสดงโดยวงจรซีเควนเชียล แสดงการทำงานได้ดังนี้

- ถ้าสัญญาณควบคุมเป็น 1 ให้ทำการเอ็กครูซีฟออร์ทั้ง 48 บิต

12 ส่วนการทำเอ็กครูซีฟออร์ 32 บิต (โมดูล XOR 32)

ส่วนนี้จะทำการเอ็กครูซีฟข้อมูล 321 บิตแบบบิตต่อบิต โดยมีสัญญาณจากส่วนควบคุมเป็นตัวกำหนดการทำเอ็กครูซีฟออร์ ซึ่งสามารถแสดงได้ด้วยวงจรซีเควนเชียล ซึ่งแสดงการทำงานได้ดังนี้

- ถ้าสัญญาณควบคุมเป็น 1 ให้ทำการเอ็กครูซีฟออร์ทั้ง 32 บิต

13 ส่วนการทำเอสบล็อก

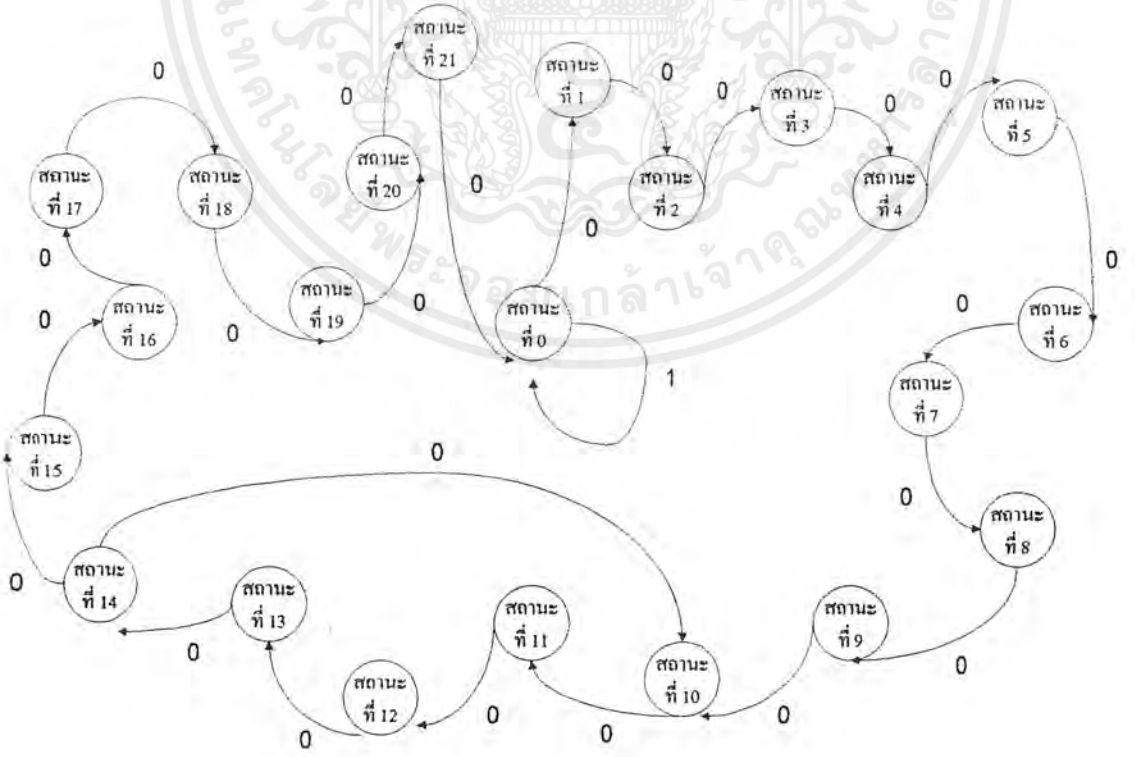
ส่วนนี้จะทำการแทนรูปแบบของข้อมูลขนาด 48บิตด้วยรูปแบบหนึ่งที่มีขนาด 32 บิต ตามวิธีการดีอีเอส สามารถแสดงด้วยวงจรซีเควนเชียล

14 ส่วนการทำพีบล็อก

ส่วนนี้จะทำการสลับตำแหน่งของข้อมูลขนาด 32 บิต ตามวิธีการดีอีเอส สามารถแสดงส่วนนี้ได้โดยการใช้คุณสมบัติของภาษาวีเฮชดีแอล โดยเขียนแบบ structural

15 ส่วนควบคุมกลาง (โมดูล Signal control)

ส่วนนี้จะส่งสัญญาณเริ่มต้นและสัญญาณควบคุมให้กับทุกๆ ส่วนโดยอาศัยสัญญาณนาฬิกาภายนอกเป็นตัวกำหนดจังหวะ สามารถแสดงได้โดยไฟไนสเตดแมชชีนได้ดังรูปที่ 5.8



รูปที่ 5.8 ไฟไนสเตดแมชชีนของ Signal control

มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการตั้งสถานะของตนเองเป็นสถานะเริ่มต้นหรือ สถานะ 0
- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ไปยังขาสัญญาณเริ่มต้นและขาสัญญาณควบคุม ให้กับทุกๆ ส่วนที่เป็นไฟไนสเตดแมชชีน จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 1
- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังขาสัญญาณเริ่มต้นและขาสัญญาณควบคุม ซึ่งขณะนี้ทุกส่วนจะมีสถานะเป็นสถานะเริ่มต้น จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 2
- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ไปยังขาสัญญาณเริ่มต้นและขาสัญญาณควบคุม ให้กับทุกๆ ส่วนที่เป็นไฟไนสเตดแมชชีน จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 3
- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้กับส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์เพื่อสั่งให้ทำการรับข้อมูล 16 บิตแรก จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 4
- ถ้าอยู่ในสถานะที่ 4 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ให้กับส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์ จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 5
- ถ้าอยู่ในสถานะที่ 5 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้กับส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์เพื่อสั่งให้ทำการรับข้อมูล 16 บิต ที่ 2 จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 6
- ถ้าอยู่ในสถานะที่ 6 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ให้กับส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์ จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 7
- ถ้าอยู่ในสถานะที่ 7 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้กับส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์เพื่อสั่งให้ทำการรับข้อมูล 16 บิต ที่ 3 จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 8
- ถ้าอยู่ในสถานะที่ 8 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ให้กับส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์ จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 9
- ถ้าอยู่ในสถานะที่ 9 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้กับส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์เพื่อสั่งให้ทำการรับข้อมูล 16 บิต ที่ 4จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 10

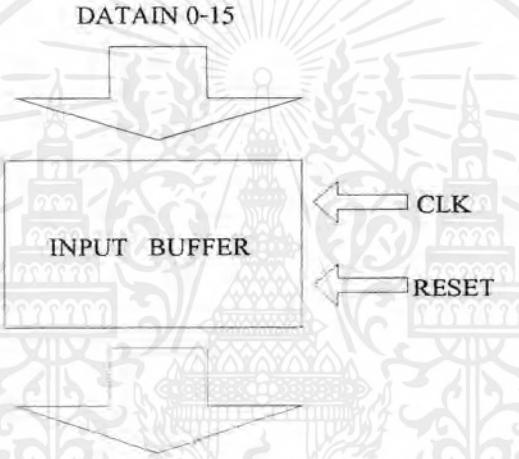
- ถ้าอยู่ในสถานะที่ 10 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้ส่วนเลือกข้อมูลเข้าและส่วนจัดการคีย์เพื่อให้ทำงาน จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 11
- ถ้าอยู่ในสถานะที่ 11 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้ส่วนการทำเอ็กครูซีฟออร์ 48 บิตเพื่อให้ทำงาน จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 12
- ถ้าอยู่ในสถานะที่ 12 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้กับส่วนการทำเอ็กครูซีฟออร์ 32 บิตเพื่อให้ทำงาน จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 13
- ถ้าอยู่ในสถานะที่ 13 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้ส่วนเลือกข้อมูลออกเพื่อให้ทำงาน จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 14
- ถ้าอยู่ในสถานะที่ 14 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการตรวจสอบว่าได้กระทำตามสถานะ 10 ถึง 14 กี่ครั้งแล้ว ถ้าทำครบ 16 ครั้ง ก็จะเปลี่ยนสถานะเป็น 15 แต่ถ้ายังไปครบให้เปลี่ยนสถานะเป็น 10
- ถ้าอยู่ในสถานะที่ 15 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออกเพื่อสั่งให้ทำการส่งข้อมูลออก 16 บิตแรก จากนั้นเปลี่ยนสถานะเป็น 16
- ถ้าอยู่ในสถานะที่ 16 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ให้ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออก จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 17
- ถ้าอยู่ในสถานะที่ 17 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออกเพื่อสั่งให้ทำการส่งข้อมูลออก 16 บิตที่ 2 จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 18
- ถ้าอยู่ในสถานะที่ 18 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ให้ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออก จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 19
- ถ้าอยู่ในสถานะที่ 19 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออกเพื่อสั่งให้ทำการส่งข้อมูลออก 16 บิตที่ 3 จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 20
- ถ้าอยู่ในสถานะที่ 20 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ให้ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออก จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 21

- ถ้าอยู่ในสถานะที่ 21 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ให้ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออกเพื่อสั่งให้ทำการส่งข้อมูลออก 16 บิตที่ 4 จากนั้นเปลี่ยนสถานะไปเป็นสถานะ 0

การทำงานของแต่ละโมดูล

1.โมดูล Input Buffer และ Key Buffer มีการประกาศ Entity ดังนี้

```
Entity IN_BUF is
        Port(CLK : IN vlbit;
             DATAIN : IN vlbit_1d(0 to 15);
             DATAOUT : OUT vlbit_1d(0 to 63);
             RESET : IN vlbit);
End IN_BUF;
```



รูปที่ 5.9 Input Buffer

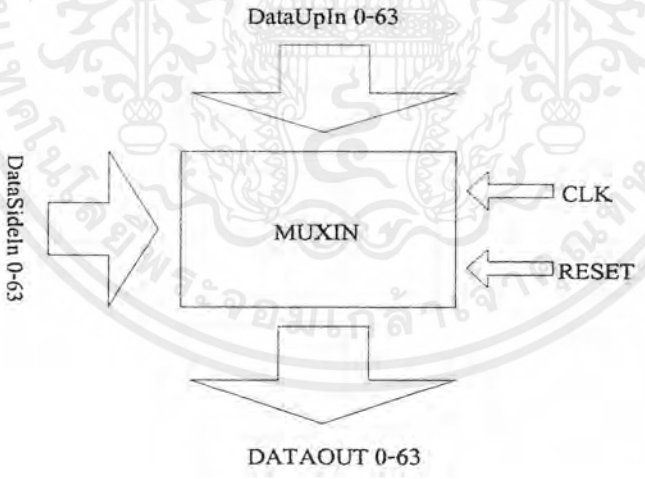
มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานและเปลี่ยนสถานะ สัญญาณ RESET เป็นสัญญาณให้มีการเริ่มการทำงาน Datain ขนาด 16 บิตเป็นขาที่รับข้อมูลที่จะเข้ามาทำการเข้าหรือถอดรหัส และขาออกคือ Dataout ขนาด 64 บิตเป็นขาที่จะส่งข้อมูลออกจากโมดูล

โดยที่ขา Key Input Buffer จะมีการประกาศ Entity คล้ายกันแต่ Datain และ Dataout จะเป็น Key ส่วน Input Buffer จะเป็นข้อมูลที่ทำการเข้ารหัสและถอดรหัส

โมดูลนี้มีการทำงานคือการรับข้อมูลจำนวน 16 บิต 4 ชุดมาทำการมัลติเพลกเซอร์ เป็นข้อมูลขนาด 64 บิตเพื่อส่งให้โมดูลอื่นเพื่อที่ทำงานต่อไป

2. โมดูล Input Multiplexor มีการประกาศ Entity ดังนี้

```
Entity MUXIN is
    Port(CI\LK : IN
        DataUpIn0, DataUpIn1, DataUpIn2, DataUpIn3, DataUpIn4, DataUpIn5,
        DataUpIn6, DataUpIn7, DataUpIn8, DataUpIn9, DataUpIn10, DataUpIn11,
        DataUpIn12, DataUpIn13, DataUpIn14, DataUpIn15, DataUpIn16, DataUpIn17,
        DataUpIn18, DataUpIn19, DataUpIn20, DataUpIn21, DataUpIn22, DataUpIn23,
        DataUpIn24, DataUpIn25, DataUpIn26, DataUpIn27, DataUpIn28, DataUpIn29,
        DataUpIn30, DataUpIn31, DataUpIn32, DataUpIn33, DataUpIn34, DataUpIn35,
        DataUpIn36, DataUpIn37, DataUpIn38, DataUpIn39, DataUpIn40, DataUpIn41,
        DataUpIn42, DataUpIn43, DataUpIn44, DataUpIn45, DataUpIn46, DataUpIn47,
        DataUpIn48, DataUpIn49, DataUpIn50, DataUpIn51, DataUpIn52, DataUpIn53,
        DataUpIn54, DataUpIn55, DataUpIn56, DataUpIn57, DataUpIn58, DataUpIn59,
        DataUpIn60, DataUpIn61, DataUpIn62, DataUpIn63 : IN vlbit;
        DataSidein : IN vlbit_1d(0 to 63);
        DataOut : OUT vlbit_1d(0 to 63);
        RESET : IN vlbit);
End MUXIN;
```



รูปที่ 5.10 Input Multiplexor

มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้ทีการทำงานและเปลี่ยนสถานะ DataUpIn ขนาด 64 บิตจะรับข้อมูลจากโมดูล Input Buffer ซึ่งจะเป็นข้อมูลที่จะทำการเข้ารหัสหรือถอดรหัส โดยรับเฉพาะในรอบแรกของการทำงานเท่านั้น

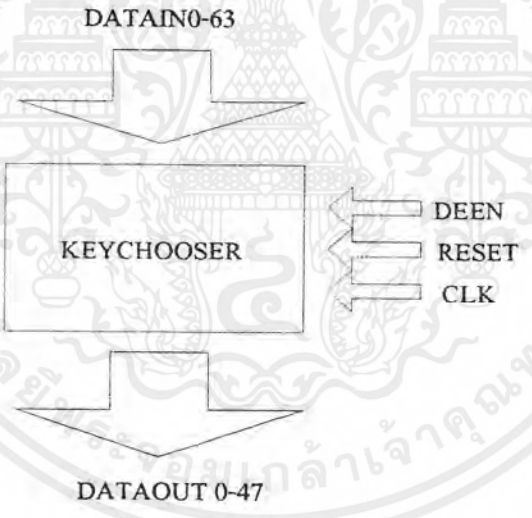
DataSidein ขนาด 64 บิตจะรับข้อมูลที่ผ่านการทำงานในแต่ละรอบมาแล้ว โดยจะรับมาจากโมดูล Output Multiplexor

ขาออก คือ Dataout ขนาด 64 บิตจะส่งข้อมูลให้โมดูลต่อไป

โมดูลนี้มีการทำงานคือ ในรอบที่ 1 จะรับข้อมูลจากขา DataUpIn ซึ่งเป็นขาที่จะรับข้อมูลมาจากภายนอก ส่วนรอบที่ 2 ถึง 16 จะรับข้อมูลที่จะวนเข้ามาทำการเข้ารหัส แล้วส่งข้อมูลไปยังโมดูลอื่นให้ทำงานต่อ

3. โมดูล Key มีการประกาศ Entity ดังนี้

```
Entity KEYCHOOSE is
    Port( CLK : IN vlbit;
          Datain : IN vlbit_1d(0 to 55);
          DataOut : OUT vlbit_1d(0 to 55);
          DEEN : IN vlbit;
          Reset : IN vlbit );
End KEYCHOOSE;
```



