การออกแบบวงจรดิจิตอลด้วยภาษาวีเอชดีแอล

DIGITAL SYSTEM DESIGN BY VHDL

โดย

นายจักรกริศน์ เขียวสะอาด

นายนริศ ภิญโญวัฒยากร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบวงจรดิจิตอลด้วยภาษาวีเฮชดีแอล

ผู้จัดทำ

1. นายจักรกริศน์ เขียวสะอาด รหัส 35104061
2. นายนริศ ภิญโญวัฒยากร รหัส 35104208

..... อาจารย์ที่ปรึกษา

(อาจารย์ บรรจง ปิยชำรง)

การออกแบบวงจรดิจิตอลด้วยภาษาวีเฮชดีแอล

จักรกริศน์ เขียวสะอาด 35104061
นริศ ภิญโญวัฒยากร 35104208
อาจารย์บรรจง ปิยชำรง อาจารย์ที่ปรึกษา
ปีการศึกษา 2538

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการศึกษาการนำภาษาวีเฮชดีแอล (VHDL) ซึ่งเป็นภาษาซึ่งสามารถบรรยายลักษณะของฮาร์ดแวร์ มาใช้ในการออกแบบวงจรดิจิตอลในการเข้ารหัสและถอดรหัสข้อมูล โดยใช้อัลกอริทึมดีอีเอส [DES (Data Encryption Standard)] ซึ่งสามารถนำไปประยุกต์ใช้เพื่อความปลอดภัยในการสื่อสารข้อมูล วัตถุประสงค์ในการนำภาษาวีเฮชดีแอล มาใช้ในการออกแบบวงจรดิจิตอลเพื่อเป็นการศึกษาถึงแนวทางใหม่ ๆ ซึ่งจะช่วยลดเวลาและค่าใช้จ่ายในการออกแบบวงจรดิจิตอลขนาดใหญ่ได้ และวีเฮชดีแอลเป็นภาษาที่สนับสนุนโดยกล่าวถึงพฤติกรรมต่างๆ หรือโครงสร้างของวงจรดิจิตอล นอกจากนี้ภาษาวีเฮชดีแอลยังสนับสนุนการออกแบบโปรแกรมแบบท้อปดาวน์ (TOP-DOWN) และ บัททอมอัพ (BUTTOM-UP) ซึ่งสามารถออกแบบได้ตั้งแต่ระดับสถาปัตยกรรมถึงระดับเกท วีเฮชดีแอลจึงเป็นแนวทางใหม่ในการออกแบบวงจรดิจิตอลที่น่าทำการศึกษา

Digital System Design by VHDL

Chakkrit Kheawsa-ad 35104061
Naris Pinyowattayakorn 35104208
Bunjong Piyatamrong Advisor
1995

ABSTRACT

This thesis is to study how to use VHDL , which is a hardware description language, to design data encryption and decryption digital system by using DES's (Data Encryption Standard) algorithm which can be applied to use for data communication security. Using VHDL to design digital system is to study a new trend which can reduce time and cost in digital system development ,and VHDL also supports top-down and buttom-up design methodologies. So the system can be described from architecture level to gate level. Conclude that VHDL is a new way to digital system design that should study.

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

ปริญญานิพนธ์ฉบับนี้นำเสนอการออกแบบวงจรรวมดิจิตอลด้วยภาษาวีเอชดีแอล โดยใช้คอมพิวเตอร์ช่วยในการออกแบบ เป็นการออกแบบในแนวใหม่ซึ่งสะดวกรวดเร็วและมีประสิทธิภาพ ภาษาวีเอชดีแอลเป็นภาษาบรรยายการทำงานของฮาร์ดแวร์เพื่อใช้อธิบายการทำงานของวงจรดิจิตอลตัวภาษานี้นประกอบไปด้วยรายละเอียดและส่วนประกอบต่าง ๆ ซึ่งใช้อธิบายพฤติกรรม (Behavioral) หรือโครงสร้าง (Structural) ของระบบ โดยพิจารณาถึงฐานเวลา (Timing) ของระบบเป็นหลักเนื่องจากว่าระบบที่เราจะออกแบบโดยวีเอชดีแอลนั้น ผู้ออกแบบไม่จำเป็นต้องรู้ถึงวงจรภายในว่ามีอะไรต่อกันอย่างไรบ้าง เพียงแต่กำหนดอินพุท เอาท์พุท และฟังก์ชันการทำงานของระบบเขียนเป็นโปรแกรมแสดงการไหลของข้อมูล (Dataflow) , การทำงานของระบบหรือโครงสร้างของระบบขึ้นมา จากนั้นก็ทำการคอมไพล์และซิมูเลทโมเดลที่ออกแบบมาเพื่อทำการวิเคราะห์ดูฟังก์ชันฐานเวลาของระบบว่าตรงตามต้องการหรือไม่ ถ้ามีการแก้ไขอย่างไรก็ทำการแก้ไขซอสโคด (Source Code) ใหม่ แล้วทำการคอมไพล์และซิมูเลทซ้ำ ๆ จนกว่าจะได้โมเดลที่มีฐานเวลาของระบบตามต้องการ ซึ่งมีความง่ายและสะดวกรวดเร็วกว่าเมื่อก่อนมากเพราะไม่ต้องเสียเวลากับการสร้างวงจรทางฮาร์ดแวร์จริง ๆ ขึ้นมาทดสอบ ซึ่งทำให้เสียเวลาและค่าใช้จ่ายสูง โมเดลที่ออกแบบโดยใช้ดีเอชดีแอลสามารถทำการสังเคราะห์ (Synthesis) เพื่อให้ได้ชีเมติกไดอะแกรม (Schematic Diagram) และเอาไปสร้างเป็นวงจรจริง ๆได้ โดยอาจนำเอาไปสร้างเป็นเอเอสไอซี (Application Specific Integrated Circuit) หรือนำไปเบิร์น (Burn) ลง อีพีแอลดี หรือเอฟพีจีเอ เพื่อนำไปใช้งานจริงต่อไป ประโยชน์ของภาษาวีเอชดีแอลใช้กันมากในการออกแบบชิปเอเอสไอซี หรือออกแบบไมโครโปรเซสเซอร์ โดยมีซอฟแวร์หลายตัวสนับสนุนตั้งแต่การคอมไพล์ , ซิมูเลท , การสังเคราะห์ เช่น เมนเตอร์กราฟฟิก (Mentor Graphics) , วิวลอจิก (Viewlogic) ประกอบกับคอมพิวเตอร์ระดับเวอร์กสเตชันที่มีระบบกราฟิกที่ดีและการประมวลผลด้วยความเร็วสูงในปัจจุบัน ทำให้การออกแบบทุกขั้นตอนเป็นไปอย่างรวดเร็วและมีประสิทธิภาพ ซอฟแวร์ซีเออี (CAE) ที่ใช้ในการพัฒนาโครงงานนี้ขึ้นมาคือ วิวลอจิก (View Logic™) ตั้งแต่การเขียนซอสโคด การคอมไพล์วีเอชดีแอลซอสโคด การทดสอบฟังก์ชันการทำงาน การจำลองการทำงานตรวจสอบความถูกต้องของวงจร เมื่อได้ซอสโคดที่สมบูรณ์ของโมเดลแล้วสามารถนำไปทำการแปลงซอสโคดให้อยู่ในรูปชีเมติกไดอะแกรม (Schematic diagram) โครงงานนี้ได้ทดลองทำการออกแบบวงจรเข้ารหัสถอดรหัสข้อมูล โดยใช้อัลกอริทึมดีอีเอส (Data Encryption Standard's Algorithm) ซึ่งเป็นอัลกอริทึมการเข้ารหัสและถอดรหัสข้อมูลที่ใช้กันเป็นมาตรฐานสากลและมีความปลอดภัยสูง โดยผลที่ออกมานั้น สามารถนำมาใช้งานได้จริง ในการศึกษาเรื่องภาษาวีเอช

ดีแอลนั้น ผู้จัดทำยอมรับว่ายังเป็นเรื่องที่ใหม่ และมีข้อผิดพลาดหลายประการ จึงมีความบกพร่องและไม่สมบูรณ์อยู่ทั้งในตัวรายงานและโครงงาน ซึ่งผู้จัดทำยินดีรับคำติชมและจะปรับปรุงแก้ไขให้ดีขึ้นต่อไป หวังว่ารายงานฉบับนี้จะเป็นประโยชน์ต่อท่านผู้ที่สนใจเพื่อเป็นแนวทางการศึกษาภาษาวีเอชดีแอลต่อไป

ผู้จัดทำ

บทที่ 2
ภาษาวีเอชดีแอล(VHDL)

วีเอชดีแอล(VHDL) ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC ย่อมาจาก Very High Speed Integrated Circuit) เป็นภาษาคอมพิวเตอร์ระดับสูง (High Level Language)ซึ่งใช้อธิบายการทำงานของระบบดิจิตอลฮาร์ดแวร์ สามารถใช้อธิบายฟังก์ชันการทำงานได้หลาย ๆ ระดับ ตั้งแต่ระดับบล็อก จนถึงระดับเกท ความซับซ้อนของระบบสามารถจะเขียนได้ตั้งแต่ระดับเกท ประกอบกันจนเป็นระบบที่สมบูรณ์ รูปแบบของภาษาวีเอชดีแอลนั้น จะประกอบไปด้วย 2 ส่วนใหญ่ ๆ ได้แก่ ส่วนของภาษาซีเควนเชียล (Sequential Language) และภาษาคอนเคอร์เรนท์ (Concurrent Language) การโปรแกรมด้วยภาษาวีเอชดีแอลสามารถเขียนได้ทั้ง 2 รูปแบบรวมกัน เพราะในการทำงานของระบบใด ๆ ย่อมจะมีการทำงานในแบบ ซีเควนเชียล และ คอนเคอร์เรนท์ อยู่รวมกัน นอกจากนี้ตัวภาษาวีเอชดีแอลยังสามารถอธิบายถึงการเชื่อมต่อระหว่างระบบย่อย ๆ เข้าด้วยกันเพื่อให้เป็นระบบใหญ่ได้ ตัวภาษาวีเอชดีแอลนอกจากจะกำหนดรูปแบบไวยากรณ์ (Syntax) ของตัวภาษาแล้ว ยังมีการตรวจสอบความหมายของตัวภาษาว่าจะซิมูลเลชัน (Simulation) ได้หรือไม่ เพราะโปรแกรมที่เขียนโดยวีเอชดีแอลต้องผ่านการซิมูลเลชันเพื่อตรวจสอบดูการทำงาน ฉะนั้นในการคอมไพล์ (Compile) จะมีการตรวจสอบทั้งไวยากรณ์และซิมูลเลชันซีแมนติค (Semantics) อย่างไรก็ตามแม้ตัวภาษาจะมีความซับซ้อนในรูปแบบและกฎเกณฑ์ของภาษา แต่การเรียนรู้เพียงบางส่วนของภาษาก็สามารถนำไปใช้งานโดยไม่จำเป็นต้องศึกษารายละเอียดทั้งหมด เนื่องจากตัวภาษาวีเอชดีแอลออกแบบมาให้ใช้สำหรับการออกแบบตั้งแต่วงจรที่มีขนาดเล็กจนถึงวงจรที่มีขนาดใหญ่และซับซ้อน

2.1 ประวัติความเป็นมาของภาษาวีเอชดีแอล

ความต้องการภาษานี้เริ่มจากโครงการวีเอชเอสไอซี (VHSIC) ของ Department Of Defence ของสหรัฐอเมริกาเนื่องจากมีบริษัทที่สร้างวีเอชเอสไอซีชิป (Chip) หลายบริษัทได้ร่วมโครงการที่จะพัฒนา ในขณะนั้นหลายบริษัทใช้ภาษาวีเอชดีแอล ซึ่งแตกต่างกันในการที่จะอธิบายการทำงานชิปของตน ด้วยเหตุนี้ทำให้เกิดความแตกต่าง แต่ละบริษัทไม่สามารถแลกเปลี่ยนเทคโนโลยีให้กันและกันได้ ทำให้ DOD เกิดปัญหาในการที่จะพัฒนาและซ่อมบำรุงในภายหลัง จึงเกิดความต้องการภาษาวีเอชดีแอล ซึ่งเป็นมาตราฐานในการที่จะอธิบายถึงตัวแบบ นั้น ๆ ดังนั้น DOD จึงมอบให้บริษัทไอบีเอ็ม(IBM) ,เท็กซัสอินสทรูเมนต์(TEXAS INSTRUMENT) และ อินเตอร์เมติกส์(INTERMETICS) 3 บริษัทร่วมกันพัฒนาและกำหนดมาตรฐานของวีเอชดีแอลขึ้นมา ในปี 1983 หลังจากนั้น วีเอชดีแอลเวอชั่น 7.2(VHDL VERTION 7.2) ได้ทำการพัฒนาและออกเผยแพร่ต่อสาธารณะชนในปี 1985 ได้รับความสนใจเป็นอย่างมากในอุรสาหกรรม โดยเฉพาะอย่างยิ่งบริษัทที่ทำวีเอชเอสไอซีชิป จากผลสำเร็จนี้ทำให้เกิดมาตรฐาน IEEE ของวีเอชดีแอล ในปี 1986 หลังจากนั้นก็มีการพัฒนาขยายขีดความสามารถของภาษาวีเอชดีแอลเพิ่มขึ้น และ DOD ก็มีการปรับปรุงและจดมาตราฐานใหม่ IEEE ในปี 1987 อีกครั้ง ซึ่งเป็นที่รู้จักกันในชื่อของ IEEE STD 1076 - 1987 หลังจากกันยายน 1988 บริษัทใด ๆ ที่ทำการพัฒนาเอเอสไอซี (ASIC) ชิป ใน Department Of Defence ของอมริกาต้องส่งตัววีเอชดีแอลโมเดล พร้อมกับชุดทดสอบตามมาตรฐานที่ได้กำหนดเอาไว้

2.2 ความสามารถของภาษาวีเอชดีแอล (CAPABILITY)

- ตัวภาษาวีเอชดีแอลสามารถใช้เป็นสื่อกลางในการแลกเปลี่ยนระหว่างผู้ผลิตชิปกับผู้ออกแบบ (CAD Tools)

- ใช้เป็นการสื่อกลางในการแลกเปลี่ยนสื่อสารระหว่างซีเออี(CAE) และซีเอดีทูล(CAD Tools) เช่นตัวภาษาซ้อสโค้ด (Source Code) ของวีเอชดีแอล สามารถคอมไพล์โดยใช้คอมไพล์เลอร์ (Compiler) และซิมูลเลเตอร์ (Simulator) ได้หลายตัวแตกต้างกัน

- ภาษาวีเอชดีแอลสนับสนุนการออกแบบ แบบท็อปดาวน์(Top Down Design) และแบบบัททอมอับ(Bottom Up Design) หรือผสมกันทั้ง 2 แบบ

- ตัวภาษาวีเอชดีแอลเป็นแบบทั่วไป (Generic) คือไม่อิงเทคโนโลยีอันใดอันหนึ่ง สามารถอิงเทคโนโลยีใดก็ได้ และในขณะเดียวกันก็สามารถสนับสนุนหลาย ๆ เทคโนโลยี

- สนับสนุนการออกแบบทั้งระบบซิงโครนัส (Synchronous) และอะซิงโครนัส (Asynchronous)

- สนับสนุนการออกแบบระบบดิจิตอล ในหลาย ๆ เทคนิค เช่นไฟไนท์สเตทแมชชีน (Finite State Machine) , อัลกอริทะมิก (Algorithmic) หรือสมการบูลีน (Boolean Equation)

- ตัวภาษาวีเอชดีแอลสามารถอ่านและทำความเข้าใจได้โดยมนุษย์

- ภาษาวีเอชดีแอลเป็นมาตราฐานรับรองโดย IEEE และ ANSI ทำให้โมเดลที่ออกแบบโดยภาษาวีเอชดีแอล สามารถเคลื่อนย้ายไปยังระบบใด ๆ ก็ได้ และสามารถนำกลับมาใช้ใหม่ได้

- ภาษาวีเอชดีแอลสนับสนุนรูปแบบการเขียนถึง 3 รูปแบบ ได้แก่ แบบบีเฮฟวีเออร์ (Behavioral Style) ,แบบสทรักเทอรัล(Structural style) ,แบบดาต้าโฟลว์(Data Flow) หรือสามารถเขียนรวมกันทั้ง 3 รูปแบบ

- สนับสนุนการออกแบบขนาดใหญ่โดยใช้ความสามารถของ ส่วนประกอบ (Component) , ฟังก์ชั่นโพลซีเดอร์(Function Procedure) และ แพคเกต(Package)

- ไม่จำเป็นต้องศึกษาซอฟท์แวร์(Software)ซิมูลเลเตอร์เพราะซิมูลเลชันโมเดลสามารถเขียนได้โดยใช้ภาษาวีเอชดีแอล เช่นกัน

- สามารถเขียนโมเดลได้ขนาดไม่จำกัด ไม่มีข้อจำกัดในตัวภาษาเรื่องขนาดของโมเดล (ขึ้นอยู่กับซอฟท์แวร์)

- สามารถอธิบายตัวแปรที่เกี่ยวกับฟังก์ชั่นทางด้านเวลา เช่น Propagation Delay , Min-Max Delay , Setup , Holding Time , Spike Detection สามารถอธิบายได้ภายในตัวภาษา

- เจเนริกส์(GENERICS) ช่วยให้เราสามารถสร้างตัวแปรของรูปแบบ (Design)

- โมเดลที่สร้างด้วยภาษาวีเอชดีแอลนั้น ไม่เพียงแต่จะอธิบายฟังก์ชั่นการทำงานเท่านั้น แต่ยังสามารถอธิบายถึงรายละเอียดของตัวโมเดล เช่น Total Area และ Speed ของโมเดล

- ภาษาวีเอชดีแอลเป็นมาตรฐานที่ใช้โดยบริษัทและผู้ออกแบบหลาย ๆ แห่ง ฉะนั้นจึงเป็นการง่ายที่จะทำความเข้าใจ ถึงแม้ว่าจะมาจากแหล่งต่าง ๆ

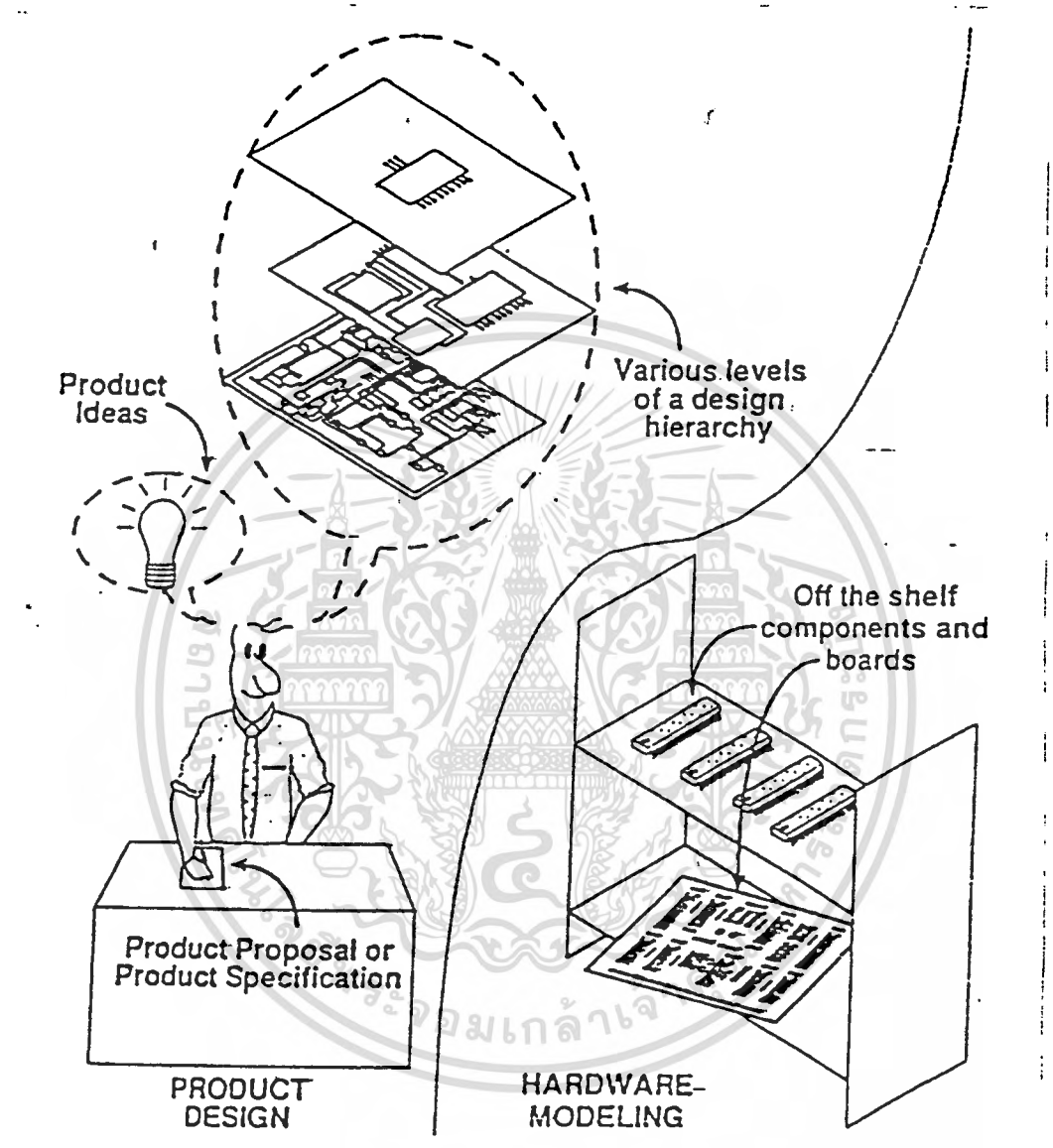
- โมเดลที่สร้างขึ้นสามารถจำลองการทำงานได้ เพราะว่าตัวแปลภาษาได้ตรวจสอบไวยากรณ์ทางด้านซิมูลเลชันซีแมนติคไว้ด้วย

- การอธิบายโมเดลด้วยแบบบีเฮฟวีเออร์สามารถ Synthesis ไปเป็นระดับเกท - เลเวลได้ ถ้าทำตามกฎของ Synthesis Guideline

- มีความสามารถที่ให้เราออกแบบข้อมูลชนิดใหม่ ๆ ได้ทำให้วีเอชดีแอลโมเดล เป็นการออกแบบในระดับสูง ที่ไม่ต้องคำนึงถึงว่าจะสร้างตัวโมเดลนั้นขึ้นมาได้อย่างไร

2.3 หลักการสร้างโมเดลโดยภาษาวีเอชดีแอล

(General VHDL Modelling Principles)



รูปที่ 2.1 สิ่งต่าง ๆ ที่สามารถอธิบายด้วยวีเอชดีแอลได้

วีเอชดีแอลเป็นภาษาที่ใช้สำหรับอธิบายการทำงานของฮาร์ดแวร์ในรูปแบบฟอร์มที่อ่านเข้าใจได้ ซึ่งช่วยในการสร้างและออกแบบวงจรรวมดิจิตอล และ ส่วนประกอบต่าง ๆ อาจใช้อธิบายระบบทั้งระบบ หรืออธิบายเพียงบางส่วน ซึ่งอยู่ในรูปของ Component Block จากนั้นก็ทำการจำลองการทำงาน (Simulate) โดยที่รูปแบบนั้นยังไม่ได้สร้างขึ้นจริง หรือเพียงแต่อยู่ในรูป

ของคำอธิบายเท่านั้น (Textual Format) หลังจากจำลองการทำงานจนได้ตามที่ต้องการจึงนำไปทำการ Synthesis เพื่อให้ได้วงจรเกทเลเวลต่อไป

ประโยชน์จริงของการใช้ วีเอชดีแอลเป็น Design Tools แทนการสร้างต้นแบบ (Prototype) ขึ้นมาจริง คือเราสามารถอธิบาย Product Idea , Product Proposal , Product Specification เป็นลักษณะในรูปของ Text จากนั้นก็นำไปคอมไพล์เพื่อดู Timing การทำงาน จากนั้นก็ทำการ Refine แก้ไขจนกว่าจะได้ Specification ตามต้องการ เมื่อ Product ได้ผลตามที่ต้องการแล้วจึงนำไปเข้าสู่การ Synthesis เพื่อให้ได้เกทเลเวล Schematic แล้วนำไปสร้างเป็นต้นแบบจริงต่อไป ซึ่งต้นแบบที่สร้างนั้นทำงานได้จริงเพราะได้ทำการ Simulate เรียบร้อยแล้ว เป็นการลดเวลาและค่าใช้จ่ายในการสร้างต้นแบบได้มาก

เนื่องจากว่าภาษาวีเอชดีแอลเป็นภาษาที่มีประสิทธิภาพสูง เราจึงใช้ภาษาวีเอชดีแอลอธิบายฮาร์ดแวร์ เพราะว่ามีข้อดี 2 ประการ คือ

1. เข้าใจได้ง่าย
2. สามารถแก้ไขได้ง่าย

การเข้าใจได้ง่ายมีประโยชน์ต่อใครก็ได้ซึ่งมีความจำเป็นที่จะต้องอ่านรหัส ที่ได้ออกแบบมาแล้วโดยไม่จำเป็นต้องให้ผู้ออกแบบมาอธิบายให้ฟัง ตัวภาษาวีเอชดีแอลอธิบายการทำงานภายในตัวอยู่แล้ว ส่วนอีกประการหนึ่งก็คือ ความต้องการในการเปลี่ยนแปลงฮาร์ดแวร์ ที่ได้ออกแบบแล้ว คือว่า หลังจากทดสอบแล้วพบข้อที่ผิดพลาดซึ่งต้องแก้ไข หรือว่า ระหว่างพัฒนามีการเปลี่ยนแปลงความต้องการของระบบ หรือต้องการเพิ่มการทำงานบางส่วนลงไป ตัวภาษาวีเอชดีแอล นั้นสนับสนุนหลักการต่าง ๆ ให้เขียนแก้ไข และบำรุงรักษาวงจรดิจิตอล ที่มีความซับซ้อนเป็นไปอย่างรวดเร็ว และมีประสิทธิภาพ ซึ่งหลักการมีดังนี้

1. Top Down Design
2. Modularity
3. Abstraction
4. Information Hiding
5. Uniformity

ซึ่งจะอธิบายประโยชน์ของหลักการต่าง ๆ ในหัวข้อต่อไป และแสดงให้เห็นว่า ภาษาวีเอชดีแอลนั้นช่วยในการพัฒนาวงจรดิจิตอล ขนาดใหญ่และซับซ้อนนั้นให้อ่านเข้าใจได้ง่าย และแก้ไขได้ง่ายอย่างไร

