การออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต

นายพีระ ด้นธีรวงศ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2550

DESIGN OF DYNAMICALLY RECONFIGURABLE AES ENCRYPTION CIRCUIT

Mr. Peera Thontirawong

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2007

หัวข้อวิทยานิพนธ์	การออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต
โดย	นายพีระ ต้นธีรวงศ์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รองศาสตราจารย์ ดร.สมชาย ประสิทธิ์จูตระกูล)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา)

.....กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.เขมะชัต วิภาตะวนิช)

.....กรรมการ
(อาจารย์ ดร.เศรษฐา ปานงาม)

พีระ ต้นธีรวงศ์ : การออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต. (DESIGN OF DYNAMICALLY RECONFIGURABLE AES ENCRYPTION CIRCUIT) อ. ที่ปรึกษา : ศ.ดร.ประภาส จงสถิตย์วัฒนา, 70 หน้า.

วิทยานิพนธ์นี้เสนอการออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต โดยใช้เอฟพีจีเอของ Xilinx รุ่น Spartan-3 การออกแบบดังกล่าวนำทรัพยากรของเอฟพีจีเอกลับมาใช้ใหม่ได้ โดยนำแนวคิดของการเปลี่ยนโครงแบบอย่างพลวัตมาประยุกต์เพื่อลดจำนวนทรัพยากรที่ใช้ในวงจร เนื่องจากทรัพยากรที่ใช้ในวงจรที่เปลี่ยนโครงแบบได้อย่างพลวัตนั้นจะถูกจำกัดโดยขนาดของวงจรส่วนที่ไม่สามารถเปลี่ยนโครงแบบได้และขนาดของวงจรย่อยส่วนที่สามารถเปลี่ยนโครงแบบได้ที่ใหญ่ที่สุด ดังนั้นวงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัตจึงสามารถมีขนาดเล็กกว่าวงจรปกติ ในการสร้างวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตนี้บนเอฟพีจีเอ XC3S200-4FT256 ต้องการทรัพยากรเพียง 349 สไลซ์ ในขณะที่ได้ปริมาณงาน 25 กิโลบิตต่อวินาที และ 16 เมกะบิตต่อวินาทีเมื่อสมมติให้ไม่เสียเวลาการเปลี่ยนโครงแบบ

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....
สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา .....
ปีการศึกษา 2550

4930487421 : MAJOR COMPUTER ENGINEERING

KEY WORD: AES / FPGA / Dynamic Reconfig

PEERA THONTIRAWONG : DESIGN OF DYNAMICALLY RECONFIGURABLE AES ENCRYPTION CIRCUIT. THESIS ADVISOR : PROF.PRABHAS CHONGSTITVATANA, Ph.D., 70 pp.

This thesis presents a design of a Dynamic Reconfigurable Advanced Encryption Standard (AES) encryption unit based on the Xilinx Spartan-3 FPGA platform. The proposed designs reuse the resource of FPGA by adapting the dynamic reconfiguration concept to reduce the number of resource used in the circuit. Since the resource used in a dynamic reconfigurable circuit is constrained by the size of static module and the largest reconfigurable module, so the dynamic reconfigurable circuit can be smaller than an ordinary circuit. The implementation of the dynamic reconfigurable AES encryption circuit on XC3S200-4FT256 requires only 349 slices, while achieving the throughput of 25 Kbps. If assume that there is no reconfiguration delay, the throughput becomes 16 Mbps.

Department Computer Engineering Student's signature

Field of study Computer Engineering Advisor's signature

Academic year 2007

กิตติกรรมประกาศ

ขอขอบคุณอาจารย์ที่ปรึกษาวิทยานิพนธ์ ศ.ดร.ประภาส จงสถิตย์วัฒนา ผู้เสียสละเวลาอันมีค่าให้คำปรึกษาที่เป็นประโยชน์ต่องานวิจัยนี้ และยังเป็นผู้ชี้แนะแนวทางในการทำวิทยานิพนธ์ ขอขอบคุณคณะกรรมการสอบวิทยานิพนธ์ ผู้ให้คำแนะนำและชี้จุดบกพร่องที่ควรแก้ไข ทำให้วิทยานิพนธ์นี้เสร็จสมบูรณ์ได้

ขอขอบคุณสมาชิกในห้องปฏิบัติการวิจัยระบบอัจฉริยะ และเพื่อนๆในระดับบัณฑิตศึกษาทุกคน ที่คอยเอาใจใส่ ให้คำปรึกษา และช่วยสร้างบรรยากาศที่ดีในการทำงานตลอดมา

ขอขอบคุณคณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ที่ช่วยสั่งสอนและวางรากฐานความรู้อันเป็นประโยชน์ต่อการทำวิทยานิพนธ์ รวมทั้งให้คำชี้แนะที่เป็นประโยชน์เสมอมา

ขอขอบคุณจุฬาลงกรณ์มหาวิทยาลัย สถาบันที่ให้ความรู้ที่มีประโยชน์อย่างยิ่ง รวมถึงโรงเรียนเตรียมอุดมศึกษา และโรงเรียนสาธิต มศว ปทุมวัน ซึ่งเป็นสถาบันที่ช่วยสร้างพื้นฐานความรู้

สุดท้ายนี้ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ผู้ซึ่งเป็นกำลังใจ และคอยให้ความช่วยเหลืออยู่เสมอมา ความสำเร็จนี้คงเกิดขึ้นไม่ได้หากไม่มีท่านทั้งสองให้การสนับสนุน

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1. ความเป็นมาและความสำคัญของปัญหา

ระบบฝังตัว (embedded system) หมายถึงระบบประมวลผลเฉพาะทางที่ถูกออกแบบมาให้ทำงานอย่างใดอย่างหนึ่งโดยเฉพาะ ต่างจากคอมพิวเตอร์ทั่วไป หรือคอมพิวเตอร์ส่วนบุคคล (PC) ที่ถูกออกแบบมาให้รองรับการทำงานที่หลากหลาย

เนื่องจากระบบฝังตัวถูกออกแบบมาให้ทำงานอย่างใดอย่างหนึ่งโดยเฉพาะ จึงสามารถทำงานเฉพาะทางได้ดีกว่าคอมพิวเตอร์ทั่วไป เพราะผู้ออกแบบสามารถพัฒนาระบบฝังตัวให้มีประสิทธิภาพมากที่สุดได้ ไม่ว่าจะเป็นทางด้านสมรรถนะ, ทรัพยากร, พลังงาน และต้นทุนในการผลิต เราจึงเห็นได้ว่าระบบฝังตัวนั้นสามารถพบได้ทั่วไปในชีวิตประจำวัน ไม่ว่าจะเป็นในโทรศัพท์มือถือ, โทรทัศน์, วิทยุ, รถยนต์ หรือแม้กระทั่งในนาฬิกาดิจิทัล

เอฟพีจีเอ (FPGA: Field Programmable Gate Array) เป็นวงจรรวม (IC: Integrated Circuit) ชนิดหนึ่งที่นิยมใช้ในการออกแบบหน่วยประมวลผลของระบบฝังตัวก่อนจะนำไปสร้างเป็นวงจรรวมเฉพาะงาน, ASIC (Application-Specific Integrated Circuit) เพราะเอฟพีจีเอมีความสามารถในการในการเปลี่ยนโครงแบบ (Reconfigurable) ทำให้ช่วยลดค่าใช้จ่ายในการออกแบบได้ ทำให้ผู้ออกแบบสามารถออกแบบระบบดิจิทัลที่มีความซับซ้อนได้รวดเร็วยิ่งขึ้น

เมื่อหน่วยประมวลผลมีความซับซ้อนมากขึ้น ทำให้มีความจำเป็นต้องเพิ่มทั้งขนาดของวงจรและพลังงานที่ใช้ในการประมวลผล แต่ด้วยความสามารถในการเปลี่ยนโครงแบบของเอฟพีจีเอ ทำให้เราสามารถออกแบบวงจรที่มีความซับซ้อนได้มากขึ้นในขณะที่ขนาดของวงจรเท่าเดิม เพราะเราสามารถเปลี่ยนโครงแบบของเอฟพีจีเอได้อย่างพลวัต (Dynamic Reconfiguration) หรือเรียกอีกอย่างว่าเปลี่ยนโครงแบบของเอฟพีจีเอขณะทำงาน (Run-Time Reconfiguration) ซึ่งหนึ่งในประโยชน์ของความสามารถในการเปลี่ยนโครงแบบบางส่วน (Partial Reconfiguration) ของเอฟพีจีเอ [1] ที่ทำให้เราสามารถเปลี่ยนวงจรในส่วนที่ไม่ถูกใช้งานในขณะหนึ่ง ให้กลายเป็นอีกวงจรหนึ่ง จึงทำให้เราสามารถใช้ทรัพยากรที่มีอยู่ได้อย่างเต็มที่

การออกแบบหน่วยประมวลผลที่มีขนาดใหญ่และซับซ้อนให้มีประสิทธิภาพนั้นทำได้ยาก การประยุกต์แนวคิดการเปลี่ยนโครงแบบได้สามารถทำให้การออกแบบวงจรดังกล่าวให้มีประสิทธิภาพง่ายขึ้น ดังจะเห็นได้จากงานวิจัยหลายงาน [2]-[5] ที่พยายามพัฒนาการออกแบบวงจรที่สามารถเปลี่ยนโครงแบบได้ อย่างไรก็ดี การประยุกต์แนวคิดการเปลี่ยนโครงแบบนั้น ยังต้องอาศัยการแบ่งส่วนวงจรออกเป็นส่วนย่อยๆ

วงจรเข้าและถอดรหัสเออีเอส (AES: Advance Encryption Standard) [6] ก็เป็นหน่วยประมวลผลที่มีความซับซ้อน และมีขนาดใหญ่ ที่ยังสามารถพัฒนาให้มีประสิทธิภาพเพิ่มขึ้นได้อีก

มาก ดังจะเห็นได้ว่ามีงานวิจัยหลายงาน [7]-[13] ที่นำเอาแนวคิดต่างๆมาพัฒนาวงจรนี้ รวมถึงแนวคิดการเปลี่ยนโครงแบบด้วย แต่ก็ยังไม่มีงานวิจัยใดที่สร้างวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบและทำงานอย่างต่อเนื่อง หรือมีการเปลี่ยนโครงแบบอย่างพลวัตได้จริง และเนื่องจากการเข้าและถอดรหัสเออีเอสเป็นการเข้ารหัสที่เป็นมาตรฐาน และใช้กันอย่างแพร่หลาย นอกจากนี้ยังมีขั้นตอนวิธีการเข้ารหัสที่ชัดเจน สามารถแบ่งการออกแบบวงจรออกเป็นส่วนย่อยๆได้โดยง่าย ซึ่งจะทำให้การประยุกต์แนวคิดการเปลี่ยนโครงแบบทำได้ง่ายขึ้น

โครงงานวิจัยนี้จะนำเสนอการออกแบบของวงจรเข้ารหัสเออีเอส 128 บิต โดยใช้อาศัยความสามารถในการเปลี่ยนโครงแบบได้ของเอฟพีจีเอเป็นแกนหลัก ซึ่งคาดว่าจะทำให้ได้วงจรที่มีขนาดเล็กลงกว่าที่มีอยู่ในปัจจุบัน

1.2. วัตถุประสงค์ของการวิจัย

เพื่อพัฒนาวงจรเข้ารหัสแบบเออีเอส 128 บิต ที่สามารถเปลี่ยนแปลงโครงแบบบางส่วนได้อย่างพลวัตบนอุปกรณ์เอฟพีจีเอ

1.3. ขอบเขตของการวิจัย

1. งานวิจัยนี้จะสร้างวงจรเข้ารหัสแบบเออีเอสที่สามารถเปลี่ยนแปลงโครงแบบบางส่วนได้อย่างพลวัตบนอุปกรณ์เอฟพีจีเอที่สนับสนุนการเปลี่ยนโครงแบบบางส่วนของ Xilinx เท่านั้น
2. วงจรเข้ารหัสแบบเออีเอสที่พัฒนาขึ้นสามารถเข้ารหัสข้อมูลแบบเออีเอส 128 บิต ได้อย่างถูกต้อง

1.4. ประโยชน์ที่คาดว่าจะได้รับ

สามารถสร้างวงจรเข้ารหัสแบบเออีเอส 128 บิตที่สามารถเปลี่ยนแปลงโครงแบบบางส่วนได้อย่างพลวัตบนอุปกรณ์เอฟพีจีเอ

1.5. วิธีดำเนินงานวิจัย

1. ศึกษาการออกแบบวงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัต
2. ศึกษาการเข้ารหัสแบบเออีเอส
3. ออกแบบเข้ารหัสเออีเอสวงจรที่สามารถเปลี่ยนแปลงโครงแบบบางส่วนได้อย่างพลวัต
4. ทดลองและปรับปรุง
5. พัฒนาประสิทธิภาพ
6. สรุปผลและเรียบเรียงวิทยานิพนธ์

1.6. ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

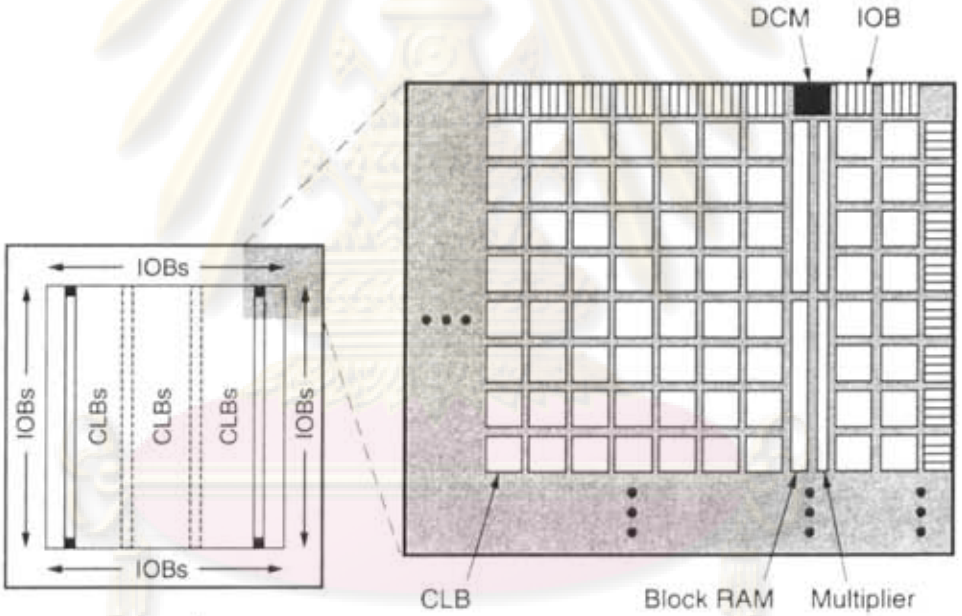
ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง "Augmenting a Stack-based Virtual Machine with One-address Instructions for Performance Enhancement" โดย พีระ ต้นธีรวงศ์ และ ศ.ดร.ประภาส จงสถิตย์วัฒนา ในงานประชุมวิชาการ "The International Conference in Embedded Systems and Intelligent Technology 2008 (ICESIT2008)" ณ โรงแรม Grand Mercure Fortune จังหวัดกรุงเทพ ในระหว่างวันที่ 27 - 29 กุมภาพันธ์ 2551

นอกจากนี้ยังได้รับการตอบรับให้ตีพิมพ์เป็น "Dynamic Reconfigurable AES Encryption Circuits" โดย พีระ ต้นธีรวงศ์ และ ศ.ดร.ประภาส จงสถิตย์วัฒนา ในงานประชุมวิชาการ "The 5th International Joint Conference on Computer Science and Software Engineering (JCSSE2008)" ณ Felix River Kwai Resort จังหวัดกาญจนบุรี ในระหว่างวันที่ 7 - 9 พฤษภาคม 2551

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในการพัฒนาวงจรที่สามารถเปลี่ยนแปลงโครงแบบบางส่วนได้อย่างพลวัตบนอุปกรณ์เอฟพีจีเอ เราจำเป็นต้องมีความรู้เกี่ยวกับสถาปัตยกรรมของอุปกรณ์เอฟพีจีเอ [14],[15] และต้องมีความรู้เกี่ยวกับวิธีการเปลี่ยนโครงแบบของอุปกรณ์เอฟพีจีเอ [16] เป็นอย่างดี เพราะการออกแบบวงจรที่สามารถเปลี่ยนแปลงโครงแบบบางส่วนได้อย่างพลวัตนั้น ต้องอาศัยวิธีการเปลี่ยนโครงแบบบางส่วน (Partial Reconfiguration) [17] เพราะวงจรต้องมีส่วนที่เปลี่ยนแปลงโครงแบบไม่ได้ไว้ควบคุมการเปลี่ยนโครงแบบ และเก็บข้อมูลที่ได้จากโครงแบบเก่า และส่งต่อไปยังโครงแบบใหม่ นอกจากนี้การพัฒนาวงจรเข้ารหัสแบบเออีเอส 128 บิตนั้น จะต้องใช้วิธีการเข้ารหัสโดยใช้ขั้นตอนวิธี Rijndael โดยทั้งหมดมีรายละเอียดดังนี้

2.1. สถาปัตยกรรมของ Xilinx เอฟพีจีเอรุ่น Spartan-3 (Xilinx Spartan-3 FPGA Architecture)



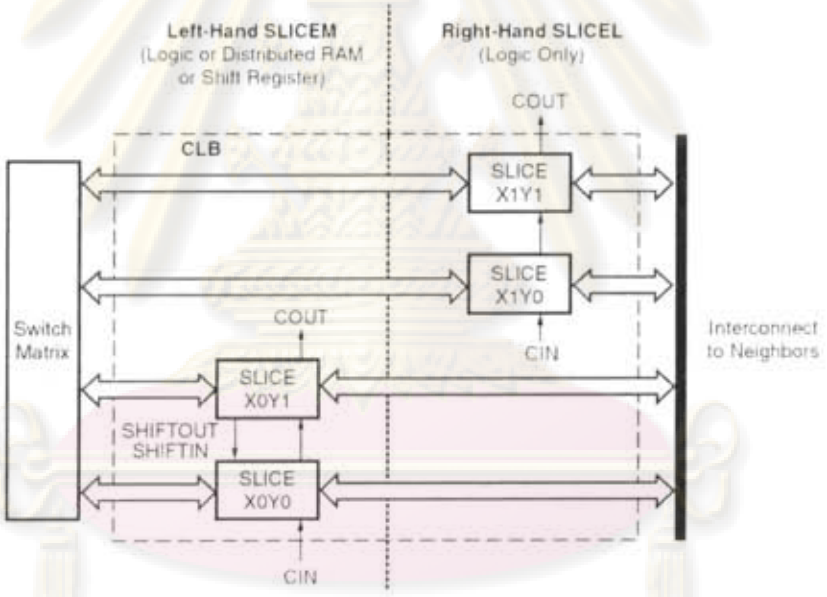
รูปที่ 2.1 สถาปัตยกรรมของ Xilinx เอฟพีจีเอรุ่น Spartan-3

อุปกรณ์เอฟพีจีเอของ Xilinx รุ่น Spartan-3 มีส่วนประกอบที่สามารถโปรแกรมการทำงานได้อยู่ 5 ส่วน คือ

1. บล็อกตรรกะที่สามารถเปลี่ยนโครงแบบได้ (CLB: Configurable Logic Block) ภายในบรรจุ slice ซึ่งประกอบด้วยตารางค้นหา (LUT: Look-Up Table) ซึ่งใช้ดำเนินการทางตรรกะ และหน่วยเก็บข้อมูลทำหน้าที่แทน ฟลิปฟล็อป (flip-flop) หรือแลตช์ (latch) โดย slice จะมีการจัดเรียงตัวเป็นคู่ดังรูปที่ 2.2

2. บล็อกอินพุต/เอาต์พุต (IOB: Input/Output Block) ควบคุมการไหลของข้อมูลระหว่างขั้วไอ/โอ (I/O pin) กับหน่วยตรรกะภายในอุปกรณ์
3. บล็อกแรม (Block RAM) เป็นหน่วยเก็บข้อมูลขนาด 18 กิโลบิต แบบช่องทางคู่ โดยแบ่งเป็น 16 กิโลบิตสำหรับข้อมูล และ 2 กิโลบิตสำหรับภาวะคู่หรือคี่ (parity)
4. บล็อกตัวคูณ (Multiplier Block) รับข้อมูลเลขฐานสองขนาด 18 บิตจำนวน 2 ตัวเพื่อคำนวณหาผลคูณ
5. บล็อกจัดการสัญญาณนาฬิกาแบบดิจิทัล (Digital Clock Manager, DCM) ใช้ปรับพิกัดด้วยตัวเอง (self-calibrating), กระจายสัญญาณนาฬิกา, หน่วงสัญญาณนาฬิกา, คูณสัญญาณนาฬิกา, หารสัญญาณนาฬิกา และปรับเฟสของสัญญาณนาฬิกา

ส่วนประกอบเหล่านี้ถูกจัดเรียงไว้ดังแสดงในรูปที่ 2.1 โดยส่วนประกอบทั้งห้านี้ได้ถูกเชื่อมโยงไว้ด้วยเครือข่ายเชื่อมโยง (interconnect) และสวิตช์เมทริกซ์ (switch matrix) โดยแต่ละหน่วยจะมีสวิตช์เมทริกซ์เป็นของตัวเอง



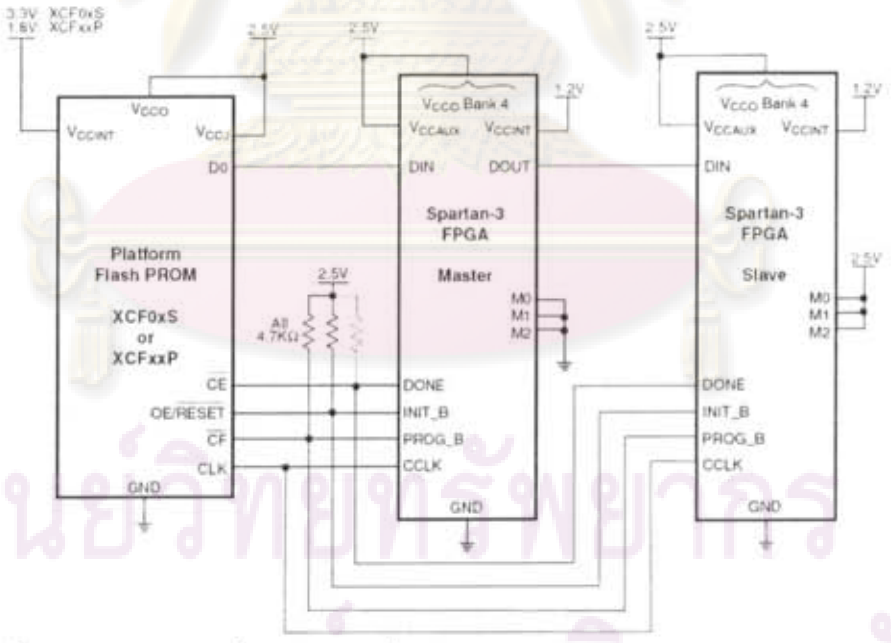
รูปที่ 2.2 การจัดเรียงของ Slice ภายใน CLB

2.2. การเปลี่ยนโครงแบบของ Xilinx เอฟพีจีเอรุ่น Spartan-3 (Xilinx Spartan-3 FPGA Configuration)

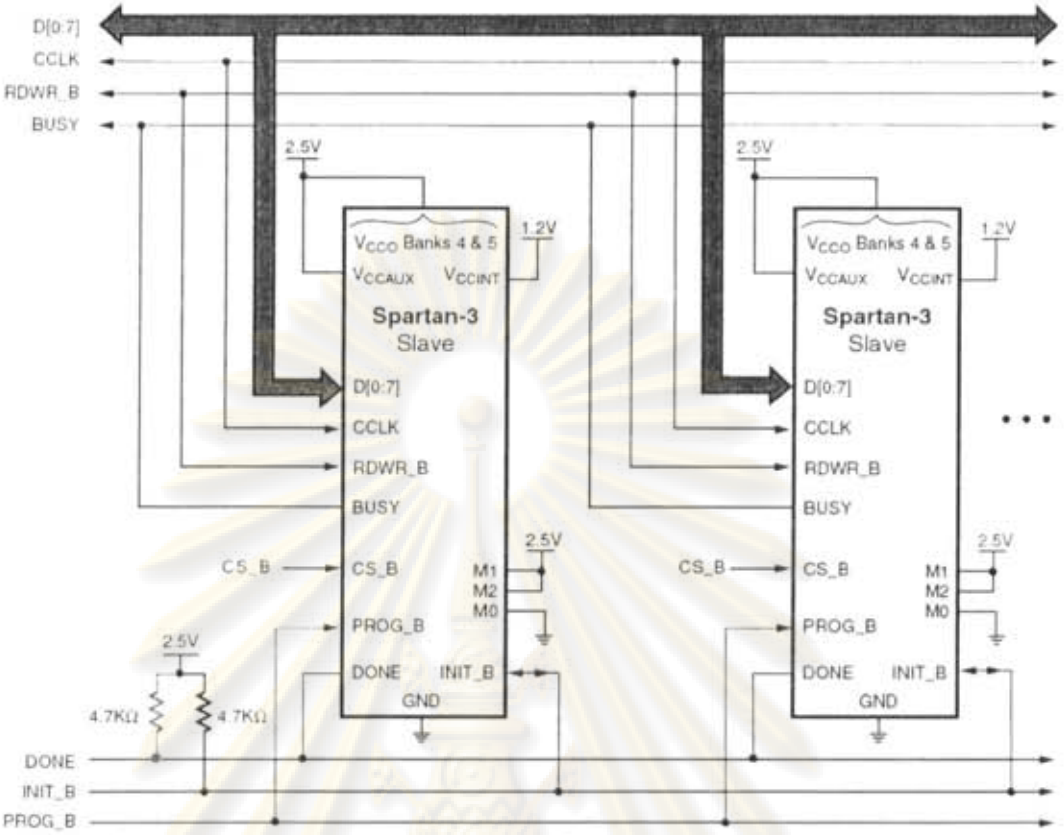
การเปลี่ยนโครงแบบ (Configuration) หรือการโปรแกรม Xilinx เอฟพีจีเอรุ่น Spartan-3 สามารถทำได้โดยการเขียนข้อมูลโครงแบบลงไปยังแลตช์โครงแบบแบบซีมอส (CCL: CMOS Configuration Latch) ซึ่งทำหน้าที่ควบคุมหน่วยการทำงานทั้ง 5 แบบและเส้นทางการเชื่อมโยงทรัพยากร

การเปลี่ยนโครงแบบของ Xilinx เอฟพีจีเอรุ่น Spartan-3 มี 5 วิธี ได้แก่

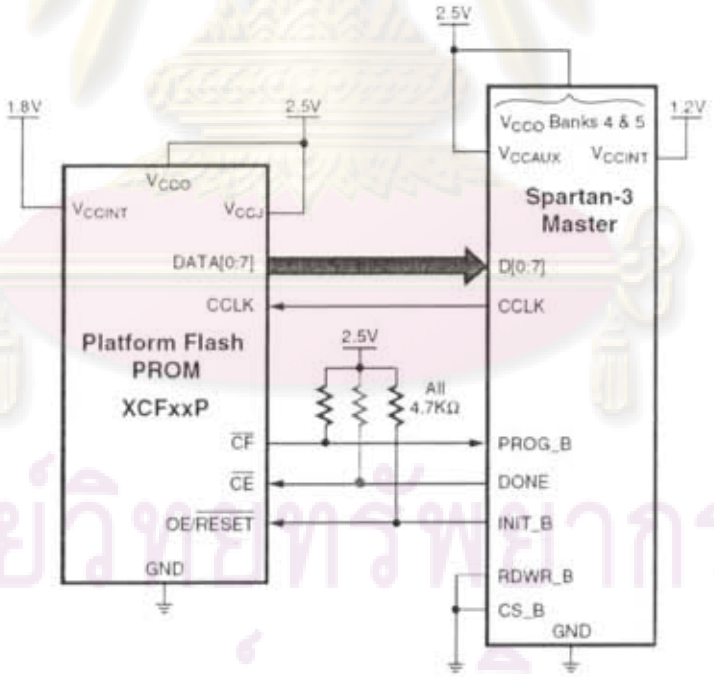
1. Slave Serial Mode ในวิธีนี้เอฟพีจีเอจะรับข้อมูลโครงแบบในลักษณะของบิตอนุกรม (bit-serial) จากพรอมแบบอนุกรม หรือแหล่งข้อมูลแบบอนุกรมอื่นๆ โดยเข็ม CCLK ของเอฟพีจีเอในวิธีนี้จะทำงานแบบรับเข้า ดังแสดงให้เห็นโดยเอฟพีจีเอทางขวาในรูปที่ 2.3
2. Master Serial Mode ในวิธีนี้คล้ายกับวิธี Slave Serial Mode แต่ต่างกันที่วิธีนี้เอฟพีจีเอจะเป็นผู้กำหนดสัญญาณนาฬิกา โดยเข็ม CCLK ของเอฟพีจีเอในวิธีนี้จะทำงานแบบสองทาง ดังแสดงให้เห็นเอฟพีจีเอตรงกลางในรูปที่ 2.3
3. Slave Parallel (SelectMAP) Mode วิธีนี้เป็นวิธีการเปลี่ยนโครงแบบที่รวดเร็วที่สุด โดยข้อมูลกว้างหนึ่งไบต์ (byte-wide) จะถูกเขียนลงยังเอฟพีจีเอโดยมีสัญญาณ BUSY เป็นตัวบ่งชี้เพื่อควบคุมการไหลของข้อมูล ดังแสดงในรูปที่ 2.4
4. Master Parallel (SelectMAP) Mode ในวิธีนี้คล้ายกับวิธี Slave Parallel Mode แต่ต่างกันที่วิธีนี้เอฟพีจีเอจะเป็นผู้กำหนดสัญญาณนาฬิกา โดยเข็ม CCLK ของเอฟพีจีเอในวิธีนี้จะทำงานแบบสองทางคือรับเข้าและส่งออก ดังแสดงให้เห็นในรูปที่ 2.5
5. Boundary-Scan (JTAG) Mode (IEEE 1532 / IEEE 1149.1) ในวิธีนี้การเปลี่ยนโครงแบบของเอฟพีจีเอจะทำผ่าน IEEE 1149.1 Test Access Port (TAP) วิธี Boundary-Scan นี้เป็นไปตามมาตรฐาน IEEE 1149.1-1993 และ IEEE1532 สำหรับอุปกรณ์ที่สามารถเปลี่ยนโครงแบบได้ภายในระบบ (In-System Configurable, ISC)



รูปที่ 2.3 แผนภาพการเชื่อมต่อการเปลี่ยนโครงแบบด้วยวิธี Master และ Slave Serial



รูปที่ 2.4 แผนภาพแสดงการเชื่อมต่อการเปลี่ยนโครงแบบด้วยวิธี Slave Parallel



รูปที่ 2.5 แผนภาพแสดงการเชื่อมต่อการเปลี่ยนโครงแบบด้วยวิธี Master Parallel

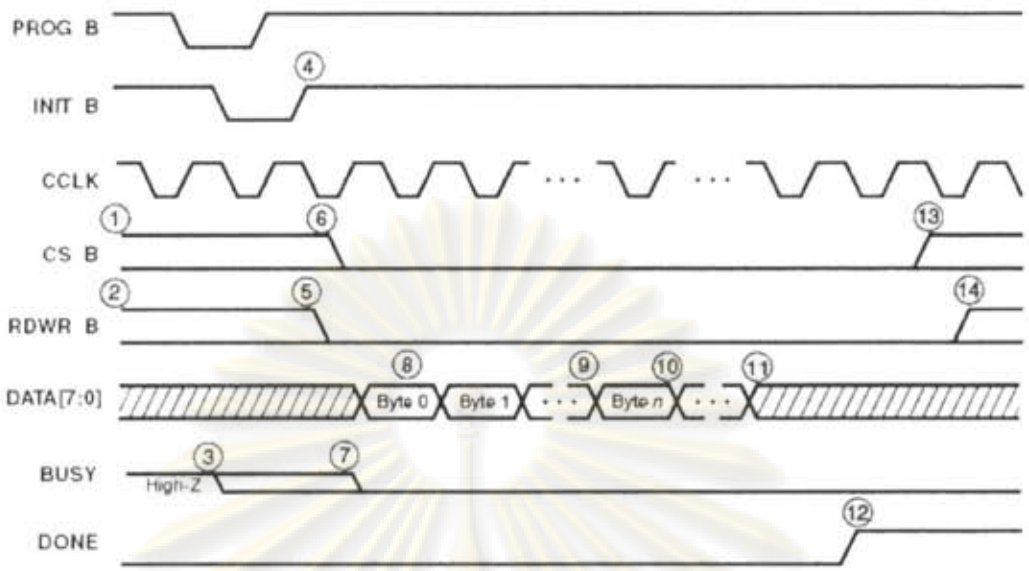
2.2.1. การเปลี่ยนโครงแบบด้วยวิธี SelectMAP

ในการเปลี่ยนโครงแบบของ Xilinx เอฟพีจีเอรุ่น Spartan-3 ด้วยวิธี SelectMAP จะมีสัญญาณที่เกี่ยวข้องทั้งหมด 10 สัญญาณ ได้แก่ HSWAP_EN, M, D, BUSY, CS_B, RDWR_B, CCLK, INIT_B, DONE และ PROG_B ซึ่งมีรายละเอียดดังตารางที่ 2.1

ตารางที่ 2.1 สัญญาณสำหรับการเปลี่ยนโครงแบบด้วยวิธี SelectMAP

Pin Name	FPGA Direction	Description
HSWAP_EN	Input	User I/O Pull-Up Control.
M[2:0]	Input	Mode Select. Selects the FPGA configuration mode.
D[7:0]	Input	Data Input.
BUSY	Output	Busy Indicator.
CS_B	Input	Chip Select Input. Active Low.
RDWR_B	Input	Read/Write Control. Active Low write enable.
CCLK	Input	Configuration Clock.
INIT_B	Open-drain bidirectional I/O	Initialization Indicator. Active Low. Goes Low at the start of configuration during the Initialization memory clearing process. Released at the end of memory clearing, when mode select pins are sampled.
DONE	Open-drain bidirectional I/O	FPGA Configuration Done. Low during configuration. Goes high when FPGA successfully completes configuration.
PROG_B	Input	Program FPGA. Active Low. When assert Low for 500 ns or longer, forces the FPGA to restart its configuration process by clearing configuration memory and resetting the DONE and INIT_B pins once PROG_B returns High.