รูปที่ 5.11 Key

มีสัญญาณเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานและเปลี่ยนสถานะDataIn ขนาด 64 บิตจะรับข้อมูลจากโมดูล Key Input Buffer

DEEN เป็นขาสัญญาณเพื่อสั่งให้ทำการเข้ารหัสหรือถอดรหัส

Reset เป็นขาสัญญาณเพื่อให้เริ่มต้นการทำงานของโมดูล

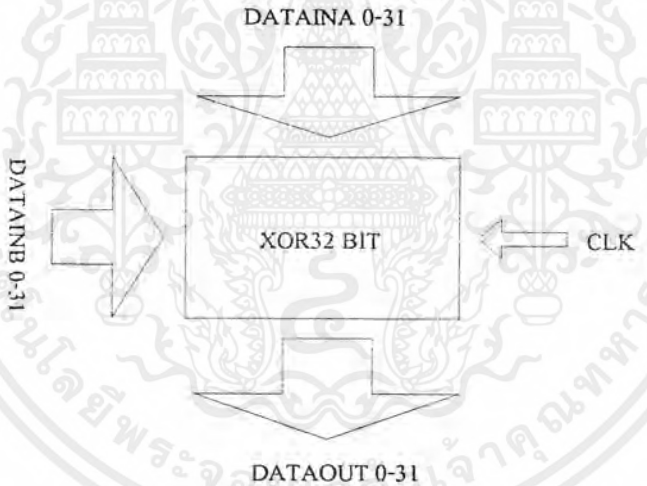
ขาออก คือ DataOut ขนาด 56 บิตจะส่งข้อมูลให้โมดูลอื่นทำงานต่อไป

โมดูลนี้มีการทำงานคือ จะรับ Key ขนาด 64 บิตจากโมดูล Key Input Buffer มาแล้วทำการ

ตัดบิตที่ 8 ,16 ,24 ,... ,56 ,64 ออก แล้วทำการดูขา DEEN ว่าเป็น 0 หรือ 1 โดยถ้าเป็น 0 จะเป็นการเข้ารหัส และถ้าเป็น 1 จะเป็นการถอดรหัส จากนั้นจะทำการสร้างข้อมูลเพื่อใช้ในการเข้ารหัสหรือถอดรหัสสำหรับแต่ละรอบทั้ง 16 รอบ

4. โมดูล Exclusive OR 32 บิตและ 48 บิต มีการประกาศ Entity ดังนี้

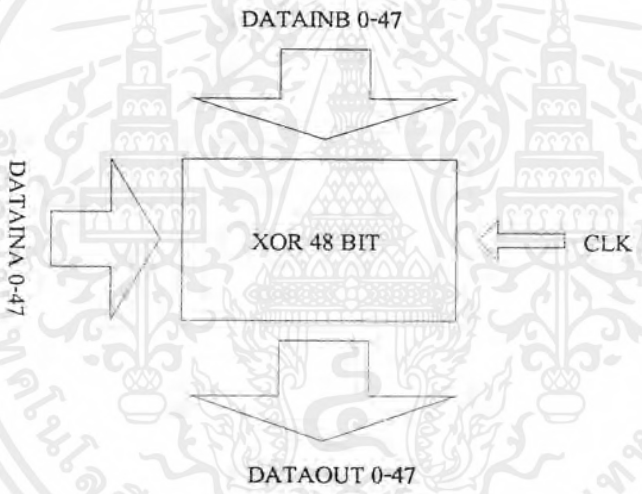
```
Entity XOR32 is
    port( DatainA0, DatainA1, DatainA2, DatainA3, DatainA4, DatainA5, DatainA6,
          DatainA7, DatainA8, DatainA9, DatainA10, DatainA11, DatainA12, DatainA13,
          DatainA14, DatainA15, DatainA16, DatainA17, DatainA18, DatainA19, DatainA20,
          DatainA21, DatainA22, DatainA23, DatainA24, DatainA25, DatainA26, DatainA27,
          DatainA28, DatainA29, DatainA30, DatainA31 : IN vlbit;
          DatainB : IN vlbit_1d(0 to 31);
          Dataout : Out vlbit_1d(0 to 31);
          CLK : IN vlbit);
end XOR32;
```



รูปที่ 5.12 Exclusive OR 32

```
Entity XOR48 is
    port( DatainA0, DatainA1, DatainA2, DatainA3, DatainA4, DatainA5, DatainA6,
          DatainA7, DatainA8, DatainA9, DatainA10, DatainA11, DatainA12,
          DatainA13, DatainA14, DatainA15, DatainA16, DatainA17, DatainA18, DatainA19,
          DatainA20, DatainA21, DatainA22, DatainA23, DatainA24, DatainA25, DatainA26,
          DatainA27, DatainA28, DatainA29, DatainA30, DatainA31, DatainA32, DatainA33,
          DatainA34, DatainA35, DatainA36, DatainA37, DatainA38, DatainA39, DatainA40,
```

```
DatainA41, DatainA42, DatainA43, DatainA44, DatainA45, DatainA46, DatainA47 : IN
vlbit;
DatainB0, DatainB1, DatainB2, DatainB3, DatainB4, DatainB5, DatainB6, DatainB7,
DatainB8, DatainB9, DatainB10, DatainB11, DatainB12,
DatainB13, Datain14, DatainB15, DatainB16, DatainB17, DatainB18,
DatainB19, DatainB20, DatainB21, DatainB22, DatainB23, DatainB24, DatainB25,
DatainB26, DatainB27, DatainB28, DatainB29, DatainB30, DatainB31, DatainB32,
DatainB33, DatainB34, DatainB35, DatainB36, DatainB37, DatainB38, DatainB39,
DatainB40, DatainB41, DatainB42, DatainB43, DatainB44, DatainB45, DatainB46,
DatainB47 : IN vlbit;
Dataout : Out vlbit_1d(0 to 47);
CLK : IN vlbit);
end XOR48
```



รูปที่ 5.13 Exclusive OR 48

มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานและเปลี่ยนสถานะ

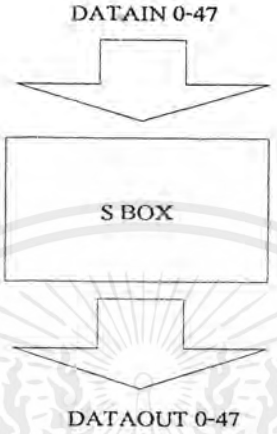
DatainA และDatainB เป็นขาข้อมูลที่จะทำการ Exclusive OR

ขาออกคือ Dataout เพื่อส่งข้อมูลให้โมดูลอื่น

โมดูลนี้มีการทำงานคือ จะรับข้อมูลเข้ามา 2 ชุดแล้วทำการ Exclusive OR จากนั้นจะทำการส่งข้อมูลให้โมดูลอื่นเพื่อทำงานต่อไป

5. โมดูล S_Box มีการประกาศ Entity ดังนี้

```
Entity S_Box is
        port( DataIn : IN vlbit_1d(0 to 47);
              DataOut : Out vlbit_1d(0 to 31);
end S_BOX
```



รูปที่ 5.14 S_Box

มีสัญญาณขาเข้าคือ DataIn ขนาด 48 บิต รับข้อมูลเข้า

ขาออกคือ DataOut ขนาด 32 บิตส่งข้อมูลออก

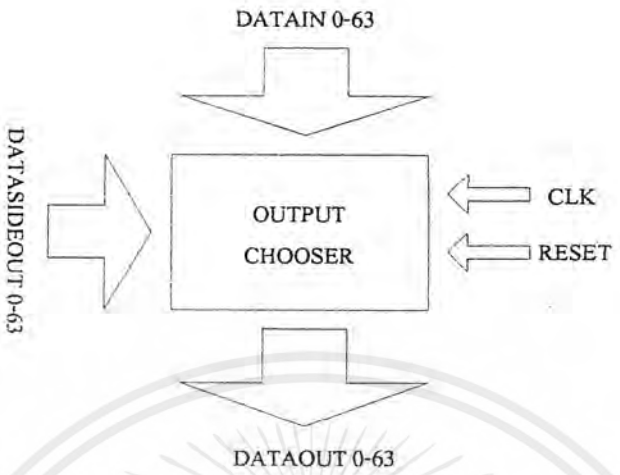
โมดูลนี้มีการทำงานคือ นำข้อมูล 48 บิตมาแบ่งเป็น 8 ชุดๆละ 6 บิต และทำการผ่านตารางเพื่อลดจำนวนลงเหลือ 8 ชุดๆละ 4 บิตรวม 32 บิต แล้วส่งให้โมดูลอื่นทำงานต่อไป

6. โมดูล Output Multiplexor มีการประกาศ Entity ดังนี้

```
Entity MUXOUT is
        port ( CLK : IN vlbit;
               DataDownout : Out vlbit_1d(0 to 63);
               DataIn0, DataIn1, DataIn2, DataIn3, DataIn4, DataIn5, DataIn6, DataIn7, DataIn8,
               DataIn9, DataIn10, DataIn11, DataIn12, DataIn13, DataIn14, DataIn15, DataIn16,
               DataIn17, DataIn18, DataIn19, DataIn20, DataIn21, DataIn22, DataIn23, DataIn24,
               DataIn25, DataIn26, DataIn27, DataIn28, DataIn29, DataIn30, DataIn31, DataIn32,
               DataIn33, DataIn34, DataIn35, DataIn36, DataIn37, DataIn38, DataIn39, DataIn40,
               DataIn41, DataIn42, DataIn43, DataIn44, DataIn45, DataIn46, DataIn47, DataIn48,
               DataIn49, DataIn50, DataIn51, DataIn52, DataIn53, DataIn54, DataIn55, DataIn56,
               DataIn57, DataIn58, DataIn59, DataIn60, DataIn61, DataIn62, DataIn63 : IN vlbit;
               DataSideout : OUT vlbit_1d(0 to 63);
```

RESET : IN vlbit);

end MUXOUT;



รูปที่ 5.15 Output Multiplexor

มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานและเปลี่ยนสถานะ

Reset เป็นสัญญาณเพื่อเริ่มต้นการทำงานของโมดูล

DataIn ขนาด 64 บิต เป็นขาข้อมูลที่รับจากโมดูลอื่น

ขาออกคือ DataDownout ขนาด 64 บิตเป็นขาข้อมูลที่จะส่งไปยังโมดูล Output Buffer ในรอบที่ 16 ของการทำงาน เพื่อส่งเป็น Output ต่อไป

DataSideout ขนาด 64 บิตเป็นขาข้อมูลที่จะส่งไปยังโมดูล Input Multiplexor ในรอบที่ 1 ถึง 15 ของการทำงาน

โมดูลนี้มีการทำงานคือ รับข้อมูลเข้ามาจากนั้นจะทำการดูสถานะว่าเป็นรอบที่เท่าใด ถ้าเป็นรอบที่ 1 ถึง 15 จะนำข้อมูลออกที่ขา DataSideout เพื่อนำข้อมูลวนกลับไปทำงานอีก แต่ถ้าเป็นรอบที่ 16 จะส่งข้อมูลออกที่ขา DataDownout เพื่อส่งข้อมูลเป็น Output ต่อไป

7. โมดูล Output Buffer มีการประกาศ Entity ดังนี้

Entity OUT_BUF is

port (CLK : In vlbit;

DATAIN0, DATAIN1, DATAIN2, DATAIN3, DATAIN4, DATAIN5, DATAIN6, DATAIN7,

DATAIN8, DATAIN9, DATAIN10, DATAIN11, DATAIN12,

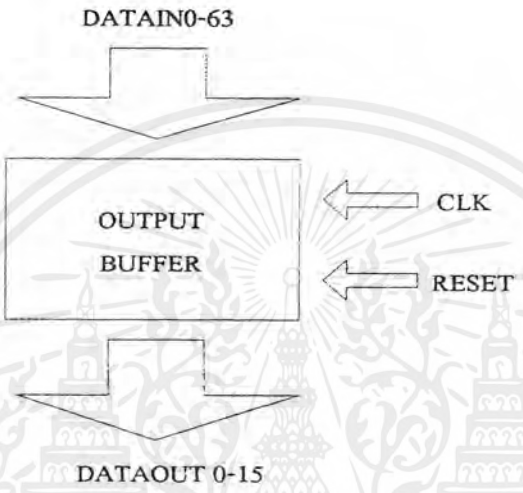
DATAIN13, DATAIN14, DATAIN15, DATAIN16, DATAIN17, DATAIN18, DATAIN19,

DATAIN20, DATAIN21, DATAIN22, DATAIN23, DATAIN24, DATAIN25, DATAIN26,

DATAIN27, DATAIN28, DATAIN29, DATAIN30, DATAIN31, DATAIN32, DATAIN33,

DATAIN34, DATAIN35, DATAIN36, DATAIN37, DATAIN38, DATAIN39, DATAIN40,

```
        DATAIN41, DATAIN42, DATAIN43, DATAIN44, DATAIN45, DATAIN46, DATAIN47,
        DATAIN48, DATAIN49, DATAIN50,DATAIN51,DATAIN52,DATAIN53,DATAIN54,
        DATAIN55, DATAIN56, DATAIN57, DATAIN58, DATAIN59, DATAIN60, DATAIN61,
        DATAIN62, DATAIN63 : IN vlbit;
        DATAOUT : OUT vlbit_1d(0 to 15);
        RESET : IN vlbit);
end OUT_BUF;
```



รูปที่ 5.16 Output Buffer

มีขาสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานเปลี่ยนสถานะ

Reset เป็นขาสัญญาณเพื่อให้โมดูลเริ่มต้นทำงาน

DataIn ขนาด 64 บิต เป็นขาข้อมูล,ที่รับมาจากการเข้ารหัสและถอดรหัสที่ทำครบ 16 รอบ

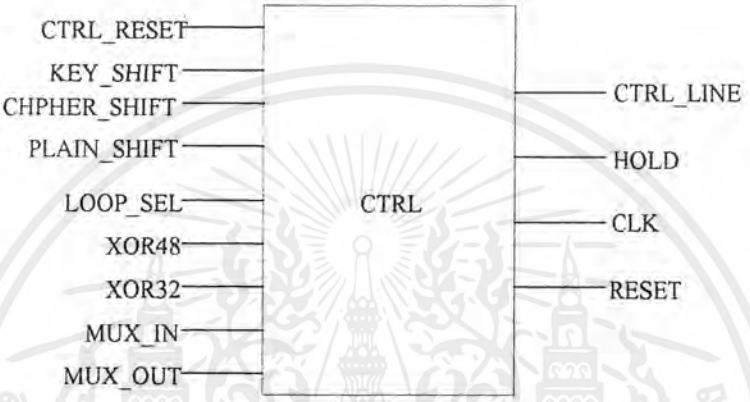
ขาออกคือ DataOut ขนาด 16 บิตเป็นขาข้อมูลที่จะนำข้อมูล 64 บิตมาแบ่งส่งครั้งละ16 บิตจำนวน 4 ชุด

โมดูลนี้มีการทำงานคือ นำข้อมูลขนาด 64 บิต มาทำการมัลติเพลกออกครั้งละ 16 บิต4 ครั้ง

8. โมดูล Singnal Control มีการประกาศ Entity

```
  Entity CRTL is
        port( CLK          : in vlbit;
              RESET        : in vlbit;
              HOLD         : out vlbit;
              CTRL_RESET   : out vlbit;
              CTRL_LINE    : out vlbit;
              PLAIN_SHIFT  : out vlbit;
```

```
        CIPHER_SHIFT : out vlbit;
        KEY_SHIFT    : out vlbit;
        LOOP_SEL     : out vlbit;
        XOR48        : out vlbit;
        XOR32        : out vlbit;
        MUX_IN       : out vlbit;
        MUX_OUT      : out vlbit);
end CTRL;
```



