การวิเคราะห์ประสิทธิภาพและกำลังงานใน S-box ด้วย GF สำหรับการเข้ารหัสแบบ AES

Performance and Power Analysis of S-box using GF for AES

วรรณจันทน์ วิทยาพันธ์ประชา

Wannajan Wittayapanpracha

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Computer Engineering
Prince of Songkla University
2554

ชื่อวิทยานิพนธ์ การวิเคราะห์ประสิทธิภาพและกำลังงานใน S-box ด้วย GF สำหรับการเข้ารหัสแบบ AES

ผู้เขียน นางสาววรรณจันทน์ วิทยาพันธ์ประชา

สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

.....

(ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต)

คณะกรรมการสอบ

.....ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์)

.....กรรมการ

(รองศาสตราจารย์ ดร.สมศักดิ์ มิตะถา)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

.....

(ศาสตราจารย์ ดร.อมรรัตน์ พงศ์ดารา)

คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์ การวิเคราะห์ประสิทธิภาพและกำลังงานใน S-box ด้วย GF สำหรับการเข้ารหัสแบบ AES

ผู้เขียน นางสาววรรณจันทน์ วิทยาพันธ์ประชา

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ปีการศึกษา 2553

บทคัดย่อ

งานวิจัยนี้นำเสนอวงจรคำนวณฟังก์ชัน S-box โดยใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ สำหรับการเข้ารหัสแบบ AES ที่มีการใช้พลังงานอย่างมีประสิทธิภาพ เนื่องจากการออกแบบวงจร S-box โดยใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ จะช่วยลดจำนวนการใช้ทรัพยากร และสามารถเพิ่มความเร็วในการทำงานให้สูงขึ้นได้ โดยที่งานวิจัยได้นำเสนอวิธีการลดรูปวงจรของโครงสร้างการทำงานแบบกาลัวร์ฟิลด์ $GF(2^4)^2$ จากการทดสอบจำลองการทำงานบนโครงสร้างเอฟพีจีเอพบว่า ผลของวงจร S-box $GF(2^4)^2$ ที่ได้จากเทคนิคการลดรูปวงจร มีการใช้กำลังงานน้อยกว่าวงจร S-box $GF(2^4)^2$ แบบดั้งเดิมประมาณ 19 %

นอกจากนี้ยังได้ทำการสร้างระบบการเข้ารหัสแบบ AES ในรูปแบบไปป์ไลน์ และแบบขนานที่สร้างขึ้นด้วยวงจร S-box แบบกาลัวร์ฟิลด์ $GF(2^4)^2$ และแบบตารางการเทียบค่า เพื่อนำไปวิเคราะห์การใช้งาน จากผลของการจำลองการทำงานพบว่า ระบบการเข้ารหัสแบบ AES ด้วยสถาปัตยกรรมแบบขนานที่ใช้ S-box แบบกาลัวร์ฟิลด์ $GF(2^4)^2$ ให้ประสิทธิภาพในการคำนวณสูงถึง 1.184 Gbps และใช้กำลังงานเพียง 408 mW เมื่อนำไปวิเคราะห์เทียบกับความเร็วในการเข้ารหัสแบบ AES บนโมดูล CC2420 ในระบบเครือข่ายเซนเซอร์ไร้สายพบว่า ระบบการเข้ารหัสแบบ AES ที่นำเสนอสามารถเข้ารหัสได้เร็วกว่าประมาณ 550 เท่า

คำสำคัญ: การเข้ารหัสแบบ AES, S-box, กาลัวร์ฟิลด์ $GF(2^4)^2$, ตารางการเทียบค่า, เครือข่ายเซนเซอร์ไร้สาย

Thesis Title	Performance and Power Analysis of S-box using GF for AES
Author	Miss Wannajan Wittayapanpracha
Major Program	Computer Engineering
Academic Year	2010

ABSTRACT

S-box circuits using $GF(2^4)^2$ for energy efficient AES encryption has been proposed in this research work. According to the design technique using $GF(2^4)^2$, the number of resource usage will be decreased and is able to improve the speed. We propose the technique to reduce the number of logic gate when the S-box circuits using $GF(2^4)^2$ has been implemented. From the simulation results on FPGA technology, we found that our S-box dissipate the power less than the original $GF(2^4)^2$ S-box circuits about 19%.

In addition, the several types of AES such as pipelining and parallel architectures have been implemented using our $GF(2^4)^2$ S-box and LUT S-box in order to analyze the performance. We found that the parallel architecture of AES using our $GF(2^4)^2$ S-box can compute at 1.184 Gbps and consume the power only 408mW. Our AES can perform very well suited with wireless sensor network application because it can run faster than the AES in CC2420 module about 550 times.

Keywords: AES encryption, S-box, Galois Field $GF(2^4)^2$, LUT, Wireless Sensor Networks.

กิตติกรรมประกาศ

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้เสียสละเวลาในการให้คำปรึกษา พร้อมทั้งแนะนำแนวคิดในการทำวิทยานิพนธ์ รวมถึงการแนะนำวิธีการแก้ไขปัญหาที่เกี่ยวกับการวิทยานิพนธ์ ตลอดจนตรวจสอบและแก้ไขวิทยานิพนธ์ให้ดำเนินไปอย่างลุล่วงสมบูรณ์

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์ ที่ได้กรุณาเสียสละเวลาเป็นประธานกรรมการสอบวิทยานิพนธ์ รวมถึงให้คำแนะนำในการแก้ไขปัญหาและตรวจทานแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

ขอขอบพระคุณ รองศาสตราจารย์ ดร.สมศักดิ์ มิตะถา ที่กรุณาเสียสละเวลาเป็นกรรมการสอบวิทยานิพนธ์ อีกทั้งตรวจทานและแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์ยิ่งขึ้น

ขอขอบพระคุณ บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ ที่ให้การสนับสนุนทุนในการทำวิจัยและให้ความช่วยเหลือด้านการประสานงานต่าง ๆ

ขอขอบพระคุณ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ที่กรุณาให้ทุนศิษย์ก้นกุฏิแก่ข้าพเจ้า

ขอขอบพระคุณ คณาจารย์ บุคลากร นักศึกษาปริญญาเอก และนักศึกษาปริญญาโท ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกคนที่ได้ให้คำปรึกษา และเป็นกำลังใจในการทำงานเป็นอย่างดีเสมอมา

และท้ายสุดนี้ ข้าพเจ้าขอน้อมรำลึกถึงพระคุณของบิดามารดา และครอบครัวที่ส่งเสริมและสนับสนุนข้าพเจ้าในทุก ๆ เรื่องจนกระทั่งข้าพเจ้าสำเร็จการศึกษา

วรรณจันทน์ วิทยาพันธ์ประชา

สารบัญ

หน้า

รายการตาราง

หน้า

รายการภาพประกอบ

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

เครือข่ายเซนเซอร์ไร้สาย (Wireless Sensor Network) เป็นเครือข่ายที่ประกอบด้วยเซนเซอร์โหนด (Sensor Node) หลายๆ ตัวทำงานร่วมกันโดยใช้ตัวตรวจจับ ตรวจสอบการทำงานเพื่อเก็บข้อมูลหรือส่งผ่านข้อมูลจากโหนดตัวหนึ่งไปยังอีกตัวหนึ่ง แล้วนำข้อมูลที่ได้รับไปประมวลผลที่สถานีฐาน (Base Station) เซนเซอร์โหนดเป็นอุปกรณ์ที่ประกอบด้วยหน่วยประมวลผลซึ่งโดยทั่วไปใช้ไมโครคอนโทรลเลอร์ นอกจากนั้นยังมีตัวตรวจจับซึ่งเป็นอุปกรณ์ตรวจจับเฉพาะงานและมีอุปกรณ์ในการรับ-ส่งสัญญาณแบบไร้สายผ่านทางคลื่นวิทยุความถี่ 2.4 GHz อีกด้วย เซนเซอร์โหนดสามารถเชื่อมต่อกันเป็นเครือข่ายในการสื่อสารตามมาตรฐาน IEEE 802.15.4

การทำงานของเครือข่ายเซนเซอร์ไร้สายเป็นการรับส่งข้อมูลระหว่างโหนดในการประยุกต์ใช้สำหรับงานบางประเภทเช่น การใช้งานทางการแพทย์หรือการทหารจำเป็นที่จะต้องมีการเข้ารหัสข้อมูลเพื่อให้เกิดความปลอดภัยจากผู้ไม่หวังดี หรือทำให้ผู้ที่ไม่มีส่วนเกี่ยวข้องไม่สามารถนำข้อมูลไปใช้งานต่อได้ เนื่องจากความปลอดภัยของข้อมูลที่ใช้ในระหว่างการติดต่อสื่อสารในเครือข่ายเซนเซอร์ไร้สายก็เหมือนกับระบบเครือข่ายอื่นทั่วไปเพราะงานเหล่านี้ต้องการความเป็นส่วนตัว (Privacy) ความสมบูรณ์ของข้อมูล (Data Integrity) ความน่าเชื่อถือ (Authorization) และเป็นความลับ (Confidential) [1]

วิธีการที่ช่วยให้ข้อมูลมีความปลอดภัยคือ การเข้ารหัสข้อมูลก่อนที่จะมีการส่งออกไปทางคลื่นความถี่วิทยุ ดังนั้นการเข้ารหัสข้อมูลจึงถูกพิจารณาเพื่อนำมาใช้งานในเครือข่ายเซนเซอร์ไร้สาย แต่เนื่องด้วยข้อจำกัดของทรัพยากรบนเซนเซอร์โหนดเช่น มีแหล่งจ่ายพลังงานจากแบตเตอร์รี่จำกัด ขนาดของหน่วยความจำที่จำกัด รวมไปถึงประสิทธิภาพในการประมวลผลของไมโครคอนโทรลเลอร์ที่ไม่สูงมาก ซึ่งข้อจำกัดที่ได้กล่าวมาข้างต้นล้วนเป็นปัจจัยที่เกี่ยวเนื่องถึงการรักษาความปลอดภัยให้กับข้อมูล จากเดิมการเข้ารหัสข้อมูลใช้วิธีการเขียนโปรแกรมควบคุมบนไมโครคอนโทรลเลอร์ด้วยอัลกอริทึมการเข้ารหัสแบบต่างๆ ซึ่งทำให้หน่วยความจำของไมโครคอนโทรลเลอร์ถูกใช้ไปมาก ทำให้ประสิทธิภาพการใช้งานของไมโครคอนโทรลเลอร์ลดลง จึงได้มีวิธีการใหม่นั่นคือการสร้างวงจรเข้ารหัสเพื่อใช้งานแทนวิธีดังกล่าว ทั้งนี้เพื่อลดเวลาของการ