การเปลี่ยนโครงแบบด้วยวิธี SelectMAP สามารถกระทำได้โดยส่งสัญญาณควบคุมไปยังเอฟพีจีเอดังรูปที่ 2.6 โดยสัญญาณในรูปเป็นการเปลี่ยนโครงแบบด้วยการส่งข้อมูลอย่างต่อเนื่อง สำหรับการเปลี่ยนโครงแบบโดยการส่งข้อมูลแบบไม่ต่อเนื่องจะมีขั้นตอนต่างไป



รูปที่ 2.6 ขั้นตอนการเปลี่ยนโครงแบบด้วยวิธี SelectMAP โดยการส่งข้อมูลอย่างต่อเนื่อง

ขั้นตอนการเปลี่ยนโครงแบบโดยการส่งข้อมูลอย่างต่อเนื่องมีรายละเอียดดังนี้

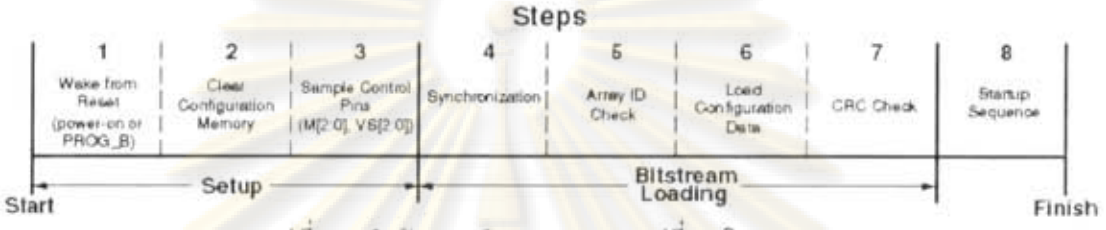
1. สัญญาณ CS_B สามารถเป็นสถานะต่ำตลอดเวลาได้ หากมีอุปกรณ์เพียงตัวเดียวที่ต่ออยู่บนสาย SelectMAP โดยสัญญาณนี้สามารถถูกดึงให้เป็นสถานะต่ำเมื่อไรก็ได้
2. สัญญาณ RDWR_B สามารถเป็นสถานะต่ำตลอดเวลาได้ หากไม่มีการอ่านข้อมูลกลับ โดยสัญญาณ RDWR_B นี้จะต้องไม่มีการเปลี่ยนแปลงค่าขณะที่ CS_B เป็นสถานะต่ำเพราะจะทำให้เกิดการยกเลิกการเปลี่ยนโครงแบบ
3. ถ้าสัญญาณ CS_B ถูกทำให้เป็นสถานะต่ำตลอดเวลา สัญญาณ BUSY จะถูกดึงลงเป็นสถานะต่ำก่อนสัญญาณ INIT_B จะกลับเป็นสถานะสูง
4. เอฟพีจีเอจะอ่านสัญญาณ M[2:0] เพื่อเลือกวิธีการเปลี่ยนโครงแบบในขณะที่สัญญาณ INIT_B เปลี่ยนเป็นสถานะสูง
5. ควรดึงสัญญาณ RDWR_B ให้เป็นสถานะต่ำก่อนสัญญาณ CS_B เพื่อป้องกันการยกเลิกการเปลี่ยนโครงแบบโดยไม่ตั้งใจ
6. สัญญาณ CS_B ถูกดึงลงต่ำเพื่อเปิดการทำงานของ SelectMAP
7. สัญญาณ CS_B จะเป็น High-Z จนกว่ายันสัญญาณ CS_B จะถูกดึงลงต่ำ
8. ข้อมูลไบต์แรกของสัญญาณ D[7:0] จะถูกอ่านในขอบสัญญาณขาขึ้นของสัญญาณ CCLK หลังจากสัญญาณ CS_B ถูกดึงลงต่ำ
9. ข้อมูลโครงแบบจะถูกอ่านหนึ่งไบต์ต่อหนึ่งขอบสัญญาณขาขึ้นของสัญญาณ CCLK
10. เอฟพีจีเอจะเข้าสู่ขั้นตอน Startup หลังจากข้อมูลไบต์สุดท้ายถูกอ่าน
11. ขั้นตอน Startup จะดำเนินอยู่อย่างน้อย 8 รอบสัญญาณ CCLK
12. สัญญาณ DONE จะถูกดึงขึ้นสูงระหว่างขั้นตอน Startup

13. หลังจากการเปลี่ยนโครงแบบสิ้นสุด สัญญาณ CS_B สามารถถูกดึงขึ้นสูงได้

14. หลังจากสัญญาณ CS_B ถูกดึงขึ้นสูง สัญญาณ RDWR_B สามารถถูกดึงขึ้นสูงได้

2.2.2. ขั้นตอนการเปลี่ยนโครงแบบ

ขั้นตอนในการเปลี่ยนโครงแบบของ Xilinx เอฟพีจีเอรุ่น Spartan-3 มีทั้งหมด 8 ขั้นตอน โดยทั้ง 8 ขั้นตอนจะแบ่งออกเป็น 3 กลุ่มได้แก่ การจัดเตรียม, การบรรจุกระแสข้อมูล และการเริ่มงาน ซึ่งมีลำดับการทำงานดังรูปที่ 2.7



รูปที่ 2.7 ลำดับการทำงานของการเปลี่ยนโครงแบบ

โดยแต่ละขั้นตอนมีรายละเอียดดังนี้

Wake from Reset

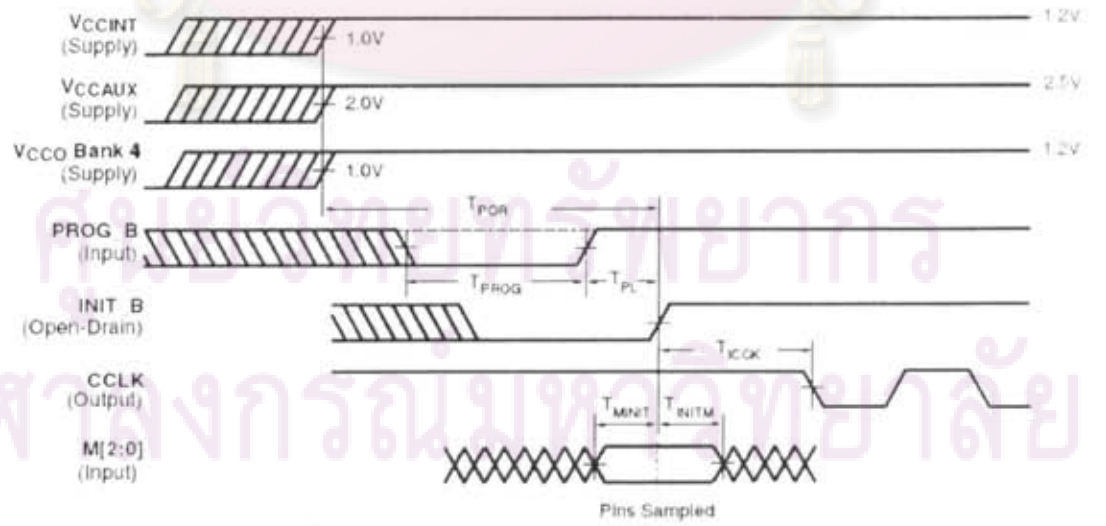
ขั้นตอนนี้สามารถเริ่มได้หลายวิธี ได้แก่

1. เมื่อเปิดเอฟพีจีเอ วงจร Power-On Reset (POR) ภายในเอฟพีจีเอจะทำให้เอฟพีจีเอรีเซตจนกว่าแรงดันไฟฟ้าจะถึงระดับหนึ่ง

2. ระบบส่งสัญญาณ PROG_B ต่ำเพื่อรีเซตเอฟพีจีเอ

3. รีเซตโดยคำสั่ง JPROGRAM ผ่าน JTAG

รูปที่ 2.8 และตารางที่ 2.2 แสดงความสัมพันธ์และเวลาของแรงดันไฟฟ้า, สัญญาณ INIT_B และสัญญาณ PROG_B



รูปที่ 2.8 กราฟแสดงเวลาเปิดเครื่องของเอฟพีจีเอ

ตารางที่ 2.2 แสดงเวลาเปิดเครื่องของเอฟพีจีเอ

สัญลักษณ์	คำอธิบาย	เวลา
T_{POR}	เวลาตั้งแต่แรงดันไฟฟ้ามีค่าถึงระดับที่กำหนดจนกระทั่งเอฟพีจีเอล้างหน่วยความจำโครงแบบเสร็จและสัญญาณ INIT_B เปลี่ยนเป็นสถานะสูง	5 – 7 มิลลิวินาที
T_{PL}	เวลาตั้งแต่สัญญาณ PROG_B เป็นสถานะสูงจนกระทั่งเอฟพีจีเอล้างหน่วยความจำโครงแบบเสร็จและสัญญาณ INIT_B เปลี่ยนเป็นสถานะสูง	2 – 3 มิลลิวินาที
T_{PROG}	ระยะเวลาน้อยที่สุดสำหรับพัลส์ของ PROG_B ที่ใช้ในการรีเซตเอฟพีจีเอ	300 นาโนวินาที
T_{ICCK}	เวลาตั้งแต่สัญญาณ INIT_B เปลี่ยนเป็นสถานะสูงจนกระทั่งเอฟพีจีเอเริ่มขับสัญญาณ CCLK สำหรับวิธีการเปลี่ยนโครงแบบที่ให้เอฟพีจีเอเป็นตัวควบคุม	0.5 – 4 ไมโครวินาที
T_{MINIT}	เวลาที่ใช้ในการตั้งค่าวิธีการเปลี่ยนโครงแบบ	50 นาโนวินาที

Clear Configuration Memory (Initialization)

หน่วยความจำโครงแบบจะถูกล้างโดยอัตโนมัติหลังจากขั้นตอน Wake from Reset ในระหว่างนี้อินพุต/เอาท์พุตจะเป็น High-Z ยกเว้นส่วนที่เกี่ยวกับการเปลี่ยนโครงแบบและ JTAG สัญญาณ INIT_B จะถูกดึงลงต่ำและจะถูกปล่อยเมื่อผ่านช่วง T_{POR} และ T_{PL} ตามรูปที่ 2.8 ถ้าสัญญาณ INIT_B ถูกบังคับให้เป็นสถานะต่ำได้จากภายนอก เอฟพีจีเอจะอยู่ในขั้นตอนนี้จนกว่าสัญญาณจะถูกปล่อย

Sample Control Pin

เอฟพีจีเอจะอ่านสัญญาณเลือกวิธีการเปลี่ยนโครงแบบ M[2:0] เมื่อสัญญาณ INIT_B กลับเป็นสถานะสูงหลังจากขั้นตอนการล้างหน่วยความจำโครงแบบ จากนั้นเอฟพีจีเอจะปล่อยสัญญาณนาฬิกา CCLK ถ้าเลือกให้เอฟพีจีเอเป็นผู้ควบคุมการเปลี่ยนโครงแบบ และเอฟพีจีเอจะเริ่มอ่านข้อมูลโครงแบบทุกๆขอบขาขึ้นของสัญญาณนาฬิกา

Synchronization

ภายในกระแสข้อมูลการเปลี่ยนโครงแบบของเอฟพีจีเอจะมี Synchronization Word อยู่ ซึ่งคำพิเศษนี้จะทำให้เอฟพีจีเอรู้ว่าข้อมูลถัดจากคำนี้เป็นข้อมูลโครงแบบและยังช่วยจัดเรียงข้อมูลโครงแบบให้ตรงกัน

ความยาวและเนื้อหาของ Synchronization Word จะแตกต่างกันออกไปตามรุ่นของเอฟพีจีเอ โดยเอฟพีจีเอรุ่น Spartan-3 จะมี Synchronization Word ยาว 32 บิต และมีเนื้อความแสดงโดยเลขฐานสิบหกเป็น 0xAA995566

Check Array ID

ในขั้นตอนนี้เอฟพีจีเอจะทำการตรวจสอบแถวข้อมูลไอดี (Array ID) ที่อยู่ภายในกระแสข้อมูลโครงแบบว่าตรงกับแถวข้อมูลไอดีภายในเอฟพีจีเอหรือไม่ เพื่อป้องกันการบรรจุข้อมูลโครงแบบลงบนเอฟพีจีเอผิดรุ่น

Load Configuration Data Frames

ในขั้นตอนนี้จะเป็นการบรรจุข้อมูลโครงแบบลงบนหน่วยความจำโครงแบบของเอฟพีจีเอ

Cyclic Redundancy Check

หลังจากบรรจุข้อมูลโครงแบบเสร็จสิ้น เอฟพีจีเอจะทำการคำนวณซีอาร์ซี (CRC) จากข้อมูลโครงแบบและตรวจสอบกับข้อมูลซีอาร์ซีภายในกระแสข้อมูล ถ้าผลลัพธ์ไม่ตรงกัน เอฟพีจีเอจะดึงสัญญาณ INIT_B ลงต่ำและยกเลิกการเปลี่ยนโครงแบบ

Startup

ในขั้นตอนนี้จะเป็นการเริ่มการทำงานของเอฟพีจีเอ ซึ่งในระหว่างการทำงานของขั้นตอนนี้สัญญาณ DONE จะถูกปล่อยให้เป็น High-Z

ขั้นตอนการเปลี่ยนโครงแบบทั้งหมดสามารถนำมาเขียนเป็นแผนภาพสายงานได้ดังแสดงในรูปที่ 2.9 และรูปที่ 2.10 โดยแผนภาพสายงานของวิธีอนุกรมและขนานจะเป็นดังรูปที่ 2.9 ส่วนวิธี Boundary-Scan จะเป็นดังรูปที่ 2.10

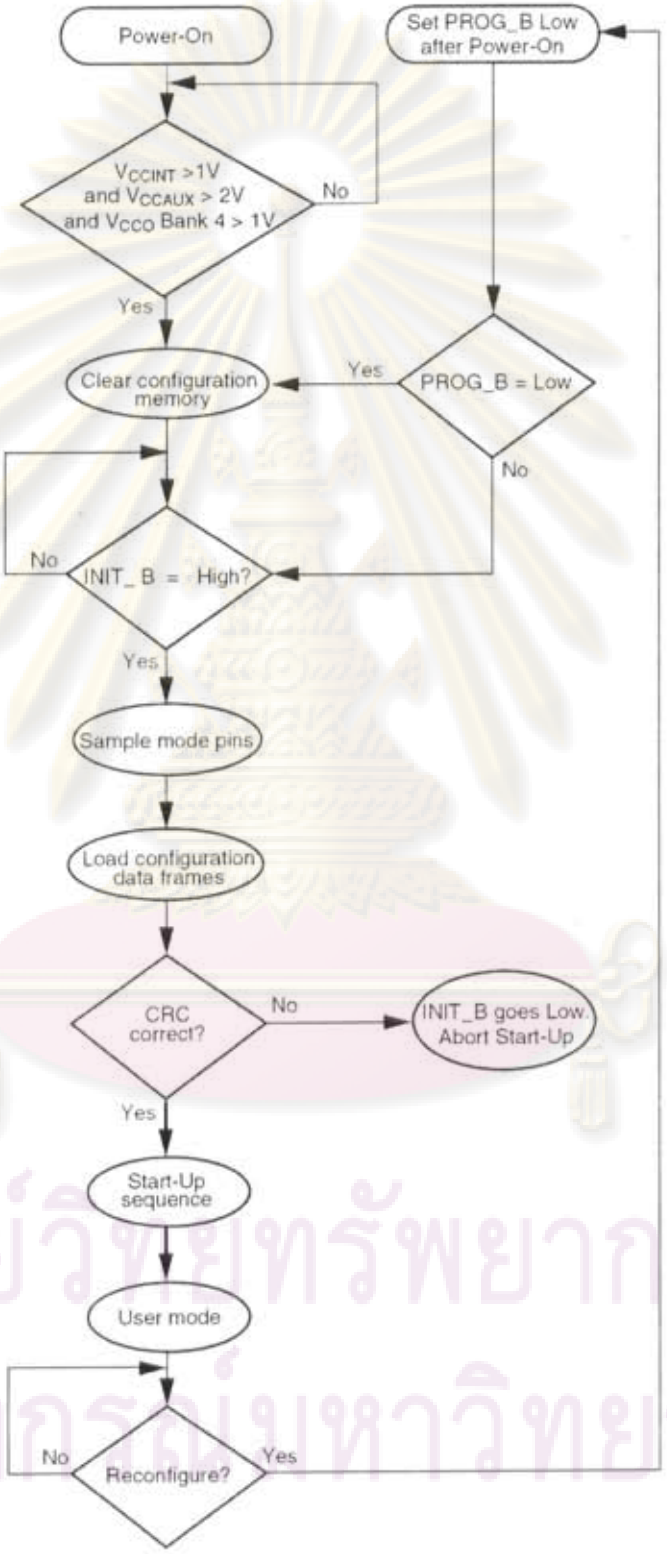
2.3. การเปลี่ยนโครงแบบบางส่วน (Partial Reconfiguration)

การเปลี่ยนโครงแบบบางส่วนขณะดำเนินงาน (active partial reconfiguration) หรือการเปลี่ยนโครงแบบบางส่วน (partial reconfiguration) คือการเปลี่ยนโครงแบบที่กระทำขณะอุปกรณ์กำลังทำงาน โดยบางส่วนของอุปกรณ์สามารถถูกเปลี่ยนโครงแบบ ในขณะที่ส่วนที่เหลือที่ไม่เกี่ยวข้องกับส่วนที่เปลี่ยนโครงแบบสามารถทำงานได้อย่างปกติและไม่ได้รับผลกระทบจากการเปลี่ยนโครงแบบ

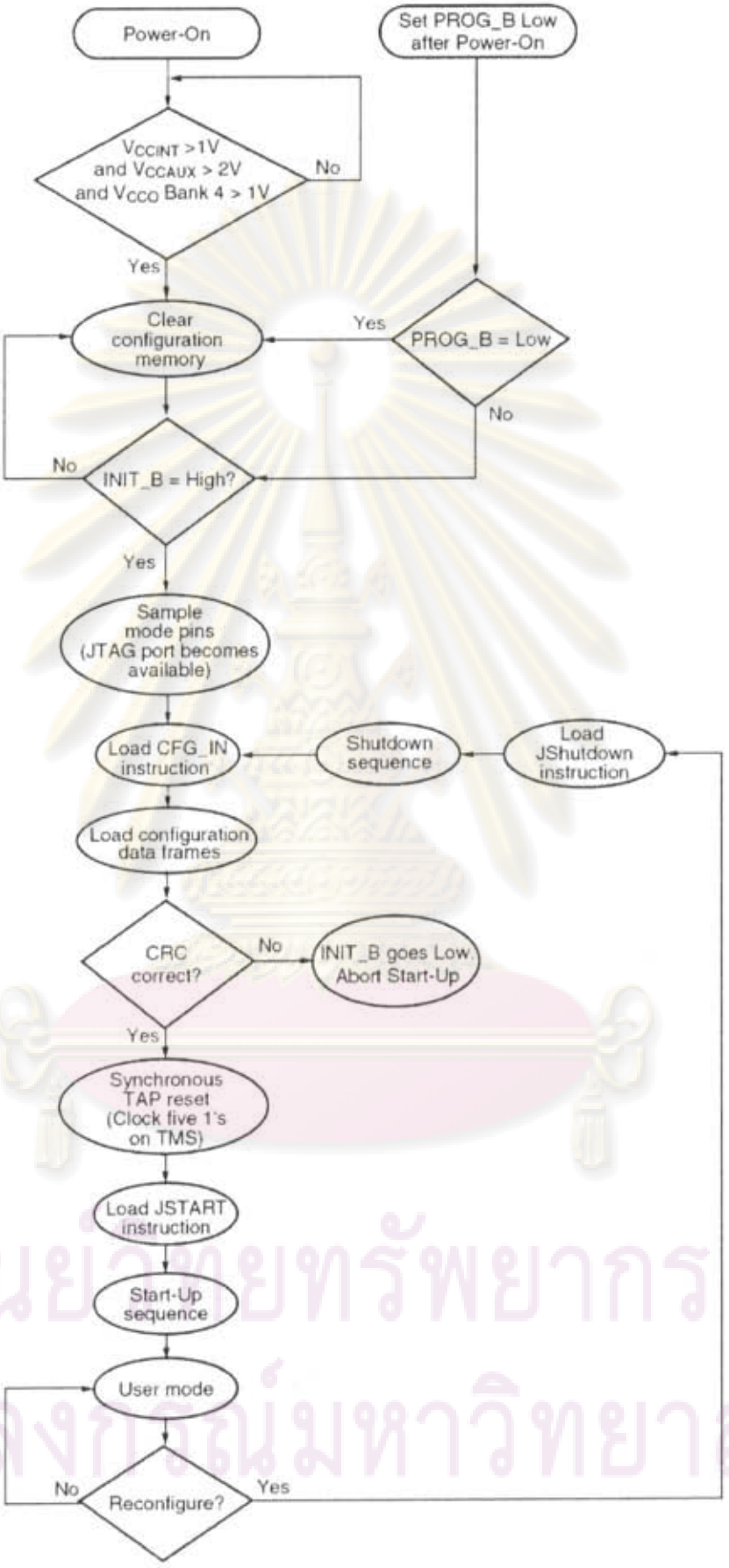
อุปกรณ์เอฟพีจีเอของ Xilinx มีรูปแบบการเปลี่ยนโครงแบบบางส่วนอยู่ 2 รูปแบบ [17] คือ

1. การใช้มอดูลเป็นฐาน (Module-Based) เป็นการเปลี่ยนโครงแบบทั้งมอดูล โดยจะแบ่งมอดูลต่างๆที่ต้องการเปลี่ยนโครงแบบออกตามการติดต่อระหว่างมอดูล โดยใช้ bus macro เป็นตัวกลางในการติดต่อระหว่างมอดูล เนื่องจาก bus macro จะไม่เปลี่ยนแปลงระหว่างการเปลี่ยนโครงแบบเพื่อรับประกันความถูกต้องของการเชื่อมต่อ

2. การใช้ความแตกต่างเป็นฐาน (Difference-Based) สามารถทำได้โดยเปลี่ยนแปลงเพียงเล็กน้อยในการออกแบบจากนั้นจึงสร้าง bitstream เฉพาะส่วนที่มีความแตกต่าง การเปลี่ยนโครงแบบในรูปแบบนี้สามารถทำได้อย่างรวดเร็ว เนื่องจาก bitstream ที่ได้มีขนาดเล็ก



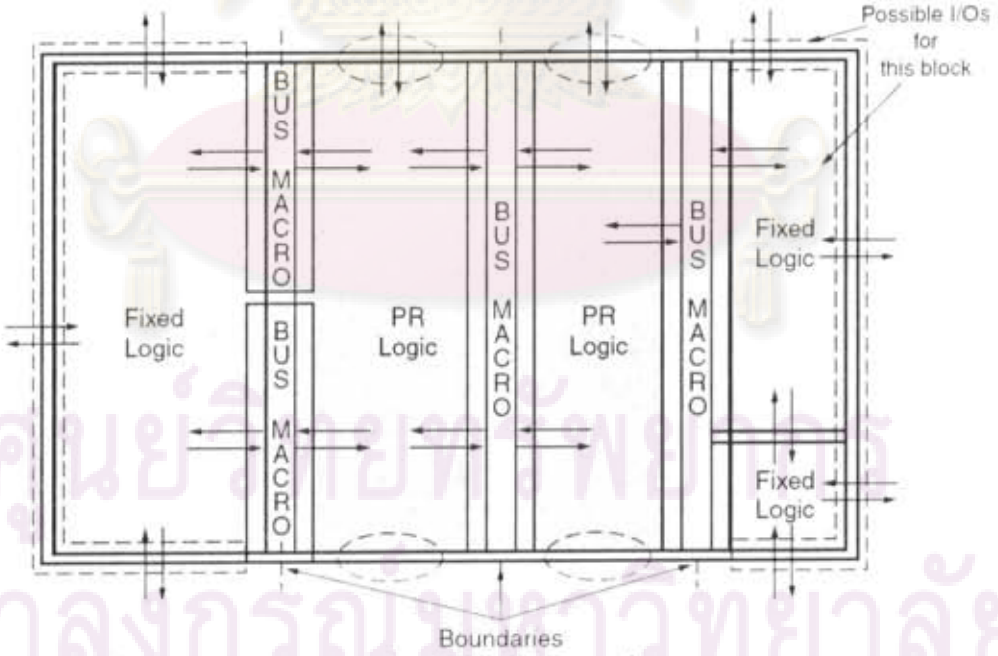
รูปที่ 2.9 แผนภาพสายงานของการเปลี่ยนโครงแบบด้วยวิธีอนุกรมและขนาน



รูปที่ 2.10 แผนภาพสายงานของการเปลี่ยนโครงแบบด้วยวิธี Boundary-Scan

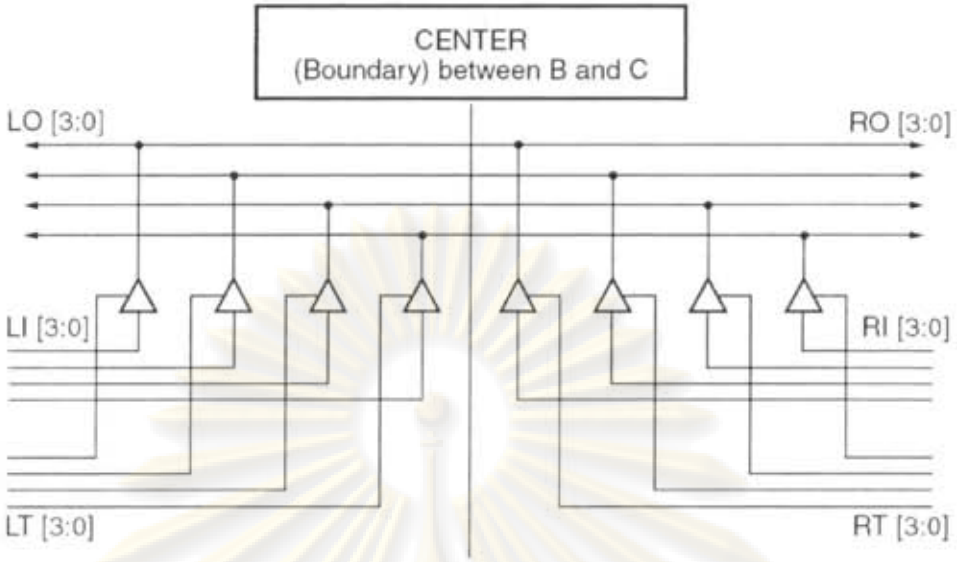
สำหรับการเปลี่ยนโครงแบบบางส่วนแบบ Module-Based นั้น มอดูลที่สามารถเปลี่ยนแปลงโครงแบบได้จะต้องมีคุณสมบัติดังต่อไปนี้

1. มอดูลที่สามารถเปลี่ยนแปลงโครงแบบได้จะต้องมีความสูงเท่ากับความสูงของอุปกรณ์
2. มอดูลที่สามารถเปลี่ยนแปลงโครงแบบได้จะต้องมีความกว้างอย่างน้อย 1 CLB (4 slices) และมากที่สุดเท่ากับความกว้างของอุปกรณ์ โดยขนาดจะต้องเป็นจำนวนเต็มของ CLB
3. ตำแหน่งตามแนวนอนของมอดูลที่สามารถเปลี่ยนแปลงโครงแบบได้จะต้องเป็นไปตามขอบเขตของ CLB (4 slices)
4. ทุกสิ่งที่อยู่ในขอบเขตของมอดูลที่สามารถเปลี่ยนแปลงโครงแบบได้ไม่ว่าจะเป็น Slice, TBUF, Block RAM, Multiplier, IOB และทรัพยากรเส้นทางเชื่อมโยง จะถูกรวมอยู่ในกรอบ bitstream ของมอดูลสามารถที่เปลี่ยนแปลงโครงแบบได้ ยกเว้นหน่วยตรรกะของสัญญาณนาฬิกา เพราะหน่วยเหล่านี้มีกรอบ bitstream เป็นของตัวเอง
5. IOB ที่อยู่ติดกับมอดูลที่สามารถเปลี่ยนแปลงโครงแบบได้ ถือเป็นส่วนหนึ่งของมอดูลที่สามารถเปลี่ยนแปลงโครงแบบได้
6. มอดูลที่สามารถเปลี่ยนแปลงโครงแบบได้จะติดต่อกับมอดูลอื่นด้วย bus macro ไม่ว่าจะเป็นมอดูลที่สามารถหรือไม่สามารถเปลี่ยนแปลงโครงแบบได้ก็ตาม
7. สถานะของหน่วยเก็บข้อมูลภายในมอดูลที่สามารถเปลี่ยนแปลงโครงแบบได้จะไม่เปลี่ยนแปลงขณะเปลี่ยนโครงแบบหรือภายหลังการเปลี่ยนโครงแบบ



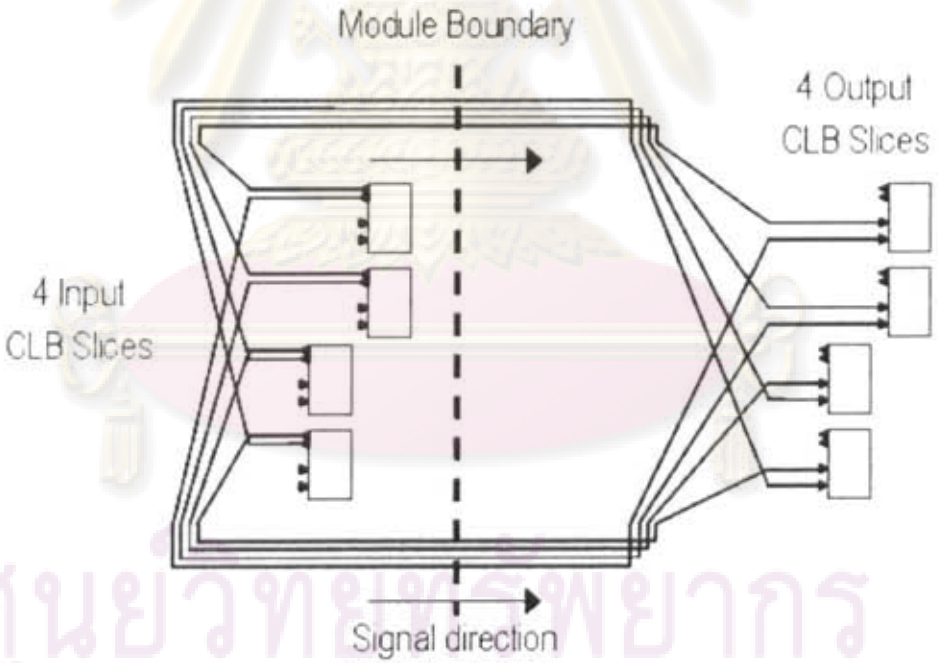
รูปที่ 2.11 ผังการออกแบบที่มีมอดูลที่สามารถเปลี่ยนโครงแบบได้ 2 มอดูล

ตามปกติแล้ว bus macro จะถูกสร้างมาจาก TBUF (3-state buffer) ดังรูปที่ 2.12 แต่วิธีนี้ไม่สามารถใช้ได้กับ Xilinx เอฟพีจีเอรุ่น Spartan-3 เนื่องจาก Spartan-3 ไม่มี TBUF



รูปที่ 2.12 การสร้าง bus macro ด้วย 3-state buffer

ในงานวิจัยของ Patrick Lysaght, Brandon Blodget, Jeff Mason, Jay Young และ Brenden Bridgford [18] ได้นำเสนอวิธีการสร้าง bus macro โดยใช้ Slice แทน TBUF ดังรูปที่ 2.13 ซึ่งวิธีนี้ทำให้เราสามารถสร้าง bus macro บน Xilinx เอฟพีจีเอรุ่น Spartan-3 ได้



รูปที่ 2.13 การสร้าง bus macro ด้วย CLB

2.4. เออีเอส (AES: Advance Encryption Standard)

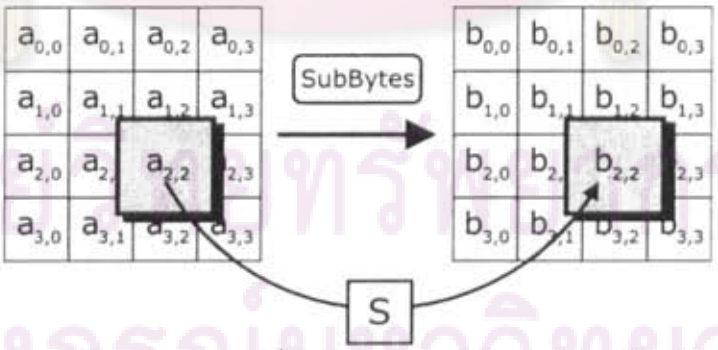
เออีเอส (AES: Advance Encryption Standard) [6] มีพื้นฐานมาจากทฤษฎี Rijndael [22] โดยเออีเอสจะมีความยาวของคีย์และข้อความเป็น 128, 192 หรือ 256 บิต ขั้นตอนการทำงานของการเข้ารหัสเออีเอสประกอบด้วย

1. ขั้นตอนการขยายคีย์ (KeyExpand)
2. ขั้นตอนการแทนที่ไบต์ (ByteSub)
3. ขั้นตอนการเลื่อนแถว (ShiftRow)
4. ขั้นตอนการผสมหลัก (MixColumn)
5. ขั้นตอนการบวกคีย์แต่ละรอบ (AddRoundKey)

ขั้นตอนการขยายคีย์ (KeyExpand) เป็นกระบวนการเพื่อหาคีย์ที่ใช้เพื่อเปลี่ยนสถานะ (State: สถานะของข้อความ เมื่อข้อความมีการเปลี่ยนแปลงโดยกระบวนการใดๆก็ตาม สถานะก็จะเปลี่ยนไป) ของข้อความที่เข้ามาจนได้เป็นผลลัพธ์สุดท้ายออกไป ซึ่งจะหาคีย์ทั้งหมด 10 รอบด้วยกันโดยระเบียบวิธีเป็น