รูปที่ 5.17 Control Unit

มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาของระบบเพื่อให้ดีอีเอสทั้งหมดนี้มีการทำงานและเ
เปลี่ยนสถานะReset เป็นขาสัญญาณเพื่อเริ่มต้นการทำงานของดีอีเอส และเคลียร์สถานะทั้งหมดของ
ระบบขาออกคือ CTRL_RESET เป็นขาสัญญาณที่ส่งไปควบคุมโมดูลต่างๆ เพื่อทำการเซตแต่ละโมดูล
CTRL_LINE เป็นขาสัญญาณที่จะส่งไปยังอุปกรณ์ที่จะนำมาใช้กับดีอีเอส เพื่อบอกให้รู้ว่าพร้อมที่จะ
รับหรือส่งข้อมูลที่ทำการเข้ารหัสหรือถอดรหัส

PLIAN_SHIFT เป็นขาสัญญาณที่จะส่งไปเป็น clock ให้กับโมดูล Input Buffer

CIPHER_SHIFT เป็นขาสัญญาณที่จะส่งไปเป็น clock ให้กับโมดูล Output Buffer

KEY_SHIFT เป็นขาสัญญาณที่จะส่งไปเป็น clock ให้กับโมดูล Key Input Buffer

LOOP_SEL เป็นขาสัญญาณที่จะส่งไปเป็น clock ให้กับโมดูล Key

XOR48 และ XOR32 เป็นขาสัญญาณที่จะส่งไปเป็น clock ให้กับโมดูล Exclusive OR 48 และ Exclusive OR 32

MUX_IN เป็นขาสัญญาณที่จะส่งไปเป็น clock ให้กับโมดูล Input Multiplexor

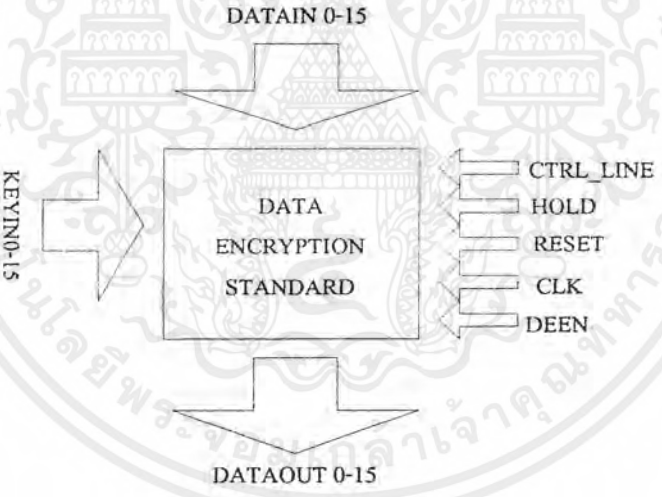
MUX_OUT เป็นขาสัญญาณที่จะส่งไปเป็น clock ให้กับโมดูล Output Multiplexor

โมดูลนี้มีการทำงานคือ จะสร้างสัญญาณเพื่อรีเซตโมดูลต่างๆ และสร้าง clock เพื่อให้

โมดูลต่างๆมีการทำงานสัมพันธ์กัน เพื่อควบคุมการทำงานให้สอดคล้องกัน โดยจะเคลียร์สถานะทั้งหมดเมื่อมีสัญญาณ Reset ซึ่งเปลี่ยนจาก 1 เป็น 0 จากนั้นจะสร้างสัญญาณ Ctrl_reset เพื่อเคลียร์สถานะของแต่ละโมดูลย่อย แล้วทำการส่ง clock ให้แต่ละโมดูลย่อยทำงาน

9. โมดูล DES มีการประกาศ Entity ดังนี้

```
Entity DES is
        port ( DATAIN          : IN vlbit_1d(0 to 15);
               HOLD            : OUT vlbit;
               CTRL_LINE       : OUT vlbit;
               KEYIN           : IN vlbit_1d(0 to 15);
               DATAOUT : OUT vlbit_1d(0 to 15);
               CLK             : IN vlbit;
               DEEN            : IN vlbit;
               CS              : IN vlbit;
end DES;
```



รูปที่ 5.18 DES

มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาของระบบซึ่งจะรับมาจากภายนอกดีอีเอส

CS เป็นสัญญาณ Chip Select เพื่อบอกให้รู้ว่าจะให้ดีอีเอสทำงาน

DataIn และKeyIn ขนาด 16 บิต เป็นขาข้อมูลซึ่งจะรับข้อมูลและ Key 64 บิตโดยรับเป็น 16 บิต 4 ชุด

DEEN เป็นขาสัญญาณเพื่อบอกว่าเป็นการเข้ารหัสหรือถอดรหัส

ขาออกคือ HOLD เป็นขาสัญญาณเพื่อให้อุปกรณ์ภายนอกรู้ว่าพร้อมจะรับและจะมีการส่งข้อมูล โดยจะเปลี่ยนจาก 0 เป็น 1

CTRL_LINE เป็นขาสัญญาณเพื่อให้อุปกรณ์ภายนอกส่งหรือรับข้อมูล 16 บิตชุดต่อไป

DataOut ขนาด 16 บิต เป็นขาข้อมูลที่ออกจากดีอีเอสซึ่งจะส่งเป็นจำนวน 4 ชุดเป็นข้อมูลขนาด 64 บิต

โมดูลนี้เป็นโมดูลซึ่งรวมโมดูลย่อยทุกๆโมดูล โดยจะเริ่มทำงานเมื่อได้รับสัญญาณ CS จากอุปกรณ์ภายนอก และจะทำงานตาม clock ของระบบ เมื่อจะทำการรับข้อมูลจะส่งสัญญาณ HOLD พร้อมกับ CTRL_LINE โดยที่ HOLD จะบอกให้รู้ว่าพร้อมที่จะรับข้อมูลโดยจะเปลี่ยนจาก 0 เป็น 1 ส่วน CTRL_LINE จะบอกให้รู้ว่าให้ส่งข้อมูล 16 บิตแต่ละชุดที่ DataIn ได้เช่นเดียวกับการส่งข้อมูลจะมีสัญญาณ HOLD และ CTRL_LINE เช่นกัน แต่จะรอรับข้อมูลออกจากขา DataOut ซึ่งจะส่งเป็นชุดๆโดยใช้ CTRL_LINE บอกเช่นกัน ในส่วนของการติดต่อยูเอสบีเราได้ออกแบบไว้ในภาคผนวก

บทที่ 6
การออกแบบโปรแกรมสนับสนุนการทำงานของวงจร

โปรแกรมที่สนับสนุนการทำงานของวงจรป้องกันการคัดลอก แบ่งออกได้เป็น 2 ส่วน

1 ส่วนการทำงานในช่วงก่อนการนำไปใช้

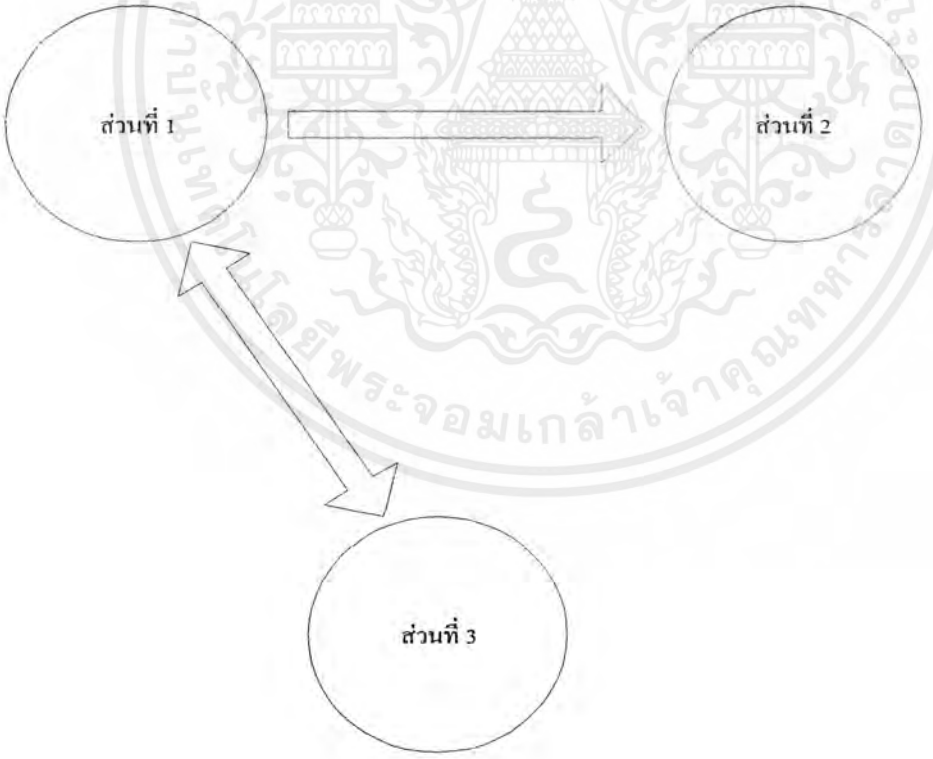
2 ส่วนการทำงานในช่วงที่ใช้งานจริงกับวงจรที่คิดขึ้น

1 **ส่วนการทำงานในช่วงก่อนการนำไปใช้**

โปรแกรมที่เราจะคิดขึ้นนี้เป็นโปรแกรมที่ใช้ในการเข้ารหัส เพื่อให้ข้อมูลที่ต้องการเป็นความลับนั้น อยู่กับผู้ผลิต เราจึงให้ส่วนนี้เก็บไว้กับผู้ผลิตซอฟต์แวร์เอง ในการทำงานของโปรแกรมส่วนนี้จะจัดการเกี่ยวกับ การเข้ารหัส โปรแกรมที่ต้องการจะป้องกัน โดย มีการออกแบบโปรแกรมแบบออปเจกต์ โดยแบ่งออกได้เป็น ส่วน แต่ละส่วนเป็น

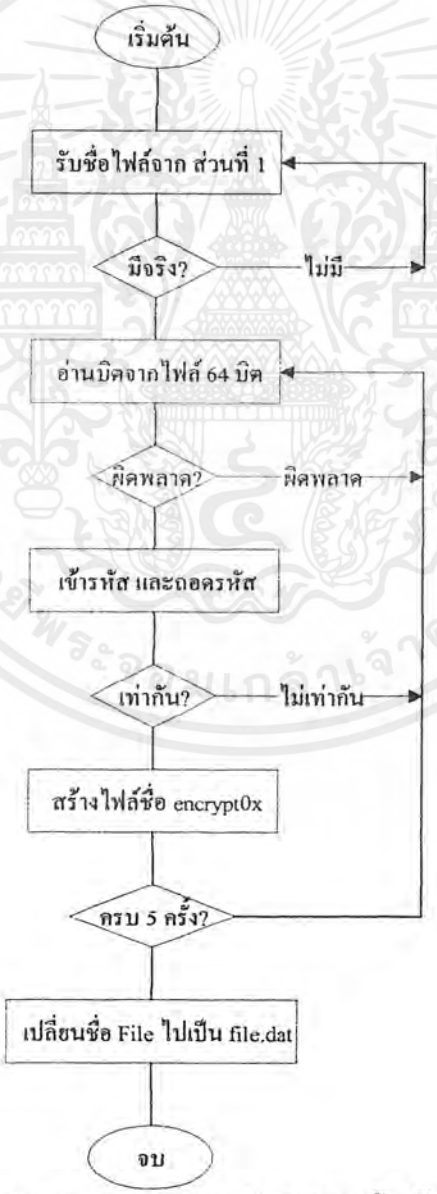
1 ส่วนที่ใช้เชื่อมต่อกับผู้ใช้
2 ส่วนที่ทำงานการเข้ารหัส
3 ส่วนตรวจสอบรหัสผ่าน

เขียนโดยมีการเชื่อมต่อดังรูป ที่ 6.1



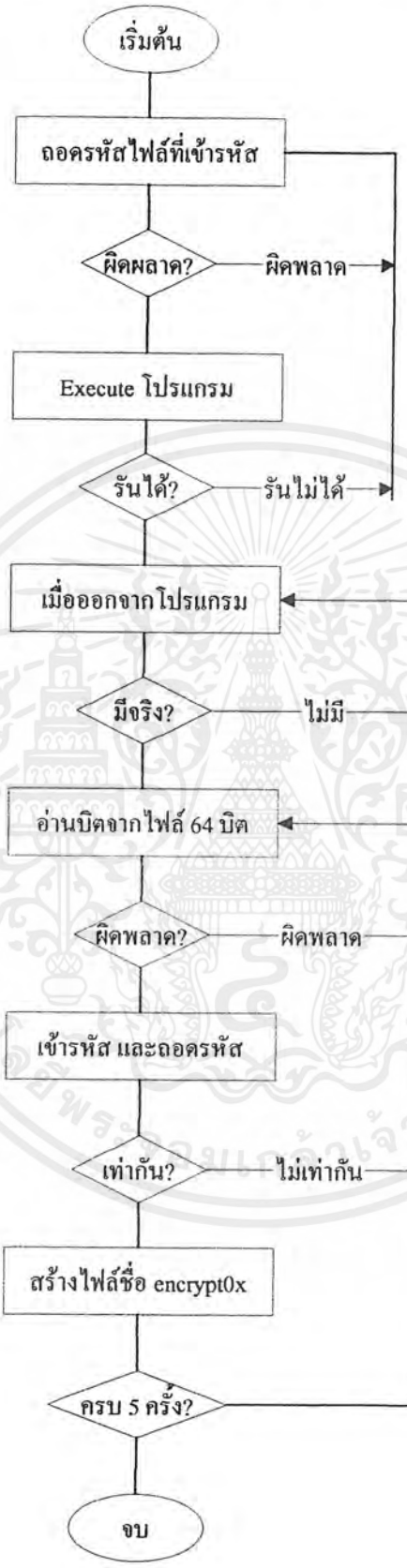
รูปที่ 6.1 การเชื่อมต่อระหว่างออปเจกต์ของโปรแกรมส่วนแรก

จากรูป 6.1 ข้อมูลการติดต่อระหว่างส่วนที่ 1 กับส่วนที่ 2 คือชื่อของโปรแกรมที่ต้องการจะทำการป้องกัน ส่วนข้อมูลที่ติดต่อระหว่างส่วนที่ 1 และ ส่วนที่ 3 เป็นบูลีนที่ได้ตรวจสอบรหัสผ่านว่าถูกต้องหรือไม่ ส่วนหลักที่เป็นหัวใจของโปรแกรมอยู่ที่ส่วนที่ 2 ซึ่งเราได้จำลองการเข้ารหัสและถอดรหัสลงเป็นภาษา C++ สามารถ ดูได้ในภาคผนวก การทำงานภายในเกิดขึ้นเมื่อ ส่วนที่ 1 ได้ส่งชื่อของโปรแกรมไปที่ส่วนที่ 2 ส่วนที่ 2 จะทำการตรวจสอบว่าโปรแกรมมีอยู่จริงแล้วทำการ อ่านบิตข้อมูลแบบสุ่มขึ้นมา 48 บิตและรวมกับ 16 บิตสุดท้ายซึ่งเป็นตำแหน่งของที่อยู่ของข้อมูลที่จะทำการเข้ารหัส แล้วส่งไปทำการเข้ารหัส เพื่อความสมบูรณ์ของการเข้ารหัสเราได้ทำการตรวจสอบว่าการเข้ารหัสนั้นสามารถถอดรหัสได้ข้อมูลเดิมหรือไม่ ถ้าไม่ได้ให้พยายามหาตำแหน่งอ่านใหม่พอตรวจสอบถูกต้องแล้วก็จะเก็บส่วนนี้ไว้เป็นไฟล์ ชื่อ Encrypt0X.dat ซึ่งเราจะทำอย่างนี้ 5 ครั้งจะได้ออกมา 5 ไฟล์ คือ encrypt00 ถึง encrypt04 แล้วส่วนที่อ่านจะถูกเปลี่ยนไปเป็นข้อมูลขยะเพื่อวางทำให้เกิดบักในโปรแกรมซึ่งทำให้โปรแกรมใช้งานไม่ได้ เมื่อทำเสร็จแล้วก็เปลี่ยนไฟล์เป็น File.dat เขียนเป็น Flow Chart ได้ดังรูปที่ 6.2