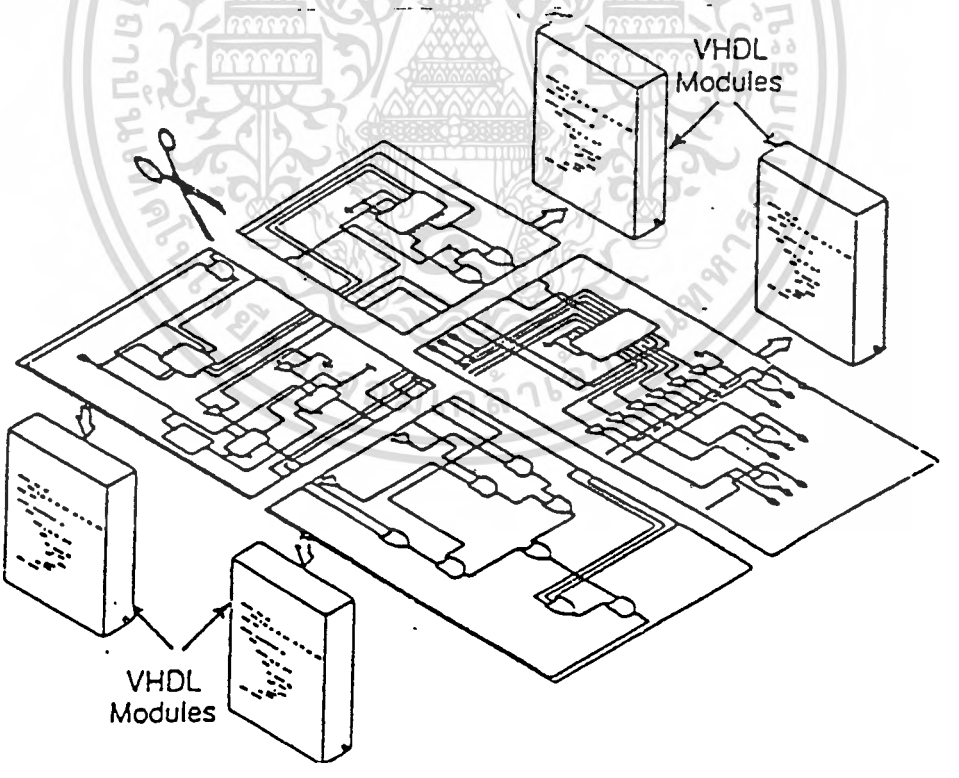
2.3.1. Top Down Design

ในการพัฒนาวงจรรวมดิจิตอล ขนาดใหญ่ที่มีความซับซ้อน เช่น ASIC (Application Specific Integrated Circuit) วิศวกรหรือผู้ออกแบบมักจะมองรูปแบบให้อยู่ในรูปของ Block Diagram เสียก่อน ก่อนที่จะย่อยรูปแบบ ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษาวีเอชดีแอลนั้นอนุญาตให้

- อธิบายการทำงานของแต่ละ Block
- วิเคราะห์การทำงาน (Analyze)
- จัดการแก้ไขและปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามที่ต้อง-

การ ก่อนที่จะทำการออกแบบให้ละเอียดลึกลงในขั้นตอนต่อไป การแก้ไขในขั้นตอนนี้จะทำให้ลดค่าใช้จ่ายกว่าการแก้ไขในช่วงของการพัฒนาในระดับสร้างซิลิกอนชิป (Silicon Chip)

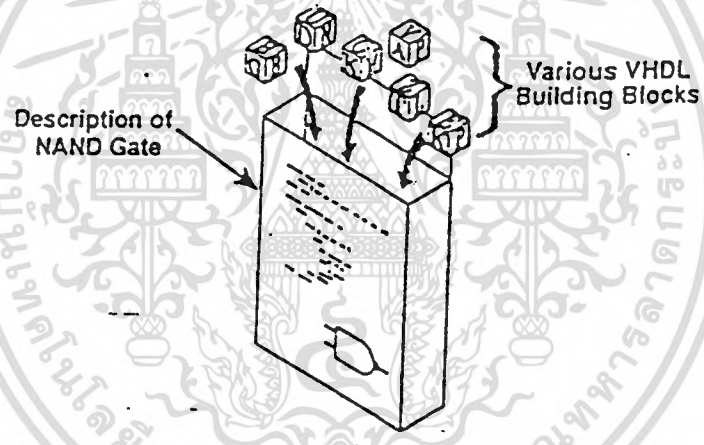
2.3.2. Modularity



รูปที่ 2.2 การแบ่งย่อยในระดับราบของการออกแบบฮาร์ดแวร์

Modularity คือ หลักการในการแยกส่วน (Partitioning) ฮาร์ดแวร์ ออกเป็นส่วนย่อยเล็ก ๆ ลงไป ซึ่งปกติการทำงานของฮาร์ดแวร์ใหญ่ต้องประกอบด้วยฮาร์ดแวร์ย่อย ๆ ลงไป ดังรูปที่ 2.2 แสดงรูปแบบ ซึ่งแสดงวงจรทั้งหมดในรูปเดียว (Flatten Design) หลังจากนั้นตัดออกเป็นส่วนย่อย ๆ เล็กลงมา เมื่อเราออกแบบโดยใช้ภาษาวีเอชดีแอลหน้าที่การทำงานของแต่ละส่วนสามารถอธิบายได้โดย Module ของ Code (คล้าย Function หรือ Procedure) ซึ่งแสดงการทำงานของส่วนย่อยนั้นอย่างชัดเจน ซึ่งการแยกรูปแบบใหญ่ ๆ ออกเป็นส่วนย่อย ๆ นี้ทำให้ง่ายต่อการจัดการและง่ายต่อการทำความเข้าใจ

ตัวภาษาวีเอชดีแอลประกอบขึ้นมาด้วย Language Building Block ซึ่งประกอบไปด้วย 75 Reverse Word และมากกว่า 200 Combination words รูปที่ 2.3 แสดงให้เห็นว่า ภาษาวีเอชดีแอลแต่ละ Module นั้นประกอบด้วย Language Building Block อะไรบ้าง และอธิบายการทำงานของ NAND เกท



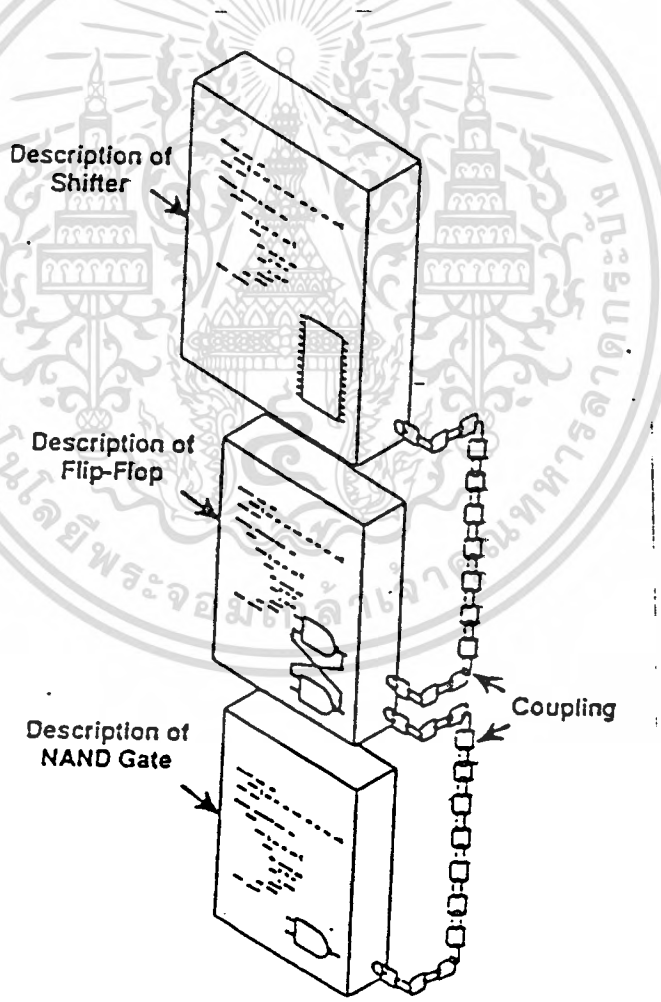
รูปที่ 2.3 ฮาร์ดแวร์โมดูลซึ่งสร้างจากการสร้างบล็อคของ VHDL

จากรูปที่ 2.4 แสดง Hierarchy Method โดยการแยกส่วนรูปแบบออกเป็นส่วนย่อย ๆ ส่วนบนสุดอธิบายการทำงานของ Shifter ส่วนล่าง ๆ ลงมาคือการแยกส่วนของ Shifter ออกเป็นฟลิปฟลอป จาก ฟลิปฟลอป แยกเป็น NAND เกท ภายใน Shifter ได้อธิบายการทำงานโดยใช้การต่อกันของฟลิปฟลอป ในระดับต่ำลงมา ฟลิปฟลอปก็เกิดจากการใช้ NAND เกท ต่อกัน 2 ตัว ในระดับต่ำลงมาอีกก็เป็น NAND เกท ซึ่งมีการอธิบายการทำงานอยู่ภายใน ซึ่งแต่ละ Module จะมีคำอธิบายการทำงานในตัวของมันเองอยู่แล้ว คำอธิบายภายในแต่ละ Module มีไว้เพื่อให้สามารถใช้ฟลิปฟลอป Module ได้ ส่วน ฟลิปฟลอป Module ก็อธิบายการเชื่อมต่อไว้อย่างดีทำให้สามารถเชื่อมต่อกับ NAND เกท ในระดับล่างสุดได้

ประโยชน์อย่างหนึ่งของการแยกส่วนฟลิปฟลอป และ NAND เกท ออกจากกัน เนื่องจากว่าทำให้ง่ายในการที่จะใช้ NAND เกทตัวนี้ในรูปแบบไฮเลเวล (High Level) ตัวอื่น ๆ ทำให้นำออกใช้ได้อีก และลดความซับซ้อนในการใช้อุปกรณ์ส่ง เป็นการง่ายที่จะแก้ไขการทำงานของ Shifter โดยปราศจากการแก้ไข Flip-Flop และ NAND เกท จากประโยชน์ที่ได้ของ Modularity นี้ทำให้รูปแบบที่เราออกแบบนั้นง่ายต่อการเข้าใจและแก้ไขได้เสมอ

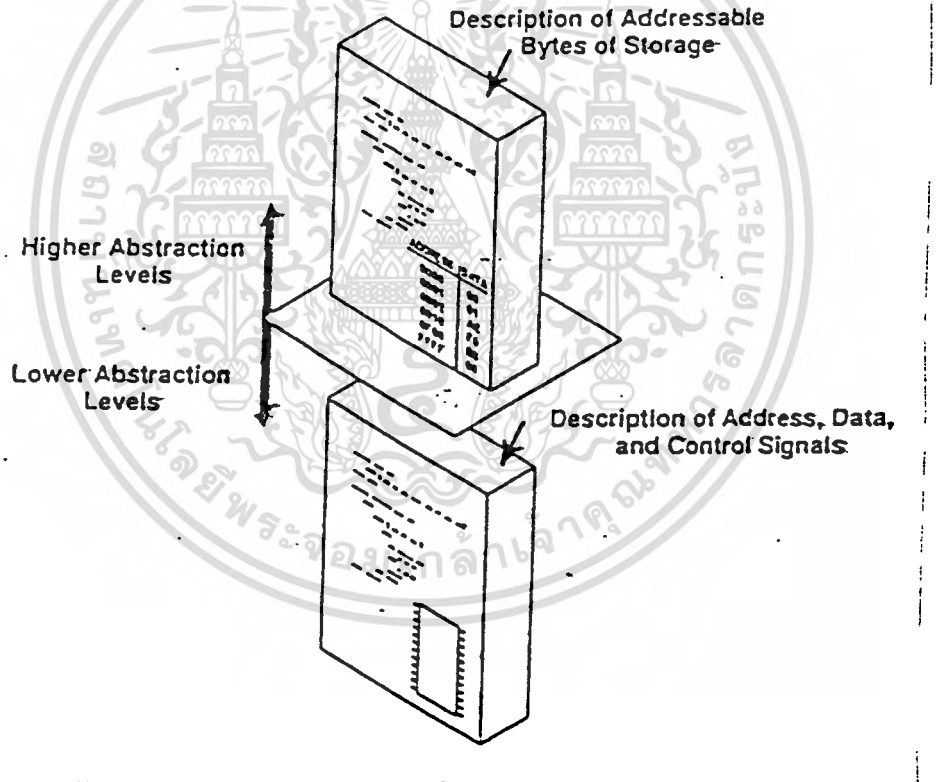
2.3.3. Abstraction

คำนิยามของรูปแบบจะอธิบายการทำงานของตัวรูปแบบ มากกว่าที่จะอธิบายถึงว่าจะพัฒนาตัว รูปแบบนั้นอย่างไร หลักการนี้จะมีความสัมพันธ์อย่างใกล้ชิดกับหลักการ Modularity ในรูปที่ 2.4 Flip-Flop เป็นนิยามในการใช้ NAND เกท และ Shifter นิยามการใช้ฟลิปฟลอป



รูปที่ 2.4 การแบ่งแบบ Hierarchy ของ VHDL Shifter Description

รูปที่ 2.5 แสดงอีกวิธีการหนึ่งซึ่งแสดงถึงการอธิบายการทำงานของรูปแบบโดยใช้ VHDL ในหลาย ๆ ระดับของการนิยาม ROM (Read Only Memory) อธิบายโดยใช้ภาษาระดับสูงไฮเลเวล แสดงถึงตำแหน่ง (Address) ต่าง ๆ ซึ่งเก็บข้อมูลไว้ในตำแหน่งนั้น ๆ ที่ระดับนี้ไม่ต้องสนใจถึง Address Line , Data Line หรือ Control Line เราสามารถเพ่งจุดสนใจไปที่ขนาดของข้อมูล โดยไม่ต้องคิดถึงสัญญาณควบคุมต่าง ๆ มากมายภายใน เพราะว่าส่วนนั้นจะถูกจัดการเองในระดับต่ำลงมา ในระดับล่างลงมาเราสามารถอธิบายการทำงานของสัญญาณแต่ละเส้นภายใน ROM ในการจัดการสัญญาณภายในทุกเส้นภายในการที่จะอ่านข้อมูลหรือโปรแกรมข้อมูลใน ROM ถ้าต้องการเปลี่ยนค่าข้อมูลภายใน ROM เราควรขึ้นมาแก้ไขในระดับที่สูงขึ้นมา (High Level) จะทำให้ง่ายกว่าในการที่จะควบคุมสัญญาณภายใน ซึ่ง

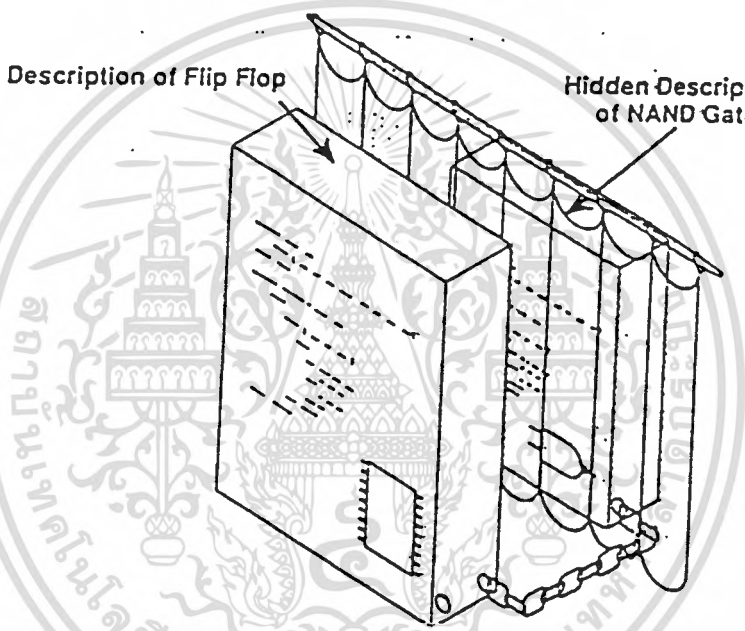


รูปที่ 2.5 Applying Abstraction to a ROM Description

เราจะเห็นว่าในแต่ละระดับมีความเหมาะสมแตกต่างกันไป และตรงจุดนี้เองทำให้รูปแบบที่เราออกแบบง่ายต่อการแก้ไขโดยการใช้ประโยชน์ของ Abstraction

2.3.4. Information Hiding

เมื่อเราทำการเขียน VHDL Code ขึ้นมาเพื่ออธิบายการทำงานของฮาร์ดแวร์ตัวหนึ่ง บางครั้งเราอาจต้องการที่จะซ่อนรายละเอียดการพัฒนา Module นั้น ๆ โดยไม่ต้องการให้ส่วน Module อื่น ๆ รู้การทำงานภายใน Information Hiding มีประโยชน์คือ ทำให้รูปแบบภาษาวีเอชดีแอลนั้นสามารถจัดการได้ง่าย และสามารถอ่านและทำความเข้าใจได้ง่ายกว่า หลักการนี้จะใช้สนับสนุนหลักการ Abstraction คือจะสนใจรายละเอียดในการใช้งานมากกว่าจะสนใจว่ารูปแบบนั้นจะถูกสร้างขึ้นมาอย่างไร มีวงจรอย่างไรบ้าง เป็นต้น การซ่อนรายละเอียดภายใน Module ทำให้ความสนใจของผู้ออกแบบสนใจไปในส่วนที่สำคัญมากกว่า ในส่วนที่ไม่น่าสนใจจะซ่อนไว้และเข้าถึงไม่ได้ ในรูปที่ 2.4 คำอธิบายของ NAND เกทนั้น



รูปที่ 2.6 การซ่อนรายละเอียดที่ไม่จำเป็นของระดับ NAND เกท

จะถูกปิดบังเอาไว้จากคนที่เขียนอธิบายฟลิปฟลอป ดูรูปที่ 2.6 คนที่เขียนอธิบายการทำงานของฟลิปฟลอป ไม่ต้องสนใจเลยว่า NAND เกท จะทำงานอย่างไร จะต่อกันภายในอย่างไร โดย NAND เกท สามารถเขียนขึ้นมาแล้วคอมไพล์เก็บไว้ใน Library ผู้ที่ออกแบบฟลิปฟลอประดับสูงขึ้นมาเพียงแต่ต้องรู้ว่าจะเชื่อมต่ออินพุท/เอาท์พุทของ NAND เกท มาใช้งานได้อย่างไร โดยไม่ต้องสนใจว่า NAND เกท จะถูกสร้างและพัฒนามาอย่างไร และประโยชน์อีกอย่างของ Information Hiding ก็คือป้องกันข้อมูลภายใน ในกรณีที่แจกจ่าย วีเอชดีแอลโมเดล ไปยังที่อื่น ๆ เช่น ส่งไปให้อีกบริษัทใช้พัฒนาร่วมกับวีเอชดีแอลอื่น ๆ โดยเป็นการแจกจ่าย อาจส่งไปแค่ภาษาวีเอชดีแอลที่คอมไพล์แล้ว ไม่ต้องส่งตัวซ้อสโค้ดไป ทำให้เราป้องกันทรัพย์สินทางปัญญาได้ในอีกระดับหนึ่ง

2.3.5. Uniformility

Uniformility เป็นหลักการอีกอย่างที่ช่วยในการอธิบายฮาร์ดแวร์ด้วยภาษาวีเอชดีแอล หมายถึงการสร้าง Module ของรหัสในลักษณะคล้ายกันโดยใช้ตัวภาษา VHDL Building Block ทำให้เกิดการเขียน รหัสที่ดีอย่างเช่น มีการใช้ย่อหน้า มีการใช้คำอธิบาย (Comment) เป็นต้น ทำให้การพัฒนา Module อ่านและทำการเข้าใจง่าย

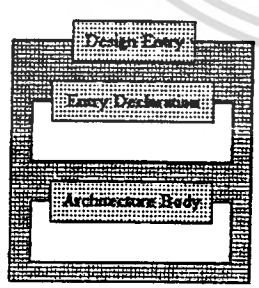
2.4 รูปแบบพื้นฐานของภาษา

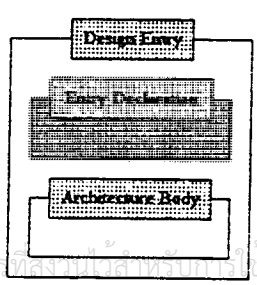
Primary Language Abstraction

ระหว่างการออกแบบ นักออกแบบมักจะพยายามย่อยแตกวงจรออกเป็นส่วนย่อย ๆ เพื่อที่จะจัดการได้ง่ายขึ้นภาษาวีเอชดีแอล สนับสนุนการแยกส่วนฮาร์ดแวร์ และทำให้ง่ายที่จะเขียนพฤติกรรมการทำงานของแต่ละส่วนย่อย ๆ นั้น บางที Unit ย่อย ๆ สามารถใช้ได้ในหลาย ๆ ส่วนของวงจรที่เราจะออกแบบ หรือแม้กระทั่งนำไปใช้ได้ในรูปแบบอื่น ๆ รูปแบบพื้นฐานของภาษา วีเอชดีแอล ในการสร้าง Hardware โมเดล ก็คือ Design Entity ซึ่งสามารถแทน เซล , ชิป , บอร์ด หรือ ระบบย่อย (Subsystem) ได้

Design Entity ประกอบด้วย 2 ส่วนใหญ่คือ ส่วนของ Entity Declaration และส่วนของ Architecture Body

Entity Declaration และ Architecture Body เป็น 2 ส่วนหลักที่สำคัญของภาษา VHDL Language Library คือ ส่วนของ Hardware Description หรือโมเดลซึ่งสามารถจะอยู่ใน Design File ใด ๆ หรือถูกคอมไพล์อยู่ใน Design File หลาย ๆ File ที่แยกจากกัน ความสามารถอันนี้ทำให้นักออกแบบสามารถจัดวางวงจรให้เป็น Module เขียนอธิบายแต่ละ Module จากนั้นก็คอมไพล์แต่ละ Entity แยกจากกัน โดยมี Architecture Body ของแต่ละ Entity ที่แตกต่างกันออกไป การประกาศ Entity Declaration เป็นการกำหนดการเชื่อมต่อ (Interface) ระหว่าง Design Entity กับวงจรส่วนอื่น ๆ ภายนอก โครงสร้างของ Entity Declaration แสดงตัวอย่างต่อไปนี้






```
Entity indentifier is
        Entity_header
        --(generic and/or port clause)
        Entity_declarative_part
        --(Declarations for
        --subprograms,types,signal,..)
begin
        Entity_Statement_part
end identifier;
```

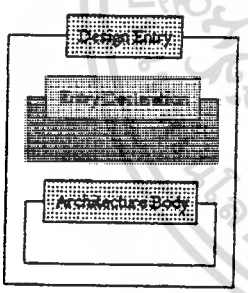
Entity Identifier คือชื่อของ Entity ที่เรากำหนดขึ้น แต่ละ Design Entity รับข้อมูลจากภายนอกโดย Port Mode In และส่งข้อมูลออกไปภายนอกโดย Port Mode Out ดูรูปที่ 2.7

ตัวอย่างต่อไปนี้แสดง Entity Declaration (รวม Port Clause ด้วย) ของ AND เกท 2 Input


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

รูปที่ 2.7 แสดงพอร์ท In และ Out ของ AND เกท

Architecture Body อธิบายความสัมพันธ์ระหว่างอินพุทและเอาท์พุทของ Design Entity โครงสร้างของ Architecture แสดงดังต่อไปนี้

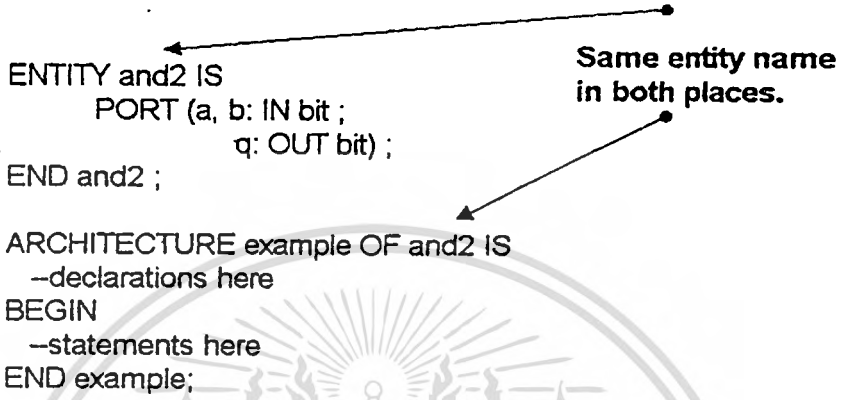



```
Architecture identifier of Entity_name is
        Architecture_declarative_part
begin
        Architecture_statement_part
end identifier;
```

Identifier และ Entity name คือ words ที่คุณจะต้องเขียนไว้ใน ภาษาวีเอชดีแอลโค้ด สิ่งสำคัญที่จะต้องคำนึงถึงก็คือ ชื่อของ Entity name ใน Architecture Body จะต้องสัมพันธ์กัน ดังแสดงในรูปที่ 2.8

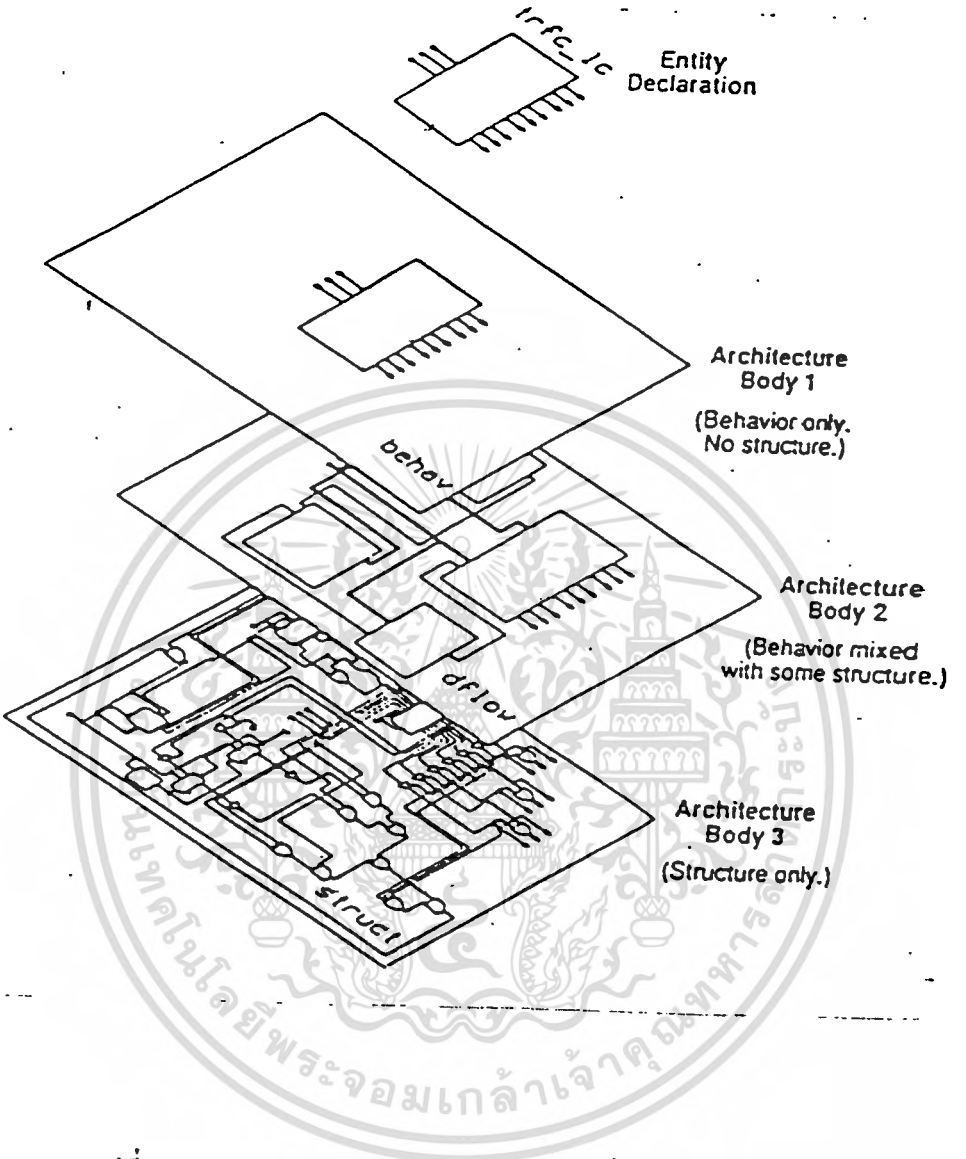
นักออกแบบกำหนดพฤติกรรมการทำงานหรือโครงสร้างของ Design Entity ใน Architecture Body โดยใช้วิธีการเขียนอธิบายโดยใช้ภาษาวีเอชดีแอล(Design Description Methods) ดังรูปที่ 2.9 และ Design Entity ซึ่งมี Architecture Body มากกว่า 1 Architecture โดยนักออกแบบจะต้องเขียน Entity Declaration (ชื่อควรเป็น "Trfc_ic") แล้วก็คอมไพล์ หลังจากนั้นจึงเขียนและคอมไพล์ส่วนของการบรรยายแบบบีเฮฟวีเออร์ของวงจรซึ่ง Architecture name ควรเป็น "Behav" ดังแสดงที่มุมล่างด้านขวาของ Architecture Body 1 เมื่อพอใจกับการทำงาน บีเฮฟวีเออร์ของวงจรที่ออกแบบแล้ว (ที่ระดับ High-Abstraction) เราสามารถที่จะเขียน Architecture Body ขึ้นมาอีกเพื่อเป็นการทดสอบ การทำงานของวงจร ณ ที่