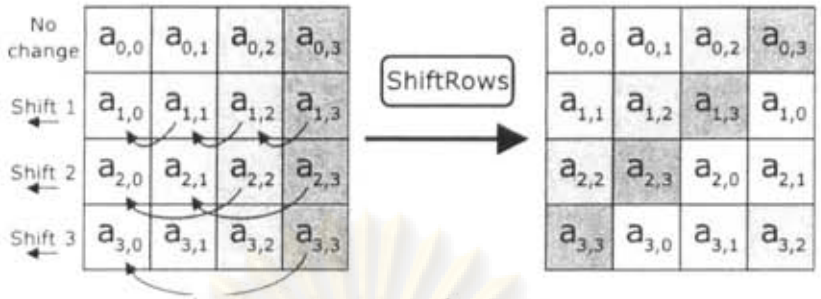
```
Nk = 4:
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)])
{
      for(i = 0; i < Nk; i++)
            W[i] = (Key[4*i],Key[4*i+1],Key[4*i+2],Key[4*i+3]);
      for(i = Nk; i < Nb * (Nr + 1); i++)
      {
            temp = W[i - 1];
            if (i % Nk == 0)
            temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];
            W[i] = W[i - Nk] ^ temp;
      }
}
*Nk - ความยาวของคีย์หารด้วย 32 , Nb - ความยาวของ block (ข้อความ) หารด้วย 32
*SubByte เป็นฟังก์ชันที่สร้างโดยตาราง look-up ซึ่งจะคืนค่าคงที่กลับมา
*Rcon เป็นค่าคงที่ที่ใช้ในสร้างคีย์ในแต่ละรอบ
```

ขั้นตอนการแทนที่ไบต์ (ByteSub) เป็นการแทนค่าเดิมด้วยวิธีการทำคูณแบบผกผัน (Multiplicative Inverse) บนฟังก์ชัน GF(28) หรือวิธีใช้ตารางแทนค่าโดยการนำข้อมูลเข้าไปยังมอดูลที่ชื่อว่าเอสบ็อกซ์ (S-box) ดังรูปที่ 2.14



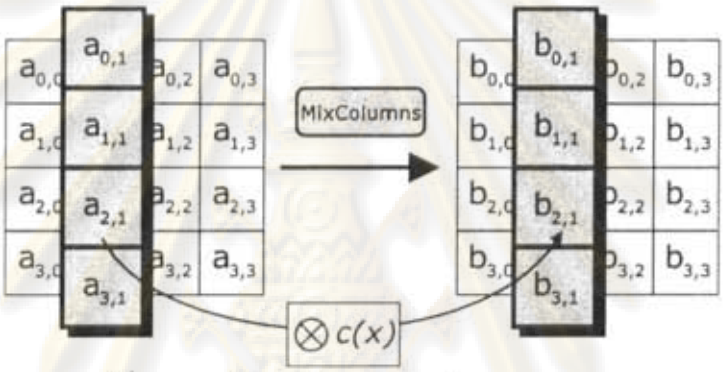
รูปที่ 2.14 ระเบียบวิธีการแทนที่ไบต์ (ByteSub) โดยอาศัยเอสบ็อกซ์ (S-box)

ขั้นตอนการเลื่อนแถว (ShiftRow) เป็นสถานะที่ทำการเลื่อนซ้ายแบบเป็นวงหรือ cyclic left shift โดยจำนวนครั้งที่เลื่อนขึ้นอยู่กับแถวนั้นๆ ดังแสดงในรูปที่ 2.15



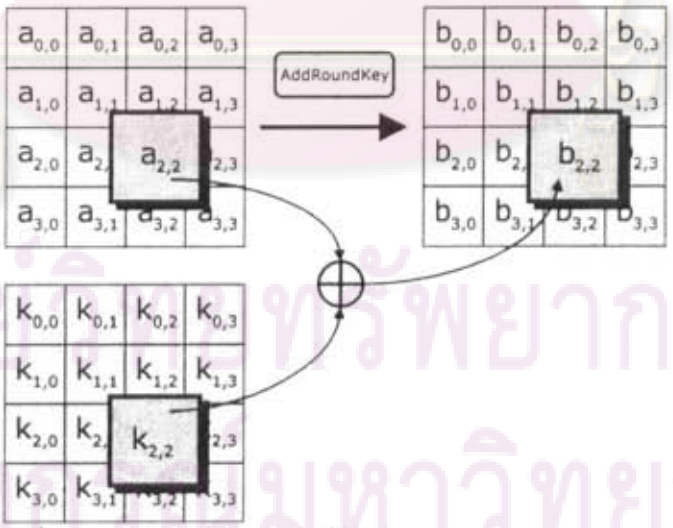
รูปที่ 2.15 ระเบียบวิธีการเลื่อนแถว (ShiftRow)

ขั้นตอนการผสมหลัก (MixColumn) คือการนำหลักแต่ละหลักของอินพุตมาพิจารณาแบบ GF(28) แล้วนำไปคูณมอดูโล (Multiply modulo) ด้วย x4 + 1 กับพหุนาม '03'x3 + '01'x2 + '01'x + '02' ดังรูปที่ 2.16



รูปที่ 2.16 ระเบียบวิธีการผสมหลัก (MixColumn)

ขั้นตอนการบวกคีย์แต่ละรอบ (AddRoundKey) เป็นกระบวนการเปลี่ยนสถานะของข้อความโดยการทำ XOR กับคีย์ในรอบนั้นๆดังที่แสดงในรูปที่ 2.17



รูปที่ 2.17 ระเบียบวิธีการบวกคีย์แต่ละรอบ (AddRoundKey)

ลำดับการทำงานในรอบที่ 1 – 9 จะใช้ RoundKey(State, RoundKey) เมื่อสิ้นสุดแต่ละรอบก็จะใช้ KeyExpand(RoundKey, Rcon) ในการหา RoundKey ของรอบถัดไป

```
Round(State,RoundKey)
{
      ByteSub(State);
      ShiftRow(State);
      MixColumn(State);
      AddRoundKey(State,RoundKey);
}
```

ส่วนการทำงานรอบสุดท้ายจะละการทำ MixColumn(State) แล้วจบการทำงานด้วย AddRoundKey(State)

```
FinalRound(State,RoundKey)
{
      ByteSub(State);
      ShiftRow(State);
      AddRoundKey(State,RoundKey);
}
```

สำหรับการถอดรหัสมีขั้นตอนกระบวนการที่ไม่แตกต่างจากการเข้ารหัสเท่าไรนัก เพียงแค่เปลี่ยนลำดับการทำงานของแต่ละรอบเท่านั้นดังนี้

```
InvRound(State,RoundKey)
{
      AddRoundKey(State,RoundKey);
      InvMixColumn(State);
      InvShiftRow(State);
      InvByteSub(State);
}
```

เช่นเดียวกับการเข้ารหัส การถอดรหัสในรอบสุดท้ายก็มีความแตกต่างเพียงเล็กน้อย

```
InvRound(State,RoundKey)
{
      AddRoundKey(State,RoundKey);
      InvShiftRow(State);
      InvByteSub(State);
}
```

โดยฟังก์ชั่นที่ขึ้นต้นด้วย Inv มีการทำงานตรงข้ามกับฟังก์ชั่นที่ไม่มี Inv นำหน้า

นอกจากนี้ในการนำไปใช้งานจริงจำเป็นต้องมีวิธีการดำเนินการ (Operation) มาเกี่ยวข้อง โดยทั่วไปแล้วการดำเนินการมีอยู่ 5 โหมดได้แก่ อีบีซี (ECB: Electronic Codebook), ซีบีซี (Cipher Block Chaining), ซีเอฟบี (CFB: Cipher Feedback), โอเอฟบี (OFB: Output Feedback) และ ซีทีอาร์ (CTR: Counter)

2.4.1. อีซีบี (ECB: Electronic Codebook)

เป็นการดำเนินการแบบพื้นฐานที่สุด ทำโดยการแบ่งข้อความที่เข้ามาให้มีขนาดพอเหมาะกับบล็อคไซเฟอร์จากนั้นเข้ารหัสแล้วก็จะได้ผลลัพธ์

กำหนดให้ P เป็นข้อความใหญ่ที่แยกออกเป็นข้อความย่อยๆดังนี้

$$P = \{P_1, P_2, \ldots, P_L\}$$

เมื่อข้อความย่อยเข้ารหัสด้วยฟังก์ชั่น E_k ได้เป็นข้อความที่ถูกเข้ารหัสแล้ว

$$C = \{C_1, C_2, \ldots, C_L\}$$

โดย $C_j = E_K(P_j)$ คือการเข้ารหัสของ P_j โดยใช้คีย์ K

ข้อเสียของการดำเนินการโหมดอีซีบีคือการถูกโจมตีได้ง่าย ผู้ไม่หวังดีสามารถทำการถอดรหัสได้โดยง่ายโดยอาศัยจุดอ่อนจากการสังเกตการเข้ารหัสที่ได้ผลลัพธ์เหมือนเดิมตลอด

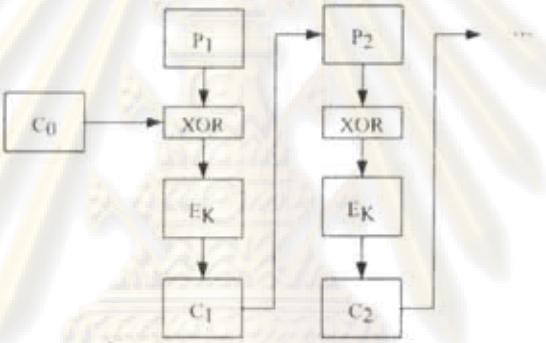
2.4.2. ซีบีซี (CBC: Cipher Block Chaining)

การดำเนินการประเภทนี้ถูกสร้างเพื่อปิดช่องโหว่ที่เกิดขึ้นในอีซีบีด้วยการเข้ารหัสแบบต่อเนื่อง ข้อความขาออกชุดต่อไปที่ถูกนำมาเข้ารหัสมีค่าเป็นเท่าไรก็ขึ้นอยู่กับข้อความในรอบที่แล้วด้วย ส่งผลให้ผู้ไม่หวังดีถอดรหัสได้ยากขึ้นมีวิธีการเข้าและถอดรหัสดังนี้

เข้ารหัส $C_j = E_K(P_j \text{ xor } C_{j-1})$

ถอดรหัส $P_j = D_K(C_j) \text{ xor } C_{j-1}$

แสดงได้เป็นดังรูปที่ 2.18



รูปที่ 2.18 การดำเนินการโหมดซีบีซี

2.4.3. ซีเอฟบี (CFB: Cipher Feedback)

เป็นโหมดการดำเนินการชนิดที่กระทำเป็นแบบสตรีม (Stream) ขนาด k บิต มีหลักการทำงานดังนี้ เริ่มแรกข้อความที่เข้ามาจะถูกแยกออกเป็นข้อความย่อยขนาด 8 บิต (ในที่นี้ขอยกตัวอย่างในกรณี k=8 บิต) จากนั้นกำหนด X1 ขนาดเท่ากับขนาดของบล็อคไซเฟอร์ที่เราใช้ให้ค่าเริ่มต้น (ในที่นี้ขอใช้เป็น 64 บิต) จะได้ว่า

$$O_j = L_8(E_K(X_j))$$

$$C_j = P_j \text{ xor } O_j$$

$$X_{j+1} = R_{56}(X_j) || C_j$$

โดยที่ $L_8(X)$ บิตซ้ายสุด 8 บิตของ X

$R_{56}(X)$ บิตขวาสุด 56 บิตของ X

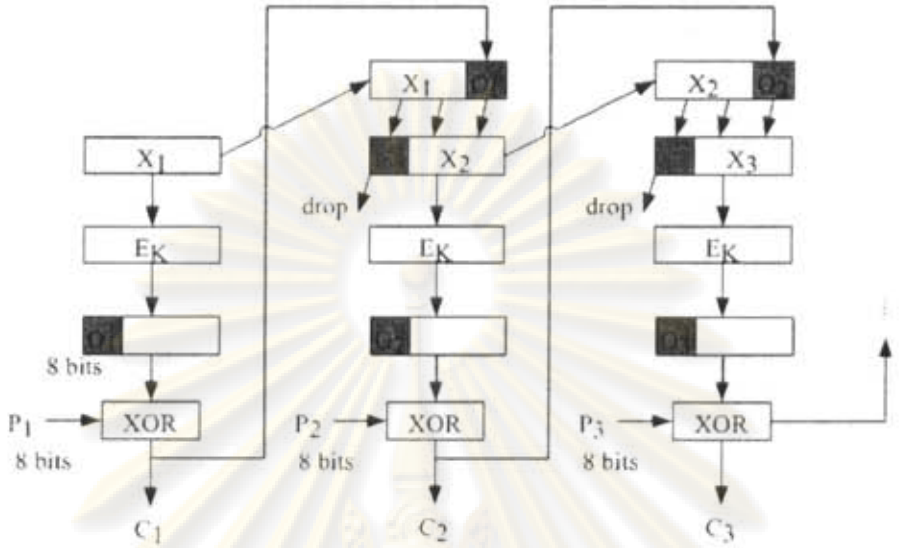
$X||Y$ คือการนำสตริง X มาต่อกับ Y

สามารถสรุปได้เป็นดังรูปที่ 2.19

ในโหมดนี้การถอดรหัสจะใช้สมการดังนี้

$$P_j = C_j \text{ xor } L_8(E_K(X_j))$$

$$X_{j+1} = R_{56}(X_j)||C_j$$



รูปที่ 2.19 การดำเนินการโหมดซีเอฟบี

เห็นได้ว่าไม่มีการใช้ฟังก์ชั่นถอดรหัสมาเกี่ยวข้องเนื่องจากฟังก์ชั่นถอดรหัสใช้เวลาทำงานที่มากกว่าฟังชั่นการเข้ารหัส ข้อเสียของวิธีดำเนินการแบบนี้คือเมื่อข้อมูลที่ผู้รับรับได้เกิดความผิดพลาดขึ้นก็จะทำให้ข้อมูลที่ถูกส่งต่อถัดมาผิดพลาดไปด้วย

2.4.4. โอเอฟบี (OFB: Output Feedback)

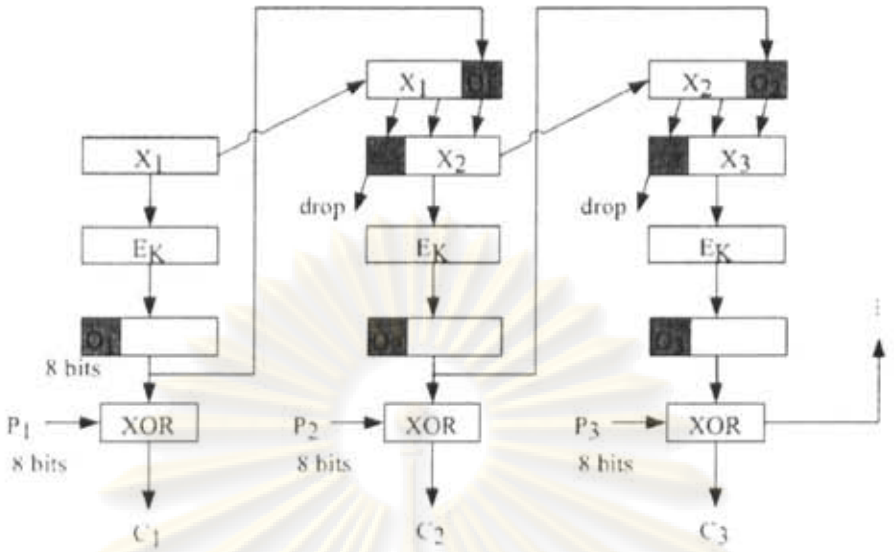
ลักษณะการทำงานคล้ายๆกับซีเอฟบีสามารถแสดงด้วยสมการดังนี้

$$O_j = L_8(E_K(X_j))$$

$$X_{j+1} = R_{56}(X_j)||O_j$$

$$C_j = P_j \text{ xor } O_j$$

สิ่งที่แตกต่างกันคือการต่อสตริงด้วย O_j จุดประสงค์ก็เพื่อเป็นการปิดกั้นไม่ให้ความผิดพลาดในการรับส่งถูกส่งต่อไปในข้อมูลที่ตามมาภายหลัง ดังรูปที่ 2.20



รูปที่ 2.20 การดำเนินการโหมดโอเอฟบี

ข้อเสียของการดำเนินการโหมดนี้คือสามารถถูกโจมตีได้ง่ายเช่นในกรณีที่ผู้ไม่หวังดีรู้ถึง P_j และ C_j ก็รู้ถึง O_j ได้จากสมการ

$$O_j = C_j \text{ xor } P_j$$

เมื่อรู้ O_j ก็สามารถสร้างข้อความเท็จขึ้นมาได้จาก

$$C'_j = P'_j \text{ xor } O_j$$

2.4.5. ซีทีอาร์ (CTR: Counter)

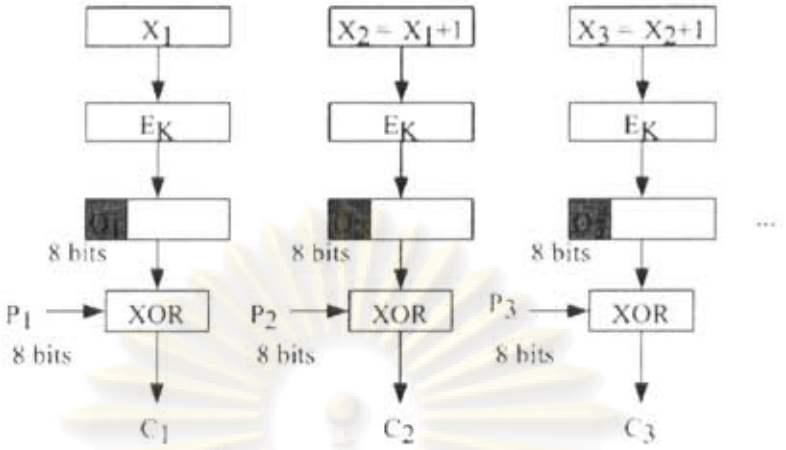
วิธีดำเนินการนี้เหมาะสำหรับกับการนำไปประยุกต์ในการทำงานแบบขนานเพราะข้อความถูกเข้ารหัสไปพร้อมๆกันไม่ต้องรอข้อความก่อนหน้า สมการการดำเนินการเป็นดังนี้

$$X_j = X_{j-1}+1$$

$$O_j = L_s(E_k(X_j))$$

$$C_j = P_j \text{ xor } O_j$$

ดังรูปที่ 2.21

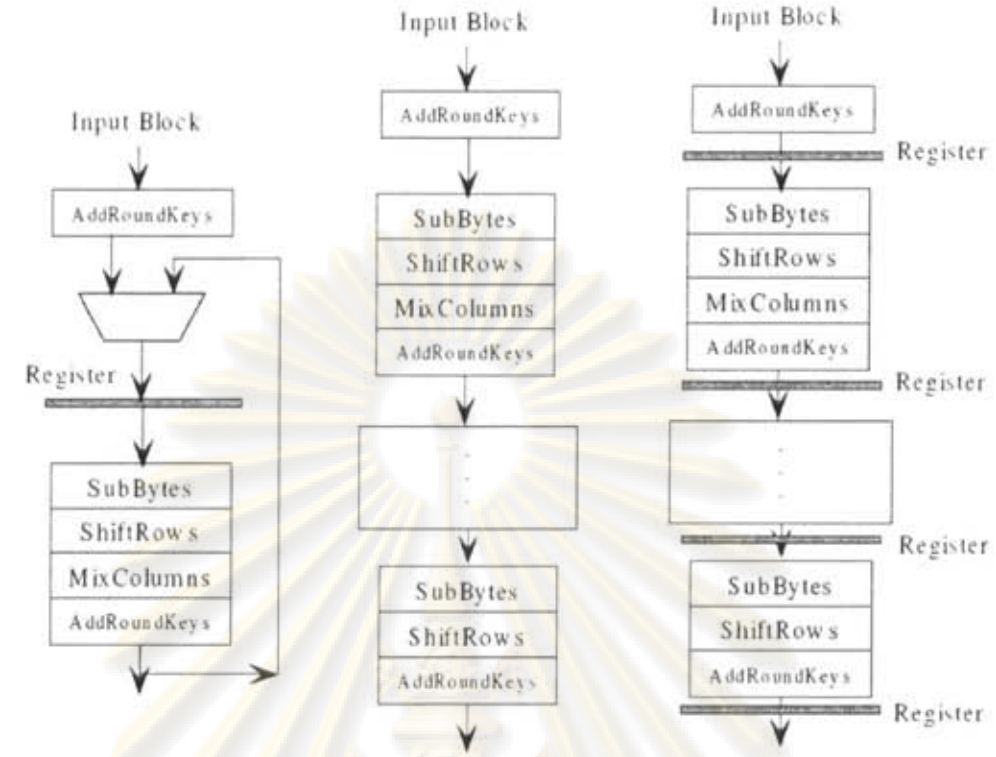


รูปที่ 2.21 การดำเนินการโหมดซีทีอาร์

2.5. งานวิจัยที่เกี่ยวข้อง

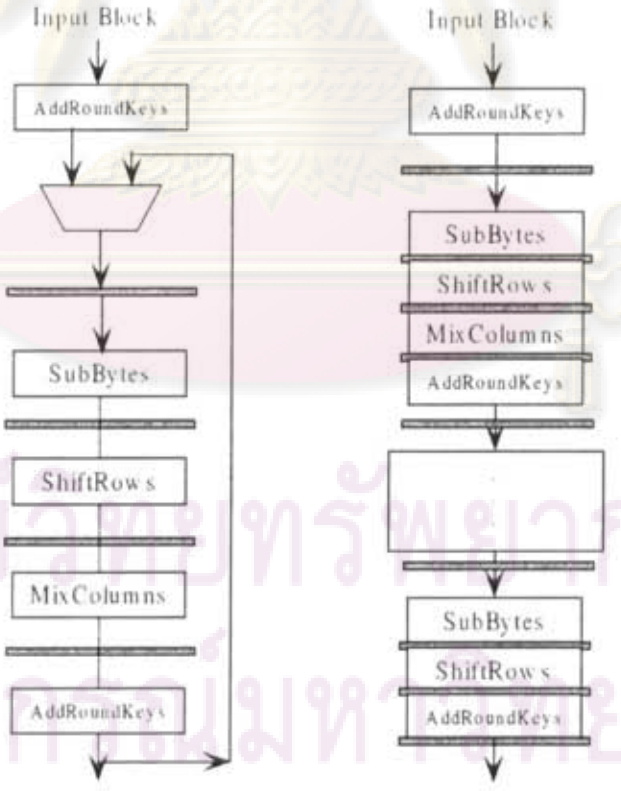
2.5.1. การออกแบบหน่วยประมวลผลเออีเอสแบบตัวนับแบบเปลี่ยนโครงแบบได้ที่มีสมรรถนะสูงอย่างยิ่ง (Design of an Extremely High Performance Counter Mode AES Reconfigurable Processor)

งานวิจัยของ Yongzhi Fu, Lin Hao, Xuejie Zhang และ Rujin Yang [7] นี้ได้เสนอวิธีการสร้างวงจรเออีเอสแบบตัวนับ (counter mode) บน Xilinx เอฟพีจีเอรุ่น Virtex-II โดยในงานวิจัยนี้ใช้วิธีการ loop unrolling, inner round pipelining และ outer round pipelining ซึ่งมีรายละเอียดดังแสดงในรูปที่ 2.22 และรูปที่ 2.23 ซึ่งวงจรสุดท้ายที่ได้มีสมรรถนะสูงมากถึง 27.1 กิกะบิตต่อวินาที แต่ก็ใช้ทรัพยากรสูงมากด้วยเช่นกัน



a. General Architecture b. Loop Unrolled Architecture c. Outer Round Pipelining

รูปที่ 2.22 การปรับปรุงการออกแบบวงจรเออีเอส 1



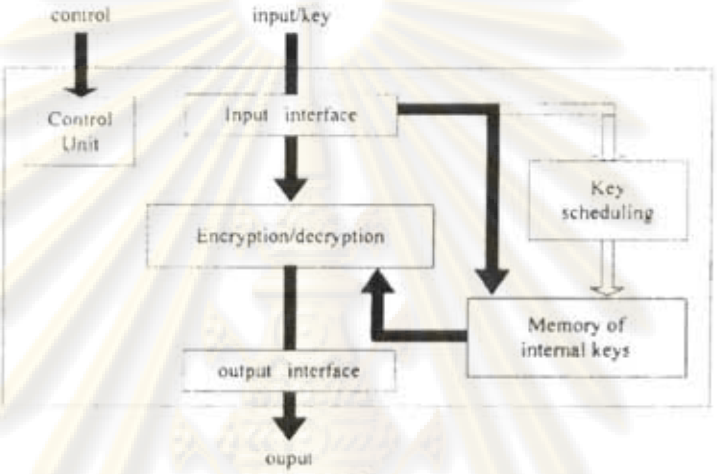
a. Inner Round Pipelining b. Mixed Inner an Outer Round Pipelining

รูปที่ 2.23 การปรับปรุงการออกแบบวงจรเออีเอส 2

2.5.2. การออกแบบและสร้างแกนไอพีเออีเอสแบบเปลี่ยนโครงแบบได้ด้วยเอฟพีจีเอ

(Design and Implementation of Reconfigurable AES IP Core using FPGAs)

งานวิจัยของ Xu Jian, Liu Yuan-feng, Dai Zi Bin และ Sun Yi [8] ได้เสนอแกนไอพี (IP Core) ที่ใช้เข้าหรือถอดรหัสเออีเอส โดยแกนไอพีนี้สามารถทำงานได้ทั้งในแบบ 128 บิต, 192 บิต และ 256 บิต และด้วยการออกแบบโดยอาศัยการเปลี่ยนโครงแบบทำให้ประหยัดหน่วยตรรกะ (Logic Element) ได้ถึงร้อยละ 43 และประหยัดบิตหน่วยความจำ (Memory Bit) ได้ร้อยละ 17 สถาปัตยกรรมจะเป็นดังรูปที่ 2.24



รูปที่ 2.24 แกนไอพีวงจรเข้ารหัสเออีเอสแบบเปลี่ยนโครงแบบได้ (Reconfigurable AES IPCore)

เนื่องจากต้องการประหยัดทรัพยากรที่ใช้งาน ดังนั้นงานวิจัยชิ้นนี้จึงมีการประยุกต์ระบบการเปลี่ยนโครงแบบเข้าไปใช้กับหน่วยเอสบ็อกซ์ (S-Box unit) โดยค่าในเอสบ็อกซ์จะเปลี่ยนไปให้สอดคล้องกับการเข้าหรือถอดรหัสในขณะนั้นเพื่อลดทรัพยากรที่ใช้ ซึ่งสมรรถนะที่ได้เป็นดังตารางที่ 2.3 โดยจะเห็นว่าแกนไอพีในแบบสุดท้ายจะประหยัดทรัพยากรได้มากที่สุดเนื่องจากอาศัยเอสบ็อกซ์ที่เปลี่ยนโครงแบบได้

ตารางที่ 2.3 สมรรถนะของแกนไอพีเออีเอส

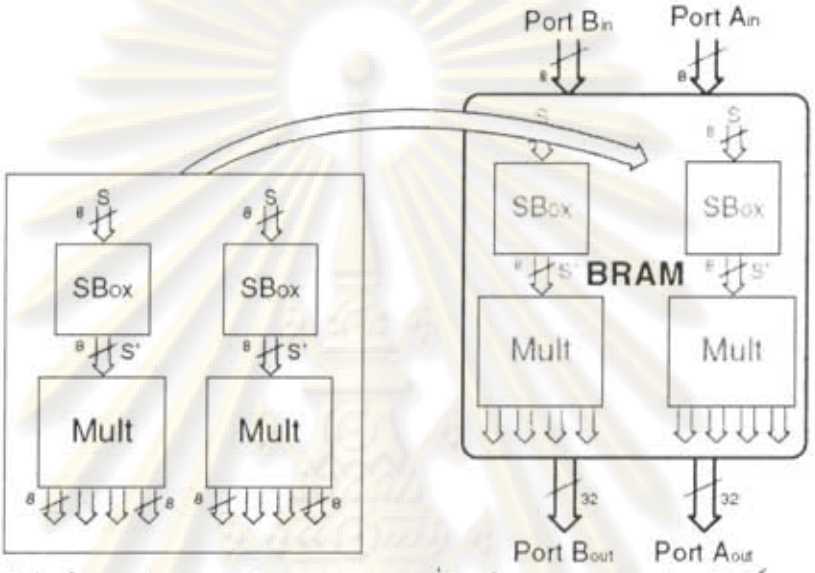
IP-CORE support key lengths(bits)	**System Clock fmax(MHz)**	**Area**	
		Memory bits	LEs
128	70.11	65536	1550
192	65.34	65536	2052
256	61.26	65536	2554
128/192/256	60.34	32768	2612

* 2 LEs เทียบเท่า 1 CLB

2.5.3. หน่วยประมวลผลร่วมเออีเอสแบบหน่วยความจำที่เปลี่ยนโครงแบบได้

(Reconfigurable Memory Based AES Co-Processor)

งานวิจัยของ Ricardo Chaves, Georgi Kuzmanov, Stamitis Vassiliadis และ Leonel Sousa [9] นี้ได้เสนอวงจรเข้าและถอดรหัสเออีเอสที่พัฒนาขึ้นบน Xilinx เอฟพีจีเอรุ่น Virtex-II Pro โดยอาศัยบล็อกหน่วยความจำ (BRAM) แบบช่องทางคู่ (dual port) ในการพัฒนาสมรรถนะของการเข้าและถอดรหัส



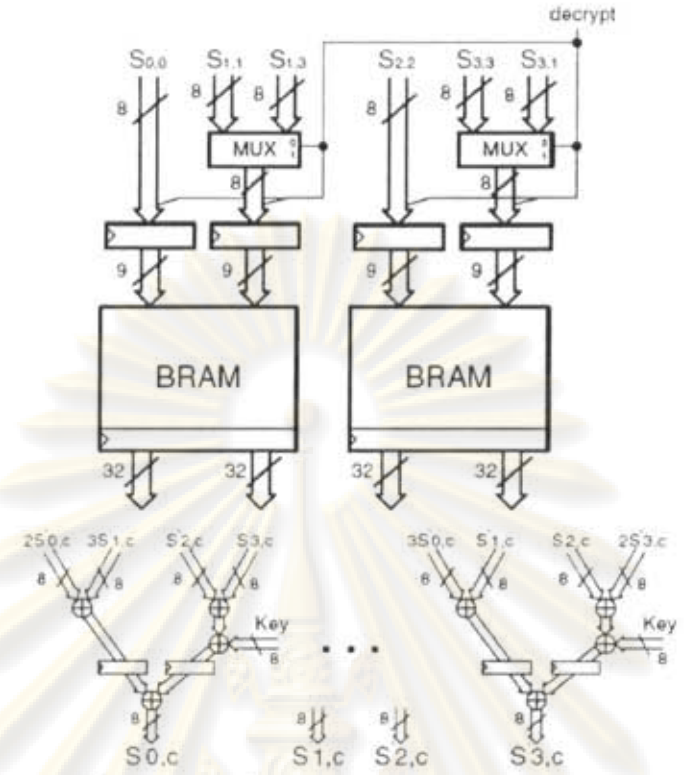
รูปที่ 2.25 การใช้บล็อกหน่วยความจำแทนการแทนที่ไบต์และการหาสัมประสิทธิ์ของการผสมหลัก

จากรูปที่ 2.25 จะเห็นว่างานวิจัยนี้ได้ใช้บล็อกหน่วยความจำแบบช่องทางคู่ 1 บล็อกแทนการแทนที่ไบต์ 2 อันและการหาสัมประสิทธิ์ที่ต้องใช้ในการผสมหลัก 2 อันเนื่องจากการเลื่อนแถวและการแทนที่ไบต์สามารถสลับกันได้ ซึ่งจะได้วงจรที่ทำการแทนที่ไบต์, เลื่อนแถว และผสมหลักสำหรับการเข้าและถอดรหัสดังรูปที่ 2.26

จากแนวคิดข้างต้น ทำให้ได้วงจรที่มีสมรรถนะสูงและใช้ทรัพยากรต่ำดังแสดงให้เห็นในตารางที่ 2.4

ตารางที่ 2.4 สมรรถนะและทรัพยากรที่ใช้ของวงจรเออีเอส

	Slices	**BRAM**	**Frequency (MHz)**	**Throughput (GBPs)**	**Latency (cycles)**
folded	515(5%)	12	182	2.33	10
unfolded (inter-pipelining)	3168(32%)	80	156	19.95	10
unfolded (intra-pipelining)	3513(36%)	80	271	34.7	30



รูปที่ 2.26 ส่วนหนึ่งของวงจรเข้าและถอดรหัสเออีเอส

2.5.4. การเปรียบเทียบกลวิธีที่ใช้ในการสร้างวงจรเข้ารหัสแบบเออีเอสบนเอฟพีจีเอ

(Comparison of various strategies of implementation of the algorithm of encryption AES on FPGA)

ในงานวิจัยของ Oscar Perez, Yves Berviller, Camel Tanougast และ Serge Weber [10] นี้ ได้นำเสนอวงจรเข้ารหัสแบบเออีเอส 128 บิตด้วยวิธีการทำงานแบบสายท่อ (Pipelining) และการเปลี่ยนโครงแบบ (Reconfiguration) โดยใช้ Xilinx เอฟพีจีเอ XC2V6000 ซึ่งในการวิจัยนี้จะแบ่งส่วนของวงจรเข้ารหัสออกเป็นสองมอดูลเพื่อใช้ในการเปลี่ยนโครงแบบ คือ

1. มอดูลขยายคีย์ (Expansion Ki Module)
2. มอดูลไซเฟอร์ (Cipher Module)

โดยทั้งสองส่วนนี้จะถูกโปรแกรมลงบนเอฟพีจีเอเมื่อจำเป็นต้องใช้งานเท่านั้น นอกจากนี้ยังได้ทดลองสร้างวงจรเข้ารหัสโดยใช้แนวคิดการเปลี่ยนโครงแบบออกมาสองแบบ ได้แก่ Reconf2 และ Reconf3 ซึ่งทั้งสองแบบนี้มีข้อแตกต่างกัน คือ Reconf2 จะมีการกระจายลูป (Unrolling Loop) แต่ Reconf3 ไม่ทำ ซึ่งผลลัพธ์ที่ได้จากวงจรทั้งสองเป็นดังตารางที่ 2.5

