สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การจำลองการเข้าและถอดรหัสด้วยภาษาวีเอชดีแอล

CRYPTOGRAPHY SIMULATION USING VHDL

นายอรรถวุฒิ สุขผดุง

นายเอกรินทร์ เสรีชื่นพอจิต

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เลขหมู่.....
เลขทะเบียน 42840
วัน, เดือน, ปี 10 มิ.ย. 2545

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจำลองการเข้าและถอดรหัสด้วยภาษาวีเอชดีแอล

CRYPTOGRAPHY SIMULATION USING VHDL

โดย

นายอรรถวุฒิ สุขผดุง

นายเอกรินทร์ เสรีชื่นพอจิต

อาจารย์ที่ปรึกษา

รศ. สมศักดิ์ มิตะถา

อาจารย์อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การจำลองการเข้าและถอดรหัสด้วยภาษาวีเอชดีแอล

CRYPTOGRAPHY SIMULATION USING VHDL

ผู้จัดทำ

1. นาย อรรถวุฒิ สุขผดุง รหัสประจำตัว 40010968
2. นาย เอกรินทร์ เสรีชื่นพอจิต รหัสประจำตัว 40011039

_____ อาจารย์ที่ปรึกษา

(รศ. สมศักดิ์ มิตะถา)

_____ อาจารย์ที่ปรึกษา

(อาจารย์อัครเดช วัชระภูพงษ์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจำลองการเข้าและถอดรหัสด้วยภาษาวีเอชดีแอล

นายอรรถวุฒิ สุขผดุง 40010968
นายเอกรินทร์ เสรีชื่นพอจิต 40011039
รศ. สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา
อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

โครงงานนี้เสนอการจำลองการเข้าและถอดรหัสลับ สำหรับทำหน้าที่เป็นคอมโพเน็นต์รักษาความปลอดภัยในระบบคอมพิวเตอร์โดยใช้ภาษาวีเอชดีแอล ซึ่งเป็นภาษาที่บรรยายถึงลักษณะการทำงานของฮาร์ดแวร์ และการออกแบบโปรแกรมที่สนับสนุนการออกแบบจากบนลงล่าง (Top-Down Design) โดยไม่อิงกับเทคโนโลยีเฉพาะผู้ผลิตรายใด ทั้งยังสามารถบรรยายลักษณะการทำงานได้ถึงระดับเกต จึงเหมาะสมในการนำมาใช้สำหรับการออกแบบ ในโครงงานนี้เราได้ออกแบบการทำงานของกระบวนการเข้าและถอดรหัส พร้อมทั้งองค์ประกอบที่จำเป็นในการเข้าและถอดรหัสโดยประยุกต์ด้วยภาษาวีเอชดีแอล โดยเลือกอัลกอริทึมที่เป็นที่นิยมอย่างแพร่หลายได้แก่ Triple DES, Blowfish, SHA, MD5 และยังมีส่วนของ Psudo Random Number Generator เพื่อสุ่มค่าตัวเลขให้กับการคำนวณ ทั้งหมดนี้เป็นประโยชน์แก่ผู้พัฒนาภายหลังให้สามารถใช้งานได้จริงบนฮาร์ดแวร์ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

CRYPTOGRAPHY SIMULATION USING VHDL

Attawoot Supadoong

Ekkarin Serichoenporchit

Assoc. Prof. Somsak Mitatha Advisor

Akkaradech Watcharapuphong Advisor

ABSTRACT

This thesis presents the cryptography simulation for security system using VHDL (Very high-speed integrated circuit Hardware Description Language). VHDL is a language that uses for hardware description and supports top-down design it does not depend on any technology. So it is suitable for hardware design

We designed processes of encryption and decryption includes some needed components i.e. the famous algorithm Triple Data Encryption Standard (3DES), BlowFish, Secure Hash Algorithm (SHA) and Message Degest Algorithm (MD5), and also Pseudo Random Number Generator (PRNG) for key generation. All of these components are very useful for developers to implement on hardware devices.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดีจากความร่วมมือของหลายๆ ฝ่าย ขอบคุณอาจารย์ที่ปรึกษา รศ. สมศักดิ์ มิตะถา และ อาจารย์ อัครเดช วัชระภูพงษ์ ที่คอยให้ความเอาใจใส่ช่วยเหลือในการทำงาน รวมทั้งพี่ๆ เพื่อนๆ ห้องฮาร์ดแวร์ (Hardware LAB) และห้องโอลันลา (OLALA) ที่คอยให้คำปรึกษาในการทำงานที่ดียิ่ง

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมาในทุกๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

อรรถวุฒิ สุขผดุง

เอกรินทร์ เสรีชื่นพอจิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ที่มาของงานโครงงาน

ปัจจุบันเทคโนโลยีคอมพิวเตอร์ได้พัฒนาไปอย่างรวดเร็ว การสื่อสารข้อมูลต่างๆก็สามารถทำได้อย่างรวดเร็วเช่นกัน ดังนั้นความปลอดภัยในการรักษาข้อมูลที่เป็นความลับจึงมีความสำคัญอย่างยิ่ง ถึงแม้ว่าปัจจุบันจะมีระบบควบคุมความปลอดภัยข้อมูลบนคอมพิวเตอร์ที่คอยช่วยอำนวยความสะดวกอย่างมากมายก็ตาม เช่น การใช้รหัสผ่าน, การเข้ารหัสข้อมูล เป็นต้น แต่สิ่งเหล่านี้ล้วนต้องมีเครื่องคอมพิวเตอร์เพื่อใช้ในการประมวลผลทั้งสิ้น ดังนั้นหากเราสามารถสร้างคอมโพเน็นต์ (Component) เฉพาะสำหรับการเข้าและถอดรหัสที่สามารถอิมพลีเมนต์ (Implement) บนชิป (Chip) ที่เป็นฮาร์ดแวร์ (Hardware) ได้ จะทำให้เราสามารถนำอุปกรณ์ดังกล่าวเพื่อใช้ในการเข้าและถอดรหัสได้อย่างสะดวก ทั้งยังสามารถนำไปประยุกต์ใช้กับวงจรเข้าและถอดรหัสอื่นๆได้ด้วย

การสร้างคอมโพเน็นต์สำหรับการเข้าและถอดรหัสจำเป็นต้องมีจะต้องเลือกอัลกอริทึมที่ได้รับความนิยมและเป็นที่ยอมรับในการใช้งาน ในที่นี้เราเลือกอัลกอริทึม Triple Data Encryption Standard (3DES) และ BlowFish เพื่อใช้ในการเข้าและถอดรหัส โดยทั้งสองอัลกอริทึมนี้เป็นอัลกอริทึมที่ได้รับความนิยมในแง่ความซับซ้อนในการทำงาน ความยาวคีย์ (Key) ที่มีขนาดพอเหมาะ สำหรับ Blowfish นั้นจะใช้เป็นตัวเลือกสำหรับการเข้าและถอดรหัส อัลกอริทึมต่อมาคือ Message Digest Algorithm (MD5) และ Secure Hash Algorithm (SHA) เป็นอัลกอริทึมที่ใช้ในการลงลายมือชื่อดิจิตอล

สำหรับภาษาที่ใช้ในการออกแบบจะใช้ภาษาวีเอชดีแอล (VHDL) ซึ่งเป็นภาษาที่บรรยายสถาปัตยกรรมของฮาร์ดแวร์ พร้อมทั้งสนับสนุนการออกแบบจากบนลงล่าง (Top-Down Design) เป็นภาษาที่เหมาะจะนำไปพัฒนาและอิมพลีเมนท์บนฮาร์ดแวร์ เพื่อเป็นชิปเข้ารหัสที่สามารถใช้งานได้จริง

1.2 วัตถุประสงค์

1. เพื่อศึกษากระบวนการเข้าและถอดรหัสเพื่อนำมาประยุกต์ด้วยภาษาวีเอชดีแอล VHDL
2. เพื่อศึกษาแนวทางการออกแบบและหน่วยการทำงานที่จำเป็นสำหรับชิปเข้าและถอดรหัส

1.3 ขอบเขตของการโครงงาน

1. ออกแบบโปรแกรมด้วยภาษาวีเอชดีแอลเพื่อสร้างคอมโพเน็นต์ในการเข้าและถอดรหัส
2. ออกแบบโปรแกรมด้วยภาษาวีเอชดีแอลเพื่อสร้างอัลกอริทึมในการหาค่าแฮช (Hash Algorithm)
3. ออกแบบโปรแกรมการสุ่มตัวเลขเทียม (Pseudo Random Number Generator)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.4 ผลที่คาดว่าจะได้รับ

เอชดีแอลดีไซน์ (HDL Design) สำหรับกระบวนการเข้าและถอดรหัส โดยประกอบด้วยอัลกอริทึมการเข้าและถอดรหัส อัลกอริทึมในการลงลายมือชื่อดิจิตอล และ อัลกอริทึมการสุ่มตัวเลขเทียม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ภาษาวีเอชดีแอล (VHDL)

วีเอชดีแอล (VHDL) ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC ย่อมาจาก Very Hign Speed Integrated Circiut) เป็นภาษาคอมพิวเตอร์ระดับสูง (Hign Level Language) ซึ่งใช้อธิบายการทำงานของระบบดิจิตอลฮาร์ดแวร์ สามารถอธิบายการทำงานของฟังก์ชันได้หลายระดับ ตั้งแต่ระดับบล็อกจนถึงระดับเกต ความซับซ้อนของระบบสามารถจะเขียนได้ตั้งแต่ระดับเกต ประกอบกันจนเป็นระบบที่สมบูรณ์ รูปแบบของภาษาวีเอชดีแอลนั้นจะประกอบไปด้วยสองส่วนใหญ่ๆได้แก่ ส่วนของภาษาซีเควนเชียล (Sequential Language) และภาษาคอนเคอร์เรนต์ (Concurrent Language) การโปรแกรมด้วยภาษาวีเอชดีแอลสามารถเขียนได้ทั้งสองรูปแบบรวมกัน เพราะในการทำงานของระบบใดๆย่อมมีการทำงานในระบบซิเควนเชียล และคอนเคอร์เรนต์อยู่รวมกัน นอกจากนี้ภาษาวีเอชดีแอลยังสามารถอธิบายการเชื่อมต่อระหว่างระบบย่อยเข้าด้วยกันเพื่อเป็นระบบใหญ่ได้ ตัวภาษาวีเอชดีแอลนอกจากจะกำหนดรูปแบบในไวยากรณ์ของตัวภาษาแล้วยังสามารถตรวจสอบว่าจะซิมูเลต (Simulate) ได้หรือไม่ เพราะภาษาวีเอชดีแอลต้องผ่านการซิมูเลตเพื่อตรวจสอบการทำงาน ฉะนั้นการคอมไพล์ (Compile) จะต้องมีการตรวจสอบทั้งไวยากรณ์และซิมูเลชันซีแมนติค (Simulation Schemantic) อย่างไรก็ตาม แม้ตัวภาษาจะมีความซับซ้อนในรูปแบบและกฎเกณฑ์ของภาษา แต่การเรียนรู้เพียงบางส่วนของภาษาก็สามารถนำไปใช้งานได้โดยไม่จำเป็นต้องศึกษารายละเอียดทั้งหมด เนื่องจากตัวภาษาวีเอชดีแอลออกแบบมาให้ใช้สำหรับการออกแบบตั้งแต่วงจรที่มีขนาดเล็กจนถึงวงจรที่มีขนาดใหญ่และซับซ้อน

2.1 ประวัติความเป็นมาของภาษาวีเอชดีแอล

วิวัฒนาการของภาษาวีเอชดีแอล นั้นเริ่มต้นประมาณปี ค.ศ. 1981 โดยที่กระทรวงกลาโหมสหรัฐอเมริกา หรือ ดีโอดี (DoD: Department of Defense) มองเห็นว่าอุปกรณ์อิเล็กทรอนิกส์ (Electronic)และคอมพิวเตอร์ที่ใช้ในกิจการทางทหาร เป็นอุปกรณ์ที่ได้รับการพัฒนามาเมื่อประมาณ 20 ปีก่อน เพราะเทคโนโลยีในขณะนั้นทำให้การพัฒนาอุปกรณ์อิเล็กทรอนิกส์เป็นไปอย่างล่าช้า ซึ่งเป็นสภาพที่ไม่อาจยอมรับได้ในปัจจุบัน เพราะเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์ ได้รับการพัฒนาไปอย่างรวดเร็ว ดังที่จะเห็นได้ว่ามีวงจรดิจิตอลอิเล็กทรอนิกส์หลายวงจรที่แต่เดิมถูกสร้างขึ้นมาจากชิ้นส่วน ถูกนำประกอบกันอยู่บนแผงวงจรไฟฟ้าที่มีขนาดใหญ่ แต่ในปัจจุบันสามารถที่จะใช้เทคโนโลยีการออกแบบและผลิตวงจรรวมขนาดใหญ่มาก (VLSI: Very Large Scale Integration) รวมอุปกรณ์ต่างๆ เหล่านั้นให้อยู่บนชิ้นอุปกรณ์สารกึ่งตัวนำ ที่มีขนาดประมาณ 1-2 ตร.ซม. ได้ ซึ่งเป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้น (ความเร็วในการทำงานของวงจร) ตลอดจนความน่าเชื่อถือในการทำงาน และความคงทนต่อสภาพแวดล้อมสูง ขณะเดียวกันในวงการทหารได้มีการนำระบบคอมพิวเตอร์และอิเล็กทรอนิกส์ มาใช้ในระบบอาวุธอย่างแพร่หลาย ดังนั้นอุปกรณ์ที่มีใช้อยู่จึงไม่เหมาะสมกับเทคโนโลยีด้านอาวุธของประเทศคู่แข่งขัน การที่จะเปลี่ยนอุปกรณ์ใหม่เป็นสิ่งที่ต้องใช้งบประมาณมาก และก็จะประสบกับปัญหาเช่นเดิมคือ อุปกรณ์ใหม่ได้รับการพัฒนามานานแล้วเช่นกัน เพราะในขณะนั้นขั้นตอนของการออกแบบ การผลิต และ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การตรวจสอบวงจรต้นแบบ เป็นขบวนการที่ต้องใช้วิศวกรและเวลาสำหรับดำเนินการมาก ฉะนั้นทางดีโอดีจึงตั้งโครงการขึ้นมาเพื่อศึกษาวิธีการที่จะช่วยพัฒนาวงจรอิเล็กทรอนิกส์

โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอล ให้สามารถนำไปผลิตได้เร็วขึ้นและโครงการดังกล่าวมีชื่อว่า "Very High Speed Integrated Circuits" หรือ วีเอชเอสไอซี (VHSIC) ในระยะแรกนั้นโครงการเป็นความลับทางด้านความมั่งคงของประเทศและอยู่ในความดูแลควบคุมของ United States International Traffic and Arms Regulations หรือ ไอทีเออาร์ (ITAR) ในปี ค.ศ. 1983 ตามคำแนะนำของคณะทำงาน (Woods Hole workshop) ทางดีโอดีได้ออกความต้องการมาตรฐานของภาษาที่ใช้สำหรับบรรยายพฤติกรรมของวงจรหรือฮาร์ดแวร์ของระบบสำหรับโครงการวีเอชเอสไอซี ซึ่งมีสาระสำคัญสรุปได้ดังนี้

1. ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอลและมีคุณสมบัติที่สามารถจะเข้าใจได้ทั้งคนและเครื่องโดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก
2. สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้
3. ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

ฉะนั้นภาษาดังกล่าวนี้จึงจัดเป็นภาษาโปรแกรมระดับสูงเช่นเดียวกับภาษาปาสคาล (Pascal) หรือภาษาซี ซึ่งในทางวิศวกรรมการออกแบบฮาร์ดแวร์เรียกว่า "Hardware Description Language" หรือ เอชดีแอล (HDL) เริ่มต้นโครงการดีโอดีได้มอบหมายให้บริษัทไอบีเอ็มและบริษัทเท็กซัสอินสตูลเมนท์และบริษัทอินเตอร์เมทริกซ์เป็นผู้ศึกษาและพัฒนา การดำเนินการได้กระทำไปอย่างต่อเนื่องและได้ผลเป็นที่น่าพอใจ จนกระทั่งปี ค.ศ. 1985 ทางไอทีเออาร์ได้ยกเลิกข้อจำกัดในการถ่ายทอดเทคโนโลยีทางทหารออกจากโครงการนี้ ดังนั้นภาษาวีเอชดีแอลจึงเริ่มเป็นที่รู้จักกันโดยทั่วไป จนกระทั่งทางไออีอีอี (IEEE) ได้รับภาษานี้เข้ามาศึกษาและประมาณปี ค.ศ. 1987 ได้ยอมรับกำหนดมาตรฐานของภาษา โดยให้ชื่อว่า IEEE 1076-1987 และมีชื่อเรียกว่าวีเอชดีแอล มาตรฐานนี้ก็ได้รับการปรับปรุงจนปัจจุบัน ได้ชื่อว่า IEEE 1076-1993 หรือ วีเอชดีแอล 1993 การที่ทางดีโอดีในขณะนั้นเป็นลูกค้ารายใหญ่ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์ จึงมีผู้รับโครงการต่างๆ จากดีโอดี ไปดำเนินการด้านวิจัยและพัฒนามาก เพื่อที่จะให้เป็นมาตรฐานเดียวกันหมด ทางดีโอดี จึงกำหนดว่า ในการส่งโครงการนั้นจะต้องเขียนอยู่ในรูปของภาษาวีเอชดีแอลเท่านั้น ซึ่งทำให้เกิดข้อดีต่อดีโอดีเองที่เป็นมาตรฐานเดียวกัน สามารถนำไปจำลองกับเครื่องคอมพิวเตอร์ได้หลายๆ ระบบ

2.2 ส่วนประกอบต่างๆ ของภาษาวีเอชดีแอล

ในการเขียนรูปแบบบรรยายระบบดิจิตอลในมุมมองของการออกแบบลักษณะจากบนลงล่างจะต้องทำความเข้าใจในเรื่องของโครงสร้างและส่วนประกอบต่างๆ ของรูปแบบภาษาวีเอชดีแอลเสียก่อน ซึ่งส่วนประกอบที่สำคัญและเป็นพื้นฐานของการเขียนมี 4 หน่วยคือ

1. หน่วยการออกแบบเอนติตี (Entity Design Unit)
2. หน่วยการออกแบบสถาปัตยกรรม (Architecture Design Unit)
3. หน่วยการออกแบบแพ็กเกจ (Package Design Unit)
4. หน่วยการออกแบบโครงแบบ (Configuration Design Unit)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.1 การออกแบบเอนติตี (Entity)

หน่วยการออกแบบนี้เป็นส่วนที่ใช้สำหรับติดต่อระหว่างโลกภายนอกกับรูปแบบที่เขียนขึ้น ที่เรียกว่า หน่วยการออกแบบเอนติตี ในส่วนนี้ใช้กำหนดจุดเชื่อมต่อของรูปแบบ กำหนดทิศทางการไหลของสัญญาณ และประเภทของค่าที่สามารถกำหนดให้กับสัญญาณตามจุดต่างๆ ของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้น รูปที่ 2-1 แสดงให้เห็นโครงสร้างอย่างง่ายๆ ของ หน่วยการออกแบบเอนติตี

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

รูปที่ 2-1 แสดงโครงสร้างของหน่วยการออกแบบเอนติตี

การกำหนดพอร์ตชนิดต่างๆ

พอร์ตที่ใช้กันส่วนใหญ่ในภาษา VHDL มีดังนี้

1. *in :* รับข้อมูลไหลเข้าวงจรได้อย่างเดียว
2. *out :* ส่งข้อมูลไหลออกจากวงจรได้อย่างเดียว
3. *inout :* ข้อมูลสามารถไหลเข้าและออกจากพอร์ตนี้ได้

การตั้งชื่อ (Identifier)

ชื่อ (Identifier) ที่จะตั้งในภาษาวีเอชดีแอลจะต้องประกอบด้วยลำดับของตัวอักขระตั้งแต่ 1 ตัวขึ้นไป โดยมีความยาวของชื่อได้ไม่จำกัด ค่าที่เป็นไปได้คือ ตัวอักษรพิมพ์ใหญ่ (A...Z) ตัวอักษรพิมพ์เล็ก (a...z) ตัวเลข (0...9) และเครื่องหมาย underscore (_) โดยมีกฎอยู่ว่าชื่อจะต้องขึ้นต้นด้วยตัวอักษร (A...Z, a...z) และต้องไม่ลงท้ายด้วยเครื่องหมาย underscore โดยในภาษาวีเอชดีแอลนั้นจะถือว่าตัวอักษรพิมพ์เล็กและพิมพ์ใหญ่ไม่มีความแตกต่างกัน (non-case sensitive) ยกตัวอย่างเช่น ADDER, adder, Adder และ AddER จะถือว่าเป็นตัวเดียวกัน ในการใช้เครื่องหมาย underscore นั้นจะต้องไม่ใช้ติดกัน นอกจากนี้แล้ว ในการตั้งชื่อจะต้องไม่ใช้คำสงวน (reserved word) ของภาษาวีเอชดีแอลด้วย

Data Objects

Data Object มีอยู่ 3 ชนิด อันได้แก่

1. Constant ออบเจ็กต์ที่เป็นชนิด constant (ค่าคงที่) จะสามารถเก็บค่าของข้อมูลชนิดที่ระบุให้กับออบเจ็กต์ได้แค่หนึ่งค่าเท่านั้น โดยค่าดังกล่าวจะถูกกำหนดให้กับออบเจ็กต์ก่อนที่จะมีการซิมูเลชันเกิดขึ้น และค่าของมันจะไม่เปลี่ยนแปลงเลยตลอดการซิมูเลชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. Variable ออบเจ็กต์ชนิด variable (ตัวแปร) นี้สามารถเก็บค่าของข้อมูลชนิดที่ระบุให้กับออบเจ็กต์นี้ได้ แต่ออบเจ็กต์ชนิดนี้สามารถรับค่าอื่นเข้าไปเก็บไว้ในภายหลังได้ด้วยการใช้ variable assignment statement (ซึ่งก็คือเครื่องหมาย := นั่นเอง)

3. Signal ออบเจ็กต์ชนิด signal (สัญญาณ) นี้จะเป็นของสัญญาณซึ่งจะมีค่าของเดิมของสัญญาณ, ค่าปัจจุบัน และเซตของค่าในอนาคต โดยที่ค่าในอนาคตของสัญญาณนี้จะถูกกำหนดให้โดยใช้ signal assignment statement (ซึ่งก็คือเครื่องหมาย <= นั่นเอง)

ชนิดของข้อมูล (Data type)

Data type ที่มีอยู่แล้วในภาษาวีเอชดีแอล (predefined type) ได้แก่

1. CHARACTER
2. BIT มีค่าที่เป็นไปได้ คือ '0' และ '1'
3. BIT_VECTOR เป็นอาร์เรย์ของข้อมูลชนิดบิต
4. BOOLEAN มีค่าที่เป็นไปได้คือ true และ false
5. INTEGER มีค่าที่เป็นไปได้คือ ตั้งแต่ $-(2^{31}-1)$ ไปจนถึง $+(2^{31}-1)$
6. REAL ซึ่งมีค่าได้ตั้งแต่ $-(1.0 * 10^{38})$ ไปจนถึง $+(1.0 * 10^{38})$ ค่าที่สามารถรับได้จะมีความละเอียดของเลขทศนิยมไม่เกิน 6 หลัก
7. STRING เป็นอาร์เรย์ของข้อมูลชนิด CHARACTER

2.2.2 การออกแบบสถาปัตยกรรม

คือส่วนที่ใช้เขียนบรรยายพฤติกรรมของรูปแบบ ในมุมมองของการจำลองการทำงาน พฤติกรรมต่างๆ ที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่านเข้าและออก ตรงช่องทางตลอดจนพารามิเตอร์ (Parameter) ต่างๆ ที่กำหนดในหน่วยการออกแบบเอนติตี รูปที่ 2-2 แสดงให้เห็นถึงโครงสร้างอย่างง่ายๆ ของหน่วยการออกแบบสถาปัตยกรรม

```
ARCHITECTURE identifier OF component_name IS
[declaration]
BEGIN
   Specification of the functionality of the
   component in terms of its input lines and as
   influenced by physical and other parameters
END [identifier];
```

รูปที่ 2-2 แสดงโครงสร้างการออกแบบสถาปัตยกรรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปแบบการเขียนของภาษาวีเอชดีแอลเพื่ออธิบายสถาปัตยกรรมของฮาร์ดแวร์มีสามรูปแบบใหญ่ๆ คือ

- **Structural Description Method** ใช้อธิบายตัวรูปแบบ ในรูปแบบของการเชื่อมต่อคอมโพเน็นต์ต่างๆเข้าด้วยกัน
- **Behavior** เป็นการบรรยายการทำงานของฮาร์ดแวร์ ในรูปแบบของ Circuit, Signal ที่ตอบสนองกับสัญญาณที่รับเข้ามาจากภายนอก พฤติกรรมการทำงาน (Hardware Behavior) จะถูกอธิบายด้วยอัลกอริทึม
- **Data Flow Description Method** มีความคล้ายคลึงกับ Register transfer Language วิธีการนี้จะอธิบายฟังก์ชันการทำงานของรูปแบบ โดยการกำหนดการไหล (Flow) ของข้อมูลจากอิตพุต หรือรีจิสเตอร์ ไปยังตัวเอาต์พุต หรือรีจิสเตอร์อีกตัว วิธีการทั้งสามแบบที่ใช้อธิบายสถาปัตยกรรมของฮาร์ดแวร์ สามารถทำงานร่วมกันได้

2.2.3 การออกแบบแพ็กเกจ

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย ที่เป็นประโยชน์ต่อการเขียนรูปแบบบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนของแพ็กเกจได้ และข้อมูลเหล่านี้สามารถเรียกไปใช้ได้โดย หน่วยการออกแบบเอนติตี หน่วยการออกแบบสถาปัตยกรรม หรือ จากหน่วยการออกแบบแพ็กเกจอื่นๆ นอกจากนั้นสิ่งที่นิยมทำกันมากคือรูปแบบมาตรฐานต่างๆ เช่น อุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74XX เป็นต้น) จะถูกเก็บไว้ในแพ็กเกจ ที่ทุกคนสามารถเข้าถึง โดยปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (Package declaration) และ ส่วนของบอดี้แพ็กเกจ (Package body) เนื่องจากแพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษาวีเอชดีแอล สามารถกระทำได้ด้วยชุดคำสั่ง USE

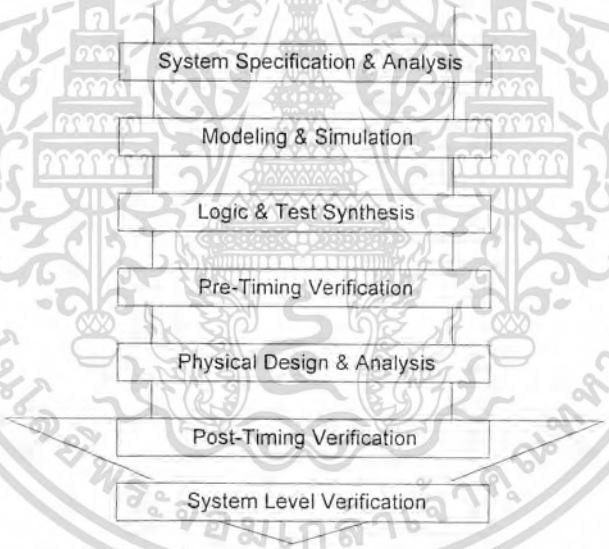
2.2.4 การออกแบบโครงแบบ

ดังที่ทราบกันแล้วว่ารูปแบบหนึ่งของระบบดิจิตอลไม่ว่าจะเป็นอะไร จะมีหน่วยการออกแบบเอนติตีได้เพียงหนึ่งเดียวเท่านั้น แต่ในขณะที่หน่วยการออกแบบเอนติตีหนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมี หน่วยการออกแบบโครงแบบมาเพื่อกำหนดการใช้โครงแบบ (Configuration) ประกอบเอนติตีกับหน่วยการออกแบบสถาปัตยกรรมหน่วยไหนเข้าด้วยกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 การออกแบบจากบนลงล่าง (Top-Down Design)

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน เช่น วงจรรวม (ASIC: Application Specific Integrated Circuit) วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของของ บล็อกไดอะแกรมเสียก่อน ก่อนที่จะวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษาวีเอชดีแอลนั้นอนุญาตให้อธิบายการทำงานของแต่ละบล็อกและวิเคราะห์การทำงาน แก้ไขและปรับปรุงการทำงานจากผลที่วิเคราะห์ เพื่อให้ได้การทำงานตามที่ต้องการ และเพิ่มเติมในรายละเอียดที่ละชั้น นี่คือหลักการออกแบบจากบนลงล่าง (Top-Down Design) ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Botom-Up Design) จะเห็นได้ว่าการออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เพราะเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic Capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบ จำลองการทำงาน ตรวจสอบความถูกต้อง ซึ่งใช้เวลามาก และถ้าวงจรที่ต้องการออกแบบมีความซับซ้อนก็จะเป็นเรื่องที่ยากมากให้การออกแบบในลักษณะนี้ ดังนั้นการใช้ภาษาวีเอชดีแอลกับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรออกแบบที่จะสามารถออกแบบและพัฒนาวงจรที่มีซับซ้อนได้มากขึ้น และช่วยลดเวลาและค่าใช้จ่ายในการออกแบบ



รูปที่ 2-3 ขั้นตอนการออกแบบจากบนลงล่าง

ขั้นตอนการออกแบบจากบนลงล่าง

1. ขั้นตอนการสร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ในการแก้ปัญหา
2. ขั้นตอนการเขียนรูปแบบของระบบที่ต้องการออกแบบโดยใช้ภาษาวีเอชดีแอล หรือ ภาษาเอชดีแอลอื่นๆ สำหรับบรรยายพฤติกรรมการทำงาน พร้อมทั้งจำลองการทำงาน เพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. หลังจากที่ได้หลักการขั้นต้นพร้อมกับแนวความคิดที่ผ่านการตรวจสอบแล้ว หลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมาเป็นลำดับขั้นที่สอง จนกระทั่งอยู่ในระดับที่จะนำไปผลิตวงจร หรือสังเคราะห์ ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับวงจรออกแบบจะถูกกำหนดขึ้น และระบบช่วยการออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้น ให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจรในระดับเกต และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้น หรือไม่ก็อยู่ในรูปของเน็ตลิสท์ (Netlist) ที่สามารถนำไปผลิตลงบนอุปกรณ์อื่นได้
4. หลังจากการสังเคราะห์วงจรให้อยู่ในรูประดับเกตหรือเน็ตลิสท์แล้ว ข้อมูลที่ได้จากผู้ผลิตอุปกรณ์วงจรนั้น นอกจากจะเป็นข้อมูลสำหรับจำลองการทำงาน ในเรื่องของความถูกต้องของฟังก์ชันแล้วยังมีข้อมูลที่เกี่ยวกับเวลาด้วย ซึ่งเป็นความจริงที่ว่า อุปกรณ์ทางอิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการแพร่กระจาย (Propagation delay time) เสมอ ถึงแม้ว่าจะเป็นเวลาที่น้อยมากในระดับ นาโนวินาที (10^{-9} นาที) แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกตของฟังก์ชันต่างๆ จำนวน 10,000 เกต ขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้น จนอาจจะทำให้การทำงานของวงจรรวมทั้งหมดผิดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาที่สูงได้
5. ขั้นตอนของการผลิตเป็นวงจรจริง (Technology and device mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจจะอยู่ในรูปของแผงวงจรไฟฟ้า ที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้น หรืออยู่ในรูปของวงจรรวม (ASIC)
6. หลังจากที่ได้วงจรจริงมาแล้ว ยังต้องมีความจำเป็นที่ต้องตรวจสอบการทำงานที่คำนึงถึงเวลาด้วย เพื่อความถูกต้องของวงจรครั้งสุดท้ายก่อนที่จะนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เพราะในขั้นตอนนี้วงจรที่ออกแบบจะประกอบด้วยอินพุตและเอาต์พุตแพด (Pad) ซึ่งเป็นจุดต่อสำหรับรับและส่งสัญญาณกับภายนอก
7. หลังจากที่นำวงจรที่ออกแบบรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอลแล้วนั้น จะต้องทดสอบการทำงานรวมทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้ง เป็นการควบคุมคุณภาพของผลิตภัณฑ์

2.4 ข้อมูลอุปกรณ์เอฟพีจีเอ (FPGA)

FPGA (Field Programmable Gate Array) เป็นอุปกรณ์ที่ถูกสร้างมาเมื่อประมาณปี ค.ศ. 1986 โดยบริษัท ไซริ้ง (Xilinx) เป็นอาร์เรย์ที่ประกอบด้วยเกตจำนวนมาก สามารถโปรแกรมและลบได้โดยใช้กระแสไฟฟ้า (Static RAM Based) ภายในจัดเรียงเป็นเมทริกซ์ของลอจิกเซลล์ (Logic Cell) ล้อมรอบภายนอกด้วยอินพุต เอาต์พุตเซลล์ ประกอบด้วยเซลล์ที่เรียงตัวเป็นเมทริกซ์ แต่ละเซลล์เรียกว่า ซีแอลบี (CLB Confuguration Logic Block)

ผู้ใช้สามารถกำหนดฟังก์ชันการทำงานได้ตามความต้องการ โดยผ่านการโปรแกรมเอฟพีจีเอได้ ซึ่งช่วยในการออกแบบการผลิต และลดเวลาที่จะส่งตัวผลิตภัณฑ์ออกตลาด โดยที่นักออกแบบเพียงกำหนดฟังก์ชันการทำงานของวงจรเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คุณสมบัติโดยทั่วไปของเอฟพีจีเอ

- เป็นอุปกรณ์ที่มีฟลิปฟลอป (Flip Flop) จำนวนมาก
- มีความยืดหยุ่นสูงในการผลิตฟังก์ชันการทำงาน
- มีบัสภายใน 3 สถานะ
- ทำงานกับสัญญาณ TTL และ CMOS
- มีหน่วยความจำ RAM ภายในที่มีความเร็วสูง
- เส้นทางการเชื่อมต่อ (Interconnect Line) เป็นแบบลำดับชั้น
- มีการกระจายกำลังงานของสัญญาณที่ต่ำ
- มีสถาปัตยกรรมของลอจิกบล็อค (Logic Block) และไอโอบล็อค (I/O Block) ที่สามารถโปรแกรมได้
- ไม่จำกัดจำนวนครั้งในการโปรแกรมซ้ำ
- มีโปรแกรมการวางและเชื่อมโยงอุปกรณ์ภายในแบบอัตโนมัติ (Automatic Place and Routing) ที่ครบสมบูรณ์