รูปที่ 6.2 Flowchart ของส่วนการทำงานการเข้ารหัสโปรแกรม

รูปที่ 6.3 การทำงานของส่วนที่ใช้ในการรันโปรแกรมของผู้ผลิต



2 ส่วนโปรแกรมที่ไปใช้งานร่วมกับวงจร

ส่วนนี้จะเป็นส่วนที่ทางผู้ผลิตซอฟต์แวร์ส่งไปให้กับผู้ใช้ทั่วไปใช้งาน การทำงานของส่วนนี้แบ่งออกไปเป็น 2 ออปเจกต์ ดังนี้

1 ส่วนเชื่อมต่อระหว่างผู้ใช้

2 ส่วนที่ใช้ทำการ Execute โปรแกรมของผู้ผลิตซอฟต์แวร์

การเชื่อมต่อการทำงานระหว่าง 2 ส่วนนี้ไม่มีข้อมูลส่งผ่านกันเป็นเพียงการเรียกใช้ออปเจกต์ ส่วนที่ 2 เท่านั้น การทำงานหลักทั้งหมดจะตกอยู่กับส่วนที่ 2

การทำงานในส่วนที่ 2 คือ การที่เอาไฟล์ที่ชื่อ File.dat และ Encrypt0x.dat มาทำการผสมกันเพื่อให้โปรแกรมสามารถทำงานได้โดยไม่มีบักเกิดขึ้น โดยจะ อ่าน Encrypt0X.dat แล้วทำการถอดรหัสแล้วต่อเข้ากับไฟล์ที่ชื่อ File.dat แล้วไปทำการ execute ในภายหลัง

เมื่อทำการ Execute เสร็จ หรือผู้ใช้เลิกการทำงานโปรแกรม ส่วนที่ 2 จะทำงานคล้ายกับส่วนที่ 2 ของโปรแกรมส่วนแรก โดยจะเริ่มทำการสุ่มอ่านบิตข้อมูล 48 บิต รวมกับตำแหน่ง 16 บิต เข้ารหัสไปเก็บไว้เป็น ไฟล์ที่ชื่อ Encrypt00 ทำซ้ำกัน 5 ครั้ง เหมือนกัน จะได้รูปแบบใหม่ของไฟล์ของชุดที่เราเข้ารหัสเป็นชุดใหม่ คือทุกครั้งที่เราทำการ Execute โปรแกรม ข้อมูลทุกอย่างในไฟล์ทั้งหมดจะเปลี่ยนไป ไม่เหมือนเดิม ยากแก่การแกะรอย การทำงานทั้งหมดของส่วนนี้แสดงเป็น Flow chart ได้ดังรูปที่ 6.3

บทที่ 7
บทสรุปและวิจารณ์

บทสรุป

ในการออกแบบวงจรป้องกันการคัดลอกโดยใช้เอฟพีจีเอนั้น ทำให้เกิดความปลอดภัยกับซอฟต์แวร์มากยิ่งขึ้นเนื่องจากเอฟพีจีเอนั้น ไม่สามารถลอกเลียนวงจรภายในได้ ถึงในปัจจุบันอาจจะมีการลอกเลียนแบบได้ แต่ใช้ค่าใช้จ่ายสูงมาก ในการนำเอาเอฟพีจีเอไปลอกเลียนแบบ ดังนั้นการที่เราใช้เอฟพีจีเอในการทำวงจรป้องกันการคัดลอกนั้น เป็นผลดีกว่าวงจรเบื้องต้นทั่วไป

บทวิจารณ์

เพื่อความปลอดภัยมากยิ่งขึ้นแล้ว การเข้ารหัสดีอีเอส ในสมัยนี้สามารถถอดหาคีย์ได้อย่างรวดเร็ว ถ้าต้องการความปลอดภัยสูงยิ่งขึ้นควรจะใช้ วิธีการ เข้ารหัส 2 หรือ 3 ชั้น ในอัลกอริทึมของดีอีเอส ส่วนซอฟต์แวร์นั้นเป็นตัวอย่างง่ายๆ ถ้าต้องการให้ยากยิ่งยิ่งขึ้นไป ควรจะเข้ารหัสข้อมูลขนาดมากกว่านี้ นี่เป็นตัวอย่างเบื้องต้นให้กับผู้ที่คิดจะออกแบบและ นำไปใช้ นำไปปรับปรุงเพื่อให้ใช้ได้มีประสิทธิภาพสูงขึ้น ในการออกแบบการทำงานกับยูเอสบี ควรใช้ดูความเร็วของโหมดการทำงานด้วย อาจจะเกิดสัญญาณรบกวนได้ง่าย ในความเร็วสูงๆ การมีตัวเก็บประจุจะช่วยให้กระแสเดินเรียบขึ้น เป็นข้อควรระวังที่สำคัญของการออกแบบวงจรติดต่อกับยูเอสบี

ภาคผนวก

Spartan and SpartanXL Families

Field Programmable Gate Arraye

FPGA ในตระกูล Spartan และ SpartanXL นี้ จัดเป็นวงจรรวมเฉพาะกิจชนิดหนึ่งที่สามารถโปรแกรมเป็นวงจรดิจิตอลใดๆ ก็ได้ ซึ่งโปรแกรมวงจรลงบนสแตติกแรมภายในด้วยข้อมูลที่อยู่ภายนอก และสามารถโปรแกรมใหม่ได้โดยการรีเซ็ตด้วยสัญญาณไฟฟ้า นอกจากนี้ยังประหยัดไฟ โดยเฉพาะ ตระกูล SpartanXL นั้นใช้ไฟเลี้ยงแค่ 3.3 โวลต์ ส่วนตระกูล Spartan ใช้ไฟเลี้ยง 5.0 โวลต์ และมีความสามารถจุวงจรที่มีขนาดใหญ่ๆ (จำนวนเกทมากๆ)ได้อีกด้วย

คุณลักษณะเฉพาะ

-เป็น FPGA ที่มี RAM อยู่ในตัว chip ด้วย

-มีลักษณะทางสถาปัตยกรรมคล้ายๆ กับ XC4000

-มีความเร็วในการทำงาน 80 MHz

-สามารถจะโปรแกรมใหม่ได้โดยไม่จำกัดจำนวนครั้ง

-มีความจุสูงสุดถึง 1862 logic cells หรือ 40,000 gates

-มี cell ขนาด 0.35µm/0.50µm

-ต้องใช้ไฟเลี้ยง 3.0 โวลต์ในตระกูล XL และ 3.0 โวลต์ในตระกูลธรรมดา

โครงสร้างภายใน

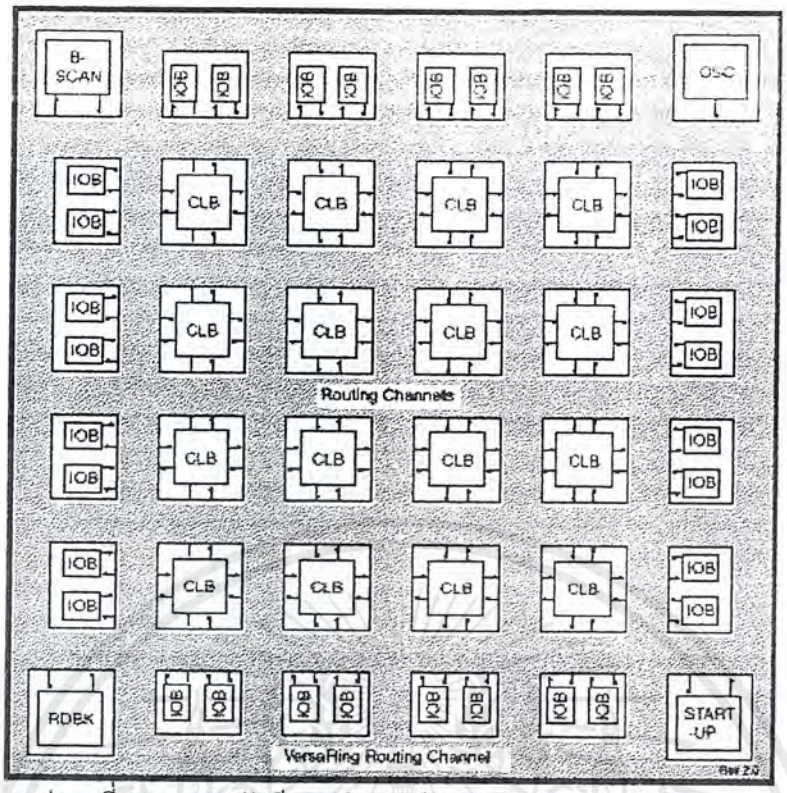
วงจรรวมเฉพาะกิจชนิดนี้ผลิตโดยบริษัท Xilinx สร้างเป็นอะเรย์ที่ประกอบด้วยเกทจำนวนสูงได้ถึง 40,000 เกท ดังตารางที่ 1

Device	Logic Cells	Max System Gates	Typical Gate Range (Logic and RAM)*	CLB Matrix	Total CLBs	Number of Flip-Flops	Max. Available User I/O
XCS05 & XCS05XL	238	5,000	2,000 - 5,000	10 x 10	100	360	77
XCS10 & XCS10XL	466	10,000	3,000 - 10,000	14 x 14	196	616	112
XCS20 & XCS20XL	950	20,000	7,000 - 20,000	20 x 20	400	1,120	160
XCS30 & XCS30XL	1368	30,000	10,000 - 30,000	24 x 24	576	1,536	192
XCS40 & XCS40XL	1862	40,000	13,000 - 40,000	28 x 28	784	2,016	205

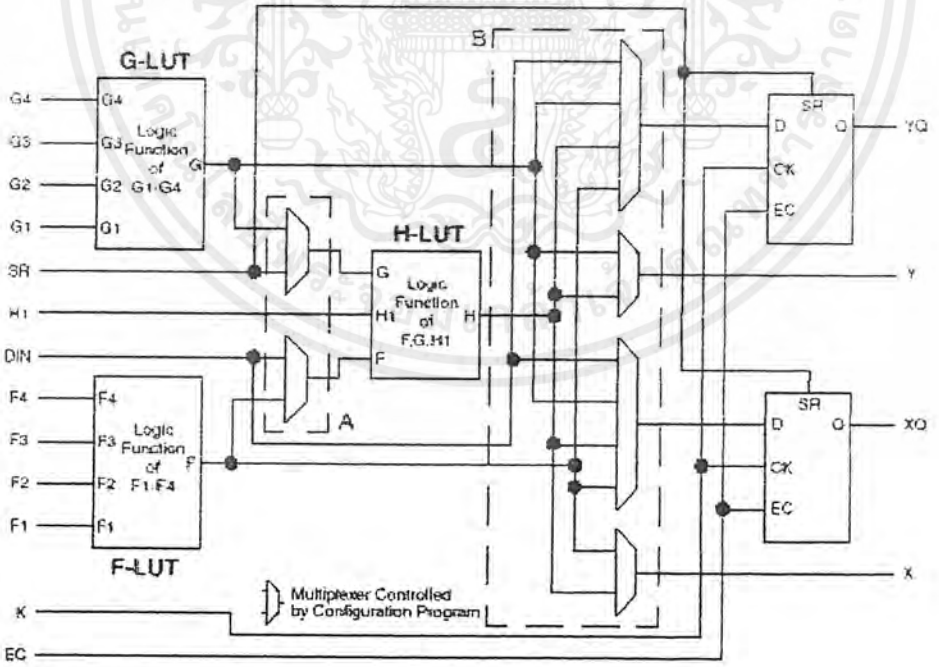
* Max values of Typical Gate Range include 20-30% of CLBs used as RAM.

ตารางที่1 แสดงคุณสมบัติต่างๆ ของ FPGA ตระกูล Spartan และ SpartanXL

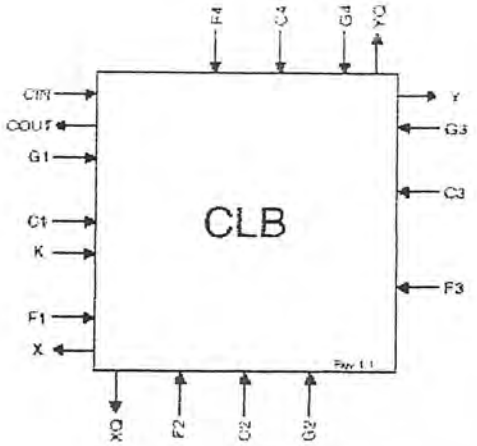
FPGA มีโครงสร้างภายในใกล้เคียงกับสถาปัตยกรรมของเกทอะเรย์มากสามารถโปรแกรมและลบ Configuration สเตติกแรมภายในได้โดยกระแสไฟฟ้า ซึ่งจะทำการโปรแกรมได้โดยดึงข้อมูลฐานสิบหกมาจากภายนอก ภายใน FPGA จัดเรียงเป็นลอจิกเซลล์ ลักษณะเป็นเมทริกซ์ ล้อมรอบภายนอกด้วยอินพุตเอ้าพุตเซลล์ดังรูปภาพที่ 1 แต่ละเซลล์เรียกว่า CLB (Configurable Logoc Block) โครงสร้างภายในของ CLB มีลักษณะดังรูปที่ 2 การเชื่อมต่อสัญญาณของ CLB เป็นดังรูปที่ 3 และอินพุตเอ้าพุตที่ล้อมรอบอยู่เรียกว่า IOB (Input/Output Block) มีลักษณะโครงสร้างภายในดังรูปที่ 4



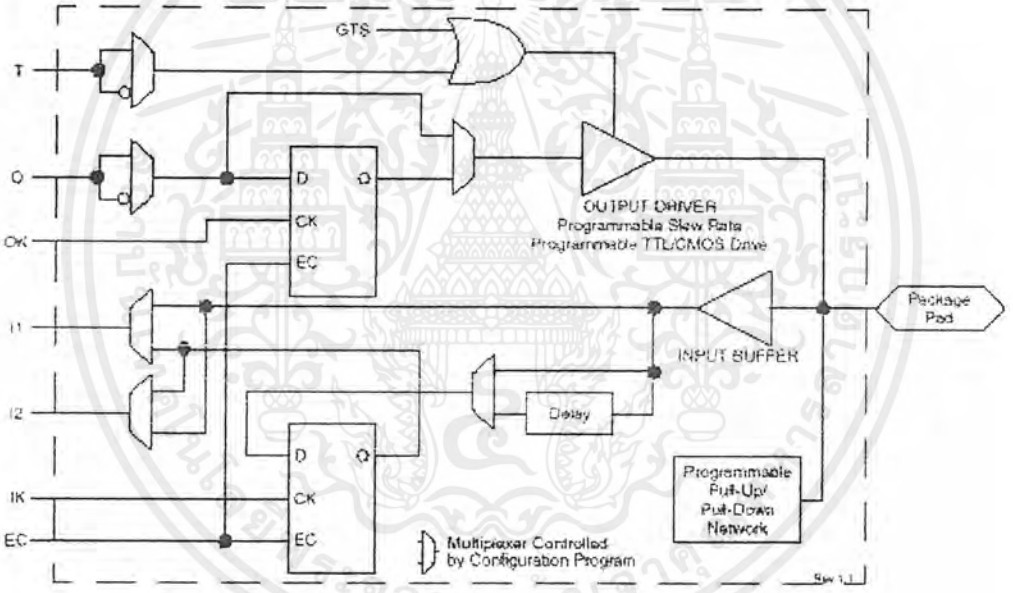
รูปภาพที่ 1 แสดงการจัดเรียงลอจิกเซลล์ของ FPGA ตระกูล Spartan และ SpartanXL