ตารางที่ 2.5 ประสิทธิภาพของวงจร Reconf2 และ Reconf3

	CLBs	**BRAMs**	**Throughput (Mbps)**
Reconf2	2905	0	43448
Reconf3	321	0	4273

แต่ในงานวิจัยนี้ ไม่ได้กล่าวถึงวิธีและจำนวนบิตที่ใช้ในการเปลี่ยนโครงแบบ แต่ได้คำนวณเวลาที่ใช้ในการเปลี่ยนโครงแบบเอาไว้

2.5.5. วงจรเข้ารหัสเออีเอสชนิดเปลี่ยนโครงแบบได้

ในงานวิจัยของเจนโชติ ศรีพรประเสริฐ และประภาส จงสถิตย์วัฒนา [12], [13] ได้เสนอวงจรเข้ารหัสเออีเอสชนิดเปลี่ยนโครงแบบได้ โดยใช้มัลติเพลกเซอร์ช่วยในการเปลี่ยนโครงแบบ โดยจะแบ่งวงจรออกเป็น 6 ส่วน ดังนี้

1. หน่วยควบคุม (Control Unit) ใช้ควบคุมการทำงานทั้งหมด
2. รีจิสเตอร์แบงก์ (RegisterBank) ใช้เก็บข้อมูลสถานะและคีย์
3. วงจรสลับเปลี่ยน (Transposer) ทำหน้าที่สลับข้อมูลที่จะเข้าวงจรไซเฟอร์เอสบี
4. วงจรขยายคีย์ (KeyExpansion) ทำกระบวนการขยายคีย์
5. วงจรไซเฟอร์เอสบี (CipherSB) ทำกระบวนการแทนที่ไบต์
6. วงจรไซเฟอร์เอ็มเอ็กซ์ (CipherMX) ทำกระบวนการเลื่อนแถว, กระบวนการผสมหลัก และกระบวนการบวกคีย์แต่ละรอบ

วงจรนี้ใช้ทรัพยากรเพียง 10,830 เกตสมมูล และสามารถทำงานได้ที่ความถี่สูงสุด 75.6 เมกะเฮริตซ์ โดยใช้จำนวนสัญญาณนาฬิกาทั้งสิ้น 202 รอบในการเข้ารหัส

แต่ในงานวิจัยชิ้นนี้ไม่ได้ทำการเปลี่ยนโครงแบบของเอฟพีจีเอ แต่ใช้มัลติเพลกเซอร์ช่วยในการจำลองการเปลี่ยนโครงแบบ นั่นหมายความว่าถ้ามีการเปลี่ยนโครงแบบจริง อาจจะต้องใช้ทรัพยากรมากขึ้น หรือมีสมรรถนะลดลงได้

จากงานวิจัยที่เกี่ยวข้องทั้งหมดข้างต้น จะเห็นว่ามีงานวิจัยหลายชิ้นที่นำเอาแนวคิดของการเปลี่ยนโครงแบบมาใช้เพื่อลดจำนวนทรัพยากรที่ใช้ แต่ยังไม่มีงานวิจัยใดที่แสดงเห็นเห็นว่ามีการสร้างวงจรที่เปลี่ยนโครงแบบอย่างพลวัตได้จริง ดังจะเห็นได้ว่างานวิจัยของ Xu Jian, Liu Yuan-feng, Dai Zi Bin และ Sun Yi [8] ได้ใช้วิธีการเปลี่ยนข้อมูลในบล็อกแรมเพื่อทำให้วงจรเอสบ็อกซ์สามารถใช้ได้ทั้งการเข้ารหัสและถอดรหัส ซึ่งไม่ได้เป็นการทำงานต่อเนื่องกัน ต่อมา Oscar Perez, Yves Berviller, Camel Tanougast และ Serge Weber [10] ได้เปรียบเทียบวงจรเข้ารหัสที่สามารถเปลี่ยนโครงแบบได้กับวงจรอื่น แต่ไม่ได้กล่าวถึงวิธีการเปลี่ยนโครงแบบ แต่ได้คำนวณเวลาที่ใช้ในการเปลี่ยนโครงแบบไว้ ทำให้ไม่อาจสรุปได้ว่าวงจรในงานวิจัยนี้มีการเปลี่ยนโครงแบบได้จริง จากนั้นเจนโชติ ศรีพรประเสริฐ และประภาส จงสถิตย์วัฒนา [12], [13] ได้เสนอวงจรเข้ารหัสเออีเอสชนิดเปลี่ยนโครงแบบได้ แต่ใช้มัลติเพลกเซอร์ช่วยในการเปลี่ยนโครงแบบแทนการเปลี่ยนโครงแบบของเอฟพีจีเอ ซึ่งจะเห็นได้ว่าการเปลี่ยนโครงแบบอย่างพลวัตของวงจรเข้ารหัสนั้นเป็นหัวข้อที่น่าสนใจ และมีแนวโน้มที่จะช่วยประหยัดทรัพยากรได้

บทที่ 3
การออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต

3.1. แนวคิดในการออกแบบ

การออกแบบวงจรเข้ารหัสเออีเอสให้เป็นวงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัตนั้น จำเป็นต้องมีความเข้าใจการเข้ารหัสเออีเอสอย่างละเอียด จากนั้นจึงนำเอาระเบียบวิธีหรือขั้นตอนการเข้ารหัสมาวิเคราะห์อย่างละเอียด ว่าส่วนใดควรอยู่ด้วยกัน ส่วนใดควรแยกออกจากกัน โดยส่วนที่สามารถแยกออกจากกันได้ ควรจะเป็นส่วนที่ไม่ได้ทำงานเกี่ยวข้องกัน เพื่อให้สะดวกต่อการออกแบบและพัฒนาประสิทธิภาพ

3.1.1. การออกแบบวงจรเข้ารหัสเออีเอส

เนื่องจากการออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบอย่างพลวัตในงานวิจัยชิ้นนี้มีจุดประสงค์เพื่อออกแบบวงจรเข้ารหัสเออีเอสที่มีขนาดเล็ก วงจรเข้ารหัสเออีเอสที่เหมาะสมสำหรับการนำมาเปลี่ยนโครงแบบจึงควรมีขนาดเล็ก แต่มีโครงสร้างไม่ซับซ้อน เพื่อให้สามารถแบ่งส่วนวงจรได้ง่าย

เพื่อสร้างวงจรเข้ารหัสเออีเอสที่มีขนาดเล็ก แต่มีโครงสร้างไม่ซับซ้อน ความเข้าใจในระเบียบวิธีการเข้ารหัสเออีเอสจึงเป็นสิ่งที่สำคัญ ซึ่งการเข้ารหัสเออีเอสแบบ 128 บิตมีการทำงานทั้งสิ้น 10 รอบ โดยในแต่ละรอบสามารถแบ่งออกเป็น 2 ขั้นตอนใหญ่ๆ คือ

1. กระบวนการขยายคีย์ (KeyExpansion)
2. กระบวนการรหัส (Cipher)

โดยทั้งสองขั้นตอนมีความสัมพันธ์กันดังรูปที่ 3.1

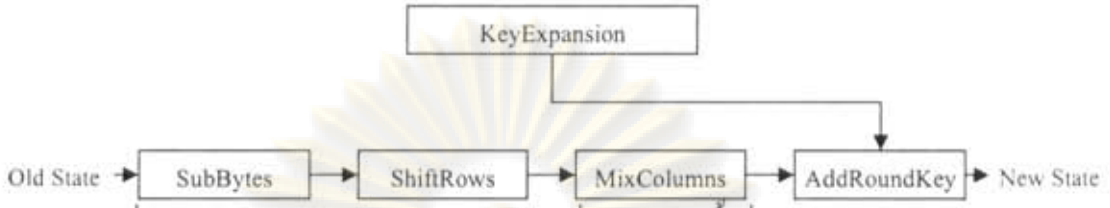
รูปที่ 3.1 ความสัมพันธ์ระหว่างกระบวนการขยายคีย์และกระบวนการรหัส

โดยกระบวนการรหัสยังสามารถแบ่งย่อยออกไปได้เป็นอีก 4 ขั้นตอน ได้แก่

1. กระบวนการแทนที่ไบต์ (SubBytes) – เป็นกระบวนการเปลี่ยนแปลงสถานะของข้อความโดยการแทนที่ด้วยค่าในเอสบ็อกซ์
2. กระบวนการเลื่อนแถว (ShiftRows) – เป็นกระบวนการเปลี่ยนแปลงสถานะของข้อความโดยการเลื่อนแถว
3. กระบวนการผสมหลัก (MixColumns) - เป็นกระบวนการเปลี่ยนสถานะของข้อความโดยการผสมแต่ละหลัก

4. กระบวนการบวกคีย์แต่ละรอบ (AddRoundKey) – เป็นกระบวนการเปลี่ยนแปลงสถานะของข้อความโดยการทำเอ็กซ์ออร์กับคีย์ในรอบนั้นๆ

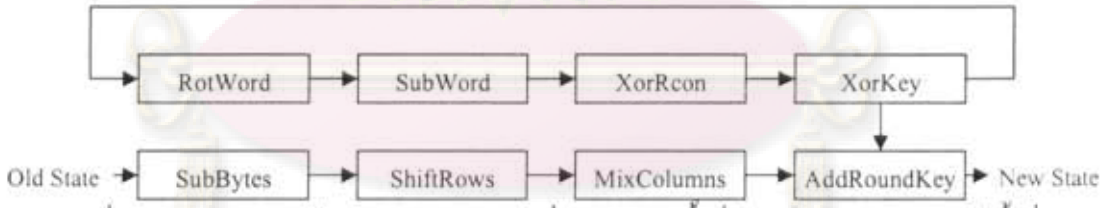
เมื่อนำกระบวนการทั้งสี่มาผนวกเข้ากับกระบวนการหาคีย์จะได้ความสัมพันธ์ดังรูปที่ 3.2



รูปที่ 3.2 ความสัมพันธ์ของกระบวนการเปลี่ยนสถานะทั้งสี่และกระบวนการหาคีย์

นอกจากนี้การบวนการหาคีย์สำหรับเข้ารหัสเพื่อเปลี่ยนสถานะของข้อความยังสามารถแบ่งออกเป็น 4 กระบวนการย่อยๆ ได้แก่

1. กระบวนการเลื่อนคำ (RotWord) – เป็นกระบวนการเปลี่ยนแปลงคีย์ชั่วคราวโดยนำคีย์ในหลักสุดท้ายมาทำการเลื่อนหลัก
2. กระบวนการแทนที่คำ (SubWord) – เป็นกระบวนการเปลี่ยนแปลงคีย์ชั่วคราวโดยการแทนที่ด้วยค่าในเอสบ็อกซ์ ขั้นตอนนี้สามารถสลับลำดับกับกระบวนการเลื่อนคำได้
3. กระบวนการบวกค่าคงที่ของแต่ละรอบ (XorRcon) – เป็นกระบวนการเปลี่ยนแปลงคีย์ชั่วคราวที่ผ่านกระบวนการทั้งสองข้างต้นโดยการทำเอ็กซ์ออร์กับค่าคงที่ของแต่ละรอบ (Round Constant: Rcon)
4. กระบวนการบวกคีย์ (XorKey) – เป็นกระบวนการเปลี่ยนแปลงคีย์โดยการทำเอ็กซ์ออร์กับคีย์ชั่วคราวที่ผ่านกระบวนการทั้งสามข้างต้นมาแล้ว

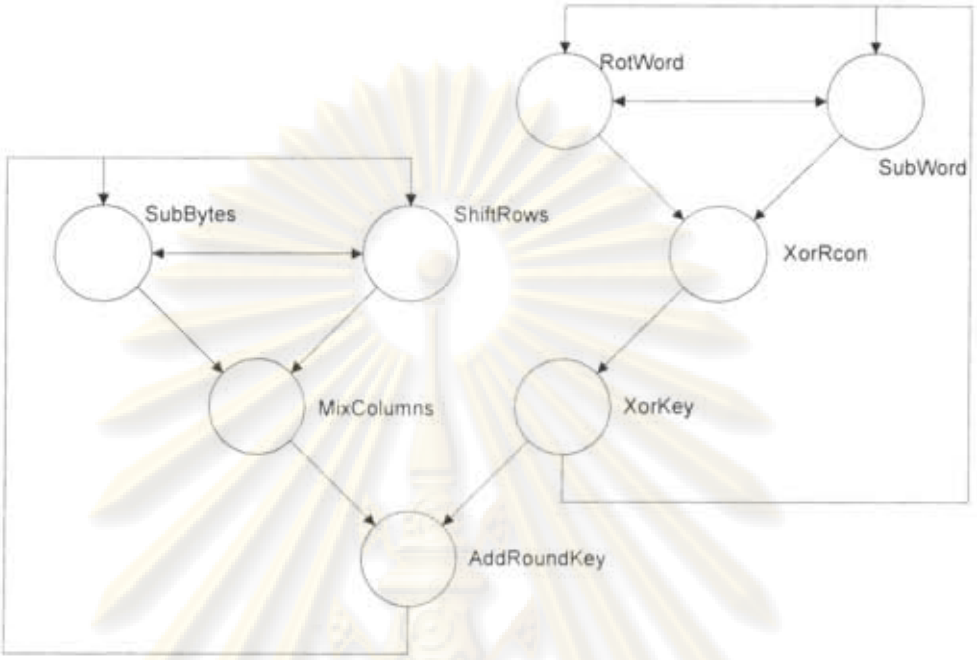


รูปที่ 3.3 ความสัมพันธ์ของกระบวนการเปลี่ยนสถานะทั้งสี่และกระบวนการหาคีย์ทั้งสี่

นอกจากการทำงานที่สามารถแบ่งออกได้เป็นสองขั้นตอนใหญ่ๆข้างต้นแล้ว กระบวนการเข้ารหัสเออีเอสสามารถวิเคราะห์ด้วยกราฟแสดงความขึ้นต่อกันของกระบวนการได้ การวิเคราะห์วงจรด้วยกราฟแบบนี้จะพิจารณาจากความเป็นอิสระต่อกันในการทำงาน ซึ่งจะช่วยให้เห็นว่าขั้นตอนใดมีความสำพันธ์ต่อกันได้ง่ายยิ่งขึ้น

กราฟที่ใช้อธิบายประกอบด้วยจุดยอดหรือกระบวนการทำงาน และเส้นเชื่อมแบบมีทิศทางแสดงความขึ้นต่อกันของกระบวนการทำงาน โดยจุดยอดที่มีเส้นเชื่อมถึงกันจะไม่เป็นอิสระต่อกันหรือไม่สามารถทำงานพร้อมๆกันได้ โดยกระบวนการที่อยู่ตรงปลายลูกศรของเส้นเชื่อมจะต้องรอ

ให้กระบวนการก่อนหน้าหรือกระบวนการที่อยู่ตรงหางลูกศรของเส้นเชื่อมเสร็จก่อนแล้วจึงจะสามารถทำงานได้ ซึ่งกราฟแสดงความขึ้นต่อกันของกระบวนการเข้ารหัสเออีเอสเป็นดังรูปที่ 3.4



รูปที่ 3.4 กราฟแสดงความขึ้นต่อกันของกระบวนการเข้ารหัสเออีเอส

จากรูปที่ 3.4 จะเห็นได้ว่ากระบวนการแทนที่ไบต์และกระบวนการเลื่อนแถวสามารถสลับลำดับการทำงานได้ แต่มีความขึ้นต่อกัน เช่นเดียวกับกระบวนการเลื่อนคำและกระบวนการแทนที่คำ กระบวนการผสมหลักจะต้องทำงานหลังจากกระบวนการแทนที่ไบต์และกระบวนการเลื่อนแถว กระบวนการบวกคีย์จะต้องทำหลังกระบวนการบวกค่าคงที่ของแต่ละรอบซึ่งจะทำงานหลังจากกระบวนการเลื่อนคำและกระบวนการแทนที่คำเท่านั้น สำหรับกระบวนการบวกคีย์แต่ละรอบนั้นจะเป็นกระบวนการสุดท้ายของการทำงานในแต่ละรอบของการเข้ารหัสเออีเอส

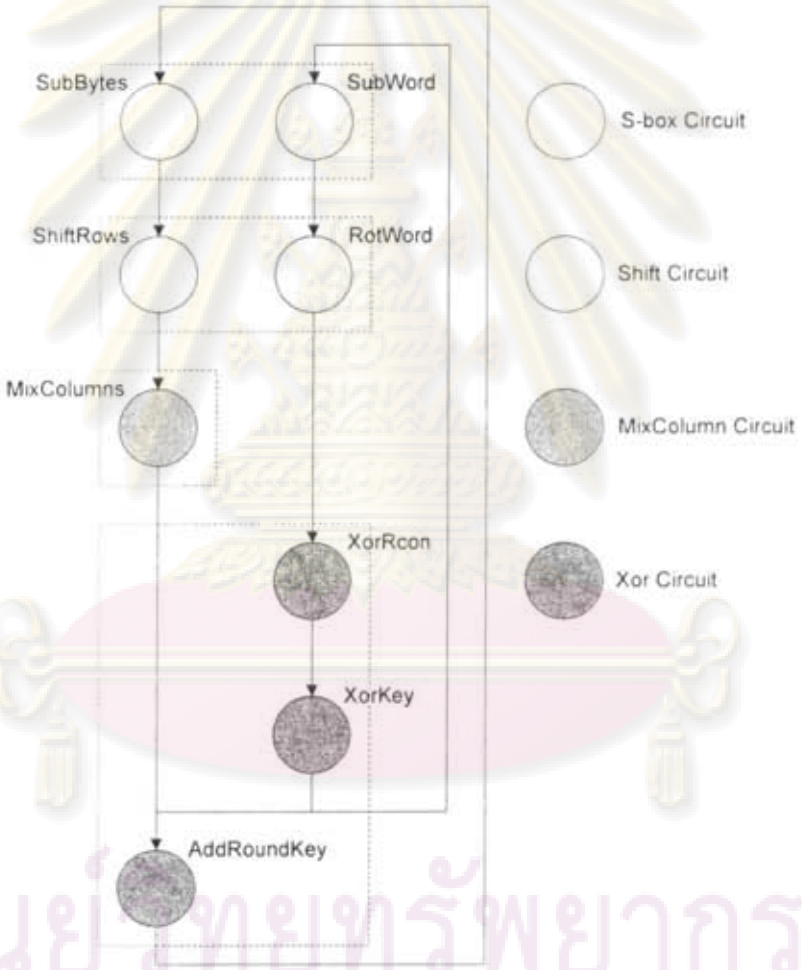
เมื่อวิเคราะห์กระบวนการทั้งแปดของการเข้ารหัสเออีเอส จะพบว่าในการเข้ารหัสเออีเอสนั้นจะมีวงจรย่อยที่สำคัญในการเข้ารหัสทั้งสิ้น 4 วงจร ได้แก่

1. วงจรเอสบ็อกซ์ (S-box Circuit) – ใช้ในกระบวนการแทนที่ไบต์และกระบวนการแทนที่คำ
2. วงจรเลื่อน (Shift Circuit) – ใช้ในกระบวนการเลื่อนแถวและกระบวนการเลื่อนคำ
3. วงจรผสมหลัก (MixColumn Circuit) – ใช้ในกระบวนการผสมหลัก
4. วงจรเอ็กซ์ออร์ (Xor Circuit) – ใช้ในกระบวนการบวกคีย์แต่ละรอบ, กระบวนการบวกค่าคงที่ของแต่ละรอบ และกระบวนการบวกคีย์

เนื่องจากกระบวนการขยายคีย์สามารถทำได้ 2 วิธี คือ ทำไปพร้อมๆกับกระบวนการรหัส หรือทำกระบวนการขยายคีย์ให้เสร็จสิ้นทุกรอบก่อนและจำคีย์ของแต่ละรอบไว้ แล้วจึงค่อยเริ่มกระบวนการรหัส เมื่อพิจารณาถึงจุดประสงค์ของการออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครง

แบบได้นี้จะพบว่า การทำกระบวนการขยายคีย์ไปพร้อมๆกับกระบวนการรหัสเป็นทางเลือกที่เหมาะสมกว่า เนื่องจากจะช่วยประหยัดวงจรหน่วยความจำที่ใช้จำคีย์ได้

เมื่อจัดลำดับขั้นตอนการทำงานทั้งแปดตามลักษณะของวงจรย่อยที่ต้องใช้ทั้งสี่ประเภทแล้ว จะได้ลำดับของวงจรที่จะต้องใช้ในการเข้ารหัสเออีเอสเป็นดังแสดงด้วยกราฟแสดงลำดับการทำงานของวงจรเข้ารหัสเออีเอสดังรูปที่ 3.5 โดยจุดยอดแสดงกระบวนการทำงาน เส้นเชื่อมแบบมีทิศทางแสดงความไม่เป็นอิสระต่อกัน โดยกระบวนการที่อยู่ตรงปลายลูกศรของเส้นเชื่อมจะต้องรอให้กระบวนการก่อนหน้าหรือกระบวนการที่อยู่ตรงหางลูกศรของเส้นเชื่อมเสร็จก่อนแล้วจึงจะสามารถทำงานได้ และกรอบแสดงวงจรที่ใช้ในการทำงาน ส่วนลำดับของการทำงานจะแสดงโดยการเรียงลำดับจากบนลงล่าง โดยกระบวนการที่อยู่ด้านบนจะทำงานก่อนกระบวนการด้านล่าง

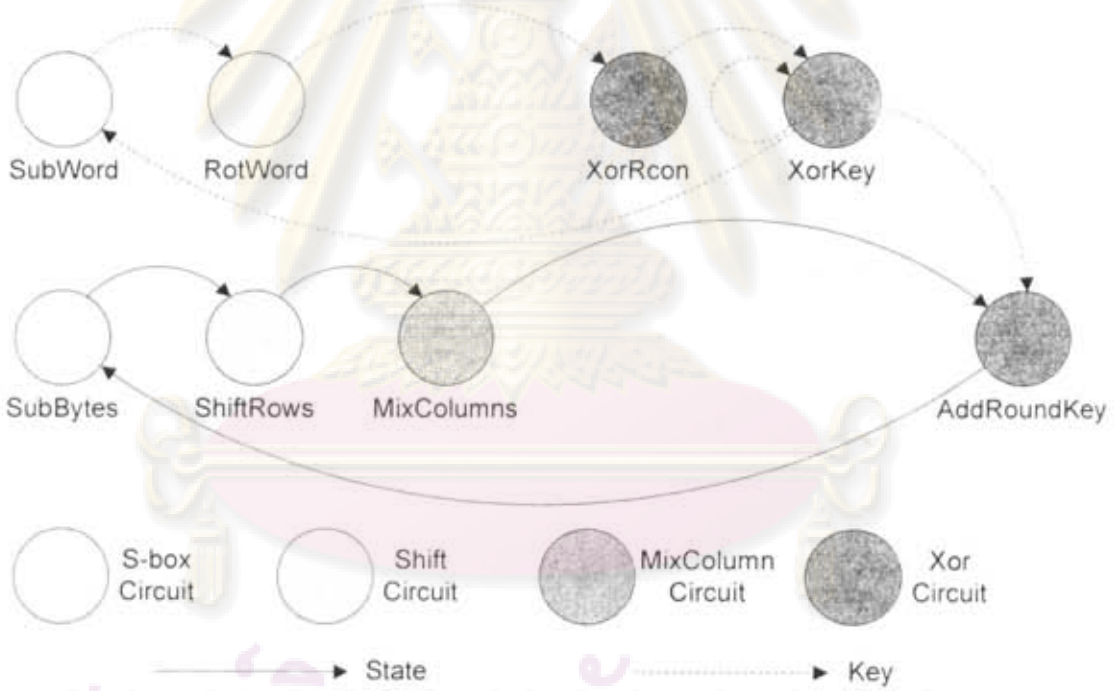


รูปที่ 3.5 กราฟแสดงลำดับการทำงานเมื่อพิจารณาตามลักษณะของวงจรย่อย

จะเห็นได้ว่าในรูปที่ 3.5 กระบวนการเข้ารหัสเออีเอสสามารถจัดลำดับการทำงานตามประเภทของวงจรย่อยทั้งสี่ได้อย่างสมบูรณ์ นั่นหมายความว่าในการทำงานหนึ่งรอบของการเข้ารหัสเออีเอส เราสามารถจะใช้วงจรย่อยต่างๆเพียงวงจรละหนึ่งครั้งเท่านั้น ซึ่งความสามารถนี้จะทำให้การแบ่งวงจรเพื่อเปลี่ยนโครงแบบตามลักษณะของวงจรย่อย เป็นไปได้โดยง่าย

3.1.2. การแยกย่อยวงจร

ในหัวข้อที่แล้ว วงจรเข้ารหัสเออีเอสได้ถูกแบ่งออกเป็น 4 ส่วน ซึ่งวงจรย่อยทั้งสี่นี้จะเป็นวงจรส่วนที่ถูกเปลี่ยนโครงแบบ นั่นหมายความว่า ในขณะใดขณะหนึ่งจะมีวงจรย่อยเพียงวงจรเดียวในสี่วงจรนี้ทำงานอยู่ ซึ่งการแยกย่อยวงจรเออีเอสออกเป็นสี่ส่วนเพื่อการเปลี่ยนโครงแบบนั้น เราจะต้องพิจารณาถึงการส่งข้อมูลระหว่างวงจรย่อย ซึ่งเมื่อนำการส่งข้อมูลระหว่างกระบวนการทั้งสี่มาเขียนเป็นกราฟ จะได้กราฟแสดงการส่งข้อมูลดังรูปที่ 3.6 โดยจุดยอดของกราฟแทนกระบวนการทำงาน และเส้นเชื่อมแบบมีทิศทางแสดงการส่งข้อมูลจากกระบวนการหนึ่งไปสู่อีกกระบวนการหนึ่ง โดยให้จุดยอดที่ปลายเส้นเชื่อมด้านหางลูกศรแทนกระบวนการที่ส่งข้อมูล และจุดยอดที่ปลายเส้นเชื่อมด้านหัวลูกศรแทนกระบวนการที่รับ โดยในที่นี้จะให้เส้นเชื่อมแบบเส้นทึบแทนการส่งข้อมูลสถานะของข้อความ และเส้นเชื่อมแบบเส้นประแทนการส่งข้อมูลคีย์ข้อมูล ลำดับของกระบวนการแสดงด้วยลำดับของจุดยอด โดยจุดยอดทางซ้ายแทนกระบวนการที่เกิดก่อน และจุดยอดทางขวาแสดงกระบวนการที่เกิดขึ้นทีหลัง



รูปที่ 3.6 กราฟแสดงการส่งข้อมูลระหว่างกระบวนการทั้งสี่

จากรูปที่ 3.6 จะเห็นว่ามีการส่งข้อมูลสองประเภท ได้แก่ ข้อมูลสถานะของข้อความ และข้อมูลคีย์ และเมื่อพิจารณาลำดับของการเปลี่ยนวงจรย่อยและการส่งข้อมูล จะพบว่ามีการส่งข้อมูลข้ามวงจรย่อย ดังจะเห็นได้จากการส่งข้อมูลคีย์จากกระบวนการเลื่อนคำซึ่งเป็นวงจรเลื่อนไปยังกระบวนการบวกค่าคงที่ของแต่ละรอบซึ่งเป็นวงจรเอ็กซ์ออร์นั้น เป็นการส่งข้อมูลข้ามวงจรผสมหลักไป เพื่อให้ความสะดวกในการออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต และเพื่อให้วงจรส่วนที่สามารถเปลี่ยนโครงแบบได้มีขนาดเล็ก การส่งข้อมูลระหว่างวงจรย่อยทั้งสี่

จึงควรส่งข้อมูลผ่านวงจรเรจิสเตอร์ที่ทำหน้าที่เก็บข้อมูลสถานะของข้อความและข้อมูลคีย์ ซึ่งรวมไปถึงข้อมูลคีย์ชั่วคราว (Temporary Key: Temp) และค่าคงที่ของแต่ละรอบ

เมื่อพิจารณาขนาดข้อมูลขาเข้าและข้อมูลขาออกของวงจรย่อยทั้งสี่แล้ว จะพบว่าวงจรทั้งสี่มีขนาดข้อมูลขาเข้าและขาออกไม่เท่ากันดังแสดงในตารางที่ 3.1

ตารางที่ 3.1 ตารางแสดงจำนวนข้อมูลขาเข้าและขาออกของวงจรย่อยทั้งสี่

Circuit	Input 1 (bit)	Input 2 (bit)	Output (bit)
S-box	8	-	8
Shift	32	-	32
MixColumn	32	-	32
Xor	1	1	1

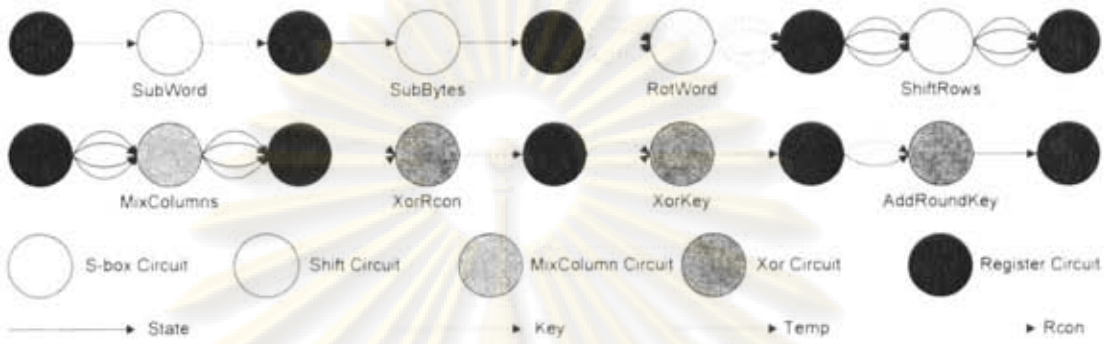
จากตารางที่ 3.1 จะเห็นว่าวงจรเอสบ็อกซ์มีข้อมูลขาเข้า 8 บิตและมีข้อมูลขาออก 8 บิต วงจรผสมหลักใช้ข้อมูลขาเข้าขนาด 32 บิตและให้ข้อมูลขาออกขนาด 32 บิต ซึ่งถือว่ามีขนาดใหญ่ที่สุดในกระบวนการทั้งสี่ และวงจรเอ็กซ์ออร์มีข้อมูลขาเข้า 2 บิต เนื่องจากมีต้องใช้ข้อมูลขาเข้า 2 ตัวในการเอ็กซ์ออร์ และให้ข้อมูลขาออก 1 บิต

สำหรับวงจรเลื่อนนั้นถือเป็นวงจรที่ไม่เหมือนกับวงจรย่อยอื่นๆ โดยวงจรอื่นๆจะมีตำแหน่งของข้อมูลขาเข้าตรงกับตำแหน่งของข้อมูลขาออก แต่ตำแหน่งของข้อมูลขาเข้าของวงจรเลื่อนนั้น ไม่ตรงกับตำแหน่งของข้อมูลขาออก แต่เมื่อพิจารณาที่ข้อมูลขนาด 32 บิต จะพบว่าตำแหน่งของข้อมูลนั้นตรงกัน

เมื่อพิจารณาจากข้อมูลขาเข้าและขาออกของวงจรย่อยทั้งสี่แล้ว จะพบว่าการใช้ทางเดินข้อมูล (datapath) ขนาด 32 บิต เป็นทางเลือกที่เหมาะสมที่สุด เพราะจะทำให้สามารถออกแบบวงจรย่อยที่สามารถทำงานเสร็จได้ภายในหนึ่งรอบสัญญาณนาฬิกาทั้งสี่วงจร แต่เนื่องจากงานวิจัยนี้มีจุดมุ่งหมายเพื่อสร้างวงจรเข้ารหัสเออีเอสที่มีขนาดเล็ก การใช้ทางเดินข้อมูลขนาดเล็กจึงน่าจะทำให้ได้วงจรขนาดเล็กกว่าวงจรที่ใช้ทางเดินข้อมูลขนาดใหญ่ และเมื่อพิจารณาถึงการออกแบบวงจรย่อยทั้งสี่นี้ จะพบว่าวงจรเอสบ็อกซ์เป็นวงจรที่มีขนาดใหญ่ที่สุด ดังจะเห็นได้ว่ามีงานวิจัยหลายชิ้นที่พยายามลดขนาดวงจรเอสบ็อกซ์ การใช้ทางเดินข้อมูลขนาด 32 บิตจะทำให้ต้องมีวงจรเอสบ็อกซ์ถึง 4 วงจร ซึ่งเป็นการสิ้นเปลืองทรัพยากร ในงานวิจัยนี้จึงเลือกใช้ทางเดินข้อมูลขนาด 8 บิตซึ่งมีขนาดเล็กกว่า เพราะจะทำให้ได้วงจรย่อยที่มีขนาดเล็กกว่า

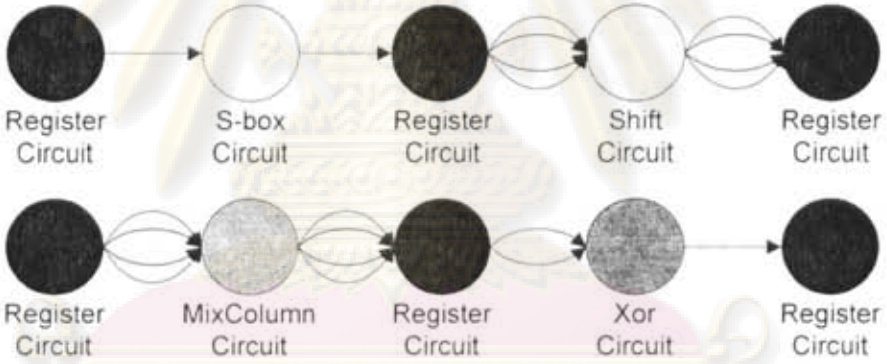
เมื่อพิจารณาเฉพาะการส่งข้อมูลระหว่างกระบวนการทั้งแปดกับวงจรเรจิสเตอร์ และพิจารณาถึงการใช้ทางเดินข้อมูลขนาด 8 บิต จะได้กราฟการส่งข้อมูลดังรูปที่ 3.7 โดยให้เส้นเชื่อมแต่ละเส้นแทนทางเดินข้อมูลขนาด 8 บิต กราฟการส่งข้อมูลในรูปที่ 3.7 นี้ไม่มีการแสดงลำดับการทำงานของกระบวนการต่างๆ เนื่องจากเป็นกราฟการส่งข้อมูลระหว่างวงจรย่อย ซึ่งจะเห็นได้ว่าใน