Device	System Gates	CLB (1 CLB = 4 slices = Max 128 bits)			Multiplier Blocks	SelectRAM Blocks		DCMs	Max I/O Pads
		Array Row x Col.	Slices	Maximum Distributed RAM Kbits		18-Kbit Blocks	Max RAM (Kbits)		
XC2V40	40K	8 x 8	256	8	4	4	72	4	88
XC2V80	80K	16 x 8	512	16	8	8	144	4	120
XC2V250	250K	24 x 16	1,536	48	24	24	432	8	200
XC2V500	500K	32 x 24	3,072	96	32	32	576	8	264
XC2V1000	1M	40 x 32	5,120	160	40	40	720	8	432
XC2V1500	1.5M	48 x 40	7,680	240	48	48	864	8	528
XC2V2000	2M	56 x 48	10,752	336	56	56	1,008	8	624
XC2V3000	3M	64 x 56	14,336	448	96	96	1,728	12	720
XC2V4000	4M	80 x 72	23,040	720	120	120	2,160	12	912
XC2V6000	6M	96 x 88	33,792	1,056	144	144	2,592	12	1,104
XC2V8000	8M	112 x 104	46,592	1,456	168	168	3,024	12	1,108
XC2V10000	10M	128 x 120	61,440	1,920	192	192	3,456	12	1,108

ตารางที่ 2-1 แสดงข้อมูลชิปเอฟพีจีเอเบอร์ต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
ทฤษฎีอัลกอริทึม

3.1 ดีอีเอส (DES: Data Encryption Standard)

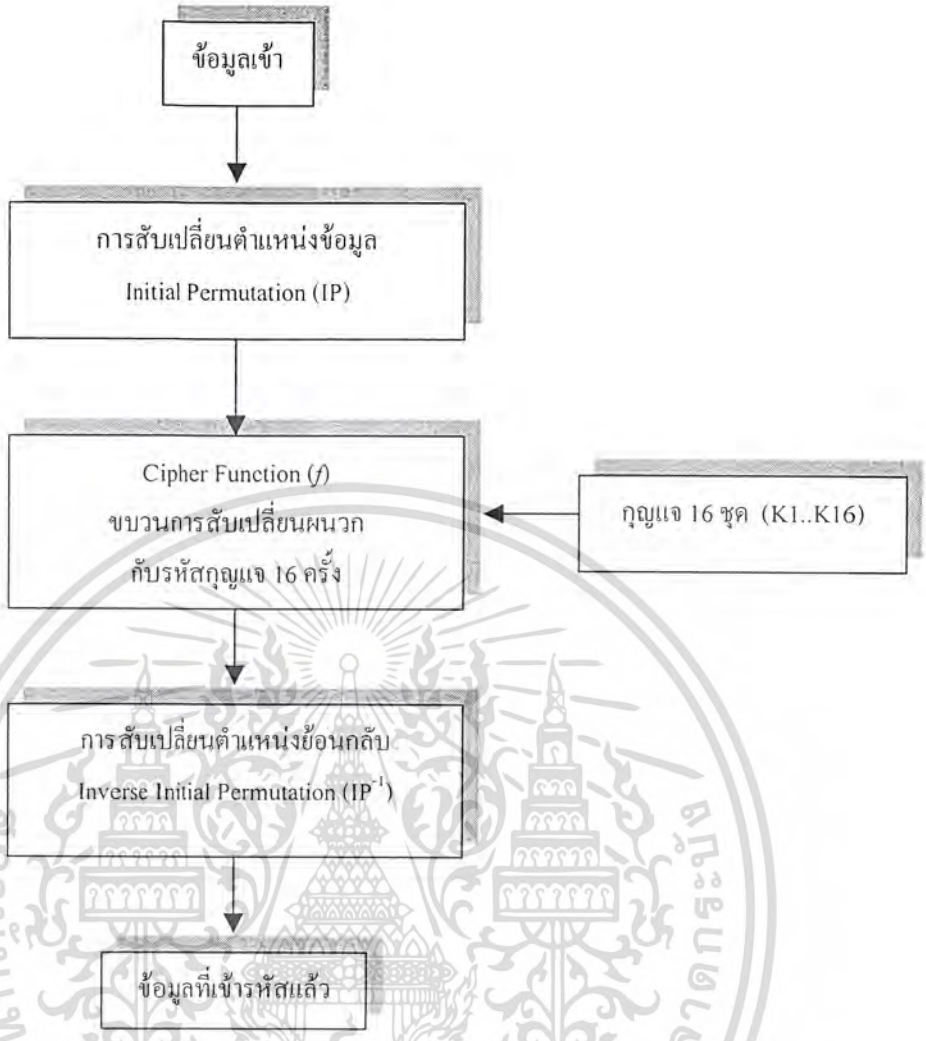
ก่อนที่เราจะศึกษาการทำงานของอัลกอริทึมการทำงานของ Triple Data Encryption Standard (3DES) เราจำเป็นต้องทำการศึกษาการทำงานของอัลกอริทึม DES เสียก่อน เนื่องจากการทำงานของ 3DES นั้นมีพื้นฐานมาจากการอัลกอริทึม DES นั่นเอง

DES (Data Encryption Standard) เป็นการรหัสข้อมูลที่จะเข้ารหัสข้อมูลโดยผ่านอัลกอริทึมทางคณิตศาสตร์ สำหรับกระบวนการเข้ารหัสนั้น ข้อมูลที่เราต้องการจะข้ารหัสเราเรียกว่า เพลนเท็กซ์ (Plain Text) จะถูกนำมาผ่าน อัลกอริทึมทางคณิตศาสตร์โดยใช้คีย์ชุดหนึ่ง ข้อมูลที่ได้จากการเข้ารหัสเราจะเรียกว่า ไซเฟอร์เท็กซ์ (Cipher Text)ซึ่งเป็นข้อมูลที่เราไม่สามารถอ่านเข้าใจได้ และข้อมูลดังกล่าวนี้จะถูกส่งไปยังผู้รับ ฝ่ายผู้รับเมื่อรับข้อมูลที่เป็น ไซเฟอร์เท็กซ์มาแล้วก็จะทำการถอดรหัสโดยใช้คีย์ชุดเดียวกันกับที่ใช้เข้ารหัส จะทำให้ได้ข้อมูลที่เป็นเพลนเท็กซ์ออกมา คีย์ที่ใช้จะประกอบด้วยข้อมูลเลขฐานสองขนาด 64 บิต โดยที่ 56 บิตมาจากการสุ่มตัวเลขส่วนอีก 8 บิต เกิดจากการทำพาริตีบิต เพื่อทำการตรวจสอบความผิดพลาดของข้อมูลนั่นเอง

ผู้ที่จะทำการเข้ารหัสข้อมูลจะมีคีย์เพื่อใช้ในการเข้ารหัส อัลกอริทึมในการเข้ารหัสนั้นจะเป็นมาตราฐานที่เรารู้จักกันดี แต่ความแตกต่างในการเข้ารหัสอยู่ที่คีย์ที่เราสร้างขึ้นมาใช้ในการเข้ารหัสนั่นเอง โดยคีย์แต่ละคีย์นั้นจะทำให้ได้ข้อมูลที่ผ่านการเข้ารหัสหรือไซเฟอร์เท็กซ์ที่แตกต่างกัน ดังนั้นความปลอดภัยในการเข้ารหัสข้อมูลจึงอยู่ที่การจัดการคีย์ที่เหมาะสมนั่นเอง ข้อมูลที่ผ่านการเข้ารหัสจะถูกถอดรหัสกลับมาคืนโดยการใช้คีย์เดียวกันกับที่ใช้เข้ารหัสเท่านั้น สำหรับผู้ที่รับข้อมูลที่ผ่านการเข้ารหัสไป ถึงแม้ว่าจะรู้อัลกอริทึมก็ตามแต่ถ้าหากไม่มีคีย์ที่ถูกต้องก็ไม่สามารถทำการถอดรหัสข้อมูลออกมาได้ และแน่นอนว่าใครก็ตามที่ทราบคีย์และมีอัลกอริทึมก็ย่อมสามารถถอดรหัสข้อมูลออกมาได้อย่างง่ายดาย

อัลกอริทึมนี้ถูกออกแบบเพื่อให้สามารถทำการเข้ารหัสและถอดรหัสข้อมูลหนาด 64 บิตบล็อก ที่อยู่ภายใต้การควบคุมของคีย์ขนาด 64 บิต สำหรับการถอดรหัสจะสามารถทำได้ก็ต่อเมื่อมีคีย์อันเดียวกันกับที่ใช้เข้ารหัส โดยการถอดรหัสจะเป็นการทำย้อนกลับของกระบวนการเข้ารหัสนั่นเอง บล็อกข้อมูลที่เข้ารหัสจะถูกทำการสับเปลี่ยนตำแหน่งก่อน (Initial Permutation IP) จากนั้นจะนำไปผ่านไซเฟอร์ฟังก์ชันจนในขั้นสุดท้ายจึงทำการสับเปลี่ยนตำแหน่งย้อนกลับ (Inverse Initial Permutation IP^{-1}) ดังแสดงในรูปที่ 3-1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-1 บล็อกไดอะแกรมของการเข้ารหัสดีอีเอส

3.1.1 กระบวนการสับเปลี่ยนตำแหน่ง **(Initial Permutation)**

สำหรับขั้นตอนแรกของการเข้ารหัสข้อมูลแบบ DES นั้นเริ่มจากกระบวนการสับเปลี่ยนตำแหน่ง (Initial Permutation IP) ในขั้นตอนนี้นั้นข้อมูลแต่ละบล็อกที่เข้ามาจะทำการสลับตำแหน่งตามตารางที่ 3-1

ในการใช้ตารางนั้น ตำแหน่งของตารางที่ปรากฏจะแทนตำแหน่งที่บิตนั้นๆ อยู่หลังจากที่ทำการสลับตำแหน่งแล้วนั่นเอง โดยที่หมายเลขข้างในคือหมายเลขที่แทนตำแหน่งข้อมูลก่อนการสลับตำแหน่ง เช่น จากตาราง IP บิตที่ 58 ของข้อมูลเมื่อผ่านกระบวนการสลับตำแหน่งบิตดังหล่าวจะย้ายมาอยู่บิตที่1 ในตาราง ส่วนบิตที่ 50 นั้นเมื่อผ่านการสับเปลี่ยนก็จะเลื่อนมาอยู่ในตำแหน่งบิตที่ 2 ของตาราง เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

ตารางที่ 3-1 แสดงค่าการสับเปลี่ยนตำแหน่ง IP

3.1.2 การคำนวณการเข้ารหัส

ในการคำนวณการเข้ารหัสข้อมูลมีขั้นตอนดังนี้

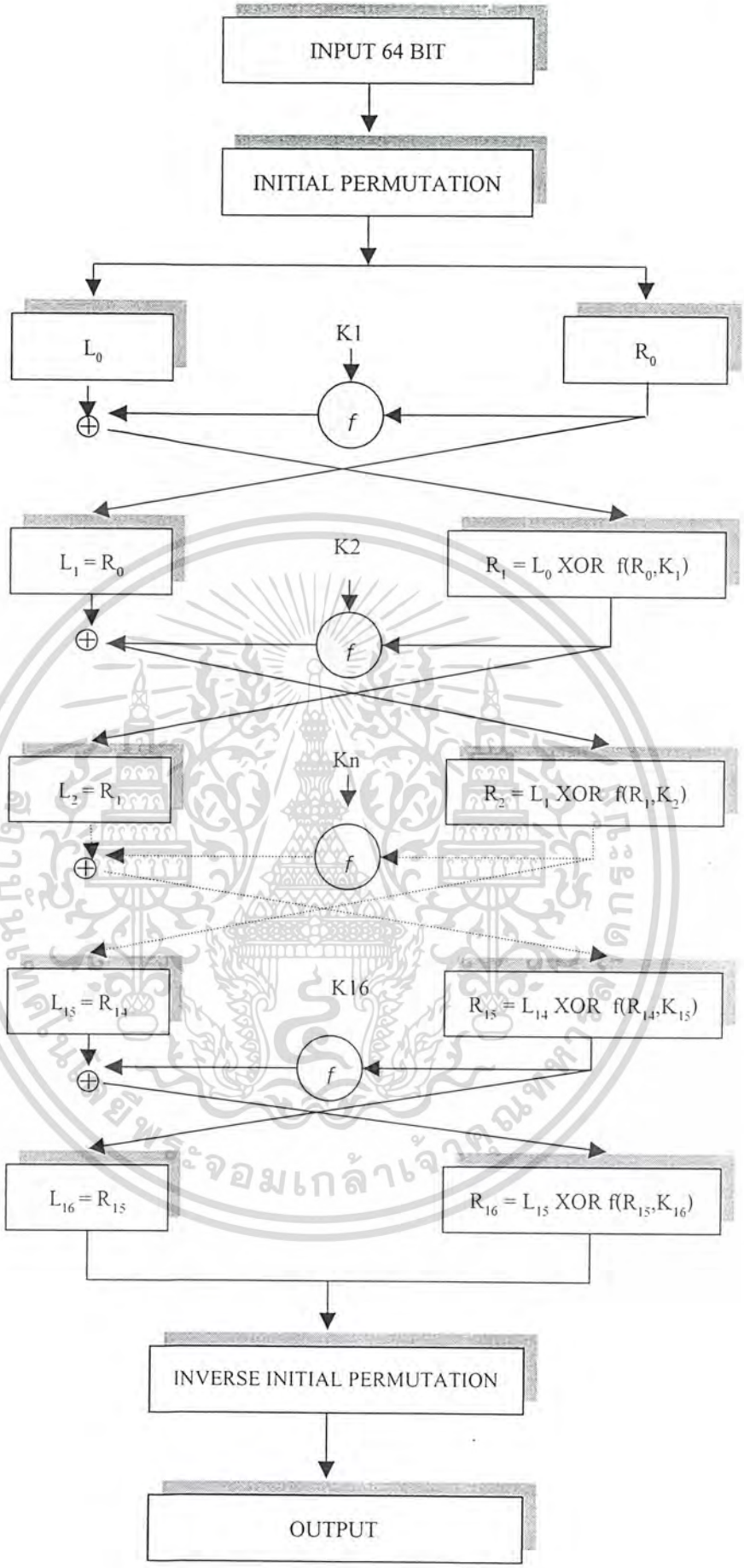
1. ข้อมูลที่ผ่านการสับเปลี่ยนตำแหน่งแล้วจะมีขนาดเท่าเดิมคือมีขนาด 64 บิต ซึ่งจะถูกนำมาเป็นอินพุตในการคำนวณการเข้ารหัส
2. ข้อมูล 64 บิต ดังกล่าวจะถูกแบ่งออกเป็นสองส่วนคือ บล็อกซ้าย L และบล็อกขวา R โดยแต่ละบล็อกจะมีขนาดเท่ากันคือ 32 บิต บล็อกเริ่มต้นนั้นเราจะทำการอินนิเชียลค่าที่ L_0 และ R_0
3. การทำงานบนบล็อก L และ R จะมีทั้งสิ้น 16 รอบการทำงาน โดยแต่ละรอบการทำงานเราจะแทนด้วยค่า n ซึ่ง n จะมีค่าตั้งแต่ 1 ถึง 16 โดยแต่ละบล็อกจะมีค่าการคำนวณดังนี้

$$L_n = R_{n-1}$$

$$R_n = L_n \text{ XOR } f(R_{n-1}, K_n)$$

โดยบล็อก L ใดๆ ที่เกิดขึ้นใหม่นั้นจะมาจากค่าของบล็อก R ก่อนหน้านี้ และบล็อก R ที่เกิดขึ้นใหม่ในแต่ละรอบจะเกิดจากการ exclusive or ระหว่าง L บล็อกในรอบนั้นกับผลลัพธ์ของไซเฟอร์ฟังก์ชันที่มีอินพุตเป็นบล็อก R ในรอบก่อนหน้านี้และค่าของคีย์ย่อยในรอบนั้นๆ ดังแสดงในรูปที่ 3-2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-2 แผนภาพการคำนวณการเข้ารหัสดีอีเอส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.3 การสับเปลี่ยนตำแหน่งย้อนกลับ (Inverse Initial Permutation)

กระบวนการสับเปลี่ยนตำแหน่งย้อนกลับ (Inverse Initial Permutation IP^{-1}) มีลักษณะเช่นเดียวกันกับกระบวนการสับเปลี่ยนตำแหน่งในตอนต้น เพียงแต่ค่าในตารางมีค่าแตกต่างกันเท่านั้นเอง

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

ตารางที่ 3-2 แสดงค่าการสับเปลี่ยนตำแหน่งย้อนกลับ IP^{-1}

3.1.4 การทำงานของไซเฟอร์ฟังก์ชัน และเอสบ็อกซ์ (Cipher function and S-Box)

การทำงานของไซเฟอร์ฟังก์ชัน (Cipher Function) มีการทำงานดังนี้ เริ่มจากการกำหนดฟังก์ชัน E เป็นฟังก์ชันที่มีการรับอินพุตขนาด 32 บิต แล้วสามารถสร้างเอาต์พุตออกมาได้ขนาด 48 บิต โดยค่าอินพุตที่รับเข้ามานั้นคือ บล็อกส่วน R ขนาด 32 บิตนั่นเอง ฟังก์ชัน E มีลักษณะเช่นเดียวกันกับการสลับเปลี่ยนตำแหน่ง IP เพียงแต่ค่าในตำแหน่งจะมีการใช้มากกว่าหนึ่งครั้ง ดังแสดงในตารางที่ 3-3

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

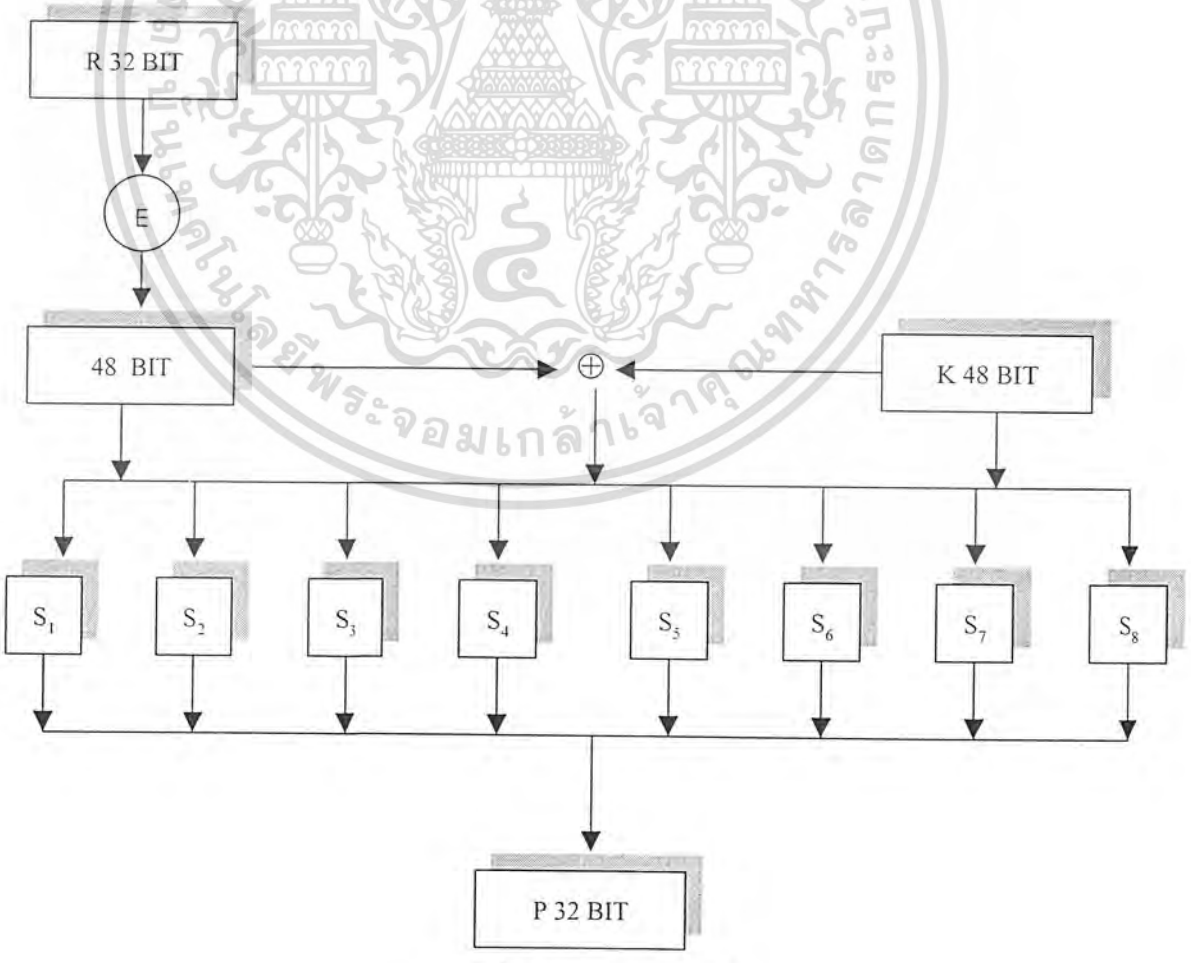
ตารางที่ 3-3 แสดงค่าฟังก์ชัน E

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลลัพธ์ที่ได้จากการผ่านฟังก์ชัน E จะทำให้ได้ข้อมูลขนาด 48 บิต โดยข้อมูลขนาด 48 บิตนี้จะนำมา XOR กับ K ซึ่งมีขนาด 48 บิต ข้อมูลที่ได้จะถูกแบ่งออกเป็น 8 ส่วนๆละ 6 บิต แล้วส่งไปให้ S_box เพื่อทำการหาผลลัพธ์ออกมาเพียงแค่ 4 บิต รวมกันได้ 32 บิต ดังแสดงในรูปที่ 3 จากนั้นข้อมูลขนาด 32 บิตนี้จะนำไปทำการสับเปลี่ยนตำแหน่งในตาราง P ตารางที่ 3-4

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

ตารางที่ 3-4 แสดงค่าสลับตำแหน่ง P



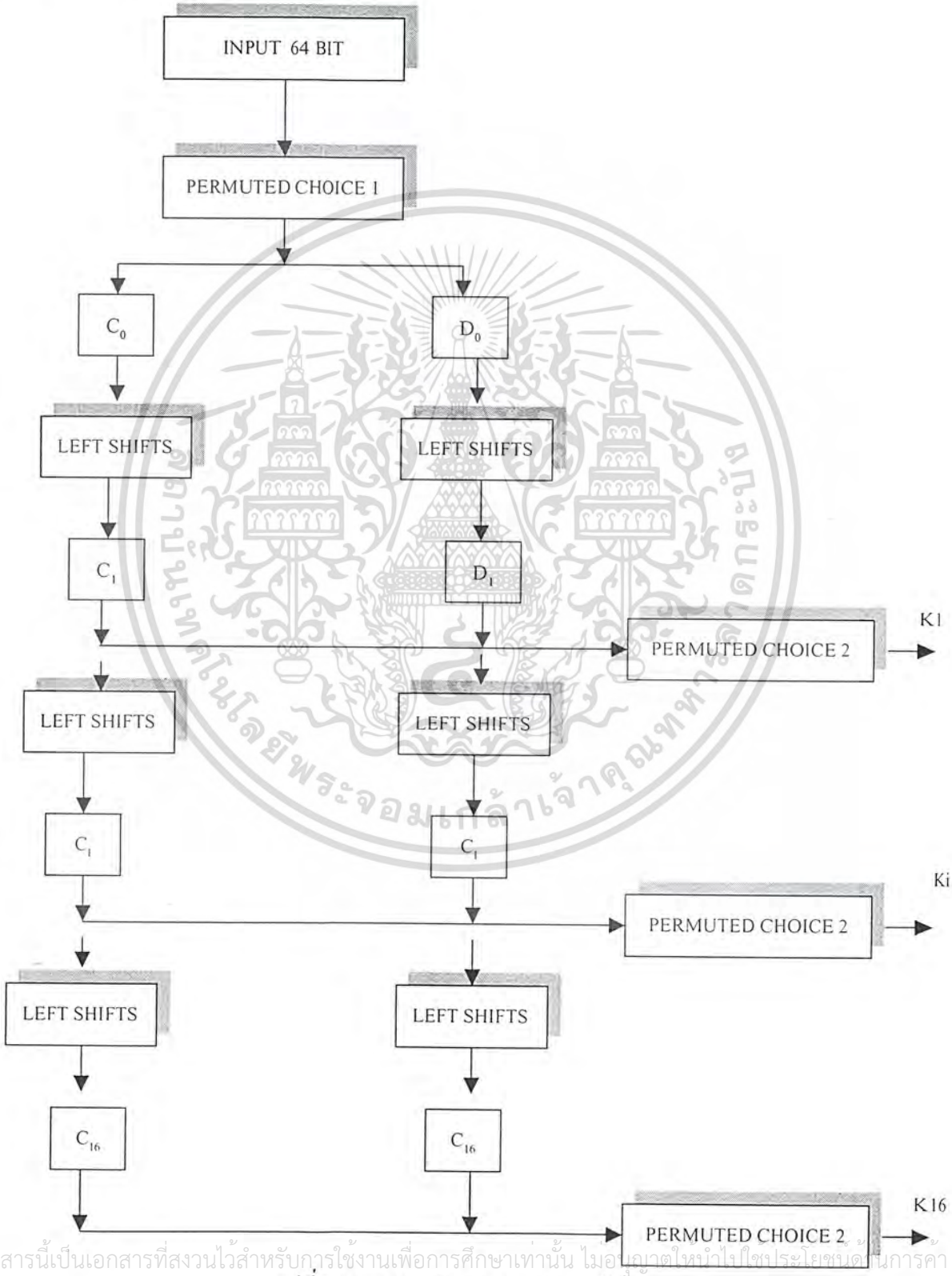
รูปที่ 3-3 การเข้าเอสบอกซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

3.1.5 วิธีการเข้าเอสบ็อกซ์

ให้ Bi เป็นข้อมูลที่ผ่านเข้ามที่ Si หลังจากนำ E มา XOR กับ K แล้ว โดยมีขนาด 6 บิต ($b_1b_2b_3b_4b_5b_6$) ให้ j เป็นเลขจำนวนเต็มที่มีค่า 1 ถึง 8 เราจะแบ่ง Bj ออกเป็นสองส่วนเรียกว่า Row และ Column โดยที่ค่าของ Row = b_0b_6 และส่วนของ Column คือ $b_2b_3b_4b_5$ นำค่า Row และ Column มาเทียบในตาราง S จะทำให้ได้ค่าผลลัพธ์ 4 บิตออกมา



รูปที่ 3-4 แสดงการคำนวณหาหมายเลขคีย์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

42840

3.1.6 การคำนวณหมายเลขกุญแจ

ในแต่ละรอบของกระบวนการเข้ารหัสข้อมูลกับคีย์จำนวน 16 คีย์นั้นจะใช้ค่าของคีย์แต่ละชุดที่มีขนาด 48 บิต โดยที่ค่าของคีย์ทั้ง 16 ชุดได้มาจากคีย์ คีย์ซึ่งเป็นข้อมูลขนาด 64 บิต โดยบิตตำแหน่งที่ 8;16,..,64 เป็นบิตที่ใช้ในการตรวจสอบความผิดพลาด ในการทำกระบวนการสับเปลี่ยน Permuted Choice PC_1 จะทำการตัดในส่วนของบิตตรวจสอบความผิดพลาดออกโดยมีการ ใช้ข้อมูลแค่ 56 บิตที่เหลือเท่านั้น

ผลลัพธ์จากการทำ PC_I จะถูกแบ่งออกเป็นสองส่วน ส่วนละ 24 บิต เรียกว่า C และ D ใช้ในการหาค่าหมายเลขกุญแจ Ki โดยกำหนดให้ Ci และ Di ซึ่งได้มาจาก C และ D ใช้หาค่า Ki มีสมการดังนี้

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

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

LS_i คือการเลื่อนบิตไปทางซ้ายเท่ากับจำนวนครั้งที่กำหนดในตาราง LS โดย C_0 และ D_0 เป็นค่าเริ่มต้นและสมการหมายเลขกุญแจ Ki คือ

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

ส่วนการสับเปลี่ยนตำแหน่ง PC_2 นั้นแสดงดังตาราง

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

ตารางที่ 3-5 ตารางแสดง PC_1

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

ตารางที่ 3-6 ตารางแสดง PC_2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

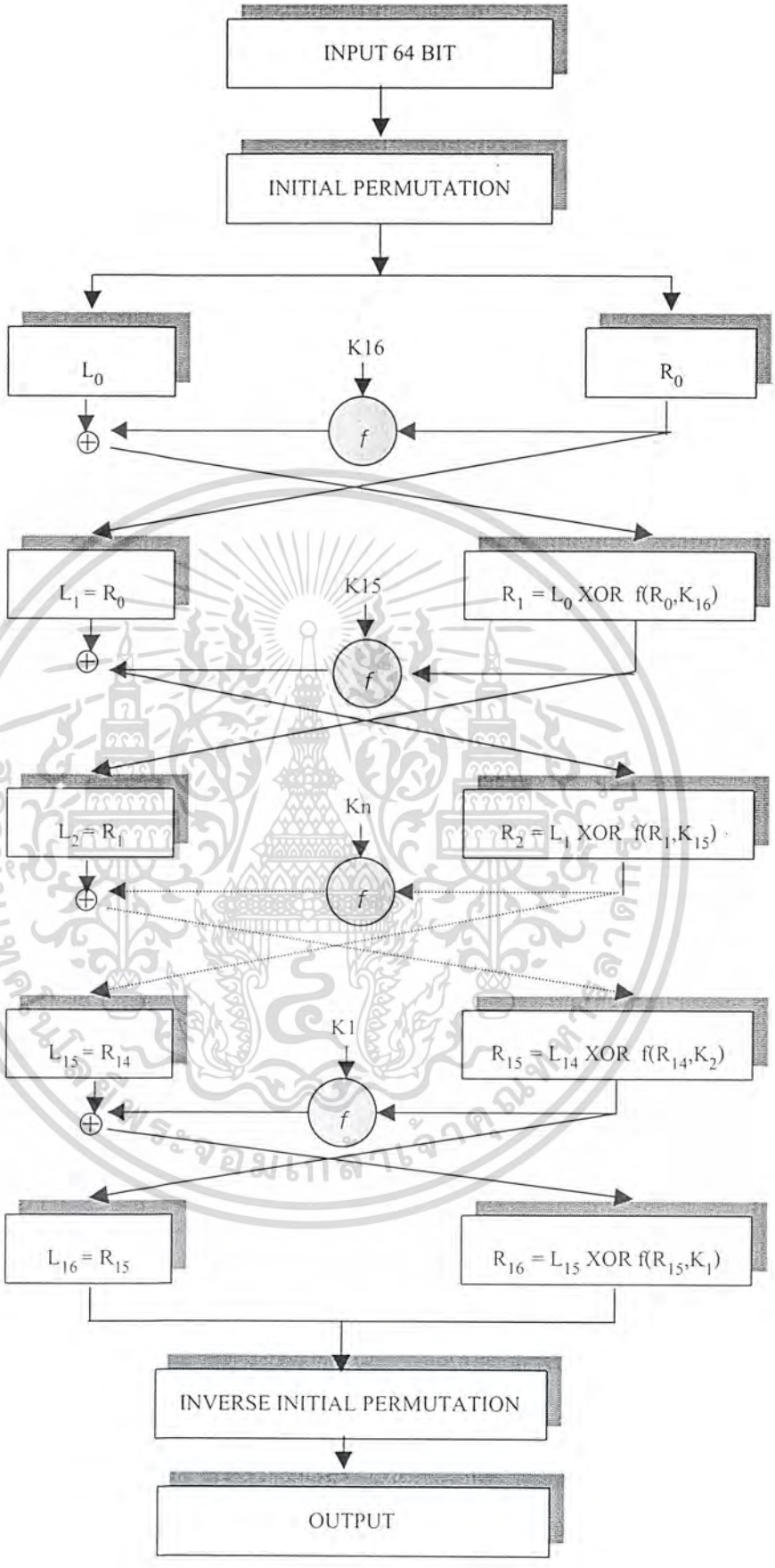
ลำดับ	จำนวนครั้ง	ลำดับ	จำนวนครั้ง
1	1	9	1
2	1	10	2
3	2	11	2
4	2	12	2
5	2	13	2
6	2	14	2
7	2	15	2
8	2	16	1

ตารางที่ 3-7 แสดงจำนวนครั้งที่ทำการชิฟท์เลื่อนในแต่ละรอบ

3.1.7 กระบวนการถอดรหัส

สำกรับกระบวนการถอดรหัสจะมีลักษณะคล้ายกับขั้นตอนการเข้ารหัส เพียงแต่คีย์ที่ใช้ในแต่ละรอบจะสลับกัน ดังแสดงในรูปที่ 3-5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-5 แสดงกระบวนการถอดรหัสดีอีเอส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 Triple Data Encryption Standard (3DES)

ให้ $E_K(I)$ และ $D_K(I)$ แทนการเข้ารหัสและถอดรหัสข้อมูลด้วย DES ตามลำดับโดยใช้คีย์ K ของ DES ตามลำดับ กระบวนการเข้าและถอดรหัสของ 3DES ประกอบขึ้นจากกระบวนการเข้าและถอดรหัสของ DES หลายขั้นตอน (ดังที่ระบุไว้ใน ANSI X9.52) ดังต่อไปนี้

1. การเข้ารหัสของ 3DES โดยแปลงบล็อก P (Plain Text) ขนาด 64 บิตไปเป็นบล็อก C (Cipher text) ขนาด 64 บิตนิยามดังต่อไปนี้

$$C = E_{K3}(D_{K2}(E_{K1}(P)))$$

2. การถอดรหัสของ 3DES โดยแปลงบล็อก C (Cipher text) ขนาด 64 บิตไปเป็นบล็อก P (Plain text) ขนาด 64 บิตนิยามดังต่อไปนี้

$$P = D_{K1}(E_{K2}(D_{K3}(C)))$$

มาตรฐานระบุตัวเลือกการปรับแต่งสำหรับกลุ่มคีย์ (K_1, K_2, K_3) ดังต่อไปนี้

1. การปรับแต่งแบบที่ 1: K_1, K_2 และ K_3 เป็นคีย์อิสระ (Independent key)
2. การปรับแต่งแบบที่ 2: K_1 และ K_2 เป็นคีย์อิสระและ $K_3 = K_1$;
3. การปรับแต่งแบบที่ 3: $K_1 = K_2 = K_3$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 Blowfish

บรูซ ไชน์เออร์ (Bruce Schneier) เป็นผู้คิดค้น Blowfish ความยาวคีย์ของ Blowfish ปรับขนาดได้สูงสุดถึง 448 บิต ดังแสดงรูปที่ 3-6

คำอธิบายอัลกอริทึม Blowfish

Blowfish เป็นบล็อกไซเฟอร์ขนาด 64 บิต ที่มีความยาวคีย์ปรับขนาดได้ อัลกอริทึมประกอบด้วยการสลับตำแหน่งที่ขึ้นอยู่กับคีย์ (Key-dependent permutation) และการเข้ารหัสลับข้อมูล (Data encryption) คีย์จะถูกขยายโดยการแปลงคีย์จากความยาวเดิมที่มีได้ถึง 448 บิตไปเป็นอาร์เรย์ของคีย์ย่อย (Subkey) หลาย ๆ คีย์ที่มีขนาดทั้งหมด 4168 ไบต์