รูปภาพที่ 2 แสดงถึงลักษณะโครงสร้างภายในของ CLB



รูปที่3 แสดงถึงการเชื่อมต่อสัญญาณของCLB



รูปภาพที่4 แสดงถึงลักษณะโครงสร้างภายในของ IOB

รายละเอียดของขา FPGA **ตระกูล** Spartan **และ** SpartanXL

หน้าที่ของแต่ละขามีดังตารางที่ 2 และตำแหน่งและชื่อของขาดังตารางที่ 3(XCS05/XL)

ตารางที่2 สัญลักษณ์และรายละเอียดของขา

ชื่อขา	รายละเอียดทั่วๆ ไป
VCC	มีอยู่ 8 ขา หรือมากกว่านี้(ขึ้นอยู่กับชนิดของตัวถัง) ทุกขาต้องต่อไฟ +5.0 โวลต์หรือ +3.0 โวลต์ (ขึ้นอยู่กับว่าเป็นตระกูล spartan หรือ spartanXLX)
GND	มีอยู่ 8 ขา หรือมากกว่านี้(ขึ้นอยู่กับชนิดของตัวถัง) ทุกขาต้องต่อกับground
CCLK	เป็นสัญญาณคล็อกเอ้าพุตออกจากตัว FPGA เพื่อใช้โหลดคอนฟิกในโหมด Master แต่จะเป็นอินพุตในโหมด Slave
DONE	เป็นทั้งสัญญาณเอ้าพุตและอินพุตถ้าเป็นเอ้าพุตจะมีลอจิกเป็น "1" เมื่อโหลดคอนฟิกเสร็จสมบูรณ์ ถ้าเป็นอินพุตจะใช้โหลดคอนฟิกใหม่โดยการป้อนสัญญาณ "0" เข้าไป
$\overline{\text{PROGRAM}}$	ถ้าต้องการลบโปรแกรมเก่าออกต้องให้อินพุตเป็น "0" และต้องการโปรแกรมใหม่ต้องให้เอ้าพุตเป็น "1"
MODE	ถ้าต้องการให้เป็นโหมด slave ให้ปล่อยขานี้ไว้เฉยๆ ไม่ต้องต่อกับอะไรเลย ถ้าต้องการให้เป็นโหมด master ให้ต่อขานี้กับ ground
Don't Connect	เป็นขาสำหรับโรงงานทำการทดสอบตัว chip ให้ปล่อยไว้เฉยๆ
TDO	เมื่อเราทำการใช้งานโปรแกรม จะเป็นขาสำหรับทดสอบเอ้าพุตที่ออกมา
TDI,TCK,TMS	เป็นขาที่ใช้สำหรับทดสอบข้อมูลที่เข้ามา,ทดลอบสัญญาณคล็อกและทดลอบโหมดที่เลือกว่าถูกต้องหรือไม่
HDC	ระหว่างการโหลดคอนฟิกจะมีเอ้าพุตเป็น "1" เพื่อแสดงให้รู้ว่าการโหลดคอนฟิกยังไม่สมบูรณ์ หลังจากโหลดคอนฟิกเสร็จจะเป็นขา I/O
$\overline{\text{LDC}}$	ระหว่างการโหลดคอนฟิกจะมีเอ้าพุตเป็น "0" เพื่อแสดงให้รู้ว่าการโหลดคอนฟิกยังไม่สมบูรณ์ หลังจากโหลดคอนฟิกเสร็จจะเป็นขา I/O
INIT	เอ้าพุตแบบ open-drain แอคทีฟ "0" ใช้เคลียร์หน่วยความจำที่ทำการคอนฟิก และยังใช้กำหนดสถานะเพื่อใช้ในการโหลดคอนฟิก เมื่อโหลดเสร็จแล้วจะเป็นขา I/O

PGCK1-PGCK2	เป็น Primary Global ซึ่งเป็นขาอินพุตสำหรับการเชื่อมต่อภายใน ถ้าไม่ใช้สามารถทำให้เป็นขา I/O ได้
SGCK1-SGCK2	เป็น Secondary Global ซึ่งเป็นขาอินพุตสำหรับการเชื่อมต่อภายใน ถ้าไม่ใช้สามารถทำให้เป็นขา I/O ได้
DIN	ระหว่างการคอนฟิกจะเป็นขาอินพุต รับข้อมูลมาจาก CCLK หลังจากคอนฟิกเสร็จ จะทำหน้าที่เป็นขา I/O
DOUT	ระหว่างการตอนฟิกใช้เป็นเอ้าพุตของข้อมูลที่ทำการคอนฟิก เพื่อส่งต่อไปยัง FPGA ที่เป็น slave หลังจากโหลดคอนฟิกเสร็จจะเป็นขา I/O
I/O	เป็นได้ทั้งอินพุตและเอ้าพุตขึ้นอยู่กับการคอนฟิก

ตารางที่3 แสดงถึงตำแหน่งและชื่อของสัญญาณของXCS05/XL

PIN No.	XCS05/XL	PIN No	XCS05/XL
P1	GND	P2	VCC
P3	I/O	P4	I/O
P5	I/O	P6	I/O
P7	I/O	P8	I/O
P9	I/O	P10	I/O, SGCK1
P11	VCC	P12	GND
P13	I/O, PGCK1	P14	I/O
P15	I/O, TDI	P16	I/O, TCK
P17	I/O, TMS	P18	I/O
P19	I/O	P20	I/O
P21	GND	P22	VCC
P23	I/O	P24	I/O
P25	I/O	P26	I/O
P27	I/O	P28	I/O
P29	I/O, SCGK2	P30	Don't Connect
P31	GND	P32	MODE
P33	VCC	P34	Don't Connect
P35	I/O, PGCK2	P36	I/O (HDC)
P37	I/O ($\overline{\text{LDC}}$)	P38	I/O
P39	I/O	P40	I/O
P41	I/O ($\overline{\text{INIT}}$)	P42	VCC
P43	GND	P44	I/O
P45	I/O	P46	I/O

P47	I/O	P48	I/O
P49	I/O	P50	I/O
P51	I/O, SGCK3	P52	GND
P53	DONE	P54	VCC
P55	$\overline{\text{PROGRAM}}$	P56	I/O
P57	I/O, PGCK3	P58	I/O
P59	I/O	P60	I/O
P61	I/O	P62	I/O
P63	VCC	P64	GND
P65	I/O	P66	I/O
P67	I/O	P68	I/O
P69	I/O	P70	I/O
P71	I/O (DIN)	P72	I/O, SGCK4 (DOUT)
P73	CCLK	P74	VCC
P75	O, TDO	P76	GND
P77	I/O	P78	I/O, PGCK4
P79	I/O	P80	I/O
P81	I/O	P82	I/O
P83	I/O	P84	I/O

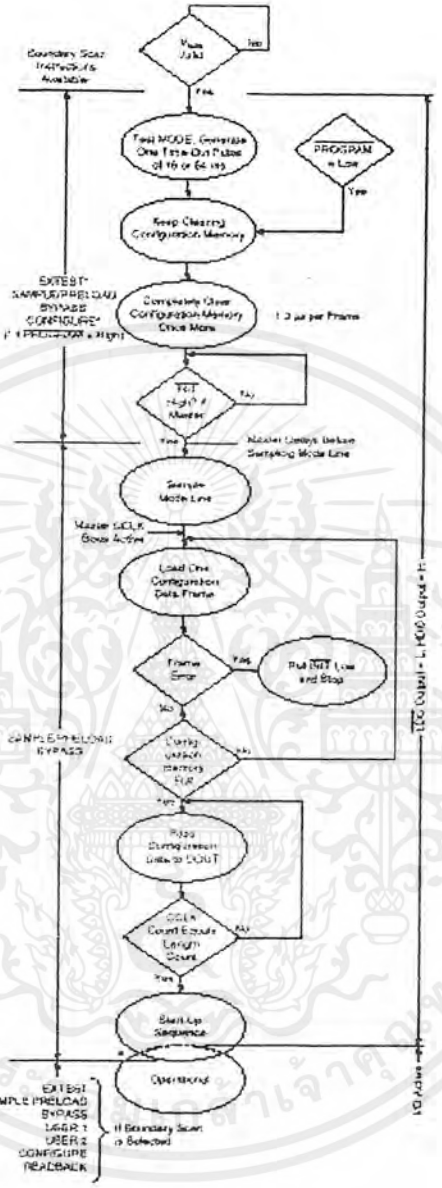
รายละเอียดการใช้งาน

ในการใช้งานของ FPGA นั้นมี 4 ขั้นตอนในการทำคอนฟิกดังนี้

1. เคลียร์คอนฟิกเดิม
2. การเตรียมก่อนทำคอนฟิก
3. การทำคอนฟิก

4. การโหลดคอนฟิก

ดังจะแสดงในรูปที่ 5



รูปภาพที่ 5 แสดงการทำงานของ FPGA ตามลำดับ

เคลียร์คอนฟิกเดิม

ต้องเซ็ตให้ขา $\overline{\text{PROGRAM}}$ เป็น 0 และสามารถทดสอบได้ที่ขา $\overline{\text{INIT}}$

การเตรียมก่อนทำคอนฟิก

จะต้องใช้ขา HDC, $\overline{\text{LDC}}$, $\overline{\text{INIT}}$ และ DONE โดยการเซ็ตให้ $\overline{\text{LDC}}$, $\overline{\text{INIT}}$ และ DONE ให้เป็น 0 และเซ็ต HDC ให้เป็น 1

การทำคอนฟิก

เซ็ต $\overline{DOUT}$ ให้เป็น 1 และทำการโหลดคอนฟิก โดยสัญญาณเป็นแบบ Open-Drain ขา INIT เป็น 0

การทำงานของโหมดต่าง ๆ

FPGA ในการที่จะเซ็ตโหมดนั้น เราต้อง

- MODE = 1 คือ การเซ็ตให้เป็นโหมด Slave
- MODE = 0 คือ การเซ็ตให้เป็นโหมด Master

ในการทำคอนฟิกนั้นจะมีขาที่ใช้ประกอบกับการเซ็ตโหมด ดังตารางที่ 4

CONFIGURATION MODE <MODE Pin>		
SLAVE SERIAL <High>	MASTER SERIAL <Low>	USER OPERATION
MODE (I)	MODE (I)	MODE
HDC (HIGH)	HDC (HIGH)	I/O
$\overline{LDC}$ (LOW)	$\overline{LDC}$ (LOW)	I/O
$\overline{INIT}$	$\overline{INIT}$	I/O
DONE	DONE	DONE
PROGRAM (I)	PROGRAM (I)	PROGRAM
CCLK (I)	CCLK (O)	CCLK (I)
DIN (I)	DIN (I)	I/O
DOUT	DOUT	SGCK4-I/O
TDI	TDI	TDI-I/O
TCK	TCK	TCK-I/O
TMS	TMS	TMS-I/O
TDO	TDO	TDO (O)
		ALL OTHERS

Notes: 1. A shaded table cell represents the internal pull-up used before and during configuration.
2. (I) represents an input. (O) represents an output.
3. $\overline{INIT}$ is an open-drain output during configuration.

ตารางที่ 4 แสดงถึงขาและการเซ็ตระหว่างการทำคอนฟิก

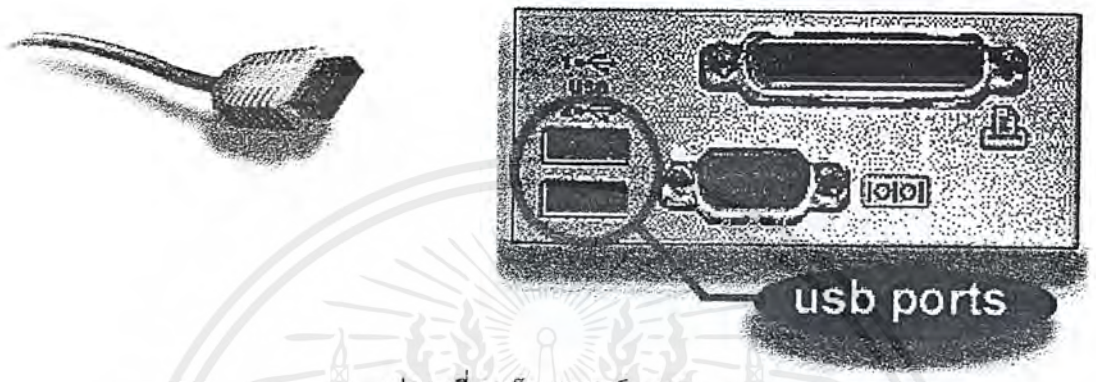
ข้อควรระวังในการใช้ FPGA

สิ่งแรกที่สำคัญคือ เป็น IC ที่กลัวความร้อนที่สุด การบัดกรีโดยหัวแร้งกำลังสูง หรือบัดกรีที่หัวแร้งที่ขา IC นาน ๆ จะทำให้ IC เสียหายง่าย ระยะเวลาในการบัดกรี 1 จุด ไม่ควรเกิน 5-10 วินาที ควรใช้ ซ็อคเก็ต IC ในการประกอบลงบนแผ่นปริ้นต์

การป้อนแหล่งจ่ายไฟให้ IC ไม่ควรผิดขั้ว ถ้าสลับขั้วจะทำให้ IC เสียได้ นอกจากนั้นแรงดันของแหล่งจ่ายไฟต้องอยู่ในช่วงที่โรงงานกำหนดมา สำหรับตระกูล Spartan แรงดันที่ IC อยู่ในช่วงทำงานได้ VCC = 4.75-5.25 โวลต์ และตระกูล SpartanXL แรงดัน IC อยู่ในช่วงที่ทำงานได้ VCC = 3.05-3.55 โวลต์ ดังนั้นก่อนป้อนแรงดันควรตรวจเช็คให้แน่ใจก่อน

Universal Serial Bus (USB)

USB เป็นมาตรฐานของที่ใช้รับส่งข้อมูลระหว่าง ๆ อุปกรณ์ต่าง ๆ ของพีซี ลักษณะเดียวกับบัสที่เชื่อมต่อระหว่างสล็อตต่าง ๆ บนเมนบอร์ดของพีซีบัส USB สามารถเชื่อมต่อกับอุปกรณ์ได้สูงสุดถึง 127 อุปกรณ์ โดยที่ไม่มีปัญหาเรื่อง IRQ หรือ DMA ไม่พอให้ใช้งาน เหมือนกับกรณีของการใส่การ์ดเพิ่มเข้าไปในเครื่องพีซี เพราะ USB จะมีกลไกที่ทำให้อุปกรณ์ทั้ง 127 ตัว ใช้ทรัพยากรร่วมกันได้ โดยไม่มีการยึดทรัพยากรที่มีจำกัดอย่าง IRQ ไปไว้ที่อุปกรณ์ตัวใดตัวหนึ่ง