ระดับ Low-Abstraction Architecture 2 รูปที่ 2.9 แสดงโครงสร้างและรายละเอียดการไหลของข้อมูล (Data Flow) ซึ่งมี Architecture name คือ "Dflow" หลังจากนั้นเราจึงทำการ Simulate Design ที่ระดับนี้จากนั้นทำการแก้ไขปรับปรุงจนได้ผลการทำงานที่น่าพอใจ



รูปที่ 2.8 การใช้ชื่อ Entity ใน Entity Declaration และ Architecture Body

ระดับการเขียนอธิบายด้วยภาษาวีเอชดีแอล ขั้นต่ำสุด (Lowest Abstraction Level) คือ การเขียนในแบบโครงสร้าง Structural Description ซึ่งเป็นการอธิบายการทำงานของวงจร ณ ระดับ Component Level Architecture Body 3 รูปที่ 2.9 แสดงให้เห็นถึงการอธิบายในระดับ Structural ชื่อ Architecture นี้คือ "Struct" ในการใช้วิธีการอธิบายที่แตกต่างกัน 3 วิธีนี้ ทำให้เราสามารถพัฒนาออกแบบ Design 1 ตัว โดยใช้วิธีการ Top-Down Design ในแต่ละ Abstraction Level จะถูกเขียนและเก็บไว้ใน Design File แยกจากกัน



รูปที่ 2.9 หลาย Architecture Body สำหรับหนึ่ง Entity Declaration

2.5 วิธีการเขียนอธิบายในรูปแบบต่าง ๆ

Design Description Methodes

ภาษาวีเอชดีแอล เป็นวิธีการเขียนอธิบายการทำงานของฮาร์ดแวร์ ในลักษณะของ Texual Format แทนที่จะใช้ Schematic Diagram เหมือนเมื่อก่อน หัวข้อต่าง ๆ ดังนี้คือ วิธีการเขียนอธิบาย ภาษาวีเอชดีแอลในหลาย ๆ วิธีเพื่อที่จะอธิบาย Hardware Architecture

- Structural Description Method อธิบายตัวรูปแบบ ในรูปแบบของการเชื่อมต่อ Component ต่าง ๆ เข้าด้วยกัน

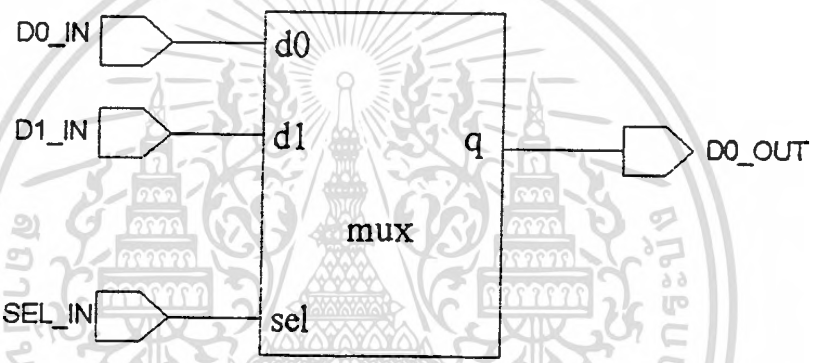
- วิธีการบรรยายแบบบีเฮฟวีเออร์อธิบายฟังก์ชันการทำงานของรูปแบบฮาร์ดแวร์ ในรูปแบบของ Circuit , Signal ที่ตอบสนองกับสัญญาณที่รับเข้ามาจากภายนอก พฤติกรรมการทำงาน (Hardware Behavior) จะถูกอธิบายด้วย Algorithm โดยไม่ว่าจะสร้างขึ้นมาอย่างไร

- Data Flow Description Method มีความคล้ายคลึงกับ Register-Tranfer Language วิธีการนี้อธิบายฟังก์ชันการทำงานของรูปแบบ โดยการกำหนดการไหล (Flow) ของข้อมูลจากอินพุท หรือ รีจิสเตอร์ ไปยังตัวเอาท์พุท หรือ รีจิสเตอร์ อีกตัววิธีการทั้ง 3 แบบที่ใช้อธิบาย Architecture ของ Hardware สามารถอธิบายร่วมกันโดยใช้ทั้ง 3 แบบต่อ 1 รูปแบบก็ได้

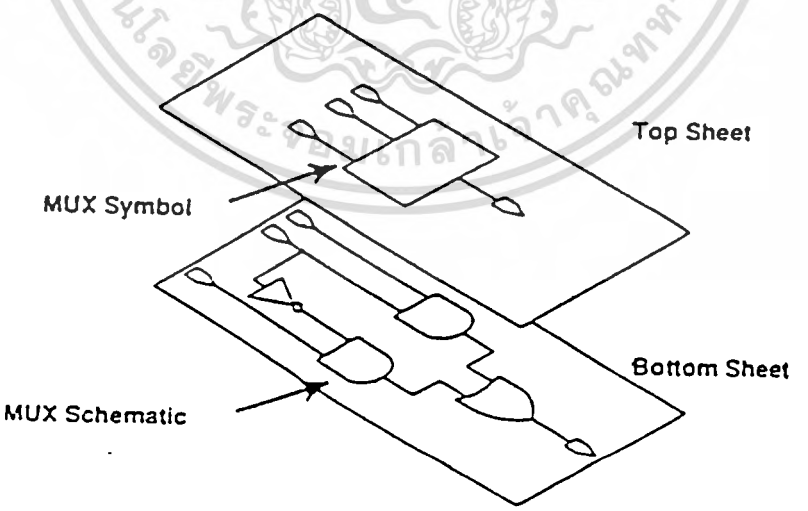
2.5.1 Structural Description

ในส่วนนี้จะอธิบายในส่วนของภาษาเพียงบางส่วนในการที่จะแสดง VHDL Structural Description โดยใช้ตัวอย่างคือ มัลติเพล็กเซอร์ 2 อินพุท จะไม่ได้อธิบายโครงสร้างของภาษาในส่วนนี้ทั้งหมด เพียงต้องการยกตัวอย่างให้เห็นเท่านั้น รายละเอียดทั้งหมดให้ดูได้ที่ VHDL Reference Manual

VHDL Structural Description เป็นวิธีการอธิบายตัวรูปแบบฮาร์ดแวร์ ที่คล้ายกับแสดงโดยใช้ Schematic Diagram เพราะว่าแสดงให้เห็นถึงการเชื่อมต่อของส่วนประกอบ ตัวอย่างต่าง ๆ ที่จะแสดงให้เห็นในส่วนต่อไปนี้ เป็นการเปรียบเทียบวงจรง่าย ๆ 1 วงจร ซึ่งแทนด้วย VHDL และแทนด้วย Schematic Diagram ว่าเป็นอย่างไร



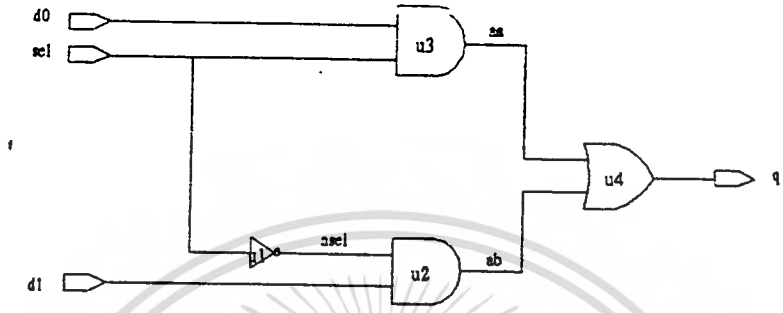
รูปที่ 2.10 สัญลักษณ์แสดงสองอินพุทมัลติเพลกเซอร์



รูปที่ 2.11 การออกแบบแบบ Hierachy บน Schematic Editor ของมัลติเพลกเซอร์

รูปที่ 2.10 แสดงสัญลักษณ์ของ มัลติเพล็กเซอร์ 2 อินพุท วงจร MUX นี้เป็นการออกแบบใจลักษณะของ Hierarchical Design (รูปที่ 2.11) ซึ่งวงจรในระดับล่างสุดเป็น Schematic

Diagram หรืออยู่ในรูปแบบของการอธิบายถึงการเชื่อมต่อภายใน ดังรูปที่ 2.12 (หมายเหตุ - จะสังเกตเห็นว่าชื่อ Pin ใน MUX Symbol ในรูปที่ 2.10 จะตรงกับชื่อ Net ของอินพุท/เอาท์พุท ของ Schematic ในรูปที่ 2.12



รูปที่ 2.12 แสดงระดับเกทของสองอินพุทมัลติเพลกเซอร์

รูปที่ 2.13 แสดง VHDL Structural Description ของมัลติเพล็กเซอร์ 2 อินพุท โดย VHDL Code สามารถเขียนส่วนประกอบ โดยการเขียน Double Dash (--) ข้อความหรืออักษรใด ๆ ที่อยู่หลังจากเครื่องหมายจะถูกมองว่าเป็น Comment ไม่สนใจโดย Compiler (บรรทัด 1,2,5,7,17,19 ถึง 21,24 ในรูปที่ 2.13) Comment ทำให้รหัสอ่านง่าย

```
1 EANTITY mux IS          -- Entity Declaration
2 PORT(d0,d1,set : IN bit ; q : OUT bit);  -- Port Clause
3 END mux;
4
5                         -- Architecture Body
6 ARCHITECTURE  struct OF mux IS
7 COMPONENT and2
8 PORT (A,B : IN bit; c : OUT bit);
9 END COMPONENT;
10 COMPONENT or2
11 PORT (a,b : IN bit; c : OUT bit);
12 END COMPONENT;
13 COMPONENT inv
14 PORT (a : IN bit; c : OUT bit);
15 END COMPONENT;
16
```

```
17 SIGNAL aa,ab,nsel : bit;     --Signal Declaration
18
19 FOR u1 : inv USE ENTITY WORK.invrt(behav); -- config.
20 FOR u2,u3 : and2 USE ENTITY WORK.and_gt(dFlow); -- specif.
21 FOR u4 : or2 USE ENTITY WORK.or_gt(archl); --
22
23 BEGIN
24      u1 : inv PORT MAP (sel,nsel);  -- Architecture Statement Part
25      u2 : and2 PORT MAP (nsel,d1,ab);
26      u3 : and2 PORT MAP (d0,sel,aa);
27      u4 : or2 PORT MAP (aa,ab,q);
28 END struct;
```

รูปที่ 2.13 ตัวอย่างโปรแกรมของ Structural Description สำหรับมัลติเพลกเซอร์

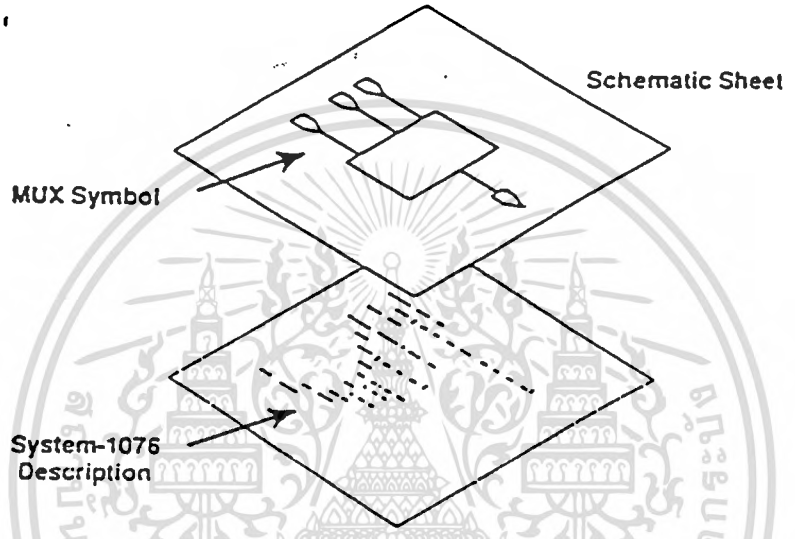
มัลติเพล็กเซอร์ 2 อินพุท แสดงในรูปที่ 2.13 เป็นวงจรพื้นฐาน Entity Declaration อยู่ที่ด้านบน บรรทัด 1 ถึง 3 กำหนดการ Interface Design Entity และสภาพแวดล้อม หรือวงจรอื่น ๆ ภายนอก

Entity Declaration มี Port Clause ซึ่งบอกช่องสัญญาณอินพุท (Input Channels) (d0,d1 และ set) และช่องสัญญาณเอาท์พุท (Output Channels) (q) สัญญาณของแต่ละ Channel ถูกกำหนดให้เป็น บิท คือมีสถานะเป็น 0 กับ 1 ซึ่ง Entity Declaration นี้สามารถเปรียบเทียบกับ MUX Symbol ใน Schematic รูปที่ 2.11 ได้

Architecture Body ในรูปที่ 2.13 (บรรทัดที่ 6 ถึง 28) อธิบายความสัมพันธ์ระหว่าง Design Entity Input และ Output ในลักษณะของโครงสร้าง Architecture นี้สามารถทำงานได้เหมือนกับ Schematic ดังรูปที่ 2.11 Component หลาย ๆ ตัว (AND2 , OR2 , INV) ที่ประกอบรวมกันจนเป็น MUX Design Entity ในรูปที่ 2.13 ถูกประกาศไว้ในส่วนของ Architecture Declaration (บรรทัด 7 ถึง 15) Signal (aa , bb , nset) ถูกประกาศไว้ใน Architecture Body (บรรทัดที่ 17) ด้วยเช่นกัน เพื่อที่จะแทน เอาท์พุทของ AND เกท 2 ตัว (U2 , U3) และ Inverter (U1)

Configuration Specification ในบรรทัดที่ 19 ถึง 21 เชื่อมเอาส่วนประกอบแต่ละตัวที่ประกาศไว้เข้ามายัง Design Entity เพื่อบอก Design Entity ว่าส่วนประกอบแต่ละตัวทำงานอย่างไร ยกตัวอย่างเช่น ส่วนประกอบ U1 ในบรรทัดที่ 24 รูปที่ 2.13 ถูกกระโดดมายัง Architecture Body ที่ชื่อ "Behav" สำหรับ Design Entity ที่เรียกว่า Invrt

Architecture Statement Part (บรรทัดที่ 24 ถึง 27) อธิบายการเชื่อมต่อ (Connection) ระหว่างส่วนประกอบที่ประกอบด้วย Design Unit ในส่วนนี้จะมีการประกาศการใช้ Component รูปที่ 2.14 แสดงให้เห็นว่า Schematic Sheet ที่มี MUX Symbol มีความเกี่ยวข้องกับ VHDL Structural Description อย่างไร แทนที่จะใช้การลากเส้นใน Schematic Sheet VHDL Structure Description กำหนดการเชื่อมต่อภายในของส่วนประกอบ



รูปที่ 2.14 สองอินพุทมัลติเพลกเซอร์ซึ่งมี Structural Description ที่เกี่ยวข้องกัน

2.5.2 Behavioral Description

ในส่วนนี้จะเป็นการอธิบายถึงส่วนสำคัญของภาษาส่วนหนึ่ง นั่นคือ Behavioral Description โดยใช้วงจรซึ่งได้ยกตัวอย่างมาแล้วก่อนหน้านี้ คือ MUX และ 4 Bit Shifter หลังจากอ่านส่วนที่ได้อธิบายไปก่อนหน้านี้ คือ Structural Description เราสามารถเปรียบเทียบกันได้ระหว่างการอธิบายด้วย Behavioral Description กับ Structural Method ว่าแตกต่างกันอย่างไร

VHDL Behavioral Description แทนฟังก์ชันการทำงานของรูปแบบ ในรูปแบบของวงจร และสัญญาณที่ตอบสนองต่อการกระตุ้นการจากภายนอก แสดงในรูปที่ 2.11 ถึง 2.14 พฤติกรรมการทำงานของ MUX ถูกกำหนดด้วยการเชื่อมต่อระหว่าง Inverter , AND เกท และ OR เกท ซึ่งฟังก์ชันของเกทแต่ละตัวนี้เป็นที่เข้าใจกันดีอยู่แล้ว ในรูปแบบที่มีความซับซ้อนมากขึ้น ส่วนประกอบ U1 ถึง U5 ในรูปที่ 2.13 ต้องสามารถแทนด้วย Entity ที่มีฟังก์ชันการทำงานแต่ละส่วนประกอบ ด้วย Behavioral Description

VHDL Description แสดงในรูปที่ 2.14 แสดง Behavioral Description แทนที่จะเป็นแบบ Structure Description เหมือนหัวข้อก่อนหน้านี้ ในครั้งนี้เราสามารถวาง MUX Symbol ลงใน Schematic Sheet แต่ในที่นี้เราใช้ Behavioral ของส่วนประกอบระหว่างการทำ Circuit ซิมูลเลชัน รูปที่ 2.15 แสดงภาษาวีเอชดีแอลโค้ด ซึ่งกำหนดการทำงานของ MUX ในรูปแบบ Behavioral

Behavioral Description ในรูปที่ 2.15 และ Structural Description ในรูปที่ 2.13 ทั้งคู่มี Entity Declaration และ Architecture Body เช่นกัน ในทางปฏิบัติเราไม่จำเป็นจะต้องเขียน 2 Architecture Body ไว้ในไฟล์เดียวกัน (ซึ่งสามารถทำได้) เราควรเขียน Entity Declaration ไว้ที่ไฟล์หนึ่ง แล้วเขียน Behavioral Description ไว้อีก File หนึ่ง และ Structural Description ไว้อีกไฟล์เช่นกัน ในการออกแบบจริง ๆ หลังจากการเขียน (Write) และ คอมไพล์ Entity Declaration เสร็จเรียบร้อยแล้ว ขั้นตอนต่อไปควรเขียน Behavioral Architecture เป็นขั้นตอนต่อไปในการที่จะทดสอบการทำงานของวงจรโดยรวมทั้งหมด หลังจากนั้นเราจึงทำการเขียน Simulate และ Refine ฟังก์ชันการทำงานของโมเดลจนกว่าจะทำงานได้ถูกต้อง จึงจะทำการเขียน Structural Architecture จากนั้นก็เปลี่ยน Structural Description เข้าไปแทนที่ Behavioral Description เพื่อที่จะทำการ Simulate ดูอีกครั้ง

```
1 ENTITY mux IS              -- Entity Declaration
2 PORT (d0,d1,sel : IN bit; q : OUT bit);   --Port Clause
3 END mux;
4
5 ARCHITECTURE behav OF mux IS
```

```
6 BEGIN
7 f1 :                          -- Process Statement
8 PROCESS (d0,d1,sel)           -- Sensitivity List
9 BEGIN
10 IF sel = '0' THEN            -- Process Statement Part
11 q <= d1;
12 ELSE
13 q <= d0;
14 END IF;
15 END PROCESS f1;
16 END behav;
```

รูปที่ 2.15 ตัวอย่างโปรแกรมของ Behavioral Description สำหรับมัลติเพลกเซอร์

Behavior Description Mode มีประโยชน์ต่อการกำเนิดสัญญาณอินพุทขึ้นมาทดสอบ VHDL โมเดล ในส่วนอื่น ๆ เมื่อต้องการทำซิมูลเลชัน เช่น เราต้องการออกแบบ Traffic Light Controller โดยใช้ Structural Description และเราต้องการจะทดสอบโมเดลนั้นโดยที่ Traffic Light Controller นั้น อินพุทจะต้องรับจาก Sensor ที่ต่อเข้ามา สำหรับการซิมูลเลชัน เราต้องเขียน Behavior Description โมเดล ขึ้นมาเพื่อทำจำลองสัญญาณที่ออกจาก Sensor (แทนการสร้างจริง ๆ)

ข้อแตกต่างที่เห็นได้ชัดระหว่าง Structural และ Behavioral Description ของ MUX คือ Architecture Body ดังรูปที่ 2.15 มี Process Statement ซึ่งแทนการทำงานที่เป็นอิสระกำหนดพฤติกรรมการทำงานของฮาร์ดแวร์ หรือส่วนใดส่วนหนึ่งของรูปแบบ รูปแบบของการเขียน Process Statement แสดงดังต่อไปนี้

```
Process Statement.....label:
                    Process (sensitivity_list)
                            Process_declarative_part
                    begin
                            Process_Statement_part
                    end Process label;
```

Process Statement ในรูปที่ 2.15 Process Label f1 ตามด้วยเครื่องหมาย Colon ; (บรรทัดที่ 7) Process Label เป็น Optional แต่มีประโยชน์ในการที่จะช่วยแยกให้เห็นความแตกต่าง Process หลายตัวให้รูปแบบใหญ่ ๆ

ในวงเล็บที่ต่อจาก "Process" คือ Option Sensitivity List ดังรูปที่ 2.15 (บรรทัดที่ 8) ประกอบไปด้วยสัญญาณ (d0 , d1 , sel) ระหว่างการทำซิมูลเลชัน ถ้าสัญญาณใดสัญญาณหนึ่งใน Sensitivity List เกิดการเปลี่ยนแปลง State , Process จะทำการ Execute และ State ของเอาท์พุท ก็จะเป็นไปตามเช่นกัน แต่ละ Process ในการออกแบบวีเอชดีแอล จะ Execute 1 ครั้งระหว่างการทำ Initialize VHDL Hardware โมเดล

หัวใจสำคัญของ Process Statement ในรูปที่ 2.15 คือ If Statement ที่อยู่ภายใน Process Statement Part รูปแบบพื้นฐานของการเขียน If Statement แสดงดังข้างล่าง

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

If Statement ในภาษาวีเอชดีแอล จะถูกตีความคล้าย ๆ กับประโยคในภาษาอังกฤษ ยกตัวอย่างจากประโยคข้างล่าง

If the traffic light is green then proceed across the intersection or else (if the traffic light is not green) remain stopped.

ในประโยคนี้จะอยู่ในสภาวะทำงานก็ต่อเมื่อไฟเป็นสีเขียว ก่อนที่คำสั่งต่าง ๆ จะถูก Execute "else" Statement จะเป็นทางเลือกอีกทางหนึ่ง เมื่อสถานะอื่นที่ไฟไม่เป็นสีเขียวตรงตามเงื่อนไข

If Statement ในรูปที่ 2.15 (บรรทัดที่ 10 ถึง 14) สามารถเขียนได้ใหม่ดังรูปต่อไปนี้

if Signal sel(select) is equal to 0 , then assign the value of the Waveform on Signal d1 to target Signal q or else assign the value of the Waveform on Signal d0 to target Signal q.

เมื่อใดก็ตามที่สภาวะภายใน if condition หรือ else condition ในตัวอย่าง ได้รับสภาพความตรงตามเงื่อนไข (sel = '0' หรือสภาวะตรงข้ามของ sel <> '0') target Signal q จะถูก Modified ตามความเหมาะสม ซึ่งขึ้นอยู่กับ Signal Assignment Statement รูปแบบพื้นฐานของ Signal Assignment มีดังนี้

Signal Assignment Statement : target (= transport Waveform)

(Note transport, Optional)

Signal Assignment Statement ประโยคแรกในรูปที่ 2.15 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;
```

จะทำการ Assign Waveform ให้กับสัญญาณ q การใช้งานอีกอย่างของ compound delimeter คือใช้เป็น relational operator "น้อยกว่าหรือเท่ากับ" เช่นประโยคตัวอย่างดังนี้

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

สรุปว่า Signal Assignment delimeter <= ใช้สำหรับ Assign ค่าที่อยู่ทางขวามือให้กับสัญญาณที่อยู่ด้านซ้ายมือ และ delimeter ตัวเดียวกันนั้นก็ใช้ในการทำ relational operator เปรียบเทียบ "น้อยกว่าหรือเท่ากับ" ใช้ในการเปรียบเทียบ condition ใน If Statement ตัวอย่างในรูปที่ 2.16 แสดงการเขียน VHDL Behavioral Description ของ 4 bit Shifter เพื่อแสดงให้เห็นว่า accurate และ Succinet ของ VHDL Description เป็นอย่างไรเมื่อเปรียบเทียบกับรูปแบบของ textual Description ดังนี้

The four bit shifter has four input data lines and two control lines. When both control lines 1 are low , the input levels are passed directly to the corresponding output. When control line 0 is high and control line is low , output line 0 is low; input line 0 is passed to output line 1; input line 1 is passed to output line 2; and input line 2 is passed to output line 3.

When control line 0 is low and control line 1 is high; input line 2 is passed to output line 1; input line 3 is passed to output line 2; and output line 3 is low. When both control lines are high , input line 0 is passed to both output line 0 and line 1; input line 1 is passed to output line 2; and input line 2 is passed to output line 3.