กระบวนการแทนที่คำและกระบวนการแทนที่ไบต์มีข้อมูลขาเข้าและขาออกขนาด 8 บิต กระบวนการเลื่อนคำ, กระบวนการเลื่อนแถวและกระบวนการผสมหลักมีข้อมูลขาเข้าและขาออกขนาด 32 บิต ส่วนกระบวนการบวกค่าคงที่ของแต่ละรอบ, กระบวนการบวกคีย์และกระบวนการบวกคีย์แต่ละรอบมีข้อมูลขาเข้าขนาด 16 บิตและข้อมูลขาออกขนาด 8 บิต



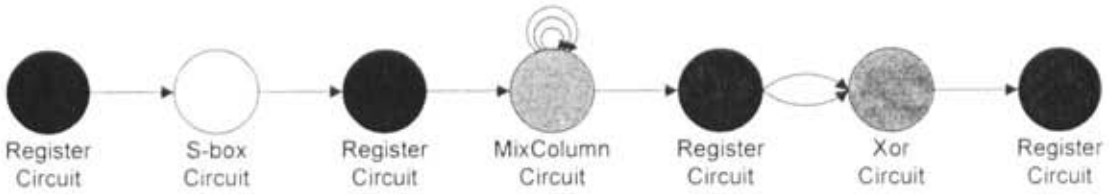
รูปที่ 3.7 กราฟแสดงการส่งข้อมูลระหว่างกระบวนการทั้งแปดและวงจรเรจิสเตอร์

จากรูปที่ 3.7 เพื่อให้สามารถมองภาพของการส่งข้อมูลระหว่างวงจรได้ง่ายขึ้น กราฟแสดงการส่งข้อมูลระหว่างวงจรย่อยทั้งสี่และวงจรเรจิสเตอร์จึงเป็นดังรูปที่ 3.8



รูปที่ 3.8 กราฟแสดงการส่งข้อมูลระหว่างวงจรย่อย

จากรูปที่ 3.8 จะเห็นว่า วงจรเลื่อนและวงจรผสมหลักนั้น มีการใช้ทางเดินข้อมูลทั้งขาเข้าและขาออกมากกว่า 8 บิต ซึ่งเมื่อพิจารณาที่วงจรเลื่อนนั้นจะพบว่า วงจรเลื่อนนั้นสามารถสร้างได้โดยการเลือกการสลับไบต์ของข้อมูลให้ถูกต้องเท่านั้น และเนื่องจากภายในวงจรเรจิสเตอร์นั้นจะต้องมีมัลติเพลกเซอร์ (Multiplexor) และดีโคดเดอร์ (Decoder) อยู่เพื่อเลือกข้อมูลขาเข้าและข้อมูลขาออกของวงจรย่อยทั้งสี่อยู่แล้ว ซึ่งถ้าออกแบบวงจรเรจิสเตอร์ให้เหมาะสม วงจรเรจิสเตอร์จะสามารถใช้แทนวงจรเลื่อนได้ ซึ่งรายละเอียดของวงจรเรจิสเตอร์จะแสดงในหัวข้อที่ 3.2.5 นอกจากนี้เรายังสามารถออกแบบวงจรผสมหลักให้มีข้อมูลขาเข้าและข้อมูลขาออกเพียงขาละ 8 บิต ได้โดยอาศัยฟลิปฟลอปช่วยจำข้อมูลไว้ ซึ่งจะทำให้ข้อมูลขาออกไบต์แรกถูกหน่วงไว้ 4 รอบสัญญาณนาฬิกาหลังจากใส่ข้อมูลขาเข้าไบต์แรก โดยรายละเอียดของการออกแบบวงจรผสมหลักนี้จะแสดงในหัวข้อที่ 3.2.3 ซึ่งกราฟการส่งข้อมูลสุดท้ายจะเป็นดังรูปที่ 3.9



รูปที่ 3.9 กราฟแสดงการส่งข้อมูลระหว่างวงจรย่อยภายหลังการปรับปรุง

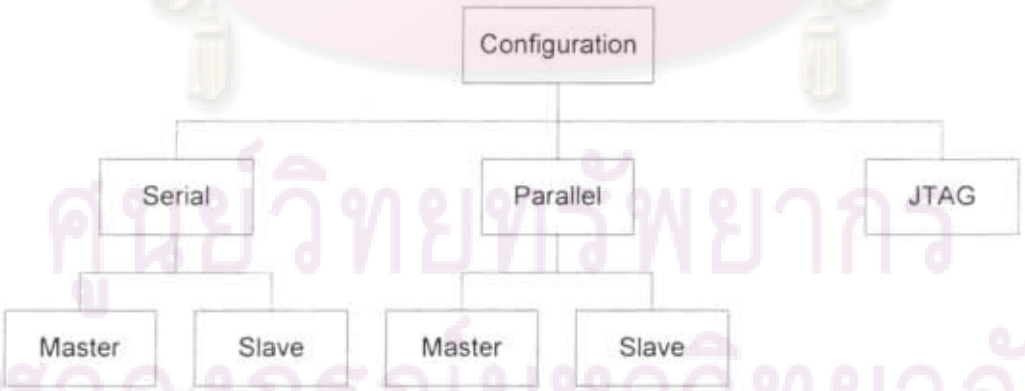
จากรูปที่ 3.9 วงจรเข้ารหัสเออีเอสนี้จะประกอบด้วยวงจรย่อยที่ใช้ในการเปลี่ยนแปลงข้อมูลทั้งสิ้น 4 วงจร ซึ่งแบ่งออกเป็นสองประเภท ได้แก่

1. วงจรที่ไม่สามารถเปลี่ยนโครงแบบได้ ได้แก่ วงจรเรจิสเตอร์ (Register Circuit)
2. วงจรที่สามารถเปลี่ยนโครงแบบได้ ได้แก่ วงจรเอสบ็อกซ์ (S-box Circuit), วงจรผสมหลัก (MixColumn Circuit) และวงจรเอ็กซ์ออร์ (Xor Circuit) ซึ่งวงจรทั้งสามวงจรนี้จะถูกเปลี่ยนโครงแบบสลับกันไปเรื่อยๆ

3.1.3. การเปลี่ยนโครงแบบ

จากที่ได้กล่าวไว้ข้างต้น วิธีการเปลี่ยนโครงแบบของเอฟพีจีเอรุ่น Spartan-3 มี 5 วิธี ซึ่งในวิธีทั้งห้านี้ จะสามารถแบ่งออกได้เป็น 3 วิธีใหญ่ๆ ได้แก่

1. วิธีการเปลี่ยนโครงแบบแบบอนุกรม (Serial Mode) - เป็นวิธีการเปลี่ยนโครงแบบโดยการป้อนข้อมูลโครงแบบทีละบิต
2. วิธีการเปลี่ยนโครงแบบแบบขนาน (Parrallel or SelectMAP Mode) - เป็นวิธีการเปลี่ยนโครงแบบโดยการป้อนข้อมูลโครงแบบทีละไบต์
3. วิธีการเปลี่ยนโครงแบบแบบเจแท็ก (Boundary Scan or JTAG Mode) - เป็นวิธีการเปลี่ยนโครงแบบด้วยวิธีตามมาตรฐาน IEEE 1149.1-1993 และ IEEE1532 สำหรับอุปกรณ์ที่สามารถเปลี่ยนโครงแบบได้ภายในระบบ (In-System Configurable, ISC)



รูปที่ 3.10 แผนภาพแสดงวิธีการเปลี่ยนโครงแบบของเอฟพีจีเอรุ่น Spartan-3

วิธีการเปลี่ยนโครงแบบแบบอนุกรมและขนานจะแบ่งออกเป็น 2 แบบย่อยๆอีก คือ

1. วิธีการเปลี่ยนโครงแบบแบบแม่ข่าย (Master Mode) – เป็นวิธีการเปลี่ยนโครงแบบโดยให้เอฟพีจีเอเป็นตัวส่งสัญญาณนาฬิกาเพื่อควบคุมการเปลี่ยนโครงแบบ
2. วิธีการเปลี่ยนโครงแบบแบบลูกข่าย (Slave Mode) – เป็นวิธีการเปลี่ยนโครงแบบโดยให้เอฟพีจีเอเป็นตัวรับสัญญาณนาฬิกาที่ใช้ควบคุมการเปลี่ยนโครงแบบ

โดยแผนภาพของการเปลี่ยนโครงแบบจะเป็นดังรูปที่ 3.10

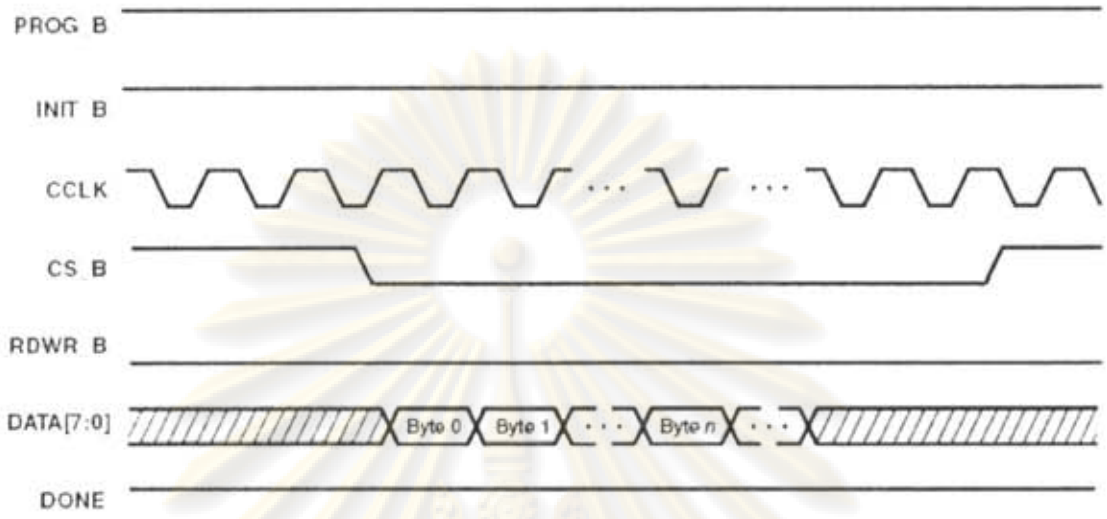
การเปลี่ยนโครงแบบแต่ละวิธีนั้นมีข้อดีและข้อเสียแตกต่างกันออกไป แต่เนื่องจากประยุกต์แนวคิดของการเปลี่ยนโครงแบบในงานวิจัยชิ้นนี้มีจุดประสงค์เพื่อลดขนาดของวงจรเพื่อให้สามารถนำไปใช้ในระบบที่มีทรัพยากรจำกัดได้ วิธีการเปลี่ยนโครงแบบแบบ Boundary-Scan จึงเป็นทางเลือกที่ไม่เหมาะสม เพราะการเปลี่ยนโครงแบบแบบ Boundary-Scan จะต้องทำงานตามมาตรฐาน IEEE 1149.1-1993 และ IEEE1532 สำหรับอุปกรณ์ที่สามารถเปลี่ยนโครงแบบได้ภายในระบบ ซึ่งจะทำให้การเปลี่ยนโครงแบบโดยมาตรฐานนี้มีความปลอดภัยมากที่สุด แต่วงจรควบคุมการเปลี่ยนโครงแบบแบบ Boundary-Scan นั้นมีความซับซ้อนและมีขนาดใหญ่ซึ่งไม่สอดคล้องกับจุดมุ่งหมายของวงจร นอกจากนี้การเปลี่ยนโครงแบบแบบ Boundary-Scan ยังทำงานได้ช้าเมื่อเปรียบเทียบกับวิธีการเปลี่ยนโครงแบบที่เหลือ และเนื่องจากวิธีการเปลี่ยนโครงแบบแบบอนุกรมไม่สามารถรองรับการเปลี่ยนโครงแบบบางส่วนของเอฟพีจีเอได้ งานวิจัยชิ้นนี้จึงเลือกใช้วิธีการเปลี่ยนโครงแบบแบบขนานในการสร้างวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต

เมื่อพิจารณาถึงลักษณะการทำงานของวงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัตแล้วจะพบว่าวงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัตต้องควบคุมการเปลี่ยนโครงแบบได้อย่างสมบูรณ์ งานวิจัยชิ้นนี้จึงเลือกใช้วิธีการเปลี่ยนโครงแบบแบบ Slave Parallel เนื่องจากต้องการให้วงจรควบคุมการเข้ารหัสบนเอฟพีจีเอควบคุมการเปลี่ยนโครงแบบของเอฟพีจีเอได้

เมื่อพิจารณากราฟแสดงขั้นตอนการเปลี่ยนโครงแบบในรูปที่ 2.6 และแผนภาพสายงานแสดงขั้นตอนการเปลี่ยนโครงแบบในรูปที่ 2.9 จะพบว่าการเลือกวิธีการเปลี่ยนโครงแบบ จะกระทำหลังการล้างหน่วยความจำโครงแบบ นั่นหมายความว่าการเปลี่ยนวิธีเปลี่ยนโครงแบบ ไม่สามารถกระทำได้ในการเปลี่ยนโครงแบบบางส่วน หรือการเปลี่ยนวิธีเปลี่ยนโครงแบบจะกระทำได้เมื่อมีการเปลี่ยนโครงแบบทั้งหมดเท่านั้น

นอกจากนี้เมื่อพิจารณากราฟในรูปที่ 2.6 โดยละเอียด จะพบว่าในการเปลี่ยนโครงแบบบางส่วนนั้นจะมีสัญญาณควบคุมที่สำคัญอยู่ 2 สัญญาณ ได้แก่ สัญญาณ CCLK และสัญญาณ CS_B โดยสัญญาณ CCLK สามารถปล่อยให้สัญญาณนาฬิกาวิ่งตลอดเวลาได้ ส่วนสัญญาณ CS_B จะต้องคงให้เป็นสถานะสูงตลอดเวลา และจะเปลี่ยนเป็นสถานะต่ำเมื่อจะเปลี่ยนโครงแบบเท่านั้น ส่วนสัญญาณควบคุมการเปลี่ยนโครงแบบอื่นๆให้คงไว้เหมือนเดิม นั่นคือ สัญญาณ PROG_B เป็นสถานะสูง สัญญาณ INIT_B เป็นสถานะสูง แต่สัญญาณ RDWR_B เป็นสถานะต่ำ ส่วนสัญญาณที่

เหลือให้เป็นสัญญาณขาออกจากเอฟพีจีเอ โดยกราฟแสดงสัญญาณควบคุมการเปลี่ยนโครงแบบบางส่วนนี้จะเป็นดังรูปที่ 3.11



รูปที่ 3.11 กราฟแสดงสัญญาณควบคุมการเปลี่ยนโครงแบบบางส่วน

ในการเปลี่ยนโครงแบบข้อมูลโครงแบบหรือโปรแกรมโครงแบบนั้นเป็นส่วนสำคัญที่จะใช้กำหนดคุณสมบัติของวงจร ซึ่งในการออกแบบวงจรให้สามารถเปลี่ยนโครงแบบของตัวเองได้นั้น จะต้องมีหน่วยความจำโครงแบบที่ใช้สำหรับเก็บข้อมูลโครงแบบที่จะเปลี่ยน ซึ่งข้อมูลโครงแบบนี้โดยปกติแล้วจะมีขนาดใหญ่ จึงสมควรที่จะใช้หน่วยความจำภายนอกในการเก็บข้อมูลส่วนนี้ เนื่องจากหน่วยความจำภายในเอฟพีจีเอ (BlockRAM) นั้นมีขนาดเล็กเกินไปดังแสดงให้เห็นในตารางที่ 3.2 และการใช้หน่วยความจำภายในเอฟพีเอจะทำให้การออกแบบยากขึ้นเนื่องจากจะต้องวางสายสัญญาณข้อมูลโครงแบบไม่ให้ผ่านส่วนที่สามารถเปลี่ยนโครงแบบได้ดังที่ได้กล่าวไว้ในบทที่ 2 หัวข้อที่ 2.3

ตารางที่ 3.2 ตารางแสดงจำนวนบิตที่ต้องใช้ในการเก็บข้อมูลโครงแบบ

รุ่น	Spartan-3 (XC3S200-4FT256)
จำนวน BlockRAM	12
จำนวนบิตต่อ BlockRAM *	16,384
จำนวนบิตทั้งหมดที่สามารถใช้ได้ *	196,608
จำนวนบิตที่ใช้เก็บข้อมูลโครงแบบ 4 หลัก	69,024
จำนวนบิตทั้งหมดที่ต้องใช้ในการเก็บข้อมูลโครงแบบของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต	207,072

* จำนวนบิตที่แสดงในตารางไม่นับรวมบิตภาวะคู่หรือคี่ (parity bit)

3.1.4. หน่วยควบคุม

ในระบบทั่วไปหน่วยควบคุม (Control Unit) มีหน้าที่ควบคุมการทำงานของวงจร โดยการจัดการการไหลของข้อมูล ซึ่งหน่วยควบคุมจะควบคุมการทำงานภายในระบบผ่านสัญญาณควบคุม (Control Signal) เมื่อระบบได้รับสัญญาณควบคุม ระบบก็จะทำงานตามที่กำหนดไว้ และจะทำงานนั้นไปจนกว่าจะมีการเปลี่ยนแปลงสัญญาณควบคุมเป็นอย่างอื่น โดยในหน่วยควบคุมทั่วไปที่เป็นเครื่องสถานะจำกัด (Finite State Machine) สัญญาณควบคุมจะเปลี่ยนแปลงตามสถานะ (State) ของหน่วยควบคุม ซึ่งในการควบคุมสมวาร (Synchronous Control) การเปลี่ยนสถานะของหน่วยควบคุมจะเปลี่ยนแปลงตามขอบสัญญาณนาฬิกา

จากแนวคิดในการออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตข้างต้น ระบบมีการทำงานหลักๆ 2 อย่าง ได้แก่

1. การเข้ารหัส – เป็นการเข้ารหัสเออีเอสแบบ 128 บิต โดยทำงานตามลำดับที่ได้กำหนดไว้ตามมาตรฐานการเข้ารหัสเออีเอสที่ได้กล่าวไว้ในบทที่ 2 หัวข้อที่ 2.4
2. การเปลี่ยนโครงแบบ - เป็นการเปลี่ยนโครงแบบของวงจรย่อยที่สามารถเปลี่ยนโครงแบบได้ทั้งสามวงจรที่ได้ออกแบบไว้ โดยการเปลี่ยนโครงแบบจะเกิดขึ้นเมื่อวงจรย่อยที่ต้องการใช้งานไม่อยู่ในระบบ หรือยังไม่ได้ถูกโปรแกรมลงไปยังเอฟพีจีเอ

จะเห็นได้ว่าการทำงานทั้งสองอย่างมีความเกี่ยวข้องกัน คือ การเข้ารหัสจะเป็นตัวเลือกวงจรย่อยที่จะใช้ในการเปลี่ยนโครงแบบ แต่ไม่ได้ยุ่งเกี่ยวกับขั้นตอนการเปลี่ยนโครงแบบ และการเปลี่ยนโครงแบบก็ไม่ได้ยุ่งเกี่ยวกับการเข้ารหัส เพื่อความง่ายในการออกแบบหน่วยควบคุม และลดความซับซ้อนของหน่วยควบคุม หน่วยควบคุมของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบอย่างพลวัตในงานวิจัยนี้จึงถูกแบ่งออกเป็น 2 ส่วนตามลักษณะการทำงาน ดังนี้

1. หน่วยควบคุมการเข้ารหัส (Encryption Control Unit) – ทำหน้าที่ควบคุมการเข้ารหัสเออีเอส และเลือกวงจรย่อยที่จะใช้ในการเข้ารหัส โดยการสั่งการเปลี่ยนโครงแบบไปยังหน่วยควบคุมการเปลี่ยนโครงแบบ
2. หน่วยควบคุมการเปลี่ยนโครงแบบ (Reconfiguration Control Unit) - ทำหน้าที่ควบคุมการเปลี่ยนโครงแบบโดยส่งสัญญาณที่ใช้ในการเปลี่ยนโครงแบบตามรูปที่ 3.11 โดยการเปลี่ยนโครงแบบจะเกิดขึ้นเมื่อหน่วยควบคุมการเข้ารหัสเลือกวงจรย่อยที่ไม่ตรงกับวงจรย่อยที่ถูกโปรแกรมอยู่ในระบบในขณะนั้น เมื่อเกิดการเปลี่ยนโครงแบบ หน่วยควบคุมการเปลี่ยนโครงแบบจะเป็นผู้เลือกข้อมูลโครงแบบในหน่วยความจำตามที่หน่วยควบคุมการเข้ารหัสต้องการ และจะส่งสัญญาณควบคุมการเปลี่ยนโครงแบบและข้อมูลโครงแบบที่เลือก ผ่านสายสัญญาณภายนอกเอฟพีจีเอและย้อนกลับเข้าไปยังเอฟพีจีเอเพื่อควบคุมการเปลี่ยนโครงแบบ เนื่องจากเอฟพีจีเอรุ่น Spartan-3 ไม่มีหน่วย ICAP ที่ใช้ในการเปลี่ยน

โครงแบบภายในตัวเอฟพีจีเอ ซึ่งการเชื่อมต่อระหว่างหน่วยควบคุมการเปลี่ยนโครงแบบ หน่วยความจำโครงแบบ และเอฟพีจีเอ เป็นดังรูปที่ 3.12

เมื่อพิจารณาการทำงานของหน่วยควบคุมทั้งสอง จะพบว่าหน่วยควบคุมการเข้ารหัสและหน่วยควบคุมการเปลี่ยนโครงแบบนั้น จะต้องมีการติดต่อและส่งสัญญาณเพื่อควบคุมการทำงานของกันและกัน โดยหน่วยควบคุมการเข้ารหัสจะต้องส่งสัญญาณเลือกวงจรย่อยไปให้หน่วยควบคุมการเปลี่ยนโครงแบบ และหน่วยควบคุมการเปลี่ยนโครงแบบก็จะส่งสัญญาณบอกหน่วยควบคุมการเข้ารหัสเมื่อการเปลี่ยนโครงแบบเสร็จสิ้น และเนื่องจากหน่วยควบคุมการเปลี่ยนโครงแบบจะทำการเปลี่ยนโครงแบบเมื่อวงจรย่อยที่เลือกไม่ถูกโปรแกรมอยู่ในขณะนั้นเท่านั้น วงจรย่อยที่สามารถเปลี่ยนโครงแบบได้ทั้งสามจึงมีการเพิ่มวงจรส่วนที่ใช้ในการตรวจสอบว่าเป็นวงจรที่ต้องการหรือไม่

3.1.5. แนวคิดโดยสรุป

เมื่อนำแนวคิดทั้งหมดที่ได้กล่าวไว้ข้างต้นมาประยุกต์เข้าด้วยกัน จะได้แนวคิดของการออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต โดยวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตนี้จะมีส่วนประกอบหลักๆของวงจรเมื่อแบ่งตามลักษณะของโครงแบบ 3 ส่วน ได้แก่

1. วงจรส่วนที่สามารถเปลี่ยนโครงแบบได้ (Reconfigurable Module)
2. วงจรส่วนที่ไม่สามารถเปลี่ยนโครงแบบได้ (Static Module)
3. หน่วยความจำภายนอกเอฟพีจีเอ (External Memory)

โดยวงจรส่วนที่สามารถเปลี่ยนโครงแบบได้จะมีทั้งหมด 3 วงจรย่อย ซึ่งวงจรย่อยเหล่านี้จะถูกผลัดเปลี่ยนกันทำงานในระบบด้วยการเปลี่ยนโครงแบบ ดังนี้

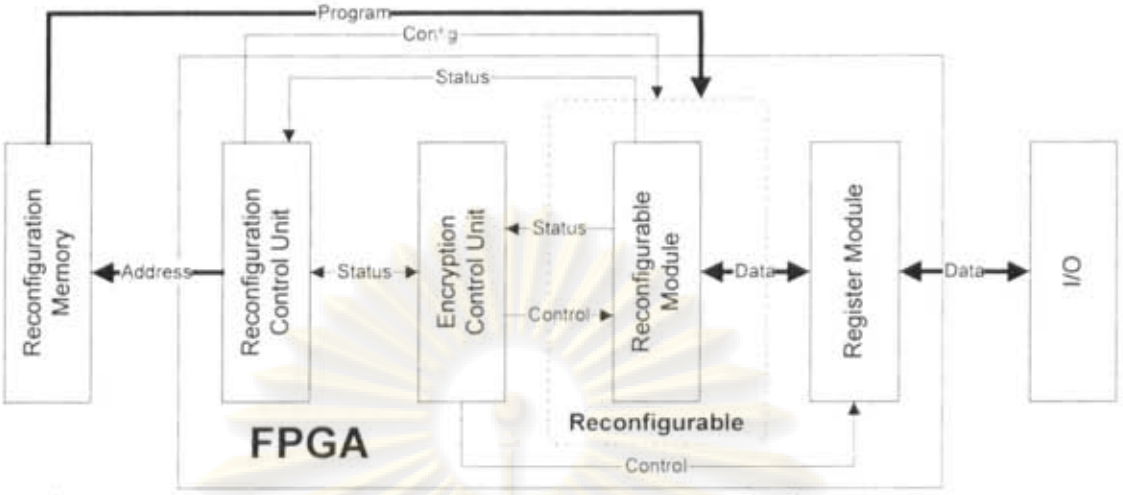
1. วงจรย่อยเอสบ็อกซ์ (S-box Module)
2. วงจรย่อยผสมหลัก (MixColumn Module)
3. วงจรย่อยเอ็กซ์ออร์ (Xor Module)

วงจรส่วนที่ไม่สามารถเปลี่ยนโครงแบบได้แบ่งออกเป็น 3 วงจรย่อย ได้แก่

1. วงจรย่อยเรจิสเตอร์ (Register Module)
2. หน่วยควบคุมการเข้ารหัส (Encryption Control Unit)
3. หน่วยควบคุมการเปลี่ยนโครงแบบ (Reconfiguration Control Unit)

สำหรับหน่วยความจำภายนอกเอฟพีจีเอจะใช้สำหรับเก็บข้อมูลโครงแบบสำหรับใช้ในการเปลี่ยนโครงแบบของเอฟพีจีเอ หรือเรียกว่าหน่วยความจำโครงแบบ (Reconfiguration Memory)

จากรายละเอียดข้างต้นวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตนี้จะประกอบด้วยวงจรย่อยทั้งสิ้น 7 วงจรย่อย และมีการเชื่อมต่อกันดังแสดงในรูปที่ 3.12



รูปที่ 3.12 แผนภาพแสดงสถาปัตยกรรมของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต

3.2. รายละเอียดการออกแบบ

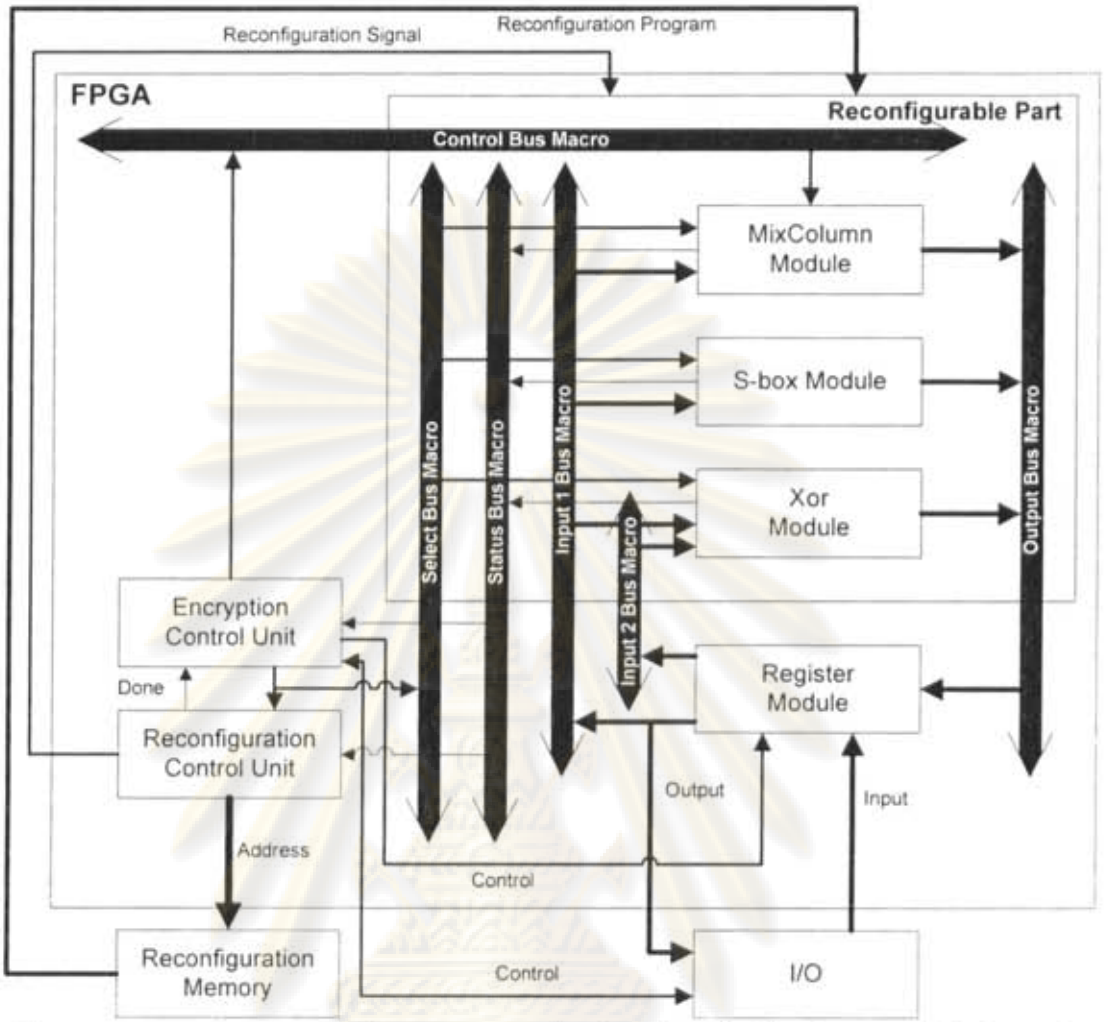
ในหัวข้อนี้จะกล่าวถึงรายละเอียดของการออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตทั้งหมด โดยจะเริ่มจากการออกแบบทางเดินข้อมูล ซึ่งรวมถึงการออกแบบช่องทางการสื่อสารระหว่างวงจรส่วนที่สามารถเปลี่ยนโครงแบบได้กับส่วนที่ไม่สามารถเปลี่ยนโครงแบบได้ และการออกแบบหน่วยควบคุม

3.2.1. วงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต

ในการออกแบบทางเดินข้อมูลของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตในงานวิจัยนี้ จะเห็นได้ว่าส่วนที่ยากที่สุดในการออกแบบทางเดินข้อมูลคือส่วนที่เชื่อมต่อกันระหว่างส่วนที่สามารถเปลี่ยนโครงแบบได้ (Reconfigurable Module) กับส่วนที่ไม่สามารถเปลี่ยนโครงแบบได้ (Static Module) ซึ่งการเชื่อมต่อระหว่างสองส่วนนี้เราจะใช้บัสแมโคร (Bus Macro) เป็นตัวเชื่อมต่อ โดยรายละเอียดของบัสแมโครนี้ได้กล่าวไว้ในบทที่ 2 หัวข้อที่ 2.3

ดังแสดงให้เห็นในกราฟแสดงการส่งข้อมูลระหว่างวงจรในรูปที่ 3.9 จะเห็นว่าวงจรย่อยเอสบ็อกซ์และวงจรย่อยผสมหลักต้องการทางเดินข้อมูลขนาด 8 บิต 2 ช่องทางสำหรับส่งข้อมูลขาเข้าและขาออก ส่วนวงจรย่อยเอ็กซ์ออร์จะใช้ทางเดินข้อมูลขนาด 8 บิต 3 ช่องทาง 2 ช่อทางสำหรับข้อมูลขาเข้า และอีก 1 ช่องทางสำหรับข้อมูลขาออก โดยทางเดินข้อมูลขาเข้าและขาออกของวงจรย่อยทั้งสามสามารถใช้ร่วมกันได้ นั่นหมายความว่าวงจรที่สามารถเปลี่ยนโครงแบบได้ทั้งสามจะใช้ทางเดินข้อมูลขนาด 8 บิต 3 ช่องทาง นั่นคือจะมีบัสแมโครขนาด 8 บิต 3 โดยทางเดินข้อมูลของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้เป็นดังรูปที่ 3.13

จากรูปที่ 3.13 จะเห็นว่าการส่งสัญญาณควบคุมและข้อมูลระหว่างส่วนที่ไม่สามารถเปลี่ยนโครงแบบได้กับส่วนที่สามารถเปลี่ยนโครงแบบได้ สามารถกระทำได้ด้วยการส่งสัญญาณผ่านบัสแมโครทั้งหก ได้แก่



รูปที่ 3.13 แผนภาพแสดงทางเดินข้อมูลของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต

1. บัสแมโครอินพุต 1 (Input 1 Bus Macro) เป็นช่องทางส่งข้อมูลขนาด 8 บิตจากวงจรย่อยเรจิสเตอร์ (Register Module) ไปยังวงจรย่อยที่สามารถเปลี่ยนโครงแบบได้ทั้งสามวงจรเพื่อทำการเปลี่ยนแปลงสถานะและคีย์
2. บัสแมโครอินพุต 2 (Input 2 Bus Macro) เป็นช่องทางส่งข้อมูลขนาด 8 บิตจากวงจรย่อยเรจิสเตอร์ (Register Module) ไปยังวงจรย่อยเอ็กซ์ออร์ (Xor Module) เพื่อใช้เป็นข้อมูลขาเข้าอีกหนึ่งตัวในการเอ็กซ์ออร์
3. บัสแมโครเอาท์พุต (Output Bus Macro) เป็นช่องทางส่งข้อมูลขนาด 8 บิตจากวงจรย่อยที่สามารถเปลี่ยนโครงแบบได้ทั้งสามวงจรไปยังวงจรย่อยเรจิสเตอร์ (Register Module) เพื่อนำไปเก็บไว้ใช้ในขั้นตอนต่อไป
4. บัสแมโครควบคุม (Control Bus Macro) เป็นช่องทางส่งสัญญาณขนาด 2 บิตสำหรับควบคุมการทำงานของวงจรย่อยผสมหลัก เนื่องจากวงจรย่อยนี้เป็นวงจรที่ทำงานทีละ 32 บิต จึงต้องใช้เวลา 4 รอบสัญญาณนาฬิกาในการทำงาน