ขั้นตอนการเข้ารหัสลับข้อมูลประกอบด้วยการทำงานของฟังก์ชันทั้งหมด 16 รอบ แต่ละรอบประกอบด้วยการสลับตำแหน่งที่ขึ้นอยู่กับคีย์ควบคู่กับคีย์หนึ่งและการแทนที่แบบขึ้นอยู่กับข้อมูล (Data-dependent permutation) การทำงานทั้งหมดคือการบวกและ XOR บนเวิร์ดข้อมูลขนาด 32 บิต มี indexed array lookup ในแต่ละรอบเหมือนกับ DES

Blowfish ใช้คีย์ย่อยจำนวนมากที่ถูกคำนวณไว้ล่วงหน้า จากรูปที่ 3-6 จะเห็นความสัมพันธ์ P อาร์เรย์กับกระบวนการเข้ารหัส ซึ่งประกอบด้วยคีย์ย่อย 32 บิตทั้งหมด 18 คีย์ย่อย

$$P_1, P_2, \ldots, P_{18}$$

S_box 32 บิต 4 ตัว แต่ละตัวมี 256 รายการ

$$S_{(1,0)}, S_{(1,1)}, \ldots, S_{(1,256)}$$
$$S_{(2,0)}, S_{(2,1)}, \ldots, S_{(2,256)}$$
$$S_{(3,0)}, S_{(3,1)}, \ldots, S_{(3,256)}$$
$$S_{(4,0)}, S_{(4,1)}, \ldots, S_{(4,256)}$$

3.3.1 การเข้ารหัส

จริง ๆ แล้ว Blowfish เป็นเครือข่ายทางคณิตศาสตร์ชนิดพิเศษชนิดหนึ่ง เรียกว่า Feistal network มีการทำงานทั้งหมด 16 รอบ อินพุตเป็นสมาชิกข้อมูล x ขนาด 64 บิต วิธีการเข้ารหัสมีดังต่อไปนี้

1. แบ่ง x ออกเป็นครึ่งละ 32 บิตเป็น x_L, x_R
2. For $i = 1$ to 16:

$$x_L = x_L \oplus P_i$$

3. สลับค่า x_L และ x_R
4. ทำซ้ำขั้นตอนที่ 2 สำหรับรอบต่อไปจนกว่าจะครบ 16 รอบ ($i = 16$)
5. รวมทั้ง 2 ครึ่งเข้าด้วยกัน
6. ฟังก์ชัน F มีสูตรดังต่อไปนี้ กำหนดให้แบ่ง x_L ออกเป็น 4 ส่วนคือ a, b, c, d

$$F(x_L) = ((S_{(1,a)} + S_{(2,b)} \bmod 2^{32}) \oplus S_{(3,c)}) + S_{(4,d)} \bmod 2^{32}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.2 การถอดรหัส

การถอดรหัสเหมือนกับการเข้ารหัสทุกประการยกเว้น P1, P2, ..., P18 ที่เรียงลำดับย้อนกลับ คีย์ย่อยถูกคำนวณโดยใช้อัลกอริทึม Blowfish วิธีการก็คือ

1. ตั้งค่าเริ่มต้นให้กับอาร์เรย์ P และต่อด้วย S_box ทั้ง 4 ตามลำดับด้วยสตริงตายตัว (fixed string) สตริงนี้ประกอบด้วยเลขฐานสิบหกของค่า π (อย่างน้อยเริ่มต้นที่ 3) ตัวอย่างเช่น

   P1 = 0x243f6a88

   P2 = 0x85a308d3

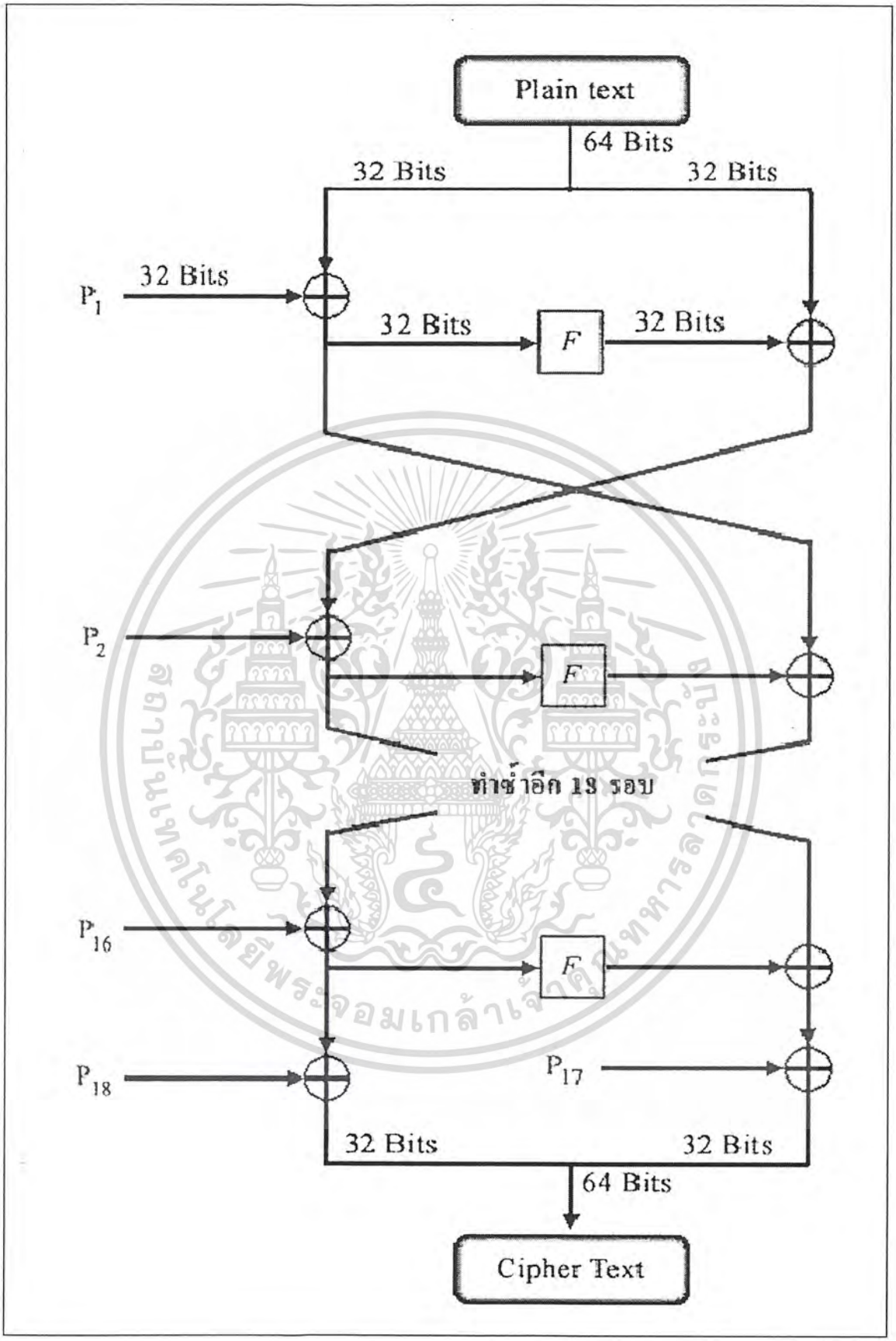
   P3 = 0x13198a2e

   P4 = 0x03707344

2. XOR P1 ด้วย 32 บิตแรกของคีย์ต่อด้วย XOR P2 ด้วย 32 บิตส่วนที่สองของคีย์และทำต่อไปเรื่อย ๆ จนครบทุกๆ บิตของคีย์ (ได้สูงสุดถึง P14) ทำวนซ้ำเช่นเดิมจนกระทั่งทั้งอาร์เรย์ P ถูก XOR กับบิตของคีย์ (สำหรับคีย์สั้นทุกคีย์ จะมีคีย์สมมูลอย่างน้อย 1 คีย์ที่ยาวกว่า เช่น ถ้า A เป็นคีย์ 64 บิตแล้ว AA, AAA, ... เป็นคีย์สมมูลของคีย์ A)
3. เข้ารหัสสตริงทั้งหมดด้วยอัลกอริทึม Blowfish โดยใช้คีย์ย่อยที่ได้อธิบายไปแล้วในขั้นตอนที่ (1) และ (2)
4. แทนที่ P1 และ P2 ด้วยเอาต์พุตที่ได้จากขั้นตอนที่ (3)
5. เข้ารหัสเอาต์พุตของขั้นตอนที่ 3 โดยใช้อัลกอริทึม Blowfish ด้วยคีย์ย่อยที่แก้ไขแล้ว
6. แทนที่ P3 และ P4 ด้วยเอาต์พุตของขั้นตอนที่ (5)
7. แทนที่ให้ครบทั้งอาร์เรย์ P แล้วตามด้วย S_box ทั้ง 4 ตามลำดับ ด้วยเอาต์พุตของอัลกอริทึม Blowfish ที่ทำการเปลี่ยนแปลงอย่างต่อเนื่อง

ในการทำงานทั้งหมด การคำนวณคีย์ย่อยที่ต้องการทั้งหมดต้องทำถึง 521 รอบ จึงอาจมีหลายแอพพลิเคชั่นที่เก็บคีย์ย่อยนี้ไว้มากกว่าการที่ต้องทำงานใน กระบวนการนี้ใหม่ทุกครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-6 แสดงขั้นตอนการทำงานของอัลกอริทึม Blowfish

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 Message Digest Algorithm (MD5)

MD5 เป็นอัลกอริทึมที่รับข้อความความยาวไม่จำกัดเข้ามาเป็นอินพุตแล้วสร้างเอาต์พุตเป็น "เมสเสจไดเจสต์" ขนาด 128 บิต โดยอาศัยฟังก์ชันทางคณิตศาสตร์ซึ่งมีคุณสมบัติเป็นฟังก์ชันทางเดียว นั่นก็หมายความว่าเป็นไปได้ยากมากที่จะมีข้อความ 2 ข้อความใดๆ ที่สร้างเมสเสจไดเจสต์ได้เหมือนกัน และไม่สามารถสร้างข้อความใดๆ จากเมสเสจไดเจสต์นี้ได้เลย อัลกอริทึม MD5 นี้มีจุดมุ่งหมายสำหรับแอพพลิเคชั่นที่ใช้ในการสร้างลายมือชื่อดิจิตอล

คำศัพท์และสัญลักษณ์ที่ใช้ในการอธิบาย

"ไบต์" ข้อมูลขนาด 8 บิต $X <<< s$ แทนการหมุนบิต (circular shift) ของ X ไปทางซ้ายจำนวน s บิต

"เวิร์ด" ข้อมูลขนาด 32 บิต "+" แทนการบวกแบบมอดูลโล (Modulo) ของเวิร์ด (เช่น การบวกแบบมอดูลโลกับ 2^{32})

คำอธิบายอัลกอริทึม MD5

เราเริ่มโดยการสมมุติว่าเรามีข้อความขนาด b บิตเป็นอินพุต และเราต้องการหาเมสเสจไดเจสต์ของข้อความนี้ ในที่นี้ b เป็นเลขจำนวนเต็มที่ไม่ติดลบซึ่งมีค่าไม่จำกัด โดยไม่จำเป็นต้องหาร 8 ลงตัว เราจะแทนบิตของข้อความดังต่อไปนี้

$$m_0 m_1 \dots m_{(b-1)}$$

3.4.1 ขั้นตอนการคำนวณหาเมสเสจไดเจสต์

การเติมเต็มบิตต่อท้าย

ข้อความจะถูกต่อเติมเพื่อให้ความยาวของมันเท่ากับ 448 บิตจาก 512 บิต นั่นคือข้อความถูกแบ่งเป็นบล็อก ๆ ละ 512 บิตและบล็อกสุดท้ายจะมีการเติมบิตให้ครบ 448 บิตเพื่อเหลือพื้นที่ไว้ 64 บิตในการเติมความยาวบิตของข้อความในขั้นตอนต่อไป

การเติมบิตถูกกระทำโดยต่อท้ายบิตของข้อความบล็อกสุดท้ายด้วยบิต "1" แล้วตามด้วยบิต "0" จนกว่าความยาวบิตของบล็อกข้อความนี้จะเท่ากับ 448 บิต

การเติมความยาวบิตของข้อความ

จากที่กล่าวไว้ข้างต้นว่าให้ b แทนความยาวบิตของข้อความ ดังนั้นที่เหลืออีก 64 บิตจากขั้นตอนที่แล้วมีไว้สำหรับเก็บค่าความยาวของข้อความต้นฉบับ เนื่องจากไม่น่าจะมีข้อความใดเกิน 2^{64} บิต ดังนั้น 64 บิตก็เพียงพอในการเก็บค่าความยาวบิตของข้อความ โดยเติมเวิร์ด 32 บิตที่มีลำดับต่ำก่อนแล้วตามด้วยเวิร์ดที่มีลำดับสูง

ตรงจุดนี้เราจะได้ข้อความที่มีขนาดเป็นหลายเท่าของ 512 บิตแล้ว หรือกล่าวได้ว่าข้อความมีความยาวเป็นหลายเท่าของ 16 เวิร์ด (32 บิต) ให้ M[0 ... N-1) แทนเวิร์ดของข้อความที่ผ่านกระบวนการต่อเติมแล้ว โดยที่ N คือจำนวนเท่าของ 16

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กำหนดค่าเริ่มต้นของบัฟเฟอร์ของเมสเสจไดเจสต์

เราจะใช้บัฟเฟอร์ 4 เวิร์ด (A B C D) ในการคำนวณหาเมสเสจไดเจสต์ ในที่นี้ A B C D เป็นรีจิสเตอร์ขนาด 32 บิตซึ่งมีค่าเริ่มต้นในรูปของเลขฐานสิบหกดังนี้ เรียงลำดับให้ไบต์ที่มีลำดับต่ำสุดก่อน

เวิร์ด A: 01 23 45 67

เวิร์ด B: 89 ab cd ef

เวิร์ด C: fe dc ba 98

เวิร์ด D: 76 54 32 10

ประมวลข้อความเป็นบล็อกขนาด 16 บิต

เราเริ่มด้วยการนิยามฟังก์ชันช่วยที่รับอินพุตเวิร์ด 32 บิตจำนวน 3 เวิร์ดเข้ามาแล้วสร้างเอาต์พุตเป็นเวิร์ดขนาด 32 บิต

F(X,Y,Z) = **(X AND Y) OR (NOT(X) AND Z)**

G(X,Y,Z) = **(X AND Z) OR (Y AND NOT(Z))**

H(X,Y,Z) = **X XOR Y XOR Z**

I(X,Y,Z) = **Y XOR (X OR NOT(Z))**

ในขั้นตอนนี้ใช้ตารางที่มีสมาชิก 64 ตัว T[1 ... 64] ที่ถูกสร้างมาจากฟังก์ชันไซน์ (sine function) ให้ T[i] แทนสมาชิกลำดับที่ i ของตาราง ซึ่งสามารถดูค่าได้ที่ภาคผนวก ข.

กระบวนการหาเมสเสจไดเจสต์

```
/* การประมวลของแต่ละบล็อก 16 เวิร์ด */
FOR i = 0 TO ( N/16 ) - 1 DO
    /* Copy บล็อก i ลงใน X */
    FOR j = 0 TO 15 DO
        X[ j ] = M[ i*16+j ];
    END /* ของ loop j */

    /* บันทึกค่า A เก็บไว้ชั่วคราวที่ AA, B ไว้ที่ BB, C ไว้ที่ CC, D ไว้ที่ DD */
    AA = A;          BB = B;          CC = C;          DD = D;

    /* รอบที่ 1 * ให้ [abcd k s i] แทนโอเปอเรชั่นต่อไปนี้
        a = b + (( a + F(b,c,d) + X[k] + T[i]) <<< s)  ทำ 16 โอเปอเรชั่น
    /* รอบที่ 2 * ให้ [abcd k s i] แทนโอเปอเรชั่นต่อไปนี้
            a = b + (( a + G(b,c,d) + X[k] + T[i]) <<< s)  ทำ 16 โอเปอเรชั่น
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

/* **รอบที่ 3** * ให้ [abcd k s i] แทนโอเปอเรชั่นต่อไปนี้

a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s) ทำ 16 โอเปอเรชั่น

/* **รอบที่ 4** * ให้ [abcd k s i] แทนโอเปอเรชั่นต่อไปนี้

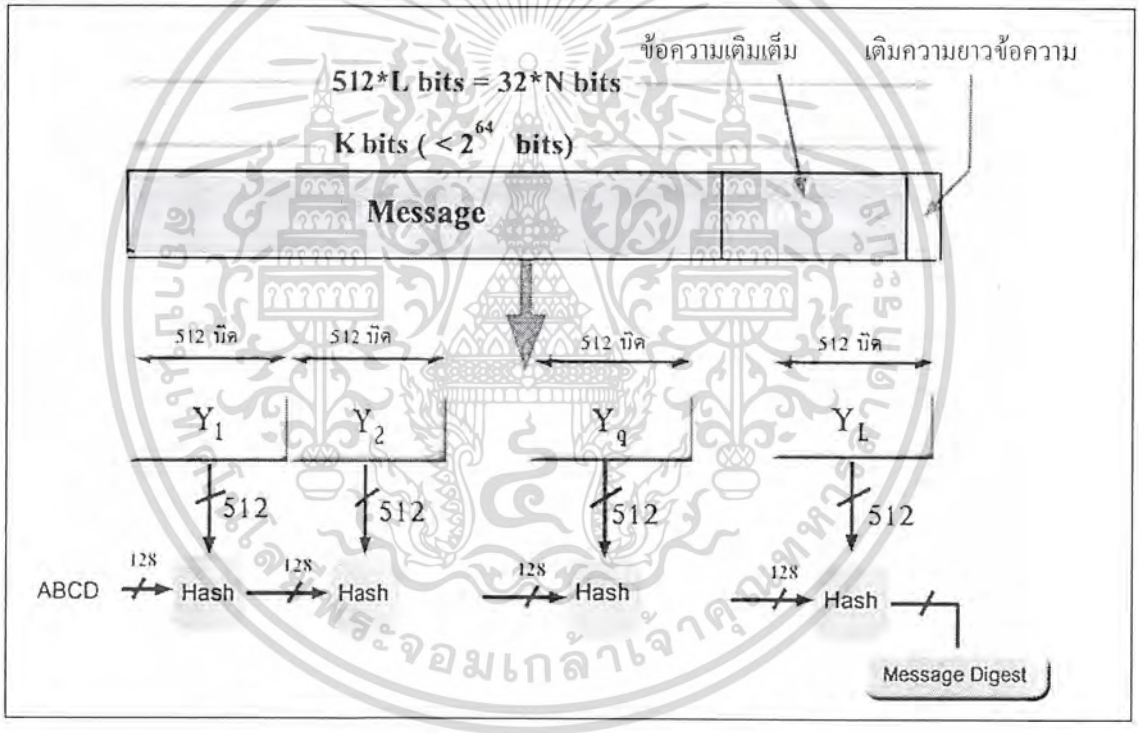
a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s) ทำ 16 โอเปอเรชั่น

/* แล้วจึงทำการบวกค่าดังต่อไปนี้ (นั่นก็คือ เพิ่มค่าในรีจิสเตอร์แต่ละตัวจากค่าที่มีอยู่ก่อนบล็อกนี้จะถูกประมวล) */

A = A + AA; B = B + BB; C = C + CC; D = D + DD;

END /* ของ loop i */

เมสเสจไดเจสต์ที่ถูกสร้างเป็นเอาต์พุตก็คือ A B C D โดยเรียงลำดับเริ่มต้นที่ลำดับไบต์ต่ำของ A (low-order byte) และสิ้นสุดที่ลำดับไบต์สูงของ D (high-order byte) ขั้นตอนการทำงานทั้งหมดแสดงดังรูป 3-7



รูปที่ 3-7 แสดงการทำงานของ MD5

3.4.2 สรุปอัลกอริทึม MD5

อัลกอริทึมเมสเสจไดเจสต์ MD5 สามารถอิมพลีเมนท์ได้ง่าย และสนับสนุนการสร้างเมสเสจไดเจส์จากข้อความที่มีความยาวไม่จำกัดได้ อีกทั้งเป็นเรื่องยากมากที่จะหาข้อความที่ต่างกัน 2 ข้อความใด ๆ ที่มีเมสเสจไดเจสต์เหมือนกัน และแทบจะเป็นไปไม่ได้เลยที่จะสามารถสร้างข้อความใด ๆ จากเมสเสจไดเจสต์ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 Secure Hash Algorithm (SHA)

เอสเอชเอ (SHA) ถูกออกแบบมาเพื่อใช้งานร่วมกับดีเอสเอ (DSA: Digital Signature Algorithm) เพื่อสร้างลายมือชื่อดิจิตอล ซึ่งเป็นมาตรฐานที่มีการนำเสนอแก่สารธารนชนแล้ว โดยเป็นเพียงอัลกอริทึมในการทำแฮชชิ่ง (Hashing)ที่ได้รับการยอมรับเพื่อนำมาทำงานร่วมกับดีเอสเอ

ภาพรวมเบื้องต้น

อัลกอริทึมเอสเอชเอเป็นอัลกอริทึมที่จำเป็นสำหรับการทำให้มั่นใจในความปลอดภัยของอัลกอริทึม ดีเอสเอ อินพุตเป็นข้อความขนาดใดก็ได้ไม่เกิน 2^64 บิต เอาต์พุตที่ได้มีขนาด 160 บิต เรียกว่า Message Digest จากนั้นเมสเสจไดเจสต์จะถูกส่งไปเป็นอินพุตของดีเอสเอ ซึ่งจะทำการสร้างลายมือชื่อดิจิตอลสำหรับ ข้อความนั้นออกมา (Digital signature) การเซ็นรับรองเมสเสจไดเจสต์แทนการเซ็นข้อความธรรมดาช่วยให้ประสิทธิภาพ (Efficiency) ของโพรเซสดีขึ้น เพราะว่าปกติเมสเสจไดเจสต์จะมีขนาดน้อยกว่าข้อความธรรมดามาก การเปลี่ยนแปลงข้อความระหว่างการส่งถ่ายจะมีโอกาสสูงมากที่ผลลัพธ์ของเมสเสจไดเจสต์ที่ได้นั้นเปลี่ยนไป ทำให้การตรวจสอบความถูกต้องของลายมือชื่อไม่ถูกต้อง

เอสเอชเอถูกออกแบบมาเพื่อให้มีคุณสมบัติดังนี้คือ ไม่สามารถกู้ข้อความกลับมาได้จากเมสเสจ-ไดเจสต์หรือเป็นไปยากมากที่จะมีข้อความต่างกัน 2 ข้อความใด ๆ ที่สร้างเมสเสจไดเจสต์ได้เหมือนกัน

3.5.1 ขั้นตอนการหาเมสเสจไดเจสต์

Message Padding

เอสเอชเอ รับบิตสตริงเป็นอินพุต ดังนั้นข้อความจะถูกมองเป็นบิตสตริง ความยาวของข้อความเป็นจำนวนบิต (ข้อความว่างเปล่ามีความยาวเป็น 0) ถ้าจำนวนของบิตในข้อความเป็นหลายเท่าของ 8 บิตแล้ว เพื่อความกระทัดรัดในการแทนค่าเราสามารถแทนข้อความนี้ด้วยเลขฐานสิบหก

สมมุติว่าข้อความมีความยาว L < 2^64 ก่อนที่มันจะไปเป็นอินพุตให้กับ เอสเอชเอข้อความนั้นจะถูกเติมเต็มที่ด้านขวาด้วยบิต "1" หนึ่งบิตและเติมบิต "0" จนกว่าบล็อกสุดท้ายจะมีขนาด 448 บิต

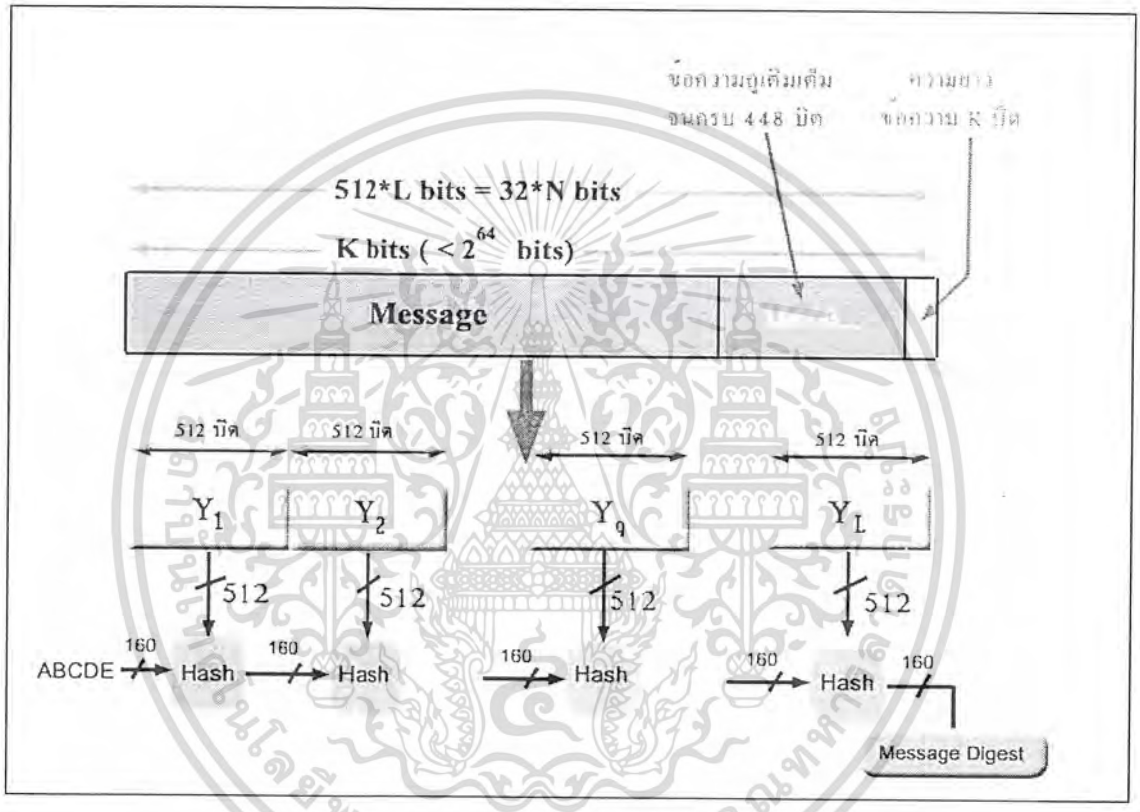
เติมค่าความยาวของข้อความ

ให้ L เป็นจำนวนบิตของข้อความเริ่มต้นโดย L มีความยาว 2 เวิร์ด (64 บิต) ถ้า L < 2^32 แล้วเวิร์ดแรกจะเป็น 0 ทั้งหมด ให้ต่อท้าย 2 เวิร์ดนี้เข้าไปที่ข้อความที่ถูกเติมเต็มแล้ว (L ถูกคำนวณเอาไว้ก่อนการเติมเต็ม) ข้อความที่ถูกเติมเต็มเสร็จเรียบร้อยแล้วจะมีจำนวนเวิร์ดทั้งสิ้น 16N เวิร์ด โดยที่ N > 0 ข้อความที่เติมเต็มสมบูรณ์แล้วอาจจะถูกมองเป็นลำดับของบล็อกจำนวน N บล็อกคือ M(1), M(2), ... , M(N) โดยที่แต่ละ M(i) มี 16 เวิร์ดและ M(1) อยู่ด้านซ้ายสุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การคำนวณหา Message Digest

การคำนวณหาเมสเสจไดเจสต์ทำได้โดยใช้ข้อความที่ผ่านการเติมเต็ม (Padding) เรียบร้อยแล้ว การคำนวณใช้บัฟเฟอร์ 2 ตัว แต่ละตัวประกอบด้วยเวิร์ดขนาด 32 บิตจำนวน 5 เวิร์ด และกลุ่มของเวิร์ดขนาด 32 บิตจำนวน 80 เวิร์ดเรียงกัน บัฟเฟอร์ตัวแรกกำหนด แต่ละเวิร์ดเป็น A, B, C, D, E และบัฟเฟอร์ 5 เวิร์ดตัวที่สองมีกำหนดเป็น h0, h1, h2, h3, h4 ส่วนเวิร์ดที่มีจำนวน 80 เวิร์ดเรียงกันมี กำหนดเป็น W(0), W (1), ..., W(79) และมีบัฟเฟอร์ชั่วคราว (TEMP) ขนาด 1 เวิร์ดไว้ใช้งานด้วย



รูปที่ 3-8 การคำนวณหาเมสเสจไดเจสต์ด้วยอัลกอริทึม SHA

การสร้างเมสเสจไดเจสต์นั้น กำหนดให้แต่ละบล็อกข้อความมีขนาด 16 เวิร์ด M(1), M(2),... , M(N) จะถูกประมวลผลตามลำดับ การประมวลผลของแต่ละ M(i) นั้นมี 80 ขั้นตอนด้วยกัน ก่อนที่จะประมวลผลบล็อกใด ๆ ให้กำหนดค่า h_j เริ่มต้นดังต่อไปนี้ในค่าเลขฐานสิบหก

h0 = 67452301 h1 = efcdab89 h2 = 98badcfe h3 = 10325476 h4 = c3d2e1f0

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การประมวลผล M(1), M(2), ..., M(N) มีดังต่อไปนี้

1. แบ่ง M(i) ออกเป็น 16 เวิร์ด W(0), W(1), .. , W(15) โดยที่ W(0) เป็นเวิร์ดด้านซ้ายสุด
2. ตั้งแต่ t = 16 ถึง 79 ให้ W(t) = W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)
3. ให้ A = h0, B = h1, C = h2, D = h3, E = h4
4. ตั้งแต่ t = 0 ถึง 79 ให้ทำดังนี้

$$TEMP = S(5,A) + f(t,B,C,D) + E + W(t) + K(t);$$

$$E = D;\ D = C;\ C = S(30,B);\ B = A;\ A = TEMP;$$

5. ให้ h0 = h0 + A, h1 = h1 + B, h2 = h2 + C, h3 = h3 + D, h4 = h4 + E

หลังจากการประมวลผล M(N) แล้วจะได้เมสเสจไดเจสต์ขนาด 160 บิตที่แทนด้วยเวิร์ด 5 เวิร์ด เป็น h0 h1 h2 h3 h4

3.6 Pseudo Random Number Generator (PRNG)

PRNG ทำหน้าที่ในการสร้างสตรีม (Stream) ของตัวเลขซึ่งมีการสุ่มแบบกระจาย PRNG ไม่ใช่ตัวเลขที่ถูกสร้างขึ้นมานั้นจากการสุ่มโดยทั่วไป แต่มีการใช้อัลกอริทึมดีเทอร์มินิสติก (Deterministic Algorithm) และค่าที่ได้จากการสุ่มแต่ละครั้งที่โปรแกรมทำงานจะได้ค่าที่แตกต่างกันทำให้ยากต่อการคาดเดา การใช้อัลกอริทึมแบบดีเทอร์มินิสติกจะช่วยให้ PRNG สามารถสร้างสุ่มค่าตัวเลขออกมาได้เหมาะสมมากยิ่งขึ้น เนื่องจากลำดับการสุ่มมีลักษณะเป็นลำดับที่ไม่แน่นนอน และเป็นลำดับที่บริสุทธ์ (Purely Deterministic)

ในบางการคำนวณหากมีการแยกการคำนวณ ลักษณะของโปรแกรมจะต้องมีลักษณะไม่ขึ้นอยู่กับข้อมูล (Data Independency) เพราะหากแยกการคำนวณแล้วมาการรับค่า PRNG เข้ามาในแต่ละหน่วยของการคำนวณอาจจะทำให้ผลลัพธ์ผิดพลาดได้ การหาค่าจาก PRNG จำเป็นต้องมีการระบุค่าเริ่มต้น (Initial Value) หรือสีด (Seed) ดังนั้นการกำหนดค่าเริ่มต้น หรือสีดที่เหมือนกันจะทำให้ได้ค่าลำดับการสุ่มที่เหมือนกัน ดังนั้นหากเราต้องการได้ค่า PRNG ที่ต่างกันก็เพียงกำหนดค่าเริ่มต้นที่ต่างกัน

ไม่มีกฎตายตัวที่พิสูจน์ว่า PRNG แบบใดมีลักษณะที่ดี แต่ลักษณะโดยทั่วไปของ PRNG ที่ดีควรมีลัษณะดังนี้

1. ต้องคาดเดาได้ยาก นั่นคือเอาต์พุตที่ได้จาก PRNG ต้องไม่สามารถคาดเดาผลลัพธ์ได้ เช่นมีการสุ่มค่าในลักษณะที่ไม่เป็นลำดับที่แน่นอนเป็นต้น
2. ต้องไม่เกิดค่าซ้ำกันบ่อยๆ PRNG ที่ดีต้องไม่สร้างค่าที่ซ้ำกันบ่อยๆซึ่งจะทำให้เกิดการคาดเดาได้ง่าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อย่างไรก็ตามถึงแม้ว่าจะมีการใช้ PRNG ที่ดีที่ผ่านกฎมาตราฐานของอัลกอริทึมการสุ่มตัวเลขแต่ก็ไม่ได้เป็นการรับประกันว่าแอพพลิเคชันจะทำงานได้ดีตามด้วยเสมอไป เพราะ PRNG บางตัวอาจจะเหมาะสำหรับแอพพลิเคชันหนึ่งแต่อาจจะไม่เหมาะสำหรับอีกแอพพลิเคชันหนึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
โมเดลการออกแบบและแนวทางการเขียนโค้ด

4.1 โมเดลการออกแบบ

ในบทนี้เราอธิบายถึงโมเดลหนึ่งที่เราได้คิดขึ้นสำหรับช่วยให้มองการออกแบบได้ง่ายๆ โดยที่เราจะมองแต่ละคอมโพเน็นต์เสมือนเป็นอุปกรณ์ใดๆ ที่มีสวิตช์เปิดปิด มีอินเทอร์เฟซ (Interface) ในการควบคุมและมีเส้นสัญญาณในการรับส่งข้อมูล โดยเราได้กำหนดกฎกติกาหรือโพรโตคอล (Protocol) ในการใช้อุปกรณ์นี้ในลักษณะเดียวกันซึ่งจะอธิบายต่อไป โมเดลที่เราได้คิดขึ้นมี 2 แบบดังนี้

4.1.1 โมเดลสวิตชิงแบบพาสซีฟ (Passive switching)

รูปที่ 4-1 โมเดลแบบพาสซีฟ

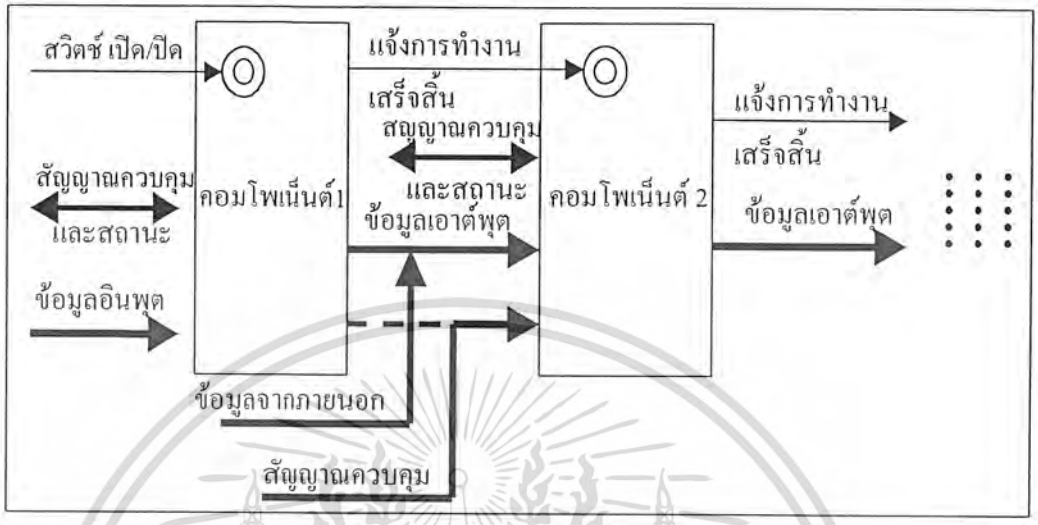
ข้อกำหนดการใช้งานของโมเดลนี้คือ

1. คอมโพเน็นต์จะทำงานก็ต่อเมื่อมีการเปิดสวิตช์เท่านั้น
2. หน้าที่การทำงานภายในคอมโพเน็นต์กำหนดโดยผู้ออกแบบ
3. คอมโพเน็นต์คุยกับคอมโพเน็นต์อื่น ๆ ผ่านสัญญาณควบคุม
4. รับข้อมูลมาประมวลผลและส่งผลลัพธ์ออกไปให้คอมโพเน็นต์อื่น
5. แจ้งให้ภายนอกทราบว่าการทำงานเสร็จสิ้นแล้วรอการปิดสวิตช์
6. ภายในคอมโพเน็นต์สามารถมีคอมโพเน็นต์ย่อย ๆ ซึ่งอาจใช้โมเดลแบบอื่นร่วมได้

เมื่อคอมโพเน็นต์ถูกสั่งการจากภายนอกและจะทำงานเพียงรอบเดียวเท่านั้นจนกว่าจะเสร็จสิ้น แล้วรอให้ภายนอกส่งสัญญาณมาปิดสวิตช์เสียก่อนจึงจะเริ่มการทำงานครั้งใหม่ต่อไปได้ เหตุที่กำหนดให้การทำงานอยู่ในลักษณะนี้ก็คือ สัญญาณที่แจ้งว่าการทำงานเสร็จสิ้นนั้นสามารถทำหน้าที่เป็นสวิตช์ที่ไปเปิดการทำงานของคอมโพเน็นต์อื่น ๆ ได้ สมมุติในกรณีที่เรามีคอมโพเน็นต์หลาย ๆ ตัวที่ต้องทำงานร่วมกัน

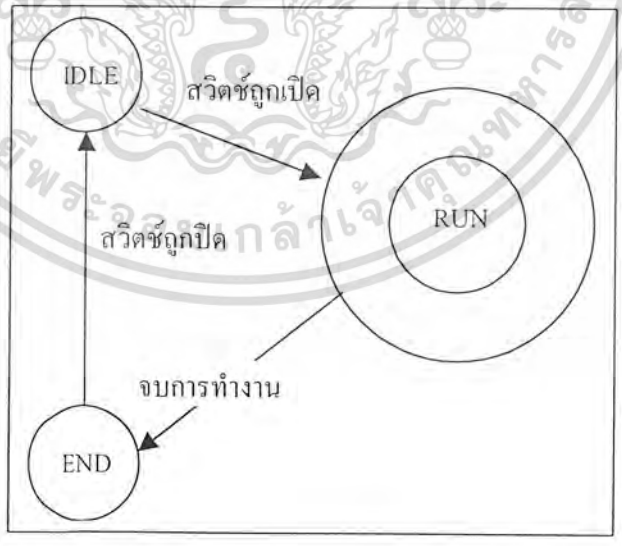
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แบบลำดับขั้น (Sequential) และแต่ละคอมโพเน็นต์ก็มีฟังก์ชันการทำงานต่างๆ กันไป เราก็สามารถใช้สัญญาณที่แจ้งการทำงานเสร็จสิ้นไปเปิดสวิตช์การทำงานของคอมโพเน็นต์ตัวต่อไปได้ในทันที ดังแสดงในรูปที่ 4-2



รูปที่ 4-2 ตัวอย่างการจัดวางคอมโพเน็นต์ให้ทำงานร่วมกัน

ภายในคอมโพเน็นต์อาจมีโพรเซสในการทำงานมากกว่า 1 โพรเซส ซึ่งการทำงานร่วมกันของโพรเซสเหล่านี้ก็ใช้โมเดลในลักษณะนี้เช่นเดียวกัน เราสามารถเขียนเป็นเสตตการทำงานอย่างง่ายได้ดังรูปที่ 4-3



รูปที่ 4-3 เสตตการทำงานที่ระดับบนสำหรับคอมโพเน็นต์หรือโพรเซส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.2 โมเดลสวิตช์ชิ่งแบบแอกทีฟ (Active switching)

โมเดลนี้มีพื้นฐานมาจากโมเดลแบบแรก แต่เพิ่มส่วนของสัญญาณที่ใช้ในการควบคุมสวิตช์ของคอมโพเน็นต์อื่นๆ ได้หลายคอมโพเน็นต์ในเวลาเดียวกัน หากจะเปรียบเทียบแล้วก็เหมือนหัวหน้าที่บริหารการสั่งงานให้ลูกน้องนั่นเอง เรามักใช้โมเดลนี้ในการออกแบบคอนโทรลเลอร์นำมาประกอบเข้ากับโมเดลแบบที่ I ทำให้เราได้ระบบการทำงานระบบหนึ่งที่มีการประสานงานร่วมกันระหว่างคอมโพเน็นต์เพื่อบรรลุการทำงานหนึ่ง โดยการจัดวางคอมโพเน็นต์และเชื่อมการติดต่อระหว่างคอมโพเน็นต์ให้เหมาะสม

รูปที่ 4-4 โมเดลแบบแอกทีฟสำหรับทำหน้าที่ในการควบคุมการทำงานหลัก

สำหรับโมเดลนี้เราอาจให้คอมโพเน็นต์รับผิดชอบการประมวลผลข้อมูลบางส่วนด้วยก็ได้ ซึ่งมักจะเป็นการประมวลผลหรือจัดการกับข้อมูลในระดับเล็ก ๆ ที่ไม่สำคัญอะไรมากนัก ต่อไปเราจะนำโมเดลทั้งสองแบบนี้มาใช้ร่วมกับแนวการเขียนโค้ดภาษาวีเอชดีแอลที่ช่วยออปติไมซ์ (Optimize) ในด้านของความเร็วและพื้นที่ได้ดีในระดับหนึ่งตามที่ผู้จัดทำได้ทดลองการเขียนมาในหลาย ๆ แบบ

4.2 แนวทางการเขียนโค้ด

สำหรับการเขียนโค้ดภาษาวีเอชดีแอล เราต้องการให้โค้ดทำงานได้รวดเร็วและใช้ทรัพยากรน้อย ทางผู้จัดทำจึงนำเสนอแนวการเขียนโค้ดอีกแบบหนึ่ง ซึ่งหวังว่าจะเป็นประโยชน์แก่ผู้ออกแบบระบบที่ใช้ภาษาวีเอชดีแอลในการทำงาน จากหัวข้อที่แล้วเราได้กล่าวถึงโมเดลในการออกแบบ เราจะนำมาประยุกต์ใช้ในการกำหนดอินเทอร์เฟซการเชื่อมต่อสัญญาณของแต่ละคอมโพเน็นต์ เพื่อให้พวกมันสามารถสื่อสารกันได้และมีลำดับการทำงานที่ถูกต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการเขียนโค้ดภาษาวีเอชดีแอลเพื่อบรรยายการทำงาน เราจะแบ่งส่วนของโค้ดออกเป็น 2 ส่วนหลักคือ

1. เสตตแมชีน (State Machine): เป็น Process ที่ทำหน้าที่ในการย้ายเสตต (Transition function) การทำงานโดยตรวจสอบสัญญาณเงื่อนไข (Condition signals) การใช้เสตตทำให้รู้ลำดับขั้นตอนการทำงาน ตัวอย่างโค้ดในส่วนนี้แสดงอยู่ในโค้ดรูปที่ 4-5
2. ส่วนของการกำหนดค่าของสัญญาณทั้งใน Sequential statement ที่อยู่ใน Processและ Concurrent statement ที่อยู่นอกโพรเซสโดยส่วนใหญ่แล้วเราพยายามที่จะให้โค้ดอยู่ใน concurrent statement เพราะการมีโพรเซสเพิ่มจะทำให้ความเร็วโดยรวมลดลง อีกทั้งการทำงานภายในโพรเซสยังเป็นแบบ ลำดับขั้น ตัวอย่างแสดงอยู่ในโค้ดรูปที่ 4-6