เข้ารหัส ประหยัดพลังงานและเพิ่มความปลอดภัยมากขึ้นเนื่องจากกุญแจของการเข้ารหัสจะถูกฝังอยู่บนวงจรเข้ารหัส

ดังนั้นวิทยานิพนธ์ฉบับนี้จึงนำเสนอการวิเคราะห์วงจรเข้ารหัสและถอดรหัส โดยเลือกใช้อัลกอริทึมแบบ AES ซึ่งเหมาะสมกับเครือข่ายเซนเซอร์ไร้สายตามข้อเสนอแนะในงานวิจัย [1], [2], [3] ซึ่งจะเน้นการวิเคราะห์ในเรื่องของการใช้กำลังงานและความเร็วในการเข้ารหัส ซึ่งเกี่ยวเนื่องถึงพลังงานที่ใช้ อีกทั้งจะได้นำเสนอการปรับปรุงประสิทธิภาพในการเข้ารหัสโดยให้มีการใช้พลังงานอย่างมีประสิทธิภาพ (Energy Efficiency) ด้วยการเปรียบเทียบเทคนิคของการสร้างวงจร S-box ซึ่งเป็นวงจรส่วนสำคัญของการเข้ารหัสแบบ AES โดยเปรียบเทียบใน 2 เทคนิคคือ การใช้เทคนิคกาลัวร์ฟิลด์ (Galois Field, GF) และการใช้ตารางการเทียบค่า (LUT) โดยจะกำหนดให้มีการทำงานใกล้เคียงกับความเร็วในการส่งข้อมูลที่ 250 kbps (อัตราการส่งข้อมูลตามมาตรฐานของเครือข่ายเซนเซอร์ไร้สาย) เพื่อเป็นแนวทางนำไปประยุกต์ใช้งานเครือข่ายเซนเซอร์ไร้สายได้จริง

1.2 ตรวจเอกสาร

ในที่นี้แบ่งหัวข้อของการตรวจสอบเอกสารที่เกี่ยวข้องออกเป็น 2 ส่วนคือ 1) การเข้าและถอดรหัสที่เหมาะสมสำหรับเครือข่ายเซนเซอร์ไร้สายบนมาตรฐาน IEEE 802.15.4 และ 2) เอกสารที่เกี่ยวข้องกับเทคนิคการเพิ่มประสิทธิภาพและลดการใช้กำลังงานของวงจรเข้ารหัสแบบ AES นำมาซึ่งการทำวิทยานิพนธ์ฉบับนี้

1.2.1 การเข้าและถอดรหัสบนเครือข่ายเซนเซอร์ไร้สายมาตรฐาน IEEE 802.15.4

การเลือกพัฒนาวงจรเข้ารหัสแบบ Advanced Encryption Standard (AES) เนื่องจาก AES ถูกนำมาใช้เป็นมาตรฐานเข้ารหัสในการสื่อสารบนเครือข่ายเซนเซอร์ไร้สาย IEEE 802.15.4 ที่รองรับการส่งข้อมูลด้วยอัตราความเร็ว 250 kbps โดยในบทความวิจัยของ Shammi Didla, Aaron Ault และ Saurabh Bagchi, 2009 [2] นำเสนอวิธีการลดรูปโปรแกรมเพื่อให้กระบวนการเข้ารหัสบนซอฟต์แวร์ประหยัดพลังงานและทำงานเร็วขึ้นกว่าเดิมและสอดคล้องกับการใช้งานจริงบนโหนด Platform ของ MSP430 และ Micaz

ในขณะที่บทความวิจัยของ M Healy, T Newe และ E Lewis [3] ได้กล่าวไว้ว่า วิธีที่ดีที่สุดสำหรับความปลอดภัยของข้อมูลคือ ใช้อัลกอริทึมเข้ารหัสข้อมูลก่อนที่จะส่งข้อมูลออกไป อย่างไรก็ตามก็ต้องคำนึงถึงข้อจำกัดของเซนเซอร์โหนดด้วย จากงานวิจัยนี้เห็นได้อย่างชัดเจนว่าการใช้วงจรการเข้ารหัสแบบ AES บน CC2420 ซึ่งเป็นไอซีสำหรับใช้ในการรับส่งข้อมูลระหว่างเซนเซอร์โหนดแบบไร้สายที่นิยมเช่นใน Platform ของ Micaz และ Tmote Sky โดยจะสามารถ

ทำงานได้เร็วและประหยัดทรัพยากรของหน่วยความจำได้ดีกว่าการเข้ารหัสด้วยซอฟต์แวร์ดังตารางที่ 1-1, 1-2, 1-3 และ 1-4

ตารางที่ 1-1 เวลาเริ่มต้นการทำงาน(μs) [3]

	MICAz	**Tmote SKY**
CC2420_encrypt	294.00	2060.34
Software_encrypt_ref	652.15	940.83
Software_encrypt	565.37	752.27
Software_encrypt_minRAM	574.85	934.83

ตารางที่ 1-2 เวลาการเข้ารหัสขนาด 16 ไบต์(μs) [3]

	MICAz	**Tmote SKY**
CC2420_encrypt	29.83	449.20
Software_encrypt_ref	1495.78	1941.08
Software_encrypt	1457.53	1870.64
Software_encrypt_minRAM	1547.72	1943.74

ตารางที่ 1-3 เวลารวมการทำงานทั้งหมด (μs)

	MICAz	**Tmote SKY**
CC2420_encrypt	323.834	2509.543
Software_encrypt_ref	2147.930	2881.910
Software_encrypt	2022.900	2622.910
Software_encrypt_minRAM	2122.570	2878.570

จากตารางที่ 1-3 เวลารวมที่ใช้ในการเข้ารหัสข้อมูลบนอุปกรณ์ฮาร์ดแวร์จะใช้เวลาน้อยกว่าการใช้ซอฟต์แวร์ หากพิจารณา Tmote SKY เพื่อนำไปประยุกต์ใช้งาน จะเห็นว่าการเข้ารหัสข้อมูลบนตัว Tmote SKY ซึ่งมองโดยภาพรวมแล้วเวลาการเข้ารหัสบนซอฟต์แวร์และฮาร์ดแวร์นั้นไม่ค่อยจะมีความต่างกันมากนักห่างกันในระดับ 100 μs ซึ่งไม่ค่อยน่าสนใจมากในการลดช่วงเวลาในระดับนี้ แต่หากพิจารณาในส่วนเวลาตั้งต้นแล้วการทำงานด้วยฮาร์ดแวร์ใช้เวลาน้อย

กว่าซอฟต์แวร์ประมาณ 2 เท่าตัว และใช้เวลาในการเข้ารหัสเร็วกว่าถึง 4 เท่าด้วย นอกจากนั้นการใช้ทรัพยากร (ในงานวิจัย [3] นี้พิจารณาถึงหน่วยความจำ) มีส่วนสำคัญในการพิจารณาเลือกใช้งานด้วยเพราะอุปกรณ์เครือข่ายเซนเซอร์ไร้สายมีขนาดเล็ก ดังนั้นจึงต้องเลือกใช้อัลกอริทึมให้เหมาะสมของข้อจำกัดในส่วนนี้ด้วย

หน่วยความจำเป็นปัจจัยหนึ่งที่สำคัญในการทำงานของเซนเซอร์โหนด จากตารางที่ 1-4 พบว่าอุปกรณ์ฮาร์ดแวร์ (CC2420_encrypt) นั้นมีการใช้ทรัพยากรน้อยกว่าแม้จะไม่ได้น้อยกว่ามากก็ตาม จากงานวิจัยดังกล่าวพบว่า การใช้อุปกรณ์ฮาร์ดแวร์สำหรับการเข้ารหัสสามารถช่วยให้ประสิทธิภาพทั้งในส่วนของการใช้พลังงานและความเร็วในการทำงานดีกว่าการเขียนโปรแกรมเข้ารหัสซึ่งสอดคล้องกับแนวทางงานวิจัยในวิทยานิพนธ์ฉบับนี้

ตารางที่ 1-4 หน่วยความจำแบบ ROM และ RAM ที่ใช้ (bytes) [3]

	MICAz		**Tmote SKY**	
	ROM	**RAM**	**ROM**	**RAM**
CC2420_encrypt	10058	437	11872	407
Software_encrypt_ref	11980	2844	13206	2811
Software_encrypt	12720	1915	13980	1883
Software_encrypt_minRAM	12748	625	14200	887

1.2.2 เทคนิคที่ใช้ในการปรับปรุงการเข้ารหัสแบบ Advance Encryption Standard

การพัฒนาการเข้ารหัสข้อมูลแบบ AES พบได้หลายหลายรูปแบบ แต่ในงานวิจัยฉบับนี้ได้นำเสนอเกี่ยวกับการสร้างวงจร เพื่อให้ได้ต้นแบบของวงจรในการนำไปใช้งานได้จริง การปรับปรุงการเข้ารหัสแบบ AES เพื่อให้สามารถใช้งานในเครือข่ายเซนเซอร์ไร้สายนั้นได้คำนึงถึงปริมาณงานต่อหน่วยเวลาที่เพิ่มขึ้นอีกด้วย โดยใช้เทคนิคโครงสร้างการทำงานแบบไปป์ไลน์ (Pipeline) [4] หรือการประหยัดพลังงาน โดยการทำให้กระบวนการเร็วขึ้นกว่าเดิมเรียกวิธีการในการทำงานนั้นว่า “DOR + K scheme” [5] หรือการเชื่อมต่อโดยตรงอย่างเหมาะสม พร้อมกับคีย์ (Direct Optimized Routing with Key Storage) ซึ่งเป็นการสร้าง S-box โดยใช้ตารางการเทียบค่า โครงสร้างการทำงานแบบนี้ใช้ทรัพยากรไปเพียง 2439 แผ่นลอจิกและลดพลังงานลงจากรูปแบบของมาตรฐานการเข้ารหัสของ NIST หรือการใช้เพียง DOR ไม่รวม K scheme ซึ่งหาก