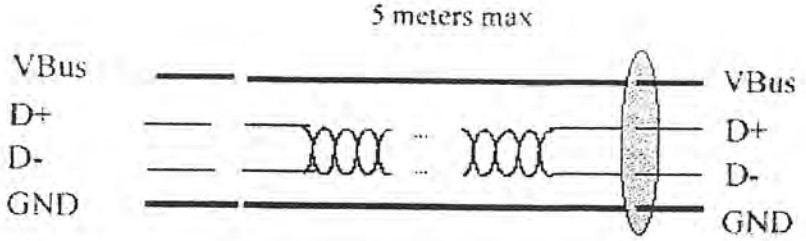
รูปภาพที่ 1 แจ็กและพอร์ต USB

ลักษณะที่สำคัญของ USB

- ง่ายต่อการใช้เพราะมีความสามารถที่เรียกว่า Plug and Play
- อุปกรณ์แบบ USB ทั้งหมดจะมีความสามารถที่เรียกว่า Hot Swapping ซึ่งทำให้สามารถที่จะต่ออุปกรณ์ต่าง ๆเข้ากับพอร์ต USB ได้ทันทีโดยไม่จำเป็นต้องปิดเครื่องก่อน และพีซีก็จะรับรู้ทันทีว่ามีอุปกรณ์ชิ้นใหม่ต่อเข้ามา โดยไม่ต้องรีสตาร์ทวินโดว์ใหม่ และจะทำการติดตั้งโปรแกรมต่าง ๆ ที่จำเป็นให้โดยอัตโนมัติ ในการถอดอุปกรณ์ออกจากระบบก็ทำได้ตลอดเวลาเช่นเดียวกัน โดยจะไม่ทำให้เครื่องเสียหายหรือโปรแกรมหยุดทำงาน
- USB สามารถเชื่อมต่ออุปกรณ์ได้สูงสุดถึง 127 อุปกรณ์ โดยใช้อุปกรณ์ขยายพอร์ตที่เรียกว่า HUB มาต่อเข้ากับบัส
- มีความเร็ว 2 ระดับคือ Low Speed (1.5 Mb/s) และ High Speed (12 Mb/s)
- บัส USB สามารถจ่ายกระแสไฟฟ้าให้อุปกรณ์ต่าง ๆ ที่ต่ออยู่ได้ โดยจ่ายกระแส 100 มิลลิแอมป์ สำหรับอุปกรณ์ความเร็วต่ำ (1.5 เมกกะบิต/วินาที) และ 500 มิลลิแอมป์สำหรับอุปกรณ์ความเร็วสูง (12 เมกกะบิต/วินาที)
- สนับสนุนการส่งข้อมูลทั้งแบบ Isochronous และแบบ Asynchronous
- USB มีความสามารถในการส่งข้อมูลแบบ 2 ทิศทาง

โครงสร้างของ USB

ภายในสายเคเบิ้ลของ USB จะมีสายไฟ 4 เส้น โดย 2 เส้นจะใช้ในการส่งข้อมูลและอีก 2 เส้นใช้ในการจ่ายไฟเลี้ยงอุปกรณ์ต่าง ๆ ที่ต่อกับบัสดังภาพ



รูปภาพที่ 1 แสดงถึงโครงสร้างภายในของบัส USB

-เส้น VBus และเส้น GND เป็นเส้นที่ใช้ในการจ่ายไฟเลี้ยงให้กับอุปกรณ์ต่างๆ ที่ต่อกับบัส

-เส้น D+ และ D- เป็นเส้นที่ใช้ในการรับ-ส่งข้อมูล โดยถ้าเราต้องการความเร็วสูง (12 Mb/s) เราจะใช้เส้น D+ ในการส่งข้อมูล และถ้าต้องการความเร็วต่ำ(1.5 Mb/s) จะใช้เส้นD- ในการส่งข้อมูล

ตัวอย่างอุปกรณ์ที่ใช้ต่อพ่วง

- ลำโพงแบบ USB นี้ไม่ต้องใช้ซาวด์การ์ดแบบที่เสียบในเครื่อง เมื่อต่อลำโพงเข้ากับพอร์ต USB จะทำให้ข้อมูลเสียงที่เป็นสัญญาณดิจิตอล ส่งมาตามสายเคเบิ้ลจนถึงลำโพง แล้วจึงถูกวงจรแปลงสัญญาณที่ลำโพงแปลงจากดิจิตอลเป็นอนาล็อค แล้วเข้าภาคขยายเป็นเสียง
- ลำโพงแบบ USB เท่าที่เห็นตอนนี้มีของ US Robotics 56K มาตรฐาน V.90
- กล้องดิจิตอล ปัจจุบันมักจะต่อกับพีซีทางพอร์ต หรือไมก็พอร์ตอนุกรมซึ่งต้องใช้เวลานานในการโอนข้อมูลภาพจากกล้องเข้าสู่พีซี แต่ด้วยกล้องรุ่นใหม่ที่ต่อกับพอร์ต USB คุณจะโอนข้อมูลเร็วขึ้นกว่าเดิม 10 ถึง 100 เท่า
- คีย์บอร์ดแบบ USB อุปกรณ์ตัวนี้ตัวมันเองก็มีฮับที่ใช้ต่อกับเมาส์ และจอยสติ๊กได้ทันที

Source code

1 โปรแกรมส่วนที่ 1

1.1 ส่วนเชื่อมต่อกับผู้ใช้ ส่วนหน้าต่างหลัก

```
#include <vcl\vcl.h>
#include <string.h>
#pragma hdrstop
#include "ApplicationLock.h"
#include "Config.h"
#include "EnterPassWord.h"
#include "ReadFileEnc.cpp"
#pragma resource "*.dfm"
TMainWindows *MainWindows;
__fastcall TMainWindows::TMainWindows(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TMainWindows::OpenFileMenuClick(TObject *Sender)
{
 TFileName NameOfFile;
 OpenFileDialog -> Filter = "Executable files|*.EXE;*.COM";
 if (OpenFileDialog -> Execute())
 {
        NameOfFile = OpenFileDialog->FileName;
  StatusBarMain->SimpleText=NameOfFile;
  EncryptButton->Enabled = true;
 }
}
void __fastcall TMainWindows::ExitMenuClick(TObject *Sender)
{
  MainWindows->Close();
  EnterPassWindows->Close();
}
void __fastcall TMainWindows::EncryptButtonClick(TObject *Sender)
{
  int L;
```

```
    char Name[50]={'\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0',
                   '\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0',
                   '\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0',
                   '\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0',
                   '\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0',
                   '\0','\0'};
    String FileName;
    FileName=StatusBarMain->SimpleText;
    L=FileName.Length();
    for(int i=1;i<L+2;i++)
    Name[i-1]=FileName[i];
    EncryptionFile En;
    En.EncryptFile(Name);
    StatusBarMain->SimpleText="Complete";
    EncryptButton->Enabled=false;
}
void __fastcall TMainWindows::ConfigurationMenuClick(TObject *Sender)
{
        ConfigurationWindows->Show();
}
void __fastcall TMainWindows::MainWindowDestroy(TObject *Sender)
{
        EnterPassWindows->Close();
}
void __fastcall TMainWindows::MainWindowClose(TObject *Sender,
        TCloseAction &Action)
{
   EnterPassWindows->Close();
}
void __fastcall TMainWindows::CancelButtonClick(TObject *Sender)
{
        EncryptButton->Enabled=false;
   StatusBarMain->SimpleText="";
```

1.2 ส่วนเชื่อมต่อกับผู้ใช้ หน้าต่างใส่รหัสผ่าน

```
#include <vcl\vcl.h>
#pragma hdrstop
#include "ApplicationLock.h"
#include "EnterPassWord.h"
#include "ReadiniFile.cpp"
#pragma resource "*.dfm"
TEnterPassWindows *EnterPassWindows;
__fastcall TEnterPassWindows::TEnterPassWindows(TComponent* Owner)
        : TForm(Owner)
{
        Password->PasswordChar='*';
        Password->MaxLength=7;
}
void __fastcall TEnterPassWindows::OkPassButtonClick(TObject *Sender)
{
        String Passwrd;
  ReadiniFile PasswordCheck;
  PasswordCheck.RdiniFile("Password.cfg","[password]");
  Passwrd=Password->Text;
  if (Passwrd==PasswordCheck.Value)
  {
    EnterPassWindows->Hide();
        MainWindows->Show();
  }else
  {
    Password->Text="";
        StatusCheck->SimpleText="Incorrect password";
  }
}
void __fastcall TEnterPassWindows::CancelPassButtonClick(TObject *Sender)
{
        Password->Text="";
}
```

1.3 ส่วนเชื่อมต่อกับผู้ใช้ หน้าต่างเปลี่ยนรหัสผ่าน

```
#include <vcl\vcl.h>
#pragma hdrstop
#include "ApplicationLock.h"
#include "Config.h"
#include "ReadiniFile.cpp"
#include "WrteiniFile.cpp"
#pragma resource "*.dfm"
TConfigurationWindows *ConfigurationWindows;
__fastcall TConfigurationWindows::TConfigurationWindows(TComponent* Owner)
        : TForm(Owner)
{
 OldPass->PasswordChar='*';
 OldPass->MaxLength=7;
 NewPass->PasswordChar='*';
 NewPass->MaxLength=7;
 ConfirmPass->PasswordChar='*';
 ConfirmPass->MaxLength=7;
}
void __fastcall TConfigurationWindows::ResetPasswordClick(TObject *Sender)
{
    OldPass->Text="";
    NewPass->Text="";
    ConfirmPass->Text="";
    ConfigurationWindows->Close();
}
void __fastcall TConfigurationWindows::OkPasswordClick(TObject *Sender)
{
 String OldPassText,NewPassText,ConfirmPassText;
 ReadiniFile PasswordOld;
 ChangeiniFile WritePassword;
    PasswordOld.RdiniFile("Password.cfg","[password]");
    NewPassText = NewPass->Text;
    ConfirmPassText = ConfirmPass->Text;
```

```
    OldPassText = OldPass->Text;
    if ((NewPassText==ConfirmPassText)&&(OldPassText==PasswordOld.Value))
    {
      WritePassword.NewValue=NewPassText;
      WritePassword.WrtiniFile("Password.cfg","[password]");
          OldPass->Text="";
          NewPass->Text="";
          ConfirmPass->Text="";
          ConfigurationWindows->Close();
    }
    else
    {
          OldPass->Text="";
          NewPass->Text="";
          ConfirmPass->Text="";
          StatusBar->SimpleText="Error Password";
    }
}
```

1.4 ส่วนอ่านและเขียนไฟล์รหัสผ่าน

```
#include <fstream.h>
#include <string.h>
class ReadiniFile
{
  public:
  char Value[];

  public:
        void RdiniFile(char[],char[]);
};
void ReadiniFile::RdiniFile(char FileName[],char Segment[])
{
        const MaxChar=12;
   char iniLine[MaxChar];
        int testSegment;
        unsigned int i;
        ifstream ReadFile(FileName);

        testSegment=0;
        while(ReadFile)
        {
        ReadFile.getline(iniLine,MaxChar);
    for(i=0;i<strlen(Segment)+1;i++)
    {
        if (Segment[i]!=iniLine[i])
      {
        testSegment=1;
       i=strlen(Segment)+1;
      }
    }
    if(testSegment==0)
    {
        ReadFile.getline(iniLine,MaxChar);
```

```
        strcpy(Value,iniLine);
      }
    }
  }
}
#include <fstream.h>
#include <string.h>
#include <vcl\vcl.h>
class ChangeiniFile
{
  public:
        String NewValue;
  public:
  void WrtiniFile(char[],char[]);
};
void ChangeiniFile::WrtiniFile(char FileName[],char Segment[])
{

  ofstream WriteFile(FileName);
  WriteFile<<Segment;
  WriteFile<<"\n";
  WriteFile<<NewValue;

}
```