```
1 ENTITY Shifter IS                -- Entity Declaration
```

```
2 PORT (shftin     : IN bit_vector(0 to 3);   -- Port Clause
3          shftout   : OUT bit_vector (0 to 3);
4          shftctl   : IN bit_vector (0 to 1));
5 END Shifter;
6
7 ARCHITECTURE  behav OF Shifter IS     -- Architecture Body
8 BEGIN
9     ,  f2 :                  -- Process Statement
10 PROCESS(shftin , shftctl)
11 VARIABLE Shifted : bit_vector(0 to 3)     -- Process Declaration Part
12 BEGIN
13 CASE shftctl IS
14        WHEN "00" => shifted := shftin;
15        WHEN "01" => shifted := shftin(1 to 3) & '0';
16        WHEN "10" => shifted := '0' & shftin(0 to 2);
17        WHEN "11" => shifted := shftin(0) & shftin(0 to 2);
18 END CASE;
19 shftout <= shifted AFTER 10 ns;
20 END PROCESS f2;
21 END behav;
```

รูปที่ 2.16 ตัวอย่างโปรแกรมของ Behavioral Description สำหรับ Shifter

2.5.3 Structural and Behavioral Description Summary

สรุป Strustural และ Behavioral Description ได้ดังนี้

VHDL Structural Description เป็นการกำหนดการเชื่อมต่อส่วนประกอบต่าง ๆ ในวงจรที่เราสร้างขึ้น ส่วน Behavioral Description เป็นการอธิบายถึง Algorithm ของวงจร และการทำงานของ Input/Output ภายในว่ามีการตอบสนองอย่างไรต่อสัญญาณภายนอก Design Entity เป็น Unit พื้นฐานของ Hardware Description ใช้แทนการทำงานของ เซล , ชิป , บอร์ด หรือระบบย่อย ทั้ง Structural และ Behavioral Description มีการประกาศ Design Entity โดยใช้ Entity Declaration ส่วน Architecture Body ของแต่ละ Entity มีไว้เพื่ออธิบายความสัมพันธ์ระหว่างอินพุทและเอาท์พุท ของ Entity นั้น ๆ

Structural และ Behavioral Description มีความแตกต่างกันอย่างเห็นได้ชัดในส่วนของ Architecture Body ดังแสดงไว้ในรูปที่ 2.17 เปรียบเทียบตัวอย่าง MUX Architecture Body ของ Structural Description (ส่วนบนของรูปที่ 2.17) มีส่วนของ Architecture Statement Part ซึ่งอธิบายการเชื่อมต่อของ Component ที่อยู่ในท Design Entity ส่วน Architecture Body ของ Behavioral Description (ส่วนล่างของรูปที่ 2.17) มีส่วนของ Process Statement ซึ่งอธิบายการทำงานของ Design Entity นั้น ๆ

```
1 ENTITY mux IS  -- STRUSTURAL ---------- Entity Declaration
2 PORT (d0,d1,sel : IN bit; q : OUT bit); -- Port Clause
3 End mux;
4
5                    -- Architecture Body
6 ARCHITECTURE struct OF mux IS
7 COMPONENT and2
8 PORT(a,b : IN bit; c : OUT bit);
9 END COMPONENT;
10 COMPMNENT or2
11 PORT (a,b : IN bit; c : OUT bit);
12 END COMPONENT;
13 COMPONENT inv
14 PORT (a : IN bit; c : OUT bit);
15 END COMPONENT;
16
17 SIGNALaa,ab,nsel : bit;  -- Signal Declaration
```

```
18
19 FOR u1 : inv USE ENTITY WORK.invrt(behav);   -- Config.
20 FOR u2,u3 :and2 USE ENTITY WORK.and_gt(dflow);   -- Specif
21 FOR u4 : or2 USE ENTITY WORK.or)gt(archl);   --
22
23 BEGIN
24        u1 : inv PORT MAP (sel,nsel);   -- Architecture Statement Part
25        u2 : and2 PORT MAP (nsel,d1,ab);
26        u3 : and2 PORT MAP (d0,sel,aa);
27        u4 : or2 PORT MAP (aa,ab,q);
28 END struct;
```

```
1 ENTITY mux IS ----- BEHAVIORAL ----------Entity Declaration
2 PORT (d0,d1,sel : IN bit; q ; OUT bit);   -- Port Clause
3 END mux;
4
5 ARCHITECTURE behav OF mux IS
6 BEGIN
7        f1 :                              -- Process Statement
8        PROCESS (d0,d1,sel)               -- Sensitivity List
9        BEGIN
10               IF sel = '0' THEN         -- Process Statement Part
11               q <= d1;
12               ELSE
13               q <= d0;
14               END IF;
15       END PROCESS f1;
16 END behav;
```

รูปที่ 2.17 เปรียบเทียบตัวอย่างมัลติเพลกเซอร์แบบ Structural และ Behavioral Description

2.5.4 Data Flow Description

VHDL Data Flow Description และ Register Transfer Language มีความคล้ายคลึงกัน คือ ทั้งคู่อธิบายการทำงานของรูปแบบ โดยกำหนดการไหลของข้อมูลจากอินพุทหนึ่ง หรือรีจิสเตอร์หนึ่ง ไปยังเอาท์พุทหรืออีกรีจิสเตอร์หนึ่ง Data Flow และ Behavioral Description มีความคล้ายคลึงกันคือ ทั้งคู่ใช้ Process เพื่อที่จะอธิบายฟังก์ชันการทำงานของวงจร Behavioral Description ใช้จำนวน Process น้อยกว่าโดยภายในแต่ละ Process ใช้ลักษณะของซีเควนเชียล Signal Assignment หลาย ๆ คำสั่งภายใน Process ในทางตรงกันข้าม Data Flow Description ใช้จำนวนของคอนเคอร์เรนท์ Signal Assignment มาก คอนเคอร์เรนท์Statement ใช้ใน Data Flow Description ประกอบด้วย

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

นอกจากนี้ Process Statement , Genetate Statement และ Component Instamtiation Statement เป็น คอนเคอร์เรนท์ Statement ด้วยเหมือนกัน ซึ่งโครงสร้างเหล่านี้ไม่ค่อยพบในการอธิบายแบบ Data Flow Description

คอนเคอร์เรนท์ Statement กำหนดการเชื่อมต่อ Process Blocks ซึ่งทั้งหมดรวม ๆ กัน จะอธิบายการทำงานของ Design คอนเคอร์เรนท์ Statement ทำการ Execute แบบ Asynchronous ร่วมไปกับ คอนเคอร์เรนท์ Statement อื่น ๆ

รูปที่ 2.18 แสดงให้เห็นรูปแบบการเขียนอธิบาย Mux โดยวิธี Data Flow Description ซึ่งก่อนหน้านี้ใช้ Behavioral และ Structural Description เขียนอธิบายมาแล้ว ตัวอย่างนี้ง่ายเกินไปที่จะแสดงให้เห็นถึงประโยชน์จริง ๆ ของ Data Flow Description เพราะว่ามีความคล้ายคลึงกับ Behavioral Description ในรูปที่ 2.15 มาก โดยทั้งสองตัวอย่างใช้ Process Statement แสดงด้วย คอนเคอร์เรนท์ Statement ในรูปที่ 2.18 (บรรทัดที่ 8 ถึง 10) เพื่อกำหนด Signal Behavior

```
1 ENTITY mux IS
2 PORT (d0,d1,sel : IN bit; q : OUT bit);     -- Port Clause
3 end MUX;
4
5
6 ARCHITECTURE data_flow OF mux IS
```

```
7 BEGIN
8       cls :
9       q <= d1 WHEN sel = '0' ELSE    -- Condition Signal Assignment
10      d0;
11 End data_flow;
```

รูปที่ 2.18 ตัวอย่างโปรแกรมของ Data Flow Description ของมัลติเพลกเซอร์

Data Flow Description ประกอบไปด้วย Entity Declaration (บรรทัดที่ 1 ถึง 3) เช่นเดียวกันกับตัวอย่างใน Structural และ Behavioral Description ที่อธิบายมาแล้วก่อนหน้านี้ ส่วนของ Architecture Body ประกอบไปด้วย คอนเคอร์เรนท์ Signal Assignment Statement ซึ่งทำหน้าที่เทียบอท่ากับ Process Statement (ซึ่งมีความหมายเหมือนกัน) รูปแบบของการเขียนคอนเคอร์เรนท์ Signal Assignment แสดงดังต่อไปนี้

```
Concurrent Signal Label : Conditional_signal_assignment
Assignment Statement.....
                label : selected)signal_assignment
```

ในรูปที่ 2.18 (บรรทัดที่ 9 และ 10) Conditional Signal Assignment ทำหน้าที่เป็น Signal Assignment (q < d1 หรือ q < d0) ขึ้นอยู่กับสถานะที่กำหนดใน Condition Waveform รูปแบบของการทำ Conditional Waveform มีดังนี้

```
Conditional Signal      target <= options conditional_waveform;
Assignment
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 เพื่อการเปรียบเทียบ Behavioral Description ของ 4 bit Shifter รูปที่ 2.16 แสดงให้เห็นอีกครั้งในรูปที่ 2.19 (ด้านบนของรูป) พร้อมทั้ง Data Flow Description ซึ่งอธิบาย 4 bit Shifter เช่นกัน (ด้านล่างรูป)

ข้อแตกต่างซึ่งเห็นได้ชัดระหว่างการอธิบายโดยใช้ 2 วิธีคือ 4 Process Statement ถูกแสดงเป็นนัย ๆ ใน Data Flow Description และ 4 Cnoditional Signal Assignment (บรรทัดที่ 9 ถึง 20) ในวิธี Behavioral Description มี 1 Process Statement ใช้อย่างเห็นได้ชัด (บรรทัดที่ 9 ถึง 20)

ตัวอย่างของ Data Flow Description ในรูปที่ 2.19 มีการใช้ Entity Declaration เช่นเดียวกับ Behavioral Description (บรรทัดที่ 1 ถึง 5) Architecture Body ใน Data Flow Description ใช้ คอนเคอร์เรนท์ Signal Assignment ซึ่งประกอบด้วย 4 Conditional Signal Assignment มีสำหรับแต่ละ Element ของ Shiftout Array (คอนเคอร์เรนท์ Signal Assignment Statement ไม่ได้ใช้ Optional Label เหมือนกับที่ใช้ในรูปที่ 2.18 บรรทัดที่ 8)

```
1 ENTITY shifter IS  -- BEHAVIORAL ---------- entity declaration
2 PORT (shftin : IN bit_vector(0 to 3);     --Port Clause
3         shftout : OUT bit_vector(0 to 3);
4         shftctl : IN bit_vector(0 to 1));
5 END shifter;
6
7 ARCHITECTURE behav OF shifter IS          -- Architecture Body
8 BEGIN
9       f2:                                 -- Process Statement
10      PROCESS(shftin,shftctl)
11      VARIABLE shifted : bit_vector(0 to 3)       -- Process Declaration Part
12      BEGIN
13              CASE shftctl IS
14                      WHEN "00" => shifted := shftin;
15                      WHEN "01" => shifted := shftin(1 to 3) & '0';
16                      WHEN "10" => shifted := '0' & shftin(0 to 2);
17                      WHEN "11" => shifted := shftin(0) &shftin(0 to 2);
18              END CASE;
19              shftout <= shifted AFTER 10 ns;
20      END PROCESS f2;
21 END behav;
```

```
1 ENTITY shifter IS -- DATA FLOW ---------- Entity Declaration
2 PORT (shftin : IN bit_vector(0 to 3);    -- Port Clause
3         shftout : OUT bit_vector(0 to 3);
4         shftctl : IN bit_vector(0 to 1));
5 END shifter;
6
7 ARCHITECTURE data_flow OF shifter IS  -- Architecture Body
8 BEGIN
9       shftout(3) <= '0' AFTER 10 ns WHEN shftctl = "01" ELSE
10                shftin(3) AFTER 10 ns WHEN shftctl + "00" ELSE
11                shftin(2) AFTER 10 ns;  -- End Cond. Sig. Assign.1
12      shftout(2) <= shftin(3) AFTER 10 ns WHEN shftctl = "01" ELSE
13                shftin(2) AFTER 10 ns WHEN shftctl = "00' ELSE
14                shftin(1) AFTER 10 ns;  -- End Cond. Sig. Assign.2
15      shftout(1) <= shftin(2) AFTER 10 ns WHEN shftctl = "01" ELSE
16                shftin(1) AFTER 10 ns WHEN shftctl = "00" ELSE
17                shftin(0) AFTER 10 ns;  -- End Cond. Sig. Assign.3
18      shftout(0) <= shftin(1) AFTER 10 ns WHEN shftctl = "01" ELSE
19                '0' AFTER 10 ns WHEN shftctl = "10" ELSE
20                shftin(0) AFTER 10 ns;  -- End Cond. Sig. Assign.4
21 End data_flow;
```

รูปที่ 2.19 เปรียบเทียบตัวอย่าง Shifter แบบ Behavioral และ Data Flow Description

2.6 สรุป

- ภาษาวีเอชดีแอลคือตัวภาษาที่ออกแบบมาเฉพาะเพื่อใช้อธิบายการทำงานของ Hardware ให้อยู่ในรูปที่สามารถอ่านทำความเข้าใจได้ สามารถอธิบายได้ถึงการจัดระบบและการทำงานของวงจรดิจิตอล , วงจรระดับบอร์ด และ อุปกรณ์ต่าง ๆ

- เหตุผลที่ทำให้ภาษาวีเอชดีแอล ใช้ในการออกแบบและจำลองการทำานของ Product ตัวหนึ่งซึ่งยังไม่ได้สร้างจริง ๆ เพื่อดูการทำงานก่อนลงมือสร้าง หรืออาจใช้เป็นตัวแทนแนวคิดใน Product นั้น ๆ มีดังนี้

1. ภาษาวีเอชดีแอล อนุญาตให้เราออกแบบ จำลองการทำงาน และทดสอบระบบโดยใช้รูปแบบของภาษาระดับสูงจนถึงระดับเกทเลเวล

2. ภาษาวีเอชดีแอล ถ้าเราเขียนตามรูปแบบของ VHDL Synthesis Guide จะทำให้เราสามารถใช้วีเอชดีแอลโค้ด นั้นไปทำการสร้างวงจรได้โดยใช้ VHDL Synthesis Tools

3. เพราะว่าภาษาวีเอชดีแอล เป็นภาษาที่กำหนดเป็นมาตรฐาน IEEE 1076 - 1987 IEEE Standard VHDL Reference Manual วิศวกรหรือผู้ออกแบบสามารถใช้ภาษานี้ในการพัฒนาได้เหมือนกัน ลดปัญหาความเข้ากันไม่ได้ลงไป

- ภาษาวีเอชดีแอล มีคุณสมบัติที่ดีที่ทำให้เราสามารถเขียนและแก้ไขวงจร Digital ที่มีขนาดใหญ่และซับซ้อนได้อย่างสะดวกรวดเร็ว และมีประสิทธิภาพ ดังนี้

1. Top Down Design วิธีการนี้ให้เราสามารถอธิบายการทำงานของระบบได้ในลักษณะของ Block ใหญ่ ๆ จากนั้นทำการวิเคราะห์จำลองการทำงานและแก้ไขให้ได้คุณสมบัติตามที่เราต้องการ ณ ระดับ Block ก่อนที่จะลงลึกในระดับต่ำต่อไป

2. Modularity วิธีการที่แยกส่วน (หรือการประกอบส่วนย่อย ๆ ขึ้นมา) วงจรที่เราออกแบบออกเป็นส่วนย่อย ๆ เล็ก ๆ ออกมา

3. Abstraction รายละเอียดใน Module ซึ่งอธิบายการทำงานของ Module มากกว่าที่จะอธิบายถึงการพัฒนาและการสร้าง Module นั้น

4. Information Hiding การพัฒนา Module ใหม่จาก Module อื่น ๆที่สร้างขึ้นมาแล้ว

5. Uniformility สร้าง Module โดยใช้ตัวภาษา VHDL Building Blocks

บทที่ 3

DES

DES เป็นวิธีการเข้ารหัสข้อมูล ซึ่งได้รับการเผยแพร่ในปี 1977 โดย National Bureau of Standards ประเทศสหรัฐอเมริกา (ปัจจุบันเปลี่ยนเป็น National Institute of Standards and Technology) โดยมีจุดมุ่งหมายเพื่อใช้ทำการเข้ารหัสข้อมูลทางธุรกิจและข้อมูลของรัฐบาลที่ไม่เกี่ยวข้องกับความลับระดับประเทศ อัลกอริทึมของ DES ดัดแปลงมาจากอัลกอริทึมของ IBM ที่ชื่อว่า Lucifer Cipher ซึ่งได้ทำการพัฒนาต่อร่วมกับ National Security Agency (NSA) ทำให้ DES เป็นที่ยอมรับโดยทั่วไปและยอมรับเป็นมาตรฐานของ ANSI (American National Standards Institute) ในปี 1980 โดยในระยะเริ่มต้นมีข้อกำหนดสำหรับ DES ดังนี้

- ต้องมีระดับความปลอดภัยสูง
- ต้องมีการระบุรายละเอียดอย่างสมบูรณ์ และง่ายต่อความเข้าใจ
- ตัวอัลกอริทึมจะต้องมีความปลอดภัย และความปลอดภัยจะต้องไม่ขึ้นกับความปลอดภัยของอัลกอริทึม
- ผู้ใช้ทุกคนสามารถนำไปใช้และดัดแปลงสำหรับ Application ต่างๆ ได้
- ต้องมีประสิทธิภาพและประหยัด เพื่อจะนำไปใช้ในอุปกรณ์อิเลคทรอนิคส์
- ต้องสามารถตรวจสอบความถูกต้องได้

DES เป็นวิธีการเข้ารหัสที่ใช้ Private Key โดยรับข้อมูลจำนวน 64 บิท และใช้ Key ขนาด 64 บิท แต่จะตัดทุกบิทที่ 8 ออก โดยใช้เป็น Parity bit DES เป็นวิธีที่มีประสิทธิภาพในการนำไปใช้แบบฮาร์ดแวร์ แต่จะช้าลงถ้านำไปใช้แบบซอฟท์แวร์

3.1 ความปลอดภัยของ DES

ถ้ามีข้อมูล 1 ชุด ซึ่งประกอบด้วยข้อมูลที่จะเข้ารหัส(เรียกว่า Plaintext) และข้อมูลที่เข้ารหัสแล้ว(เรียกว่า Ciphertext) การ Break DES ในกรณีนี้คือการหา Key ซึ่งจะ Map Plaintext ไปเป็น Ciphertext และด้วย DES ที่นำไปใช้แบบซอฟท์แวร์ ต้องใช้เวลาครึ่งปี ถ้าใช้ Chip ที่ Run ได้ 1ล้าน MIP เพื่อทำการหา Key ด้วยวิธีการ brute-force (ซึ่ง Key ที่ได้ อาจเป็น Key ที่ผิดซึ่งสามารถให้ข้อมูลเหมือนกันได้)

ปกติผู้เจาะระบบมักไม่มีชุด Plaintext และ Ciphertext แต่มักจะมี Ciphertext จำนวนมากแทน และเป็นที่รู้กันว่า ข้อมูลที่จะนำมาเข้ารหัสมักจะเป็น ASCII 7 บิท ในกรณีนั้น การหาแบบ Brute force ยังมีประสิทธิภาพโดย Ciphertext จะถูกถอดรหัสโดยการเดา Key และถ้าทุกบิทที่ 8

เป็น 0 (ซึ่งจะเกิดขึ้นกับ Key ซึ่งไม่ถูกต้องด้วยความน่าจะเป็น 1 ใน 256) แล้วข้อมูลถูกถอดรหัสหลังจากข้อมูลถูกถอดรหัสหลายๆ ชุด และผลที่ได้อยู่ในรูป ASCII 7 บิท Key ที่ได้มักจะมีความเป็นไปได้สูงที่จะถูกต้อง

DES Chip ที่ดีจะไม่ทำให้ Chip เองมีส่วนช่วยในการหา Key โดยจะอนุญาตให้เข้ารหัสข้อมูลจำนวนมากกับ Key 1 ชุด โดยความเร็วในการ Load Key จะน้อยกว่าความเร็วในการเข้ารหัสข้อมูลมาก อาจจะมีความเป็นไปได้ในการออกแบบและผลิต Chip ที่จะทำการหา Key แต่เป็นความโชคดีที่มันเป็นเรื่องยากสำหรับพวกที่ชอบเจาะระบบเป็นงานอดิเรก และเป็นการยากที่จะทำโดยปกปิดเป็นความลับ แต่ก็ไม่ง่ายที่จะทำอย่างเปิดเผย

ในปี 1977 นาย Diffie และ Hellman ทำการวิเคราะห์รายละเอียดของค่าใช้จ่ายเพื่อสร้างเครื่องจักรที่จะใช้ทำการ Break DES และได้ผลสรุปว่าจะต้องใช้ค่าใช้จ่าย 20 ล้านดอลล่าร์ เพื่อสร้างเครื่องจักรที่มี Chip ล้านตัว เพื่อหา DES Key ในเวลา 12 ชั่วโมง (ให้ Plaintext และ Ciphertext มา) ในปีที่ผ่านมา (1995) ประมาณว่าความก้าวหน้าในการสร้าง Chip และความเร็วของ Chip จะทำให้เครื่องที่มี Chip หลายพันตัวทำงานอย่างเดียวกัน โดยใช้ค่าใช้จ่ายน้อยกว่า 1 ล้านดอลล่าร์

มีรายงานที่ได้รับการตีพิมพ์แนะนำว่า ผู้เจาะระบบสามารถ Break DES ได้เร็วกว่าวิธีการหา Key อย่างไรก็ตามผู้เจาะระบบเหล่านี้จะต้องมีชุด Plaintext และ Ciphertext จำนวนมากที่สัมพันธ์กัน

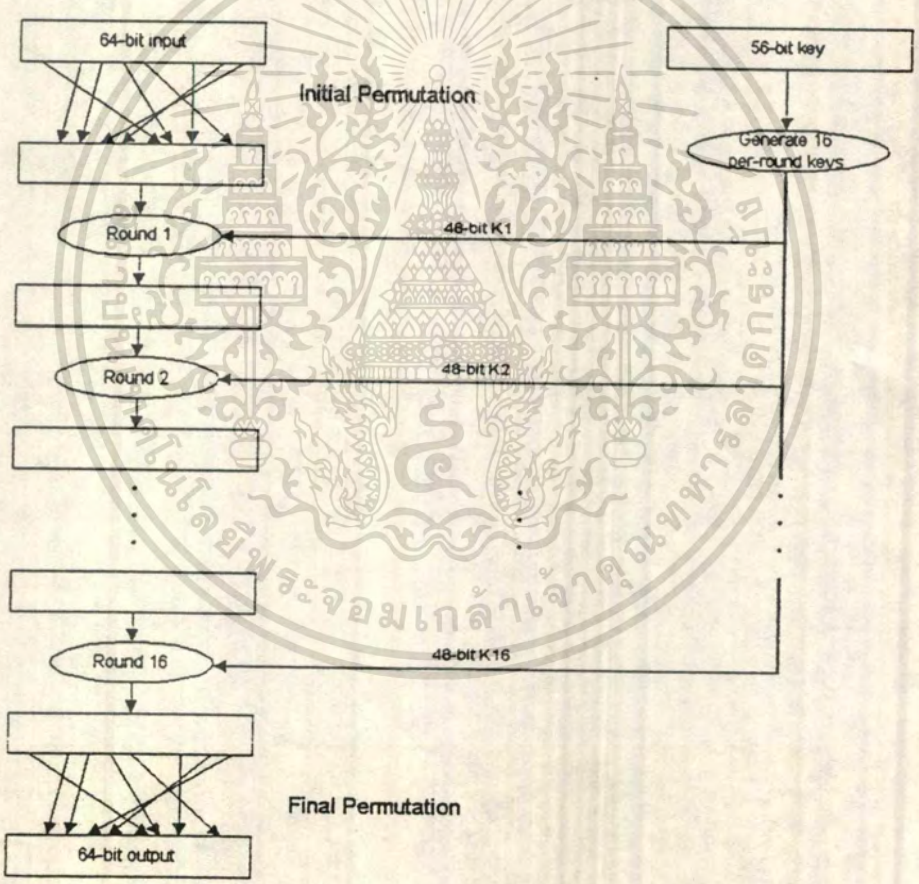
ในความเป็นจริง DES ยังเหมาะสำหรับทุก Application อย่างสมบูรณ์ เนื่องจากราคาของเครื่องจักรที่จะ Break DES ยังแพงเกินไป สำหรับพวกเจาะระบบเป็นงานอดิเรก และถึงแม้องค์การผิดกฎหมายต่างๆ สามารถที่จะทำการซื้อเครื่องจักรที่จะทำการ Break DES ได้ แต่การปล้นธนาคารยังใช้วิธีการที่ง่ายกว่ามาก

ดังนั้น NSA ได้สร้างเครื่องจักรที่จะทำการ Break DES ความจริงอาจกล่าวได้ว่าไม่ใช่หน้าที่ของ NSA แต่ NSA มีงบประมาณเหลือเฟือที่จะไม่นำเครื่องที่สร้างนี้ไปประกอบอาชญากรรม และ NSA ไม่ใช่หน่วยงานที่เกี่ยวข้องกับกฎหมาย เพราะฉะนั้นแม้กระทั่งการนำ DES ไปใช้ทางด้านอาชญากรรมยังมีความปลอดภัยสูง และถึงแม้เครื่องจักรที่จะทำการ Break DES จะอยู่ในงบประมาณของหน่วยงานอื่นของรัฐบาลสหรัฐ (เช่น FBI) แต่มันไม่ง่ายที่หน่วยงานต่างๆ จะทำการสร้างเครื่องดังกล่าวโดยปิดเป็นความลับ

ถ้าการ Break DES ง่ายขึ้น เราอาจจะทำการเข้ารหัสหลายครั้งโดยใช้ Key ต่าง ๆ (Multiple Encryption DES) เป็นที่เชื่อกันว่า DES ที่ทำการเข้ารหัส 3 ครั้ง จะยากในการ Crack ระบบ เป็น 2^{56} เท่า และจะมีความปลอดภัยเพียงพอสำหรับอนาคต

3.2 ภาพรวมของ DES

DES เป็นระบบที่ค่อนข้างเข้าใจได้ง่าย จากรูปที่ 3.1 เริ่มจากโครงสร้างพื้นฐานของ DES ข้อมูลอินพุทขนาด 64 บิท จะถูกส่งผ่านการทำ Initial Permutation ซึ่งจะได้ผล 64 บิท (เป็นการสลับตำแหน่งของอินพุทบิท) ส่วน Key ขนาด 56 บิท จะถูกใช้สร้าง Key ซึ่งใช้สำหรับแต่ละรอบขนาด 48 บิทจำนวน 16 ชุดโดยใช้ความแตกต่างของ Key ชุดย่อย 48 บิทจาก 56 บิทสำหรับแต่ละ Key แต่ละรอบจะใช้อินพุทจากเอาท์พุท 64 บิทของรอบก่อนและใช้ Key 48 บิทของแต่ละรอบ ซึ่งจะได้เอาท์พุท 64 บิท หลังจากรอบที่ 16 เอาท์พุท 64 บิท จะถูกส่งไปทำการ Final Permutation อีกครั้ง ซึ่งจะตรงข้ามกับ Initial Permutation



รูปที่ 3.1 โครงสร้างพื้นฐานของ DES

ที่กล่าวมานั้นคือการทำงานของการเข้ารหัส การถอดรหัสทำงานโดยการ Run DES ย้อนกลับโดยทำการ Run ผ่าน Initial Permutation เพื่อ Undo Final Permutation (Initial และ Final Permutation เป็น Inverse ของกันและกัน) หลังจากนั้นสร้าง Key ที่เหมือนกัน แต่ใช้ Key ใน

ลำดับตรงกันข้าม (เริ่มต้นใช้ K_{16} ซึ่งเป็น Key ที่สร้างชุดสุดท้าย) หลังจากครบ 16 รอบของการถอดรหัสเอาท์พุทจะถูกส่งผ่าน Final Permutation (เพื่อ Undo Initial Permutation)

เพื่อแสดงรายละเอียดของ DES ทั้งหมด จะต้องแสดงรายละเอียดของ Initial และ Final Permutation , แสดงว่า Key แต่ละรอบสร้างอย่างไร และเกิดอะไรขึ้นระหว่างแต่ละรอบ

3.3 การสลับตำแหน่งข้อมูล

DES ทำการ Initial และ Final Permutation ซึ่งจะไม่มีผลทำให้ความปลอดภัยของ DES ดีขึ้น แต่ช่วยทำให้การ Break DES ทำได้ยากขึ้น สิ่งที่เป็นปัญหาของการสลับตำแหน่ง คือทำให้ DES มีประสิทธิภาพลดลงในการนำไปใช้แบบซอฟท์แวร์

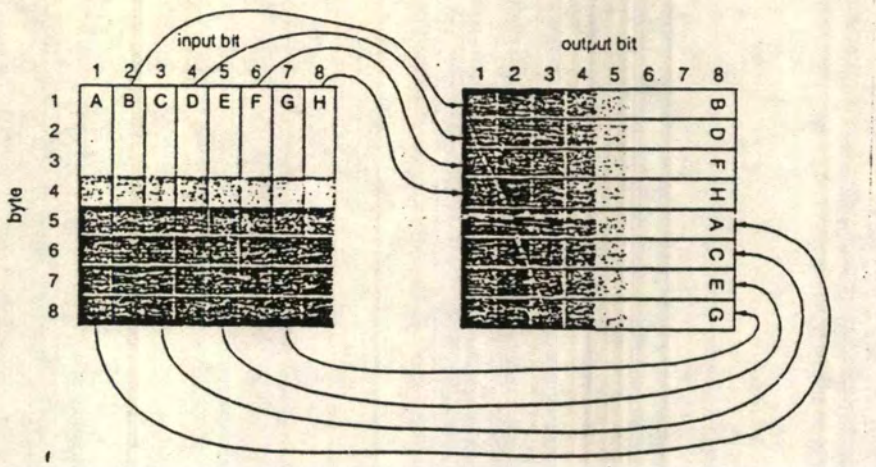
วิธีการสลับตำแหน่งใน DES ถูกกำหนดรายละเอียดดังนี้