ระบบทำงานที่ 5 MHz จะใช้กำลังงานน้อยกว่า NIST 44 % (Spatun3) และหากระบบทำงานที่ 25 MHz จะใช้กำลังงานน้อยกว่า 20% (Virtex-II)

นอกจากนั้น งาน [6] นี้ใช้วิธีการลดรูป ตาราง LUT โดยหา prime number ของฟังก์ชันการทำงาน S-box ซึ่งมีอินพุตขนาด 8 บิตและเอาต์พุตขนาด 8 บิตเช่นกัน ผลการทดลองที่ได้คือ ที่ความถี่ 512.821 MHz ไม่มีการใช้งานในส่วนของ BRAMs และจำนวนเกทที่ใช้งานคือ 786 เกท ใช้ลอจิกเกตไปจำนวน 31 ตัว โดยเมื่อเปรียบเทียบกับการใช้กาลัวร์ฟิลด์ GF(2^8) การลด prime number ในงานชิ้นนี้มีประสิทธิภาพดีกว่า 38% เมื่อทำงานที่ความถี่สูงสุด ส่วนงาน [7] เป็นการเข้ารหัส โดยใช้โครงสร้างแบบขนาน ประกอบด้วย 2 ส่วนหลักคือ encryption core (AES) และ Galois Field multiplier โดยใช้ GF128 โดยสร้างจริงบน FPGA ผลลัพธ์ที่ได้คือ ที่ความถี่ 294 MHz ให้ปริมาณงานต่อหน่วยเวลา 1.4 Gbps โดยใช้ Xilinx 5vlx50t ซึ่งเป็นฮาร์ดแวร์สำหรับ 10/100/1000 ของ Ethernet MAC Block การสร้างนี้ใช้ทรัพยากรบนอุปกรณ์ FPGA น้อยกว่า 10% ของที่มีอยู่ คือจำนวนรีจิสเตอร์ที่ใช้ 2656 ตัว และ LUTs 2506 ตัว และยังใช้ RAM ขนาด 18X2 อีก 5 ตัว ซึ่ง core ที่ได้มีความสามารถในการส่งข้อมูลแบบ real throughput ซึ่งติดต่อกับเน็ตเวิร์คแบบ Gigabit Ethernet เพื่อให้สามารถใช้งานได้จริง ความแตกต่างของงานที่สร้างขึ้นขึ้นอยู่กับ ขนาดและความเร็ว โดยงานนี้สามารถยืดหยุ่นได้ในส่วนของการใช้งานด้านของขนาดข้อมูลและความเร็ว

1.3 วัตถุประสงค์

1. เพื่อเปรียบเทียบ ทดสอบและวิเคราะห์การทำงานของอัลกอริทึมการเข้ารหัสที่มีความเหมาะสมกับการประยุกต์ใช้งานกับเครือข่ายเซนเซอร์ไร้สาย
2. เพื่อออกแบบ พัฒนาเทคนิคการลดกำลังงานและนำมาประยุกต์ใช้กับอัลกอริทึมของการเข้ารหัสได้อย่างมีประสิทธิภาพ
3. เพื่อได้วงจรการเข้ารหัส/ถอดรหัสกำลังงานต่ำและมีประสิทธิภาพสูงที่เหมาะแก่งานในเครือข่ายเซนเซอร์ไร้สาย

1.4 ขอบเขต

1. ออกแบบและพัฒนาวงจรเข้ารหัสแบบ AES ขนาด 128 บิตคีย์ สำหรับประยุกต์ใช้งานบนเครือข่ายเซนเซอร์ไร้สาย
2. ทดสอบการทำงานของวงจรเข้ารหัสที่พัฒนาบน FPGA

1.5 ขั้นตอนและวิธีดำเนินงาน

การดำเนินงานในการทำวิทยานิพนธ์ชุดนี้ มีขั้นตอนการทำงานดังต่อไปนี้

1. ศึกษาค้นคว้าข้อมูลเอกสารที่เกี่ยวข้องกับวิทยานิพนธ์ดังนี้
    - เทคนิคการประยุกต์การเข้ารหัสข้อมูลบนเครือข่ายเซนเซอร์ไร้สาย
    - กระบวนการ รูปแบบและขั้นตอนการเข้ารหัสข้อมูลแบบ AES
    - เทคนิคการลดกำลังงานให้กับวงจร
    - เทคนิคการเพิ่มประสิทธิภาพให้กับวงจร
    - งานวิจัยอื่นๆ ที่เกี่ยวข้อง
2. วิเคราะห์และออกแบบวงจรการเข้ารหัสแบบ AES
    - วิเคราะห์กระบวนการการเข้ารหัสแบบ AES
    - การออกแบบวงจรโดยใช้เทคนิคการคำนวณทางคณิตศาสตร์ที่แตกต่างกัน
    - ปรับปรุงกระบวนการ ในส่วนของ S-box โดยใช้ $GF(2^4)^2$ เพื่อให้ระบบทำงานได้อย่างมีประสิทธิภาพและประหยัดกำลังงาน
3. ทดสอบและแก้ไขกระบวนการเข้ารหัสแบบ AES
    - ใช้ภาษา VHDL บรรยายการออกแบบ โดยใช้โปรแกรม Xilinx ISE 10.1
    - ทดสอบวงจรด้วยเทสเบนช์เวฟฟอร์ม (Testbench Waveforms)
    - วิเคราะห์หาข้อผิดพลาดและแนวทางในการปรับปรุงแก้ไข
    - ปรับปรุงวงจรเพื่อให้เหมาะสมต่อการใช้งานบนเครือข่ายเซนเซอร์ไร้สาย
    - ทดสอบวงจรภายหลังจากการปรับปรุงแก้ไขแล้ว
4. จัดทำเอกสารรายงานผลการทำวิจัยฉบับสมบูรณ์

1.6 เครื่องมือที่ใช้

1. ด้านฮาร์ดแวร์
    - เครื่องคอมพิวเตอร์ ความเร็วหน่วยประมวลผลที่ 2.66 GHz และมีหน่วยความจำ 4 GB ฮาร์ดดิสก์ความจุ 320 กิกะไบต์ จำนวน 1 ชุด
2. ด้านซอฟต์แวร์
    - ระบบปฏิบัติการ Microsoft Window XP
    - โปรแกรม Xilinx ISE 10.1

1.7 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้วงจรการเข้ารหัสที่เหมาะสมสำหรับเครือข่ายเซนเซอร์ไร้สาย
2. ได้วงจรการทำงานของอัลกอริทึม AES ในการเข้ารหัส/ถอดรหัสที่ประหยัดกำลังงานและมีประสิทธิภาพสูงสำหรับเครือข่ายเซนเซอร์ไร้สาย

บทที่ 2
ทฤษฎีและหลักการ

อัลกอริทึมการเข้ารหัสที่ปลอดภัย หมายถึง สามารถถอดรหัสได้ยากหรือจำเป็นที่จะต้องใช้เวลานานหรือต้องการกระบวนการที่ซับซ้อนมากจึงจะสามารถนำข้อมูลไปใช้งานได้ ถึงแม้ว่าปัจจุบันมีหลายอัลกอริทึมให้เลือกใช้ก็ตาม แต่ละอัลกอริทึมก็มีลักษณะเด่นเฉพาะตัว และอัลกอริทึมแบบ AES เอง หากพิจารณาในด้านความปลอดภัยแล้ว ก็เป็นตัวเลือกที่เหมาะสมกับการเข้ารหัสข้อมูล เนื่องจากในการเข้ารหัสข้อมูลนั้นมีการทำงานในรูปแบบบล็อกข้อมูลขนาด 128 บิตขึ้นไป ซึ่งจะแสดงรายละเอียดเกี่ยวกับวิธีการภายในฟังก์ชันของการเข้ารหัสแบบ AES ในหัวข้อถัดไป จากนั้นจะได้กล่าวถึงเทคนิคที่ใช้ในการคำนวณการเข้ารหัสแบบ AES ในแบบต่างๆ และเทคนิคที่ทำให้วงจรเข้ารหัสแบบ AES มีประสิทธิภาพทั้งเป็นการเพิ่มความเร็วให้กับวงจร หรือการใช้เทคนิคการออกแบบวงจรเพื่อลดกำลังงาน

2.1 การเข้ารหัสแบบ AES

อัลกอริทึมแบบ AES (Advanced Encryption Standard) [8] ถูกพัฒนาขึ้นโดย Joan Daemen และ Vincent Rijmen นักวิจัยชาวเบลเยี่ยม ได้รับการพัฒนาขึ้นมาตั้งแต่ปี 1998 อัลกอริทึมแบบ AES ได้รับการปรับปรุงและพัฒนามาจากอัลกอริทึมแบบ DES (Data Encryption Standard) นอกจากนั้นยังเป็นอัลกอริทึมที่มีความเร็วสูงและมีขนาดเล็ก อีกทั้งเป็นอัลกอริทึมที่ชนะในการประกวดอัลกอริทึมของหน่วยงาน National Institute of Standard and Technology (NIST) ในสหรัฐอเมริกา จึงถูกจัดให้เป็นมาตรฐานการเข้ารหัสชั้นสูงในปี 2001 โดยอัลกอริทึมนี้เดิมทีใช้ชื่อว่า Rijndael (Rijmen & Daemen)

ตามที่ได้กล่าวไปในข้างต้น อัลกอริทึมนี้เป็นแบบ Block Cipher ข้อมูลที่ใช้จะถูกจัดเรียงเป็นบล็อก ในกระบวนการเข้ารหัสแบบ AES ที่ได้ออกแบบนี้จะมีบล็อกข้อมูลขนาด 128 บิต โดยข้อมูลนี้จะถูกจัดเรียงลงในบล็อกจำนวน 16 บล็อก ในแต่ละบล็อกมีขนาดข้อมูลจำนวน 8 บิต ซึ่งจัดเรียงในอาร์เรย์ 2 มิติ ขนาด 4×4 ไบต์ ดังภาพประกอบ 2-1 (a) การเรียงค่าของข้อมูล ซึ่งแต่ละไบต์ข้อมูลจะถูกเรียกว่าสเตท (State) แทนด้วยสัญลักษณ์ $S_{\mathrm{row,column}}$ เช่น $S_{1,2}$ ระบุว่าค่าสเตทนี้อยู่ในตำแหน่งแถวที่ 1 หลักที่ 2 เมื่อมีข้อมูลเข้ามาในกระบวนการเข้ารหัส ข้อมูลจะถูกเรียงลงในแต่ละสเตท โดยป้อนข้อมูลจากสเตท $S_{0,0}$, $S_{0,1}$, $S_{0,2}$, $S_{0,3}$, $S_{1,0}$, $S_{1,1}$, … , $S_{3,2}$, $S_{3,3}$ ตามลำดับ และ