1.5 ส่วนการเข้ารหัสและถอดรหัส

```
#include <malloc.h>
#include <fcntl.h>
#include <io.h>
#include <iostream.h>
#include <process.h>
#include <share.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


class DES
{
  public:
        void Encypt(unsigned int[]);
    void Decypt(unsigned int[]);
};
void transpose(unsigned int data[],int t[],int s)
{

        unsigned int Tr[64];
//--------------------------
        for(int n=0;n<64;n++)
    Tr[n]=data[n];
//--------------------------
        for(int i=0;i<s;i++)
        data[i]=Tr[t[i]];
}
void  rotate(unsigned int key[])
{
        unsigned int k[64];
```

```
//--------------------------
  for(int n=0;n<56;n++)
  k[n]=key[n];
//--------------------------
  for (int i=0;i<55;i++) k[i]=k[i+1];
  k[27]=key[0]; k[55]=key[28];
  for(int i=0;i<56;i++)
  key[i]=k[i];
}
void f(int i,unsigned int key[],unsigned int a[],unsigned int x[])
{
    int KeyTr2[48]          ={13,16,10,23,0,4,2,27,14,5,20,9,22,18,11,3,25,7,15,6,
                              26,19,12,1,40,51,30,36,46,54,29,39,50,44,32,47,43,48,
                              38,55,33,52,45,41,49,35,28,31};

    int etr[48]             ={31,0,1,2,3,4,3,4,5,6,7,8,7,8,9,10,11,12,11,12,13,14,
                              15,16,15,16,17,18,19,20,19,20,21,22,23,24,23,24,25,26,
                              27,28,27,28,29,30,31,0};

    int ptr[32]             ={15,6,19,20,28,11,27,16,0,14,22,25,4,17,30,9,1,7,23,13,
                              31,26,2,8,18,12,29,5,21,10,3,24};

    int s[8][64]            ={{14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,0,15,7,4,14,2,13,
                              1,10,6,12,11,9,5,3,8,4,1,14,8,13,6,2,11,15,12,9,7,3,10,
                              5,0,15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13},
                             {15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,3,13,4,7,15,2,8,
                              14,12,0,1,10,6,9,11,5,0,14,7,11,10,4,13,1,5,8,12,6,9,3,
                              2,15,13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9},
                             {10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,13,7,0,9,3,4,6,
                              10,2,8,5,14,12,11,15,1,13,6,4,9,8,15,3,0,11,1,2,12,5,
                              10,14,7,1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12},
                             {7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,13,8,11,5,6,15,
                              0,3,4,7,2,12,1,10,14,9,10,6,9,0,12,11,7,13,15,1,3,14,5,
                              2,8,4,3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14},
                             {2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,14,11,2,12,4,7,
```

```
                    13,1,5,0,15,10,3,9,8,6,4,2,1,11,10,13,7,8,15,9,12,5,6,
                    3,0,14,11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3},
                   {12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,10,15,4,2,7,12,
                    9,5,6,1,13,14,0,11,3,8,9,14,15,5,2,8,12,3,7,0,4,10,1,
                    13,11,6,4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13},
                   {4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,13,0,11,7,4,9,1,
                    10,14,3,5,12,2,15,8,6,1,4,11,13,12,3,7,14,10,15,6,8,0,
                    5,9,2,6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12},
                   {13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,1,15,13,8,10,3,
                    7,4,12,5,6,11,0,14,9,2,7,11,4,1,9,12,14,2,0,6,10,13,15,
                    3,5,8,2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11}};

  int rots[16]              ={1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};
  unsigned int e[64];
  unsigned int ikey[64];
  unsigned int y[64];
  int r,T;
  div_t D;
//------------------------
  for(int n=0;n<64;n++)
  e[n]=a[n];
//------------------------
        transpose(e,etr,48);
  for(int j=0;j<rots[i];j++) rotate(key);
//------------------------
        for(int n=0;n<64;n++)
  ikey[n]=key[n];
//------------------------
  transpose(ikey,KeyTr2,48);
        for(int j=0;j<48;j++)
  if(e[j]+ikey[j]==1) y[j]=1; else y[j]=0;

  for(int k=1;k<9;k++)
  {
   r=32*y[6*k-6]+16*y[6*k-1]+8*y[6*k-5]+4*y[6*k-4]+2*y[6*k-3]+y[6*k-2];
```

```
    D=div(s[k-1][r],8);
    T=D.quot;
    if (T%2!=0) x[4*k-4]=1; else x[4*k-4]=0;
    D=div(s[k-1][r],4);
    T=D.quot;
    if (T%2!=0) x[4*k-3]=1; else x[4*k-3]=0;
    D=div(s[k-1][r],2);
    T=D.quot;
    if (T%2!=0) x[4*k-2]=1; else x[4*k-2]=0;
    T=s[k-1][r];
    if (T%2!=0) x[4*k-1]=1; else x[4*k-1]=0;
  }

  transpose(x,ptr,32);
}
void  rotateb(unsigned int key[])
{
        unsigned int k[64];
//-------------------------
  for(int n=0;n<56;n++)
  k[n]=key[n];
//-------------------------
  for (int i=55;i>0;i--) k[i]=k[i-1];
  k[0]=key[27]; k[28]=key[55];
  for(int i=0;i<56;i++)
  key[i]=k[i];
}
void fd(int i,unsigned int key[],unsigned int a[],unsigned int x[])
{
    int KeyTr2[48]          ={13,16,10,23,0,4,2,27,14,5,20,9,22,18,11,3,25,7,15,6,
                              26,19,12,1,40,51,30,36,46,54,29,39,50,44,32,47,43,48,
                              38,55,33,52,45,41,49,35,28,31};

    int etr[48]             ={31,0,1,2,3,4,3,4,5,6,7,8,7,8,9,10,11,12,11,12,13,14,
                              15,16,15,16,17,18,19,20,19,20,21,22,23,24,23,24,25,26,
```

```
                27,28,27,28,29,30,31,0};

int ptr[32]     ={15,6,19,20,28,11,27,16,0,14,22,25,4,17,30,9,1,7,23,13,
                31,26,2,8,18,12,29,5,21,10,3,24};

int s[8][64]    ={{14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,0,15,7,4,14,2,13,
                1,10,6,12,11,9,5,3,8,4,1,14,8,13,6,2,11,15,12,9,7,3,10,
                5,0,15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13},
                {15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,3,13,4,7,15,2,8,
                14,12,0,1,10,6,9,11,5,0,14,7,11,10,4,13,1,5,8,12,6,9,3,
                2,15,13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9},
                {10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,13,7,0,9,3,4,6,
                10,2,8,5,14,12,11,15,1,13,6,4,9,8,15,3,0,11,1,2,12,
                10,14,7,1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12},
                {7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,13,8,11,5,6,15,
                0,3,4,7,2,12,1,10,14,9,10,6,9,0,12,11,7,13,15,1,3,14,5,
                2,8,4,3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14},
                {2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,14,11,2,12,4,7,
                13,1,5,0,15,10,3,9,8,6,4,2,1,11,10,13,7,8,15,9,12,5,6,
                3,0,14,11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3},
                {12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,10,15,4,2,7,12,
                9,5,6,1,13,14,0,11,3,8,9,14,15,5,2,8,12,3,7,0,4,10,1,
                13,11,6,4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13},
                {4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,13,0,11,7,4,9,1,
                10,14,3,5,12,2,15,8,6,1,4,11,13,12,3,7,14,10,15,6,8,0,
                5,9,2,6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12},
                {13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,1,15,13,8,10,3,
                7,4,12,5,6,11,0,14,9,2,7,11,4,1,9,12,14,2,0,6,10,13,15,
                3,5,8,2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11}};

int rots[16]    ={1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};
unsigned int e[64];
unsigned int ikey[64];
unsigned int y[64];
int r,T;
```

```
    div_t D;
  //------------------------
    for(int n=0;n<64;n++)
    e[n]=a[n];
  //------------------------
            transpose(e,etr,48);
    if (i<16) for(int j=0;j<rots[i];j++) rotateb(key);
  //------------------------
            for(int n=0;n<64;n++)
    ikey[n]=key[n];
  //------------------------
    transpose(ikey,KeyTr2,48);
            for(int j=0;j<48;j++)
    if(e[j]+ikey[j]==1) y[j]=1; else y[j]=0;

    for(int k=1;k<9;k++)
    {
     r=32*y[6*k-6]+16*y[6*k-1]+8*y[6*k-5]+4*y[6*k-4]+2*y[6*k-3]+y[6*k-2];
     D=div(s[k-1][r],8);
     T=D.quot;
     if (T%2!=0) x[4*k-4]=1; else x[4*k-4]=0;
     D=div(s[k-1][r],4);
     T=D.quot;
     if (T%2!=0) x[4*k-3]=1; else x[4*k-3]=0;
     D=div(s[k-1][r],2);
     T=D.quot;
     if (T%2!=0) x[4*k-2]=1; else x[4*k-2]=0;
     T=s[k-1][r];
     if (T%2!=0) x[4*k-1]=1; else x[4*k-1]=0;
    }
    transpose(x,ptr,32);
}
void DES::Encypt(unsigned int a[])
{
```

```
int InitialTr[64] ={57,49,41,33,25,17,9,1,59,51,43,35,27,19,11,3,61,
                    53,45,37,29,21,13,5,63,55,47,39,31,23,15,7,56,48,
                    40,32,24,16,8,0,58,50,42,34,26,18,10,2,60,52,44,36,
                    28,20,12,4,62,54,46,38,30,22,14,6};
I
nt FinalTr[64]  ={39,7,47,15,55,23,63,31,38,6,46,14,54,22,62,30,37,
                    5,45,13,53,21,61,29,36,4,44,12,52,20,60,28,35,3,43,
                    11,51,19,59,27,34,2,42,10,50,18,58,26,32,1,41,9,49,
                    17,57,25,32,0,40,8,48,16,56,24};

 int swap[64] ={32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,
                49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,0,1,
                2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,
                22,23,24,25,26,27,28,29,30,31};

int KeyTr1[56] ={56,48,40,32,24,16,8,0,57,49,41,33,25,17,9,1,58,50,
                42,34,26,18,10,2,59,51,43,35,62,54,46,38,30,22,14,6,
                61,53,45,37,29,21,13,5,60,52,44,36,28,20,12,4,27,19,
                11,3};

unsigned int key[64] ={1,1,0,1,0,1,0,0,0,1,1,0,0,1,0,1,1,0,0,1,1,1,1,0,0,1,0,1,
                       0,1,1,1,0,0,0,1,1,0,1,1,0,0,1,1,1,1,1,0,0,1,1,0,1,1,0,
                       0,1,1,1,1,0,1,0,1,1};

   unsigned int b[64];
   unsigned int x[64];
   transpose(a,InitialTr,64);
   transpose(key,KeyTr1,56);
   for(int i=0;i<16;i++)
   {
    for(int n=0;n<64;n++)
    b[n]=a[n];
    for(int j=0;j<32;j++)
    a[j]=b[j+32];
```

```
        f(i,key,a,x);

        for(int j=0;j<32;j++)
        if (b[j]+x[j]==1)
            a[j+32]=1;
        else
            a[j+32]=0;
    }
    transpose(a,swap,64);
    transpose(a,FinalTr,64);
}

void DES::Decypt(unsigned int a[])
{
    int InitialTr[64]={57,49,41,33,25,17,9,1,59,51,43,35,27,19,11,3,61,
                       53,45,37,29,21,13,5,63,55,47,39,31,23,15,7,56,48,
                       40,32,24,16,8,0,58,50,42,34,26,18,10,2,60,52,44,36,
                       28,20,12,4,62,54,46,38,30,22,14,6};

    int FinalTr[64] ={39,7,47,15,55,23,63,31,38,6,46,14,54,22,62,30,37,
                      5,45,13,53,21,61,29,36,4,44,12,52,20,60,28,35,3,43,
                      11,51,19,59,27,34,2,42,10,50,18,58,26,32,1,41,9,49,
                      17,57,25,32,0,40,8,48,16,56,24};

    int swap[64]    ={32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,
                      49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,0,1,
                      2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,
                      22,23,24,25,26,27,28,29,30,31};

    int KeyTr1[56]={56,48,40,32,24,16,8,0,57,49,41,33,25,17,9,1,58,50,
                    42,34,26,18,10,2,59,51,43,35,62,54,46,38,30,22,14,6,
                    61,53,45,37,29,21,13,5,60,52,44,36,28,20,12,4,27,19,
                    11,3};
```

```
    unsigned int key[64]= {1,1,0,1,0,1,0,0,0,1,1,0,0,1,0,1,1,0,0,1,1,1,0,0,1,0,1,
                           0,1,1,1,0,0,0,1,1,0,1,1,0,0,1,1,1,1,1,1,0,0,1,1,0,1,1,0,
                           0,1,1,1,1,1,0,1,0,1,1};
    unsigned int b[64];
    unsigned int x[64];
    transpose(a,InitialTr,64);
    transpose(key,KeyTr1,56);
    for(int i=16;i>0;i--)
    {
     for(int n=0;n<64;n++)
     b[n]=a[n];
     for(int j=0;j<32;j++)
     a[j]=b[j+32];

     fd(i,key,a,x);

     for(int j=0;j<32;j++)
     if (b[j]+x[j]==1)
         a[j+32]=1;
     else
         a[j+32]=0;
    }
    transpose(a,swap,64);
    transpose(a,FinalTr,64);
}
class EncryptionFile
{
        public:
    void EncryptFile(char[]);
    void DecryptFile();
};

void HextoBin(unsigned int Hex,unsigned int Bin[],int j)
{
   div_t D;
```

```
    for(int j=1;j<9;j++)
    {
          if (Hex%2!=0) Bin[8*j-i]=1; else Bin[8*j-i]=0;
      D=div(Hex,2);
      Hex=D.quot;
    }
}

unsigned int BintoHex(unsigned int Bin[],int j)
{
  unsigned int Hex;
        Hex=128*Bin[8*j-8]+64*Bin[8*j-7]+32*Bin[8*j-6]+16*Bin[8*j-5]+8*Bin[8*j-4]
    +4*Bin[8*j-3]+2*Bin[8*j-2]+Bin[8*j-1];
  return Hex;
}
void EncryptionFile::EncryptFile(char NameOfFile[])
{
  DES Encyption;
  FILE *FileWrite;
  fpos_t pos;
        int File,Data,redo,x,loop;
  unsigned int FileData[64],Temp1[64],Temp2[64];
  unsigned int *buf;
  char FileList[5][14]={"Encrypt00.dat","Encrypt01.dat","Encrypt02.dat"
                                    ,"Encrypt03.dat","Encrypt04.dat"};
  buf = (unsigned int *)malloc(1);
  *buf = 0;
  loop = 0;

  do
  {
  do
  {
    redo=0;
        File = _rtl_open(NameOfFile,SH_DENYRW);
```

```
        do
        {
        randomize();
        x=random(32);
        } while (x==0);
  for(int i=1;i<256*x+1;i++)
  Data =_rtl_read(File,buf,1);
  for(int i=1;i<8;i++)
  {
        Data=_rtl_read(File,buf,1);
        HextoBin(*buf,FileData,i);
  }
  HextoBin(x,FileData,8);

  for (int i=0;i<64;i++)
  Temp2[i]=FileData[i];

  Encyption.Encypt(FileData);
  for (int i=0;i<64;i++)
  Temp1[i]=FileData[i];
  Encyption.Decypt(Temp1);

  for (int i=1;i<64;i++)
  if (Temp1[i]!=Temp2[i]) redo=1;
        _rtl_close(File);
}while(redo==1);
FileWrite = _fsopen(NameOfFile,"r+",SH_DENYNO);
pos=256*x;
fsetpos(FileWrite,&pos);
fwrite("LockNow",7,1,FileWrite);
fclose(FileWrite);
File =  _rtl_creat(FileList[loop],0);
for (int i=1;i<9;i++)
{
```

```
        *buf = BintoHex(FileData,i);
        Data = _rtl_write(File,buf,1);
    }
    _rtl_close(File);
    loop++;
    } while (loop<5);
    free(buf);
    rename(NameOfFile,"File.dat");
}
void EncryptionFile::DecryptFile()
{
    DES Decyption;
    fpos_t pos;
    FILE *FileWrite;
            int File,Data,x,loop;
    unsigned int FileData[64];
    unsigned int *buf;
    char FileList[5][14]={"Encrypt00.dat","Encrypt01.dat","Encrypt02.dat"
                                        ,"Encrypt03.dat","Encrypt04.dat"};
    buf = (unsigned int *)malloc(1);
    *buf = 0;
    loop = 4;
    do
    {
            File = _rtl_open(FileList[loop],SH_DENYRW);
      for(int i=1;i<9;i++)
      {
            Data=_rtl_read(File,buf,1);
            HextoBin(*buf,FileData,i);
      }
      _rtl_close(File);
      Decyption.Decypt(FileData);
    x=BintoHex(FileData,8);
    FileWrite = _fsopen("File.dat","r+b",SH_DENYNO);
    pos=256*x;
```

```
    fsetpos(FileWrite,&pos);
    for (int i=1;i<8;i++)
    {
      *buf=BintoHex(FileData,i);
           fwrite(buf,1,1,FileWrite);
    }
    fclose(FileWrite);
    loop--;
    } while (loop>-1);
    free(buf);

}
```

2 โปรแกรมส่วนที่ 2 ที่ใช้ร่วมกับวงจร

2.1 ส่วนที่เชื่อมต่อกับผู้ใช้

```
#include <process.h>
#include <stdio.h>
#include <vcl\vcl.h>
#pragma hdrstop
#include "RunApp.h"
#include "ReadFileEnc.cpp"
#pragma resource "*.dfm"
TRunProg *RunProg;
__fastcall TRunProg::TRunProg(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TRunProg::RunButtonClick(TObject *Sender)
{
   int result;
   EncryptionFile DecrypttoRun;
   RunProg->Hide();
   DecrypttoRun.DecryptFile();
   result = spawnl(P_WAIT, "File.dat", "File.dat", NULL);
   DecrypttoRun.EncryptFile("File.dat");
   RunProg->Show();
}
```