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

ตัวเลขในตารางข้างบนเป็นการกำหนดบิทของอินพุทที่ผ่านการสลับตำแหน่ง ลำดับของตัวเลขในตารางจะไปเป็นตำแหน่งของเอาท์พุท ตัวอย่างเช่น Initial Permutation จะนำอินพุทบิทที่ 58 ไปเป็นเอาท์พุทบิทที่ 1 และอินพุทบิทที่ 50 ไปเป็นเอาท์พุทบิทที่ 2

การเปลี่ยนตำแหน่งในขั้นนี้ไม่ใช่วิธีเปลี่ยนตำแหน่งแบบสุ่ม จากรูปที่ 3.2 ลูกศรแสดง Initial Permutation และถ้าย้อนลูกศรกลับจะเป็น Final Permutation

บิทต่างๆ ในไบต์แรกของอินพุท จะไปเป็นบิทที่ 8 ของแต่ละไบต์ และบิทต่างๆ ในไบต์ที่ 2 จะไปเป็นบิทที่ 7ของทุกไบต์ และโดยทั่วไปบิทของไบต์ที่ n จะไปเป็นบิทที่ (9-n) ของทุกไบต์



รูปที่ 3.2 Initial Permutation ของบล็อคข้อมูล

3.4 การสร้าง Key สำหรับแต่ละรอบ

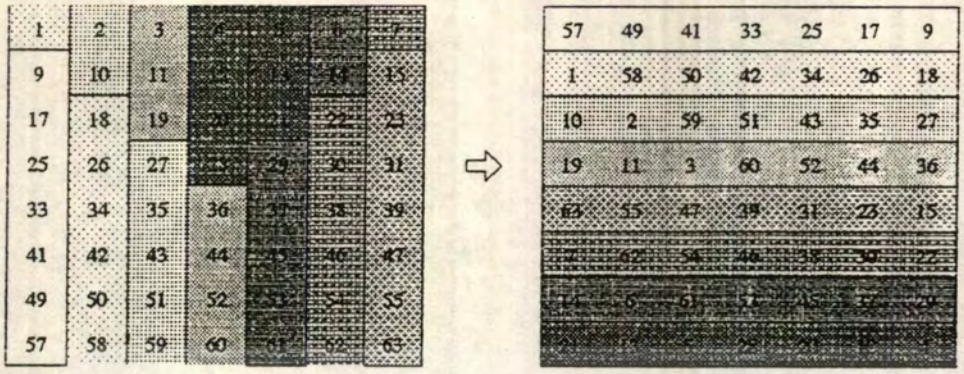
DES Key มีลักษณะยาว 64 บิท แต่จะมี 8 บิทที่เป็นบิท Parity โดยได้แก่บิทที่ 8,16,....,64 DES จะมีฟังก์ชันซึ่งนำ Key 64 บิทนี้ไปสร้าง Key ขนาด 48 บิท จำนวน 16 ชุด ได้แก่ K_1 ,K_2 ,.... ,K_{16}

เริ่มต้นด้วยการทำ Initial Permutation กับ Key ที่ตัดเหลือ 56 บิท เพื่อสร้างเอาท์พุทขนาด 56 บิท โดยแบ่งข้อมูลเป็น 28 บิท 2 ชุด เรียกว่า C_0 และ D_0 การสลับตำแหน่งจะกำหนดดังนี้

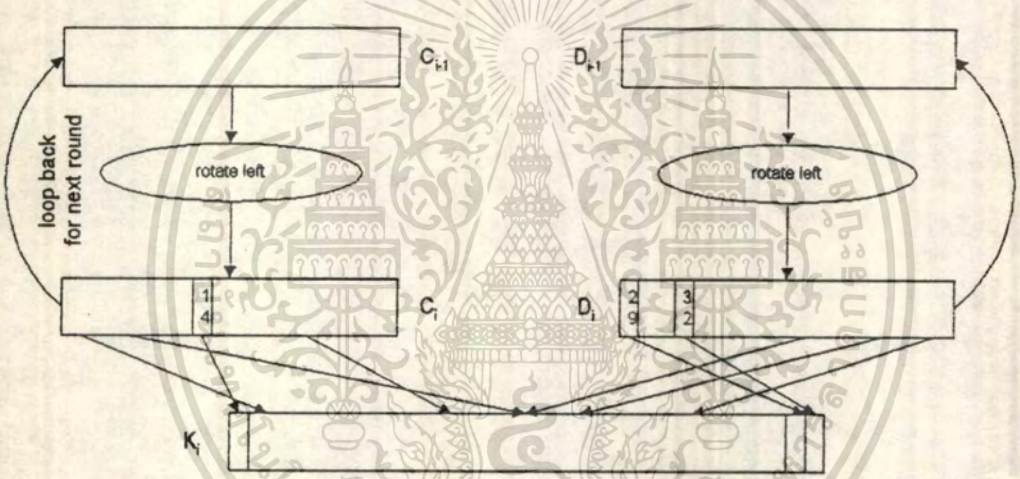
C_0							D_0						
57	49	41	33	25	17	9	63	55	47	39	31	23	15
1	58	50	42	34	26	18	7	62	54	46	38	30	22
10	2	59	51	43	35	27	14	6	61	53	45	37	29
19	11	3	60	52	44	36	21	13	5	28	20	12	4

วิธีอ่านตารางข้างบนคือบิทซ้ายสุดของเอาท์พุทจะเป็นบิทที่ 57 จาก Key และบิทต่อมาจะเป็นบิทที่ 49 และอื่นๆ จนกระทั่งบิทสุดท้ายของ D_0 จะเป็นบิทที่ 4 จาก Key

จากรูปที่ 3.3 แสดงให้เห็นว่าการสลับตำแหน่งดังกล่าวไม่ได้เป็นการสุ่มตำแหน่ง



รูปที่ 3.3 Initial Permutation ของ Key



รูปที่ 3.4 รอบที่ i สำหรับการสร้าง K_i

จากนั้นทำการสร้าง K_i เพื่อนำไปใช้กับทั้ง 16 รอบ (รูปที่ 3.4) จำนวนของบิทที่จะเลื่อนแตกต่างกันในแต่ละรอบ ในรอบที่ 1, 2, 9 และ 16 จะ rotate ไปทางซ้าย 1 บิท ส่วนในรอบอื่นๆ จะ rotate ไปทางซ้าย 2 บิท การสลับตำแหน่งในกรณีนี้จะมีผลต่อความปลอดภัย

การสลับตำแหน่งของ C_i จะสร้างครึ่งซ้ายของ K_i โดยที่บิทที่ 9, 18, 22 และ 28 จะถูกตัดทิ้ง ดังนี้

การสลับตำแหน่งครึ่งซ้ายของ K_i :

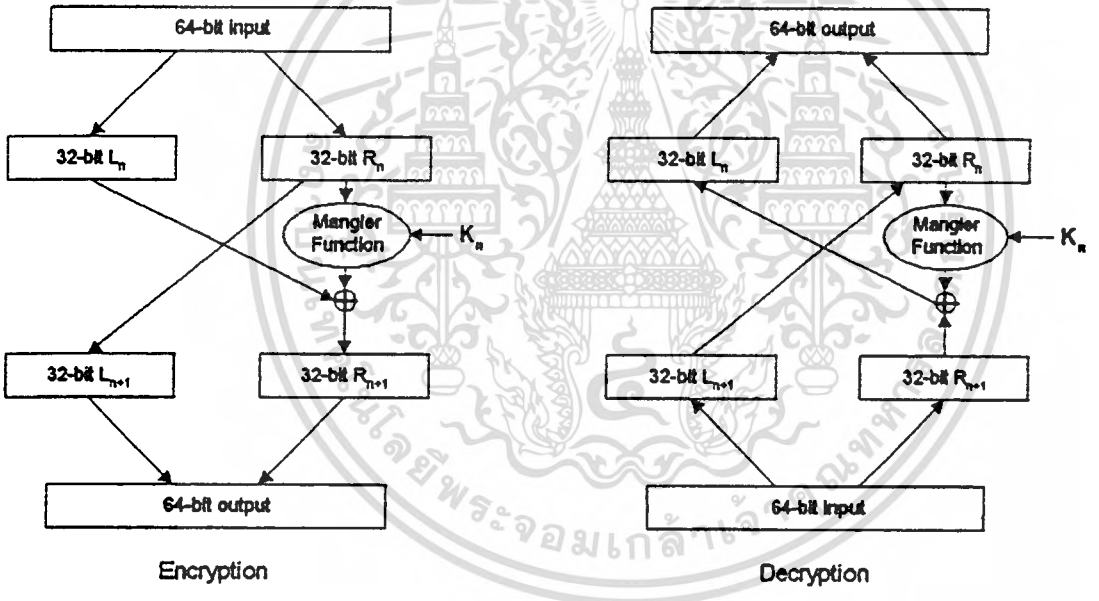
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2

การสลับตำแหน่งครึ่งขวาของ K_i : 30 40 51 45 33 48
44 49 39 56 34 53
46 42 50 36 29 32

แต่ละครึ่งของ K_i เท่ากับ 24 บิท ดังนั้น K_i จะยาว 48 บิท

5 รอบของ DES

รูปที่ 3.5 แสดงการเข้ารหัสและถอดรหัส



รูปที่ 3.5 การเข้ารหัสและถอดรหัสของ DES

ในการเข้ารหัส อินพุทขนาด 64 บิท จะถูกแบ่งเป็น 2 ครึ่งๆ ละ 32 บิท เรียกว่า L_n และ R_n แต่ละบจะสร้างเอาท์พุท 32 บิท คือ L_{n+1} และ R_{n+1} โดยนำ L_{n+1} และ R_{n+1} มารวมกันจะได้เอาท์พุทขนาด 64 ของแต่ละรอบ

L_{n+1} ได้จาก R_n ส่วน R_{n+1} ได้มาดังนี้ เริ่มจาก R_n และ K_n เป็นอินพุทเข้าสู่ Mangler Function ซึ่งได้เอาท์พุทขนาด 32 บิท นำเอาท์พุทที่ได้ไป Exclusive-OR กับ L_n จะได้ R_{n+1} Mangler Function จะอินพุทขนาด 32 บิทกับ Key ขนาด 48 บิท มาสร้างเอาท์พุทขนาด 32 บิท

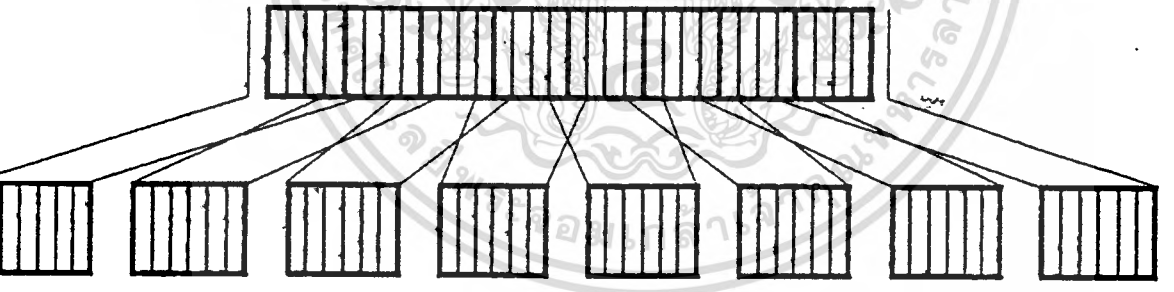
จากที่กล่าวมา ถ้าRun DES ย้อนกลับเพื่อถอดรหัส สมมุติว่ามี L_{n+1} และ R_{n+1} ทำอย่างไรจึงจะได้ $_n$ และ R_n

เราทราบว่า R_n ก็คือ L_{n+1} เพราะฉะนั้นเราจะได้ R_n ,L_{n+1} ,R_{n+1} และ K_n และจากที่ทราบว่า R_{n+1} ละ L_n Exclusive-OR กับ Mangler(R_n,K_n) จากนั้น Exclusive-OR ผลที่ได้กับ R_{n+1} จะได้ผลลัพธ์เป็น L_n

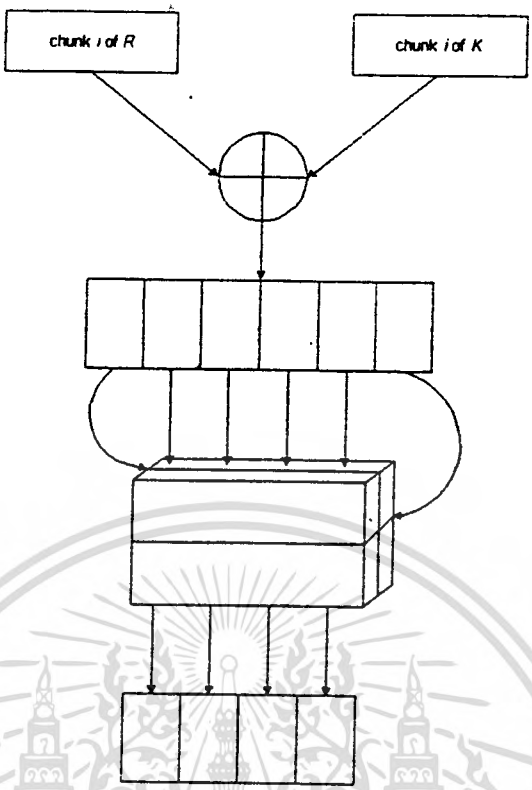
6 Mangler Function

Mangler Function นำอินพุท R_n ขนาด 32 บิท ซึ่งจะเรียกว่า R กับ K_n ขนาด 48 บิท ซึ่งจะเรียกว่า และสร้างเอาท์พุทขนาด 32 บิท ซึ่งจะนำไป Exclusive-OR กับ L_n เพื่อสร้าง R_{n+1} (Rตัวต่อไป)

เริ่มต้นด้วยการขยาย R จาก 32 บิทเป็น 48 บิท โดยแยก R เป็นชุดๆละ 4 บิท จำนวน 8 ชุด ขยายต่ละชุดเป็น 6 บิท โดยนำบิทที่ติดกันและนำบิทเหล่านั้นมาต่อกันแต่ละชุด โดยถือว่าบิทซ้ายสุดและขวาดของ R เป็นบิทที่มีตำแหน่งติดกัน ตัวอย่างเช่น R ชุดที่ 1 คือ บิทที่ 1-4 ขยายเป็น 6 บิท จะนำบิทที่ 32 ละบิทที่ 5 มาต่อเป็นบิทหน้าและบิทหลังตามลำดับ



รูปที่ 3.6 การขยาย *R* ให้เป็น 48 บิท



รูปที่ 3.7 Chunk Transformation

จากนั้นแยก K 48 บิทเป็นชุด ๆ ละ 6 บิท R ที่ขยายชุดที่ i จะนำมา Exclusive-OR กับ K ชุดที่ i จะได้เอาท์พุทขนาด 6 บิท นำเอาท์พุทที่ได้มาผ่าน S Box ซึ่งจะสร้างเอาท์พุท 4 บิท จาก อินพุท 6 บิท โดย

S Box ได้ถูกกำหนดดังต่อไปนี้

Input bits 1 and 6	Input bits 2 thru 5															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

ตารางที่ 3.1 ตารางของเอาท์พุท 4 บิทของ S Box 1 (บิทที่ 1 ถึง 4)

Input bits 7 and 12	Input bits 8 thru 11															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1111	0001	1000	1110	0110	1011	0011	0100	1001	0111	0010	1101	1100	0000	0101	1010
01	0011	1101	0100	0111	1111	0010	1000	1110	1100	0000	0001	1010	0110	1001	1011	0101
10	0000	1110	0111	1011	1010	0100	1101	0001	0101	1000	1100	0110	1001	0011	0010	1111
11	1101	1000	1010	0001	0011	1111	0100	0010	1011	0110	0111	1100	0000	0101	1110	1001

ตารางที่ 3.2 ตารางของเอาท์พุท 4 บิทของ S Box 2 (บิทที่ 5 ถึง 8)

Input bits 13 and 18	Input bits 14 thru 17															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1010	0000	1001	1110	0110	0011	1111	0101	0001	1101	1100	0111	1011	0100	0010	1000
01	1101	0111	0000	1001	0011	0100	0110	1010	0010	1000	0101	1110	1100	1011	1111	0001
10	1101	0110	0100	1001	1000	1111	0011	0000	1011	0001	0010	1100	0101	1010	1110	0111
11	0001	1010	1101	0000	0110	1001	1000	0111	0100	1111	1110	0011	1011	0101	0010	1100

ตารางที่ 3.3 ตารางของเอาท์พุท 4 บิทของ S Box 3 (บิทที่ 9 ถึง 12)

Input bits 19and 24	Input bits 20 thru 23															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0111	1101	1110	0011	0000	0110	1001	1010	0001	0010	1000	0101	1011	1100	0100	1111
01	1101	1000	1011	0101	0110	1111	0000	0011	0100	0111	0010	1100	0001	1010	1110	1001
10	1010	0110	1001	0000	1100	1011	0111	1101	1111	0001	0011	1110	0101	0010	1000	0100
11	0011	1111	0000	0110	1010	0001	1101	1000	1001	0100	0101	1011	1100	0111	0010	1110

ตารางที่ 3.4 ตารางของเอาท์พุท 4 บิทของ S Box 4 (บิทที่ 13 ถึง 16)

Input bits 25 and 30	Input bits 26 thru 29															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

ตารางที่ 3.5 ตารางของเอาท์พุท 4 บิทของ S Box 5 (บิทที่ 17 ถึง 20)

Input bits 31 and 36	Input bits 32 thru 35															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1100	0001	1010	1111	1001	0010	0110	1000	0000	1101	0011	0100	1110	0111	0101	1011
01	1010	1111	0100	0010	0111	1100	1001	0101	0110	0001	1101	1110	0000	1011	0011	1000
10	1001	1110	1111	0101	0010	1000	1100	0011	0111	0000	0100	1010	0001	1101	1011	0110
11	0100	0011	0010	1100	1001	0101	1111	1010	1011	1110	0001	0111	0110	0000	1000	1101

ตารางที่ 3.6 ตารางของเอาท์พุท 4 บิทของ S Box 6 (บิทที่ 21 ถึง 24)

Input bits 37 and 42	Input bits 38 thru 41															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0100	1011	0010	1110	1111	0000	1000	1101	0011	1100	1001	0111	0101	1010	0110	0001
01	1101	0000	1011	0111	0100	1001	0001	1010	1110	0011	0101	1100	0010	1111	1000	0110
10	0001	0100	1011	1101	1100	0011	0111	1110	1010	1111	0110	1000	0000	0101	1001	0010
11	0110	1011	1101	1000	0001	0100	1010	0111	1001	0101	0000	1111	1110	0010	0011	1100

ตารางที่ 3.7 ตารางของเอาท์พุท 4 บิทของ S Box 7 (บิทที่ 25 ถึง 28)

Input bits 43 and 48	Input bits 44 thru 47															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1101	0010	1000	0100	0110	1111	1011	0001	1010	1001	0011	1110	0101	0000	1100	0111
01	0001	1111	1101	1000	1010	0011	0111	0100	1100	0101	0110	1011	0000	1110	1001	0010
10	0111	1011	0100	0001	1001	1100	1110	0010	0000	0110	1010	1101	1111	0011	0101	1000
11	0010	0001	1110	0111	0100	1010	1000	1101	1111	1100	1001	0000	0011	0101	0110	1011

ตารางที่ 3.8 ตารางของเอาท์พุท 4 บิทของ S Box 8 (บิทที่ 29 ถึง 32)

เอาท์พุท 4 บิท ของ S Boxทั้ง8ชุดถูกรวมเป็น 32 บิท จากนั้นนำบิทเหล่านี้ไปสลับตำแหน่ง การสลับตำแหน่งในขั้นนี้มีผลต่อความปลอดภัยของ DES เพื่อให้มั่นใจว่าเอาท์พุทของ S Box ในแต่ละรอบมีผลต่ออินพุทของ S Box ในรอบต่อไป ถ้าไม่มีการสลับตำแหน่งอินพุทบิทซ้ายจะมีผลต่อเอาท์พุทบิทซ้ายเท่านั้น

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

รูปที่ 3.8 Permutation ของ S Box 32 บิท

วิธีการอ่านคือบิทที่ 1 ของเอาท์พุทมาจากบิทที่ 16 ของอินพุท ส่วนเอาท์พุทบิทที่ 2 มาจากอินพุทบิทที่ 7 , ...

.7 Weak และ Semi Weak Key

มี DES Key จำนวน 16 Key ซึ่งมีคุณสมบัติพิเศษ ซึ่งไม่เหมาะที่จะนำมาใช้เพื่อความปลอดภัย ต่ความน่าจะเป็นที่จะสร้าง Key เหล่านี้ขึ้นมามีแค่ $16/2^{56}$ โดยที่ Key 16 ชุด ดังกล่าวนี้ (จาก *3.4 การร้าง Key สำหรับแต่ละรอบ*) ซึ่งแบ่งเป็น C_0 และ D_0 เกิดจาก C_0 และ D_0 มีค่าดังนี้คือ เป็นศูนย์หมด, เป็นนึ่งหมด , สลับหนึ่งกับศูนย์ และ สลับศูนย์กับหนึ่ง โดยที่ C_0 และ D_0 มีค่าเป็นศูนย์หรือหนึ่งหมด จะยกว่า *Weak Key* ซึ่งมี 4 ชุด ส่วน Key อีก 12 ชุดที่เหลือ เรียกว่า *Semi-weak Key*

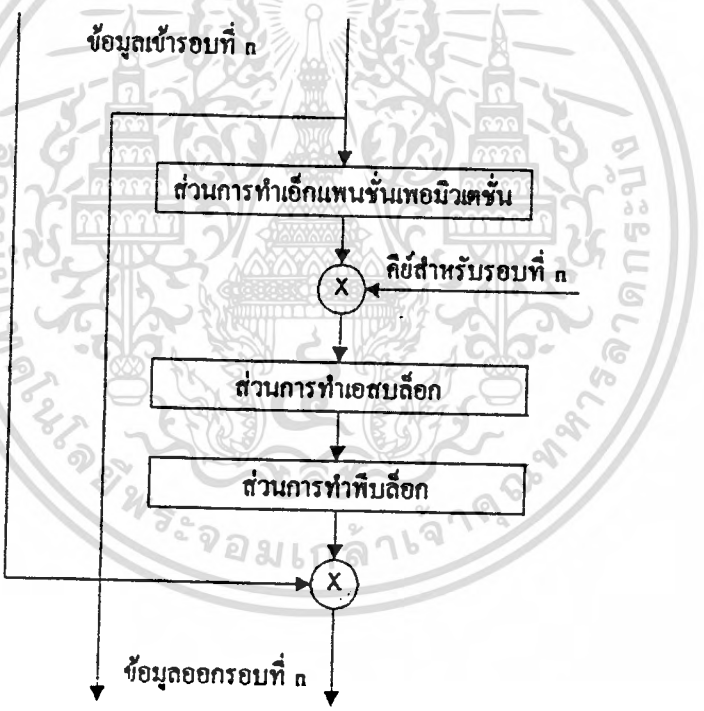
นอกจากนี้ยังมีคำแนะนำว่าไม่ควรใช้ Key ที่มีค่าต่ำกว่าหนึ่งพัน เพราะว่าผู้ที่พยายามเจาะระบบจเริ่มการหา Key จากค่าต่ำสุดได้

บทที่ 4

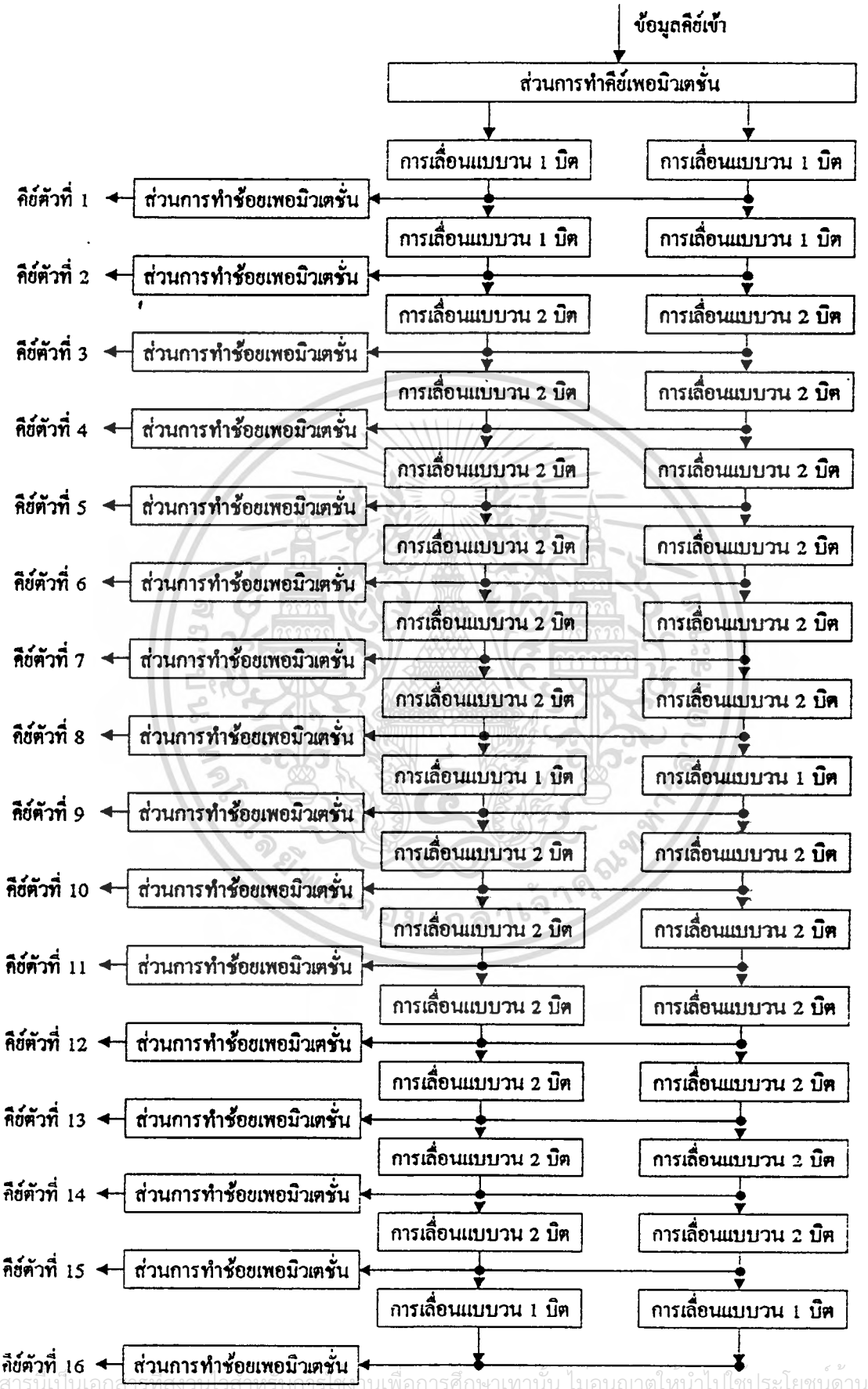
ขั้นตอนการออกแบบเป็นภาษาวีเฮชดีแอล

ขั้นตอนที่ 1 สร้างรูปแผนภาพ(BLOCK DIAGRAM)แสดงการทำงานตามอัลกอริทึมดีอีเอส (DES)