กระบวนการสร้างคีย์ก็เช่นเดียวกัน ดังภาพประกอบ 2-2 (b) การเรียงค่าของคีย์โดยค่าของสเตทคีย์แทนด้วยสัญลักษณ์ $K_{\text{row,column}}$

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

(a) การเรียงค่าของข้อมูล

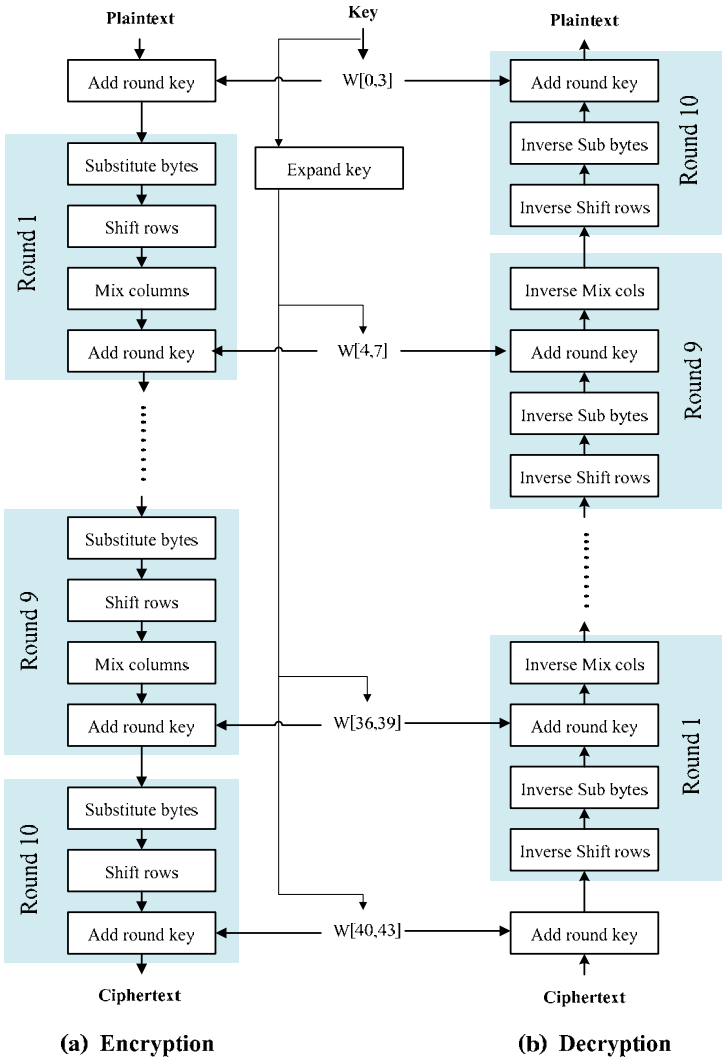
$K_{0,0}$	$K_{0,1}$	$K_{0,2}$	$K_{0,3}$
$K_{1,0}$	$K_{1,1}$	$K_{1,2}$	$K_{1,3}$
$K_{2,0}$	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$
$K_{3,0}$	$K_{3,1}$	$K_{3,2}$	$K_{3,3}$

(b) การเรียงค่าของคีย์

ภาพประกอบ 2-1 สเตทการจัดเรียงอินพุตก่อนเข้าสู่กระบวนการ

โดยกระบวนการในการทำงานของอัลกอริทึมแบบ AES สามารถแบ่งได้เป็นกระบวนการของการเข้ารหัสและถอดรหัสข้อมูล ดังแสดงในภาพประกอบ 2-2 (a) กระบวนการของการเข้ารหัส (Encryption) รับข้อมูลที่ต้องการเข้ารหัสหรือเรียกว่า เพลนเท็ก (Plaintext) และคีย์ (Key) เข้าสู่กระบวนการทำงานของการเข้ารหัสแบบ AES เมื่อเสร็จจะได้ข้อมูลที่เข้ารหัสแล้วหรือเรียกว่า ไซเฟอร์เท็ก (Ciphertext) ส่วนกระบวนการของการถอดรหัส (Decryption) ดังแสดงในภาพประกอบ 2-2 (b) นั้นจะกลับฟังก์ชันการทำงานกับกระบวนการเข้ารหัส

ขั้นตอนของการเข้า/ถอดรหัสแบบ AES ซึ่งใช้คีย์ขนาด 128 บิตมีรอบการทำงานของ round function จำนวน 10 ครั้ง โดยในแต่ละรอบประกอบด้วยฟังก์ชันในการทำงาน 4 ฟังก์ชัน คือ SubByte, ShiftRow, MixColumn และ AddRoundKey ตามลำดับ ยกเว้นในรอบสุดท้ายกระบวนการเข้ารหัสจะไม่มีฟังก์ชัน MixColumn สำหรับกระบวนการถอดรหัสแต่ละรอบการทำงานจะประกอบด้วยฟังก์ชัน InverseShiftRow, InverseSubByte, InverseMixColumn และ AddRoundKey ยกเว้นในรอบสุดท้ายของการถอดรหัสจะไม่มีฟังก์ชัน InverseMixColumn ซึ่งรายละเอียดของแต่ละฟังก์ชันมีดังนี้



ภาพประกอบ 2-2 กระบวนการในการทำงานของอัลกอริทึม AES

2.1.1 การเปลี่ยนค่าโดยการแทนที่ไบต์ (SubBytes, S-box)

การเปลี่ยนค่าโดยการแทนที่ของไบต์หรือที่เรียกว่า Subbyte เป็นการแทนที่ของไบต์แบบนอนลิเนียร์ซึ่งเป็นกระบวนการอิสระไม่ขึ้นกับไบต์อื่น จะเห็นได้ว่าการสับเปลี่ยนข้อมูลโดยการแทนที่ไบต์ใช้ตาราง S-box ในการนำค่าต่างๆ มาใช้งาน ค่าเหล่านั้นสร้างขึ้นโดยใช้สองส่วนประกอบ คือ มัลติพลิเคชันอินเวิร์ส (Multiplication Inverse) และการประยุกต์โดยการเปลี่ยนรูป (Affine Transformation และ Inverse Affine Transformation) ซึ่งได้อธิบายรายละเอียดไว้ ดังนี้

1. มัลติพลิเคชันอินเวิร์ส (Multiplication Inverse) เป็นการคำนวณหาค่าอินเวิร์สของ a (a^{-1}) คือค่า a ที่ไปคูณกับค่าอินเวิร์สของตัวมันเองแล้วได้ค่าเป็น 1 ตัวอย่างเช่น อีลีเมนต์ (Element) มีค่าเป็น {7a} จะหาค่ามัลติพลิเคชันอินเวิร์สได้คือ {d0} ซึ่งการหาค่ามัลติพลิเคชัน -

อินเวิร์สทำได้หลายวิธี อาทิเช่นวิธี Brute-force search หรือการใช้ Extended Euclidean algorithm เป็นต้น ตารางการเทียบค่าของมัลติพลิเคชันอินเวิร์สแสดงในตารางที่ 2-1 พร้อมทั้งแสดงวิธีในการคิดค่ามัลติพลิเคชันอินเวิร์สดังนี้

การคำนวณค่ามัลติพลิเคชันอินเวิร์ส [9] กำหนดให้มัลติพลิเคชันอินเวิร์สของ a (a^{-1}) เมื่ออีลีเมนต์ $a \in GF(2^8)$ คือ $\forall a \in GF(2^8) \setminus 0 : a \otimes a^{-1} = \{1\}$

<u>ตัวอย่างที่ 1</u> การคำนวณค่ามัลติพลิเคชันอินเวิร์สของอีลีเมนต์ {30}

$$\begin{aligned}\{30\} \cdot \{2c\} &= \{00110000\}\{00101100\}\\ &= \{x^5 + x^4\} \cdot \{x^5 + x^3 + x^2\}\\ &= \{x^{10} + x^8 + x^7\} + \{x^9 + x^7 + x^6\}\\ &= \{x^{10} + x^9 + x^8 + x^6\}\ modulo\ x^8 + x^4 + x^3 + x + 1\\ &= \{11101000000\}\ modulo\ \{100011011\}\end{aligned}$$

11101000000 *(mod)* 100011011
^ 100011011
01100101100
^ 100011011
0100011010
^ 100011011
000000001

$\therefore \{30\} \cdot \{2c\} = \{01\}$ ดังนั้นค่า {30} มีค่ามัลติพลิเคชันอินเวิร์สเป็น {2c} และกลับกัน ค่า {2c} มีค่ามัลติพลิเคชันอินเวิร์สเป็น {30} ดังแสดงในตารางที่ 2-1

<u>ตัวอย่างที่ 2</u> การคำนวณค่ามัลติพลิเคชันอินเวิร์สของอีลีเมนต์ {f9}

$$\begin{aligned}\{f9\} \cdot \{9c\} &= \{11111001\}\{10011100\}\\ &= \{x^7 + x^6 + x^5 + x^4 + x^3 + 1\} \cdot \{x^7 + x^4 + x^3 + x^2\}\\ &= \{x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^7\}\\ &\quad + \{x^{11} + x^{10} + x^9 + x^8 + x^7 + x^4\}\\ &\quad + \{x^{10} + x^9 + x^8 + x^7 + x^6 + x^3\}\\ &\quad + \{x^9 + x^8 + x^7 + x^6 + x^5 + x^2\}\\ &= \{x^{14} + x^{13} + x^{12} + x^{10} + x^9 + x^8 + x^5 + x^4 + x^3 + x^2\}\\ &\quad modulo\ x^8 + x^4 + x^3 + x + 1\\ &= \{111011100111100\}\ modulo\ \{100011011\}\end{aligned}$$