```
PROCESS (Clk, Switch_ON)
BEGIN
IF Switch_ON = '0' THEN          -- Switch is OFF
  State <= IDLE;
ELSIF Clk'EVENT AND Clk = '1' THEN
CASE State IS
WHEN IDLE =>
 IF Switch_ON = '1' THEN         -- Switch is ON
   State <= RUNNING;
 END IF;

WHEN RUNNING =>
  IF some_condition = TRUE THEN
    State <= END_OF_SERVICE;
  END IF;

WHEN END_OF_SERVICE =>
    State <= END_OF_SERVICE;

END CASE;
```

รูปที่ 4-5 แสดงตัวอย่างการเขียนเสตตแมชีนโดยใช้โมเดลแบบที่ 1 มาประยุกต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
MD5_RUNNING    <= '1' WHEN State = RUNNING
                  ELSE '0';
B              <= VAL1 WHEN State = RUNNING
ELSE VAL2 WHEN State = END_OF_SERVICE
                  ELSE VAL3;
-------------------------------------------------------------------------------
-- ตัวอย่างการเขียนแบบอยู่ใน PROCESS
PROCESS (Clk, State)
BEGIN
IF Clk'EVENT AND Clk = '1' THEN
CASE State IS
WHEN IDLE =>
MD5_RUNNING <= '0';
B <= VAL3;
WHEN RUNNING =>
MD5_RUNNING <= '1';
B <= VAL1;
WHEN END_OF_SERVICE =>
MD5_RUNNING <= '0';
B <= VAL2;
END CASE;
END IF;
END PROCESS;
```

รูปที่ 4-6 โค้ดแสดงตัวอย่างการกำหนดค่าสัญญาณใน concurrent statement เทียบกับใน PROCESS

จากโค้ดรูปที่ 4-5 จะเห็นว่าเสตตแมชีนจะทำงานก็ต่อเมื่อสัญญาณสวิตช์ (Switch_ON) มีค่าเท่ากับ '1' ซึ่งก็เทียบได้กับการเปิดสวิตช์ให้คอมโพเน็นต์นี้ทำงานนั่นเอง และในส่วนของการกำหนดค่าสัญญาณในโค้ดรูปที่ 4-6 จะเห็นว่าเมื่อเทียบการกำหนดค่าสัญญาณใน Process กับการเขียนใน concurrent statement นั้น การเขียนใน concurrent statement จะเขียนง่ายกว่าและทำให้คอมโพเน็นต์ทำงานได้รวดเร็วกว่ามาก อีกทั้งสัญญาณใน concurrent statement จะถูกขับเคลื่อนอยู่ตลอดเวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จึงมีข้อดีคือใช้ทรัพยากรน้อยลงและได้ความเร็วที่สูงขึ้น การเปลี่ยนแปลงค่าสัญญาณยังเป็นแบบเรียลไทม์ (Realtime) ด้วย แนวการเขียนโค้ดแบบนี้สามารถนำไปใช้ได้กับการออกแบบทั่วไป และประยุกต์ใช้กับโมเดลการออกแบบในหัวข้อที่แล้วได้เป็นอย่างดี

ข้อควรระวัง

การกำหนดค่าสัญญาณควรระวังการขับค่าสัญญาณซ้ำซ้อน (Multiple Drive) ตัวอย่างเช่น ใน concurrent statement

```
Z <= A + B;
Z <= C + D;
```

หรือมี 2 PROCESS ที่ขับค่าสัญญาณตัวเดียวกัน เช่นรูปที่ 4-7 การขับค่าสัญญาณเช่นนี้เป็นการขับแหล่งสัญญาณ 2 แหล่งมาชนกัน ทำให้ไม่สามารถตีค่าของสัญญาณที่ได้ว่ามีค่าเป็นเท่าไร

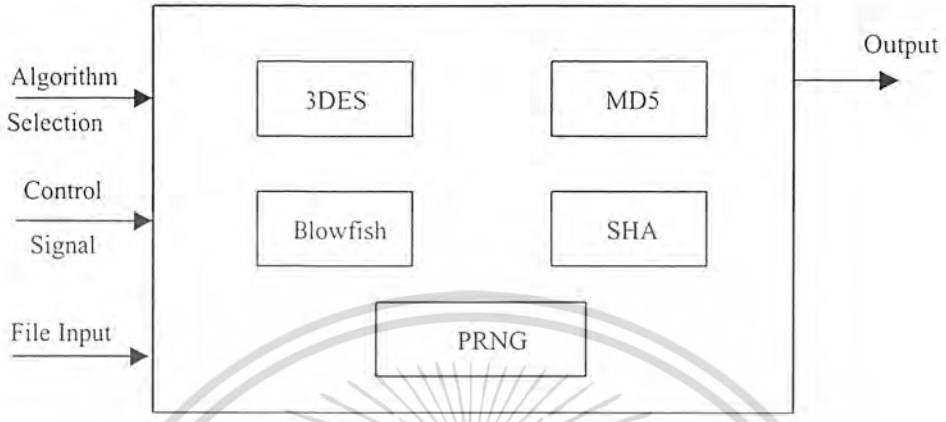
```
PROCESS_01 : PROCESS (Clk)
BEGIN
IF Clk'EVENT AND Clk = '1' THEN
   Z <= A + B;
END IF;
END PROCESS;

PROCESS_02 : PROCESS (Clk)
BEGIN
IF Clk'EVENT AND Clk = '1' THEN
    Z <= C + D;
END IF;
END PROCESS;
```

รูปที่ 4-7 แสดงตัวอย่างการชนกันของสัญญาณ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3 การออกแบบคอมโพเน็นต์สำหรับแต่ละอัลกอริทึม

4.3.1 คอมโพเน็นต์ Crypto Chip

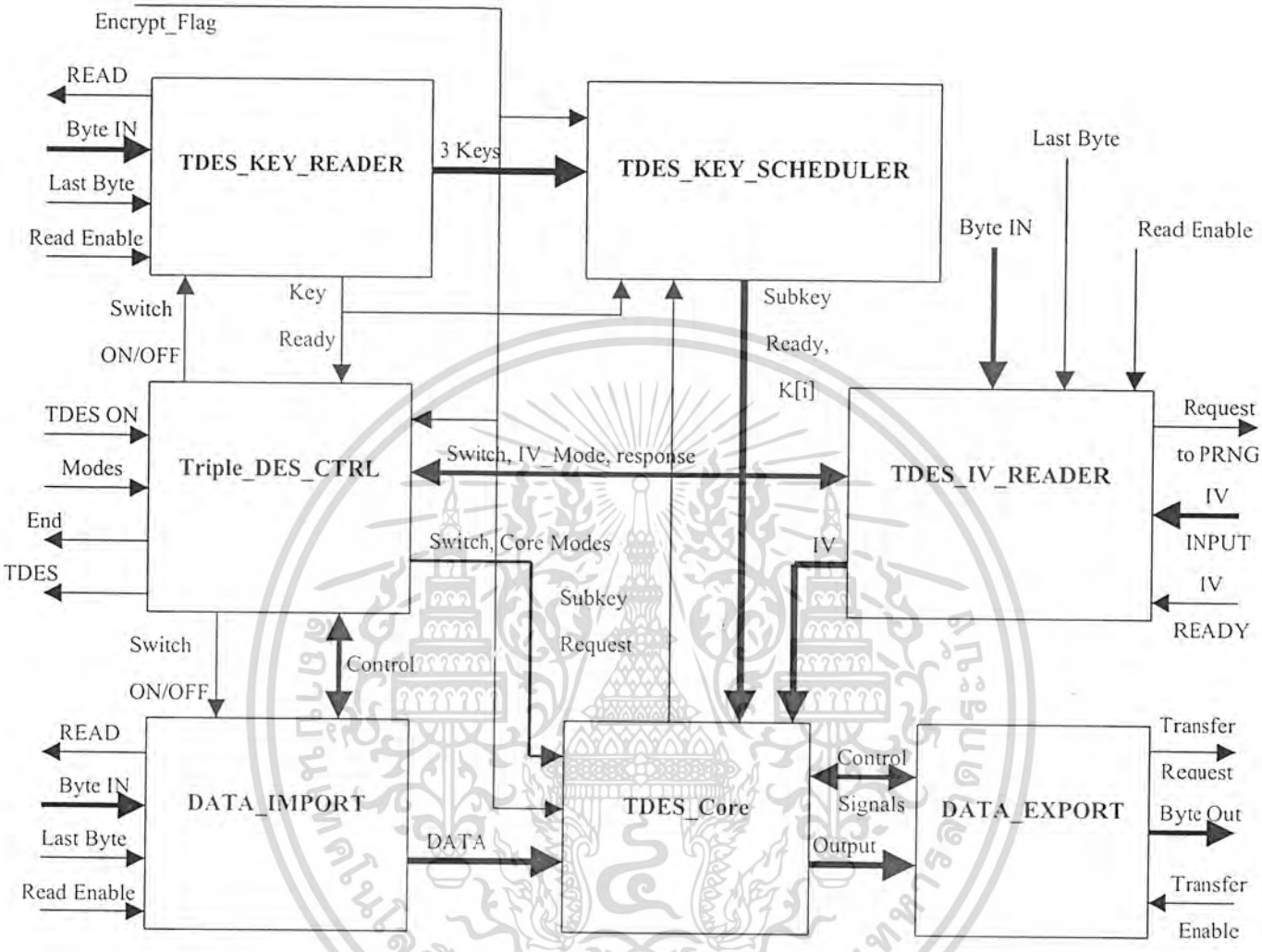


รูปที่ 4-8 แสดงคอมโพเน็นต์ Crypto Chip

คอมโพเน็นต์ของ Crypto Chip ประกอบด้วยส่วนของการเข้าและถอดรหัสซึ่งใช้อัลกอริทึม 3DES และ Blowfish มีอัลกอริทึมที่ใช้ในการลงลายมือชื่อดิจิตอลได้แก่ MD5 ซึ่งรับอินพุตข้อมูลไม่จำกัดขนาดเพื่อสร้างเอาต์พุต 128 บิต และ SHA ก็เช่นเดียวกันแต่สร้างเอาต์พุตออกมา 160 บิต นอกจากนี้ยังมีส่วนประกอบของ Random Number Generator (RNG) ช่วยในการผลิตค่าสุ่มสำหรับนำไปใช้ประกอบการทำงานของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.2 คอมโพเน็นต์ 3DES



รูปที่ 4-9 บล็อกไดอะแกรมแสดงการเชื่อมต่อของโมดูลย่อยของ Triple DES

Triple DES ประกอบด้วยคอมโพเน็นต์ย่อย 5 ตัว คือ

1. Triple_DES_CTRL
2. TDES_KEY_READER
3. TDES_KEY_SCHEDULER
4. TDES_Core
5. TDES_IV_READER
6. DATA_IMPORT
7. DATA_EXPORT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

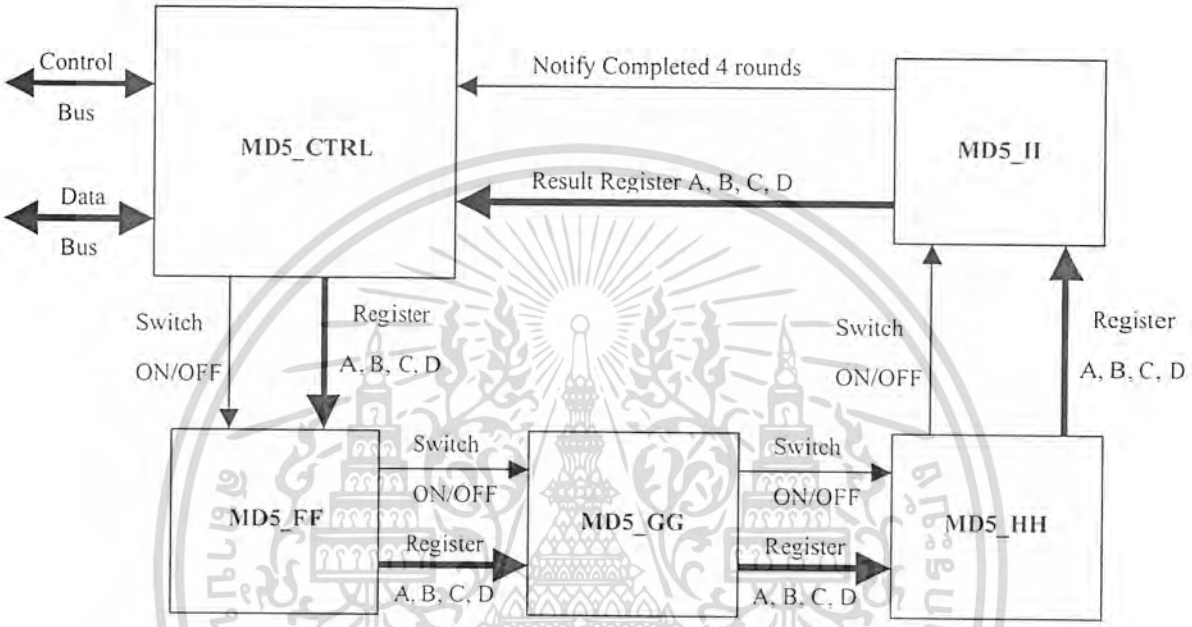
รายละเอียดการทำงานของแต่ละโมดูลของ Triple DES

1. **Triple_DES_CTRL:** ทำหน้าที่ควบคุมการเปิดปิดสวิตช์การทำงานให้กับ TDES_KEY_READER, TDES_IV_READER และ DATA_IMPORT แล้วสื่อสารกันเพื่อทำงานตามขั้นตอนที่กำหนดลำดับเอาไว้ใน Triple_DES_CTRL การทำงานทั้งหมดจะถูกปิดสวิตช์ก็ต่อเมื่อ DATA_EXPORT ส่งถ่ายผลลัพธ์เสร็จ
2. **TDES_KEY_READER:** ทำหน้าที่อ่านคีย์เข้ามาซึ่งเป็นขั้นตอนแรกที่ต้องทำ แล้วส่งคีย์ทั้ง 3 ชุด (192 บิต)ให้กับ TDES_KEY_SCHEDULER เพื่อคำนวณหาคีย์ย่อย 16 ชุดที่ต้องใช้ในการคำนวณของ TDES_Core
3. **TDES_KEY_SCHEDULER:** ทำหน้าที่รอรับคีย์ทั้ง 3 ชุดมาแล้วสร้างคีย์ย่อยส่งให้ TDES_Core สำหรับการคำนวณ 16 รอบของ TDES_Core
4. **DATA_IMPORT:** ทำหน้าที่อ่านบล็อกข้อมูลเข้ามาบล็อกละ 8, 16, 32, หรือ 64 บิต ขึ้นอยู่กับโหมดการทำงานของ Triple DES ซึ่งมี 8 โหมดให้เลือกใช้ (ดูข้างล่าง) แล้วส่งให้กับ TDES_Core
5. **TDES_IV_READER:** โมดูลนี้จะถูกเรียกใช้งานเมื่อไม่ได้อยู่ในโหมด ECB ทำหน้าในการอ่านค่า IV จากโมดูล RNG (ในกรณีการเข้ารหัส) หรืออ่านจากข้อมูลอินพุต (ในกรณีการถอดรหัส) แล้วส่งค่า IV ให้ TDES_Core เพื่อทำการคำนวณ
6. **TDES_Core:** ทำหน้าที่หลักในการเข้ารหัสและถอดรหัสลับของข้อมูลในโหมดที่ผู้ใช้เลือก โดยพารามิเตอร์ที่จำเป็นคือ Data ที่ได้จาก DATA_IMPORT, Subkey ที่ได้จาก TDES_KEY_SCHEDULER และ/หรือ IV (Initialize Vector) ที่ได้จาก TDES_IV_READER เมื่อทำการคำนวณเสร็จและไม่มีข้อมูลต่อไปแล้วก็ทำการส่งผลลัพธ์ให้กับ DATA_EXPORT เพื่อส่งผลลัพธ์ไปยังผู้ใช้
7. **DATA_EXPORT:** ทำหน้าที่ส่งถ่ายผลลัพธ์ที่ได้ไปยังผู้ใช้โดยรอการร้องขอจาก Triple_DES_CTRL เมื่อส่งถ่ายผลลัพธ์เสร็จแล้วก็แจ้งให้ Triple_DES_CTRL รับทราบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.3 คอมโพเน็นต์ MD5

MD5 ประกอบด้วยคอมโพเน็นต์ย่อย 5 ตัว คือ MD5_CTRL, MD5_FF, MD5_GG, MD5_HH, MD5_II รายละเอียดเกี่ยวกับการเชื่อมต่อและการทำงานของแต่ละโมดูลมีดังต่อไปนี้



รูปที่ 4-10 บล็อกไดอะแกรมแสดงการเชื่อมต่อระหว่างคอมโพเน็นต์ย่อยของ MD5

รายละเอียดการทำงานของแต่ละโมดูลของ MD5

1. **MD5_CTRL:** ทำหน้าที่ควบคุมโมดูลอื่น ๆ ทั้งหมดโดยสามารถเปิดหรือปิดสวิตช์การทำงานของโมดูลอื่นๆ ได้ ถ้าโมดูล MD5_FF ถูกปิดสวิตช์ โมดูลอื่น ๆ ที่ MD5_FF เปิดสวิตช์ให้ จะถูกปิดสวิตช์การทำงานไปด้วย MD5_CTRL จะอ่านบล็อกข้อมูล 512 บิตเข้ามาแล้วกระจายให้กับทุกโมดูลย่อย จากนั้นก็เปิดสวิตช์การทำงานให้กับ MD5_FF โดยให้ค่าพารามิเตอร์ที่จำเป็นในการคำนวณให้ด้วย เมื่อการคำนวณทั้งหมดทำจนถึงข้อมูลบล็อกสุดท้าย มันก็จะแจ้งให้โมดูลที่อยู่ภายนอกส่งค่าแฮชที่คำนวณได้ออกไป
2. **MD5_FF:** ทำหน้าที่รับผิดชอบการคำนวณในรอบที่ 1 ซึ่งแต่ละรอบมี 16 โอเปอเรชั่น เมื่อคำนวณเสร็จแล้ว ก็ทำการเปิดสวิตช์ MD5_GG แล้วส่งผลที่ได้ให้กับ MD5_GG ไปคำนวณต่อในรอบต่อไป จะเห็นว่ามี 4 โมดูลที่ทำการคำนวณในแต่ละรอบ เพราะว่าการคำนวณในแต่ละรอบมีฟังก์ชันต่างกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

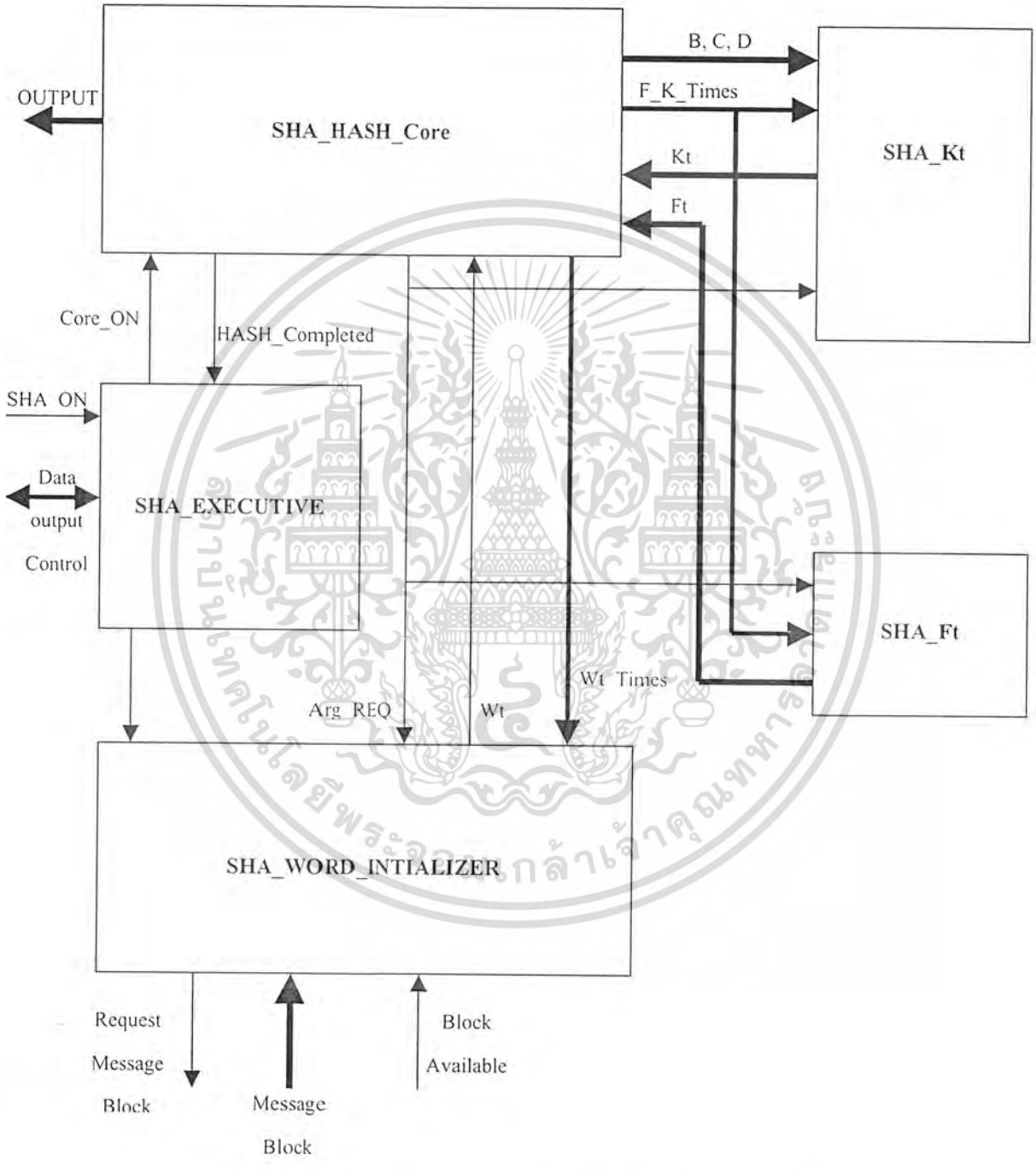
3. **MD5_GG:** ทำหน้าที่รับผิดชอบการคำนวณในรอบที่ 2 โดยรับค่าอินพุต A B C และ D ที่ได้จากการคำนวณของ MD5_FF มาคำนวณ เมื่อเสร็จแล้วก็ทำการเปิดสวิตช์ MD5_HH แล้วส่งผลลัพธ์ให้กับ MD5_HH เพื่อทำการคำนวณในรอบต่อไป
4. **MD5_HH:** ทำหน้าที่รับผิดชอบการคำนวณในรอบที่ 3 โดยรับค่าอินพุต A B C และ D ที่ได้จากการคำนวณของ MD5_GG มาคำนวณ เมื่อเสร็จแล้วก็ทำการเปิดสวิตช์ MD5_II แล้วส่งผลลัพธ์ให้กับ MD5_II เพื่อทำการคำนวณในรอบต่อไป
5. **MD5_II:** ทำหน้าที่รับผิดชอบการคำนวณในรอบที่ 4 แล้ว เมื่อคำนวณเสร็จก็แจ้งให้ MD5_CTRL ทราบพร้อมทั้งส่งผลลัพธ์ที่ได้ให้กับ MD5_CTRL ด้วย

เนื่องจากการประมวลผลข้อมูลในแต่ละบล็อกมีทั้งหมด 4 รอบ ๆ ละ 16 โอเปอเรชั่น และในแต่ละรอบนั้นก็มีการนิยามฟังก์ชันการคำนวณที่แตกต่างกัน และยังคำนวณเป็นแบบลำดับ จึงกำหนดให้มีโมดูลย่อยทั้ง 4 ที่รับผิดชอบด้านการคำนวณเพียงอย่างเดียว และให้มีโมดูลหนึ่งที่ทำหน้าที่อ่านบล็อกข้อมูลและคุยกับคอมโพเน็นต์อื่นที่อยู่ภายนอก รวมทั้งควบคุมการทำงานของโมดูลย่อยที่อยู่ภายใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.4 คอมโพเน็นต์ SHA

SHA ประกอบด้วยคอมโพเน็นต์ย่อย5ตัวคือ SHA_EXECUTIVE, SHA_WORD_INITIALIZER, SHA_HASH_Core, SHA_Ft, SHA_Kt รายละเอียดการเชื่อมต่อและการทำงานของแต่ละโมดูลมีดังต่อไปนี้



รูปที่ 4-11 บล็อกไดอะแกรมแสดงการทำงานของโมดูลย่อยของ SHA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