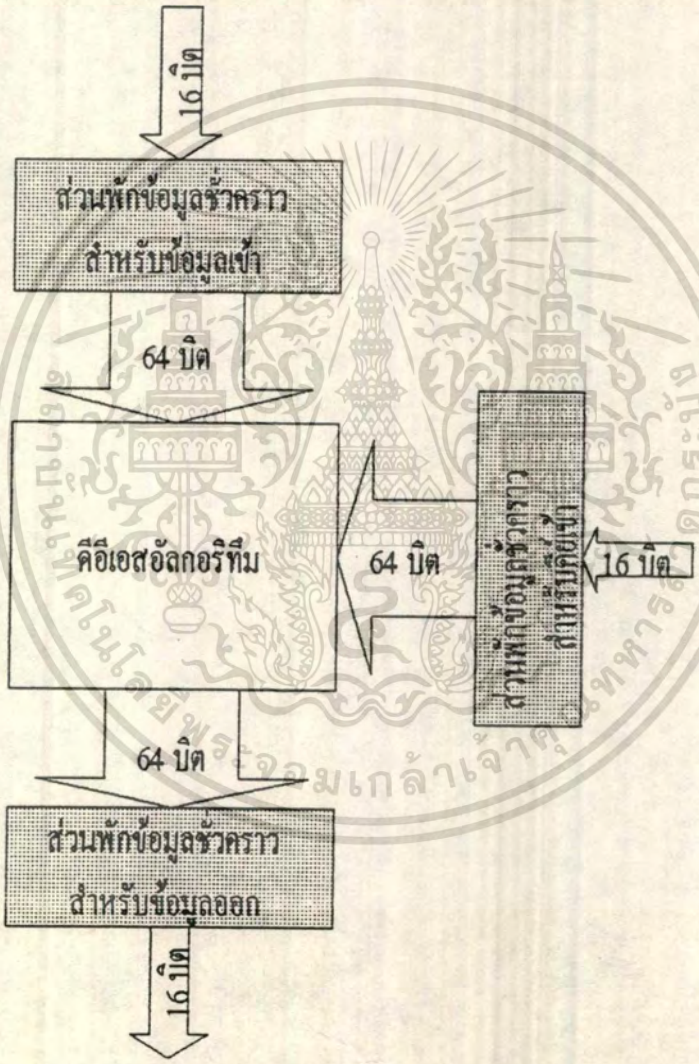
1.1 จากอัลกอริทึมดีอีเอส จะต้องทำการเข้ารหัสข้อมูลด้วยการกระทำที่เหมือนเดิมเป็นจำนวน 16 ครั้ง ดังนั้นจึงออกแบบให้มีส่วนการเข้ารหัสข้อมูลที่มีการกระทำเหมือนเดิมเพียงส่วนเดียว แล้วควบคุมการไหลของข้อมูลให้กระทำซ้ำๆกันจนครบ 16 ครั้ง ซึ่งรูปแผนภาพแสดงได้ดังนี้



1.2 ส่วนคีย์ ในแต่ละรอบของการเข้ารหัสก็จะใช้คีย์ที่แตกต่างกัน ซึ่งส่วนจัดการคีย์ จะทำหน้าที่เตรียมคีย์ไว้ 16 ตัวสำหรับแต่ละรอบของการเข้ารหัส โดยใช้ข้อมูลเริ่มต้นจากคีย์ที่ป้อนเข้ามา



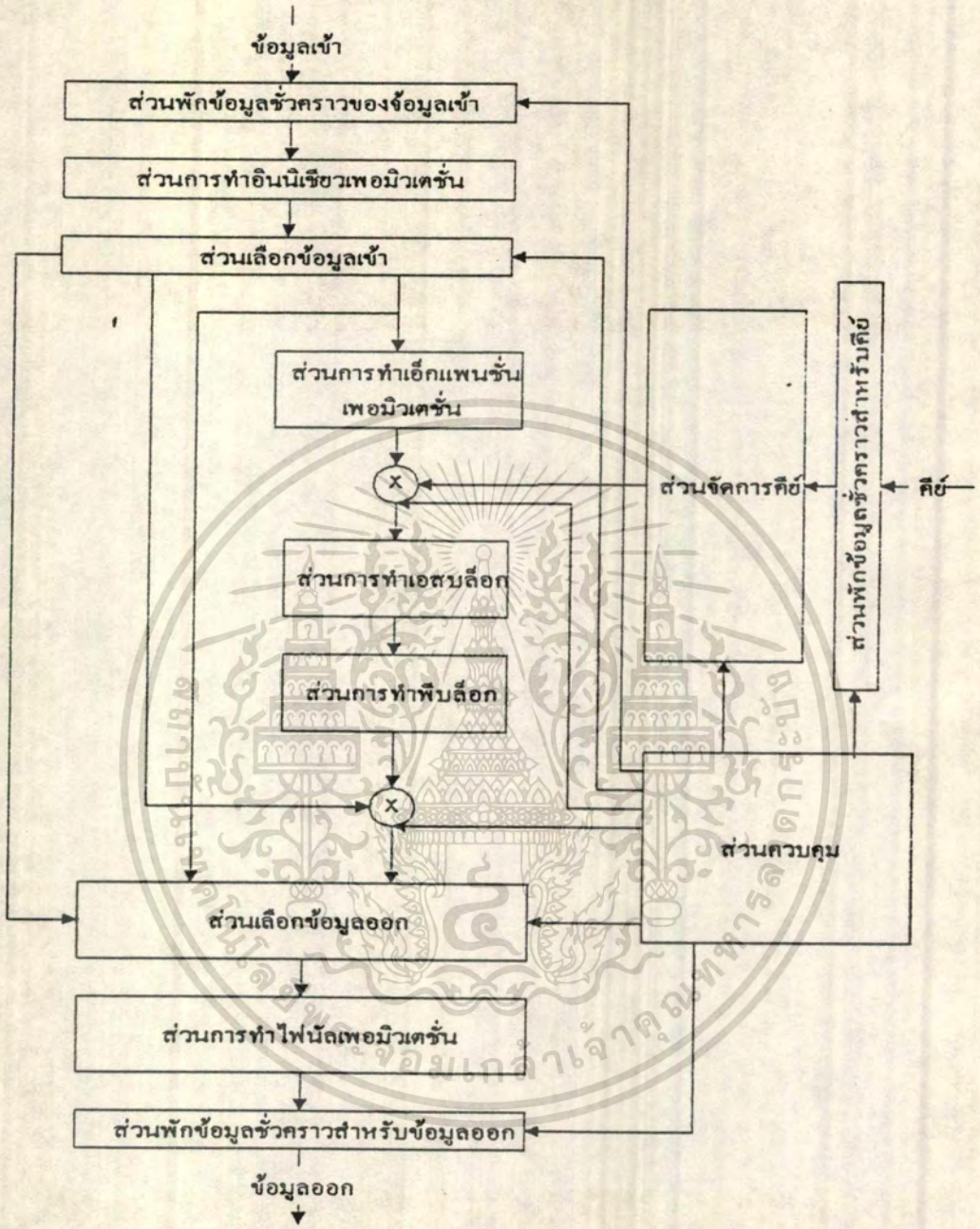
1.3 จากอัลกอริทึมมีข้อมูลเข้าจำนวน 64 บิต มีคีย์เข้าจำนวน 64 บิต และข้อมูลออกจำนวน 64 บิต จะเห็นว่าต้องใช้ขาสัญญาณจำนวนมาก ดังนั้นจึงทำการลดขาสัญญาณโดยออกแบบให้มีข้อมูลเข้าทีละ 16 บิต จำนวน 4 ครั้ง มีคีย์เข้าทีละ 16 บิต จำนวน 4 ครั้ง และข้อมูลออกทีละ 16 บิต จำนวน 4 ครั้ง มีแผนภาพดังนี้



แผนภาพแสดงการทำงานเมื่อมีการเพิ่มส่วนพักข้อมูลชั่วคราว

1.4 ส่วนควบคุมซึ่งจะทำหน้าที่ควบคุมส่วนต่างๆเพื่อให้การไหลของข้อมูลถูกต้องตามที่ได้ออกแบบไว้ โดยอาศัยสัญญาณนาฬิกาจากภายนอกเป็นตัวควบคุมจังหวะ

แผนภาพแสดงส่วนประกอบทั้งหมด



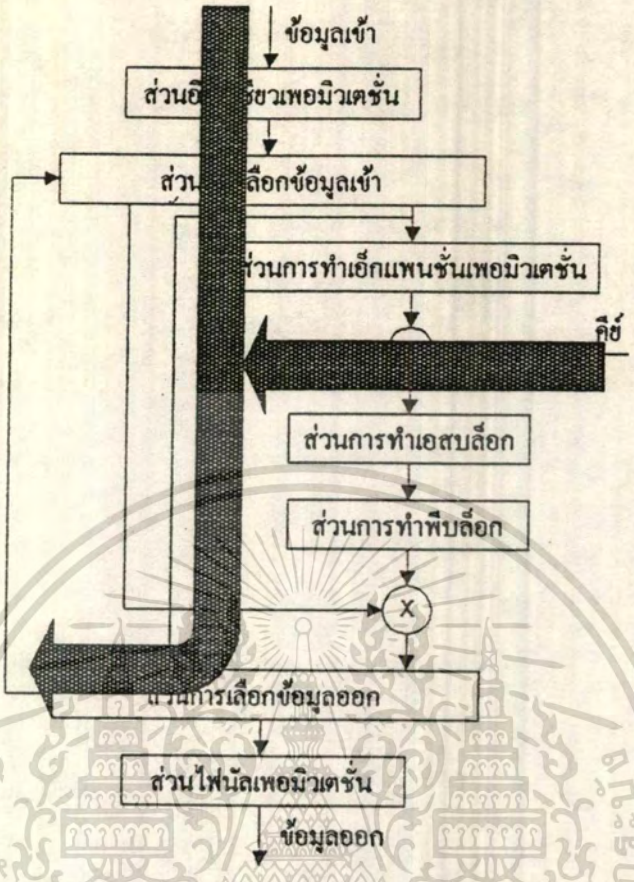
ขั้นตอนที่ 2 การไหลของข้อมูลและคีย์ในแต่ละรอบของการเข้ารหัสและถอดรหัส

ซึ่งสามารถแสดงลักษณะการไหลของข้อมูลได้ดังนี้

2.1. เมื่อทำการเข้ารหัสข้อมูล

การไหลของข้อมูลรอบที่ 1

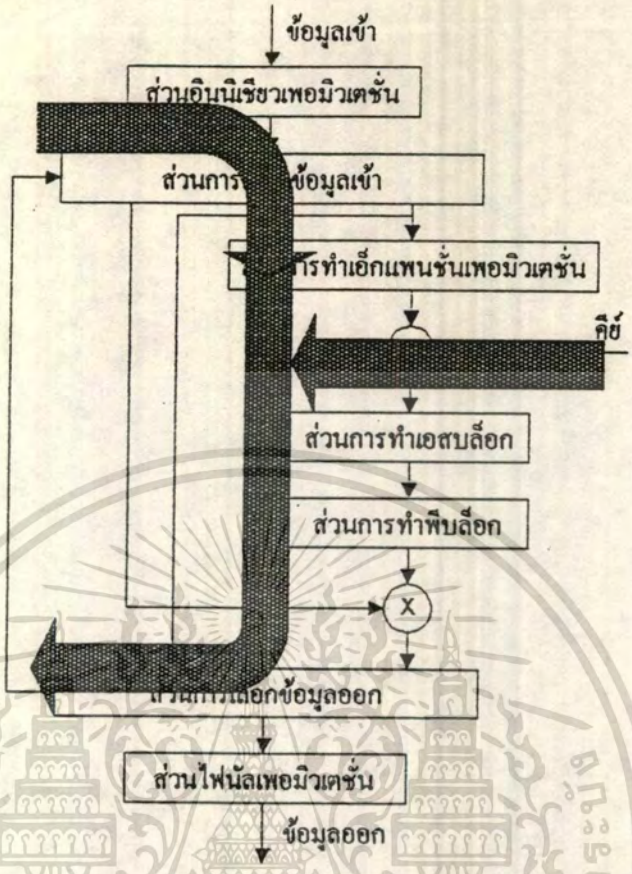
ข้อมูลจะใช้ข้อมูลเข้าจากภายนอก ส่วนคีย์จะใช้คีย์ตัวที่ 1 ผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 2



จากรูปแสดงการไหลของข้อมูลในรอบที่ 1

การไหลของข้อมูลรอบที่ 2

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 1 คีย์จะใช้คีย์ตัวที่ 2 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 3



การไหลของข้อมูลรอบที่ 3

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 2 คีย์จะใช้คีย์ตัวที่ 3 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 4

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 3 คีย์จะใช้คีย์ตัวที่ 4 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 5

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 4 คีย์จะใช้คีย์ตัวที่ 5 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 6

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 5 คีย์จะใช้คีย์ตัวที่ 6 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 7

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 6 คีย์จะใช้คีย์ตัวที่ 7 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 8

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 7 คีย์จะใช้คีย์ตัวที่ 8 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 9

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 8 คีย์จะใช้คีย์ตัวที่ 9 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 10

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 9 คีย์จะใช้คีย์ตัวที่ 10 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 11

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 10 คีย์จะใช้คีย์ตัวที่ 11 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 12

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 11 คีย์จะใช้คีย์ตัวที่ 12 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 13

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 12 คีย์จะใช้คีย์ตัวที่ 13 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 14

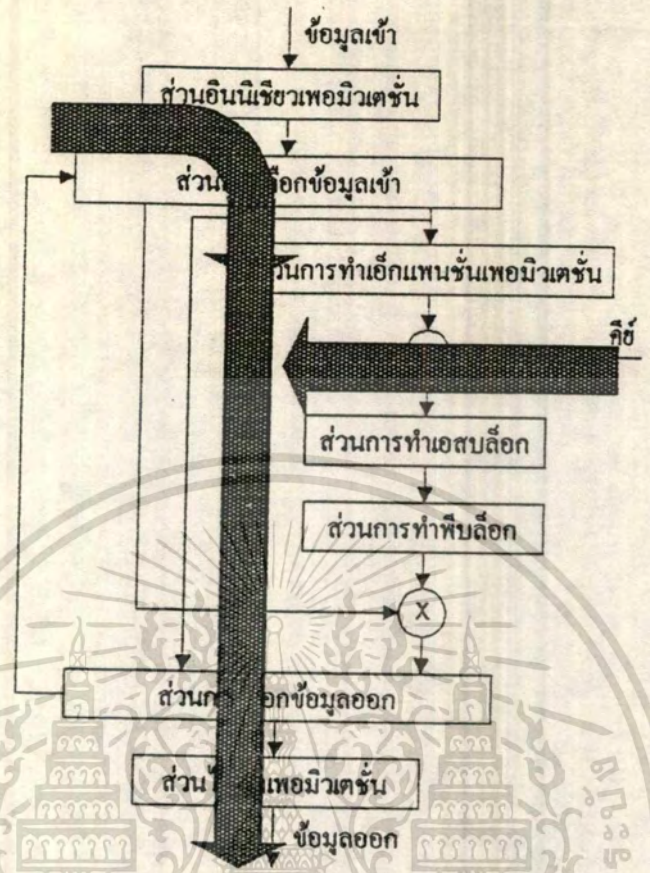
ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 13 คีย์จะใช้คีย์ตัวที่ 14 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 15

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 14 คีย์จะใช้คีย์ตัวที่ 15 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 16

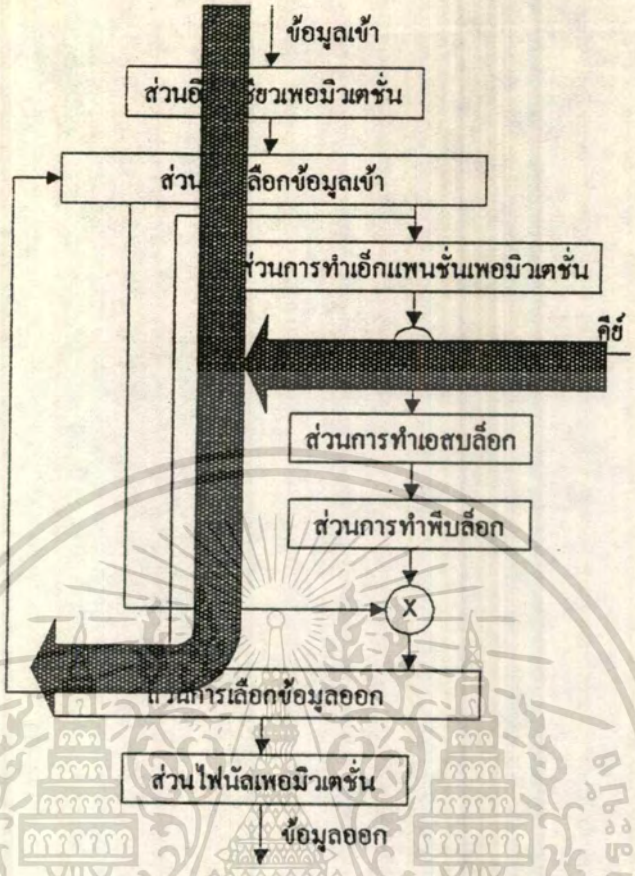
ข้อมูลเข้ามาจากผลลัพธ์ของรอบที่ 15 คีย์จะใช้คีย์ตัวที่ 16 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลออก



2.2 เมื่อทำการถอดรหัสข้อมูล

การไหลของข้อมูลรอบที่ 1

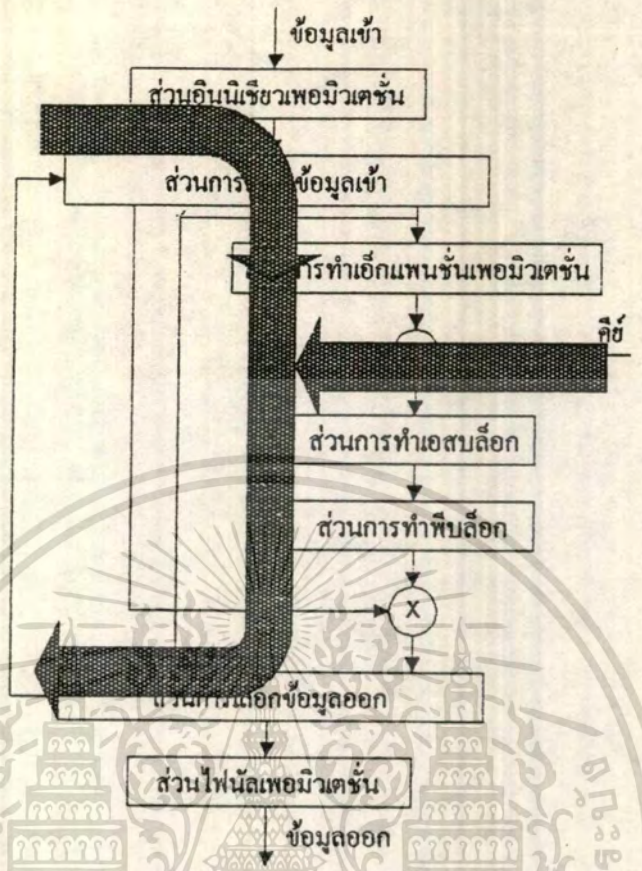
ข้อมูลจะใช้ข้อมูลเข้าจากภายนอก ส่วนคีย์จะใช้คีย์ตัวที่ 16 ผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 2



จากรูปแสดงการไหลของข้อมูลในรอบที่ 1

การไหลของข้อมูลรอบที่ 2

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่1 คีย์จะใช้คีย์ตัวที่ 15 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 3



การไหลของข้อมูลรอบที่ 3

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 2 คีย์จะใช้คีย์ตัวที่ 14 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 4

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 3 คีย์จะใช้คีย์ตัวที่ 13 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 5

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 4 คีย์จะใช้คีย์ตัวที่ 12 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 6

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 5 คีย์จะใช้คีย์ตัวที่ 11 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 7

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 6 คีย์จะใช้คีย์ตัวที่ 10 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 8

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 7 คีย์จะใช้คีย์ตัวที่ 9 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 9

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 8 คีย์จะใช้คีย์ตัวที่ 8 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 10

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 9 คีย์จะใช้คีย์ตัวที่ 7 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 11

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 10 คีย์จะใช้คีย์ตัวที่ 6 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 12

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 11 คีย์จะใช้คีย์ตัวที่ 5 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 13

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 12 คีย์จะใช้คีย์ตัวที่ 4 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 14

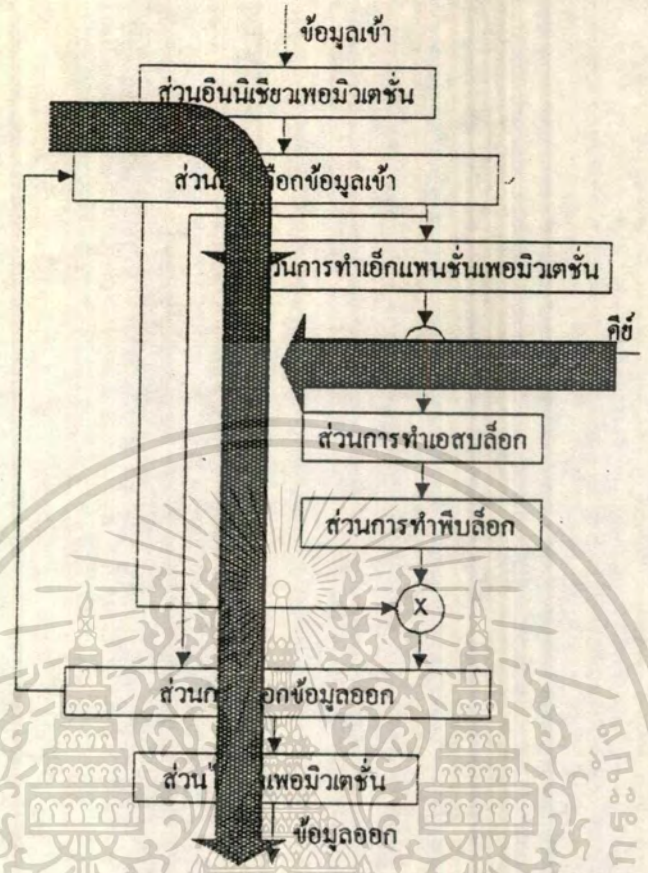
ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 13 คีย์จะใช้คีย์ตัวที่ 3 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 15

ข้อมูลเข้าจะมาจากผลลัพธ์ของรอบที่ 14 คีย์จะใช้คีย์ตัวที่ 2 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลเข้าของรอบที่ 4 ลักษณะการไหลของข้อมูลจะเหมือนกับรอบที่ 2

การไหลของข้อมูลรอบที่ 16

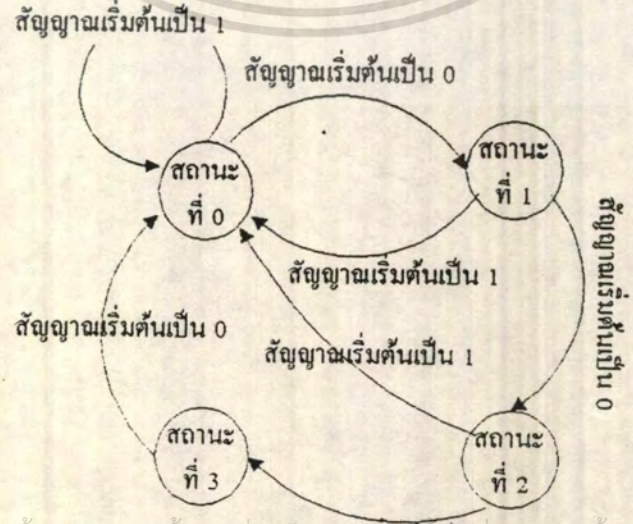
ข้อมูลเข้ามาจากผลลัพธ์ของรอบที่ 15 คีย์จะใช้คีย์ตัวที่ 1 ส่วนผลลัพธ์จะถูกส่งไปเป็นข้อมูลออก



3.แปลงแต่ละส่วน(MODULE) ให้สามารถแทนด้วยภาษาวีเฮชดีแอลที่มีการกระทำเหมือนกัน

3.1 ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า (โมดูล Input Buffer ในภาคผนวก)

ส่วนนี้จะทำหน้าที่รับข้อมูลเข้าที่ละ 16 บิต เป็นจังหวะตามสัญญาณควบคุม โดยสามารถแทนส่วนนี้ด้วยไฟไนสเตตแมชีน ดังรูป



ที่มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณควบคุมเป็นขอบขาลง จะทำการตั้งสถานะของตนเองให้เป็นสถานะเริ่มต้นหรือสถานะที่ 0

- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าจำนวน 16 บิต แล้วส่งเป็นข้อมูลออกบิตที่ 0 ถึง 15 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าจำนวน 16 บิต แล้วส่งเป็นข้อมูลออกบิตที่ 16 ถึง 31 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 2

- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าจำนวน 16 บิต แล้วส่งเป็นข้อมูลออกบิตที่ 31 ถึง 47 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 3

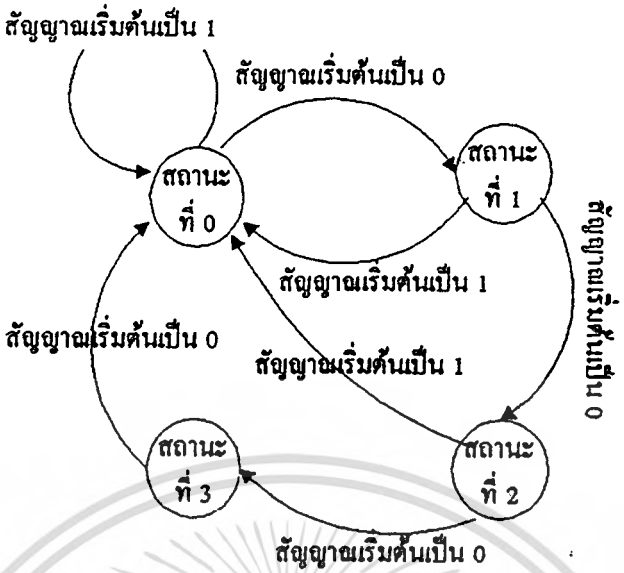
- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าจำนวน 16 บิต แล้วส่งเป็นข้อมูลออกบิตที่ 48 ถึง 63 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 0

3.2 *ส่วนพักข้อมูลชั่วคราวสำหรับคีย์* (โมดูล Key Buffer ในภาคผนวก)

ส่วนนี้จะทำหน้าที่รับคีย์เข้าทีละ 16 บิตเป็นจังหวะตามสัญญาณควบคุม โดยสามารถแทนส่วนนี้ด้วยไฟไนสเตตแมชีน ซึ่งจะมีการทำงานเหมือนกับส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า

3.3 *ส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออก* (โมดูล Output Buffer ในภาคผนวก)

ส่วนนี้จะทำการนำข้อมูลเข้าขนาด 64 บิต แล้วส่งออกทีละ 16 บิตจำนวน 4 ครั้ง โดยมีจังหวะตามสัญญาณควบคุมจากส่วนควบคุม ซึ่งสามารถแสดงการทำงานโดยไฟไนสเตตแมชีนได้ดังนี้



ที่มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณควบคุมเป็นขอบขาลง จะทำการตั้งสถานะของตนเองให้เป็นสถานะเริ่มต้นหรือสถานะที่ 0

- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าบิตที่ 0 ถึง 15 แล้วส่งเป็นข้อมูลออก จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าบิตที่ 16 ถึง 31 แล้วส่งเป็นข้อมูลออก จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 2

- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าบิตที่ 32 ถึง 47 แล้วส่งเป็นข้อมูลออก จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 3

- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับข้อมูลเข้าบิตที่ 48 ถึง 63 แล้วส่งเป็นข้อมูลออก จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 0

3.4 *ส่วนการทำอินนิเชียวเพอมิวเตชั่น*

ส่วนนี้จะทำการสลับบิตตามอัลกอริทึมดีอีเอส สามารถแสดงส่วนนี้โดยการใช้คุณสมบัติของภาษาวีเอชดีแอลโดยเขียนแบบสตรักเทอรัล ซึ่งจะสลับบิตข้อมูลตามตามอัลกอริทึมดีอีเอส ระหว่างการเชื่อมต่อส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า กับ ส่วนเลือกข้อมูลเข้า

3.5 *ส่วนการทำไฟนัลเพอมิวเตชั่น*

ส่วนนี้จะทำการสลับบิตตามอัลกอริทึมดีอีเอส สามารถแสดงส่วนนี้โดยการใช้คุณสมบัติของภาษาวีเอชดีแอลโดยเขียนแบบสตรักเทอรัล ซึ่งจะสลับบิตข้อมูลตามตามอัลกอริทึมดีอีเอส ระหว่างการเชื่อมต่อส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออก กับ ส่วนเลือกข้อมูลออก

3.6 *ส่วนการหนดีย์พอมิวเตชั่น*

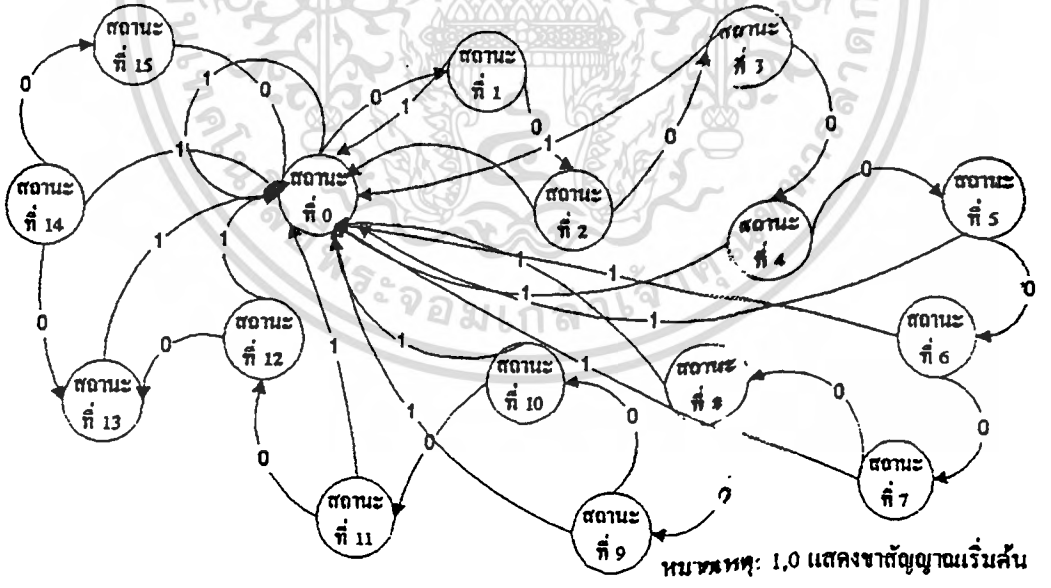
ส่วนนี้จะทำการสลับบิตตามอัลกอริทึมดีอีเอส สามารถแสดงส่วนนี้โดยการใช้คุณสมบัติของภาษาวีเอชดีแอลโดยเขียนแบบสตรักเทอรัล ซึ่งจะสลับบิตข้อมูลตามตามอัลกอริทึมดีอีเอส ระหว่างการเชื่อมต่อส่วนพักข้อมูลชั่วคราวสำหรับคีย์ กับ ส่วนจัดการกับคีย์

3.7 *ส่วนจัดการคีย์* (โมดูล Key ในภาคผนวก)

ในส่วนนี้จะทำหน้าที่เปลี่ยนตำแหน่งบิตของคีย์ตามอัลกอริทึมดีอีเอส โดยคีย์ที่ได้จะแตกต่างกันในแต่ละรอบของการเข้ารหัสและการถอดรหัส ซึ่งในส่วนนี้สามารถแยกออกได้เป็น 3 ส่วนย่อยๆคือ

3.7.1. ส่วนจัดการกับคีย์ที่ใช้ในการเข้ารหัส (โมดูล Key *for* Data Encryption ในภาคผนวก)

ในส่วนนี้จะทำการสลับบิตของคีย์ซึ่งแต่ละรอบจะแตกต่างกัน อาศัยสัญญาณควบคุมจากส่วนควบคุมเป็นตัวกำหนดรอบของการเข้ารหัส สามารถแทนด้วยไฟไนสเตตแมชีนได้ดังนี้



มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณควบคุมเป็นขอบขาลง จะทำการตั้งสถานะของตนเองให้เป็นสถานะเริ่มต้นหรือสถานะที่ 0

- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 1 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 2 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 2

- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 3 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 3

- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 4 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 4

- ถ้าอยู่ในสถานะที่ 4 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 5 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 5

- ถ้าอยู่ในสถานะที่ 5 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 6 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 6

- ถ้าอยู่ในสถานะที่ 6 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 7 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 7

- ถ้าอยู่ในสถานะที่ 7 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 8 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 8

- ถ้าอยู่ในสถานะที่ 8 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 9 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 9

- ถ้าอยู่ในสถานะที่ 9 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 10 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 10

- ถ้าอยู่ในสถานะที่ 10 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 11 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 11

- ถ้าอยู่ในสถานะที่ 11 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 12 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 12

- ถ้าอยู่ในสถานะที่ 12 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 13 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 13

- ถ้าอยู่ในสถานะที่ 13 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 14 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 14

- ถ้าอยู่ในสถานะที่ 14 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 15 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 15

- ถ้าอยู่ในสถานะที่ 15 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะรับคีย์มาแล้วจะสลับบิตตามอัลกอริทึมการเข้ารหัสดีอีเอสในรอบที่ 16 จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 0

3.7.2. ส่วนจัดการกับคีย์ที่ใช้ในการถอดรหัส (โมดูล Key for Data Decryption ในภาคผนวก)

เหมือนกับส่วนจัดการกับคีย์ที่ใช้ในการเข้ารหัสแต่การสลับคีย์ตามอัลกอริทึมการถอดรหัส

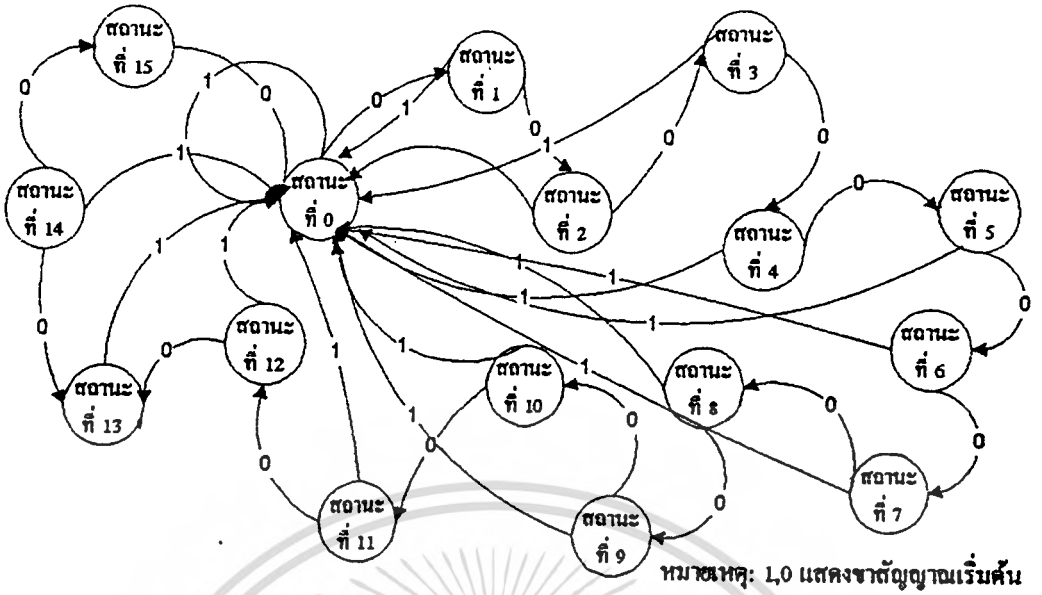
3.7.3. ส่วนเลือกว่าจะใช้คีย์เพื่อที่จะเข้ารหัสหรือถอดรหัส (โมดูล Key Selector ในภาคผนวก)

ส่วนนี้จะรับข้อมูลทั้งการเข้ารหัสและการถอดรหัส แต่จะเลือกเอาเพียง 1 ชุด โดยใช้สัญญาณการเลือกเข้ารหัส-ถอดรหัส เป็นตัวกำหนด สามารถแสดงส่วนนี้ด้วยวงจรซีเควนเชียว ซึ่งแสดงการทำงานได้ดังนี้

- ถ้าสัญญาณแสดงการเข้า-ถอดรหัสเป็น 1 ให้เลือกส่วนการเข้ารหัส นอกจากนั้นจะใช้การถอดรหัส

3.8 ส่วนการเลือกข้อมูลเข้า (โมดูล Input Buffer ในภาคผนวก)

ส่วนนี้จะทำหน้าที่เลือกข้อมูลเพื่อเข้ากระทำอัลกอริทึมดีอีเอส จะเลือกระหว่างข้อมูลจากภายนอกกับข้อมูลที่ได้จากการกระทำรอบก่อนหน้านี้ โดยอาศัยสัญญาณควบคุมจากส่วนควบคุม สามารถแสดงด้วยไฟไนสเตตแมชีน ได้ดังนี้



มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณควบคุมเป็นขอบขาลง จะทำการตั้งสถานะของตนเองให้เป็นสถานะเริ่มต้นหรือสถานะที่ 0

- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้า จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 2

- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 3

- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 4

- ถ้าอยู่ในสถานะที่ 4 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 5

- ถ้าอยู่ในสถานะที่ 5 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 6

- ถ้าอยู่ในสถานะที่ 6 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 7

- ถ้าอยู่ในสถานะที่ 7 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 8

- ถ้าอยู่ในสถานะที่ 8 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 9

- ถ้าอยู่ในสถานะที่ 9 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 10

- ถ้าอยู่ในสถานะที่ 10 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 11

- ถ้าอยู่ในสถานะที่ 11 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 12

- ถ้าอยู่ในสถานะที่ 12 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 13

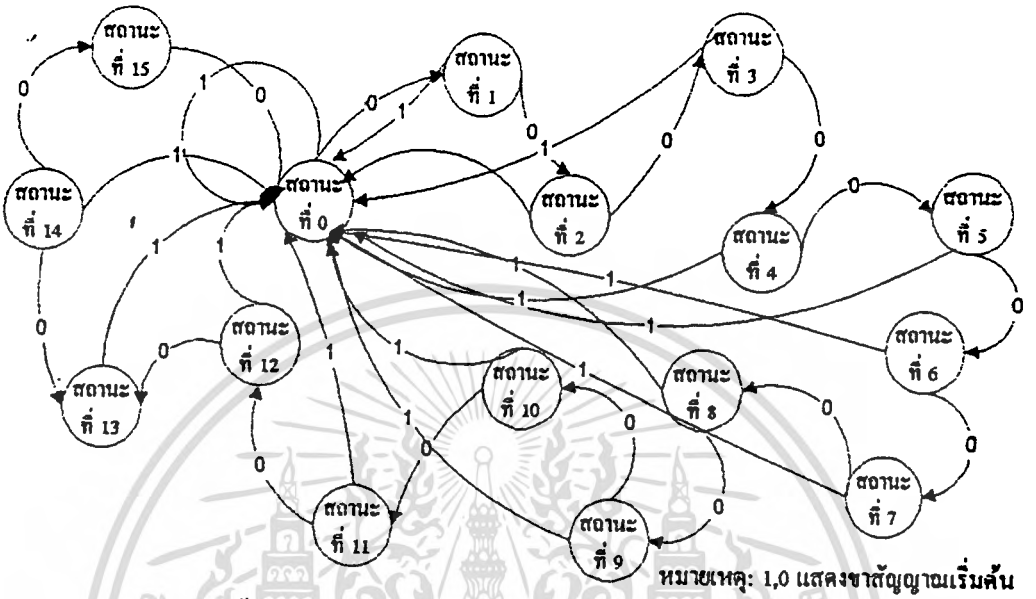
- ถ้าอยู่ในสถานะที่ 13 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 14

- ถ้าอยู่ในสถานะที่ 14 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 15

- ถ้าอยู่ในสถานะที่ 15 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลเข้าจากส่วนการเลือกข้อมูลออก ซึ่งเป็นข้อมูลที่ผ่านการกระทำรอบที่แล้ว จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 0

3.9 ส่วนการเลือกข้อมูลออก (โมดูล Output Buffer ในภาคผนวก)

ส่วนนี้จะทำการเลือกข้อมูลเพื่อส่งไปยังรอบต่อไป หรือส่งข้อมูลออกเมื่อทำครบ 16 รอบแล้ว โดยอาศัยสัญญาณควบคุมจากส่วนควบคุม สามารถแสดงด้วยไฟไนสเตตแมชีน ได้ดังนี้



หมายเหตุ: 1,0 แสดงขาสัญญาณเริ่มต้น

มีการทำงานดังนี้

- ถ้าสัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณควบคุมเป็นขอบขาลง จะทำการตั้งสถานะของตนเองให้เป็นสถานะเริ่มต้นหรือสถานะที่ 0

- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 4 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 5 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 6 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 7 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 8 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 9 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 10 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 11 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 12 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 13 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 14 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนการเลือกข้อมูลเข้าเพื่อกระทำรอบต่อไป จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 1

- ถ้าอยู่ในสถานะที่ 15 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณควบคุมเป็นขอบขาลง จะเลือกข้อมูลแล้วส่งออกไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออก จากนั้นจะเปลี่ยนสถานะให้เป็นสถานะที่ 0

3.10 ส่วนการทำเอ็กแพนชั่นเพอมิวเตชั่น

ส่วนนี้จะทำการขยายจำนวนบิตตามอัลกอริทึมดีอีเอส สามารถแสดงส่วนนี้โดยการใช้คุณสมบัติของภาษาวีเอชดีแอลโดยเขียนแบบสตรักเทอรัล ซึ่งจะสลับบิตข้อมูลตามตามอัลกอริทึมดีอีเอส ระหว่างการเชื่อมต่อส่วนพักข้อมูลชั่วคราวสำหรับคีย์ กับ ส่วนจัดการกับคีย์

3.11 ส่วนการทำเอ็กออ 48 บิต (โมดูล Exclusive-OR 48 ในภาคผนวก)

ส่วนนี้จะทำการกระทำเอ็กออแบบบิตต่อบิตกับข้อมูลจำนวน 48 บิต โดยมีสัญญาณจากส่วนควบคุมเป็นตัวกำหนดให้เริ่มการกระทำเอ็กออ ซึ่งสามารถแสดงด้วยวงจรซีเควนเชียว ซึ่งแสดงการทำงานได้ดังนี้

- ถ้าสัญญาณควบคุมเป็น 1 ให้ทำการเอ็กออค่าทั้ง 48 บิต

3.12 ส่วนการทำเอ็กออ 32 บิต (โมดูล Exclusive-OR 32 ในภาคผนวก)

ส่วนนี้จะทำการกระทำเอ็กออแบบบิตต่อบิตกับข้อมูลจำนวน 32 บิต โดยมีสัญญาณจากส่วนควบคุมเป็นตัวกำหนดให้เริ่มการกระทำเอ็กออ ซึ่งสามารถแสดงด้วยวงจรซีเควนเชียว ซึ่งแสดงการทำงานได้ดังนี้

- ถ้าสัญญาณควบคุมเป็น 1 ให้ทำการเอ็กออค่าทั้ง 32 บิต

3.13 ส่วนการทำเอ็สบล็อก (โมดูล S-Box ในภาคผนวก)

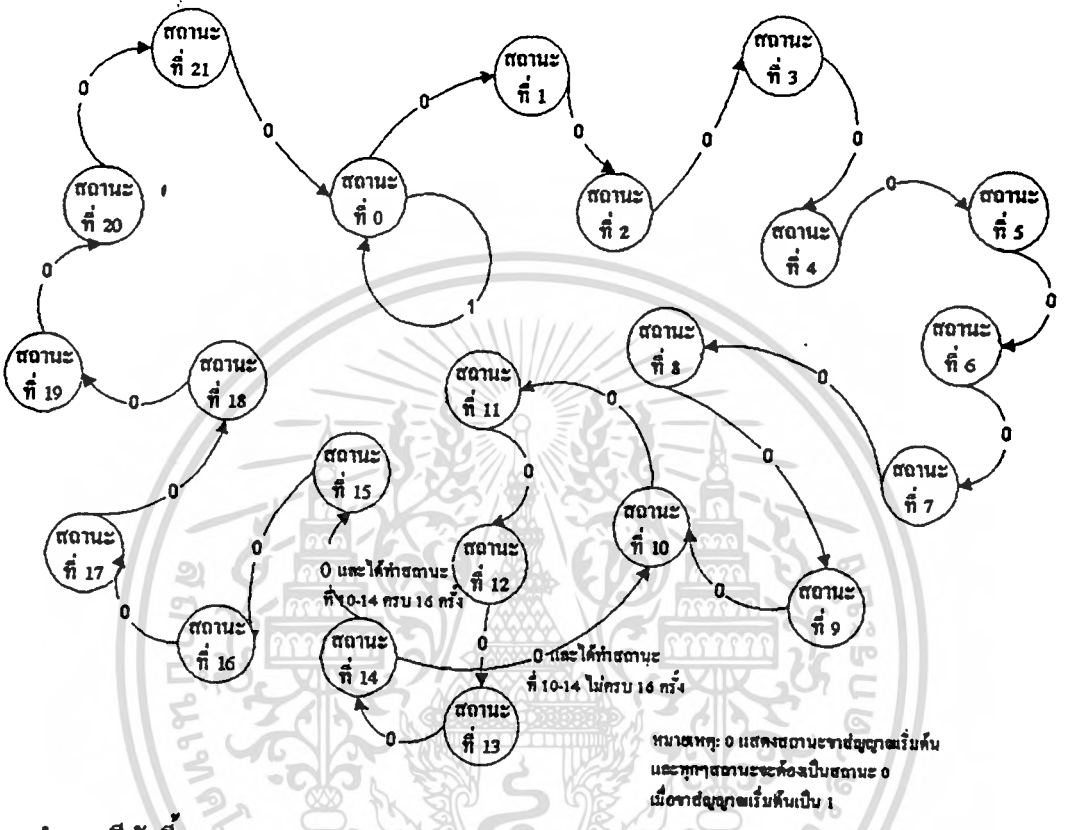
ส่วนนี้จะทำการแทนรูปแบบของข้อมูลขนาด 48 บิตด้วยรูปแบบอีกแบบหนึ่งที่มีขนาด 32 บิต ตามอัลกอริทึมดีอีเอส สามารถแสดงด้วยวงจรซีเควนเชียว

3.14 ส่วนการทำพีบล็อก

ส่วนนี้จะทำการสลับตำแหน่งของข้อมูลขนาด 32 บิต ตามอัลกอริทึมดีอีเอส สามารถแสดงส่วนนี้โดยการใช้คุณสมบัติของภาษาวีเอชดีแอลโดยเขียนแบบสตรักเทอรัล ซึ่งจะสลับบิตข้อมูลตามตามอัลกอริทึมดีอีเอส ระหว่างการเชื่อมต่อส่วนพักข้อมูลชั่วคราวสำหรับคีย์ กับ ส่วนจัดการกับคีย์

3.15 ส่วนควบคุม (โมดูล Signal Control ในภาคผนวก)

ส่วนนี้มีหน้าที่ส่งสัญญาณเริ่มต้นและสัญญาณควบคุมให้กับทุกๆส่วน โดยอาศัยสัญญาณนาฬิกาจากภายนอก เป็นตัวกำหนดจังหวะ สามารถแสดงการทำงานโดยใช้ไฟในสเตตแมชีนได้ดังนี้



การทำงานมีดังนี้

- ถ้าสัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการตั้งสถานะของตน เองให้เป็นสถานะเริ่มต้นหรือสถานะที่ 0

- ถ้าอยู่ในสถานะที่ 0 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ไปยังขาสัญญาณเริ่มต้นและขาสัญญาณควบคุม ให้กับทุกๆส่วนที่เป็นไฟในสเตตแมชีน จากนั้นเปลี่ยนสถานเป็น 1

- ถ้าอยู่ในสถานะที่ 1 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังขาสัญญาณเริ่มต้นและขาสัญญาณควบคุม ซึ่งขณะนี้ทุกๆส่วนจะมีสถานะเริ่มต้น จากนั้นเปลี่ยนสถานะของตัวเองให้เป็น 2

- ถ้าอยู่ในสถานะที่ 2 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 0 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ไปยังขาสัญญาณควบคุมของทุกๆส่วน จากนั้นเปลี่ยนสถานะเป็น 4

- ถ้าอยู่ในสถานะที่ 3 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์เพื่อสั่งให้ทำการรับข้อมูล 16 บิตแรก จากนั้นเปลี่ยนสถานะเป็น 4

- ถ้าอยู่ในสถานะที่ 4 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์ จากนั้นเปลี่ยนสถานะเป็น 5

- ถ้าอยู่ในสถานะที่ 5 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์เพื่อสั่งให้ทำการรับข้อมูล 16 บิตที่ 2 จากนั้นเปลี่ยนสถานะเป็น 6

- ถ้าอยู่ในสถานะที่ 6 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลงจะทำการส่งสัญญาณ 1 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์ จากนั้นเปลี่ยนสถานะเป็น 7

- ถ้าอยู่ในสถานะที่ 7 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์เพื่อสั่งให้ทำการรับข้อมูล 16 บิตที่ 3 จากนั้นเปลี่ยนสถานะเป็น 8

- ถ้าอยู่ในสถานะที่ 8 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์ จากนั้นเปลี่ยนสถานะเป็น 9

- ถ้าอยู่ในสถานะที่ 9 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลเข้าและส่วนพักข้อมูลชั่วคราวสำหรับคีย์เพื่อสั่งให้ทำการรับข้อมูล 16 บิตที่ 4 จากนั้นเปลี่ยนสถานะเป็น 10

- ถ้าอยู่ในสถานะที่ 10 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนเลือกข้อมูลเข้าและส่วนจัดการคีย์เพื่อให้ทำงาน จากนั้นเปลี่ยนสถานะเป็น 11

- ถ้าอยู่ในสถานะที่ 11 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนการทำเอ็กออ 48 บิตเพื่อให้ทำงาน จากนั้นเปลี่ยนสถานะเป็น 12

- ถ้าอยู่ในสถานะที่ 12 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนส่วนการทำเอ็กออ 32 บิต เพื่อสั่งให้ทำงาน จากนั้นเปลี่ยนสถานะเป็น 13

- ถ้าอยู่ในสถานะที่ 13 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนส่วนการเลือกข้อมูลออกเพื่อสั่งให้ทำงาน จากนั้นเปลี่ยนสถานะเป็น 14

- ถ้าอยู่ในสถานะที่ 14 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการตรวจสอบว่าได้กระทำตามสถานะที่ 10 ถึง 14 กี่ครั้งแล้ว ถ้าทำครบ 16 ครั้ง ก็จะเปลี่ยนสถานะเป็น 15 แต่ถ้ายังไม่ครบก็จะเปลี่ยนสถานะเป็น 10

- ถ้าอยู่ในสถานะที่ 15 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออกเพื่อสั่งให้ทำการส่งข้อมูลออก 16 บิตแรก จากนั้นเปลี่ยนสถานะเป็น 16

- ถ้าอยู่ในสถานะที่ 16 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออก จากนั้นเปลี่ยนสถานะเป็น 17

- ถ้าอยู่ในสถานะที่ 17 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออกเพื่อสั่งให้ทำการส่งข้อมูลออก 16 บิตที่ 2 จากนั้นเปลี่ยนสถานะเป็น 18

- ถ้าอยู่ในสถานะที่ 18 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออก จากนั้นเปลี่ยนสถานะเป็น 19

- ถ้าอยู่ในสถานะที่ 19 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออกเพื่อสั่งให้ทำการส่งข้อมูลออก 16 บิตที่ 3 จากนั้นเปลี่ยนสถานะเป็น 20

- ถ้าอยู่ในสถานะที่ 20 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 1 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออก จากนั้นเปลี่ยนสถานะเป็น 21

- ถ้าอยู่ในสถานะที่ 21 สัญญาณเริ่มต้นวงจรมีค่าเท่ากับ 1 และสัญญาณนาฬิกาเป็นขอบขาลง จะทำการส่งสัญญาณ 0 ไปยังส่วนพักข้อมูลชั่วคราวสำหรับข้อมูลออกเพื่อสั่งให้ทำการส่งข้อมูลออก 16 บิตที่ 4 จากนั้นเปลี่ยนสถานะเป็น 0

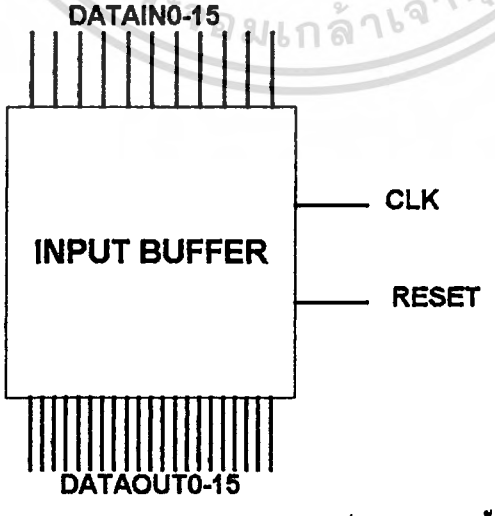
บทที่ 5
การสร้างโค้ดวีเฮชดีแอล

จากบทที่ 4 ได้กล่าวถึงขั้นตอนในการแปลงอัลกอริทึมของดีอีเอส ให้เหมาะสมในการเขียนโปรแกรมภาษาวีเฮชดีแอล ในบทนี้จะได้อธิบายถึงรายละเอียดของแต่ละโมดูลที่แปลงมาจากอัลริทึมของดีอีเอส โดยที่ส่วนของโค้ดจะอยู่ในส่วนของภาคผนวกซึ่งอยู่ในส่วนท้ายของปริญญานิพนธ์ฉบับนี้

5.2 การทำงานของแต่ละโมดูล

1. โมดูล Input Buffer และ Key Input 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;
```



มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานและเปลี่ยนสถานะ

Reset เป็นสัญญาณให้มีการเริ่มการทำงาน

Datain ขนาด 16 บิท เป็นขาที่รับข้อมูลที่จะเข้ามาทำการเข้าหรือถอดรหัส

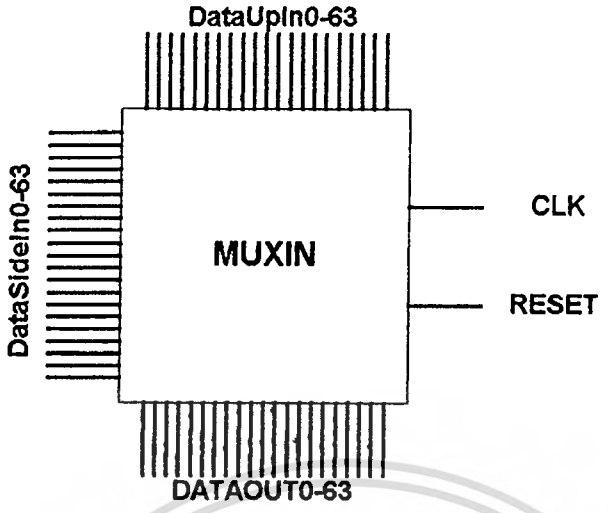
และขาออกคือ Dataout ขนาด 64 บิท เป็นขาที่จะส่งข้อมูลออกจากโมดูล

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

โมดูลนี้มีการทำงานคือรับข้อมูลจำนวน 16 บิท 4 ชุดมาทำการมัลติเพลกเซอร์ เป็นข้อมูลขนาด 64 บิทเพื่อส่งให้โมดูลอื่นเพื่อทำงานต่อไป

2. โมดูล Input Multiplexor มีการประกาศ Entity ดังนี้