5. บัสแมโครเลือก (Select Bus Macro) เป็นช่องทางสำหรับส่งสัญญาณตรวจสอบวงจรย่อยว่าเป็นวงจรเดียวกับที่ต้องการหรือไม่ และเนื่องจากวงจรย่อยที่สามารถเปลี่ยนโครงแบบได้มีทั้งหมด 3 วงจรย่อย บัสแมโครนี้จึงมีขนาด 2 บิต
6. บัสแมโครสถานะ (Status Bus Macro) เป็นช่องทางส่งสัญญาณขนาด 1 บิต ใช้ส่งสัญญาณผลลัพธ์ของการตรวจสอบวงจรย่อยว่าตรงกับวงจรที่เลือกด้วยบัสแมโครเลือกหรือไม่

นอกจากนี้สัญญาณเลือกโปรแกรมโครงแบบซึ่งเป็นตำแหน่ง (Address) ของหน่วยความจำโครงแบบ (Reconfiguration Memory) มีขนาด 16 บิต โดยจะแบ่งเป็นส่วนเลือกโครงแบบ 2 บิต และส่วนตำแหน่ง 14 บิตซึ่งเพียงพอสำหรับการเก็บข้อมูลโครงแบบขนาด 4 หลัก 3 โปรแกรม สำหรับสัญญาณข้อมูลโปรแกรมโครงแบบ (Reconfiguration Program) ที่ได้นั้นจะมีขนาด 8 บิต

เนื่องจากเราได้แบ่งหน่วยควบคุมของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตออกเป็น 2 ส่วน ได้แก่ หน่วยควบคุมการเข้ารหัส และหน่วยควบคุมการเปลี่ยนโครงแบบ ซึ่งหน่วยควบคุมทั้งสองจะมีการทำงานสัมพันธ์กันดังแสดงในรูปที่ 3.14

รูปที่ 3.14 แผนภาพสถานะแสดงการทำงานระหว่างหน่วยควบคุมทั้งสอง

จากรูปที่ 3.14 ให้สถานะเริ่มต้นของหน่วยควบคุมการเปลี่ยนโครงแบบเป็น "เปลี่ยนโครงแบบเสร็จสิ้น (ReconfigDone)" และสถานะเริ่มต้นของหน่วยควบคุมการเข้ารหัสเป็น "เข้ารหัส (Encryption)" เมื่อต้องการจะเปลี่ยนโครงแบบของวงจรย่อยเป็นวงจรย่อยเอสบ็อกซ์ หน่วยควบคุมการเข้ารหัสจะเปลี่ยนสถานะจาก "เข้ารหัส" เป็น "เปลี่ยนโครงแบบเป็นวงจรย่อยเอสบ็อกซ์ (ReconfigSBox)" ซึ่งในสถานะนี้หน่วยควบคุมการเข้ารหัสจะส่งสัญญาณเลือกโครงแบบของวงจรย่อยเอสบ็อกซ์ เมื่อวงจรย่อยที่สามารถเปลี่ยนโครงแบบได้ที่ถูกโปรแกรมอยู่ในขณะนั้นได้รับสัญญาณควบคุมนี้ วงจรย่อยจะทำการตรวจสอบว่าตรงกับโครงแบบของตนหรือไม่ ถ้าตรงกันวงจรย่อยจะไม่ส่งสัญญาณบอกหน่วยควบคุมการเปลี่ยนโครงแบบ นั่นคือจะไม่เกิดการเปลี่ยนโครงแบบ แต่ถ้าไม่ตรงกัน วงจรย่อยจะส่งสัญญาณบอกหน่วยควบคุมการเปลี่ยนโครงแบบ เมื่อหน่วยควบคุมการเปลี่ยนโครงแบบทราบว่าวงจรย่อยที่สามารถเปลี่ยนโครงแบบได้ไม่ตรงกับสัญญาณเลือกโครงแบบจากหน่วยควบคุมการเข้ารหัส หน่วยควบคุมการเปลี่ยนโครงแบบก็จะเริ่มต้นการเปลี่ยนโครงแบบโดยเปลี่ยนสถานะของตนเองจาก "เปลี่ยนโครงแบบเสร็จสิ้น" เป็น "เปลี่ยนโครงแบบ (Reconfig)" ซึ่งจะทำการเปลี่ยนโครงแบบของวงจรย่อยให้เป็นวงจรที่ต้องการ โดยในระหว่างนี้หน่วยควบคุมการเข้ารหัสจะคงอยู่ในสถานะเดิม เมื่อการเปลี่ยนโครงแบบเสร็จสิ้น หน่วยควบคุมการเปลี่ยนโครงแบบจะเปลี่ยนสถานะจาก "เปลี่ยนโครงแบบ" เป็น "เปลี่ยนโครงแบบเสร็จสิ้น" พร้อมทั้งส่งสัญญาณบอกหน่วยควบคุมการเข้ารหัสว่าเปลี่ยนโครงแบบเสร็จแล้ว เมื่อหน่วยควบคุมการเข้ารหัสได้รับสัญญาณก็จะกลับเข้าสู่สถานะ "เข้ารหัส" และทำงานต่อไป ซึ่งการเปลี่ยนโครงแบบเป็นวงจรย่อยผสมหลัก และวงจรย่อยเอ็กซ์ออร์จะมีการทำงานเหมือนกับการเปลี่ยนโครงแบบเป็นวงจรย่อยเอสบ็อกซ์ แต่จะเปลี่ยนสถานะเป็น "เปลี่ยนโครงแบบเป็นวงจรย่อยผสมหลัก (ReconfigMixColumn)" และ "เปลี่ยนโครงแบบเป็นวงจรย่อยเอ็กซ์ออร์ (ReconfigXOR)" ตามลำดับ ซึ่งสถานะทั้งสองจะมีการทำงานเช่นเดียวกับสถานะ "เปลี่ยนโครงแบบเป็นวงจรย่อยเอสบ็อกซ์" แต่จะส่งสัญญาณเลือกโครงแบบของวงจรย่อยผสมหลัก และวงจรย่อยเอ็กซ์ออร์แทนตามลำดับ

3.2.2. วงจรย่อยเอสบ็อกซ์ (S-box Module)

วงจรย่อยเอสบ็อกซ์ทำหน้าที่แปลงข้อมูลที่ได้รับจากบัสแมโครอินพุต 1 แล้วส่งข้อมูลผลลัพธ์กลับไปยังบัสแมโครเอาท์พุต โดยการแปลงข้อมูลนี้เป็นไปตามการแปลงข้อมูลโดยเอสบ็อกซ์ของการเข้ารหัสเออีเอส ซึ่งเป็นดังตารางที่ 3.3 ซึ่งเป็นตารางแสดงการแทนที่ค่าเอ็กซ์วายด้วยค่าในตารางซึ่งเป็นเลขฐานสิบหก

วงจรย่อยเอสบ็อกซ์นี้จะ ไม่ใช้บล็อกแรม (BlockRAM) ซึ่งเป็นแรมสำเร็จรูปภายในเอฟพีจีเอ เนื่องจากจะทำให้ข้อมูลโครงแบบของวงจรย่อยมีขนาดใหญ่มาก เพราะต้องเปลี่ยนโครงแบบของบล็อกแรมซึ่งมีขนาดใหญ่และไม่สามารถใช้ร่วมกับวงจรย่อยอื่นๆได้ ในงานวิจัยนี้จึงเลือกใช้

แรมแบบกระจาย (Distributed RAM) ซึ่งใช้สไลซ์ (Slice) มาสร้างเป็นแรมที่มีขนาดใหญ่แทน เนื่องจากสไลซ์สามารถนำไปเปลี่ยนโครงแบบเป็นวงจรอื่นๆได้ง่าย การสร้างวงจรย่อยเอสบ็อกซ์จะใช้รอมขนาด 256x8 บิตที่สร้างจากแรมแบบกระจาย โดยจะใช้ข้อมูลจากบัสแมโครอินพุต 1 แทนตำแหน่งของรอมส่วนข้อมูลที่อ่านได้จากรอมจะเป็นผลลัพธ์และถูกส่งออกไปยังบัสแมโครเอาท์พุต

ตารางที่ 3.3 ตารางแสดงการแทนที่เอ็กซ์วายด้วยเอสบ็อกซ์ของเออีเอส (เลขฐานสิบหก)

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

3.2.3. วงจรย่อยผสมหลัก (MixColumn Module)

เนื่องจากวงจรย่อยผสมหลักนั้นทำงานกับข้อมูลขนาด 32 บิต แต่ทางเดินข้อมูลที่ใช้มีขนาด 8 บิต จึงต้องออกแบบวงจรให้มีฟลิปฟลอปเพื่อเก็บค่าไว้ให้ครบ 32 บิต และเมื่อพิจารณาเมทริกที่นำมาคูณกับแต่ละหลักในสมการข้างล่าง

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

$$for \quad 0 \le c \le Nb$$

$$S'_{0,c} = (S_{0,c} \bullet \{02\}) \oplus (S_{1,c} \bullet \{03\}) \oplus (S_{2,c} \bullet \{01\}) \oplus (S_{3,c} \bullet \{01\})$$
$$S'_{1,c} = (S_{1,c} \bullet \{02\}) \oplus (S_{2,c} \bullet \{03\}) \oplus (S_{3,c} \bullet \{01\}) \oplus (S_{0,c} \bullet \{01\})$$
$$S'_{2,c} = (S_{2,c} \bullet \{02\}) \oplus (S_{3,c} \bullet \{03\}) \oplus (S_{0,c} \bullet \{01\}) \oplus (S_{1,c} \bullet \{01\})$$
$$S'_{3,c} = (S_{3,c} \bullet \{02\}) \oplus (S_{0,c} \bullet \{03\}) \oplus (S_{1,c} \bullet \{01\}) \oplus (S_{2,c} \bullet \{01\})$$

ซึ่งสามารถเขียนเป็นสมการที่ให้ผลลัพธ์ทีละ 8 บิตโดยได้ดังนี้

$$S'_{x,c} = (S_{x,c} \bullet \{02\}) \oplus (S_{x+1 \bmod 4,c} \bullet \{03\}) \oplus (S_{x+2 \bmod 4,c} \bullet \{01\}) \oplus (S_{x+3 \bmod 4,c} \bullet \{01\})$$

$$for \quad 0 \le c \le Nb \quad and \quad 0 \le x \le 3$$

จากสมการ จะพบว่ากระบวนการผสมหลักสามารถนำมาสร้างเป็นวงจรที่รับข้อมูลขาเข้าทีละ 8 บิตโดยจะรับมาเก็บในเรจิสเตอร์แบบเลื่อน (Shift Register) 4 รอบสัญญาณนาฬิกาเพื่อให้ได้ข้อมูลตั้งต้น 32 บิต และนำไปเข้าวงจรคูณทั้งสี่และวงจรบวกเพื่อหาผลลัพธ์ของแถวแรกได้ จากนั้นเมื่อต้องการหาผลลัพธ์ของแถวถัดมาให้ทำการเลื่อนเรจิสเตอร์ไปหนึ่งตำแหน่งแล้วใช้มัลติเพลกเซอร์ (Multiplexor) เลือกข้อมูลที่จะนำไปทำเป็นผลลัพธ์

การใช้มัลติเพลกเซอร์เลือกแทนการหมุน (Rotate) นี้จะทำให้วงจรย่อยผสมหลักสามารถรับข้อมูลของหลักต่อไปเข้ามาได้ในขณะที่คำนวณผลลัพธ์ของหลักปัจจุบันอยู่ เนื่องจากเวลาที่ใช้ในการผสมหลักของข้อมูลสถานะแต่ละหลักจะต้องใช้เวลา 7 รอบสัญญาณนาฬิกา แบ่งเป็น 3 รอบสัญญาณนาฬิกาสำหรับอ่านข้อมูลล่วงหน้า 3 ชุดและอีก 4 รอบสัญญาณนาฬิกาสำหรับการคำนวณผลลัพธ์ 4 ชุด ซึ่งถ้าใช้การหมุนจะต้องใช้เวลาทั้งหมด 28 รอบสัญญาณนาฬิกาในการผสมหลักข้อมูลสถานะขนาด 128 บิต แต่ถ้าใช้วิธีเพิ่มความยาวของเรจิสเตอร์แบบเลื่อนแล้วใช้มัลติเพลกเซอร์จะใช้เวลาเพียง 19 รอบสัญญาณนาฬิกาเนื่องจากเสียเวลาอ่านข้อมูลล่วงหน้าเพียงหลักเดียว

เนื่องจากการคูณด้วย {02} ก็คือการเลื่อนไปทางซ้าย 1 บิตแล้วพิจารณาว่าบิตที่เกินมาเป็นหนึ่งหรือไม่ ถ้าเป็นหนึ่งให้นำผลลัพธ์ที่ได้ไปเอ็กซ์ออร์กับ {1B} ซึ่งเมื่อพิจารณาสมการข้างต้น จะพบว่าเราสามารถตัดขั้นตอนการคูณด้วย {03} ออกไปได้ ดังนี้

$$S'_{x,c} = (S_{x,c} \bullet \{02\}) \oplus (S_{x+1 \bmod 4,c} \bullet \{03\}) \oplus (S_{x+2 \bmod 4,c} \bullet \{01\}) \oplus (S_{x+3 \bmod 4,c} \bullet \{01\})$$

$$S'_{x,c} = (S_{x,c} \bullet \{02\}) \oplus (S_{x+1 \bmod 4,c} \bullet \{02\}) \oplus S_{x+1 \bmod 4,c} \oplus S_{x+2 \bmod 4,c} \oplus S_{x+3 \bmod 4,c}$$

$$S'_{x,c} = ((S_{x,c} \oplus S_{x+1 \bmod 4,c}) \bullet \{02\}) \oplus S_{x+1 \bmod 4,c} \oplus S_{x+2 \bmod 4,c} \oplus S_{x+3 \bmod 4,c}$$

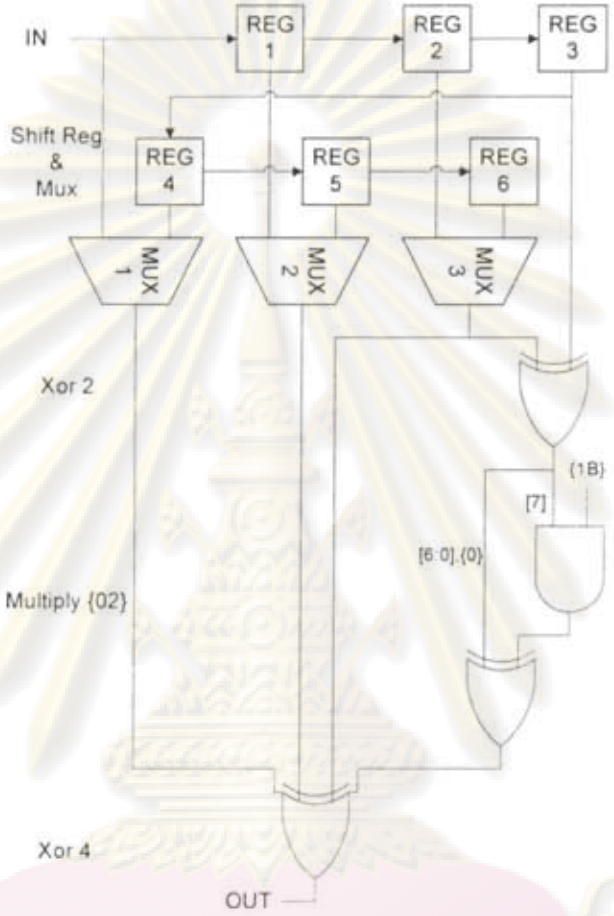
$$for \quad 0 \le c \le Nb \quad and \quad 0 \le x \le 3$$

นั่นหมายความว่าวงจรของกระบวนการผสมหลักจะสามารถออกแบบเป็นส่วนย่อยๆได้ 4 ส่วน โดยแต่ละส่วนมีรายละเอียดดังนี้

1. ฟลิปฟลอป 6 ตัวที่ต่อกันเป็นเรจิสเตอร์แบบเลื่อน (Shift Register) และมัลติเพลกเซอร์ (Multiplexor) - ทำหน้าที่รับข้อมูลและเป็นตัวป้อนข้อมูลให้ส่วนต่างๆ
2. วงจรเอ็กซ์ออร์ 2 – ทำหน้าที่เอ็กซ์ออร์ข้อมูลสองตัวที่จะต้องถูกคูณด้วย {02} และ {03} ก่อนจะนำไปเข้าวงจรคูณ {02}
3. วงจรคูณ {02} – เป็นวงจรเลื่อนซ้าย 1 บิต โดยถ้าบิตที่เกินมาเป็น 1 จะนำค่าที่ได้ไปเอ็กซ์ออร์กับ {1B}
4. วงจรเอ็กซ์ออร์ 4 – ทำหน้าที่เอ็กซ์ออร์ผลลัพธ์ที่ได้จากวงจรคูณและข้อมูลทั้งสาม

วงจรย่อยผสมหลักที่ได้จากการออกแบบตามที่กล่าวมาข้างต้นจะมีลักษณะเป็นดังรูปที่ 3.15 โดยจะเห็นว่าส่วนเรจิสเตอร์แบบเลื่อนและมัลติเพลกเซอร์จะส่งข้อมูลให้ส่วนต่างๆของวงจร โดยรายละเอียดของการส่งข้อมูลในแต่ละรอบของสัญญาณนาฬิกาจะเป็นดังตารางที่ 3.4 จากนั้นวงจรเอ็กซ์ออร์ 2 จะทำการเอ็กซ์ออร์ข้อมูลที่ได้รับแล้วส่งให้วงจรคูณ {02} ซึ่งเป็นเพียงการ

ตรวจสอบบิตหน้าสุดเพื่อเลือกว่าจะต้องบวก {1B} หรือไม่ โดยนำบิตหน้ากับ {1B} ไปและ (And) กันถ้าบิตหน้าเป็น 1 จะได้ผลลัพธ์เป็น {1B} แต่ถ้าบิตหน้าเป็น 0 จะได้ผลลัพธ์เป็น {00} จากนั้นจึงเอามาเอ็กซ์ออร์กับ 7 บิตที่เหลือซึ่งถูกเลื่อนไปทางซ้าย 1 บิตแล้วเติม 0 เข้าไปแทนบิตท้ายสุด จากนั้นจึงนำค่าที่ได้ทั้งหมดไปเอ็กซ์ออร์กันเป็นผลลัพธ์สุดท้ายของวงจรย่อยผสมหลัก



รูปที่ 3.15 แผนภาพแสดงวงจรของวงจรย่อยผสมหลัก

ตารางที่ 3.4 แสดงรายละเอียดการส่งข้อมูลของวงจรย่อยผสมหลักในแต่ละรอบสัญญาณนาฬิกา

CLOCK	IN	REG1	REG2	REG3	REG4	REG5	REG6	SELECT	MUX1	MUX2	MUX3
1	$S_{0,c}$	-	-	-	-	-	-	01	-	-	-
2	$S_{1,c}$	$S_{0,c}$	-	-	-	-	-	10	-	-	-
3	$S_{2,c}$	$S_{1,c}$	$S_{0,c}$	-	-	-	-	11	-	-	-
4	$S_{3,c}$	$S_{2,c}$	$S_{1,c}$	$S_{0,c}$	-	-	-	00	$S_{3,c}$	$S_{2,c}$	$S_{1,c}$
5	$S_{0,c+1}$	$S_{3,c}$	$S_{2,c}$	$S_{1,c}$	$S_{0,c}$	-	-	01	$S_{0,c}$	$S_{3,c}$	$S_{2,c}$
6	$S_{1,c+1}$	$S_{0,c+1}$	$S_{3,c}$	$S_{2,c}$	$S_{1,c}$	$S_{0,c}$	-	10	$S_{1,c}$	$S_{0,c}$	$S_{3,c}$
7	$S_{2,c+1}$	$S_{1,c+1}$	$S_{0,c+1}$	$S_{3,c}$	$S_{2,c}$	$S_{1,c}$	$S_{0,c}$	11	$S_{2,c}$	$S_{1,c}$	$S_{0,c}$

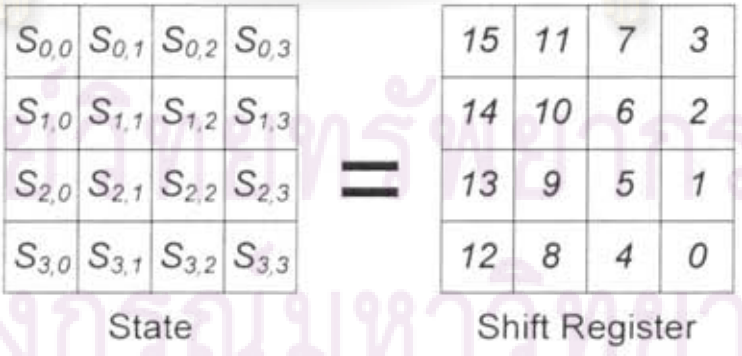
3.2.4. วงจรย่อยเอ็กซ์ออร์ (Xor Module)

วงจรย่อยเอ็กซ์ออร์เป็นวงจรที่ประกอบด้วยเกทเอ็กซ์ออร์ (Xor) จำนวน 8 เกท ทำหน้าที่บวกข้อมูลที่ได้รับจากบัสแมโครอินพุตทั้งสอง แล้วส่งผลลัพธ์กลับไปยังบัสแมโครเอาท์พุต

3.2.5. วงจรย่อยเรจิสเตอร์ (Register Module)

เนื่องจากวงจรย่อยเรจิสเตอร์มีหน้าที่เก็บข้อมูลสถานะและคีย์ของการเข้ารหัส ซึ่งข้อมูลทั้งสองมีขนาดข้อมูลละ 128 บิต ซึ่งต้องใช้ฟลิปฟลอปจำนวนมากในการเก็บข้อมูลทั้งสอง นอกจากนี้วงจรย่อยเรจิสเตอร์เป็นวงจรที่มีการติดต่อกับวงจรย่อยที่สามารถเปลี่ยนโครงแบบได้มากที่สุด เพราะทุกครั้งที่มีการใช้งานวงจรย่อยที่สามารถเปลี่ยนโครงแบบได้ จะต้องมีการส่งข้อมูลระหว่างวงจรย่อยเรจิสเตอร์กับบัสแมโครขนาด 8 บิต ซึ่งเล็กกว่าข้อมูลแต่ละตัวถึง 16 เท่า นั่นหมายความว่าวงจรย่อยเรจิสเตอร์จะต้องมีมัลติเพลกเซอร์ (Multiplexor) และดีโคดเดอร์ (Decoder) จำนวนมาก วงจรที่ได้จึงมีขนาดใหญ่และวุ่นวายเนื่องจากมีการเชื่อมต่อกันระหว่างส่วนต่างๆภายในวงจรเยอะ ซึ่งจะทำให้การสร้างบัสแมโครเป็นไปได้อย่างยากลำบาก

เพื่อออกแบบวงจรย่อยเรจิสเตอร์ที่มีขนาดเล็กและไม่วุ่นวาย การใช้เรจิสเตอร์แบบเลื่อน (Shift Register) ซึ่งเป็นการเก็บข้อมูลแบบเข้าก่อนออกก่อน (First-In-First-Out : FIFO) ซึ่งเปรียบเสมือนแถวคอยหรือคิว (Queue) จะช่วยลดการใช้มัลติเพลกเซอร์และดีโคดเดอร์ได้ นอกจากนี้ยังลดจำนวนสัญญาณควบคุมที่ใช้ในการเลือกอีกด้วย โดยจะให้เรจิสเตอร์แบบเลื่อนเก็บข้อมูลสถานะ โดยจะเรียงลำดับข้อมูลในเรจิสเตอร์โดยใช้หลักเป็นใหญ่จากหลักสุดท้ายแถวสุดท้ายมายังหลักแรกแถวแรกดังแสดงในรูปที่ 3.16 การเรียงลำดับข้อมูลแบบนี้จะทำให้ข้อมูลสถานะในหลักแรกแถวแรกออกมาก่อน และจะถูกเก็บเป็นตัวแรกตามด้วยหลักแรกแถวที่สอง ซึ่งการเรียงลำดับข้อมูลเช่นนี้จะช่วยให้การทำงานของวงจรย่อยผสมหลักเป็นไปได้อย่างสะดวก เนื่องจากวงจรย่อยผสมหลักต้องการข้อมูลทั้งหลักเข้ามาติดต่อกัน หลังจากนั้นจะให้ข้อมูลของหลักนั้นกลับออกมาอย่างต่อเนื่อง



รูปที่ 3.16 แผนภาพแสดงลำดับการเก็บข้อมูลสถานะในเรจิสเตอร์แบบเลื่อน

นอกจากนี้เมื่อพิจารณาการทำงานของวงจรย่อยเรจิสเตอร์แล้วจะพบว่าวงจรนี้ต้องทำหน้าที่แทนวงจรเลื่อนด้วย โดยในส่วนของข้อมูลสถานะกระบวนการเลื่อนแถวจะทำให้ลำดับของการส่งข้อมูลไม่เหมือนกระบวนการอื่น เมื่อพิจารณาการใช้เรจิสเตอร์แบบเลื่อนที่มีอยู่ในเอฟพีจีเอรุ่น Spartan-3 ซึ่งมีวงจรเป็นดังรูปที่ 3.17 จะพบว่าการเปลี่ยนลำดับของข้อมูลขาออกสามารถทำได้โดยการเปลี่ยนสัญญาณควบคุมมัลติเพลกเซอร์ของเรจิสเตอร์แบบเลื่อน

รูปที่ 3.17 แผนภาพแสดงวงจรเรจิสเตอร์แบบเลื่อนภายในเอฟพีจีเอรุ่น Spartan-3

เมื่อพิจารณาลำดับในการส่งข้อมูลสถานะไปยังวงจรย่อยที่สามารถเปลี่ยนโครงแบบได้เพื่อทำกระบวนการเลื่อนแถว จะพบว่าตำแหน่งของข้อมูลก่อนและหลังกระบวนการเลื่อนแถวเป็นดังรูปที่ 3.18 นั่นหมายความว่าลำดับในการส่งข้อมูลสถานะจะเป็นดังรูปที่ 3.19

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

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

รูปที่ 3.18 แผนภาพแสดงตำแหน่งของข้อมูลสถานะก่อนและหลังกระบวนการเลื่อนแถว

รูปที่ 3.19 แผนภาพแสดงลำดับในการส่งข้อมูลสถานะเพื่อทำกระบวนการเลื่อนแถว

เมื่อพิจารณาลำดับของการส่งข้อมูลสถานะเพื่อทำกระบวนการเลื่อนแถวเป็นเลขฐานสอง จะพบว่า ลำดับของการส่งข้อมูลสถานะเพื่อทำกระบวนการเลื่อนแถวนี้สามารถเขียนเป็นสมการลำดับได้ดังนี้

$$a_n = a_{n-1} - 0101_2$$

และเนื่องจากไม่สามารถเลื่อนเรจิสเตอร์แบบเลื่อนในขณะที่ทำกระบวนการเลื่อนแถวโดยการเปลี่ยนค่ามัลติเพลกเซอร์ในเรจิสเตอร์แบบเลื่อน เพราะจะทำให้ข้อมูลบางส่วนสูญหาย ในการเก็บข้อมูลสถานะจึงจำเป็นต้องใช้เรจิสเตอร์แบบเลื่อนขนาด 16x8 บิตจำนวน 2 ตัวเพื่อให้ผลัดกันอ่านและเขียนในกระบวนการเลื่อนแถวของแต่ละรอบ ซึ่งการเพิ่มเรจิสเตอร์นี้จะไม่สิ้นเปลืองวงจรมากนักเพราะเราสามารถใช้สไลซ์เอ็ม (SliceM) เพียง 8 สไลซ์ในการสร้างเรจิสเตอร์แบบเลื่อนขนาด 16x8 บิต 1 ตัว

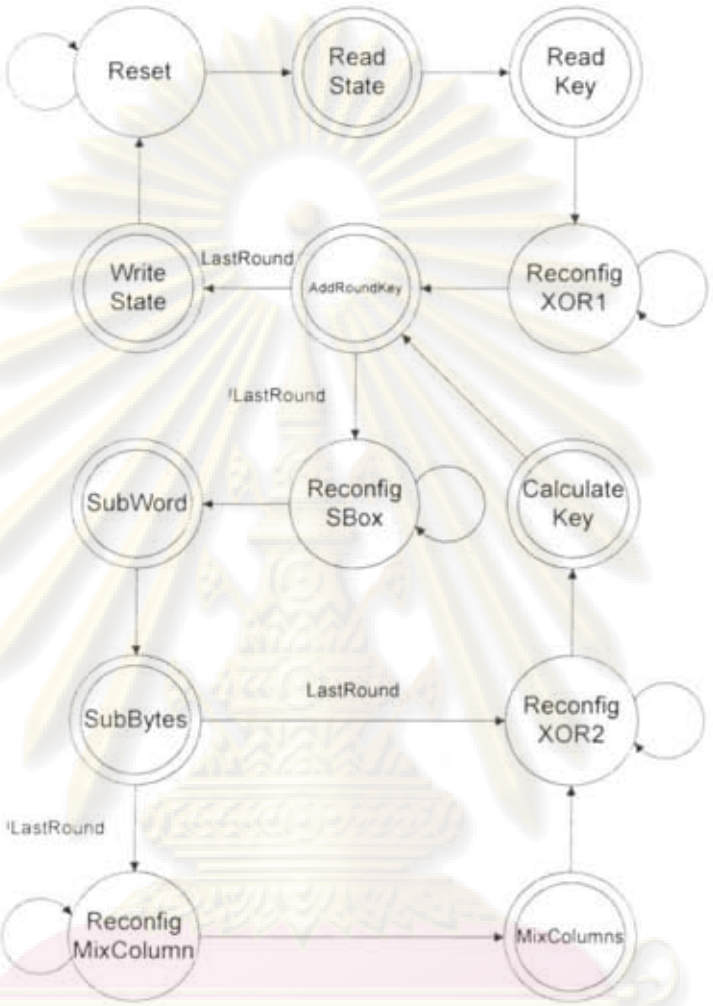
แต่กระบวนการขยายคีย์นั้นมีการทำงานที่ซับซ้อนกว่ากระบวนการรหัสเพราะมีกระบวนการเลื่อนคำที่กระทำกับข้อมูลคีย์เพียงหลักเดียว กระบวนการบวกค่าคงที่ของแต่ละรอบที่กระทำกับข้อมูลตำแหน่งเดียวเท่านั้น และกระบวนการบวกคีย์ซึ่งต้องใช้ข้อมูลคีย์ที่ผ่านกระบวนการก่อนหน้าแล้วกับข้อมูลคีย์ที่ยังไม่ผ่านกระบวนการก่อนหน้า ดังนั้นการใช้แรมซึ่งสามารถสร้างได้จากสไลซ์เอ็มจึงเป็นแนวคิดที่ง่ายที่สุดในการจัดการความยุ่งยาก โดยจะใช้แรมขนาด 16x8 บิต เพื่อเก็บข้อมูลคีย์ และแรมขนาด 4x8 บิตเพื่อเก็บข้อมูลคีย์ชั่วคราว โดยคีย์ชั่วคราวนี้จะเป็นแหล่งข้อมูลหลักที่ใช้ในกระบวนการขยายคีย์ดังแสดงในรูปที่ 3.7 คือใช้เป็นข้อมูลตั้งต้นในกระบวนการเลื่อนคำ, กระบวนการบวกค่าคงที่ของแต่ละรอบและกระบวนการบวกคีย์ ซึ่งแรมขนาด 16x8 บิตและแรมขนาด 4x8 บิตนี้สามารถสร้างได้จากสไลซ์เอ็ม 8 สไลซ์ทั้งคู่ สำหรับค่าคงที่ของแต่ละรอบนั้นจะถูกเก็บไว้ในรอมขนาด 10x8 บิตที่สร้างจากสไลซ์เอ็ม 8 สไลซ์เช่นเดียวกันเพื่อให้ได้วงจรที่มีขนาดเล็ก

เมื่อพิจารณาถึงการส่งข้อมูลจากวงจรย่อยเรจิสเตอร์ไปยังบัสแมโครอินพุตทั้งสอง ได้แก่ บัสแมโครอินพุต 1 และบัสแมโครอินพุต 2 จะพบว่าข้อมูลที่จะส่งให้บัสแมโครทั้งสองนั้นได้ถูกกำหนดโดยวงจรย่อยที่จะนำข้อมูลไปใช้ ซึ่งมีรายละเอียดดังนี้

1. วงจรย่อยเอสบ็อกซ์ต้องการข้อมูลสถานะและข้อมูลคีย์จากบัสแมโครอินพุต 1
2. วงจรย่อยผสมหลักต้องการข้อมูลสถานะจากบัสแมโครอินพุต 1
3. วงจรย่อยเอ็กซ์ออร์ต้องการข้อมูลสถานะและข้อมูลคีย์จากบัสแมโครทั้งสองพร้อมๆกัน ข้อมูลคีย์และข้อมูลคีย์ชั่วคราวจากบัสแมโครทั้งสองพร้อมๆกัน และข้อมูลค่าคงที่ของแต่ละรอบและข้อมูลคีย์ชั่วคราวจากบัสแมโครทั้งสองพร้อมๆกัน

ดังนั้นวงจรย่อยเรจิสเตอร์จะต้องสามารถส่งข้อมูลสถานะซึ่งได้จากเรจิสเตอร์แบบเลื่อน 2 ตัว, ข้อมูลคีย์ และข้อมูลคีย์ชั่วคราวไปยังบัสแมโครอินพุต 1 และส่งข้อมูลคีย์ และข้อมูลค่าคงที่ของ

แต่ละรอบไปยังบัสแมโครอินพุต 2 นั่นหมายความว่าในการออกแบบวงจรย่อยเรจิสเตอร์นี้สามารถใช้มัลติเพลกเซอร์แบบ 4 เลือก 1 เพื่อเลือกข้อมูลส่งให้บัสแมโครอินพุต 1 และใช้มัลติเพลกเซอร์แบบ 2 เลือก 1 เพื่อเลือกข้อมูลส่งให้บัสแมโครอินพุต 2



รูปที่ 3.20 แผนภาพสถานะแสดงการทำงานของหน่วยควบคุมการเข้ารหัสอย่างย่อ

3.2.6. หน่วยควบคุมการเข้ารหัส (Encryption Control Unit)