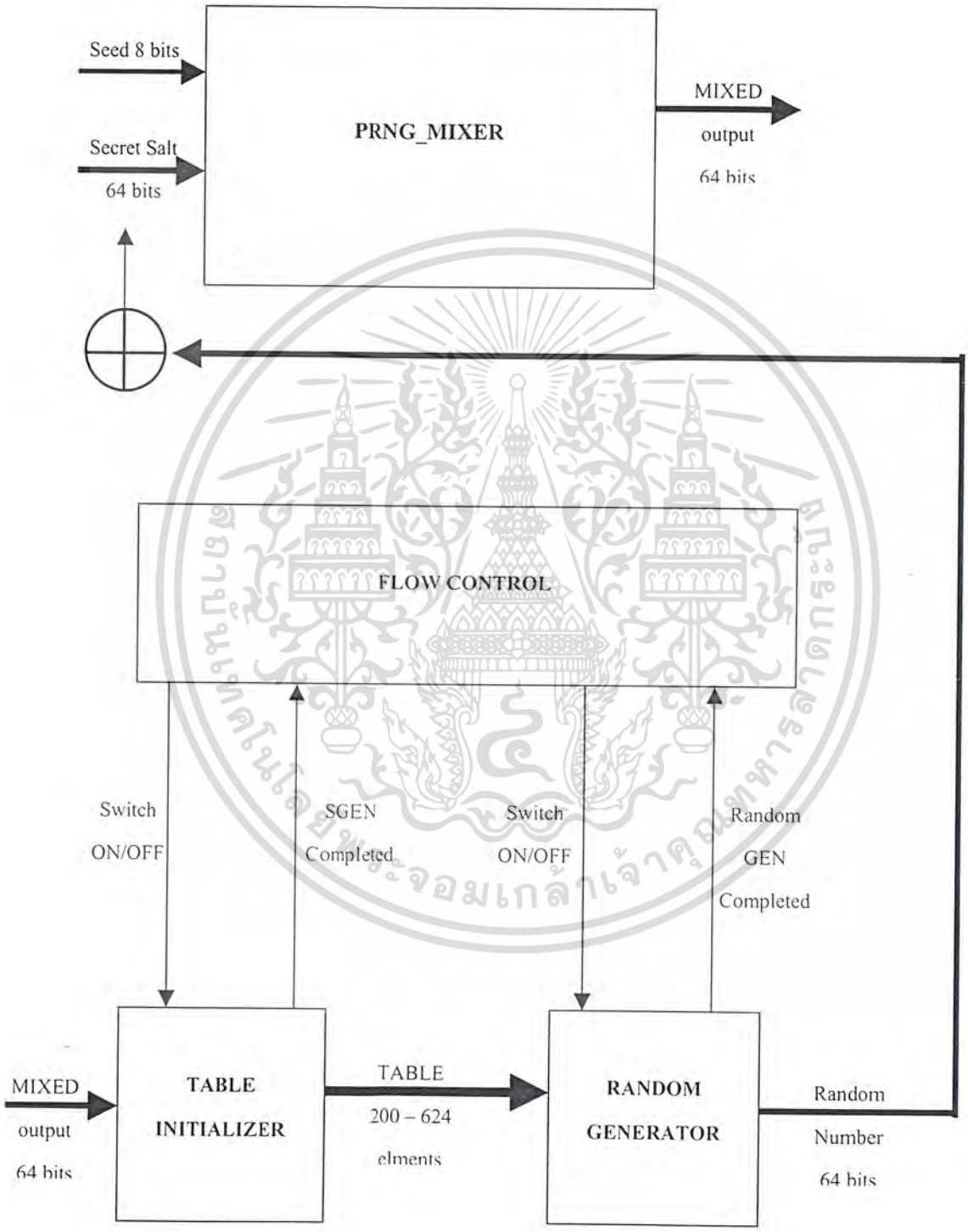
รายละเอียดการทำงานของแต่ละโมดูลของ SHA

1. **SHA_EXECUTIVE:** โมดูลนี้เป็นโมดูลหลัก เสมือนเป็นผู้สั่งการทำงานของโมดูลอื่น ๆ ทั้งหมด มันทำหน้าที่ควบคุมการเปิดสวิตช์การทำงานของSHA_HASH_Core และ SHA_WORD_INITIALIZER รวมถึงการควบคุมว่าเมื่อไหร่จะเปิดสวิตช์ของโมดูลภายนอกที่ทำหน้าที่ในการส่งผลลัพธ์ทำการส่งผลลัพธ์ออกไป
2. **SHA_HASH_Core:** เป็นแกนหลักสำหรับการคำนวณค่าแฮชโดยการร้องขอค่าอาร์กิวเม็นต์ (Argument) ที่จำเป็นสำหรับการคำนวณจาก SHA_Kt, SHA_Ft, และ SHA_WORD_INITIALIZER มันทำการประมวลผลข้อมูลในแต่ละบล็อกแล้วจ่ายผลลัพธ์ออกไปทันทีแล้วแจ้งให้ SHA_EXECUTIVE รู้ว่าประมวลเสร็จแล้วใน 1 บล็อก ผลลัพธ์จะถูกส่งออกไปยังระบบภายนอกหรือไม่นั้น SHA_EXECUTIVE จะควบคุมเอง
3. **SHA_WORD_INITIALIZER:** ทำหน้าที่ในการอ่านบล็อกข้อมูล 512 บิตเข้ามาแบ่งออกเป็น 16 เวิร์ด ๆ ละ 32 บิต แล้วนำมาทำการขยายเวิร์ดบล็อกเพิ่มเป็น 80 เวิร์ดทั้งหมด (รวม 16 เวิร์ดแรกของบล็อกข้อมูลที่อ่านมาด้วย) จากนั้นรอการร้องขอ Wt จาก SHA_HASH_Core จนกว่าจะครบทั้ง 80 เวิร์ดแล้วจึงทำการอ่านบล็อกข้อมูลบล็อกต่อไปมาเตรียมเวิร์ดทั้ง 80 เวิร์ดให้ SHA_HASH_Core ต่อไป
4. **SHA_Kt:** ทำหน้าที่ส่งค่า Kt แก่ SHA_HASH_Core โดย SHA_HASH_Core จะบอกว่าขณะนั้นเป็นการคำนวณรอบที่เท่าใด (ทั้งหมด 80 รอบ) SHA_Kt ก็จะให้ค่า Kt ออกมาตามนั้น
5. **SHA_Ft:** ทำหน้าที่เช่นเดียวกันกับ SHA_Kt คือคอยส่งค่า Ft ให้กับ SHA_HASH_Core แค่เพียงรู้ว่าเป็นการคำนวณรอบที่เท่าใด มันก็จะให้ค่า Ft ที่ต้องใช้ในการคำนวณรอบนั้นออกมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.5 คอมโพเน็นต์ PRNG (Pseudo Random Number Generator)

PRNG ประกอบด้วยคอมโพเน็นต์ PRNG_MIXER ทำงานร่วมกับ โพรเซสย่อยอีก 3 โพรเซสดังนี้



รูปที่ 4-12 บล็อกไดอะแกรมแสดงการทำงานร่วมกันของโมดูล Pseudo Random Number Generator

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รายละเอียดการทำงานของ PRNG

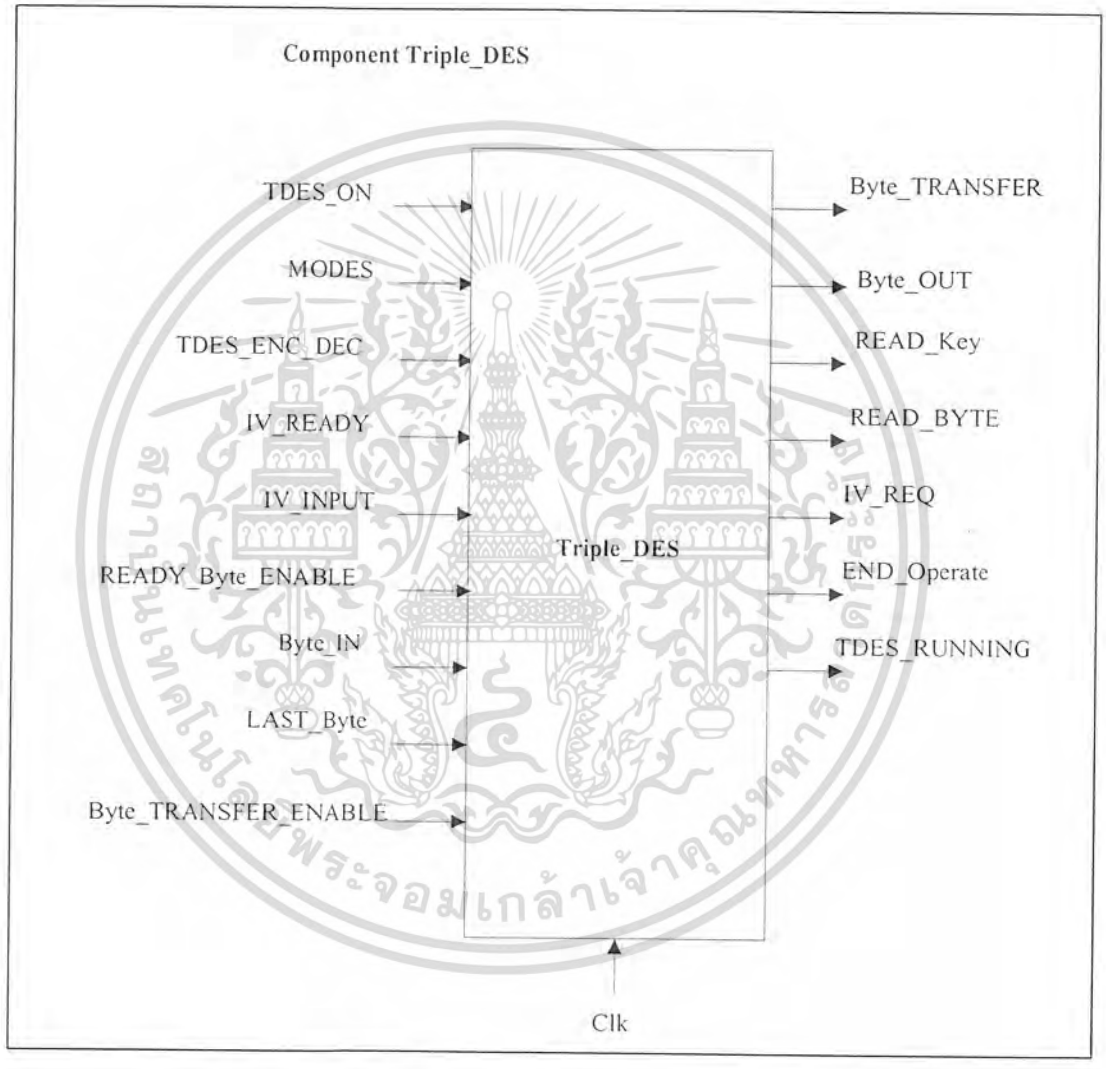
1. **PRNG_MIXER:** ทำหน้าที่รับ Seed เข้ามาตลอดเวลาแล้วนำมาผสมกับ Secret_Salt ซึ่ง Secret_Salt จะเป็นค่าที่เก็บไว้เป็นความลับแต่ก็จะถูกเปลี่ยนค่าโดยนำมาทำโอเปอเรชั่นกับเอาต์พุตที่ได้
2. **FLOW CONTROL:** ทำหน้าที่ควบคุมการทำงานของโพรเซส TABLE INITIALIZER และ RANDOM GENERATOR ซึ่งจะควบคุมให้ทั้งสองโพรเซสทำงานสลับกันไปเรื่อย ๆ ทำให้ค่า Random Number ที่ได้มีค่าเปลี่ยนแปลงอยู่ตลอดเวลา อีกทั้งอินพุตที่เข้ามาคือค่า MIXED ซึ่งเป็นค่าที่ได้จาก PRNG_MIXER ที่ทำการผสมค่า MIXED นี้มาให้อยู่ตลอดเวลา
3. **TABLE INITIALIZER:** นำค่า MIXED ณ เวลาที่ TABLE INITIALIZER ถูกเปิดสวิตช์ให้ทำงานมาสร้างตารางเพื่อให้ส่งให้โพรเซส RANDOM GENERATOR
4. **RANDOM GENERATOR:** อ่านค่าจากตารางที่ TABLE INITIALIZER สร้างมาคำนวณหาค่า random แล้วป้อนกลับไปให้โมดูล PRNG_MIXER โดยทำโอเปอเรชั่นกับ Secret_Salt

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
คำอธิบายขาสัญญาณของแต่ละคอมโพเน็นต์

5.1 Triple Data Encryption Standard (3DES)

5.1.1 Triple_DES (level 0)



รูปที่ 5-1 แสดง เอนติตีของ Triple DES ที่ระดับบนสุด

Triple DES เป็นคอมโพเน็นต์สำหรับการเข้าและถอดรหัสลับของข้อมูล โดยสามารถเลือกโหมดการทำงานได้ทั้งหมด 8 โหมด คือ ECB, CBC, CFB 8 บิต, CFB 16 บิต, CFB 32 บิต, OFB 8 บิต, OFB 16 บิต, OFB 32 บิต จากรูปที่ 5-1 แสดงคอมโพเน็นต์ของ Triple DES ที่ระดับบนสุด (Top level)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำอธิบายขาสัญญาณ

- **Clk (IN):** ขาสัญญาณนาฬิกาสำหรับ synchronize การทำงานของคอมโพเน็นต์ทั้งหมด
- **TDES_ON (IN):** สัญญาณที่ทำหน้าที่เป็นสวิตช์เปิด / ปิดการทำงานสำหรับคอมโพเน็นต์ '1' แทนการเปิดสวิตช์ '0' แทนการปิดสวิตช์ การทำงานจะดำเนินอยู่ก็ต่อเมื่อสวิตช์เปิดเท่านั้น
- **Modes (IN 2:0):** โหมดการทำงานของ Triple DES

  [2 1 0]

  "0 0 0" โหมด ECB

  "0 0 1" โหมด CBC

  "0 1 0" โหมด CFB 8 บิต

  "0 1 1" โหมด OFB 8 บิต

  "1 0 0" โหมด CFB 16 บิต

  "1 0 1" โหมด OFB 16 บิต

  "1 1 0" โหมด CFB 32 บิต

  "1 1 1" โหมด OFB 32 บิต

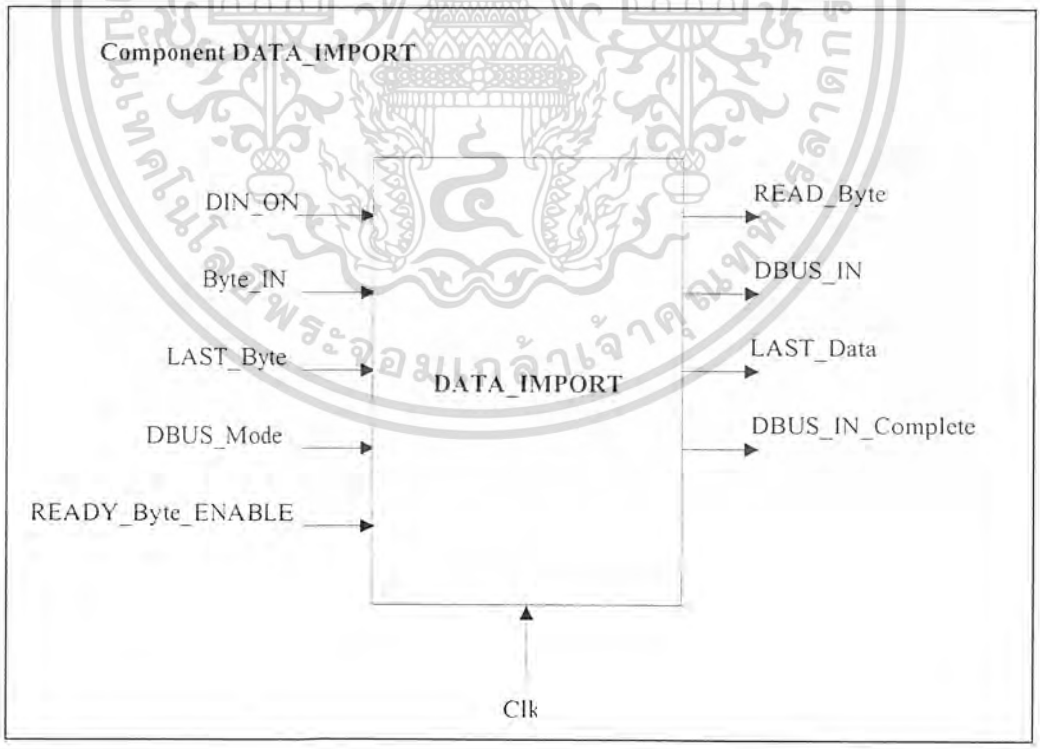
- **TDES_ENC_DEC (IN):** '1' หมายถึง การเข้ารหัส '0' หมายถึง การถอดรหัส
- **IV_READY (IN):** สัญญาณที่บอกว่าตอนนี้ค่า IV (Initialize Vector) มีอยู่ที่ขาสัญญาณ IV _INPUT แล้ว
- **IV_INPUT (IN 0:63):** บัสสัญญาณของค่า IV ที่นำมาใช้ร่วมกับการเข้าและถอดรหัสในโหมด CBC, CFB, OFB
- **READ_Byte_ENABLE (IN):** '1' หมายถึง Enable บอกให้ Triple DES รู้ว่าบัสข้อมูลอินพุตว่าง สามารถร้องขอข้อมูลได้ ถ้าเป็น '0' หมายถึงกำลังมีการใช้บัสข้อมูลอินพุตอยู่
- **Byte_IN (IN 7:0):** บัสข้อมูลอินพุตขนาด 8 บิต
- **LAST_Byte (IN):** แจ้งให้ทราบว่าตอนนี้ข้อมูลอินพุตที่กำลังอ่านอยู่นั้นเป็นไบต์สุดท้ายแล้ว
- **Byte_TRANSFER_ENABLE (IN):** สัญญาณแสดงสถานะการว่างของบัสข้อมูลเอาต์พุต ถ้ามีค่าเป็น '1' หมายถึงบัสส่งข้อมูลเอาต์พุตนั้นว่างไม่มีใครใช้ ถ้าเป็น '0' หมายถึงมีการใช้บัสข้อมูลเอาต์พุตอยู่ ห้ามโมดูลอื่นส่งสัญญาณอื่นเข้ามาชน
- **Byte_TRANSFER (OUT):** สัญญาณร้องขอเพื่อทำการส่งข้อมูลออกไปที่บัสข้อมูลเอาต์พุต โดยจะร้องขอได้ก็ต่อเมื่อ Byte_TRANSFER_ENABLE มีค่าเป็น '1' หากสามารถทำการร้องขอได้ ต้องรอให้ Byte_TRANSFER_ENABLE มีค่าเป็น '0' เสียก่อนเพื่อป้องกันไม่ให้คอมโพเน็นต์อื่นมาร้องขอส่งข้อมูล เป็นการจองการใช้บัสข้อมูล หลังจากส่งข้อมูลออกไปแล้วให้ยกเลิกการร้องขอโดยขับค่าสัญญาณให้เป็น '0' เพื่อให้ระบบภายนอกรู้ว่าข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พร้อมอยู่ที่บัสแล้ว ระบบภายนอกก็จะอ่านข้อมูลเอาต์พุตไปพร้อมทั้งปลดปล่อยการจองบัสข้อมูล

- **Byte_OUT (OUT 7:0):** บัสข้อมูลเอาต์พุต เพื่อส่งข้อมูลออกไปยังระบบภายนอก
- **READ_Key (OUT):** สัญญาณร้องขอเพื่ออ่านคีย์สำหรับการเข้าและถอดรหัสเข้ามาที่บัสข้อมูลอินพุต หากบัสข้อมูลอินพุตยังไม่มีการจองใช้งาน ให้ส่งสัญญาณ '1' ออกไปเพื่อขอจองใช้ บัส เมื่อระบบภายนอกได้รับสัญญาณร้องขอ ก็ทำการจองบัสโดยส่งสัญญาณ Byte_READ_ENABLE เป็น '0' มาให้พร้อมทั้งส่งข้อมูลมาด้วยที่บัส Byte_IN จากนั้นทำการอ่านข้อมูลแล้วทำการยกเลิกการร้องขอ
- **READ_BYTE (OUT):** สัญญาณร้องขอเพื่ออ่านข้อมูลที่จะนำมาเข้าและถอดรหัส
- **IV_REQ (OUT):** สัญญาณร้องขอค่า IV เข้ามาที่บัส IV_INPUT เพื่อใช้ในการเข้าและถอดรหัสในโหมดอื่น ๆ ที่ไม่ใช่ ECB
- **End_Operate (OUT):** สัญญาณที่แจ้งให้ระบบภายนอกรู้ว่าการทำงานทั้งหมดเสร็จสิ้นแล้วตั้งแต่เริ่มทำงานจนถึงการส่งถ่ายผลลัพธ์ออกไป
- **TDES_RUNNING (OUT):** แจ้งให้ระบบภายนอกรับรู้ว่า Triple DES กำลังทำงานอยู่

5.1.2 DATA_IMPORT (level 1)



รูปที่ 5-2 แสดงเอนติตีของ DATA_IMPORT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำหน้าที่หลักในการอ่านข้อมูลเข้ามาเพื่อทำการเข้าและถอดรหัส โดยสามารถทำการเติมเต็มบิตข้อมูลให้เต็มบล็อกได้โดยอัตโนมัติ หน้าที่ของสัญญาณต่าง ๆ มีดังนี้

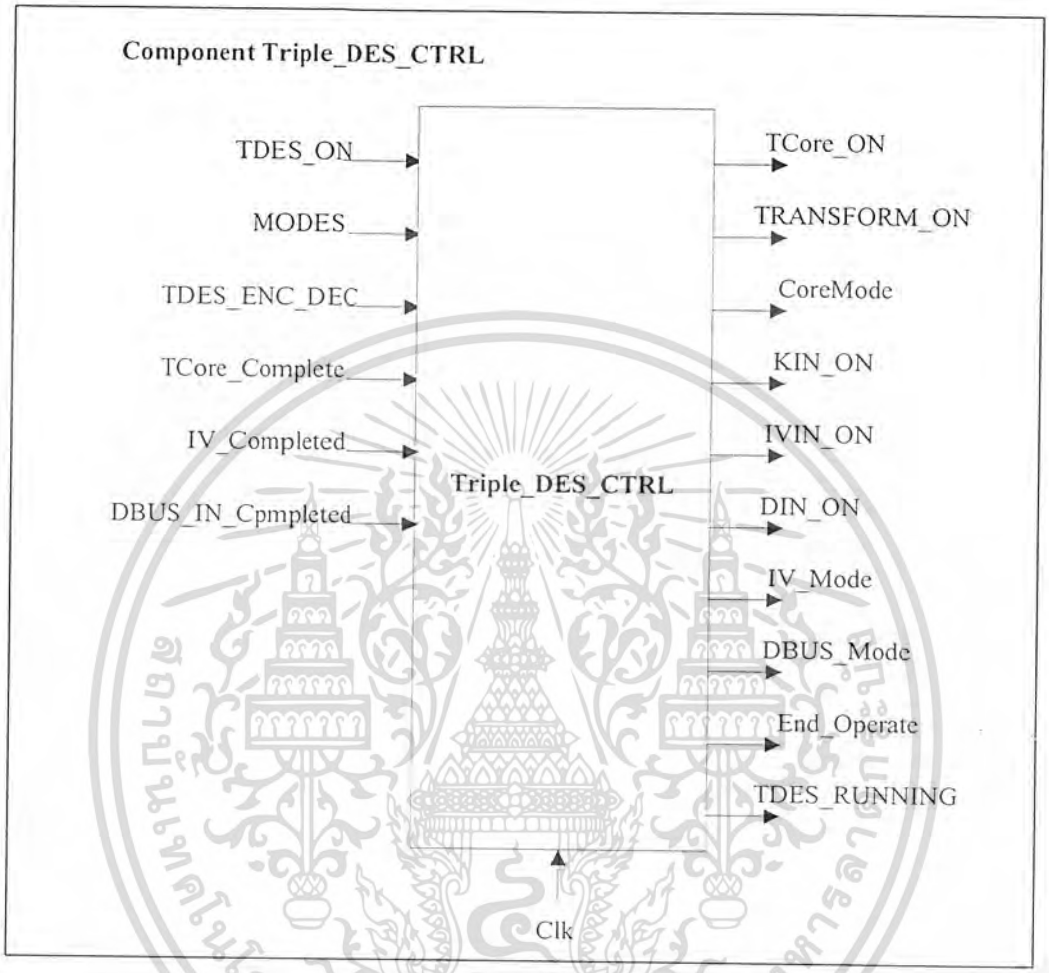
คำอธิบายขาสัญญาณ

- **DIN_ON(IN):** สัญญาณที่ทำหน้าที่เป็นสวิตช์เปิด/ปิดการทำงานของโมดูล DATA_IMPORT โดยให้ '1' คือการเปิดสวิตช์ และ '0' คือการปิดสวิตช์
- **Byte_IN (IN 7:0):** (ดูในคอมโพเน็นต์ Triple DES ที่ระดับบนสุด)
- **LAST_Byte (IN):** (ดูในคอมโพเน็นต์ Triple DES ที่ระดับบนสุด)
- **DBUS_Mode (IN 2:0):** โหมดในการอ่านข้อมูล 8, 16, 32, 64 โดยบิตซ้ายสุดของ DBUS_Mode เป็นสัญญาณร้องขอข้อมูล
-
- **READ_Byte_ENABLE (IN):** (ดูในคอมโพเน็นต์ Triple DES ที่ระดับบนสุด)
- **READ_Byte (OUT):** (ดูในคอมโพเน็นต์ Triple DES ที่ระดับบนสุด)
- **DBUS_IN (OUT 0:63) :** บัสข้อมูลอินพุตภายในขนาด 64 บิต
- **LAST_Data (OUT):** สัญญาณแจ้งให้ระบบภายในรู้ว่าบล็อกข้อมูลที่ส่งให้เป็นบล็อกสุดท้ายแล้ว
- **DBUS_IN_Completed (OUT):** สัญญาณแจ้งเพื่อให้โมดูลที่ร้องขอข้อมูลรู้ว่าบล็อกข้อมูลมีพร้อมอยู่ที่บัส DBUS_IN แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.3 Triple_DES_CTRL (level 1)

ทำหน้าที่ควบคุมการเปิดสวิตช์การทำงานของโมดูลอื่น ๆ รวมถึงการควบคุมลำดับการทำงานของ Triple DES ทั้งหมด



รูปที่ 5-3 แสดงเอนติตีของ Triple_DES_CTRL

คำอธิบายขาสัญญาณ

- **TDES_ON (IN):** เป็นสวิตช์หลักของ Triple DES
- **Modes (IN):** (ดูในคอมโพเน็นต์ Triple DES ที่ระดับบนสุด)
- **TDES_ENC_DEC (IN):** (ดูในคอมโพเน็นต์ Triple DES ที่ระดับบนสุด)
- **TCore_Completed (IN):** สัญญาณที่มาจากโมดูล TDES_Core เพื่อแจ้งว่าการทำงานของมันเสร็จสิ้นแล้ว
- **Keys_Completed (IN):** สัญญาณที่มาจาก TDES_KEY_READER เพื่อแจ้งว่าการอ่านคีย์เสร็จเรียบร้อยแล้ว

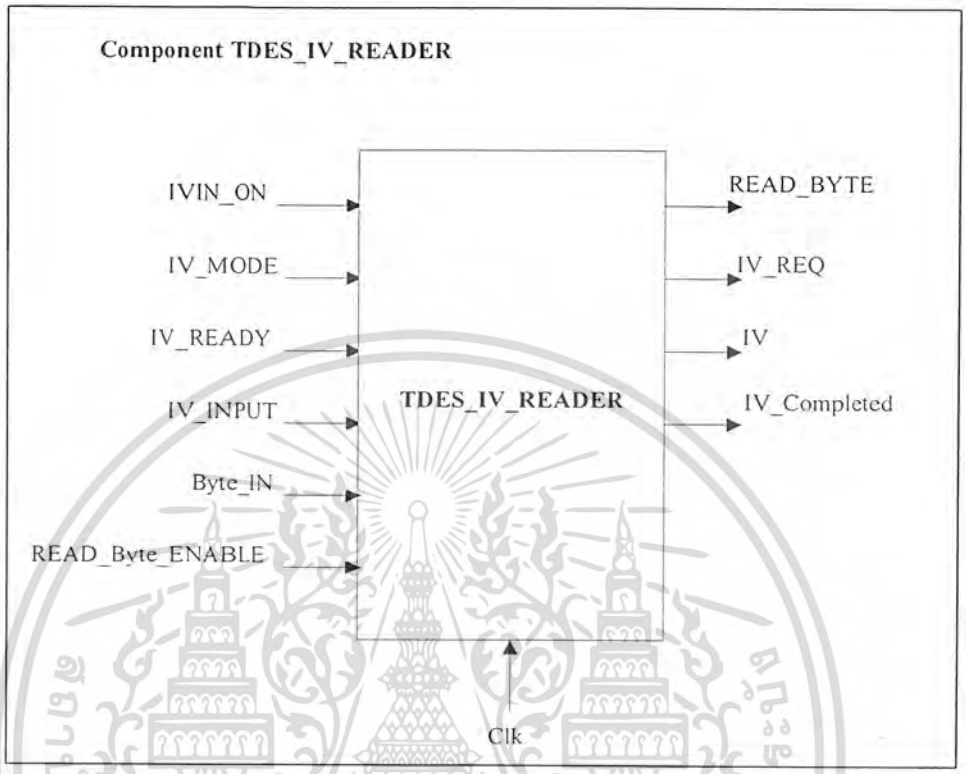
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **IV_Completed (IN):** สัญญาณที่มาจาก TDES_IV_READER เพื่อแจ้งว่าค่า IV ที่ขอไปถูกส่งมาให้แล้ว
- **DBUS_IN_Completed (IN):** สัญญาณที่มาจาก DATA_IMPORT เพื่อแจ้งว่าบล็อกข้อมูลที่ต้องการถูกส่งมาให้แล้ว
- **LAST_Data (IN):** (ดูในคอมโพเน็นต์ DATA_IMPORT)
- **TCore_ON (OUT):** สวิตช์สำหรับเปิดการทำงานให้กับ TDES_Core
- **TRANSFORM_ON (OUT):** สวิตช์สั่งให้ TCore_ON ทำการเข้าและถอดรหัสบล็อกข้อมูลที่ได้มา
- **CoreMode(OUT 2:0):** โหมดการทำงานของ Triple_DES (ดู Modes)
- **KIN_ON (OUT):** สวิตช์เปิดการทำงานให้ TDES_KEY_READER ทำการอ่านคีย์เข้ามาจำนวน 192 บิต (หรือ 24 ไบต์)
- **IVIN_ON (OUT):** สวิตช์เปิดการทำงานให้ TDES_IV_READER ทำงาน
- **DIN_ON (OUT):** สวิตช์เปิดการทำงานให้ DATA_IMPORT ทำการอ่านข้อมูลมาเก็บไว้
- **IV_Mode (OUT):** โหมดในการอ่านค่า IV ถ้าเป็นการเข้ารหัสลับ ค่า IV จะนำมาจาก RNG หากเป็นการถอดรหัสลับ ค่า IV จะถูกอ่านมาจากบล็อกข้อมูลอินพุต
- **DBUS_Mode (OUT):** (ดูใน DATA_IMPORT)
- **End_Operate (OUT):** (ดูใน Triple DES ที่ระดับบนสุด)
- **TDES_RUNNING (OUT):** (ดูใน Triple DES ที่ระดับบนสุด)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.4 TDES_IV_READER (level 1)

ทำหน้าที่ในการอ่านค่า IV ส่งให้กับ TDES_Core โดยมีการควบคุมการทำงานจาก Triple_DES_CTRL



รูปที่ 5-4 แสดงเอนติตี TDES_IV_READER

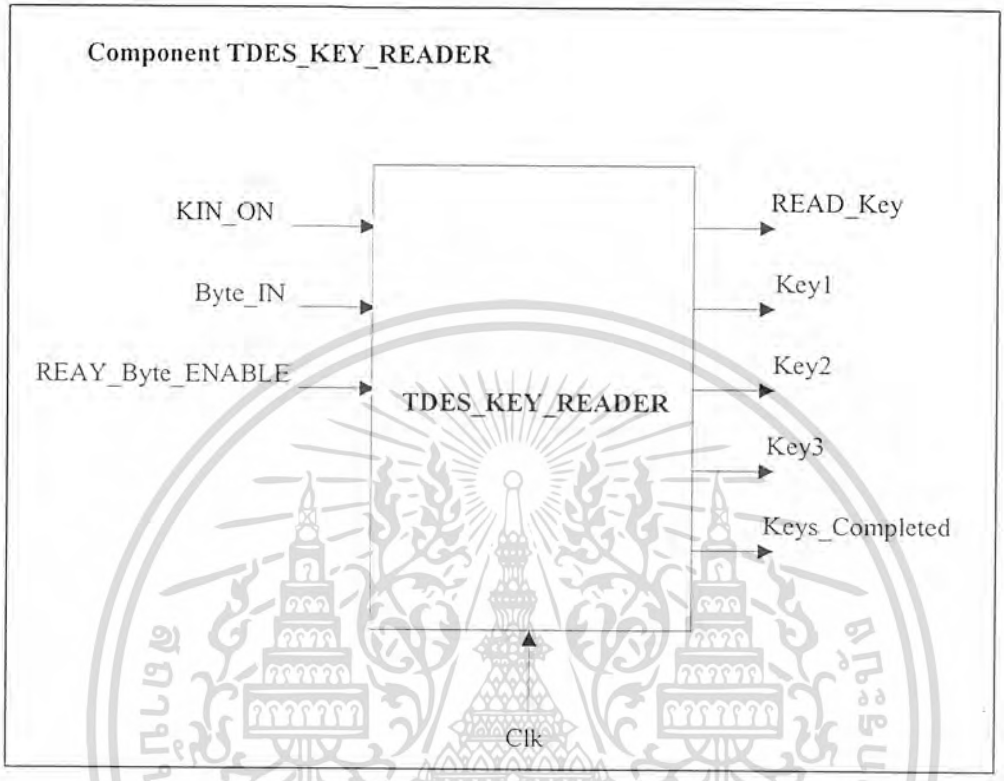
คำอธิบายขาสัญญาณ

- **IVIN_ON (IN):** สวิตช์เปิด/ปิดการทำงาน ควบคุมโดย Triple_DES_CTRL
- **IV_Mode (IN):** (ดู Triple_DES_CTRL)
- **IV_READY (IN):** สัญญาณที่ส่งมาจาก RNG เพื่อบอกว่าค่า IV_INPUT ถูกส่งเข้ามาแล้ว
- **IV_INPUT (IN 0:63):** บัสของค่า IV ที่ส่งมาจาก RNG
- **Byte_IN (IN 7:0):** บัสข้อมูลเพื่ออ่านค่า IV จากข้อมูลอินพุต
- **READ_Byte_ENABLE (IN):** (ดู Triple DES ที่ระดับบนสุด)
- **READ_BYTE (OUT):** (ดู Triple DES ที่ระดับบนสุด)
- **IV_REQ (OUT):** สัญญาณร้องไปยัง RNG เพื่อขอค่า IV เข้ามาที่บัส IV_INPUT
- **IV (OUT 0:63):** บัสของค่า IV ที่อ่านเข้ามาเพื่อส่งให้กับ TDES_Core
- **IV_Completed (OUT):** สัญญาณแจ้งให้ TDES_Core รู้ว่าได้ส่งค่า IV ไปให้แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.5 TDES_KEY_READER (level 1)

ทำหน้าที่ในการอ่านคีย์ทั้งหมด 3 ชุด (192 บิต) เพื่อส่งให้กับ TDES_KEY_SCHEDULER สำหรับการสร้างคีย์ย่อย 16 ชุด