```
Entity MUXIN is
   Port(CLK : IN
        DataUpIn0, DataUpIn1, DataUpIn2, DataUpIn3,DataUpIn4, DataUpIn5,
        DataUpIn6, DataUpIn7, DataUpIn8, DataUpIn9, DataUpIn10,DataUpIn11,
        DataUpIn12, DataUpIn13, DataUpIn14, DataUpIn15, DataUpIn16, DataUpIn17,
        DataUpIn18, DataUpIn19,DataUpIn28, 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;
```



มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานและเปลี่ยนสถานะ

DataUpIn ขนาด 64 บิท จะรับข้อมูลมาจากโมดูล Input Buffer ซึ่งจะเป็นข้อมูลที่จะทำการเข้ารหัสหรือถอดรหัสโดยจะรับเฉพาะในรอบแรกของการทำงานเท่านั้น

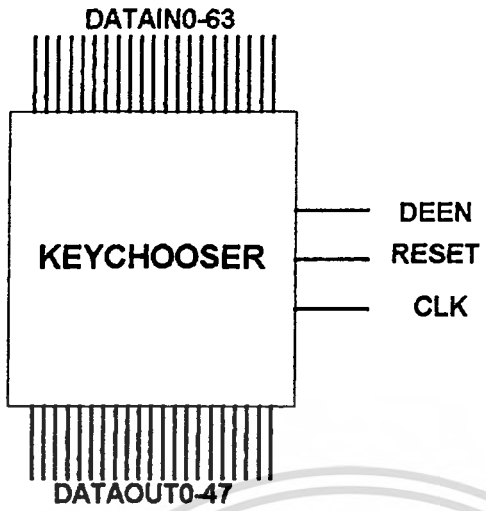
DataSidein ขนาด 64 บิท จะรับข้อมูลที่ผ่านการทำงานในแต่ละรอบมาแล้วโดยจะรับมาจากโมดูล Output Multiplexor

ขาออกคือ DataOut ขนาด 64 บิท จะส่งข้อมูลให้โมดูลต่อไป

โมดูลนี้มีการทำงานคือ ในรอบที่ 1 จะรับข้อมูลจากขา DataUpIn ซึ่งเป็นขาที่จะรับข้อมูลมาจากภายนอก ส่วนรอบที่ 2 ถึง 16 จะรับข้อมูลที่จะวนเข้ามาทำการเข้ารหัสหรือถอดรหัส 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 ;
```



มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานและเปลี่ยนสถานะ

DataIn ขนาด 64 บิทจะรับข้อมูลจากโมดูล Key Input Buffer

DeEn เป็นขาสัญญาณเพื่อสั่งให้ทำการเข้ารหัสหรือถอดรหัส

Reset เป็นขาสัญญาณเพื่อให้เริ่มต้นการทำงานของโมดูล

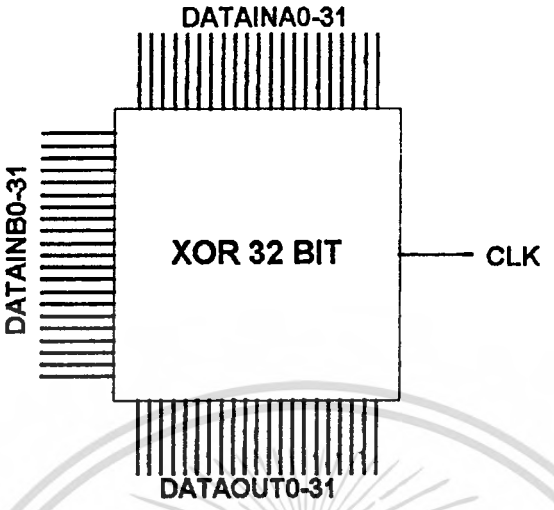
ขาออกคือ DataOut ขนาด 56 บิทจะส่งข้อมูลให้โมดูลอื่นทำงานต่อไป

โมดูลนี้มีการทำงานคือ จะรับ Key ขนาด 64 บิทจากโมดูล Key Input Buffer มาแล้วทำการตัดบิทที่ 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 บิท และ 48 บิท 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);
```

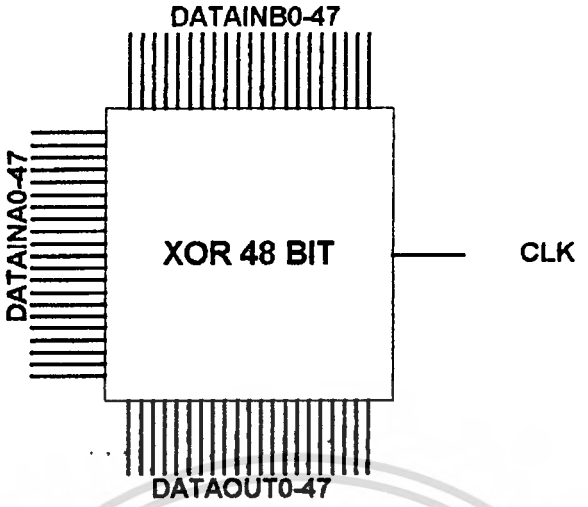
end XOR32;



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,DatainB14,
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;



มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานและเปลี่ยนสถานะ

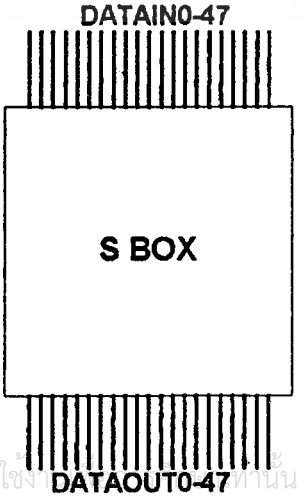
DatainA และ DatainB เป็นขาข้อมูลที่จะทำการExclusive OR

ขาออกคือ Dataout เพื่อส่งข้อมูลให้โมดูลอื่น

โมดูลนี้มีการทำงานคือรับข้อมูลเข้ามา 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;
```



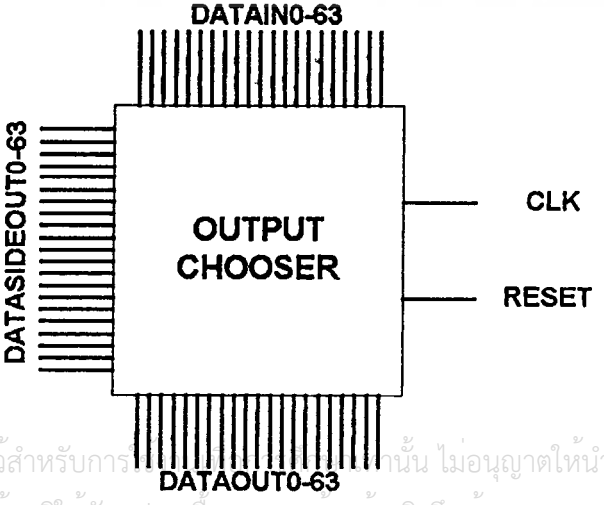
มีสัญญาณขาเข้าคือ DataIn ขนาด 48 บิท รับข้อมูลเข้า

ขาออกคือ DataOut ขนาด 32 บิท ส่งข้อมูลออก

โมดูลนี้มีการทำงานคือ นำข้อมูล 48 บิท มาแบ่งเป็น 8 ชุดๆละ 6 บิท และทำการผ่านตาราง ลดจำนวนลงเหลือ 8 ชุดๆละ 4 บิท รวม 32 บิท 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;
```



มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานและเปลี่ยนสถานะ

Reset เป็นสัญญาณเพื่อเริ่มต้นการทำงานของโมดูล

DataIn ขนาด 64 บิท เป็นขาข้อมูลที่รับจากโมดูลอื่น

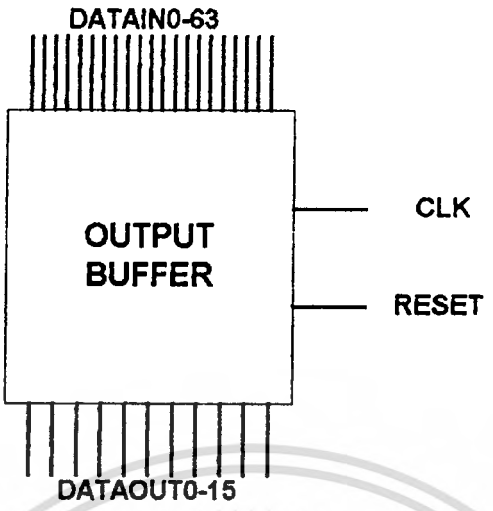
ขาออกคือ DataDownout ขนาด 64 บิท เป็นขาข้อมูลที่จะส่งไปยังโมดูล Output Buffer ในรอบที่ 16 ของการทำงาน เพื่อส่งเป็น Output ต่อไป

- DataSideout ขนาด 64 บิท เป็นขาข้อมูลที่จะส่งไปยังโมดูล 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;
```



มีขาสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาเพื่อให้โมดูลนี้มีการทำงานและเปลี่ยนสถานะ

Reset เป็นขาสัญญาณเพื่อให้โมดูลเริ่มต้นทำงาน

DataIn ขนาด 64 บิท เป็นขาข้อมูลที่รับมาจากการเข้ารหัสและถอดรหัสที่ทำครบ 16 รอบ

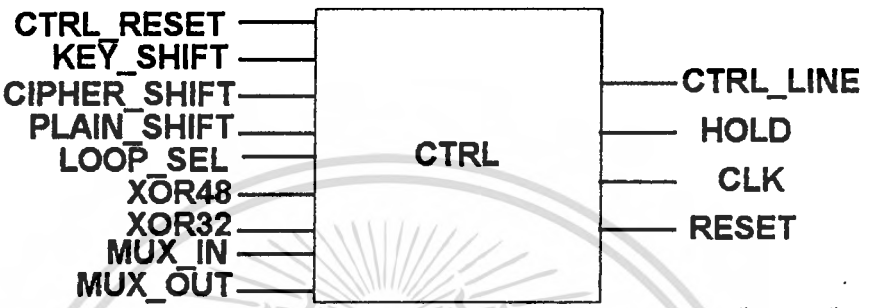
ขาออกคือ DataOut ขนาด 16 บิท เป็นขาข้อมูลที่จะนำข้อมูล 64 บิทมาแบ่งส่งครั้งละ 16 บิท 4 ชุด

โมดูลนี้มีการทำงานคือ นำข้อมูลขนาด 64 บิท มาทำการมัลติเพลกออกครั้งละ 16 บิท 4 ครั้ง

8. โมดูล Signal Control มีการประกาศ Entity

```
Entity CTRL  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;
```



มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาของระบบเพื่อให้ดีอีเอสทั้งหมดนี้มีการทำงานและเปลี่ยนสถานะ

Reset เป็นขาสัญญาณเพื่อเริ่มต้นการทำงานของดีอีเอส และเคลียร์สถานะทั้งหมดของระบบ

ขาออกคือ CTRL_RESET เป็นขาสัญญาณที่ส่งไปควบคุมโมดูลต่างๆ เพื่อทำการรีเซตแต่ละโมดูล

CTRL_LINE เป็นขาสัญญาณที่จะส่งไปยังอุปกรณ์ที่จะนำมาใช้กับดีอีเอส เพื่อบอกให้รู้ว่าพร้อมที่จะรับหรือส่งข้อมูลที่ทำการเข้ารหัสหรือถอดรหัส

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

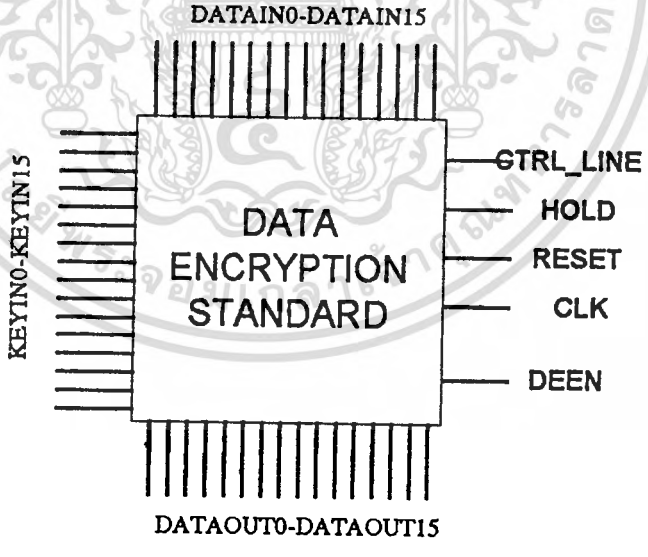
MUX_IN เป็นขาสัญญาณที่จะส่งไปเป็น clock ให้โมดูล Input Multiplexor

MUX_OUT เป็นขาสัญญาณที่จะส่งไปเป็น clock ให้โมดูล Output Multiplexor

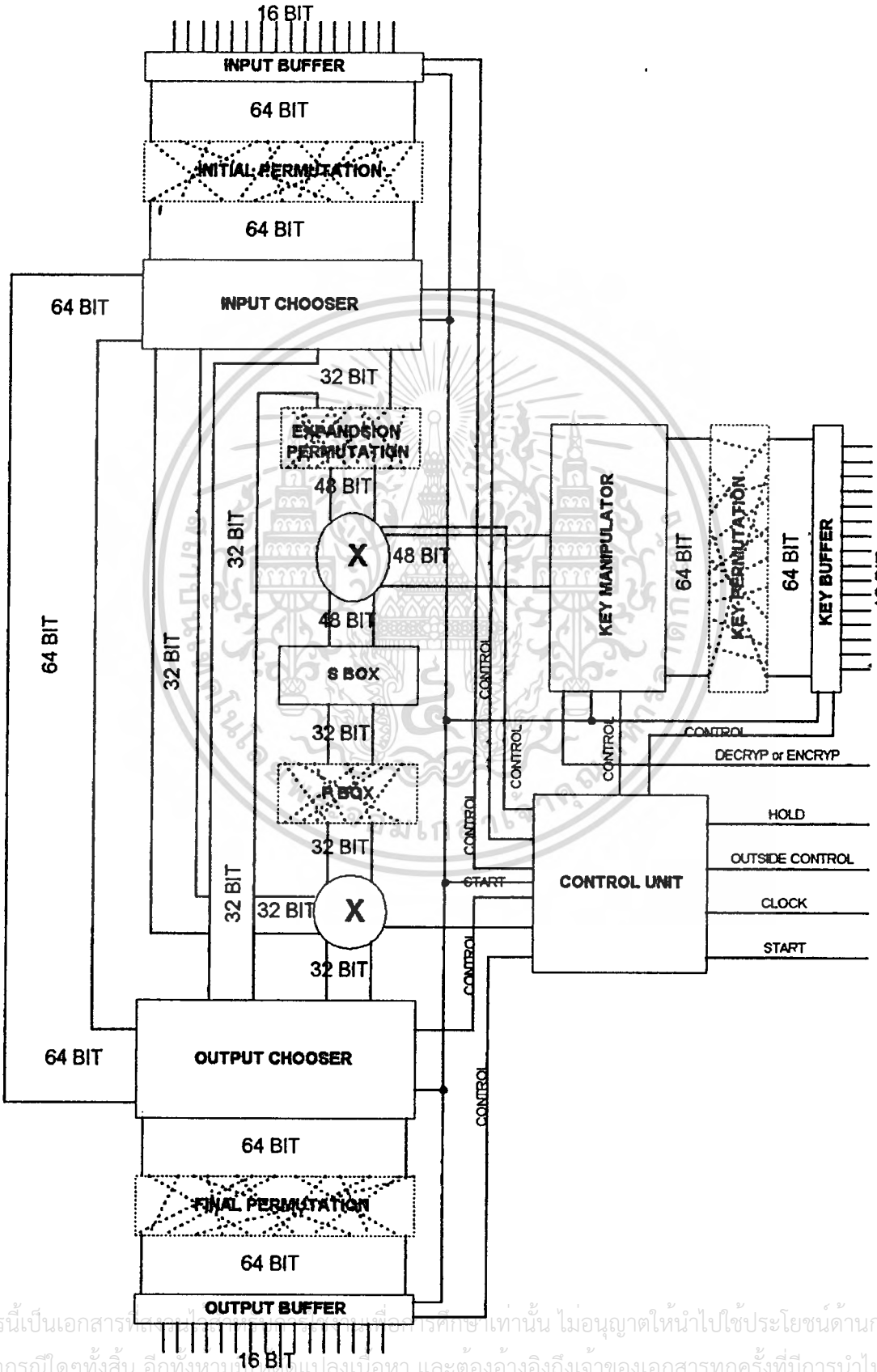
โมดูลนี้มีการทำงานคือ จะสร้างสัญญาณเพื่อรีเซตโมดูลต่างๆ และสร้าง clock เพื่อให้โมดูลต่างๆมีการทำงานซึ่งสัมพันธ์กัน เพื่อควบคุมการทำงานให้สอดคล้องกัน โดยจะเคลียร์สถานะทั้งหมดเมื่อมีสัญญาณ Reset ซึ่งเปลี่ยนจาก 1 เป็น 0 จากนั้นจะสร้างสัญญาณ Ctrl_reset เพื่อเคลียร์สถานะของแต่ละโมดูลย่อย แล้วทำการส่ง Clock 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;
```



รูปแสดงส่วนประกอบต่างๆและการเชื่อมต่อภายในวงจร



มีสัญญาณขาเข้าคือ CLK เป็นสัญญาณนาฬิกาของระบบซึ่งจะรับมาจากภายนอกดีอีเอส

CS เป็นสัญญาณ Chip Select เพื่อบอกให้รู้ว่าจะให้ดีอีเอสทำงาน

DataIn และ KeyIn ขนาด 16 บิท เป็นขาข้อมูลซึ่งจะรับข้อมูลและ Key 64 บิทโดยรับเป็น 16 บิท 4 ชุด

DeEn เป็นขาสัญญาณเพื่อบอกว่าเป็นการเข้ารหัสหรือถอดรหัส

ขาออกคือ HOLD เป็นขาสัญญาณเพื่อให้อุปกรณ์ภายนอกรู้ว่าพร้อมจะรับและจะมีการส่งข้อมูล โดยจะเปลี่ยนจาก 0 เป็น 1

CTRL_LINE เป็นขาสัญญาณเพื่อให้อุปกรณ์ภายนอกส่งหรือรับข้อมูล 16 บิท ชุดต่อไป

Dataout ขนาด 16 บิท เป็นขาข้อมูลที่ออกจากดีอีเอสซึ่งจะส่งเป็นจำนวน 4 ชุด เป็นข้อมูลขนาด 64 บิท

โมดูลนี้เป็นโมดูลซึ่งรวมโมดูลย่อยทุกๆโมดูล โดยจะเริ่มทำงานเมื่อได้รับสัญญาณ CS จากอุปกรณ์ภายนอก และจะทำงานตาม clock ของระบบ เมื่อจะทำการรับข้อมูลจะส่งสัญญาณ HOLD พร้อมกับ CTRL_LINE โดยที่ HOLD จะบอกให้รู้ว่าพร้อมที่จะรับข้อมูลโดยจะเปลี่ยนจาก 0 เป็น 1 ส่วน CTRL_LINE จะบอกให้รู้ว่าให้ส่งข้อมูล 16 บิทแต่ละชุดที่ DataIn ได้ เช่นเดียวกับการส่งข้อมูลจะส่งสัญญาณ HOLD และ CTRL_LINE เช่นกัน แต่จะรอรับข้อมูลออกจากขา DataOut ซึ่งจะส่งเป็นชุดๆ โดยใช้ CTRL_LINE บอกเช่นกัน

5.2 เวลาที่ใช้ในการเข้าหรือถอดรหัสข้อมูล

ข้อมูลที่จะส่งเข้าทำการเข้าหรือถอดรหัสจะส่งเป็นชุดละ 16 บิท 4 ชุด และจะได้ข้อมูล 16 บิท 4 ชุด แล้วจึงส่งข้อมูลชุดต่อไปเข้าไป เพราะฉะนั้นการทำงาน 1 รอบจะทำการเข้าหรือถอดรหัสได้ครั้งละ 64 บิท

ในการทดลองได้ป้อน Clock ให้ระบบมีขนาด 4×10^{-9} วินาที โดยจะใช้ Clock ทั้งหมดจำนวน 148 Cycle ต่อ 1 รอบ ซึ่งจะได้ข้อมูล 64 บิท ฉะนั้นจะใช้เวลา 5.92×10^{-6} วินาที สำหรับข้อมูล 1 ชุด 64 บิท

ดังนั้นจะแสดงให้เห็นได้ว่าถ้ามีอุปกรณ์ที่ใช้ทำงานร่วมกับดีอีเอสที่ความถี่ 250 Mhz ดีอีเอสจะสามารถเข้าและถอดรหัสข้อมูลได้ที่ความเร็ว 10.8 Mbps

บทที่ 6
บทสรุปและวิจารณ์

บทสรุป

การออกแบบวงจรรวมในปัจจุบันทำได้รวดเร็วและมีประสิทธิภาพมาก เนื่องจากการพัฒนาเครื่องมือต่าง ๆ ทั้งทางด้านฮาร์ดแวร์ และซอฟท์แวร์ ระบบที่ช่วยออกแบบทางอิเล็กทรอนิกส์ (Electronic Design Automation : EDA) ก็ได้พัฒนาขึ้นมาจนทำให้การออกแบบสามารถทำได้อย่างรวดเร็ว

วีเอชดีแอลเป็นภาษาที่ใช้เขียนอธิบายการทำงานของวงจรใด ๆ แทนการสร้างวงจรขึ้นมาจริง ใช้ประโยชน์ในการทำซิมูเลชันเพื่อนำไปสร้างวงจรจริงได้ ในปัจจุบันจะเห็นว่ามีวงจรรวมเอเอสไอซีใหม่ ๆ เป็นผลิตภัณฑ์ออกมามากมาย เนื่องมาจากเทคโนโลยีในการออกแบบวงจรรวมได้พัฒนาขึ้นไปนั่นเอง การศึกษาโครงงานนี้ทำให้สามารถนำความรู้ที่ได้ไปใช้ประโยชน์ได้ในวงการอุตสาหกรรมจริง ๆ และคิดว่าในอนาคตคงจะมีการใช้เครื่องมือที่มีอยู่ออกแบบและพัฒนาสิ่งประดิษฐ์ใหม่ ๆ สู่วงการอิเล็กทรอนิกส์ต่อไป

ส่วนการเข้ารหัสและถอดรหัสข้อมูลด้วยอัลกอริทึมดีอีเอส (DES) เป็นวิธีการที่มีความปลอดภัยสูง และสามารถนำมาประยุกต์ใช้ได้ง่าย จึงเหมาะสมแก่การนำมาใช้เพื่อความปลอดภัยของการสื่อสารข้อมูล

บทวิจารณ์

ระบบอีดีเอเป็นการออกแบบที่ใช้คอมพิวเตอร์ทั้งฮาร์ดแวร์และซอฟท์แวร์ ช่วยในการออกแบบวงจรอิเล็กทรอนิกส์ต่าง ๆ เป็นเทคโนโลยีแบบใหม่ทันสมัย มีมากมายหลายแพล็ตฟอร์ม (Platform) ทั้งพีซี (Personal Computer) และ เวิร์กสเตชัน (Work Station) ซอฟท์แวร์ที่ใช้บนแต่ละแพล็ตฟอร์มก็แตกต่างกันไป การเรียนรู้แต่ละรูปแบบใช้เวลานานมาก เนื่องจากไม่คุ้นเคย ทำให้การทำงานโครงงานเสียเวลาไปมากกับการศึกษาเครื่องมือต่าง ๆ เหล่านี้ และหลายครั้งต้องเสียเวลากับการแก้ปัญหาที่เกิดขึ้น และเนื่องจากโปรแกรมดีอีเอสที่สร้างขึ้นมีขนาดใหญ่ เครื่องคอมพิวเตอร์ที่ใช้ก็ควรจะเป็นเครื่องที่มีความสามารถและความเร็วสูงเช่น Intel Pentium-100 และจะต้องมีแรมเพียงพอ (มากกว่า 32 เมกกะไบต์) มิฉะนั้นจะไม่สามารถสังเคราะห์วงจรออกมาได้ ดังนั้นก่อนจะทำการพัฒนาโครงงานใด ควรจะศึกษาให้มั่นใจเสียก่อนว่ามีฮาร์ดแวร์ที่จะสามารถรองรับการใช้งานได้ มิฉะนั้นเมื่อพัฒนาขึ้นมาจะไม่เกิดประโยชน์เนื่องจากทำได้เพียงครึ่งๆกลางๆ ส่วนทางด้านซอฟท์แวร์ที่มีใช้อยู่ยังไม่มีตัวใดที่สามารถทำงานได้ตลอดขบวนการพัฒนาจะต้องใช้หลาย ๆ ตัวมาช่วยพัฒนา บางครั้งเกิดความเข้ากันไม่ได้ระหว่าง

ซอฟท์แวร์ต่าง ๆ เหล่านั้น ทำให้เกิดปัญหาในการพัฒนา ฉะนั้นในการออกแบบควรจะศึกษาเครื่องมือที่จะใช้ให้ดี ศึกษาความเป็นไปได้ในการใช้ซอฟท์แวร์ เครื่องมือต่าง ๆ ให้ดีเสียก่อนที่จะเริ่มลงมือทำการออกแบบ

กิตติกรรมประกาศ

ผู้จัดทำขอขอบคุณ

อาจารย์บรรจง ปิยธำรง ที่ให้คำปรึกษาและคำแนะนำ รวมทั้งเอื้อเฟื้ออุปกรณ์ต่าง ๆ ในการทำโครงงาน และ สถานที่ทำโครงงาน

นายนิวัตน์ วโรภาษ์ ที่เอื้อเฟื้อเครื่องคอมพิวเตอร์ในการทำโครงงานในภาคเรียนที่ 1

นายอนุรัตน์ ธนะโสธร ที่เอื้อเฟื้อเครื่องคอมพิวเตอร์เพื่อทำการสร้างวงจรรวม

นายชลชาสน์ โพธารส ที่เอื้อเฟื้อเครื่องคอมพิวเตอร์ในการพิมพ์ปริญญานิพนธ์

ผู้จัดทำ

บรรณานุกรม

1. Jayaram Bhasker, "VHDL Primer" , Prentice Hall , First Edition , 1992.
2. Mentor Graphics, "An Introduction to Modeling with VHDL" , Mentor Graphics , 1992
3. Viewlogic, "Using Workview Plus for Windows" , Viewlogic , 1993
4. Viewlogic, "Viewsim Reference Manual" , Viewlogic , 1993
5. Viewlogic, "VHDL User's Guide" ,Viewlogic , 1993
6. Warwick Ford, "Computer Communications Security" , Prentice Hall , First Edition , 1994 , pp. 69-71
7. Charlie Kaufman, Radia Perlman and Mike Speciner, "Network Security" , Prentice Hall , First Edition , 1995 , pp. 60-73
8. สุรเชษฐ์ ศรีพลกรัง, "การออกแบบวงจรรวมดิจิตอลโดยใช้วีแอชดีแอล" , คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2537 , หน้า 23-55