111011100111100 *(mod)* 100011011
^ 100011011
011000111111100
^ 100011011
01001010011100
^ 100011011
0001100101100
^ 100011011
0100011010
^ 100011011
000000001

$\therefore \{f9\} \cdot \{9c\} = \{01\}$ ดังนั้นค่า $\{f9\}$ มีค่ามัลติพลิเคชันอินเวิร์สเป็น $\{9c\}$ และกลับกัน ค่า $\{9c\}$ มีค่ามัลติพลิเคชันอินเวิร์สเป็น $\{f9\}$ ดังแสดงในตารางที่ 2-1

ตารางที่ 2-1 ค่าอินเวิร์สของ xy เมื่อ $\{xy\} \in GF(2^8)$

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

2. การประยุกต์โดยการเปลี่ยนรูป (Affine Transformation และ Inverse Affine Transformation) มีรายละเอียดวิธีในการคำนวณดังนี้

1) วิธีในการคำนวณค่าอินเวิร์สแอฟไฟน์ทรานสฟอร์ม [9]

วิธีในการคำนวณค่าแอฟไฟน์ทรานสฟอร์มนั้น ให้ b_i คือ บิตอินพุตและ i แสดงการระบุตำแหน่ง เมื่อ $0 \leq i \leq 8$ โดยนำค่า b_i คูณกับเมตริกค่าคงที่แสดงด้วยสัญลักษณ์ $[M]$ และนำไปบวกกับค่าคงที่ c มีค่าคงที่เท่ากับ $\{63\}_{16}$ จะมีค่าเป็น $\{01100011\}_2$ และจะได้ผลลัพธ์ของการหาค่าแอฟไฟน์ทรานสฟอร์มโดยแสดงด้วยสัญลักษณ์ b_i' ซึ่งแสดงโดยใช้สมการที่ (2-1)

$$\{b_i{}'\} = [M]\{b_i\} + \{c_i\} \tag{2-1}$$

เมื่อแทนค่าของเมตริก $[M]$ และค่าคงที่ c จะได้สมการที่ (2-2)

$$\begin{bmatrix} b_0{}' \\ b_1{}' \\ b_2{}' \\ b_3{}' \\ b_4{}' \\ b_5{}' \\ b_6{}' \\ b_7{}' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{2-2}$$

และได้สรุปการหาค่าเอาพุตในแต่ละตำแหน่งที่ i ได้ดังนี้

$$b_0{}' = b_0 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus c_0 = b_0 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus 1$$

$$b_1{}' = b_0 \oplus b_1 \oplus b_5 \oplus b_6 \oplus b_7 \oplus c_1 = b_0 \oplus b_1 \oplus b_5 \oplus b_6 \oplus b_7 \oplus 1$$

$$b_2{}' = b_0 \oplus b_1 \oplus b_2 \oplus b_6 \oplus b_7 \oplus c_2 = b_0 \oplus b_1 \oplus b_2 \oplus b_6 \oplus b_7 \oplus 0$$

$$b_3{}' = b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_7 \oplus c_3 = b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_7 \oplus 0$$

$$b_4{}' = b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_4 \oplus c_4 = b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_4 \oplus 0$$

$$b_5{}' = b_1 \oplus b_2 \oplus b_3 \oplus b_4 \oplus b_5 \oplus c_5 = b_1 \oplus b_2 \oplus b_3 \oplus b_4 \oplus b_5 \oplus 1$$

$$b_6{}' = b_2 \oplus b_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus c_6 = b_2 \oplus b_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus 1$$

$$b_7{}' = b_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus c_7 = b_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus 0$$

หรือสรุปเป็นสมการรวมได้ดังนี้

$$b_i{}' = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i \tag{2-3}$$

<u>ตัวอย่าง</u> การคำนวณค่าแอฟไฟน์ทรานสฟอร์มของค่า {2c}

$$b = \{2c\} = \{00101100\}$$

เมื่อแจกแจงในแต่ละบิต $b_0{}' = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0$

$$b_1{}' = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

$$b_2{}' = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$b_3' = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

$$b_4' = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

$$b_5' = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$b_6' = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

$$b_7' = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$$

$$\therefore b' = \{00000100\} = \{04\}$$

จากค่า {30} ที่นำเข้าสู่ฟังก์ชันมัลติพลิเคชันอินเวิร์สจะได้ค่า {2c} และเมื่อนำค่า {2c} มาหาค่าแอฟไฟน์ทรานสฟอร์มจะได้ {04} ซึ่งฟังก์ชันทั้งสองก็คือการทำฟังก์ชัน S-box นั่นเอง หากนำค่า {30} เปิดตาราง S-box (ตารางที่ 2-2) ก็จะได้ค่า {04} เช่นกัน

2) วิธีในการคำนวณค่าอินเวิร์สแอฟไฟน์ทรานสฟอร์ม [9]

วิธีในการคำนวณค่าอินเวิร์สแอฟไฟน์ทรานสฟอร์มนั้นคล้ายกับการคำนวณค่าแอฟไฟน์ทรานสฟอร์ม แต่เมตริกค่าคงที่จะใช้อีกค่าหนึ่ง โดยแสดงด้วยสัญลักษณ์ $[M_{inv\_aff}]$ และนำไปบวกกับค่าคงที่ c_{inv_aff} มีค่าคงที่เท่ากับ $\{05\}_{16}$ จะมีค่าเป็น $\{00000101\}_2$ และจะได้ผลลัพธ์ของการหาค่าอินเวิร์สแอฟไฟน์ทรานสฟอร์มโดยแสดงด้วยสัญลักษณ์ b_i' ซึ่งแสดงโดยใช้สมการที่ (2-4)

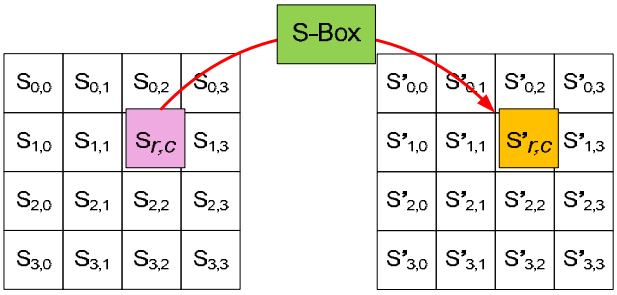
$$\{b_i'\} = [M_{inv\_aff}]\{b_i\} + \{c_{inv\_aff\ @\ i}\} \tag{2-4}$$

เมื่อแทนค่าของเมตริก $[M_{inv}]$ และค่าคงที่ $c_{inv\_aff@\ i}$ จะได้สมการที่ (2-5)

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2-5}$$

จากองค์ประกอบทั้งสองส่วนที่ได้กล่าวไป คือ ค่ามัลติพลิเคชันอินเวิร์สและค่าจากการประยุกต์โดยการเปลี่ยนรูปแสดงให้เห็นรูปแบบของการสับเปลี่ยนข้อมูลการแทนที่ของไบต์โดยใช้ตารางการเทียบค่าหรือที่เรียกว่า S-box การใช้ตารางการเทียบค่าของกระบวนการ S-box

นำเสนอในรูปแบบเลขฐาน 16 ในตารางที่ 2–2 ตัวอย่างเช่น $S_{x,y} = S_{1,2} = \{53\}$ สามารถแทนที่ได้โดยการค้นหาค่าที่ระบุไว้ตรงตำแหน่งที่ตัดกันของแถวที่ 5 และหลักที่ 3 ในตารางการเทียบค่าของการใช้งาน S-box (ตารางที่ 2–2) ซึ่งค่าที่ได้คือ $S'_{1,2} =$ ค่า {ed} และหากเป็นการถอดรหัสจะใช้ตารางการเทียบค่าสำหรับการถอดรหัสโดยแสดงไว้ในภาคผนวก ก



ภาพประกอบ 2-3 การเปลี่ยนแปลงของข้อมูลเมื่อผ่านกระบวนการ S-box จาก $S_{r,c}$ เป็น $S'_{r,c}$

ตารางที่ 2-2 ตารางการเทียบค่าของการใช้งาน S-box

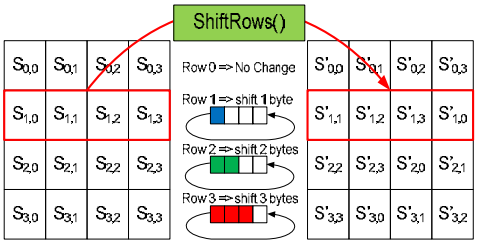
Y / X	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	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

2.1.2 การเปลี่ยนค่าโดยการเลื่อนแถว (ShiftRows)

การเปลี่ยนค่าโดยการเลื่อนแถวเป็นการสลับไบต์ข้อมูลขนาด 4×4 โดยที่ในแต่ละแถวไบต์ข้อมูลจะถูกเลื่อนไปทางขวา เริ่มจากข้อมูลในแถวแรกจะไม่มีการเลื่อนไบต์ จากนั้นแถวที่สอง สาม และสี่ จะถูกเลื่อนไปทางขวา หนึ่ง สอง และสามไบต์ตามลำดับดังภาพประกอบ 2-4 และสามารถแสดงการทำงานในรูปแบบของสมการการเลื่อนข้อมูลไบต์ได้ดังนี้

$$S_{\text{row,column}} = S_{\text{row,(column+shift(row,N}_b\text{))mod N}_b} \tag{2-6}$$

เมื่อ 0 < row < 4 และ 0 $\leq$ column < Nb (Nb คือจำนวนคอลัมน์ เท่ากับ 4)



ภาพประกอบ 2-4 การเปลี่ยนแปลงของข้อมูลเมื่อผ่านกระบวนการ ShiftRow จาก $S_{r,c}$ เป็น $S'_{r,c}$

2.1.3 การเปลี่ยนค่าโดยการเลื่อนข้อมูลในแต่ละคอลัมน์ (MixColumns)

กระบวนการเปลี่ยนค่าโดยการเลื่อนข้อมูลในแต่ละคอลัมน์ (MixColumns) จะทำแบบ column-by-column โดยจะคงรูปคอลัมน์ของ 4-term polynomial เอาไว้ คอลัมน์หนึ่งๆ จะใช้ polynomial over GF(2^8) และ multiplied modulo X^4+1 กับ fixed polynomial $a(x)$ซึ่งมีค่าดังนี้คือ