รูปที่ 5-5 แสดงเอนติตี TDES_KEY_READER

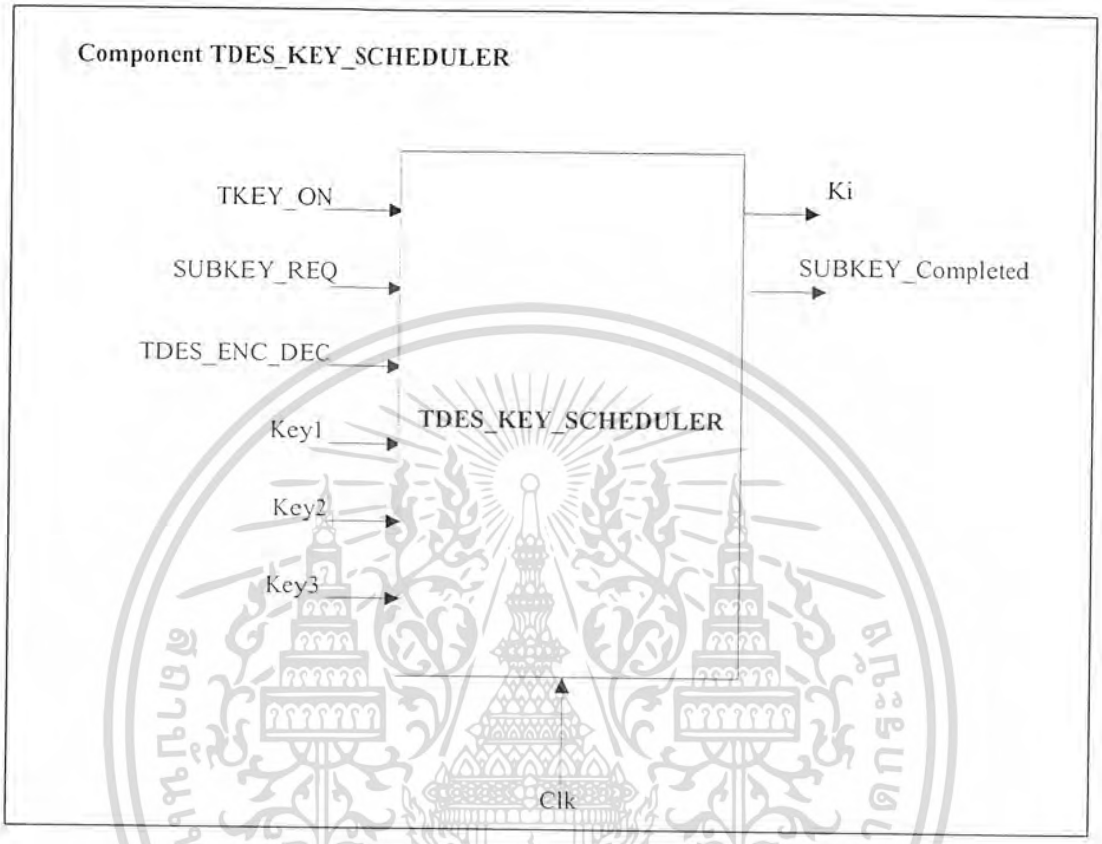
คำอธิบายขาสัญญาณ

- **KIN_ON (IN):** สวิตช์เปิด/ปิด การทำงานของ TDES_KEY_READER
- **Byte_IN (IN):** (ดู Triple DES ที่ระดับบนสุด)
- **READ_Byte_ENABLE (IN):** (ดู Triple DES ที่ระดับบนสุด)
- **READ_Key (OUT):** สัญญาณร้องขอการอ่านคีย์ 3 ชุด (ดู Triple DES ที่ระดับบนสุด)
- **Key1 (OUT 0:63):** บัสข้อมูลของคีย์ชุดที่ 1 ส่งไปยัง TDES_KEY_SCHEDULER
- **Key2 (OUT 0:63):** บัสข้อมูลของคีย์ชุดที่ 2 ส่งไปยัง TDES_KEY_SCHEDULER
- **Key3 (OUT 0:63):** บัสข้อมูลของคีย์ชุดที่ 3 ส่งไปยัง TDES_KEY_SCHEDULER
- **Keys_Completed (OUT):** สัญญาณเพื่อบอกให้รู้ว่าการอ่านคีย์ทั้ง 3 ชุด เสร็จสิ้นแล้ว และสามารถนำคีย์ทั้ง 3 ชุดไปใช้ได้โดยอ่านค่าจากบัส Key1, Key2, และ Key3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.6 TDES_KEY_SCHEDULER (level 1)

ทำหน้าที่สร้างคีย์ย่อย 16 ชุด จากคีย์ชุดที่ 1, 2 และ 3 เพื่อใช้ในกระบวนการเข้าและถอดรหัสลับ มันจะเริ่มทำงานเมื่อการอ่านคีย์ของ TDES_KEY_READER ทำเสร็จสมบูรณ์แล้ว



รูปที่ 5-6 แสดงเอนติตี TDES_KEY_SCHEDULER

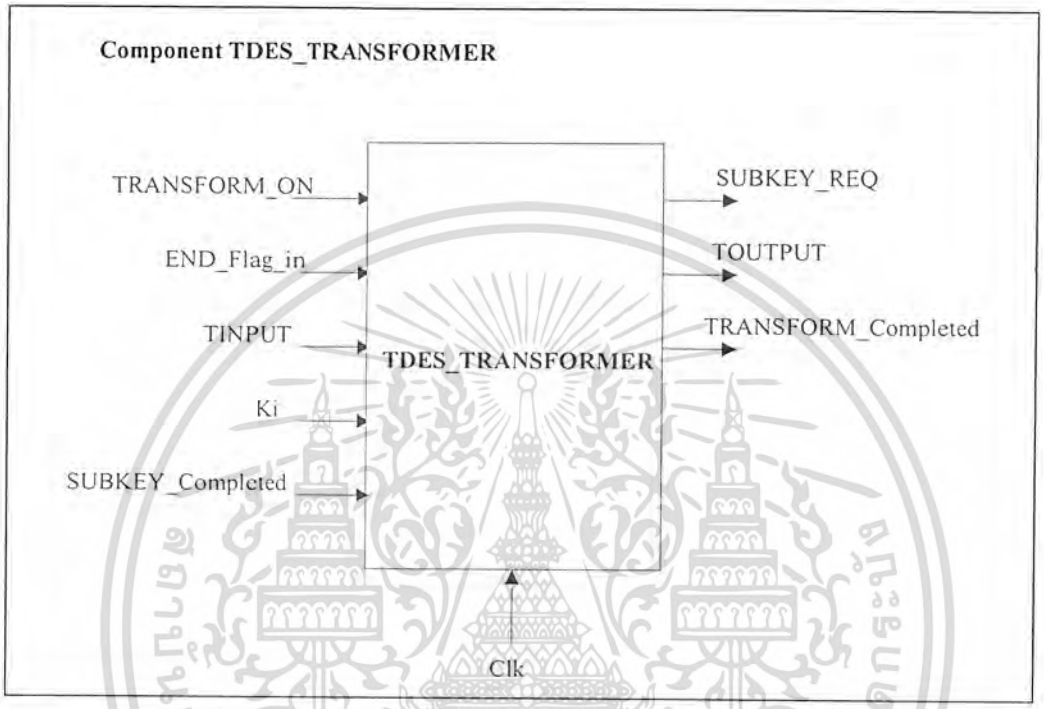
คำอธิบายขาสัญญาณ

- **TKEY_ON (IN):** สวิตช์เปิด/ปิดการทำงาน ซึ่งรับสัญญาณการเปิด/ปิดสวิตช์จากขา Keys_Completed ของ TDES_KEY_READER เพื่อให้ TDES_KEY_SCHEDULER ทำงานทันทีหลังจากอ่านคีย์เสร็จ
- **SUBKEY_REQ (IN):** สัญญาณการร้องขอคีย์ย่อยจาก TDES_TRANSFORMER เพื่อใช้ในกระบวนการเข้าและถอดรหัสในแต่ละรอบ
- **TDES_ENC_DEC (IN):** สัญญาณที่บอกว่าเป็นการเข้ารหัสหรือถอดรหัส (ดู Triple DES ที่ระดับบนสุด)
- **Key1, Key2, Key3 (IN 0:63):** คีย์ 3 ชุด (ดู TDES_KEY_READER)
- **Ki (OUT 0:47):** บัสขนาด 48 บิตสำหรับส่งคีย์ย่อยไปให้ TDES_TRANSFORMER
- **SUBKEY_Completed (OUT):** สัญญาณที่แจ้งให้รู้ว่าได้ส่งคีย์ย่อยไปให้แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.7 TDES_TRANSFORMER (level 2)

เป็นคอมโพเน็นต์ของ TDES_Core ทำหน้าที่หลักในการเข้าถอดรหัสบล็อกของข้อมูลขนาด 64 บิต โดย TDES_Core จะสั่งให้ TDES_TRANSFORMER ทำงาน 3 รอบคือ เข้ารหัส ถอดรหัส และเข้ารหัส หรือ ถอดรหัส เข้ารหัส และถอดรหัส ดังนั้นจะมีการเปิด/ปิด สวิตช์การทำงานของ TDES_TRANSFORMER 3 รอบเพื่อให้ครบกระบวนการเข้าและถอดรหัสของ Triple DES



รูปที่ 5-7 แสดงเอนติตี TDES_TRANSFORMER

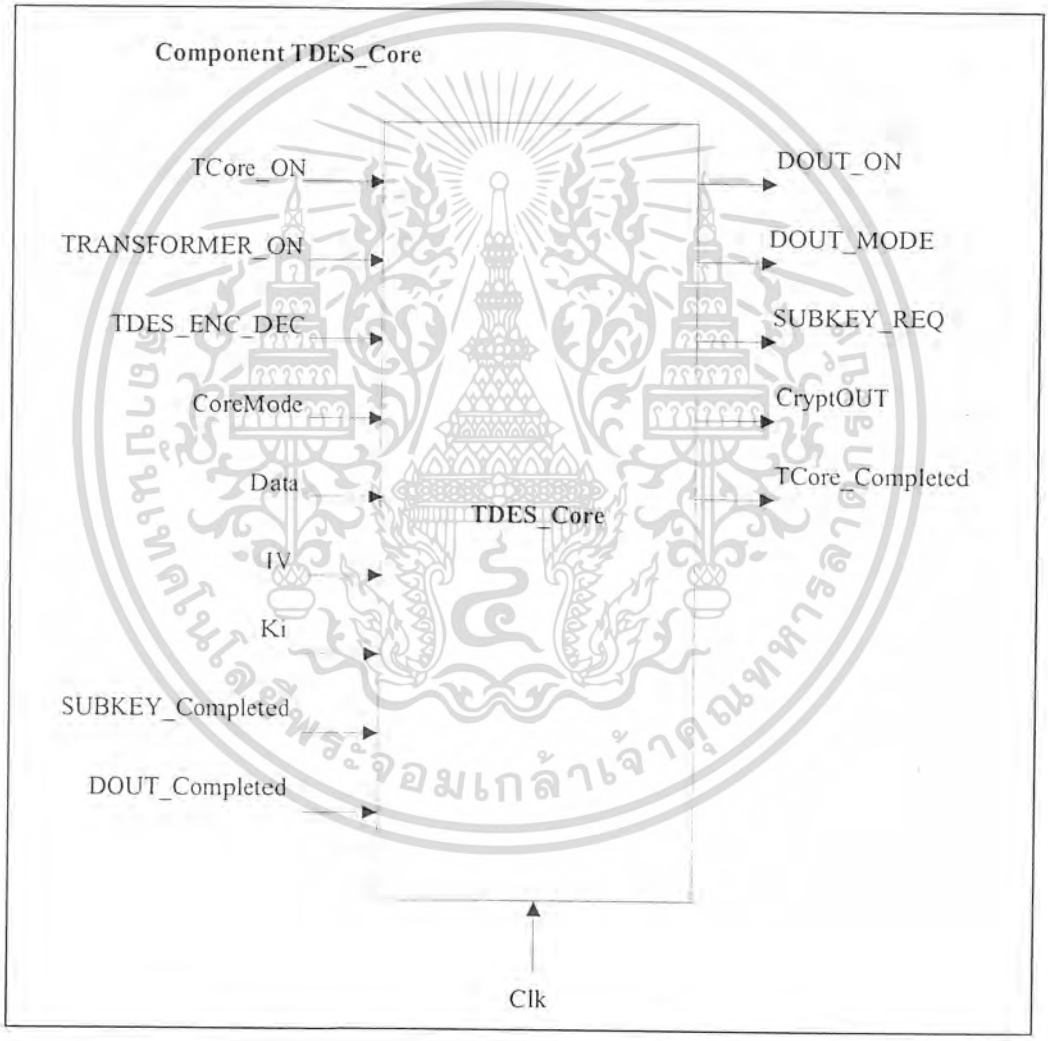
คำอธิบายขาสัญญาณ

- **TRANSFORM_ON (IN):** สวิตช์เปิด/ปิดการทำงานของ TDES_TRANSFORMER
- **ENC_Flag_in (IN):** เป็นสัญญาณบอกให้รู้ว่าการทำงานในรอบนี้เป็นการเข้าหรือถอดรหัส
- **TINPUT (IN 0:63):** บล็อกข้อมูลที่จะนำมาเข้าและถอดรหัส ซึ่งผ่านการทำ Initial Permutation มาแล้ว
- **Ki (IN 0:47):** คีย์ย่อยที่ขอมาจาก TDES_KEY_SCHEDULER
- **SUBKEY_Completed (IN):** สัญญาณที่ TDES_KEY_SCHEDULER ส่งมาแจ้งว่าส่ง Ki มาให้แล้ว
- **SUBKEY_REQ (OUT):** สัญญาณร้องขอไปยัง TDES_KEY_SCHEDULER เพื่อขอคีย์ย่อย
- **TOUTPUT (OUT 0:63):** บัสข้อมูลเพื่อส่งผลลัพธ์จากการเข้าและถอดรหัสในรอบนี้
- **TRANSFORM_Completed (OUT):** สัญญาณแจ้งให้ TDES_Core รู้ว่าการเข้าและถอดรหัสในรอบนี้เสร็จเรียบร้อยและส่งผลลัพธ์ให้แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.8 TDES_Core (level 1)

ทำหน้าที่หลักในการเข้ารหัสบล็อกข้อมูล การทำงานของ TDES_Core จะเริ่มขึ้นก็ต่อเมื่อข้อมูลที่จำเป็นสำหรับการเข้าและถอดรหัสมีพร้อมอยู่ทั้งหมดแล้วโดย Triple_DES_CTRL จะทำหน้าที่สั่งให้ DATA_IMPORT, TDES_KEY_READER, TDES_IV_READER เตรียมข้อมูลทั้งหมดให้พร้อม ส่งมาให้ TDES_Core เมื่อข้อมูลทุกอย่างพร้อมแล้ว Triple_DES_CTRL จะเปิดสวิตช์สั่งให้ TDES_Core ทำงาน แล้วแจ้งให้ TDES_CTRL รับทราบเมื่อเสร็จแล้ว



รูปที่ 5-8 แสดงเอนติตี TDES_Core

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

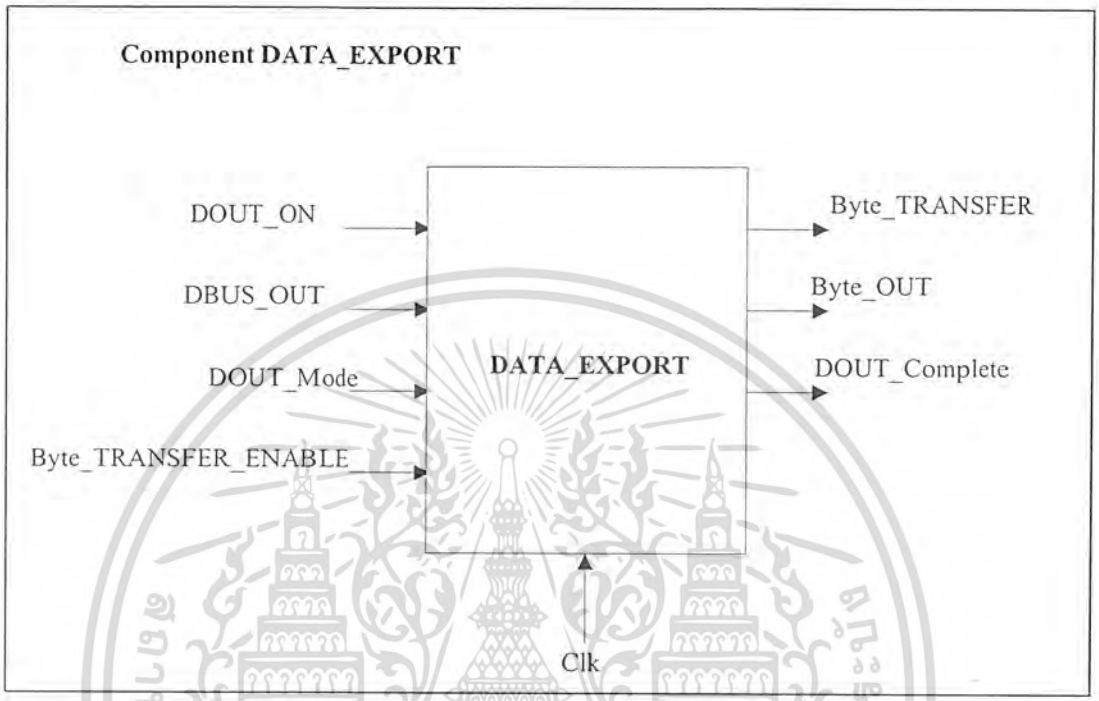
คำอธิบายขาสัญญาณ

- **TCore_ON (IN):** สวิตช์เปิด/ปิดการทำงาน ควบคุมโดย Triple_DES_CTRL
- **TRANSFORM_ON (IN):** สวิตช์สั่งให้ทำการเข้าและถอดรหัสบล็อกข้อมูล
- **TDES_ENC_DEC (IN):** สัญญาณบอกว่าเป็นการเข้ารหัสหรือถอดรหัส
- **CoreMode (IN 2:0):** สัญญาณบอกโหมดการทำงานของ Triple DES (ดู Modes)
- **Data (IN 0:63):** บัสข้อมูลขนาด 64 บิต สำหรับส่งข้อมูลเข้ามาเพื่อเข้าและถอดรหัส จาก DATA_IMPORT
- **IV (IN 0:63):** บัสขนาด 64 บิตสำหรับค่า IV เพื่อทำการเข้าและถอดรหัสในโหมดที่ต้องใช้ค่า IV
- **Ki (IN 0:47):** บัสขนาด 48 บิตสำหรับคีย์ย่อยทั้ง 16 ชุด (ดู TDES_KEY_SCHEDULER)
- **SUBKEY_Completed (IN):** (ดูใน TDES_TRANSFORMER)
- **DOUT_Completed (IN):** สัญญาณที่ DATA_EXPORT ส่งมาบอกว่าส่งบล็อกข้อมูลของผลลัพธ์ออกไปเสร็จเรียบร้อยแล้ว
- **DOUT_ON (OUT):** สัญญาณเพื่อเปิดให้ DATA_EXPORT ทำงานจนกว่าบล็อกของผลลัพธ์จะถูกส่งออกไปจนหมด
- **DOUT_Mode (OUT 2:0):** โหมดที่บอกจำนวนของข้อมูลที่จะส่งออกไปว่าเป็น 8 บิต 16 บิต 32 บิต หรือ 64 บิต ซึ่งขึ้นอยู่กับโหมดการทำงานของ Triple DES ที่ผู้ใช้เลือก
- **SUBKEY_REQ (OUT):** สัญญาณร้องขอไปยัง TDES_KEY_SCHEDULER เพื่อขอคีย์ย่อย
- **CryptOUT (OUT):** เป็นบัสขนาด 64 บิตสำหรับส่งเอาต์พุตไปให้ DATA_EXPORT ส่งถ่ายผลลัพธ์ออกไป
- **TCore_Completed (OUT):** สัญญาณแจ้งให้ Triple_DES_CTRL ทราบว่างานที่ Triple_DES_CTRL สั่งทำเสร็จแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.9 DATA_EXPORT

ทำหน้าที่หลักในการส่งถ่ายข้อมูลผลลัพธ์ออกไปให้กับระบบภายนอก โดยรับคำสั่งจาก TDES_Core



รูปที่ 5-9 แสดงเอนติตี DATA_EXPORT

คำอธิบายขาสัญญาณ

- **DOUT_ON (IN):** สวิตช์เปิดให้ DATA_EXPORT ทำงาน
- **DBUS_OUT (IN 0:63):** บัสขนาด 64 บิตของบล็อกข้อมูลที่จะส่งออกไป
- **DOUT_Mode (IN 2:0):** โหมดบอกจำนวนข้อมูลที่จะส่งถ่ายออกไป (ดู TDES_Core)
- **Byte_TRANSFER_ENABLE (IN):** (ดู Triple_DES ที่ระดับบนสุด)
- **Byte_TRANSFER (OUT):** (ดู Triple_DES ที่ระดับบนสุด)
- **Byte_OUT (OUT 7:0):** (ดู Triple_DES ที่ระดับบนสุด)
- **DOUT_Completed (OUT):** สัญญาณแจ้งให้ TDES_Core ทราบว่าการส่งถ่ายข้อมูลที่ TDES_Core สั่งมานั้นเสร็จเรียบร้อยแล้ว

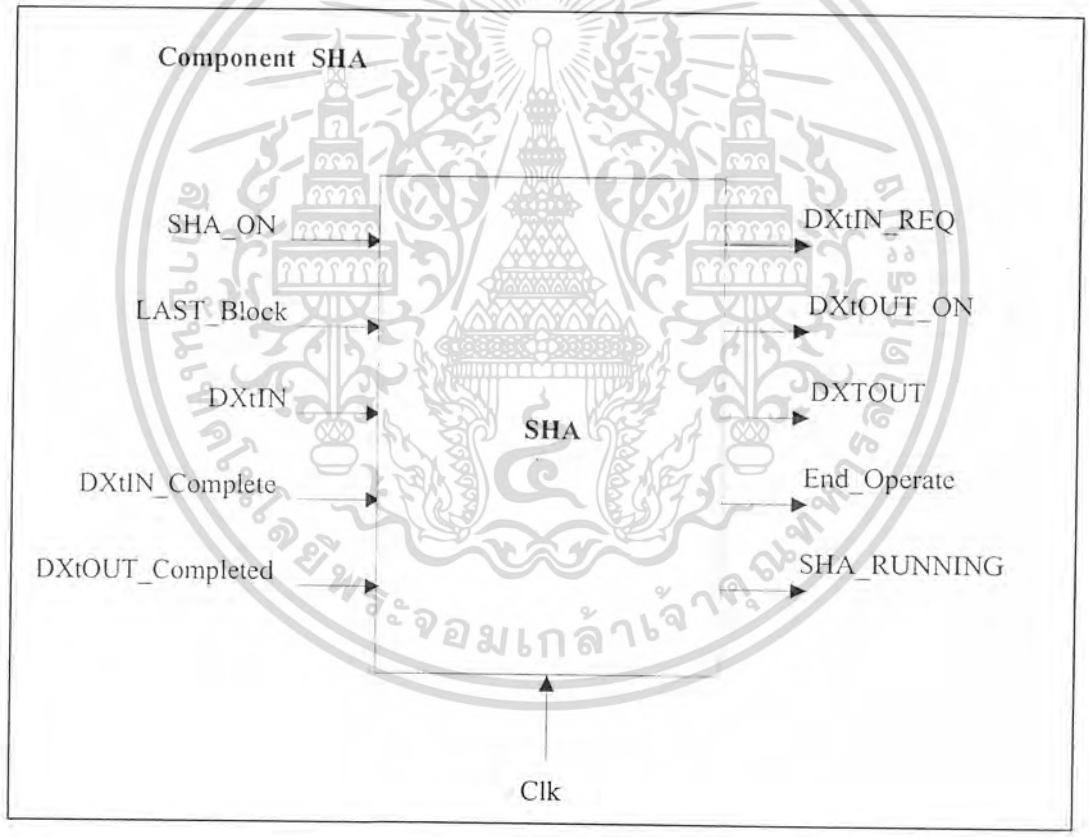
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2 Secure Hash Algorithm

5.2.1 SHA

SHA เป็นคอมโพเน็นต์สำหรับการสร้างค่าแฮชหรือเมสเสจไดเจสต์โดยมีโมดูลย่อยทั้งหมด 5 ตัว คือ

1. SHA_EXECUTIVE
2. SHA_WORD_INITIALIZER
3. SHA_HASH_CORE
4. SHA_Ft
5. SHA_Kt



รูปที่ 5-10 แสดงเอนติตี SHA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

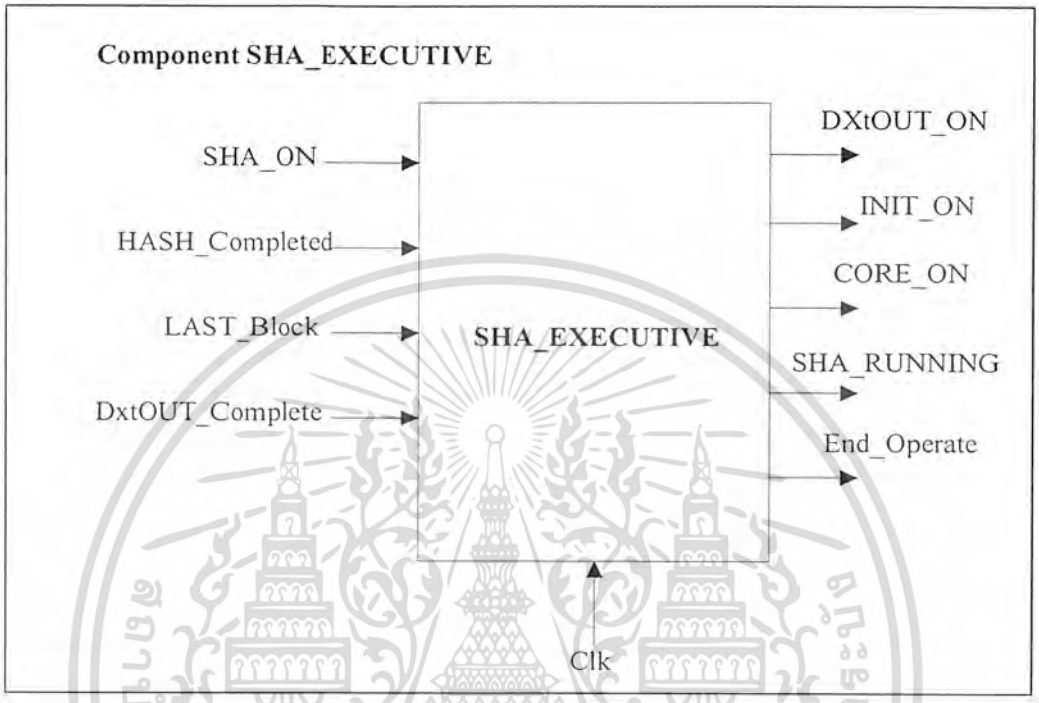
คำอธิบายขาสัญญาณ

- **SHA_ON (IN):** สวิตช์เปิด/ปิดการทำงานหลักของคอมโพเน็นต์ SHA
- **LAST_Block (IN):** สัญญาณแจ้งให้ทราบว่าเป็นบล็อกสุดท้ายสำหรับการคำนวณ
- **DXtIN (IN 0:511):** เป็นบัสขนาด 512 บิตสำหรับบล็อกข้อมูลที่จะหาค่าเมสเสจไดเจสต์ ถูกส่งมาจากคอมโพเน็นต์ HASH_IMPORT (เป็นคอมโพเน็นต์ที่ถูกแชร์ใช้งานระหว่าง SHA กับ MD5)
- **DXtIN_Completed (IN):** สัญญาณที่ HASH_IMPORT แจ้งให้ทราบว่าได้ส่งบล็อกข้อมูล 512 บิตมาให้แล้ว
- **DXtOUT_Completed (IN):** สัญญาณที่ HASH_EXPORT แจ้งให้ทราบว่าบล็อกข้อมูลของผลลัพธ์ได้ถูกส่งถ่ายออกไปทั้งหมดแล้ว (HASH_EXPORT ถูกแชร์ใช้งานระหว่าง SHA และ MD5 เช่นเดียวกัน)
- **DXtIN_REQ (OUT):** สัญญาณส่งไปยัง HASH_IMPORT เพื่อร้องขอบล็อกข้อมูลขนาด 512 บิต
- **DXtOUT_ON (OUT):** สวิตช์เปิด/ปิดการทำงานให้กับ HASH_EXPORT เพื่อส่งถ่ายผลลัพธ์ออกไป
- **DXtOUT (OUT):** บัสขนาด 160 บิตสำหรับส่งถ่ายผลลัพธ์ (เมสเสจไดเจสต์) ออกไปโดยให้ HASH_EXPORT เป็นตัวจัดการ
- **End_Operate (OUT):** สัญญาณบอกให้ระบบภายนอกทราบว่า SHA ทำงานเสร็จแล้ว และส่งถ่ายผลลัพธ์ออกไปแล้ว
- **SHA_RUNNING (OUT):** สัญญาณแสดงสถานะว่า SHA กำลังทำงานอยู่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2.2 SHA_EXECUTIVE (level 1)

ทำหน้าที่จัดการลำดับการทำงานของ SHA ทั้งหมดและเปิดสวิตช์การทำงานให้กับ SHA_WORD_INITIALIZER และ SHA_HASH_CORE ซึ่งเป็น 2 ส่วนหลักที่จัดการเกี่ยวกับการอ่านข้อมูลเข้ามาและการประมวลผลข้อมูล



รูปที่ 5-11 แสดงเอนติตี SHA_EXECUTIVE

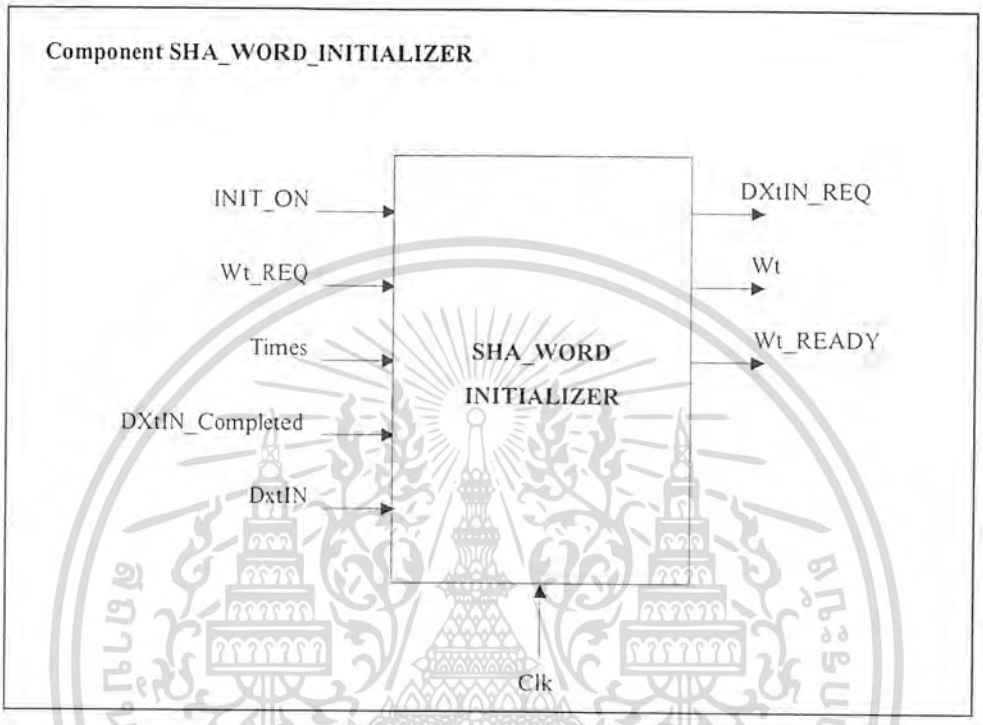
คำอธิบายขาสัญญาณ

- **SHA_ON (IN):** สวิตช์เปิดการทำงานหลักของ SHA โดย SHA_EXECUTIVE เป็นผู้ดูแล
- **HASH_Completed (IN):** สัญญาณที่ SHA_HASH_CORE แจ้งมาว่าทำการคำนวณสำหรับ 1 บล็อกข้อมูลเสร็จแล้ว
- **LAST_Block (IN):** (ดู SHA ที่ระดับบนสุด)
- **DXtOUT_Completed (IN):** (ดู SHA ที่ระดับบนสุด)
- **DXtOUT_ON (OUT):** (ดู SHA ที่ระดับบนสุด)
- **INIT_ON (OUT):** สัญญาณเปิด/ปิดสวิตช์การทำงานของ SHA_WORD_INITIALIZER
- **CORE_ON (OUT):** สัญญาณเปิด/ปิดสวิตช์การทำงานของ SHA_HASH_CORE
- **End_Operate (OUT):** (ดู SHA ที่ระดับบนสุด)
- **SHA_RUNNING (OUT):** (ดู SHA ที่ระดับบนสุด)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2.3 SHA_WORD_INITIALIZER (level 1)

ทำหน้าที่หลักในการอ่านบล็อกข้อมูล 512 บิต (16 เวิร์ด ๆ ละ 32 บิต) เข้ามาเพื่อทำการขยายเวิร์ดออกเป็น 80 เวิร์ด โดย 16 เวิร์ดแรกคือบล็อกข้อมูล 512 บิตที่อ่านเข้ามา ส่วนที่เหลือได้จากกระบวนการขยาย