2.2 ส่วนหลักในการ Execute โปรแกรม

```
#include <alloc.h>
#include <fcntl.h>
#include <io.h>
#include <iostream.h>
#include <process.h>
#include <share.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

class DES
{
  public:
        void Encypt(unsigned int[]);
    void Decypt(unsigned int[]);
};
void transpose(unsigned int data[],int t[],int s)
{

        unsigned int Tr[64];
//--------------------------
        for(int n=0;n<64;n++)
    Tr[n]=data[n];
//--------------------------
        for(int i=0;i<s;i++)
        data[i]=Tr[t[i]];
}
void  rotate(unsigned int key[])
{
        unsigned int k[64];
//-------------------------
  for(int n=0;n<56;n++)
  k[n]=key[n];
```

```
//--------------------------
  for (int i=0;i<55;i++) k[i]=k[i+1];
  k[27]=key[0]; k[55]=key[28];
  for(int i=0;i<56;i++)
  key[i]=k[i];
}
void f(int i,unsigned int key[],unsigned int a[],unsigned int x[])
{
    int KeyTr2[48]          ={13,16,10,23,0,4,2,27,14,5,20,9,22,18,11,3,25,7,15,6,
                              26,19,12,1,40,51,30,36,46,54,29,39,50,44,32,47,43,48,
                              38,55,33,52,45,41,49,35,28,31};
    int etr[48]             ={31,0,1,2,3,4,3,4,5,6,7,8,7,8,9,10,11,12,11,12,13,14,
                              15,16,15,16,17,18,19,20,19,20,21,22,23,24,23,24,25,26,
                              27,28,27,28,29,30,31,0};
    int ptr[32]             ={15,6,19,20,28,11,27,16,0,14,22,25,4,17,30,9,1,7,23,13,
                              31,26,2,8,18,12,29,5,21,10,3,24};
    int s[8][64]            ={{14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,0,15,7,4,14,2,13,
                              1,10,6,12,11,9,5,3,8,4,1,14,8,13,6,2,11,15,12,9,7,3,10,
                              5,0,15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13},
                              {15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,3,13,4,7,15,2,8,
                              14,12,0,1,10,6,9,11,5,0,14,7,11,10,4,13,1,5,8,12,6,9,3,
                              2,15,13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9},
                              {10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,13,7,0,9,3,4,6,
                              10,2,8,5,14,12,11,15,1,13,6,4,9,8,15,3,0,11,1,2,12,5,
                              10,14,7,1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12},
                              {7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,13,8,11,5,6,15,
                              0,3,4,7,2,12,1,10,14,9,10,6,9,0,12,11,7,13,15,1,3,14,5,
                              2,8,4,3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14},
                              {2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,14,11,2,12,4,7,
                              13,1,5,0,15,10,3,9,8,6,4,2,1,11,10,13,7,8,15,9,12,5,6,
                              3,0,14,11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3},
                              {12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,10,15,4,2,7,12,
                              9,5,6,1,13,14,0,11,3,8,9,14,15,5,2,8,12,3,7,0,4,10,1,
                              13,11,6,4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13},
```

```
                {4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,13,0,11,7,4,9,1,
                10,14,3,5,12,2,15,8,6,1,4,11,13,12,3,7,14,10,15,6,8,0,
                5,9,2,6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12},
                {13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,1,15,13,8,10,3,
                7,4,12,5,6,11,0,14,9,2,7,11,4,1,9,12,14,2,0,6,10,13,15,
                3,5,8,2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11}};

        int rots[16]    ={1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};
  unsigned int e[64];
  unsigned int ikey[64];
  unsigned int y[64];
  int r,T;
  div_t D;
//-----------------------
  for(int n=0;n<64;n++)
  e[n]=a[n];
//-----------------------
        transpose(e,etr,48);
  for(int j=0;j<rots[i];j++) rotate(key);
//-----------------------
        for(int n=0;n<64;n++)
  ikey[n]=key[n];
//-----------------------
  transpose(ikey,KeyTr2,48);
        for(int j=0;j<48;j++)
  if(e[j]+ikey[j]==1) y[j]=1; else y[j]=0;

  for(int k=1;k<9;k++)
  {
   r=32*y[6*k-6]+16*y[6*k-1]+8*y[6*k-5]+4*y[6*k-4]+2*y[6*k-3]+y[6*k-2];
   D=div(s[k-1][r],8);
   T=D.quot;
   if (T%2!=0) x[4*k-4]=1; else x[4*k-4]=0;
   D=div(s[k-1][r],4);
```

```
    T=D.quot;
    if (T%2!=0) x[4*k-3]=1; else x[4*k-3]=0;
    D=div(s[k-1][r],2);
    T=D.quot;
    if (T%2!=0) x[4*k-2]=1; else x[4*k-2]=0;
    T=s[k-1][r];
    if (T%2!=0) x[4*k-1]=1; else x[4*k-1]=0;
  }

  transpose(x,ptr,32);
}
void  rotateb(unsigned int key[])
{
        unsigned int k[64];
//-------------------------
  for(int n=0;n<56;n++)
  k[n]=key[n];
//-------------------------
  for (int i=55;i>0;i--) k[i]=k[i-1];
  k[0]=key[27]; k[28]=key[55];
  for(int i=0;i<56;i++)
  key[i]=k[i];
}
void fd(int i,unsigned int key[],unsigned int a[],unsigned int x[])
{
   int KeyTr2[48]          ={13,16,10,23,0,4,2,27,14,5,20,9,22,18,11,3,25,7,15,6,
                             26,19,12,1,40,51,30,36,46,54,29,39,50,44,32,47,43,48,
                             38,55,33,52,45,41,49,35,28,31};
   int etr[48]             ={31,0,1,2,3,4,3,4,5,6,7,8,7,8,9,10,11,12,11,12,13,14,
                             15,16,15,16,17,18,19,20,19,20,21,22,23,24,23,24,25,26,
                             27,28,27,28,29,30,31,0};
   int ptr[32]             ={15,6,19,20,28,11,27,16,0,14,22,25,4,17,30,9,1,7,23,13,
                             31,26,2,8,18,12,29,5,21,10,3,24};
   int s[8][64]            ={{14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,0,15,7,4,14,2,13,
                             1,10,6,12,11,9,5,3,8,4,1,14,8,13,6,2,11,15,12,9,7,3,10,
```

```
                5,0,15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13},
               {15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,3,13,4,7,15,2,8,
                14,12,0,1,10,6,9,11,5,0,14,7,11,10,4,13,1,5,8,12,6,9,3,
                2,15,13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9},
               {10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,13,7,0,9,3,4,6,
                10,2,8,5,14,12,11,15,1,13,6,4,9,8,15,3,0,11,1,2,12,5,
                10,14,7,1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12},
               {7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,13,8,11,5,6,15,
                0,3,4,7,2,12,1,10,14,9,10,6,9,0,12,11,7,13,15,1,3,14,5,
                2,8,4,3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14},
               {2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,14,11,2,12,4,7,
                13,1,5,0,15,10,3,9,8,6,4,2,1,11,10,13,7,8,15,9,12,5,6,
                3,0,14,11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3},
               {12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,10,15,4,2,7,12,
                9,5,6,1,13,14,0,11,3,8,9,14,15,5,2,8,12,3,7,0,4,10,1,
                13,11,6,4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13},
               {4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,13,0,11,7,4,9,1,
                10,14,3,5,12,2,15,8,6,1,4,11,13,12,3,7,14,10,15,6,8,0,
                5,9,2,6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12},
               {13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,1,15,13,8,10,3,
                7,4,12,5,6,11,0,14,9,2,7,11,4,1,9,12,14,2,0,6,10,13,15,
                3,5,8,2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11}};

    int rots[16]                ={1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};
    unsigned int e[64];
    unsigned int ikey[64];
    unsigned int y[64];
    int r,T;
    div_t D;
  //-----------------------
    for(int n=0;n<64;n++)
    e[n]=a[n];
  //-----------------------
          transpose(e,etr,48);
    if (i<16) for(int j=0;j<rots[i];j++) rotateb(key);
```

```
//-----------------------
            for(int n=0;n<64;n++)
   ikey[n]=key[n];
//-----------------------
   transpose(ikey,KeyTr2,48);
            for(int j=0;j<48;j++)
   if(e[j]+ikey[j]==1) y[j]=1; else y[j]=0;

   for(int k=1;k<9;k++)
   {
    r=32*y[6*k-6]+16*y[6*k-1]+8*y[6*k-5]+4*y[6*k-4]+2*y[6*k-3]+y[6*k-2];
    D=div(s[k-1][r],8);
    T=D.quot;
    if (T%2!=0) x[4*k-4]=1; else x[4*k-4]=0;
    D=div(s[k-1][r],4);
    T=D.quot;
    if (T%2!=0) x[4*k-3]=1; else x[4*k-3]=0;
    D=div(s[k-1][r],2);
    T=D.quot;
    if (T%2!=0) x[4*k-2]=1; else x[4*k-2]=0;
    T=s[k-1][r];
    if (T%2!=0) x[4*k-1]=1; else x[4*k-1]=0;
   }
   transpose(x,ptr,32);
}
void DES::Encypt(unsigned int a[])
{
    int InitialTr[64]={57,49,41,33,25,17,9,1,59,51,43,35,27,19,11,3,61,
                       53,45,37,29,21,13,5,63,55,47,39,31,23,15,7,56,48,
                       40,32,24,16,8,0,58,50,42,34,26,18,10,2,60,52,44,36,
                       28,20,12,4,62,54,46,38,30,22,14,6};

    int FinalTr[64] ={39,7,47,15,55,23,63,31,38,6,46,14,54,22,62,30,37,
                       5,45,13,53,21,61,29,36,4,44,12,52,20,60,28,35,3,43,
                       11,51,19,59,27,34,2,42,10,50,18,58,26,32,1,41,9,49,
```

```
                17,57,25,32,0,40,8,48,16,56,24};

int swap[64]    ={32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,
                49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,0,1,
                2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,
                22,23,24,25,26,27,28,29,30,31};

int KeyTr1[56]  ={56,48,40,32,24,16,8,0,57,49,41,33,25,17,9,1,58,50,
                42,34,26,18,10,2,59,51,43,35,62,54,46,38,30,22,14,6,
                61,53,45,37,29,21,13,5,60,52,44,36,28,20,12,4,27,19,
                11,3};


unsigned int key[64] ={1,1,0,1,0,1,0,0,0,1,1,0,0,1,0,1,1,0,0,1,1,1,0,0,1,0,1,
                       0,1,1,1,0,0,0,1,1,0,1,1,0,0,1,1,1,1,1,0,0,1,1,0,1,1,0,
                       0,1,1,1,1,0,1,0,1,1};

unsigned int b[64];
unsigned int x[64];
transpose(a,InitialTr,64);
transpose(key,KeyTr1,56);
for(int i=0;i<16;i++)
{
 for(int n=0;n<64;n++)
 b[n]=a[n];
 for(int j=0;j<32;j++)
 a[j]=b[j+32];

 f(i,key,a,x);

 for(int j=0;j<32;j++)
 if (b[j]+x[j]==1)
     a[j+32]=1;
 else
     a[j+32]=0;
```

```
    }
    transpose(a,swap,64);
    transpose(a,FinalTr,64);
}

void DES::Decypt(unsigned int a[])
{
    int InitialTr[64]={57,49,41,33,25,17,9,1,59,51,43,35,27,19,11,3,61,
                       53,45,37,29,21,13,5,63,55,47,39,31,23,15,7,56,48,
                       40,32,24,16,8,0,58,50,42,34,26,18,10,2,60,52,44,36,
                       28,20,12,4,62,54,46,38,30,22,14,6};
    int FinalTr[64] ={39,7,47,15,55,23,63,31,38,6,46,14,54,22,62,30,37,
                      5,45,13,53,21,61,29,36,4,44,12,52,20,60,28,35,3,43,
                      11,51,19,59,27,34,2,42,10,50,18,58,26,32,1,41,9,49,
                      17,57,25,32,0,40,8,48,16,56,24};

   int swap[64]    ={32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,
                     49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,0,1,
                     2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,
                     22,23,24,25,26,27,28,29,30,31};

  int KeyTr1[56]   ={56,48,40,32,24,16,8,0,57,49,41,33,25,17,9,1,58,50,
                     42,34,26,18,10,2,59,51,43,35,62,54,46,38,30,22,14,6,
                     61,53,45,37,29,21,13,5,60,52,44,36,28,20,12,4,27,19,
                     11,3};

 unsigned int key[64]= {1,1,0,1,0,1,0,0,0,1,1,0,0,1,0,1,1,0,0,1,1,1,0,0,1,0,1,
                        0,1,1,1,0,0,0,1,1,0,1,1,0,0,1,1,1,1,1,1,0,0,1,1,0,1,1,0,
                        0,1,1,1,1,1,0,1,0,1,1};
   unsigned int b[64];
   unsigned int x[64];
   transpose(a,InitialTr,64);
   transpose(key,KeyTr1,56);
   for(int i=16;i>0;i--)
   {
```

```
        for(int n=0;n<64;n++)
        b[n]=a[n];
        for(int j=0;j<32;j++)
        a[j]=b[j+32];

        fd(i,key,a,x);

        for(int j=0;j<32;j++)
        if (b[j]+x[j]==1)
            a[j+32]=1;
        else
            a[j+32]=0;
      }
      transpose(a,swap,64);
      transpose(a,FinalTr,64);
}
class EncryptionFile
{
        public:
    void EncryptFile(char[]);
    void DecryptFile();
};

void HextoBin(unsigned int Hex,unsigned int Bin[],int j)
{
   div_t D;
   for(int i=1;i<9;i++)
   {
        if (Hex%2!=0) Bin[8*j-i]=1; else Bin[8*j-i]=0;
     D=div(Hex,2);
     Hex=D.quot;
   }
}

unsigned int BintoHex(unsigned int Bin[],int j)
```

```
{
  unsigned int Hex;
        Hex=128*Bin[8*j-8]+64*Bin[8*j-7]+32*Bin[8*j-6]+16*Bin[8*j-5]+8*Bin[8*j-4]
    +4*Bin[8*j-3]+2*Bin[8*j-2]+Bin[8*j-1];
  return Hex;
}
void EncryptionFile::EncryptFile(char NameOfFile[])
{
  DES Encyption;
  FILE *FileWrite;
  fpos_t pos;
        int File,Data,redo,x,loop;
  unsigned int FileData[64],Temp1[64],Temp2[64];
  unsigned int *buf;
  char FileList[5][14]={"Encrypt00.dat","Encrypt01.dat","Encrypt02.dat"
                                        ,"Encrypt03.dat","Encrypt04.dat"};
  buf = (unsigned int *)malloc(1);
  *buf = 0;
  loop = 0;

  do
  {
  do
  {
    redo=0;
        File = _rtl_open(NameOfFile,SH_DENYRW);

        do
        {
        randomize();
        x=random(32);
        } while (x==0);
   for(int i=1;i<256*x+1;i++)
   Data =_rtl_read(File,buf,1);
   for(int i=1;i<8;i++)
```

```
    {
        Data=_rtl_read(File,buf,1);
        HextoBin(*buf,FileData,i);
    }
    HextoBin(x,FileData,8);

    for (int i=0;i<64;i++)
    Temp2[i]=FileData[i];

    Encyption.Encypt(FileData);
    for (int i=0;i<64;i++)
    Temp1[i]=FileData[i];
    Encyption.Decypt(Temp1);

    for (int i=1;i<64;i++)
    if (Temp1[i]!=Temp2[i]) redo=1;
        _rtl_close(File);
  }while(redo==1);
  FileWrite = _fsopen(NameOfFile,"r+",SH_DENYNO);
  pos=256*x;
  fsetpos(FileWrite,&pos);
  fwrite("LockNow",7,1,FileWrite);
  fclose(FileWrite);
  File =  _rtl_creat(FileList[loop],0);
  for (int i=1;i<9;i++)
  {
    *buf = BintoHex(FileData,i);
    Data = _rtl_write(File,buf,1);
  }
  _rtl_close(File);
  loop++;
  } while (loop<5);
  free(buf);
  rename(NameOfFile,"File.dat");
}
```

```
void EncryptionFile::DecryptFile()
{
  DES Decyption;
  fpos_t pos;
  FILE *FileWrite;
        int File,Data,x,loop;
  unsigned int FileData[64];
  unsigned int *buf;
  char FileList[5][14]={"Encrypt00.dat","Encrypt01.dat","Encrypt02.dat"
                                          ,"Encrypt03.dat","Encrypt04.dat"};
  buf = (unsigned int *)malloc(1);
  *buf = 0;
  loop = 4;
  do
  {
        File = _rtl_open(FileList[loop],SH_DENYRW);
    for(int i=1;i<9;i++)
    {
        Data=_rtl_read(File,buf,1);
        HextoBin(*buf,FileData,i);
    }
    _rtl_close(File);
    Decyption.Decypt(FileData);
  x=BintoHex(FileData,8);
  FileWrite = _fsopen("File.dat","r+b",SH_DENYNO);
  pos=256*x;
  fsetpos(FileWrite,&pos);
  for (int i=1;i<8;i++)
  {
    *buf=BintoHex(FileData,i);
        fwrite(buf,1,1,FileWrite);
  }
  fclose(FileWrite);
  loop--;
  } while (loop>-1);
```

```
    free(buf);
    rename("File.dat","Run.exe");

}
void main()
{
        EncryptionFile DecrypttoRun;
    DecrypttoRun.DecryptFile();
    system("del Encrypt*.*");
    system("run.exe");
    DecrypttoRun.EncryptFile("Run.exe");
}
```

กิตติกรรมประกาศ

ผู้จัดทำขอขอบคุณ

อาจารย์สมศักดิ์ มิตะถา ที่ให้คำปรึกษาและคำแนะนำ รวมทั้งเอื้อเฟื้อข้อมูลที่เป็นประโยชน์ต่อโครงงานนี้

นายนเรศ มาลัย ที่ได้เอื้อเฟื้อข้อมูลในการทำโครงงานนี้

ผู้จัดทำ

บรรณานุกรม

1. นายจักรกริศน์ เขียวสะอาด, นายนริศ ภิญโญวัฒยากร, "การออกแบบวงจรดิจิตอลด้วยภาษาวีเฮชดีแอล" ,, ,คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ,หน้า15-34
2. นายธีรศักดิ์ชัย อั่งสกุล ,"การออกแบบวงจรรวมเฉพาะกิจเพื่อป้องกันการเข้าถึงข้อมูลในหน่วยความจำ" ,คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ,หน้า 9-24