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \tag{2-7}$$

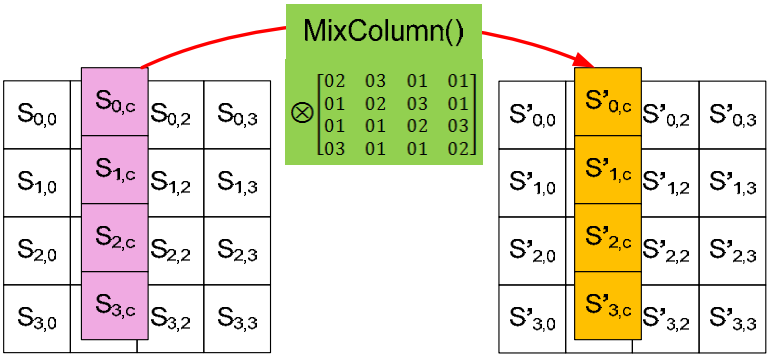
สามารถเขียนในรูปแบบของเมตริกซ์ โดย

ให้ $s'(x) = a(x) \otimes s(x)$ ได้ดังนี้

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \text{ เมื่อ } 0 \leq c < Nb \tag{2-8}$$

ผลลัพธ์ที่ได้จากการคูณ ของทั้งสี่ไบต์ในหนึ่งคอลัมน์สามารถเขียนแทนได้ด้วยสมการ(2-9) หรือสามารถอธิบายได้จากภาพประกอบ 2-5

$$
\begin{aligned}
s'_{0,c} &= (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\
s'_{1,c} &= s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c} \\
s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c}) \\
s'_{3,c} &= (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})
\end{aligned}
\tag{2-9}
$$



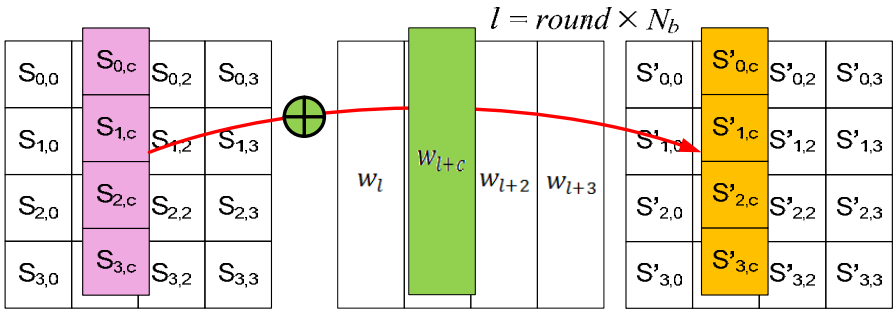
ภาพประกอบ 2-5 การเปลี่ยนแปลงของข้อมูลเมื่อผ่านกระบวนการ MixColumn จาก $S_{r,c}$ เป็น $S'_{r,c}$

2.1.4 การเปลี่ยนค่าโดยการเพิ่มคีย์ (AddRoundKey)

ขั้นตอนนี้เป็นการทำ XOR ระหว่างข้อมูล (S) ในแต่ละคอลัมน์กับคีย์ที่ตรงกับคอลัมน์นั้นๆ ซึ่งในแต่ละรอบของการเข้ารหัส ค่าคีย์จะถูกเปลี่ยนแปลงรูปแบบซึ่งจะได้อธิบายในหัวข้อที่ 2.1.5 และภาพประกอบ 2-6 แสดงขั้นตอนของการทำงานฟังก์ชัน AddRoundKey ดังสมการ (2-10)

$$
[s'_{0,c} \quad s'_{1,c} \quad s'_{2,c} \quad s'_{3,c}] = [s_{0,c} \quad s_{1,c} \quad s_{2,c} \quad s_{3,c}] \oplus [w_{round*Nb+c}] \tag{2-10}
$$

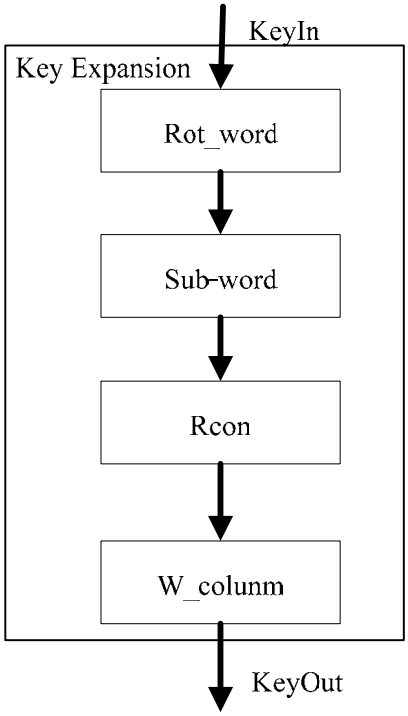
ค่าคีย์ที่ใช้ในการเริ่มต้นกระบวนการเข้ารหัสหรือถอดรหัสจะแสดงใน Round = 0 ซึ่งสำคัญมาก จึงควรเก็บรักษาคีย์ไว้เป็นอย่างดี และจะเห็นได้ว่าฟังก์ชัน AddRoundKey จะนำค่าคีย์มา XOR ทั้งหมดไม่เกินค่า Nr ซึ่งอยู่ในช่วงของ $1 \leq$ round $\leq$ Nr (Nr คือจำนวนของรอบ) แต่ละรอบแบ่งค่าคีย์ออกเป็นกลุ่มของคอลัมน์โดยเรียกว่า word โดยแสดงในภาพประกอบ 2-6 กำหนดให้รอบในการเปลี่ยนแปลงค่าคีย์แต่ละ word แทนด้วย l ซึ่ง $l = round \times N_b$ เมื่อ N_b หมายถึงจำนวนคอลัมน์



ภาพประกอบ 2-6 การเปลี่ยนแปลงของข้อมูลในกระบวนการ AddRoundKey จาก $S_{r,c}$ เป็น $S'_{r,c}$

2.1.5 การเพิ่มจำนวนคีย์ (Key Expansion)

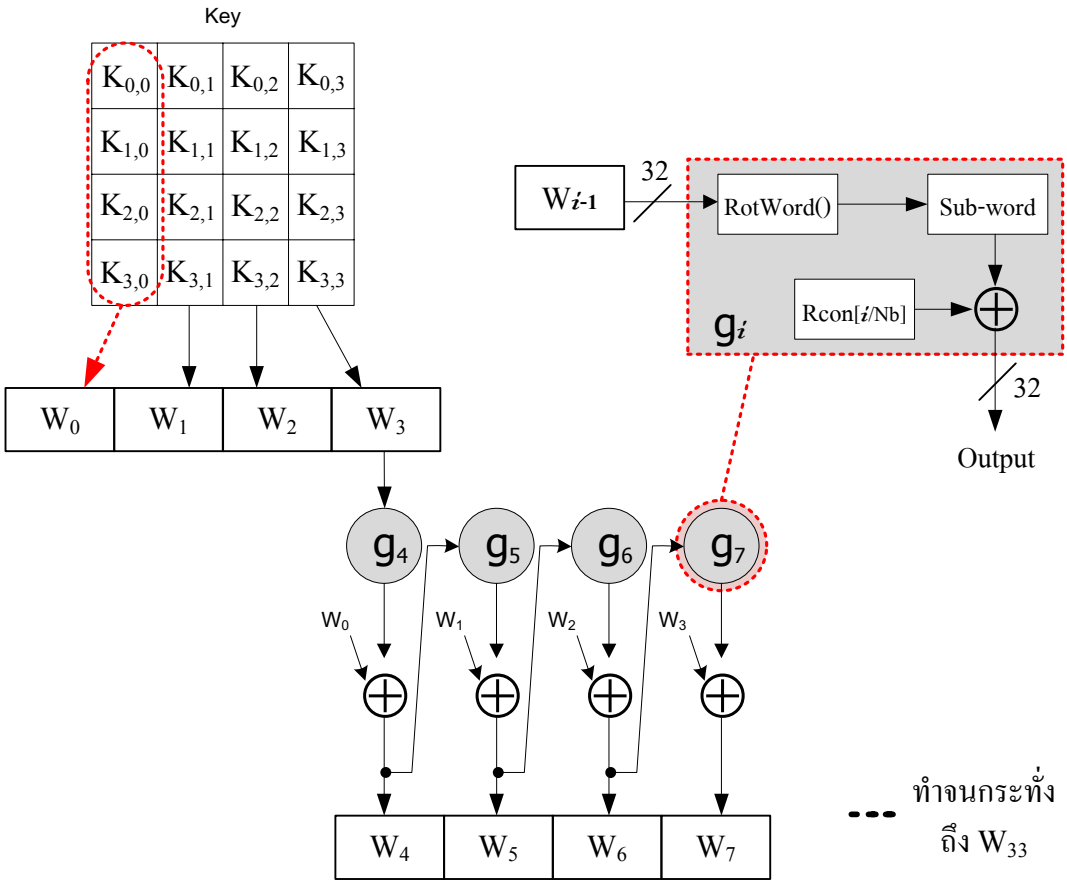
หนึ่งรอบของการเพิ่มจำนวนคีย์นั้นประกอบด้วยฟังก์ชันย่อยดังนี้ Rotage Word, Sub-Word, Round Constant (Rcon) และ Word Column (W_column) ดังภาพประกอบ 2-7



ภาพประกอบ 2-7 ขั้นตอนการทำ Key expansion

อัลกอริทึม AES นำคีย์ลับ (Cipher Key, K) มาสร้างกระบวนการเพิ่มจำนวนคีย์ (Key Expansion Routine) โดยสามารถเพิ่มจำนวนคีย์นับเป็นจำนวนคอลัมน์หรือ word ได้ทั้งหมดจำนวน N_b*(Nr+1) words เมื่อกำหนดให้ N_b คือ จำนวนคอลัมน์ ในที่นี้ N_b มีค่าเท่ากับ 4 และ Nr หมายถึง จำนวนของรอบ ซึ่ง Nr มีค่าเท่ากับ 10 รอบ