รูปที่ 5-12 แสดงเอนติตี SHA_WORD_INITIALIZER

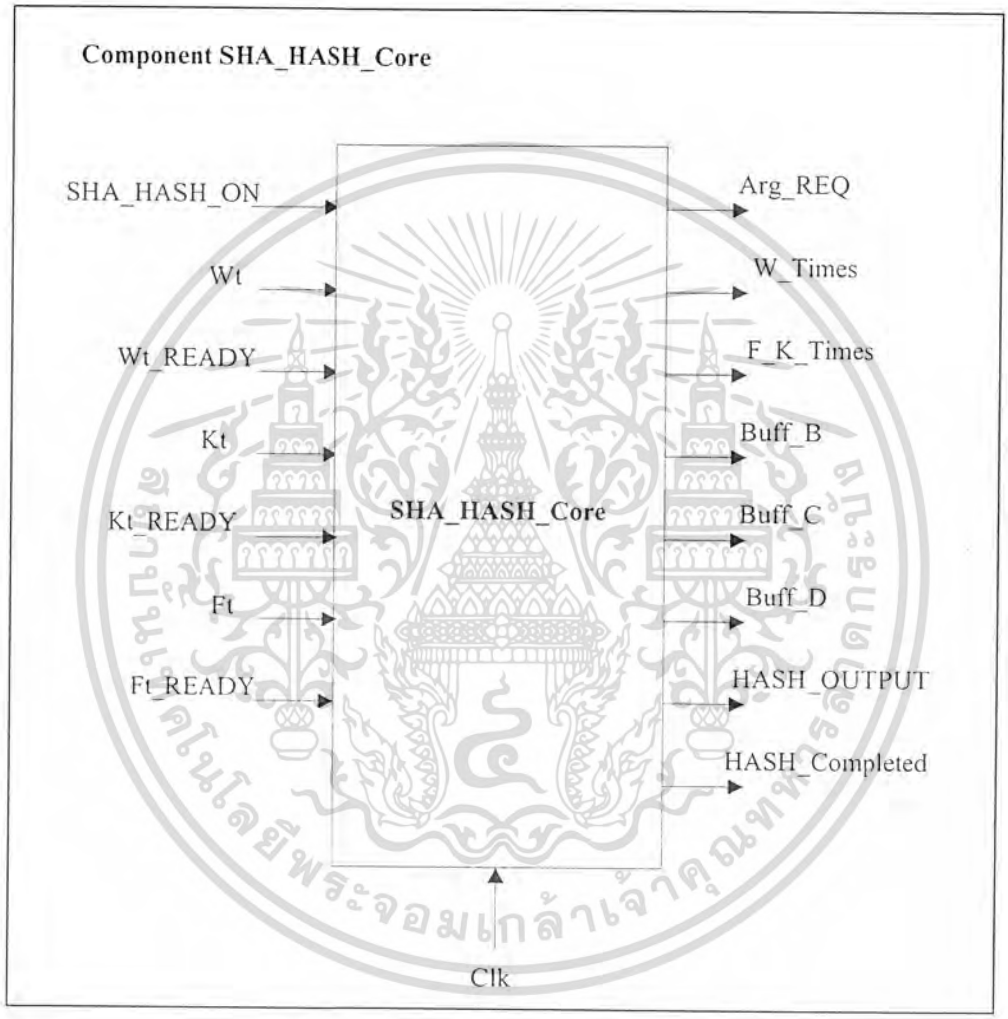
คำอธิบายขาสัญญาณ

- **INIT_ON (IN):** สวิตช์เปิด/ปิดการทำงาน โดย SHA_EXECUTIVE เป็นตัวควบคุม
- **Wt_REQ (IN):** สัญญาณร้องขอ Wt (เวิร์ดที่อยู่ใน 80 เวิร์ดที่ผ่านการขยายมาแล้ว ณ เวลา t) จาก SHA_HASH_CORE
- **Times (IN 3:0):** เวลา t ที่ขอ Wt เพื่อให้ทราบว่าเป็นค่าเวิร์ดในรอบที่เท่าใด
- **DXtIN_Completed (IN):** (ดู SHA ที่ระดับบนสุด)
- **DXtIN (IN 0:511):** (ดู SHA ที่ระดับบนสุด)
- **DXtIN_REQ (OUT):** (ดู SHA ที่ระดับบนสุด)
- **Wt (OUT 31:0):** เวิร์ดที่ขยายแล้วในรอบที่ t ของการร้องขอ
- **Wt_READY (OUT):** สัญญาณแจ้งให้ SHA_HASH_CORE ทราบว่าได้ส่ง Wt ไปให้แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2.4 SHA_HASH_CORE (level 1)

ทำหน้าที่หลักในการคำนวณหาเมสเสจไดเจสต์และส่งผลลัพธ์ออกไปภายนอก เริ่มการทำงานโดยร้องขอค่า Wt, Ft, Kt จาก SHA_WORD_INITIALIZER, SHA_Ft, SHA_Kt ตามลำดับพร้อม ๆ กัน เพื่อนำมาใช้ในการคำนวณ เมื่อค่าทั้ง 3 ค่ามีพร้อมแล้ว ก็ทำการคำนวณ ทำจนกว่าจะครบ 80 รอบ แล้วแจ้งให้ SHA_EXECUTIVE ทราบว่าคำนวณเสร็จแล้ว แล้ว SHA_EXECUTIVE จะตัดสินเองว่าจะส่งผลลัพธ์ออกไปเมื่อใด



รูปที่ 5-13 แสดงเอนติตี SHA_HASH_Core

คำอธิบายขาสัญญาณ

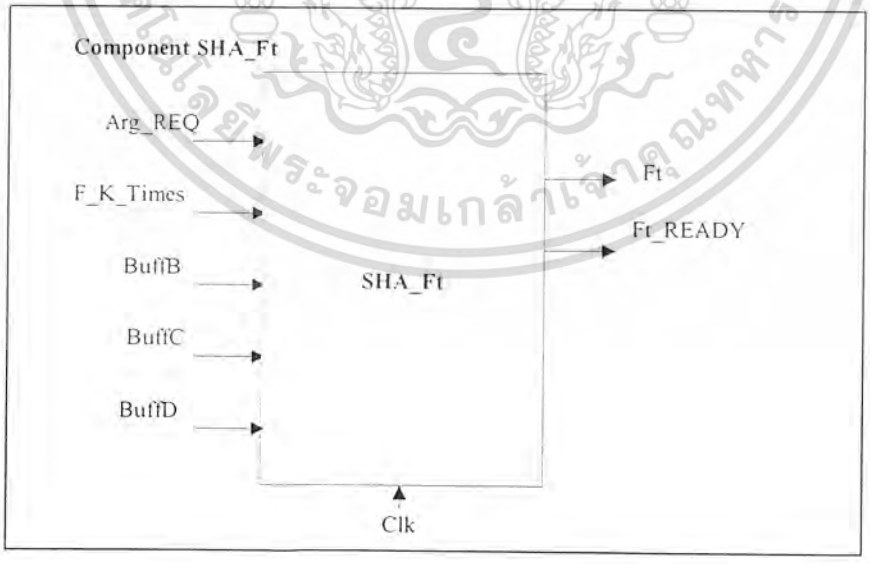
- **SHA_HASH_ON (IN):** สวิตช์เปิดการทำงานเพื่อให้ SHA_HASH_CORE กำหนดค่าเริ่มต้นของเวิร์ดรีจิสเตอร์ A B C D E
- **CORE_ON (IN):** สวิตช์เปิด/ปิดกระบวนการคำนวณหาค่าเมสเสจไดเจสต์ของ SHA
- **Wt (IN 31:0):** เวิร์ดที่ผ่านการขยายมาแล้ว (ดู SHA_WORD_INITIALIZER)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **Wt_READY (IN):** (ดู SHA_WORD_INITIALIZER)
- **Kt (IN 31:0):** ค่าคงที่ K ของรอบการคำนวณที่ t
- **Kt_READY (IN):** สัญญาณที่ SHA_Kt ส่งมาบอกว่าได้ส่ง Kt มาให้เรียบร้อยแล้ว
- **Ft (IN 31:0):** ค่าของฟังก์ชัน *F*(t, B, C, D) จาก SHA_Ft
- **Ft_READY (IN):** สัญญาณจาก SHA_Ft ส่งมาแจ้งว่าได้ส่ง Ft มาให้เรียบร้อยแล้ว
- **Arg_REQ (OUT):** สัญญาณร้องขอ Wt, Ft, Kt โดยร้องขอไปยัง SHA_WORD_INITIALIZER, SHA_Kt, SHA_Ft พร้อม ๆ กัน แล้วรอจนกว่าสัญญาณ Wt_READY, Kt_READY. และ Ft_READY จะมีค่าเป็น '1' ทั้งหมด
- **W_Times (OUT 3:0):** ค่าลำดับของเวิร์ดขยายที่อยู่ในบล็อก (1 ถึง 16)
- **F_K_Times (OUT 4:0):** แสดงช่วงของรอบการคำนวณว่าเป็นช่วงที่เท่าใด
- **Buff_B, Buff_C, Buff_D (OUT 31:0):** ค่าของรีจิสเตอร์ B, C, D ของการคำนวณในรอบหนึ่งโดยส่งให้ SHA_Ft เพื่อคำนวณหาค่า *F*(t, B, C, D) ของการคำนวณในแต่ละรอบ
- **HASH_OUTPUT (OUT 0:159):** ผลลัพธ์ของการคำนวณ (เมสเสจไดเจสต์ขนาด 160 บิต)
- **HASH_Completed (OUT):** สัญญาณแจ้งให้ SHA_EXECUTIVE ทราบว่า SHA_HASH_CORE ทำการคำนวณเสร็จทั้ง 80 รอบแล้ว

5.2.5 SHA_Ft (level 1)

ทำหน้าที่คำนวณค่า *F*(t, B, C, D) ส่งให้ SHA_HASH_CORE



รูปที่ 5-14 แสดงเอนติตี SHA_Ft

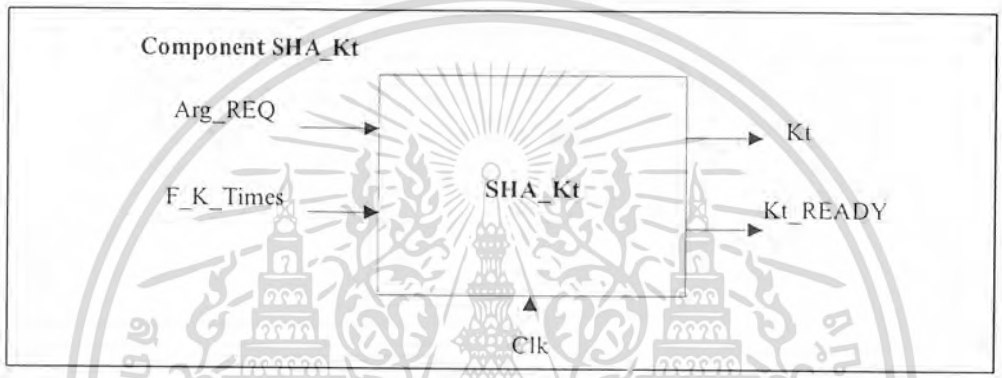
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำอธิบายขาสัญญาณ

- **Arh_REQ (IN):** สัญญาณร้องขอค่า Ft จาก SHA_HASH_CORE
- **F_K_Times (IN 4:0):** บอกช่วงของการคำนวณค่า Ft
- **Buff_B, Buff_C, Buff_D (IN 31:0):** ค่ารีจิสเตอร์ B, C, D สำหรับการหาค่า *F*(t, B, C, D)
- **Ft (OUT 31:0):** ค่า Ft ที่คำนวณแล้วส่งให้กับ SHA_HASH_CORE
- **Ft_READY (OUT):** สัญญาณแจ้ง SHA ว่าส่ง Ft ไปให้แล้ว

5.2.6 SHA_Kt (level 1)

ส่งค่าคงที่ K ที่ช่วงรอบการคำนวณที่ t (K(t))ให้กับ SHA_HASH_CORE



รูปที่ 5-15 แสดงเอนติตี SHA_Kt

คำอธิบายขาสัญญาณ

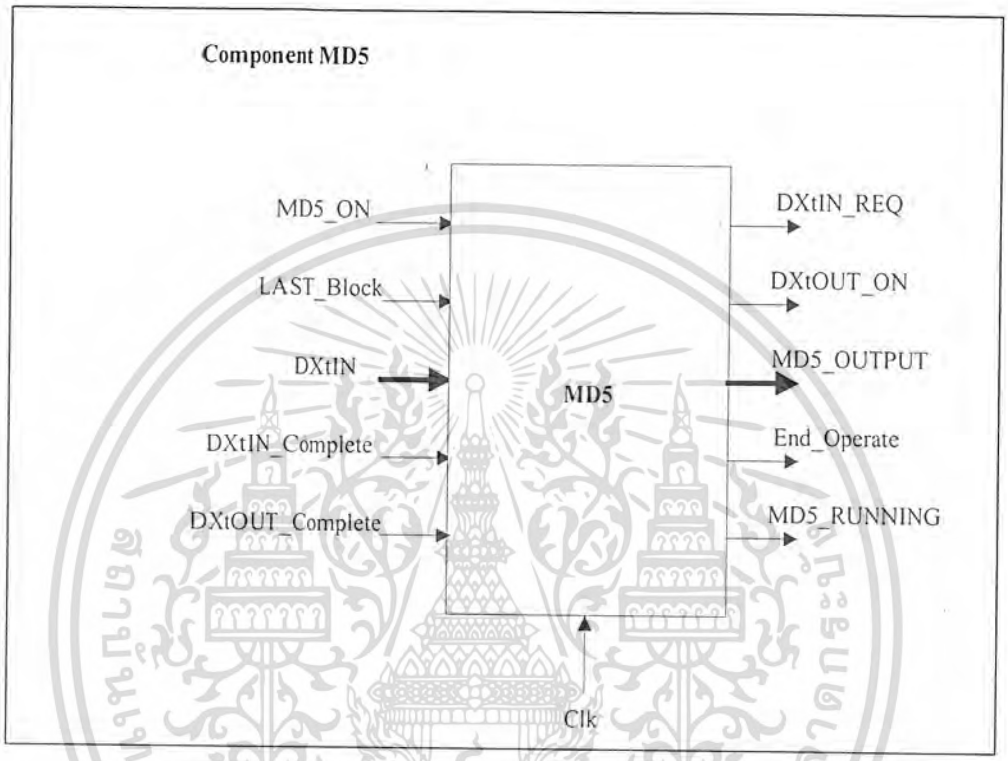
- **Arh_REQ (IN):** สัญญาณร้องขอค่า Ft จาก SHA_HASH_CORE
- **F_K_Times (IN 4:0):** บอกช่วงของการคำนวณค่า Ft
- **Kt (OUT 31:0):** ค่าคงที่ K(t) ในช่วงการคำนวณที่ t
- **Kt_READY (OUT):** สัญญาณแจ้ง SHA ว่าส่ง Kt ไปให้แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3 Message Digest Algorithm

ประกอบด้วยคอมโพเน็นต์การทำงานต่างๆดังนี้

5.3.1 MD5



รูปที่ 5-16 แสดงเอนติตี MD5

หน้าที่การทำงานคือ ทำหน้าที่ในการเชื่อมและส่งสัญญาณข้อมูลให้กับคอมโพเน็นต์ต่างๆของ MD5 และทำหน้าที่ติดต่อการทำงานของ MD5 เข้ากับระบบภายนอก

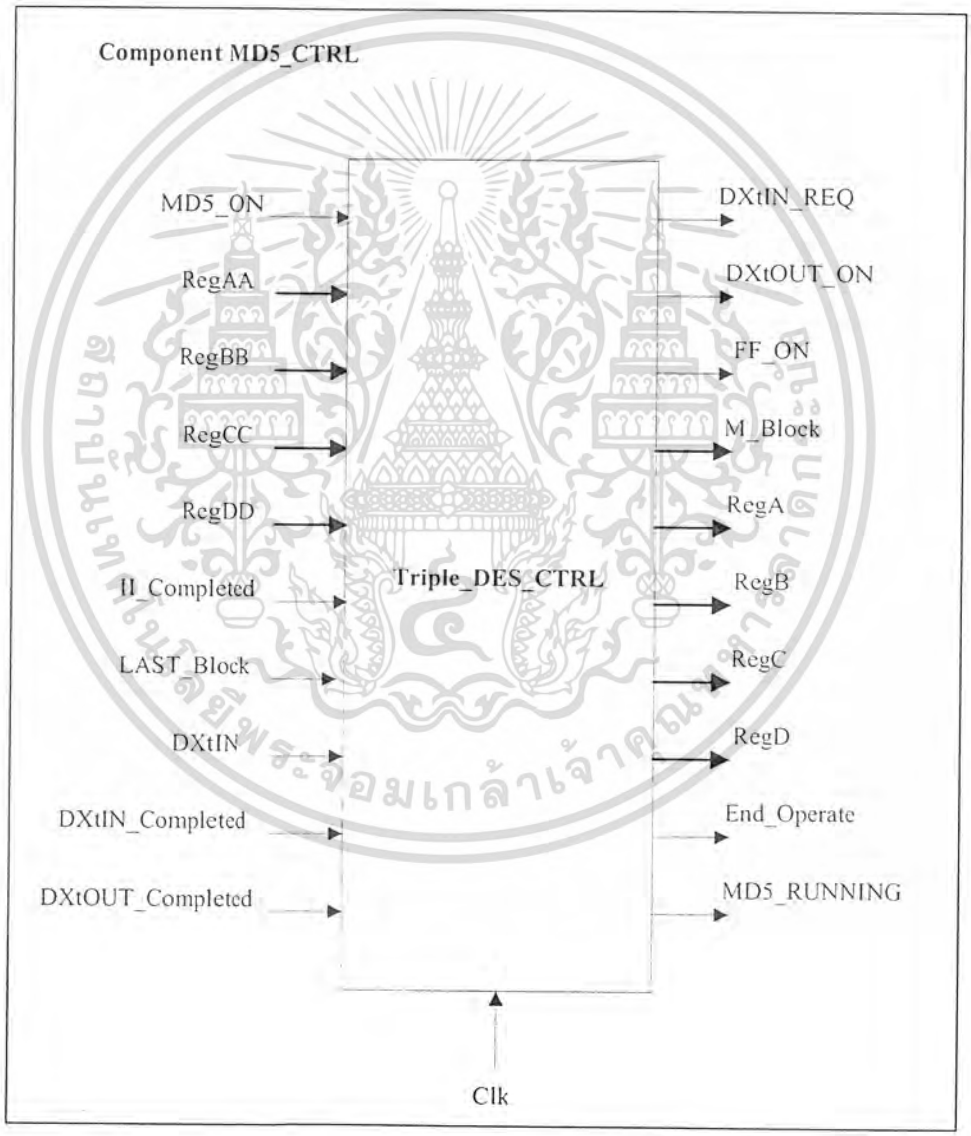
คำอธิบายขาสัญญาณ

- **MD5_ON** เป็นสัญญาณอินพุตที่รับมาจากภายนอกเพื่อสั่งให้อัลกอริทึม MD5 ทำงาน
- **LAST_Block** เป็นสัญญาณที่รับมาจากภายนอกเพื่อส่งต่อให้คอมโพเน็นต์ภายในทราบว่าข้อมูลที่อ่านเข้ามาเป็นข้อมูลบล็กสุดท้าย
- **DxtIN** เป็นบัสข้อมูลขนาด 512 บิตที่รับข้อมูลมาจากภายนอกเข้ามาประมวลผล
- **DxtIN_Complete** เป็นสัญญาณที่รับมาจากภายนอกเพื่อบอกให้คอมโพเนนท์ของ MD5 ทราบว่าข้อมูลขนาด 512 บิตส่งมาเรียบร้อยแล้ว
- **DXtOUT_Complete** เป็นสัญญาณที่รับเข้ามาเพื่อบอกว่าข้อมูลเอาต์พุตได้เขียนลงไฟล์เรียบร้อยแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **DxtIN_REQ** เป็นสัญญาณขนาด 1 บิตที่ส่งออกไปยังภายนอกเพื่อขออ่านข้อมูลขนาด 512 บิตเข้าสู่ระบบ
- **DXtOUT_ON** เป็นสัญญาณที่ส่งออกไปนอกระบบเพื่อขออนุญาตเขียนเอาต์พุตลงไฟล์
- **MD5_OUTPUT** เป็นบัสข้อมูลขนาด 128 บิตที่ส่งเอาต์พุตออกจากระบบ
- **End_Operate** เป็นสัญญาณที่บอกให้ระบบภายนอกทราบว่าเสร็จสิ้นการคำนวณแล้ว
- **MD5_RUNNING** เป็นสัญญาณที่บอกสถานะการทำงานของ MD5

5.3.2 MD5_CTRL



รูปที่ 5-17 แสดงเอนติตี MD5_CTRL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำอธิบายขาสัญญาณ

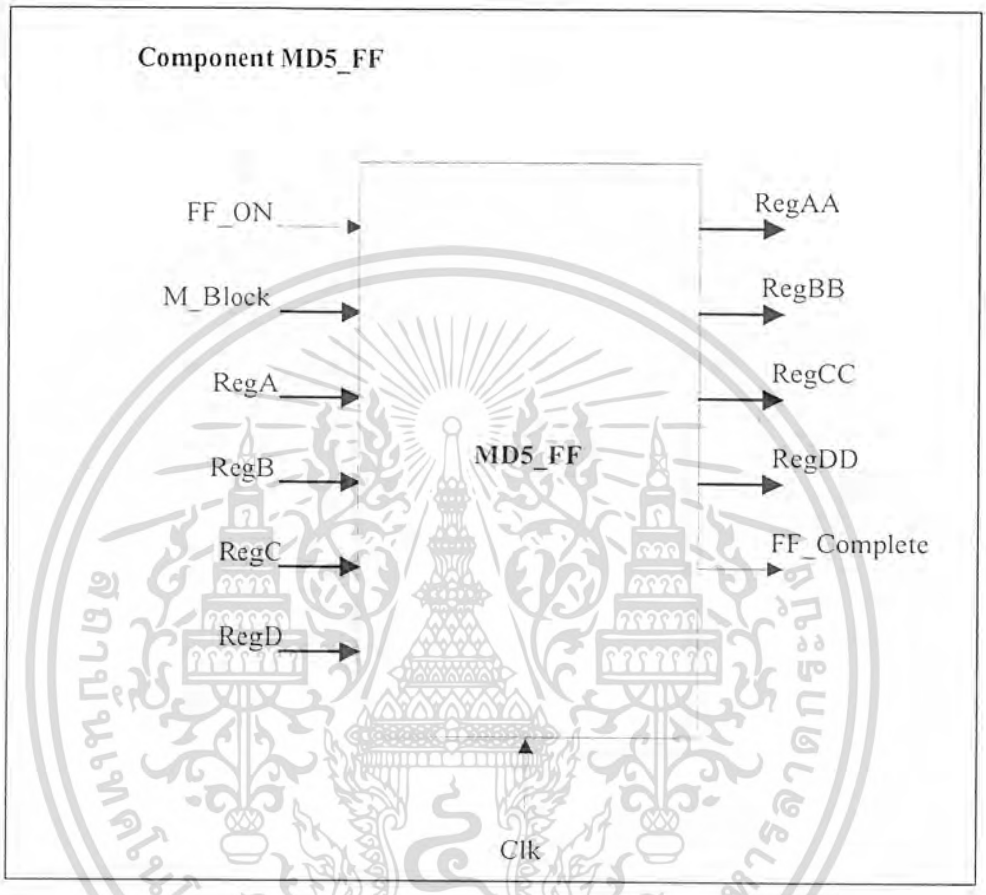
หน้าที่การทำงานคือ ทำหน้าที่ในการเชื่อมการทำงานของคอมโพเน็นต์ทุกตัวใน MD5 เข้าด้วยกัน

- **MD5_ON** เป็นสัญญาณที่รับมาจากคอมโพเน็นต์ MD5 เพื่อเป็นตัวปลุกการทำงานของ อัลกอริทึม MD5
- **RegAA , RegBB , RegCC , RegDD** เป็นค่าขนาด 32 บิตที่รับมาจากการคำนวณในฟังก์ชันช่วย (Auxiliary Function) เพื่อใช้เป็นค่าเริ่มต้นในการคำนวณรอบต่อไป
- **II_Completed** เป็นสัญญาณที่รับเข้ามาจากคอมโพเน็นต์ MD5_II เพื่อทราบว่าการคำนวณรอบที่ 4 ในฟังก์ชันช่วยได้เสร็จสิ้นแล้ว
- **LAST_Block** เป็นสัญญาณที่รับมาจาก MD5 เพื่อไว้ตรวจสอบว่าเป็นบล็อกข้อมูลบล็อกสุดท้ายที่อ่านเข้ามา
- **DxtIN** เป็นข้อมูลบัสขนาด 512 บิตที่อ่านเข้ามาในระบบ
- **DxtIN_Completed** เป็นสัญญาณที่บอกให้ทราบว่าข้อมูลบล็อก 512 บิตอ่านเข้ามาเรียบร้อยแล้ว
- **DxtOUT_Completed** เป็นสัญญาณที่รับมาจากภายนอกผ่านคอมโพเน็นต์ MD5 เพื่อทราบว่าการเขียนเอาต์พุตลงไปได้ทำเสร็จเรียบร้อยแล้ว
- **DxtIN_REQ** เป็นสัญญาณขนาด 1 บิตที่ส่งไปเพื่อขออ่านข้อมูลขนาด 512 บิตเข้าสู่ระบบ
- **DXtOUT_ON** เป็นสัญญาณที่ส่งออกไปนอกระบบเพื่อขออนุญาตเขียนเอาต์พุตข้อมูลลงไฟล์
- **FF_ON** เป็นสัญญาณที่ไปกระตุ้นการคำนวณเริ่มต้นที่คอมโพเน็นต์ MD5_FF
- **M_Block** เป็นบัสข้อมูลขนาด 512 บิตที่ใช้ส่งให้กับการคำนวณในฟังก์ชันช่วยในแต่ละรอบ
- **RegA , RegB , RegC , RegD** เป็นสัญญาณข้อมูลขนาด 32 บิตที่ส่งต่อเป็นค่าเริ่มต้นในการคำนวณในครั้งต่อไป
- **End_Operate** เป็นสัญญาณที่บอกให้ระบบภายนอกทราบว่าเสร็จสิ้นการคำนวณแล้ว
- **MD5_RUNNING** เป็นสัญญาณที่บอกสถานะการทำงานของ MD5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3.3 MD5_FF

หน้าที่การทำงาน คือเป็นฟังก์ชันช่วยที่ใช้ในการคำนวณรอบที่ 1 ของการทำงานโดยจะรับอินพุตเข้ามาเป็นค่าเริ่มต้นการคำนวณจากคอมโพเน็นต์ MD5_CTRL แล้วทำการคำนวณค่ารอบแรกจากนั้นจะส่งค่าที่ได้ให้กับคอมโพเน็นต์ MD5_GG เพื่อทำการคำนวณรอบต่อไป



รูปที่ 5-18 แสดงเอนติตี MD5_FF

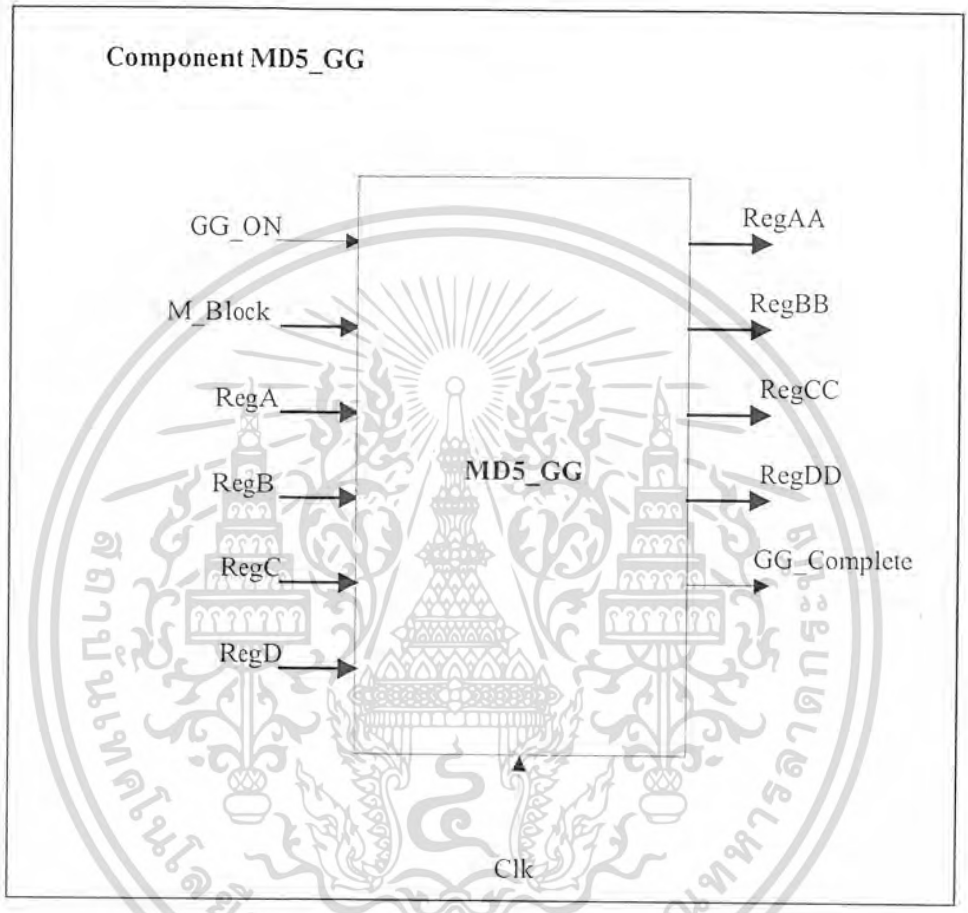
คำอธิบายขาสัญญาณ

- **FF_ON** เป็นสัญญาณที่รับมาจาก MD5_CTRL เป็นสัญญาณที่ปลุกการทำงานของฟังก์ชัน
- **M_Block** เป็นบัสข้อมูลขนาด 512 บิตที่ใช้ส่งให้กับการคำนวณในฟังก์ชันช่วยในแต่ละรอบ
- **RegA , RegB , RegC , RegD** เป็นสัญญาณข้อมูลขนาด 32 บิตที่รับมาจาก MD5_CTRL เพื่อเป็นค่าเริ่มต้นในการคำนวณในรอบที่ 1
- **RegAA , RegBB , RegCC , RegDD** เป็นค่าข้อมูลขนาด 32 บิตที่ผ่านการคำนวณในรอบที่ 1 แล้วส่งต่อให้การคำนวณในรอบต่อไป
- **FF_Completed** เป็นสัญญาณที่บอกให้ทราบว่าการคำนวณในรีจิสเตอร์ขนาด 32 บิตได้เสร็จสิ้นแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3.4 MD5_GG

หน้าที่การทำงาน คือเป็นฟังก์ชันช่วยที่ใช้ในการคำนวณรอบที่ 2 ของการทำงานโดยจะรับอินพุตเข้ามาการคำนวณจากคอมโพเน็นต์ MD5_FF แล้วทำการคำนวณค่ารอบที่ 2 จากนั้นจะส่งค่าที่ได้ให้กับคอมโพเน็นต์ MD5_HH เพื่อทำการคำนวณรอบต่อไป



รูปที่ 5-19 แสดงคอมเอนติตี MD5_GG

คำอธิบายขาสัญญาณ

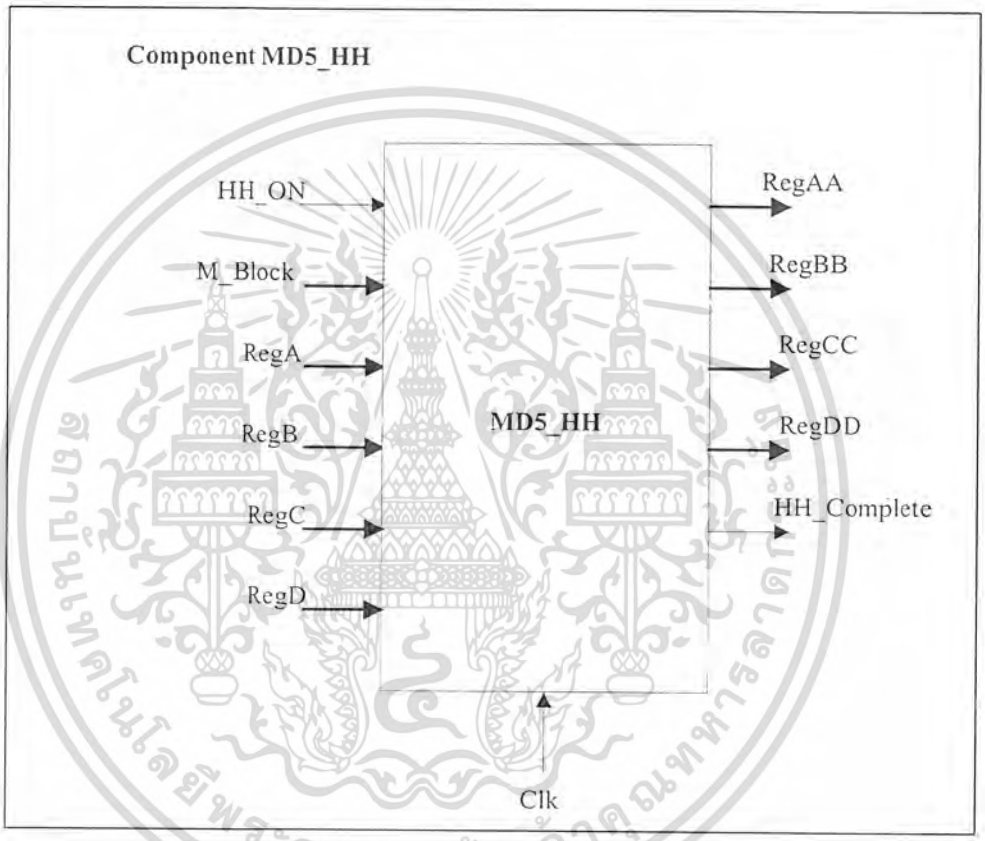
- **GG_ON** เป็นสัญญาณที่รับเข้ามาเพื่อเป็นสัญญาณปลุกการทำงานของฟังก์ชันการคำนวณ
- **M_Block** เป็นบัสข้อมูลขนาด 512 บิตที่ใช้ส่งให้กับการคำนวณในฟังก์ชันช่วยในแต่ละรอบ
- **RegA , RegB , RegC , RegD** เป็นสัญญาณข้อมูลขนาด 32 บิตที่รับมาจากการคำนวณในรอบที่ 1 เพื่อเป็นค่าที่ใช้ในการคำนวณรอบที่ 2
- **RegAA , RegBB , RegCC , RegDD** เป็นค่าข้อมูลขนาด 32 บิตที่ผ่านการคำนวณในรอบที่ 2 แล้วส่งต่อให้การคำนวณในรอบต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **GG_Completed** เป็นสัญญาณที่บอกให้ทราบว่าการคำนวณในรีจิสเตอร์ขนาด 32 บิตได้เสร็จสิ้นแล้ว

5.3.5 MD5_HH

หน้าที่การทำงาน คือเป็นฟังก์ชันช่วยที่ใช้ในการคำนวณรอบที่ 3 ของการทำงานโดยจะรับอินพุตเข้ามาการคำนวณจากคอมโพเน็นต์ MD5_GG แล้วทำการคำนวณค่ารอบที่ 3 จากนั้นจะส่งค่าที่ได้ให้กับคอมโพเน็นต์ MD5_II เพื่อทำการคำนวณรอบต่อไป