หน่วยควบคุมการเข้ารหัสมีหน้าที่ควบคุมการทำงานทั้งหมดของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต ยกเว้นการเปลี่ยนโครงแบบ โดยการควบคุมกระบวนการเปลี่ยนโครงแบบจะเป็นหน้าที่ของหน่วยควบคุมการเปลี่ยนโครงแบบ โดยหน่วยควบคุมการเข้ารหัสที่ได้ออกแบบไว้จะมีการทำงานหลายขั้นตอนซึ่งจะสามารถอธิบายได้ด้วยแผนภาพสถานะได้ดังแสดงในรูปที่ 3.20 ซึ่งจากรูปจะเห็นว่าหน่วยควบคุมการเข้ารหัสสามารถแบ่งออกเป็นสถานะหลักๆได้ 13 สถานะ ดังนี้

1. สถานะรีเซ็ท (Reset State) – สถานะนี้เป็นสถานะว่าง ไม่มีการทำงานใดๆ ใช้ในการรอคำสั่งเข้ารหัสจากภายนอกวงจร เมื่อมีการสั่งให้เข้ารหัสจะเปลี่ยนสถานะเป็น "สถานะอ่านข้อมูลสถานะ" เพื่อเริ่มขั้นตอนการเข้ารหัส

2. สถานะอ่านข้อมูลสถานะ (ReadState State) – สถานะนี้จะประกอบด้วย 16 สถานะย่อยๆ ซึ่งทำหน้าที่อ่านข้อมูลที่ต้องการจะเข้ารหัสจากภายนอกแล้วนำมาเก็บไว้ในวงจรย่อยเรจิสเตอร์ ซึ่งจะต้องใช้ 16 สถานะ เพราะสามารถอ่านข้อมูลได้ครั้งละ 8 บิตเท่านั้น
3. สถานะอ่านข้อมูลคีย์ (ReadKey State) - สถานะนี้จะประกอบด้วย 16 สถานะย่อยๆ ซึ่งทำหน้าที่อ่านคีย์ของการเข้ารหัสจากภายนอกแล้วนำมาเก็บไว้ในวงจรย่อยเรจิสเตอร์ ซึ่งจะต้องใช้ 16 สถานะ เพราะสามารถอ่านข้อมูลได้ครั้งละ 8 บิตเท่านั้น
4. สถานะเขียนข้อมูลสถานะ (WriteState State) - สถานะนี้จะประกอบด้วย 16 สถานะย่อยๆ ซึ่งทำหน้าที่ส่งข้อมูลที่เข้ารหัสเรียบร้อยไปสู่ภายนอก ซึ่งจะต้องใช้ 16 สถานะ เพราะสามารถเขียนข้อมูลได้ครั้งละ 8 บิตเท่านั้น
5. สถานะแทนที่ไบต์ (SubBytes State) – สถานะนี้จะประกอบด้วย 16 สถานะย่อยๆ ทำหน้าที่แปลงสถานะของข้อมูลในแต่ละรอบทีละ 8 บิตด้วยกระบวนการแทนที่ไบต์ นอกจากนี้ยังทำกระบวนการเลื่อนแถวไปพร้อมๆกันด้วย โดยการเลือกส่งข้อมูลตามวิธีที่ได้ออกแบบไว้ในหัวข้อ 3.2.5 ทำให้ไม่ต้องมีสถานะสำหรับกระบวนการเลื่อนแถว ซึ่งเมื่อถึงสถานะย่อยที่ 16 จะมีการตรวจสอบว่าเป็นการเข้ารหัสรอบสุดท้ายหรือไม่ ถ้าเป็นการเข้ารหัสรอบสุดท้ายจะไม่เปลี่ยนสถานะเป็น "สถานะเปลี่ยนโครงแบบเป็นวงจรย่อยผสมหลัก" ตามปกติ แต่จะเปลี่ยนสถานะเป็น "สถานะเปลี่ยนโครงแบบเป็นวงจรย่อยเอ็กซ์ออร์" เพื่อข้ามขั้นตอนการผสมหลักในรอบสุดท้าย
6. สถานะผสมหลัก (MixColumns State) – สถานะนี้จะประกอบด้วย 19 สถานะย่อยๆ ทำหน้าที่แปลงสถานะของข้อมูลในแต่ละรอบด้วยกระบวนการผสมหลัก โดยจะแบ่งเป็น 3 สถานะสำหรับการอ่านข้อมูลล่วงหน้า และอีก 16 สถานะสำหรับกระบวนการผสมหลักข้อมูลสถานะทั้ง 128 บิต
7. สถานะบวกคีย์แต่ละรอบ (AddRoundKey State) - สถานะนี้จะประกอบด้วย 16 สถานะย่อยๆ ทำหน้าที่แปลงสถานะของข้อมูลในแต่ละรอบทีละ 8 บิตด้วยกระบวนการบวกคีย์แต่ละรอบ ซึ่งเมื่อถึงสถานะย่อยที่ 16 จะมีการตรวจสอบว่าเป็นการเข้ารหัสรอบสุดท้ายหรือไม่ ถ้าเป็นการเข้ารหัสรอบสุดท้ายจะเปลี่ยนสถานะเป็น "สถานะเขียนข้อมูลสถานะ" แต่ถ้าไม่ใช่การเข้ารหัสรอบสุดท้ายก็จะเปลี่ยนสถานะเป็น "สถานะเปลี่ยนโครงแบบเป็นวงจรย่อยเอสบ็อกซ์" เพื่อทำการเข้ารหัสรอบต่อไป
8. สถานะแทนที่คำ (SubWord State) - สถานะนี้จะประกอบด้วย 4 สถานะย่อยๆ ทำหน้าที่แปลงคีย์ในแต่ละรอบทีละ 8 บิตด้วยกระบวนการแทนที่คำ นอกจากนี้ยังทำกระบวนการเลื่อนคำไปพร้อมๆกันด้วย โดยการเลือกส่งข้อมูลตามวิธีที่ได้ออกแบบไว้ในหัวข้อ 3.2.5 ทำให้ไม่ต้องมีสถานะสำหรับกระบวนการเลื่อนคำ

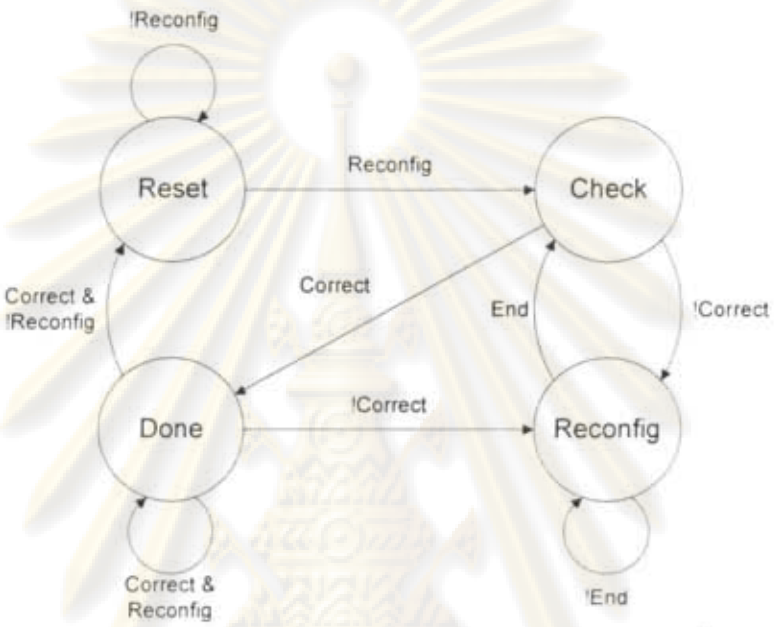
9. สถานะคำนวณคีย์ (CalculateKey State) - สถานะนี้จะประกอบด้วย 26 สถานะย่อยๆ โดยจะแบ่งเป็น 10 สถานะย่อย สำหรับกระบวนการบวกค่าคงที่ของแต่ละรอบ แต่จะมีเพียงสถานะย่อยเดียวเท่านั้นที่ถูกเลือกให้ทำงานในแต่ละรอบของการเข้ารหัส เพราะค่าคงที่ของแต่ละรอบมีค่าไม่เท่ากันจึงต้องมีสถานะแยกสำหรับแต่ละรอบเพื่อทำการเลือกค่าคงที่ที่จะนำมาบวกในแต่ละรอบ โดยกระบวนการบวกค่าคงที่ของแต่ละรอบนี้จะทำกับข้อมูลคีย์เพียง 8 บิตเท่านั้นในแต่ละรอบ ส่วนสถานะที่เหลืออีก 16 สถานะย่อยจะทำหน้าที่แปลงข้อมูลคีย์ด้วยกระบวนการบวกคีย์ซึ่งกระบวนการนี้จะกระทำกับข้อมูลคีย์ทั้ง 128 บิต โดยจะกระทำทีละ 8 บิต
10. สถานะเปลี่ยนโครงแบบเป็นวงจรย่อยเอสบ็อกซ์ (ReconfigSBox State) - เป็นสถานะที่ใช้ในการเปลี่ยนโครงแบบเป็นวงจรย่อยเอสบ็อกซ์ ซึ่งจะต้องให้รอการทำงานของหน่วยควบคุมการเปลี่ยนโครงแบบทำงานเสร็จสิ้นก่อนจึงจะเปลี่ยนสถานะเพื่อทำงานต่อไปได้ตามที่ได้ออกแบบไว้ในหัวข้อ 3.2.1
11. สถานะเปลี่ยนโครงแบบเป็นวงจรย่อยผสมหลัก (ReconfigMixColumn State) - เป็นสถานะที่ใช้ในการเปลี่ยนโครงแบบเป็นวงจรย่อยผสมหลัก ซึ่งจะต้องให้รอการทำงานของหน่วยควบคุมการเปลี่ยนโครงแบบทำงานเสร็จสิ้นก่อนจึงจะเปลี่ยนสถานะเพื่อทำงานต่อไปได้ตามที่ได้ออกแบบไว้ในหัวข้อ 3.2.1
12. สถานะเปลี่ยนโครงแบบเป็นวงจรย่อยเอ็กซ์ออร์ 1 (ReconfigXOR1 State) - เป็นสถานะที่ใช้ในการเปลี่ยนโครงแบบเป็นวงจรย่อยเอ็กซ์ออร์ ซึ่งจะต้องให้รอการทำงานของหน่วยควบคุมการเปลี่ยนโครงแบบทำงานเสร็จสิ้นก่อนจึงจะเปลี่ยนสถานะเพื่อทำงานต่อไปได้ตามที่ได้ออกแบบไว้ในหัวข้อ 3.2.1
13. สถานะเปลี่ยนโครงแบบเป็นวงจรย่อยเอ็กซ์ออร์ 2 (ReconfigXOR2 State) – เป็นสถานะที่ใช้ในการเปลี่ยนโครงแบบเป็นวงจรย่อยเอ็กซ์ออร์เหมือนกับ "สถานะเปลี่ยนโครงแบบเป็นวงจรย่อยเอ็กซ์ออร์ 1" แต่เนื่องสถานะที่จะเปลี่ยนภายหลังจากการเปลี่ยนโครงแบบเสร็จสิ้นไม่เหมือนกันจึงได้แบ่งออกเป็น 2 สถานะ

จะเห็นได้ว่าหน่วยควบคุมการเข้ารหัสนี้มีสถานะทั้งหมด 134 สถานะ และสถานะส่วนมากนั้นใช้กำหนดมีสัญญาณควบคุมสัญญาณเดียวกัน ซึ่งทำให้การออกแบบเครื่องสถานะจำกัด (Finite State Machine) ที่เข้ารหัสสถานะแบบเลขฐานสอง (Binary Encoding) นั้นมีขนาดใหญ่เนื่องจากจำนวนสถานะที่จะต้องนำไปแปลงเป็นสัญญาณควบคุมมีจำนวนมาก ในการออกแบบหน่วยควบคุมการเข้ารหัสของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตนี้จึงเลือกการเข้ารหัสสถานะด้วยวิธีการวันฮอท (One-hot Encoding) โดยการเข้ารหัสสถานะแบบวันฮอทนี้จะใช้ฟลิปฟลอปหนึ่งตัวต่อหนึ่งสถานะ โดยในขณะใดขณะหนึ่งจะมีฟลิปฟลอปเพียงตัวเดียวเท่านั้นที่เป็น 1 ซึ่งจะทำให้วงจรแปลงสถานะเป็นสัญญาณควบคุมมีขนาดเล็กและสามารถทำงานได้อย่างรวดเร็ว

เนื่องจากการแปลงสถานะเป็นสัญญาณควบคุมจะใช้เพียงเกทหรือ (Or) เพียงหนึ่งเกทเท่านั้นต่อหนึ่งสัญญาณควบคุม

3.2.7. หน่วยควบคุมการเปลี่ยนโครงแบบ (Reconfiguration Control Unit)

หน่วยควบคุมการเปลี่ยนโครงแบบเป็นเครื่องสถานะจำกัด (Finite State Machine : FSM) อย่างง่ายที่มีสถานะทั้งหมดเพียง 4 สถานะ ดังรูปที่ 3.21



รูปที่ 3.21 แผนภาพสถานะแสดงการทำงานของหน่วยควบคุมการเปลี่ยนโครงแบบ

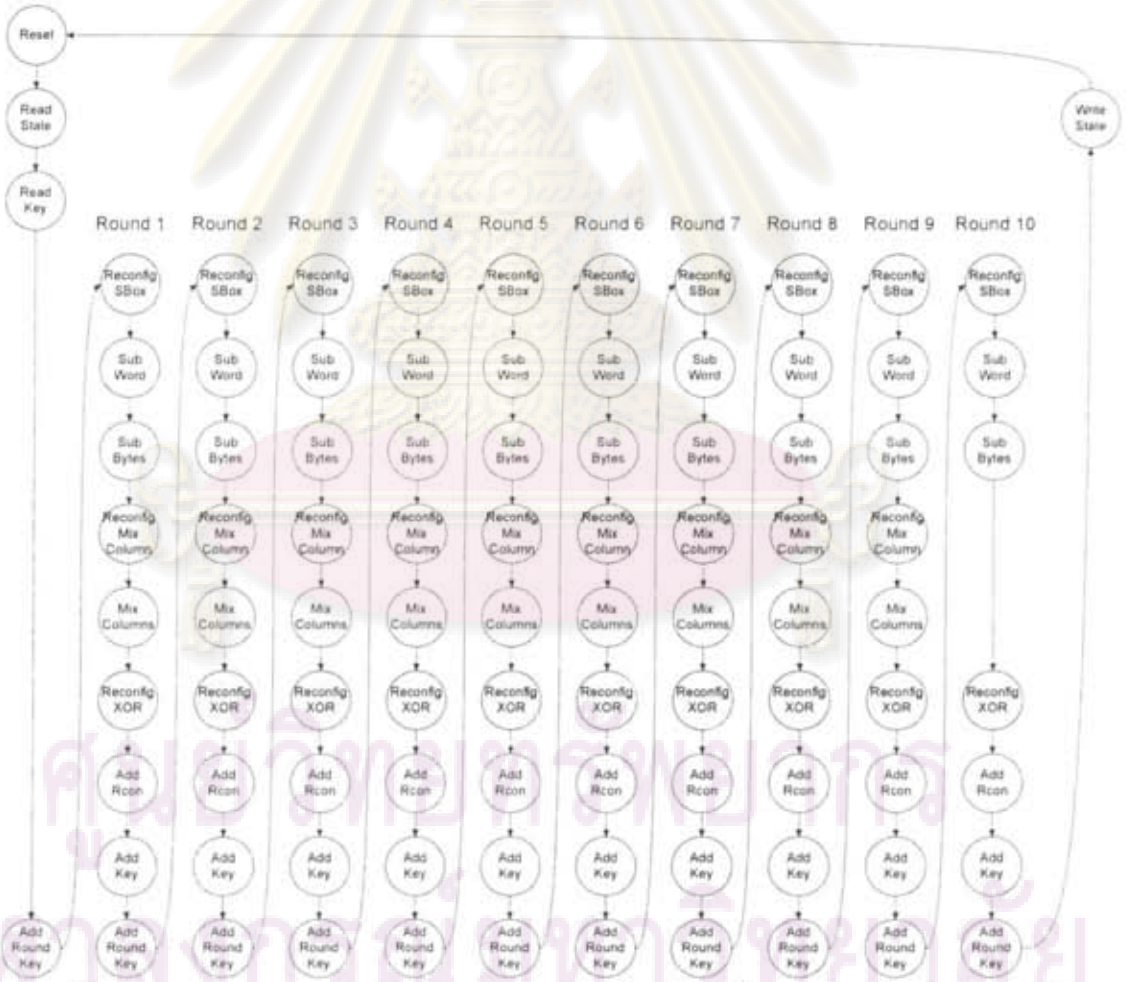
จากรูปที่ 3.21 เมื่อหน่วยควบคุมการเปลี่ยนโครงแบบได้รับสัญญาณสั่งให้เปลี่ยนโครงแบบจากหน่วยควบคุมการเข้ารหัส หน่วยควบคุมจะเปลี่ยนสถานะจาก "รีเซต (Reset)" เป็น "ตรวจสอบ (Check)" ซึ่งสถานะจะทำการตรวจสอบวงจรย่อยว่าตรงกับวงจรย่อยที่ต้องการจะเปลี่ยนโครงแบบไปหรือไม่ ถ้าตรงกันหน่วยควบคุมการเปลี่ยนโครงแบบจะเปลี่ยนสถานะกลับเป็น "เสร็จ (Done)" แต่ถ้าไม่ตรงกันก็จะเริ่มทำการเปลี่ยนโครงแบบโดยเปลี่ยนสถานะเป็น "เปลี่ยนโครงแบบ (Reconfig)" ซึ่งในสถานะนี้หน่วยควบคุมจะส่งสัญญาณควบคุมการเปลี่ยนโครงแบบ CS_B เป็นศูนย์หรือต่ำออกไปเพื่อสั่งให้เอฟพีจีเอเปลี่ยนโครงแบบตามรูปที่ 3.11 พร้อมทั้งทำการเปลี่ยนตำแหน่งของหน่วยความจำโครงแบบไปเรื่อยๆเพื่อเลือกตำแหน่งของหน่วยความจำโครงแบบที่ถูกต้อง การเปลี่ยนโครงแบบจะเสร็จสิ้นเมื่อส่งข้อมูลโครงแบบที่ต้องการไปยังเอฟพีจีเอครบถ้วนซึ่งจะทำให้เกิดการเปลี่ยนสถานะเป็น "ตรวจสอบ" อีกครั้งจนกว่าจะเปลี่ยนโครงแบบเป็นวงจรที่ต้องการได้อย่างถูกต้อง สำหรับสถานะ "เสร็จ" จะมีการตรวจสอบโครงแบบเช่นเดียวกันสถานะ "ตรวจสอบ" ถ้าวงจรย่อยที่ได้ยังไม่ตรงกับที่ต้องการจะมีการเปลี่ยนโครงแบบอีกครั้งโดยการเปลี่ยนสถานะกลับไปเป็น "เปลี่ยนโครงแบบ" แต่ถ้าถูกต้องแล้วจะมีการส่งสัญญาณบอกหน่วย

ควบคุมการเข้ารหัส และจะรอจนกว่าสัญญาณเปลี่ยนโครงแบบจะถูกยกเลิกจึงจะเปลี่ยนสถานะกลับเป็น "รีเซต"

3.2.8. หน่วยความจำโครงแบบ (Reconfiguration Memory)

เป็นหน่วยความจำภายนอกเอฟพีจีเอตามที่ได้ออกแบบไว้ในหัวข้อ 3.1.3 ซึ่งในงานวิจัยนี้ได้เลือกใช้แรมจากภายนอก เนื่องจากการอ่านข้อมูลจากแรมสามารถทำได้รวดเร็วและสามารถอ่านข้อมูลแบบขนานทีละ 8 บิตจากตำแหน่งใดๆก็ได้ ซึ่งสะดวกสำหรับการเลือกโปรแกรมโครงแบบ ซึ่งสัญญาณสำหรับการควบคุมการทำงานของแรมจะถูกบังคับไว้ให้เป็นอ่านตลอดการทำงานของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้ และตำแหน่งของข้อมูลจะได้จากหน่วยควบคุมการเปลี่ยนโครงแบบภายในเอฟพีจีเอ และข้อมูลโครงแบบที่อ่านได้จะถูกต่อตรงเข้าไปยังขาที่ใช้ในการเปลี่ยนโครงแบบของเอฟพีจีเอ ซึ่งจะทำให้การออกแบบเอฟพีจีเอสะดวกขึ้น

3.3. พฤติกรรมของวงจร



รูปที่ 3.22 แผนภาพแสดงการทำงานของวงจรเข้ารหัสเออีเอสที่สามารถเปลี่ยนโครงแบบได้

จากการออกแบบที่ได้กล่าวไว้ข้างต้น จะได้วงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตที่มีพฤติกรรมที่สามารถอธิบายได้ด้วยแผนภาพแสดงการทำงานในรูปที่ 3.22 โดยการทำงานของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตจะเริ่มต้นที่การรีเซต (Reset) เมื่อมีการสั่งให้เข้ารหัสจะเริ่มรับข้อมูลสถานะหรือข้อมูลที่ต้องการจะเข้ารหัส (ReadState) ตามด้วยข้อมูลคีย์ (ReadKey) จากนั้นจะทำการแปลงข้อมูลด้วยกระบวนการบวกคีย์แต่ละรอบ (AddRoundKey) ซึ่งเป็นการเริ่มต้นการเข้ารหัสแบบเออีเอสจากนั้นจะทำการเปลี่ยนโครงแบบเป็นวงจรย่อยเอสบ็อกซ์ (ReconfigSBox) ทำกระบวนการแทนที่คำ (SubWord) และกระบวนการแทนที่ไบต์ (SubBytes) ไปพร้อมๆกับกระบวนการเลื่อนคำ (ShiftWord) และกระบวนการเลื่อนแถว (ShiftRows) ตามลำดับ จากนั้นจึงเปลี่ยนโครงแบบเป็นวงจรย่อยผสมหลัก (ReconfigMixColumn) เพื่อทำกระบวนการผสมหลัก (MixColumns) เมื่อทำงานเสร็จแล้วจะเปลี่ยนโครงแบบเป็นวงจรย่อยเอ็กซ์ออร์ (ReconfigXOR) เพื่อทำกระบวนการบวกค่าคงที่ของแต่ละรอบ (AddRcon) กระบวนการบวกคีย์ (AddKey) และกระบวนการบวกคีย์แต่ละรอบ (AddRoundKey) ตามลำดับและจะทำเช่นนี้ไปจนครบ 10 รอบตามมาตรฐานเออีเอส โดยในรอบที่ 10 จะไม่มีการเปลี่ยนโครงแบบเป็นวงจรย่อยผสมหลักและไม่มีการแปลงข้อมูลสถานะด้วยกระบวนการผสมหลัก เมื่อทำการเข้ารหัสเสร็จสิ้นแล้วจะทำการส่งข้อมูลสถานะที่เข้ารหัสแล้วออกไป (WriteState) และกลับเข้าสู่การรีเซตอีกครั้ง ซึ่งจะเห็นได้ว่ากระบวนการที่เป็นสีเทาในรูปที่ 3.22 คือกระบวนการที่เพิ่มขึ้นมาจากการเข้ารหัสเออีเอสตามปกติ

3.4. ประสิทธิภาพของวงจร

ประสิทธิภาพของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตสามารถวัดได้จากคุณสมบัติของวงจร 2 อย่าง คือ

1. ขนาดของวงจร
2. ความเร็วในการทำงาน

โดยขนาดของวงจรซึ่งเป็นจุดประสงค์หลักของการพัฒนาวงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัตในงานวิจัยนี้ สามารถวัดได้จากจำนวนสไลซ์ (Slice) ที่ใช้ไป ซึ่งเป็นการวัดขนาดของวงจรที่เหมาะสมกับการออกแบบวงจรบนเอฟพีจีเอ เพราะเป็นการแสดงจำนวนทรัพยากรที่ใช้จริงบนเอฟพีจีเอแต่ก็มีข้อเสียคือขนาดสไลซ์ของเอฟพีจีเอแต่ละรุ่นอาจจะไม่เท่ากันทำให้ไม่สามารถเปรียบเทียบผลลัพธ์กันได้ ต่างจากการวัดขนาดวงจรโดยใช้จำนวนเกตสมมูล (Equivalent Gates) ซึ่งเป็นการแปลงวงจรที่ใช้ให้เป็นจำนวนเกตที่ต้องใช้ในการสร้างวงจร จึงสามารถเปรียบเทียบผลลัพธ์ได้ แต่เนื่องจากการเปลี่ยนโครงแบบนั้นเป็นคุณสมบัติเฉพาะตัวของเอฟพีจีเอแต่ละรุ่นอยู่แล้ว และในงานวิจัยนี้เราอาศัยความสามารถในการเปลี่ยนโครงแบบได้ของเอฟพีจีเอ การวัดขนาด

ของวงจรโดยการนับจำนวนสไลซ์จึงเป็นวิธีที่เหมาะสมกว่าการนับจำนวนเกตสมมูล โดยวงจรที่มีขนาดใหญ่จะใช้สไลซ์เป็นจำนวนมากกว่าวงจรที่มีขนาดเล็ก

สำหรับการวัดความเร็วในการทำงานซึ่งถึงแม้ว่าไม่ใช่จุดมุ่งหมายหลักของงานวิจัยนี้ แต่ก็ต้องให้ความสนใจ เพราะถ้าวงจรที่ได้ไม่สามารถทำงานได้ภายในเวลาที่สามารถรับได้ ก็อาจถือได้ว่าวงจรนั้นไม่มีประสิทธิภาพ ซึ่งการวัดประสิทธิภาพที่เหมาะสมสำหรับวงจรเข้ารหัสก็คือการวัดปริมาณงานที่สามารถทำได้ในช่วงเวลา ซึ่งวิธีในการวัดปริมาณงานนี้เราสามารถวัดได้โดยการนับจำนวนข้อมูลที่สามารถเข้ารหัสได้จริง หรือวัดโดยการวิเคราะห์และคำนวณจากความถี่ของสัญญาณนาฬิกาที่ใช้ในวงจรและการทำงานของวงจร โดยในงานวิจัยนี้จะใช้การวัดปริมาณงานจากการวิเคราะห์และคำนวณโดยนำค่าความถี่ที่วงจรสามารถทำงานได้จากการทดลองมาคำนวณด้วยสมการที่ได้จากการวิเคราะห์การทำงานของวงจร ซึ่งจะได้เป็นปริมาณงานที่วงจรสามารถทำได้ รายละเอียดของสมการที่ใช้ในการคำนวณเป็นดังนี้

เนื่องจากการเข้ารหัสเออีเอสแบบ 128 บิตหนึ่งครั้งจะได้งาน 128 บิต ดังนั้น ถ้าสามารถหาเวลาที่ใช้ในการเข้ารหัสเออีเอสแบบ 128 บิตหนึ่งครั้งได้ จะได้ว่า

$$Throughput(bps) = \frac{128}{Time(s)}$$

ซึ่งเราจะสามารถหาเวลาที่ใช้ในการเข้ารหัสเออีเอสแบบ 128 บิตหนึ่งครั้งได้จากจำนวนรอบของสัญญาณนาฬิกาที่ใช้ในการเข้ารหัสเออีเอสแบบ 128 บิตหนึ่งครั้งหารด้วยความถี่ที่ใช้

$$Time(s) = \frac{Cycle}{frequency(Hz)}$$

จะได้ว่า

$$Throughput(bps) = 128 \times \frac{frequency(Hz)}{Cycle}$$

โดยความถี่นั้นสามารถวัดได้จากการทดลอง ส่วนจำนวนรอบที่ใช้นั้นจะได้มาจากการวิเคราะห์การทำงานของหน่วยควบคุมทั้งสอง

3.4.1. ขนาดของวงจร

ขนาดของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตเป็นดังแสดงในตารางที่ 3.5 ซึ่งจะเห็นได้ว่าใช้ทรัพยากรบนเอฟพีจีเอน้อยมากคือ 349 สไลซ์ ซึ่งถ้าวงจรเข้ารหัสนี้ไม่สามารถเปลี่ยนโครงแบบได้ จะต้องใช้สไลซ์จำนวน 455 สไลซ์ นั่นหมายความว่าวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้สามารถลดจำนวนทรัพยากรที่ใช้ไปได้ถึง 23% แต่ก็ต้องใช้หน่วยความจำภายนอกเพิ่มขึ้นถึง 207,072 บิต บนเอฟพีจีเอรุ่น XC3S200-4FT256 ซึ่งถ้าเปลี่ยนเป็น XC3S50-4FT256 จะใช้หน่วยความจำเพียง 145,536 บิต หรือ 48,512 บิตต่อสไลซ์ 4 หลัก

ตารางที่ 3.5 ตารางแสดงขนาดของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต

จำนวนสไลซ์ทั้งหมด	349 สไลซ์
วงจรส่วนที่เปลี่ยนโครงแบบไม่ได้	252 สไลซ์
วงจรย่อยเอสบ็อกซ์	97 สไลซ์
วงจรย่อยผสมหลัก	73 สไลซ์
วงจรย่อยเอ็กซ์ออร์	33 สไลซ์
จำนวน BlockRAM ที่ใช้	0 บิต
จำนวนหน่วยความจำภายนอกที่ใช้	207,072 บิต

3.4.2. ความเร็วในการทำงาน

ความเร็วในการทำงานของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตในงานวิจัยนี้เป็นดังแสดงในตารางที่ 3.6 ซึ่งเป็นการสร้างวงจรนี้ลงบนเอฟพีจีเอรุ่น XC3S200-4FT256

ตารางที่ 3.6 ตารางแสดงความเร็วของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัต

จำนวนรอบสัญญาณนาฬิกา (รอบ)	**ความถี่สูงสุด (เมกะเฮิรตซ์)**	**ปริมาณงาน (บิตต่อวินาที)**
252,911	50	25,305

จากตารางที่ 3.6 จะเห็นว่าจำนวนรอบสัญญาณนาฬิกาที่ใช้ในการเข้ารหัสนั้นมีจำนวนสูงมากซึ่งเมื่อวิเคราะห์จากการทำงานของวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้จะพบว่า จำนวนรอบสัญญาณนาฬิกาที่ต้องใช้ในการทำงานเป็นดังตารางที่ 3.7 ซึ่งจะเห็นว่าขั้นตอนที่ใช้จำนวนรอบสัญญาณนาฬิกามากที่สุดคือการเปลี่ยนโครงแบบ โดยขั้นตอนนี้ใช้ไปถึง 99.70% (252,146 รอบ) ของจำนวนรอบสัญญาณนาฬิกาทั้งหมดที่ใช้ ซึ่งถ้าไม่คิดเวลาที่ใช้ในการเปลี่ยนโครงแบบจะได้ปริมาณงานเพิ่มขึ้นเป็น 8.366 เมกะบิตต่อวินาที

เนื่องจากจำนวนรอบสัญญาณนาฬิกาที่ใช้ในการเปลี่ยนโครงแบบนั้นขึ้นกับจำนวนบิตของข้อมูลโครงแบบที่ต้องใช้ในการเปลี่ยนโครงแบบ ซึ่งเมื่อเปลี่ยนเอฟพีจีเอที่ใช้เป็น XC3S50-4FT256 ขนาดของข้อมูลโครงแบบจะลดลง ซึ่งจะทำให้ใช้จำนวนรอบสัญญาณนาฬิกาเพียง 6,144 รอบในการเปลี่ยนโครงแบบหนึ่งครั้ง หรือ 178,176 รอบต่อการเข้ารหัสเออีเอสหนึ่งครั้ง ซึ่งจะทำให้ได้ปริมาณงานเพิ่มขึ้นเป็น 35.766 กิโลบิตต่อวินาที

และจากการทดลองสร้างวงจรเข้ารหัสเออีเอสที่ไม่สามารถเปลี่ยนโครงแบบได้อย่างพลวัตยังพบว่าวงจรนี้จะสามารถทำงานได้ที่ความถี่สูงสุด 100 เมกะเฮิรตซ์ ถ้าไม่มีการเปลี่ยนโครงแบบเนื่องจากการเปลี่ยนโครงแบบของเอฟพีจีเอรุ่น Spartan-3 นั้นสามารถทำได้ที่ความถี่สูงสุด 50 เมกะเฮิรตซ์ นั่นหมายความว่าถ้าไม่มีการเปลี่ยนโครงแบบ วงจรเข้ารหัสเออีเอสนี้ได้ปริมาณงานเพิ่มขึ้นเป็น 16.732 เมกะบิตต่อวินาที

ตารางที่ 3.7 ตารางแสดงจำนวนรอบสัญญาณนาฬิกาที่ใช้ในการทำงานแต่ละขั้นตอน

ขั้นตอน	จำนวนรอบสัญญาณนาฬิกา (รอบ)
การติดต่อกับวงจรภายนอก	48
การอ่านข้อมูลสถานะ	16 x 1 = 16
การอ่านข้อมูลคีย์	16 x 1 = 16
การเขียนข้อมูลสถานะ	16 x 1 = 16
การเข้ารหัส	717
กระบวนการแทนที่ไบต์	16 x 10 = 160
กระบวนการผสมหลัก	19 x 9 = 171
กระบวนการบวกคีย์แต่ละรอบ	16 x 11 = 176
กระบวนการแทนที่คำ	4 x 10 = 40
กระบวนการบวกค่าคงที่ของแต่ละรอบ	1 x 10 = 10
กระบวนการบวกคีย์	16 x 10 = 160
การเปลี่ยนโครงแบบ	252,146
การเปลี่ยนโครงแบบเป็นวงจรย่อยเอสบ็อกซ์	8,704 x 10 = 87,040
การเปลี่ยนโครงแบบเป็นวงจรย่อยผสมหลัก	8,704 x 9 = 78,336
การเปลี่ยนโครงแบบเป็นวงจรย่อยเอ็กซ์ออร์	8,704 x 10 = 87,040
รวม	**252,911**