ดังนั้นจะสามารถเพิ่มจำนวนคีย์ได้ทั้งหมด $4 \times (10 + 1) = 44$ words (w_0, …, w_{33}) ซึ่งจะแทนด้วยสัญลักษณ์ [w_i] โดยที่ค่าของ i จะอยู่ในช่วง $0 \leq i < N_b*(Nr+1)$ ขั้นตอนของการทำ Key expansion สามารถแสดงได้ดังภาพประกอบ 2- 8 ซึ่งอธิบายการทำงานได้ดังนี้ ค่าอินพุตคีย์ตั้งต้นจะถูกจัดเรียงเข้าเก็บไว้ในแต่ละ สเตทคีย์ $K_{0,0},K_{0,1},K_{0,2}…K_{3,2},K_{3,3}$ จากนั้นจะถูกรวมกลุ่มในรูปแบบคอลัมน์ เรียกว่า W_i ตัวอย่างเช่น W_0 จะเก็บค่า $K_{0,0}$, $K_{1,0},K_{2,0},K_{3,0}$ ซึ่งจะได้ Word ตั้งต้น 4 ชุด คือ W_0, W_1, W_2 และ W_3 จากนั้นจะนำเข้าสู่กระบวนการการเพิ่มจำนวนคีย์โดยนำ Word ที่ i = 3 (W_3) เข้าสู่ฟังก์ชัน Rotate Word เลื่อนค่าโดยใช้การแทนที่แบบวงกลม (Cyclic Permutation) หากอินพุตเวิร์ดเป็น [$K_{0,3},K_{1,3},K_{2,3},K_{3,3}$] แล้วจะได้ค่าเอาต์พุตเวิร์ดเป็น [$K_{1,3},K_{2,3},K_{3,3},K_{0,3}$] แล้วนำเข้าสู่ฟังก์ชัน Sub-word ในฟังก์ชันนี้จะใช้ S-box ฟังก์ชันจำนวน 4 ชุด เมื่อได้เอาพุตแล้วให้นำค่าที่ได้ไป XOR กับค่าคงที่ในแต่ละรอบจากฟังก์ชัน Rcon เมื่อได้เอาต์พุตแล้วจะนำไป XOR กับค่า word ชุดก่อนหน้า (W_{i-3}) นั่นคือ W_0 จึงจะได้เอาต์พุตเวิร์ดที่ $i = 4$ (W_4) ทำกระบวนการนี้จนกระทั่งถึง W_{33} จึงจะได้ชุดคีย์ที่สร้างใหม่ทั้งหมด 10 ชุด



ภาพประกอบ 2-8 การสร้างคีย์ใน 1 รอบ

2.2 คณิตศาสตร์และพีชคณิตเบื้องต้น

ในการเข้ารหัสข้อมูลโดยใช้อัลกอริทึมแบบ AES มีความจำเป็นในการศึกษาคณิตศาสตร์และพีชคณิตพื้นฐาน ซึ่งข้อมูลในการคำนวณอัลกอริทึมการเข้ารหัสแบบ AES จะคำนวณเป็นไบต์ให้ *a* คือ ข้อมูลในระดับไบต์

$$a = \{a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0\} \quad (2\text{-}11)$$

ซึ่งข้อมูลไบต์สามารถแทนได้ด้วย Galois Field GF(2^8) ในรูปแบบของสมการ polynomial ดังนี้

$$a_{(x)} = \sum_{i=0}^{7} a_i x^i$$

$$= \{a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0x\} \quad (2\text{-}12)$$

โดยที่ a_i เป็นค่าสัมประสิทธิ์ (coefficient) ของ polynomial *a*(*x*) ตัวอย่างเช่น ค่าของเลขฐานสอง $\{01100011\}_2$ หรือ 63_{16} สามารถเขียนในรูปแบบของสมการ polynomial ได้ดังนี้

$$63_{16} = x^6 + x^5 + x + 1$$

2.2.1 การบวก

การบวกกำหนดให้ *a*(*x*), *b*(*x*)∈GF(2^8) ดังนั้นการบวกจึงเป็นการ XOR ค่าสัมประสิทธิ์ซึ่งจะใช้สัญลักษณ์ $\oplus$ ตัวอย่างการบวกมีดังนี้

$(x^6 + x^4 + x^2 + 1) + (x^7 + x^4 + 1) = x^7 + x^6 + x^2$ (รูปแบบ Polynomial)

$\{01010101\} \oplus \{10010001\} = \{11000100\}$ (รูปแบบเลขฐานสอง)

$55_{16} \oplus 91_{16} = c4_{16}$ (รูปแบบเลขฐานสิบหก)

2.2.2 การคูณ

การคูณสำหรับการเข้ารหัสข้อมูลแบบ AES จะมี polynomial ที่ไม่สามารถลดรูปได้คือ $m(x) = x^8 + x^4 + x^3 + x + 1$ หรือ $011b_{16}$ โดยการคูณจะเป็นการคูณ 2 พจน์ polynomial แล้วทำการ modulo ด้วย *m*(*x*) กำหนดให้ผลคูณเป็น *c*(*x*) ของ *a*(*x*) และ *b*(*x*) ตัวอย่างเช่น A = $c3_{16}$ และ B = 85_{16} , $a(x) = (x^7 + x^6 + x + 1)$ และ $b(x) = (x^7 + x^2 + 1)$

$$c(x) = [a(x) \cdot b(x)] \; mod \; (x^8 + x^4 + x^3 + x + 1) \quad (2\text{-}13)$$

$$= [(x^7 + x^6 + x + 1) \cdot (x^7 + x^2 + 1)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$$

$= x^7 + x^5 + x^3 + x^2 + x$

หรือ $c \quad = 10101110_2 = \mathrm{AE}_{16}$

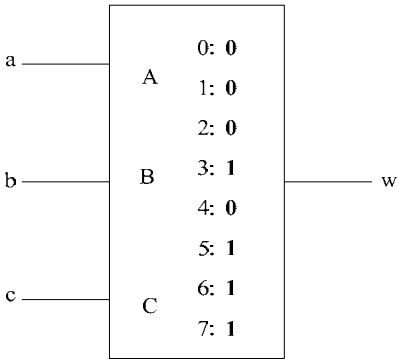
การนำ Galois Field มาใช้ในการขยายอีลีเมนต์จากเดิม GF(2) จะประกอบด้วย {0,1} ให้สามารถใช้ $GF(p^m) = GF(2^8) = GF(256)$ ประกอบด้วย 256 อีลีเมนต์ เมื่อกำหนดให้ $p(x) = x^4 + x + 1$ ซึ่ง $p(x)$ สามารถถูกใช้ในการสร้าง $GF(2^4) = GF(16)$ ซึ่ง $p = 2$ และ $m = 4$ สามารถสร้างอีลีเมนต์ได้ $n = p^m - 1$ ซึ่งหมายถึง 15 ค่า โดยมี polynomial root ที่ $x^4 = x + 1$ ดังตารางที่ 2-3 แสดงค่าเลขฐาน 2 ที่แทนแต่ละอีลีเมนต์ของ $GF(2^4)$

ตารางที่ 2-3 อีลีเมนต์ของ $GF(2^4)$

อีลีเมนต์ของ $GF(2^4)$	**รูปแบบของ polynomial**	**เลขฐานสอง**	**เลขฐานสิบ**
0	0	0000	0
1	1	0001	1
X	X	0010	2
X^2	X^2	0100	4
X^3	X^3	1000	8
X^4	$X + 1$	0011	3
X^5	$X^2 + X$	0110	6
X^6	$X^3 + X^2$	1100	12
X^7	$X^3 + X + 1$	1011	11
X^8	$X^2 + 1$	0101	5
X^9	$X^3 + X$	1010	10
X^{10}	$X^2 + X + 1$	0111	7
X^{11}	$X^3 + X^2 + X$	1110	14
X^{12}	$X^3 + X^2 + X + 1$	1111	15
X^{13}	$X^3 + X^2 + 1$	1101	13
X^{14}	$X^3 + 1$	1001	9

2.3 ตารางการเทียบค่า

โครงสร้างตารางการเทียบค่า (LUT) คือ บล็อกหน่วยความจำขนาดเล็กซึ่งมีอินพุต n จำนวนและเอาต์พุตเพียง 1 จำนวน ภายในตารางการเทียบค่าจะมีคู่ของข้อมูลที่สัมพันธ์กัน ณ ตำแหน่งหนึ่งๆ แต่บางครั้งตารางการเทียบค่า จะบรรจุเพียงค่าเดียวนั้นสื่อความหมายอยู่แล้ว (เช่น ข้อมูลชุด 1, ข้อมูลชุด 2, ข้อมูลชุด 3 เรียงกันไปเป็นต้น) ตัวอย่างเช่น ดัชนี ตารางสี บล็อกหน่วยความจำนี้สามารถโปรแกรมสร้างได้หลายๆ ฟังก์ชัน ปกติโครงสร้างนี้ถูกสร้างโดยการเชื่อมต่อกันของโครงสร้างที่ออกแบบได้บนอุปกรณ์ FPGAs [11]



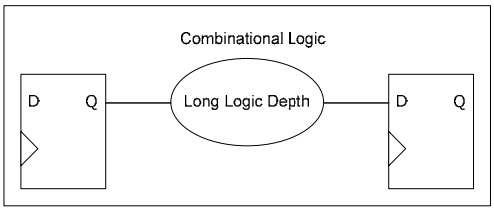
ภาพประกอบ 2-9 โครงสร้างตารางการเทียบค่า

จากภาพประกอบ 2- 9 แสดงตัวอย่างตารางการเทียบค่า ซึ่งมีอินพุตคือ a, b, c และเอาต์พุตคือ w ค่าของ abc ที่เข้ามามีค่าตรงกับ index ที่เท่าไร ก็จะได้เอาพุตตามที่ระบุไว้ในตาราง ซึ่งตารางการเทียบค่านี้เหมือนกับตารางความจริง (Truth Table) เช่น ค่า abc = “011” แปลงเป็นเลขฐาน 10 ได้ค่าเท่ากับ 3 ดังนั้นค่าเอาต์พุต w = “1” เป็นต้น [10]