รูปที่ 5-20 แสดงคอมเอนติตี MD5_HH

คำอธิบายขาสัญญาณ

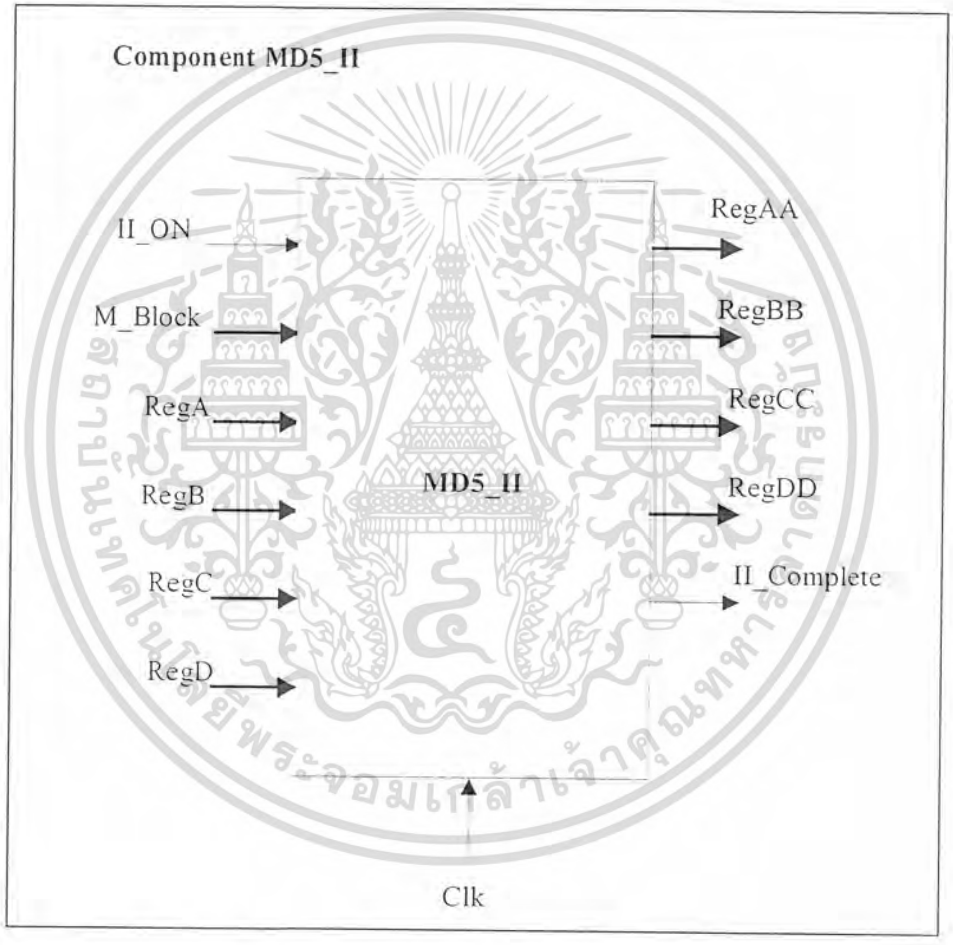
- **HH_ON** เป็นสัญญาณที่รับเข้ามาเพื่อเป็นสัญญาณปลุกการทำงานของฟังก์ชันการคำนวณ
- **M_Block** เป็นบัสข้อมูลขนาด 512 บิตที่ใช้ส่งให้กับการคำนวณในฟังก์ชันช่วยในแต่ละรอบ
- **RegA , RegB , RegC , RegD** เป็นสัญญาณข้อมูลขนาด 32 บิตที่รับมาจากการคำนวณในรอบที่ 2 เพื่อเป็นค่าที่ใช้ในการคำนวณรอบที่ 3
- **RegAA , RegBB , RegCC , RegDD** เป็นค่าข้อมูลขนาด 32 บิตที่ผ่านการคำนวณในรอบที่ 3 แล้วส่งต่อให้การคำนวณในรอบต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **HH_Completed** เป็นสัญญาณที่บอกให้ทราบว่าการคำนวณในรีจิสเตอร์ขนาด 32 บิตได้เสร็จสิ้นแล้ว

5.3.6 MD5_II

หน้าที่การทำงาน คือเป็นฟังก์ชันช่วยที่ใช้ในการคำนวณสุดท้ายของแต่ละบล็อก 512 บิต ของการทำงานโดยจะรับอินพุตเข้ามาการคำนวณจากคอมโพเน็นต์ MD5_HH แล้วทำการคำนวณค่ารอบที่ 4 จากนั้นจะส่งค่าที่ได้ให้กับคอมโพเน็นต์ MD5_CTRL เพื่อทำการคำนวณรอบต่อไป หรือนำไปเป็นเอาต์พุตของการทำงาน



รูปที่ 5-21 แสดงคอมเอนติตี MD5_II

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำอธิบายขาสัญญาณ

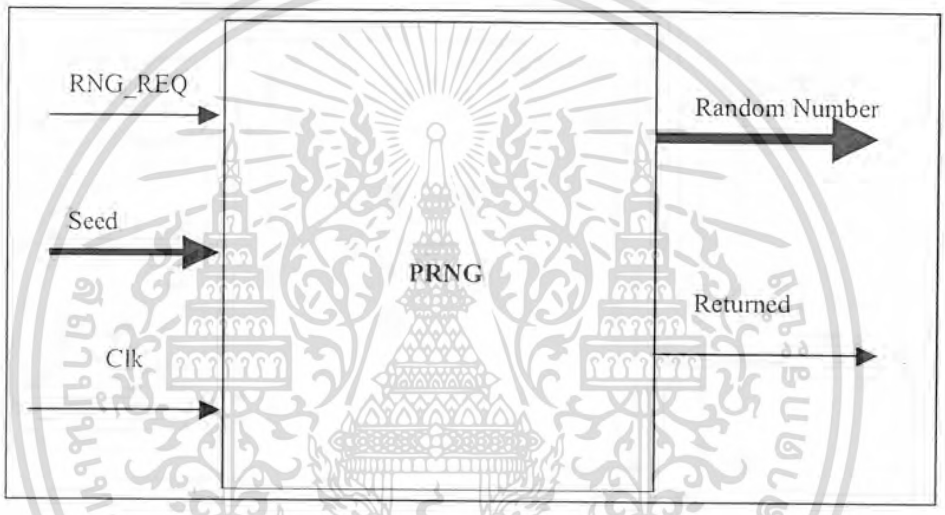
- **II_ON** เป็นสัญญาณที่รับเข้ามาเพื่อเป็นสัญญาณปลุกการทำงานของฟังก์ชันการคำนวณ
- **M_Block** เป็นบัสข้อมูลขนาด 512 บิตที่ใช้ส่งให้กับการคำนวณในฟังก์ชันช่วยในแต่ละรอบ
- **RegA , RegB , RegC , RegD** เป็นสัญญาณข้อมูลขนาด 32 บิตที่รับมาจากการคำนวณในรอบที่ 3เพื่อเป็นค่าที่ใช้ในการคำนวณรอบสุดท้ายคือรอบที่ 4
- **RegAA , RegBB , RegCC , RegDD** เป็นค่าข้อมูลขนาด 32 บิตที่ผ่านการคำนวณในรอบที่ 3 แล้วส่งต่อให้การคำนวณในรอบต่อไป
- **II_Completed** เป็นสัญญาณที่บอกให้ทราบว่าการคำนวณในรีจิสเตอร์ขนาด 32 บิตได้เสร็จสิ้นแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.4 Pseudo Random Number Generator (PRNG)

5.4.1 PRNG

ทำหน้าที่หลักในการสร้างค่าแรนดอมขนาด 64 บิต เพื่อใช้ในการสร้างคีย์ การกำหนดค่าเวกเตอร์เริ่มต้นสำหรับการเข้ารหัสของ Triple-DES ในโหมด CBC หรือค่าแรนดอมใด ๆ ขนาด 64 บิตสำหรับแอปพลิเคชั่นอื่น ๆ โดยภายในมีคอมโพเน็นต์ PRNG_MIXER เพื่อผสม Seed ขนาด 8 บิตเข้ากับ Salt ลับขนาด 64 บิตซึ่งเป็นค่าที่เก็บไว้เป็นความลับภายในตัว RNG ได้ออกเป็นเป็นค่าผสมขนาด 64 บิต โดยที่ค่าผสมนี้จะถูกสร้างขึ้นอยู่ตลอดเวลา เรานำค่าผสมนี้มาประมวลผลทางคณิตศาสตร์เพื่อสร้างค่าแรนดอมออกมา



รูปที่ 5-22 แสดงเอนติตี PRNG

คำอธิบายขาสัญญาณ

- **RNG_REQ (IN):** สัญญาณร้องขอให้ RNG ส่งค่าแรนดอมออกมาให้
- **Byte_IN (IN):** บัสข้อมูลเพื่อนำมาเป็น Seed ทำให้ค่าป่วนอยู่ตลอดเวลา
- **Rnd_OUT(OUT):** ค่าแรนดอมจะถูกส่งออกไปเมื่อมีการร้องขอมาที่ขาสัญญาณ RNG_REQ
- **Returned (OUT):** สัญญาณแจ้งให้ทราบว่าได้ส่งค่สุ่มออกไปให้แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.4.2 PRNG_MIXER

ทำหน้าที่ผสมค่า Seed 8 บิตและ Salt ลับภายในขนาด 64 บิตซึ่งมีค่าแบบไดนามิก โดยค่า Seed นั้นได้จากข้อมูลใดก็ตามที่เข้ามาในบัสข้อมูลจะถูกนำมาใช้ทำให้ค่าถูกป่วนมากขึ้น เรานำค่าทั้งสองนี้มาประมวลผลอยู่ตลอดเวลาแล้วส่งค่าที่เรียกว่าค่า MIXED ออกมาใหม่ทุก ๆ 1 รอบสัญญาณนาฬิกา

รูปที่ 5-23 แสดงเอนติตี PRNG_MIXER

คำอธิบายขาสัญญาณ

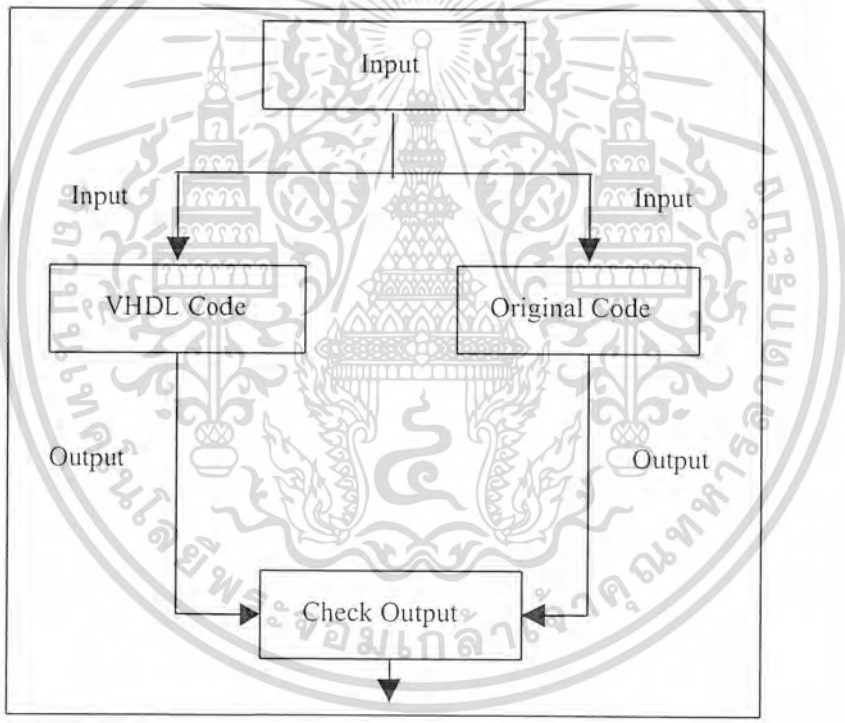
- **Salt (IN 7:0):** ค่า Seed ที่ได้จากการนำค่าสัญญาณบัสข้อมูลอินพุตขนาด 8 บิตมาผสมกับ Secret_Salt บางส่วน
- **Secret_Salt (IN 0:63):** เป็นค่า Salt ภายในที่ RNG เตรียมไว้ให้
- **Mixed_OUT (OUT 63:0):** ค่าผสมเพื่อให้ RNG นำไปใช้คำนวณหาค่าแรนดอมตลอดเวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6
ผลการทดลอง

6.1 การทดสอบผล

สำหรับการวิเคราะห์ผลการทดลองสามารถ ทำได้โดย นำโค้ดวีเอชดีแอลที่ได้ออกแบบไว้มา เอ็กซิคิวต์ (Execute)โดยให้รับค่าอินพุตค่าหนึ่ง จากนั้นนำโค้ดภาษาใดก็ได้ที่เป็นอัลกอริทึมแบบเดียวกัน มา เอ็กซิคิวต์ โดยรับค่าอินพุตที่เหมือนกันแล้ว ดูผลลัพธ์ว่าได้ผลลัพธ์ตรงกันหรือไม่ ดังแสดงรูปที่ 10

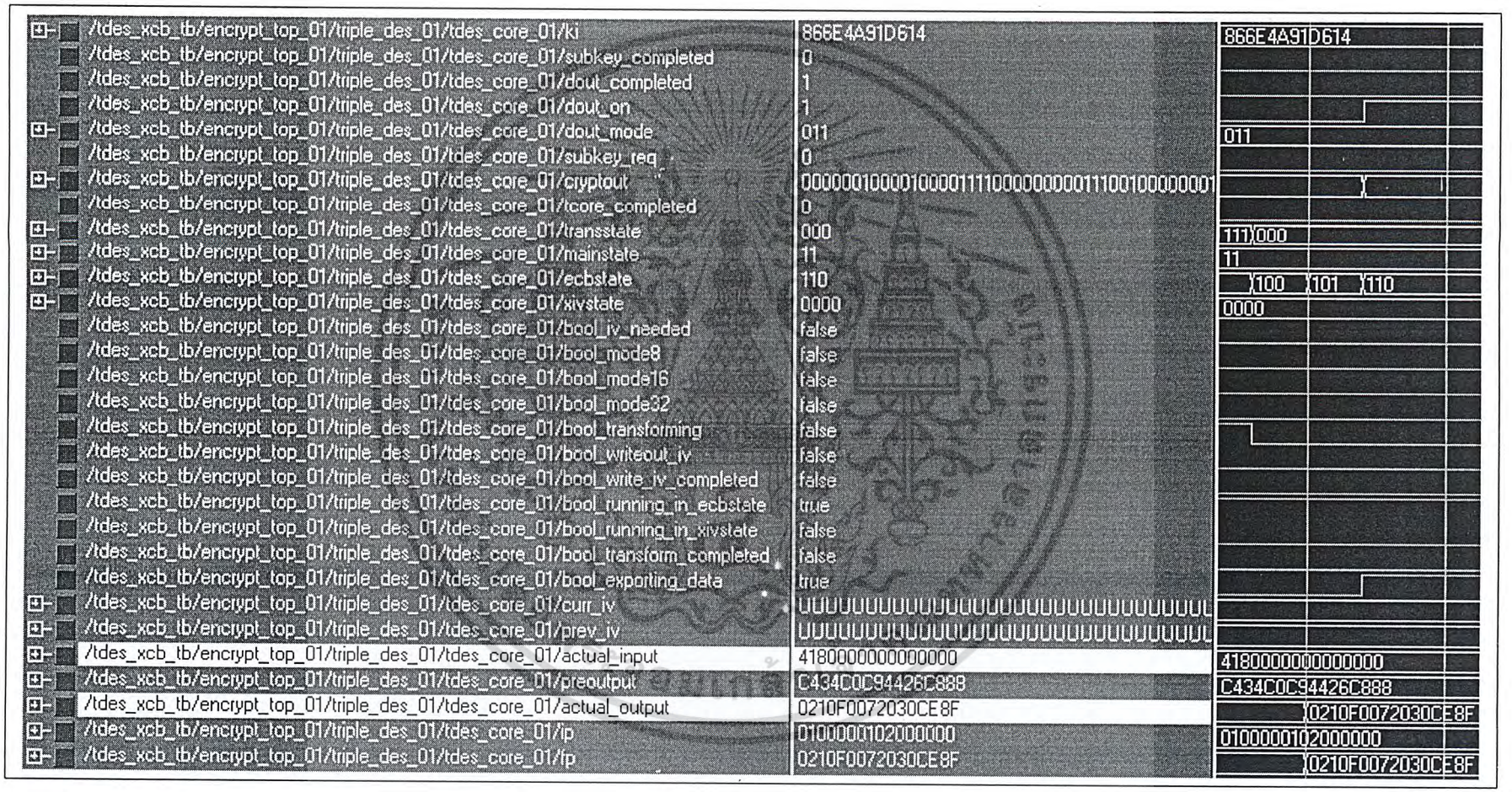


รูปที่ 6-1 แสดงขั้นตอนการตรวจสอบความถูกต้องของโค้ด

สำหรับอัลกิริทึม 3DES และ Blowfish จะต้องทำการทดสอบด้วยว่าสามารถเข้าและถอดรหัสได้ โดยการใส่ค่าอินพุตให้ อัลกอริทึมทำการคำนวณจากเพลนเท็กซ์เป็นไซเฟอร์เทกซ์ แล้วทำการถอดรหัส ไซเฟอร์เท็กซ์ ออกว่าแล้วพิจารณาดูผลลัพธ์ที่ได้ว่าสามารถถอดรหัสออกมาได้ถูกต้องหรือไม่

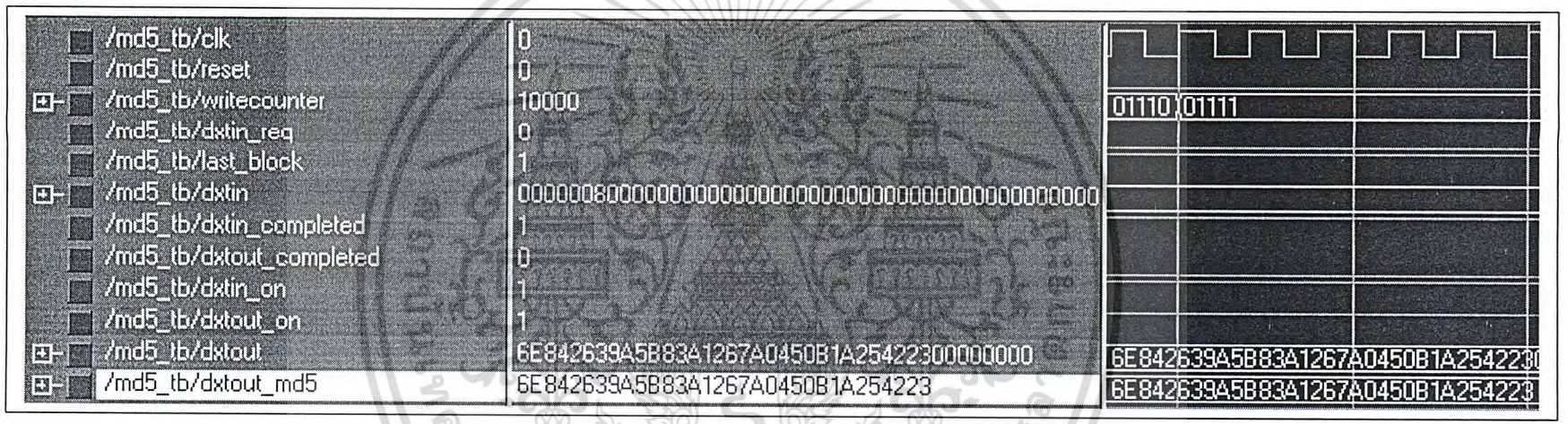
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2 ผลการซิมูเลชัน 3DES



รูปที่ 6-2 แสดงผลการซิมูเลชัน 3DES

6.3 ผลการซิมูเลชันอัลกอริทึม MD5

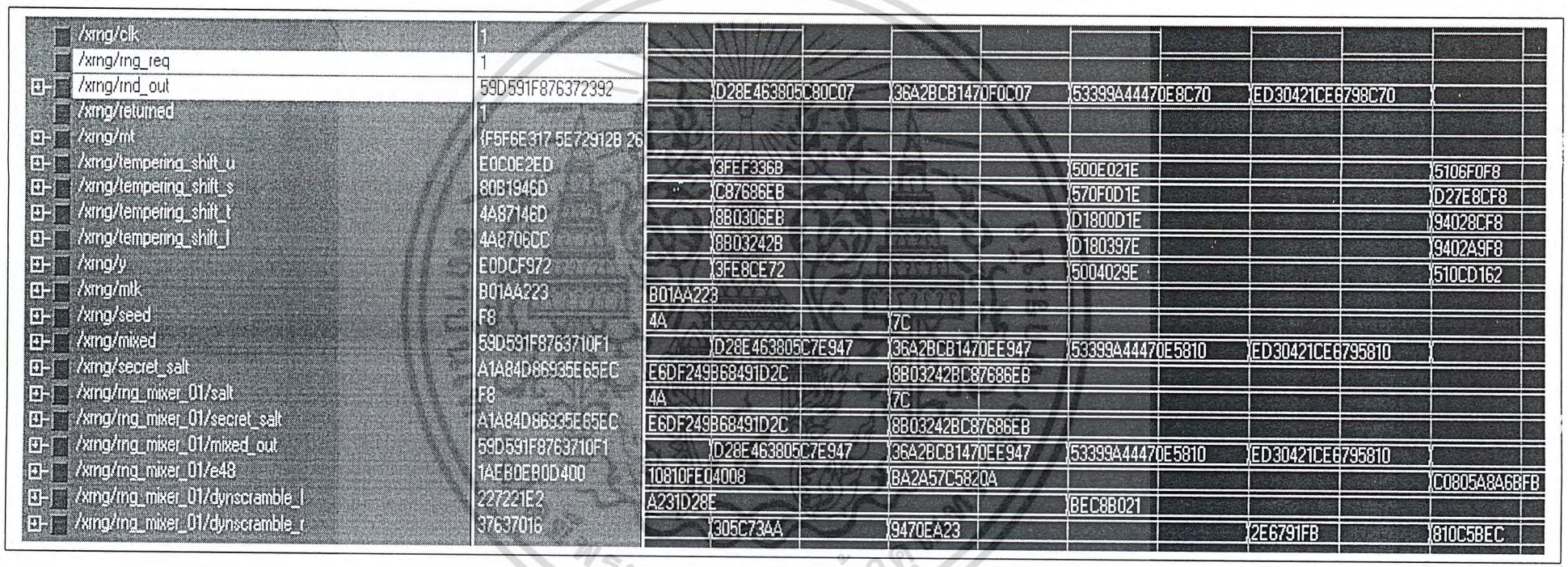


รูปที่ 6-3 แสดงผลการซิมูเลชัน อัลกอริทึม MD5

6.4 ผลการซิมูเลชันอัลกอริทึม SHA

รูปที่ 6-4 แสดงผลการซิมูเลชันอัลกอริทึม SHA

6.5 ผลการซิมูเลชันของ Random Number Generator



รูปที่ 6-5 แสดงผลการซิมูเลชันอัลกอริทึมการสุ่มตัวเลข

บทที่ 7
บทสรุปและวิจารณ์

7.1 การเลือกอัลกอริทึม

อัลกอริทึมที่เลือกใช้เป็นอัลกอริทึมที่มีมาตรฐานและได้รับความนิยมอย่างแพร่หลาย

- Triple Data Encrytion Standard (3DES) เป็นอัลกอริทึมที่มีพื้นฐานมาจาก DES แต่มีการเพิ่มความซับซ้อนของการทำงานมากขึ้น และความเร็วของการทำงานอยู่ในเกณฑ์ที่ยอมรับได้
- Blowfish เป็นอัลกอริทึมอีกตัวหนึ่งที่กำลังได้รับความนิยมอย่างแพร่หลาย เช่นเดียวกับ 3DES ใช้สำหรับการเข้าและถอดรหัส
- MD5 Message Digest Algorithm เป็นอัลกอริทึมที่มีความนิยมใช้ในการลงลายมือชื่อดิจิตอลโดยรับอินพุตขนาดความยาวไม่จำกัดแล้ว สร้างเอาต์พุตขนาด 128 บิต
- Secure Hash Algorithm (SHA) มีลักษณะคล้าย MD5 ใช้ในการลงลายมือชื่อดิจิตอล โดยรับอินพุต ขนาดความยาวไม่จำกัด แล้วสร้างเอาต์พุตขนาด 160 บิต
- Random Number Generator (RNG) ใช้สำหรับกระบวนการสร้างคีย์มีลักษณะเป็นดีเทอร์มินิสติกอัลกอริทึม (Deterministic Algorithm) ซึ่งเป็นอัลกอริทึมที่มีลำดับการสุ่มที่ไม่แน่นอน

7.2 การออกแบบอัลกอริทึม

ลักษณะอย่างหนึ่งของอัลกอริทึมเหล่านี้คือมักมีลักษณะการคำนวณที่เป็นลำดับ (Sequential) โดยการคำนวณในรอบหลังยังต้องพึ่งผลการคำนวณจากรอบก่อนๆ ทำให้การงานของโปรแกรมช้าลงไปด้วย ดังนั้นการออกแบบให้บางคอมโพเน็นต์หรือบางคำสั่งที่มีอิสระในการทำงานสามารถทำงานได้อย่างคอนเคอร์เรนต์จะช่วยให้โปรแกรมสามารถทำงานได้เร็วขึ้น สำหรับอัลกอริทึมแต่ละตัวที่ได้ออกแบบไว้ยังมีขนาดในการออกแบบที่ใหญ่และมีการทำงานที่ซับซ้อนทำให้เวลาที่ใช้ในการทำงานของโปรแกรมเพิ่มมากตามไปด้วย อาจแก้ไข โดยการออกแบบและแบ่งการทำงานของคอมโพเน็นต์ให้มีประสิทธิภาพในการทำงานมากยิ่งขึ้นและมีความซับซ้อนในการทำงานน้อยลง

7.3 การทดลอง

สำหรับโค้ดภาษาวีเอชดีแอลที่ได้ออกแบบไว้สามารถตรวจสอบการทำงานที่ถูกต้องได้โดย การเขียนโปรแกรม Test bench ขึ้นมาทดสอบการทำงาน โดยการใส่อินพุตค่าหนึ่งแล้วตรวจสอบผลลัพธ์ที่ได้ของอัลกอทึมกับโค้ดภาษาอื่น ที่รับค่าอินพุตค่าเดียวกัน ว่าให้ผลลัพธ์ตรงกันหรือไม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7.4 การใช้งาน

จากที่ได้ออกแบบโค้ดภาษาวีเอชดีแอล ของอัลกอริทึมแต่ละตัวแล้วทำการซินทีซิส (Synthesis) ผลปรากฏว่าไม่สามารถขนาดโปรแกรมมีขนาดใหญ่ ไม่สามารถนำลงชิปเอฟพีจีเอเบอร์ใดๆได้เนื่องจากจำนวนเกทของชิปเอฟพีจีเอไม่เพียงพอ ดังนั้นในอนาคตหากนักพัฒนาต้องการพัฒนาโปรแกรมให้สามารถทำงานได้จริงบน ชิป มีความจำเป็นอย่างยิ่งที่จะต้องพิจารณาถึงขั้นตอนการออปติไมซ์โค้ด (VHDL Optimization Code) เพื่อช่วยในการลดจำนวนเกตของวงจรลงได้ ทั้งนี้จะต้องประกอบการออกแบบอัลกอริทึมที่ดีด้วย

เนื่องจากปัจจัยหลายอย่างเช่น การเขียนโค้ดภาษาวีเอชดีแอลมีความยากและซับซ้อน รวมทั้งความซับซ้อนของอัลกอริทึมหลายๆตัว โดยเฉพาะอย่างยิ่ง Blowfish และระยะเวลาที่จำกัด ทำให้งานวิจัยครั้งนี้ไม่สามารถนำเสนอส่วนของการทำงานของ Blowfish ได้ มีเพียงส่วนของการออกแบบที่ได้สร้างไว้ให้นักพัฒนานำไปพัฒนาต่อ เพื่อให้ผู้ใช้สามารถเลือกอัลกอริทึมในการเข้ารหัสได้

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	Column															
Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
0	7	13	14	3	0	1	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
0	12	1	10	15	9	2	5	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	1
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	0	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข. ตาราง T[1..64] ใช้ประกอบการคำนวณหาเมสเสจไดเจสต์

รอบที่ 1	รอบที่ 2
ลำดับที่ 1 0xd76aa478	ลำดับที่ 17 0xf61e2562)
ลำดับที่ 2 0xe8c7b756	ลำดับที่ 18 0xc040b340
ลำดับที่ 3 0x242070db	ลำดับที่ 19 0x265e5a51
ลำดับที่ 5 0xf57c0faf	ลำดับที่ 21 0xd62f105d
ลำดับที่ 6 0x4787c62a	ลำดับที่ 22 0x2441453
ลำดับที่ 7 0xa8304613	ลำดับที่ 23 0xd8a1e681
ลำดับที่ 8 0xfd469501	ลำดับที่ 24 0xe7d3fbc8
ลำดับที่ 9 0x698098d8	ลำดับที่ 25 0x21e1cde6
ลำดับที่ 10 0x8b44f7af	ลำดับที่ 26 0xc33707d6
ลำดับที่ 11 0xffff5bb1	ลำดับที่ 27 0xf4d50d87
ลำดับที่ 12 0x895cd7be	ลำดับที่ 28 0x455a14ed
ลำดับที่ 13 0x6b901122	ลำดับที่ 29 0xa9e3e905
ลำดับที่ 14 0xfd987193	ลำดับที่ 30 0xfcefa3f8
ลำดับที่ 15 0xa679438e	ลำดับที่ 31 0x676f02d9
ลำดับที่ 16 0x49b40821	ลำดับที่ 32 0x8d2a4c8a

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รอบที่ 3

ลำดับที่ 33 0xfffa3942
ลำดับที่ 34 0x8771f681
ลำดับที่ 35 0x6d9d6122
ลำดับที่ 36 0xfde5380c
ลำดับที่ 37 0xa4beea44
ลำดับที่ 38 0x4bdecfa9
ลำดับที่ 39 0xf6bb4b60
ลำดับที่ 40 0xbebfbc70
ลำดับที่ 41 0x289b7ec6
ลำดับที่ 42 0xeaa127fa
ลำดับที่ 43 0xd4ef3085
ลำดับที่ 44 0x4881d05
ลำดับที่ 45 0xd9d4d039
ลำดับที่ 46 0xe6db99e5
ลำดับที่ 47 0x1fa27cf8
ลำดับที่ 48 0xc4ac5665

รอบที่ 4

ลำดับที่ 49 0xf4292244
ลำดับที่ 50 0x432aff97
ลำดับที่ 51 0xab9423a7
ลำดับที่ 52 0xfc93a039
ลำดับที่ 53 0x655b59c3
ลำดับที่ 54 0x8f0ccc92
ลำดับที่ 55 0xffeff47d
ลำดับที่ 56 0x85845dd1
ลำดับที่ 57 0x6fa87e4f
ลำดับที่ 58 0xfe2cc6e0
ลำดับที่ 59 0xa3014314
ลำดับที่ 60 0x4c0811a1
ลำดับที่ 61 0xf7537e82
ลำดับที่ 62 0xbd3af235
ลำดับที่ 63 0x2ad7d2bb
ลำดับที่ 64 0xeb86d391

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ค.

ขั้นตอนการซิมมูเลตและทดสอบผลความถูกต้อง

สำหรับการซิมมูเลตเพื่อทำการทดสอบโดยใช้โปรแกรม ModelSim เรากำหนดให้การคอมไพล์โค้ดของโมดูลในแต่ละอัลกอริทึมดังนี้

- Triple DES
  1. คอมไพล์ TDES_TRANSFORMER
  2. คอมไพล์ TDES_Core, Triple_DES_CTRL, TDES_KEY_SCHEDULER, TDES_KEY_READER, TDES_IV_READER, DATA_IMPORT, DATA_EXPORT
  3. คอมไพล์ Triple_DES, RNG, ENCRYPT_TOP, TDES_tb เรียงตามลำดับ
- MD5
  1. คอมไพล์ MD5_CTRL, MD5_FF, MD5_GG, MD5_HH, MD5_II
  2. คอมไพล์ MD5, HASH_IMPORT_MD5, HASH_EXPORT และ MD5_tb เรียงตามลำดับ
- SHA
  1. คอมไพล์ SHA_EXECUTIVE, SHA_WORD_INITIALIZER, SHA_HASH_Core, SHA_Ft, SHA_Kt
  2. คอมไพล์ SHA, HASH_IMPORT, HASH_EXPORT และ SHA_tb เรียงตามลำดับ
- PRNG
  1. คอมไพล์ PRNG_MIXER และ PRNG ตามลำดับ

หลังจากคอมไพล์โค้ดหมดทุกโมดูลแล้ว ให้ทำการ Load Design ของโมดูล testbench ที่ต้องการทดสอบ (โมดูลที่ลงท้ายด้วย _tb เช่น MD5_tb) แล้วทำการซิมมูเลตโดยนำไฟล์ทดสอบมาไว้ในโฟลเดอร์เดียวกันกับโค้ด (ดูชื่อไฟล์ในโค้ดของ testbench ของแต่ละโมดูล)

สำหรับการทดสอบผลความถูกต้อง ทำได้โดยนำโค้ดภาษาระดับสูง เช่น ภาษา C มาทำการทดสอบโดยใช้ไฟล์ทดสอบตัวเดียวกัน แล้วดูผลลัพธ์ที่ได้ว่าตรงกับผลลัพธ์ที่ได้จากการซิมมูเลตหรือไม่

สาเหตุที่ทางผู้จัดทำได้ทำโมดูล HASH_IMPORT แยกไว้สำหรับ MD5 และ SHA เนื่องจากการเรียงไบต์ในบล็อกอินพุตของทั้ง 2 โมดูล มีการจัดเรียงต่างกัน หากเราใช้โมดูลเดียวกันในการทดสอบอาจทำให้ผลลัพธ์แตกต่างจากโค้ดภาษาระดับสูงอื่นๆ ทั้งนี้ขึ้นอยู่กับผู้นำไปใช้สามารถใช้โมดูล HASH_IMPORT ตัวเดียวร่วมกันก็ได้ เพราะถึงอย่างไรก็ไม่ได้เป็นการทำให้คุณสมบัติการทำงานของอัลกอริทึมเปลี่ยนไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

[1] จักรกริศน์ เขียวสะอาด และ นริศภิญโญวัฒนากร, "*การออกแบบวงจรดิจิตอลด้วยภาษาวิเอชดีแอล*",คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2538, หน้า 1- 35

[2] โอภาส สุวิกะปกรณ์กุล , "*การออกแบบวงจรป้องกันการคัดลอกซอฟต์แวร์โดยใช้เอฟพีจีเอ*", คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2541, หน้า 5-13

[3] Sudharkar Yalamanchili, "*VHDL Starter's Guide*", 1998

[4] National Institute of Standard and Technology, "*Data Encryption Standard (DES)*", http://csrc.nist.gov/fips/, 25 ตุลาคม 2542, FIPS46-3.PDF

[5] Counterpane Internet Security, "*Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*", http://www.counterpane.com/bfsverlag.html, 2543

[6] R. Rivest, "*The MD5 Message-Digest Algorithm*", 1992, หน้า 1-21

[7] National Institute of Standard and Technology, "*Secure Hash Standard*" http://csrc.nist.gov/fips/, เมษายน 2538, FIPS180-1.PDF

[8] NHSE random number generator *library* "*An Introduction to Using Random Number Generators* ", http://nhse.npac.syr.edu/random/overview.html , 1996

[9] M. Matsumoto and T. Nishimura, "*Random Number Generator*", http://www.math.keio.ac.jp/~nisimura/random/old/19991028/int/mt19937int.c, 1998

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้