บทที่ 4

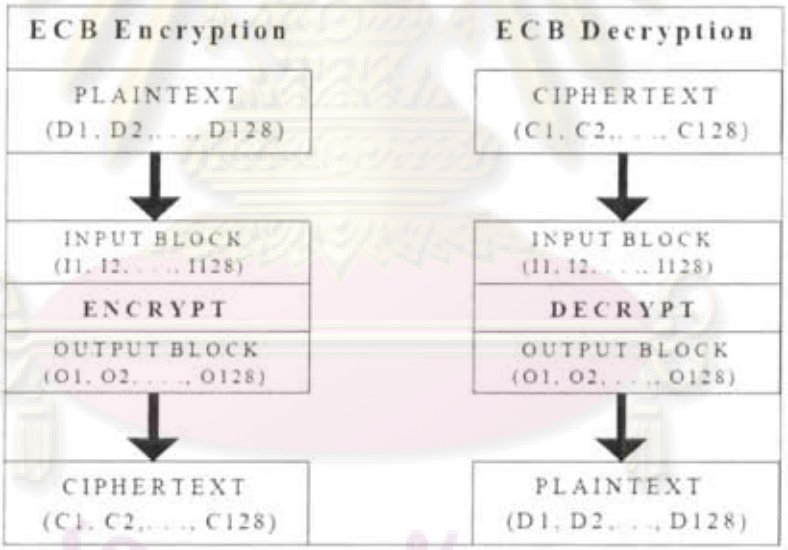
การตรวจสอบความถูกต้อง

การตรวจสอบความถูกต้องของวงจรเข้ารหัสทำโดยการใช้เวกเตอร์ทดสอบ (Test Vector) เวกเตอร์ทดสอบนี้เป็นเวกเตอร์ทดสอบชุดเดียวกันกับที่เอ็นไอเอสที (NIST: National Institute Standards and Technology) [19] ได้กำหนดไว้เพื่อใช้ทดสอบระเบียบวิธีการเข้ารหัสที่จะมาเป็นเออีเอส การทดสอบมีทั้งหมดแบ่งเป็นสองลักษณะประกอบด้วย

1. การทดสอบแบบรู้คำตอบ (KAT: Known Answer Tests) คือชุดของเวกเตอร์ทดสอบเพื่อประสิทธิภาพในแต่ละด้านของการเข้ารหัสใช้ทดสอบทีละ 128 เวกเตอร์
2. การทดสอบแบบมอนติคาร์โล (MCT: Monte Carlo Tests) คือเวกเตอร์ทดสอบที่ใช้ทดสอบทีละ 400 เวกเตอร์แต่ละเวกเตอร์ต้องเข้ารหัสย่อยๆต่อเนื่องกันอีกเป็นจำนวน 10,000 รอบ

นอกจากนี้การทดสอบในแต่ละแบบจะแยกย่อยออกเป็นอีกสองโหมดคือ

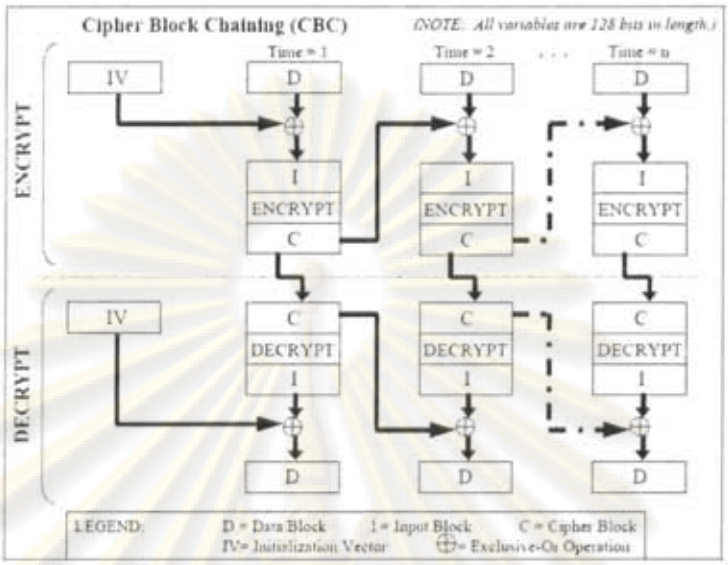
1. โหมดอีซีบี (ECB: Electronic Codebook) เป็นโหมดที่ใส่ข้อความเข้าไปโดยตรงดังในรูปที่ 4.1



รูปที่ 4.1 แผนภาพแสดงการทดสอบโหมดอีซีบี

จากรูปที่ 4.1 การเข้ารหัสจะนำอินพุต (I1, I2, ..., I128) จากข้อความทั้งหมด (Plain Text) 128 บิต (D1, D2, ..., D128) เข้ามาประมวลผลโดยตรงแล้วส่งออกไปเป็นเอาท์พุต (O1, O2, ..., O128) ซึ่งจะถ้าเป็นการเข้ารหัสผลที่ได้คือข้อความที่เข้ารหัส (C1, C2, ..., C128) เรียบร้อยแล้ว ส่วนการถอดรหัสนำอินพุต (I1, I2, ..., I128) จากข้อความที่เข้ารหัสแล้ว (C1, C2, ..., C128) เข้ามาประมวลผลได้เป็นเอาต์พุต (O1, O2, ..., O128) แล้วส่งออกเป็นข้อความที่ถูกถอดรหัส (D1, D2, ..., D128)

2. โหมดซีบีซี (CBC: Cipher Block Chaining) คือโหมดที่นำเอาอินพุตที่เป็นข้อความธรรมดามาทำการเอ็กซ์ออร์กับเวกเตอร์ค่าเริ่มต้น (Initialization Vector) ก่อนเข้ารหัสดังแสดงในรูปที่ 4.2



รูปที่ 4.2 แผนภาพแสดงการทดสอบโหมดซีบีซี

จากรูปที่ 4.2 ในกระบวนการเข้ารหัสข้อมูล (D) ที่เวลาที่หนึ่งทำการเอ็กซ์ออร์กับเวกเตอร์ค่าเริ่มต้นก่อนที่ไปเป็นอินพุต (I) เพื่อทำการเข้ารหัสแล้วนำข้อความที่เข้ารหัส (C) ไปใช้เป็นตัวกระทำเอ็กซ์ออร์ในรอบถัดไปและเป็นเช่นเดียวกันกับกระบวนการถอดรหัส

กระบวนการทดสอบที่ได้กล่าวมาต้องอาศัยชุดของเวกเตอร์ทดสอบซึ่งถูกจัดเก็บลงเป็นไฟล์ที่มีชื่อต่างกันเพื่อความสะดวกในเรียกการใช้งานและแต่ละไฟล์จะใช้งานทดสอบในกรณีที่ต่างกันตามตารางที่ 4.1 และ

ตารางที่ 4.2

ตารางที่ 4.1 ตารางแสดงการใช้ไฟล์ชุดเวกเตอร์ทดสอบแบบรู้คำตอบ

Filename	Mode	Test	Key Sizes (bits)
ecb_vk.txt	ECB	Variable Key KAT	128, 192, 256
ecb_vt.txt	ECB	Variable Text KAT	128, 192, 256
ecb_tbl.txt (if applicable)	ECB	Table KAT	128, 192, 256
? (possibly multiple files) (if applicable)	ECB	Intermediate Values KAT	128, 192, 256

ตารางที่ 4.2 ตารางแสดงการใช้ไฟล์ชุดเวกเตอร์ทดสอบแบบมอนติคาร์โล

Filename	Mode	Test	Key Sizes (bits)
ecb_e_m.txt	ECB	Encrypt MCT	128, 192, 256
ecb_d_m.txt	ECB	Decrypt MCT	128, 192, 256
cbc_e_m.txt	CBC	Encrypt MCT	128, 192, 256
cbc_d_m.txt	CBC	Decrypt MCT	128, 192, 256

จากตารางที่ 4.1 จะเห็นได้ว่าเป็นการทดสอบในโหมดอีซีบีเพียงอย่างเดียวเท่านั้น

Variable Key KAT คือการทดสอบที่ใช้ชุดของเวกเตอร์ทดสอบที่มีข้อความเดียวกันทุกเวกเตอร์ทดสอบมีเพียงคีย์เท่านั้นที่แต่ละเวกเตอร์ทดสอบมีค่าแตกต่างกัน

Variable Text KAT คือการทดสอบที่ใช้ชุดของเวกเตอร์ทดสอบที่มีคีย์เดียวกันทุกเวกเตอร์ทดสอบมีเพียงข้อความเท่านั้นที่แต่ละเวกเตอร์ทดสอบมีค่าแตกต่างกัน

Table KAT คือชุดของเวกเตอร์ทดสอบที่ใช้เพื่อทดสอบตารางเอสบ็อกซ์ (S-Box) เวกเตอร์ทดสอบภายในจะถูกบังคับให้ใช้งานเอสบ็อกซ์ทุกๆค่าที่เป็นไปได้

Intermediate Value KAT คือการทดสอบค่าของคำตอบชั่วคราวที่เกิดขึ้นระหว่างการเข้ารหัส

ดังนั้นแต่ละไฟล์ใช้เพื่อทำการทดสอบดังนี้

ecb_vk.txt – เป็นไฟล์ที่เก็บชุดของเวกเตอร์ทดสอบในโหมดอีซีบีเพื่อทดสอบการเข้ารหัสแบบข้อความคงที่

ecb_vt.txt – เป็นไฟล์ที่เก็บชุดของเวกเตอร์ทดสอบในโหมดอีซีบีเพื่อทดสอบการเข้ารหัสแบบคีย์คงที่

ecb_tbl.txt – เป็นไฟล์ที่เก็บชุดของเวกเตอร์ทดสอบในโหมดอีซีบีเพื่อทดสอบตารางแทนค่าที่ใช้เข้ารหัส (ชุดของเวกเตอร์ทดสอบนี้มีไว้สำหรับการเข้ารหัสที่มีการใช้ตาราง)

ส่วนไฟล์สุดท้ายที่ไม่มีชื่อมีไว้สำหรับทดสอบคำตอบระหว่างการเข้ารหัสซึ่งผู้ทดสอบเป็นคนตั้งขึ้นมาเองซึ่งเอ็นไอเอสทีไม่ได้กำหนดมาตรฐานไว้

จาก

ตารางที่ 4.2 จะเห็นได้ว่าการทดสอบแบบมอนติคาร์โลแยกเป็นสองประเภทคือโหมดอีซีบีและโหมดซีบีซี สองโหมดนี้ยังแยกย่อยระหว่างการเข้ารหัสและถอดรหัสด้วย ภายในไฟล์ประกอบด้วยเวกเตอร์ทดสอบจำนวน 400 เวกเตอร์ เมื่อนำมาใช้ทดสอบแบบมอนติคาร์โลแต่ละเวกเตอร์จะถูกเข้ารหัสอย่างต่อเนื่องทั้งหมด 10,000 ครั้ง การทดสอบแบบมอนติคาร์โลถูกแบ่งแยกอย่างชัดเจนว่าเข้ารหัสหรือถอดรหัสดังนั้นไฟล์ที่นำมาทดสอบจึงมีสองไฟล์ด้วยกันคือ

ecb_e_m.txt และ cbc_e_m.txt ไฟล์แรกเพื่อทดสอบการเข้ารหัสด้วยมอนติคาร์โลโหมดอีซีบี ส่วนอีกอันเพื่อทดสอบการเข้ารหัสด้วยมอนติคาร์โลโหมดซีบีซี

4.1. รายละเอียดการตรวจสอบ

เริ่มแรกทดสอบวงจรเข้ารหัสแบบเปลี่ยนโครงแบบได้นี้โดยวิธีรู้คำตอบแบบอีซีบีด้วยไฟล์ที่มีชื่อดังต่อไปนี้

ecb_vk.txt

ecb_vt.txt

ecb_tbl.txt

ecb_iv.txt (เป็นไฟล์ที่รินเดลล์แนบเพิ่มมาด้วยเพื่อทดสอบคำตอบชั่วคราว)

การทดสอบด้วยไฟล์ทั้งสี่นี้เป็นไปในลักษณะที่ใส่อินพุตเข้าไปแล้วรอรับคำตอบแบบตัวต่อตัวจึงไม่ต้องดัดแปลงวงจรให้เหมาะกับการทดสอบแต่อย่างใด

ต่อจากนั้นทำการทดสอบแบบมอนติคาร์โลโดยไฟล์

ecb_e_m.txt

cbc_e_m.txt

ดังที่ได้กล่าวมาแล้วข้างต้นเกี่ยวกับการทดสอบแบบมอนติคาร์โลซึ่งมีทั้งในแบบอีซีบีและซีบีซีมีรายละเอียดดังรูปที่ 4.3 และรูปที่ 4.4

```
Initialize KEY0, PT0

FOR I = 0 TO 399
{
      Record I, KEYi, PT0
      FOR j = 0 TO 9,999
      {
            IBj = PTj
            Perform algorithm in encrypt state, resulting in CTj
            PTj+1 = CTj
      }
      Record CTj

      KEYi+1 = KEYi ⊕ last n bits of CT, where n = 128, 192, or
256 (depending on key size)
      PT0 = CT9999
}
```

รูปที่ 4.3 การทดสอบมอนติคาร์โลแบบอีซีบี

จากรูปที่ 4.3 การทดสอบจะเริ่มต้นจากรับค่าคีย์ตั้งต้น KEY_0 และข้อความตั้งต้น PT_0 เข้ามาต่อจากนั้นจะเข้าวงวนอันนอก เก็บค่าคีย์และข้อความจากนั้นเข้าสู่วงวนอันในแล้วส่งข้อความเข้าไปเป็นพารามิเตอร์สำหรับเข้ารหัส ($IB_j = PT_j$) เมื่อเข้ารหัสเสร็จแล้วก็นำข้อความเข้ารหัสที่ได้ไปใช้เป็นอินพุตในครั้งต่อไป ($PT_{j+1} = CT_j$) วงวนด้านในก็จะวนไปเรื่อยๆ โดยใช้ข้อความที่ถูกเข้ารหัสใน

ครั้งก่อนมาเป็นอินพุต ส่วนคีย์จะใช้คีย์เดียวกันตลอดจนจบวงวนด้านใน เมื่อวงวนด้านในเสร็จสิ้นหนึ่งรอบจะมีการเปลี่ยนคีย์โดยนำเอาคีย์เดิมมาเอ็กซ์ออร์กับข้อความสุดท้ายที่เข้ารหัสเสร็จ ($KEY_{i+1} = KEY_i$ xor CT) หลังจากนั้นส่งต่อข้อความสุดท้ายที่ได้จากวงวนด้านในมาเป็นอินพุตของวงวนด้านในรอบต่อไป ($PT_0 = CT_{9999}$) ทำเช่นนี้ไปเรื่อยๆจนหมดวงวนด้านนอกก็ถือเป็นการทดสอบเสร็จสิ้น

อย่างไรก็ตามเนื่องจากการทดสอบดังกล่าวทำเป็นวงวนไปเรื่อยๆทั้งหมด 4,000,000 ครั้ง ซึ่งไม่เหมาะกับวงจรที่ได้ออกแบบมาตอนต้นจึงจำเป็นต้องดัดแปลงวงจรเพื่อให้ทดสอบได้ การดัดแปลงดังกล่าวไม่ได้ส่งผลต่อการทำงานโดยรวมของวงจรแต่เป็นการปรับแต่งอินพุตที่เข้ามาให้มีลักษณะเหมือนอย่างที่การทดสอบต้องการ วงจรจะถูกดัดแปลงให้สามารถเข้ารหัสอย่างต่อเนื่องได้โดยใช้คีย์ตัวเดิมนั่นคือสามารถทำงานของวงวนด้านในได้ แต่งานของวงวนด้านนอก (เช่น การหาคีย์ต่อไปให้วงวนด้านใน) จำเป็นต้องใช้มือเข้ามาช่วยทำเนื่องจากการดัดแปลงมากไปจะส่งผลให้เกิดความซ้ำซ้อนของงานขึ้น

```
Initialize KEY0, IV, PT0

FOR I = 0 TO 399
{
      If (i==0) CV0 = IV
      Record i, KEYi, CV0, PT0
      FOR j = 0 TO 9,999
      {
            IBj = PTj ⊕ CVj
            Perform algorithm in encrypt state, resulting in CTj
            IF (j==0)
                  PTj+1 = CV0
            ELSE
                  PTj+1 = CTj-1
            CVj+1 = CTj
      }
      Record CTj

      KEYi+1 = KEYi ⊕ last n bits of CT, where n = 128, 192, or
256 (depending on key size)
      PT0 = CT9998
      CV0 = CT9999
}
```

รูปที่ 4.4 การทดสอบมอนติคาร์โลแบบซีบีซี

การทดสอบในแบบซีบีซีจะมีหลักการคล้ายๆกับอีซีบีแตกต่างกันเพียงแค่การส่งอินพุตไปยังวงจรเข้ารหัสต้องมีการนำข้อความไปเอ็กซ์ออร์กันข้อความที่ถูกเข้ารหัสในรอบที่แล้ว ($IB_j = PT_j$ xor CV_j) ส่วนของโค้ดที่เพิ่มมาเป็นเพียงรายละเอียดปลีกย่อยที่ในการหาข้อความที่ถูกเข้ารหัสในรอบก่อนจึงไม่ขอกล่าวถึง

การทดสอบวิธีนี้จำเป็นต้องดัดแปลงวงจรที่ซับซ้อนกว่าเดิมเนื่องจากมีการเปลี่ยนแปลงอินพุตระหว่างวงวนรอบเล็กและการเปลี่ยนแปลงดังกล่าวต้องมีความเกี่ยวข้องกับข้อความในครั้งที่ผ่านมาด้วย ดังนั้นการทดสอบในแบบซีบีซีต้องเพิ่มส่วนของวงจรเข้าไปเพื่อตั้งค่าตั้งต้นใหม่ในทุกๆรอบผลการทดสอบแบบมอนติคาร์โลของทั้งสองกรณีแสดงไว้ในตารางที่ 4.3

4.2. ผลการตรวจสอบ

การทดสอบทั้งสี่เป็นไปอย่างถูกต้อง สามารถสรุปได้เป็นตารางที่ 4.3

ตารางที่ 4.3 ตารางแสดงความถูกต้องในการทดสอบการเข้ารหัส

ไฟล์ที่ใช้ทดสอบ	ความถูกต้อง(ร้อยละ)
ecb_vk.txt	100
ecb_vt.txt	100
ecb_tbl.txt	100
ecb_iv.txt	100
ecb_e_m.txt*	100
cbc_e_m.txt*	100

*เป็นการสุ่มทดสอบ เนื่องจากมีเวกเตอร์ทดสอบเป็นจำนวนมาก

จากตารางที่ 4.3 ทำให้สามารถสรุปได้ว่าวงจรเข้ารหัสแบบเปลี่ยนโครงแบบได้ทำงานได้อย่างถูกต้องตรงตามความต้องการของการเข้ารหัสแบบเออีเอสทุกประการ

บทที่ 5
สรุปผลการวิจัย และข้อเสนอแนะ

5.1. สรุปผลการวิจัย

วิธีการออกแบบวงจรเข้ารหัสเออีเอสที่เปลี่ยนโครงแบบได้อย่างพลวัตเป็นอีกแนวทางหนึ่งในที่สามารถนำไปประยุกต์ใช้กับการออกแบบวงจรอื่นๆที่มีขนาดใหญ่และต้องการใช้ทรัพยากรอย่างจำกัด อาทิเช่นระบบฝังตัวต่างๆ เพราะวงจรที่ได้จะมีขนาดเล็กและใช้ทรัพยากรน้อย เนื่องจากความสามารถในการเปลี่ยนโครงแบบ ทำให้สามารถเปลี่ยนโครงแบบบางส่วนของวงจรให้เป็นวงจรย่อยต่างๆ เพื่อทำงานย่อยๆ ในเวลาที่จำเป็นเท่านั้น ดังนั้นขนาดวงจรรวมในเวลาหนึ่งจึงมีขนาดไม่เกินขนาดของวงจรรวมทั้งหมด หรือเท่ากับวงจรส่วนที่ไม่สามารถเปลี่ยนโครงแบบได้รวมกับวงจรย่อยที่มีขนาดใหญ่ที่สุดในระบบ

แต่อย่างไรก็ดีการทำให้วงจรมีความสามารถในการเปลี่ยนโครงแบบได้อย่างพลวัต จะต้องมีส่วนควบคุมสำหรับควบคุมการเปลี่ยนโครงแบบ และหน่วยความจำจำนวนหนึ่งสำหรับเก็บข้อมูลโครงแบบ ซึ่งเมื่อนำทั้งสองส่วนนี้มารวมกับวงจรด้วยอาจจะทำให้วงจรที่ได้มีขนาดใหญ่กว่าวงจรที่ไม่สามารถเปลี่ยนโครงแบบได้ ดังนั้นวงจรที่มีขนาดใหญ่จึงเหมาะสมสำหรับการเปลี่ยนโครงแบบอย่างพลวัตมากกว่าวงจรที่มีขนาดเล็ก

ถึงแม้ว่าในงานวิจัยนี้จะ ไม่สนใจด้านความเร็วในการประมวลผล แต่ก็จะเห็นได้ว่าความเร็วที่ได้นั้นจัดอยู่ในเกณฑ์ที่น่าพอใจสำหรับวงจรที่ต้องเปลี่ยนโครงแบบอย่างต่อเนื่องในการทำงานดังจะเห็นได้ว่าเวลาที่ใช้มากที่สุดในการประมวลผล คือเวลาที่ใช้ในการเปลี่ยนโครงแบบ (99.70%) ดังนั้นถ้าเราใช้เอฟพีจีเอที่สามารถเปลี่ยนโครงแบบได้เร็วกว่านี้ เราก็สามารถสร้างวงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัตที่ทำงานได้รวดเร็วมากขึ้น

5.2. ข้อเสนอแนะ

เนื่องด้วยเทคโนโลยีเอฟพีจีเอในปัจจุบันยังไม่สนับสนุนการออกแบบวงจรที่ต้องเปลี่ยนโครงแบบระหว่างการทำงานอย่างต่อเนื่องเท่าที่ควร ทำให้ความเร็วในการประมวลผลของวงจรทำได้ไม่รวดเร็ว เนื่องจากต้องเสียเวลาในการเปลี่ยนโครงแบบมาก เพราะต้องส่งข้อมูลโครงแบบเป็นจำนวนมาก นอกจากนี้ยังจำเป็นต้องใช้หน่วยความจำสำหรับเก็บข้อมูลโครงแบบอีกเป็นจำนวนมาก ซึ่งถือเป็นข้อเสียของการประยุกต์นำแนวคิดของการเปลี่ยนโครงแบบได้อย่างพลวัตมาใช้

เพื่อให้การประยุกต์การเปลี่ยนโครงแบบอย่างพลวัตสามารถลดขนาดของวงจรได้อย่างมีประสิทธิภาพ วงจรส่วนที่จะเปลี่ยนโครงแบบควรจะมีขนาดเล็ก เพื่อให้ข้อมูลโครงแบบที่ใช้ในการเปลี่ยนโครงแบบมีขนาดเล็กซึ่งจะทำให้การเปลี่ยนโครงแบบทำได้รวดเร็วยิ่งขึ้น และยังช่วยลดขนาดของหน่วยความจำที่ใช้ในการเก็บข้อมูลโครงแบบด้วย ถึงแม้ว่าการเปลี่ยนโครงแบบจะทำให้วงจรมีขนาดเล็กลง แต่การเปลี่ยนโครงแบบแต่ละครั้งนั้นใช้เวลานาน ดังนั้นการเปลี่ยนโครงแบบ

ควรจะทำเมื่อมีความจำเป็นเท่านั้น และควรมีการเปลี่ยนโครงแบบให้น้อยที่สุดเพื่อลดเวลาที่ใช้ในการเปลี่ยนโครงแบบ นั่นหมายความว่าวงจรที่จะเปลี่ยนโครงแบบควรจะเป็นวงจรที่ต้องใช้เวลาในการทำงานนานหรือใช้งานอย่างต่อเนื่อง เพื่อให้คุ้มกับเวลาที่เสียไปกับการเปลี่ยนโครงแบบ

เนื่องจากวงจรสามารถทำงานได้ในขณะที่มีการเปลี่ยนโครงแบบ ดังนั้นการออกแบบวงจรให้ทำงานอย่างอื่นในขณะที่ทำการเปลี่ยนโครงแบบก็เป็นอีกวิธีที่จะช่วยให้ไม่เสียเวลาที่ใช้ในการเปลี่ยนโครงแบบไปอย่างเปล่าประโยชน์

เนื่องจากการประยุกต์แนวคิดในการเปลี่ยนโครงแบบอย่างพลวัตเพื่อลดขนาดของวงจรเป็นแนวคิดที่มีทั้งข้อดีและข้อเสีย ผู้ออกแบบจึงควรพิจารณาวิธีการลดขนาดวงจรด้วยวิธีที่ไม่ทำให้สมรรถนะลดลงก่อนจะนำแนวคิดการเปลี่ยนโครงแบบอย่างพลวัตไปใช้

อย่างไรก็ตาม ผู้วิจัยคาดว่าในอนาคตอันใกล้นี้จะมีการพัฒนาเอฟพีจีเอที่สามารถเปลี่ยนโครงแบบได้อย่างรวดเร็วออกมา ซึ่งจะทำให้วงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัตมีประสิทธิภาพสูงขึ้น

ในปัจจุบันเครื่องมือจำลองการทำงานที่มีอยู่ไม่สามารถใช้จำลองการทำงานของวงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัต ทำให้การพัฒนาวงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัตเป็นไปอย่างยากลำบาก และต้องอาศัยเครื่องมือภายนอกในการจับสัญญาณออกมาจากอุปกรณ์จริงเพื่อทำการตรวจสอบการทำงาน การพัฒนาเครื่องมือจำลองการทำงานจะช่วยทำให้การพัฒนาวงจรที่สามารถเปลี่ยนโครงแบบได้อย่างพลวัตทำได้สะดวกขึ้นอย่างมาก

ในการสร้างวงจรที่สามารถปรับตัวได้อย่างพลวัตนั้น จะต้องอาศัยความสามารถในการเปลี่ยนแปลงโครงแบบของเอฟพีจีเอ ซึ่งเอฟพีจีเอในปัจจุบันนั้นมีความสามารถที่จะเปลี่ยนแปลงโครงแบบได้อย่างพลวัต โดยเราสามารถที่จะโปรแกรมเอฟพีจีเอให้ทำงานตามที่เราต้องการได้โดยอาศัยกระแสข้อมูลโครงแบบเป็นตัวกำหนดโครงแบบของเอฟพีจีเอ [20] และเมื่อพิจารณาโครงสร้างของกระแสข้อมูลโครงแบบแล้ว เราจะพบว่าส่วนที่กำหนดการทำงานของเอฟพีจีเอนั้นจะขึ้นอยู่กับข้อมูลที่ถูกเขียนลงไปใน FAR (Frame Address Register) และ FDRI (Frame Data Input Register) ซึ่งถ้าเราสามารถเปลี่ยนแปลงข้อมูลในส่วนนี้ให้เหมาะสม วงจรก็จะสามารถปรับเปลี่ยนตัวเองให้ทำงานให้เหมาะสมกับสภาพแวดล้อมได้

เนื่องจากงานวิจัยชิ้นนี้เป็นการเปิดแนวทางใหม่ในการออกแบบและสร้างวงจรดิจิทัล โดยนำความสามารถในการเปลี่ยนโครงแบบได้ของเอฟพีจีเอมาประยุกต์ ซึ่งผู้วิจัยพบว่ายังมีช่องทางในการพัฒนาต่อได้อีกหลายทาง โดยจะสรุปเป็นหัวข้อต่างๆได้ ดังนี้

1. พัฒนาวิธีการเปลี่ยนโครงแบบบางส่วนให้รวดเร็วขึ้น และลดขนาดหน่วยความจำโครงแบบที่ใช้ โดยการประยุกต์วิธีการเปลี่ยนโครงแบบบางส่วนโดยการใช้ความแตกต่างเป็นฐาน (Difference-based Partial Reconfiguration) ร่วมกับวิธีการเปลี่ยนโครงแบบบางส่วนโดยการใช้มอดูลเป็นฐาน (Module-based Partial Reconfiguration) ที่ใช้ในงานวิจัยนี้ ซึ่ง

น่าจะช่วยลดขนาดของข้อมูลโครงแบบที่ต้องใช้ ทำให้ใช้เวลาในการเปลี่ยนโครงแบบและหน่วยความจำโครงแบบลดลง

2. วิเคราะห์หาจำนวนและวิธีการที่เหมาะสมในการแบ่งวงจรออกเป็นส่วนๆเพื่อให้การเปลี่ยนโครงแบบอย่างพลวัตทำได้อย่างมีประสิทธิภาพมากที่สุด
3. พัฒนาเครื่องมือสำหรับออกแบบและสร้างวงจรที่เปลี่ยนโครงแบบได้อย่างพลวัต รวมถึงเครื่องมือสำหรับตรวจสอบและจำลองวงจรที่เปลี่ยนโครงแบบได้อย่างพลวัต
4. นำวิธีการเปลี่ยนโครงแบบอย่างพลวัตไปประยุกต์ในการออกแบบและสร้างวงจรที่สามารถซ่อมแซมตัวเอง (Self-repairing Circuits) หรือวงจรที่ทนต่อความผิดพลาด (Fault-tolerant Circuits) ซึ่งรวมไปถึงวงจรที่สามารถพัฒนาตัวเองได้ (Evolvable Circuits)

รายการอ้างอิง

[1] Kao, C. Benefits of Partial Reconfiguration. Xcell Journal issue 55 (2005) : 65-67.

[2] Kalte, H., G. Lee, M. Porrmann, and U. Ruckert, REPLICA: A Bitstream Manipulation Filter for Module Relocation in Partial Reconfigurable Systems, 19th IEEE International Parallel and Distributed Processing Symposium (2005).

[3] Berthelot, F., F. Nouvel and D. Houzet, Partial and dynamic reconfiguration of FPGAs: a top down design methodology for an automatic implementation, 20th International Parallel and Distributed Processing Symposium (2006).

[4] Sedcole, P., B. Blodget, T. Becker, J. Anderson, and P. Lysaght, "Modular dynamic reconfiguration in Virtex FPGAs. IEE Proceedings Computers and Digital Techniques 153 (2006) : 157-164

[5] Heng, T. and F. D. Ronald, A Physical Resource Management Approach to Minimizing FPGA Partial Reconfiguration Overhead. IEEE International Conference on Reconfigurable Computing and FPGA's (2006) : 1-5.

[6] National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES), Federal Information Processing Standards (FIPS) Publication 197, 2001.

[7] Fu, Y., L. Hao, X. Zhang and R. Yang, Design of an extremely high performance counter mode AES reconfigurable processor, Second International Conference on Embedded Software and Systems (2005).

[8] Jian, X., L. Yuan-feng, D. Zi-bin and S. Yi. Design and Implementation of reconfigurable AES IP Core using FPGAs. The 6th International Conference On ASIC Proceeding (2005) : 765-765.

[9] Chaves, R., G. Kuzmanov, S. Vassiliadis and L. Sousa. Reconfigurable Memory Based AES Co-Processor. 20th International Parallel and Distributed Processing Symposium (2006).

[10] Perez, O., Y. Berviller, C. Tanougast, and S. Weber, Comparison of various strategies of implementation of the algorithm of encryption AES on FPGA, International Symposium on Industrial Electronics 2 (2006) : 3276-3280.

[11] Liu, T., C. Tanougast, and S. Weber, Toward a methodology for optimizing algorithm-architecture adequacy for implementation reconfigurable system, 13th IEEE International Conference on Electronics, Circuits and Systems (2006) : 1085-1088.

[12] Sripornprasert, J. and Chongstitvatana, P., The AES encryption circuit on a reconfigurable hardware, Electrical Engineering, Electronics, Computer, Telecommunications and Information Technology (ECTI) International Conference (2007) : 1139-1142.

[13] เจนโชติ ศรีพรประเสริฐ, วงจรเข้ารหัสเออีเอสชนิดเปลี่ยนโครงแบบได้, วิทยานิพนธ์ปริญญาวิศวกรรมศาสตรมหาบัณฑิต, จุฬาลงกรณ์มหาวิทยาลัย, 2549.

[14] Xilinx, Spartan-3 FPGA Family: Complete Data Sheet, DS099, 2007.

[15] Xilinx, Spartan-3 Generation FPGA User Guide, UG331, v1.2, 2007.

[16] Xilinx, Spartan-3 Generation Configuration User Guide, UG332, v1.2, 2007.

[17] Xilinx, Two Flows for Partial Reconfiguration: Module Based or Difference Based, XAPP290, v1.2, 2004.

[18] Lysaght, P., B. Brodget, J. Mason, J. Young, and B. Bridgford, Invited Paper: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAs, International Conference on Field Programmable Logic and Applications (2006).

[19] National Institute of Standards and Technology (NIST), http://csrc.nist.gov/CryptoToolkit/aes/. Computer Security Resource Center (CSRC), 1998.

[20] Xilinx, Spartan-3 Advanced Configuration Architecture, XAPP452, v1.0, 2004.

[21] Becker, J., M. Hubner, G. Hettich, R. Constapel, J. Eisenmann and J. Luka, Dynamic and Partial FPGA Exploitation, Proceedings of IEEE 95 (Feb. 2007) : 438-452.

[22] Daemon, J. and V. Rijnmen, The design of Rijndael: AES-The Advanced Encryption Standard, New York : Sringer-Verlag, 2002.

[23] National Institute of Standards and Technology (NIST), Data Encryption Standard (DES), Federal Information Processing Standards (FIPS) Publication 46-3, 1999.

[24] Chodowiec, P. and K. Gaj, Very Compact FPGA Implementation of the AES Algorithm, Cryptographic Hardware and Embedded Systems LNCS vol. 2779 (2003) : 319-333.

[25] Good, T. and M. Benaissa, AES on FPGA from the Fastest to the Smallest, Cryptographic Hardware and Embedded Systems LNCS 3659 (2005) : 427-440.

[26] Hodjat, B. and I. Verbauwhede, Area-Throughput Trade-offs for Fully Pipelined 30 to 70 Gbits/s AES Processors, IEEE Transactions on Computer 55 (Apr. 2006) : 366-372.

ประวัติผู้เขียนวิทยานิพนธ์

นายพีระ ต้นธีรวงศ์ เกิดเมื่อวันที่ 24 พฤศจิกายน พ.ศ. 2526 ที่จังหวัดกรุงเทพฯ สำเร็จการศึกษาระดับประถมศึกษาจากโรงเรียนสาธิต มศว ประสานมิตร สำเร็จการศึกษาระดับมัธยมศึกษาตอนต้นจากโรงเรียนสาธิต มศว ปทุมวัน สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนเตรียมอุดมศึกษา สำเร็จการศึกษาระดับปริญญาบัณฑิต ในสาขาวิชาวิศวกรรมคอมพิวเตอร์ จากคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยในปีการศึกษา 2548