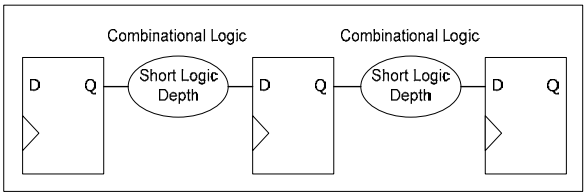
2.4 เทคนิคการเพิ่มความเร็วของวงจร

เทคนิคในการเพิ่มความเร็วให้กับการประมวลผลของวงจรลอจิกจะขึ้นอยู่กับการเลือกใช้สถาปัตยกรรมของวงจรและปรับลดลำดับขั้นการทำงานซึ่งช่วยลดลอจิกของวงจรไปในตัวอีกด้วย โครงสร้างการทำงานของวงจรมีหลายรูปแบบ ได้แก่ วงจรแบบ combinatioinal วงจรแบบไปป์ไลน์ และวงจรแบบขนาน ดังภาพประกอบ 2-10 (a) แสดงการทำงานของวงจรแบบ combinatioinal logic ที่ประกอบด้วยลอจิกเกตในการทำงาน แต่จะสามารถเก็บค่าได้ เมื่อหมดขั้นตอนการประมวลผลทำให้ข้อมูลจะต้องผ่านหลายลำดับขั้นของลอจิก ดังนั้นทำให้คาบสัญญาณนาฬิกา มีความยาวมากขึ้นส่งผลให้ภาพรวมของระบบสามารถทำงานบนความถี่ที่ต่ำกว่าระบบอื่นๆ เพื่อเพิ่มประสิทธิภาพในการทำงานโครงสร้างแบบไปป์ไลน์ดังภาพประกอบ 2-10 (b) จึงถูกนำมาใช้งานเนื่องจากการทำงานแบบไปป์ไลน์จะช่วยให้สามารถใช้ทรัพยากรของวงจรได้อย่างมี

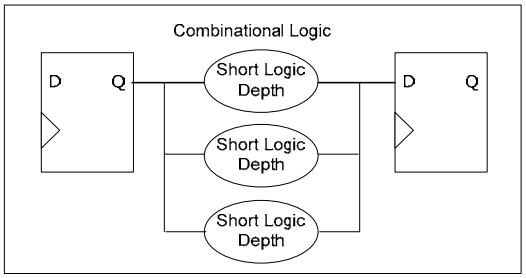
ประสิทธิภาพ ฉะนั้นจึงจะต้องมีการแบ่งการทำงานเป็นขั้นตอนให้เหมาะสม [11] นอกจากนั้นยังมีโครงสร้างวงจรแบบขนานที่สามารถรองรับการทำงานได้มากกว่าหนึ่งอย่างในเวลาเดียวกัน โดยการมีชุดวงจรการทำงานมากกว่า 1 ชุดในการทำงานดังแสดงโครงสร้างในภาพประกอบ 2-10 (c)



(a) โครงสร้างวงจรแบบไม่ใช้ไปป์ไลน์



(b) โครงสร้างวงจรแบบใช้ไปป์ไลน์



(c) โครงสร้างวงจรแบบขนาน

ภาพประกอบ 2-10 โครงสร้างวงจรรวมแบบต่างๆ

2.5 เทคนิคการลดกำลังงานของวงจร

กำลังงาน (Power) มีความสำคัญมากในการออกแบบวงจรในปัจจุบันและองค์ประกอบสำคัญของกำลังงานในวงจรลอจิกทั่วไป คือ กำลังงานแบบไดนามิก ($P_{dynamic}$) กำลังงานแบบ สแตติก (P_{static}) และกำลังงานที่เกิดจากการรั่วไหลของกระแส $\left(P_{leakage}\right)$ สามารถแสดงได้ดังสมการที่ (2-14) ซึ่งกำลังงานของวงจรที่ออกแบบจะขึ้นอยู่กับกำลังงานแบบไดนามิกโดยคำนวณจากสมการที่ (2-15)

$$P_{total} = P_{dynamic} + P_{static} + P_{leakage} \quad (2\text{-}14)$$

$$P_{dynamic} = \frac{1}{2}\alpha CV^2 \quad (2\text{-}15)$$

สมการกำลังงานแบบไดนามิค กำหนดให้ V คือโวลต์ , C คือค่าความจุตัวเก็บประจุ และα คือความถี่ ดังนั้นวิธีในการลดการใช้กำลังงานจึงต้องลดพารามิเตอร์ใดพารามิเตอร์หนึ่งเสียก่อน ในการทดสอบวงจรบน FPGA ขนาดของแรงดันไฟฟ้า (Voltage) จะถูกกำหนดไว้อยู่แล้ว จึงทำให้เหลือพารามิเตอร์เพียงสองตัว นั่นคือ C และ α ที่มีอิทธิพลต่อกำลังงาน [12][13]

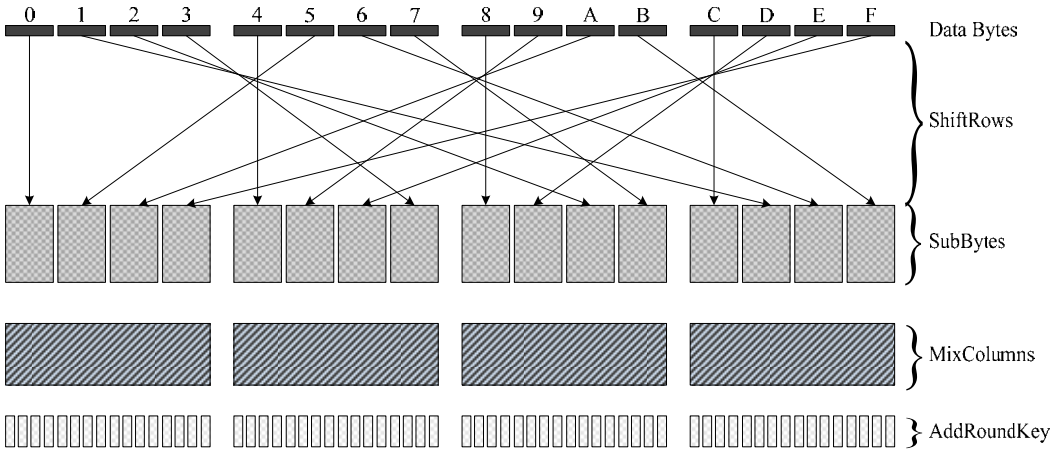
ค่าความจุตัวเก็บประจุ (C) เป็นค่าที่ได้โดยตรงจากจำนวนลอจิกเกต จึงต้องการให้วงจรใช้กำลังงานน้อย โดยจะต้องลดรูปหรือลดจำนวนลอจิกเกตที่ใช้ สำหรับค่าความถี่ (α) เป็นค่าที่ได้โดยตรงจากความถี่ของสัญญาณนาฬิกา (Clock Frequency) ดังนั้นถ้าต้องการให้วงจรมีกำลังงานต่ำ จึงนิยมใช้วงจรแบบขนานเพื่อให้มีความถี่ของ clock ในการทำงานต่ำลง แต่จะต้องแลกกับการเพิ่มจำนวนลอจิกเกต ดังนั้นการออกแบบวงจรให้มีกำลังงานต่ำจำเป็นที่จะต้องคำนึงถึงการใช้กำลังงานตั้งแต่ขั้นตอนการออกแบบ (Energy Awareness) ซึ่งการออกแบบวงจรสำหรับการเข้ารหัสแบบ AES ที่จะนำเสนอในบทที่ 3 จะได้คำนึงถึงกำลังงานที่ใช้โดยยึดหลักจากสมการการสูญเสียกำลังงานของวงจร

บทที่ 3
การออกแบบและวิธีทดสอบระบบ

ในบทนี้จะกล่าวถึงการออกแบบสถาปัตยกรรมของการเข้ารหัสแบบ AES และการออกแบบวงจรที่เกี่ยวข้อง พร้อมทั้งการวิเคราะห์และเปรียบเทียบวงจรเข้ารหัสแบบ AES บนโครงสร้างสถาปัตยกรรมต่างๆที่สร้างด้วยการใช้ตารางการเทียบค่าและใช้เทคนิคการคำนวณด้วยกาลัวร์ฟิลด์ $GF(2^4)^2$ ในการช่วยเพิ่มความเร็วและประสิทธิภาพการเข้ารหัส นอกจากนั้นยังช่วยประหยัดกำลังงาน เพื่อนำไปใช้ในการพัฒนาวงจรการเข้ารหัสบนเครือข่ายเซนเซอร์ไร้สาย โดยจะนำไปทดสอบด้วยโปรแกรม Xilinx ISE

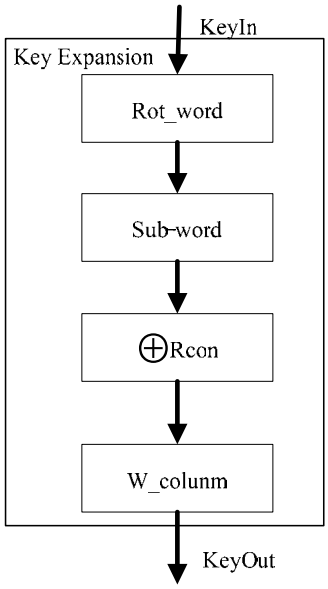
3.1 สถาปัตยกรรมการเข้ารหัสแบบ AES

การเข้ารหัสข้อมูลแบบ AES สามารถแบ่งการทำงานออกเป็น 2 ส่วนใหญ่ๆ ได้แก่ การเข้ารหัสหลัก (core AES) และการสร้างคีย์ (Key Expansion) โดยการทำงานพื้นฐานของการเข้ารหัสหลักนั้น ได้แก่ การการนำข้อมูลระดับไบต์จำนวน 16 ชุด (0-F) ดังภาพประกอบ 3-1 ซึ่งแต่ละชุดประกอบด้วยข้อมูล 8 บิต รวมแล้วมีข้อมูลขนาด 128 บิต มาดำเนินการดังนี้คือ กระบวนการ ShiftRows, SubBytes, MixColumns และ AddRoundKey ซึ่งในกระบวนการ AddRoundKey จะต้องรับคีย์มาจากการเพิ่มจำนวนคีย์ โดยกระบวนการเพิ่มจำนวนคีย์แสดงในภาพประกอบที่ 3-2



ภาพประกอบ 3-1 โครงสร้างการทำงานของการเข้ารหัสแบบ AES

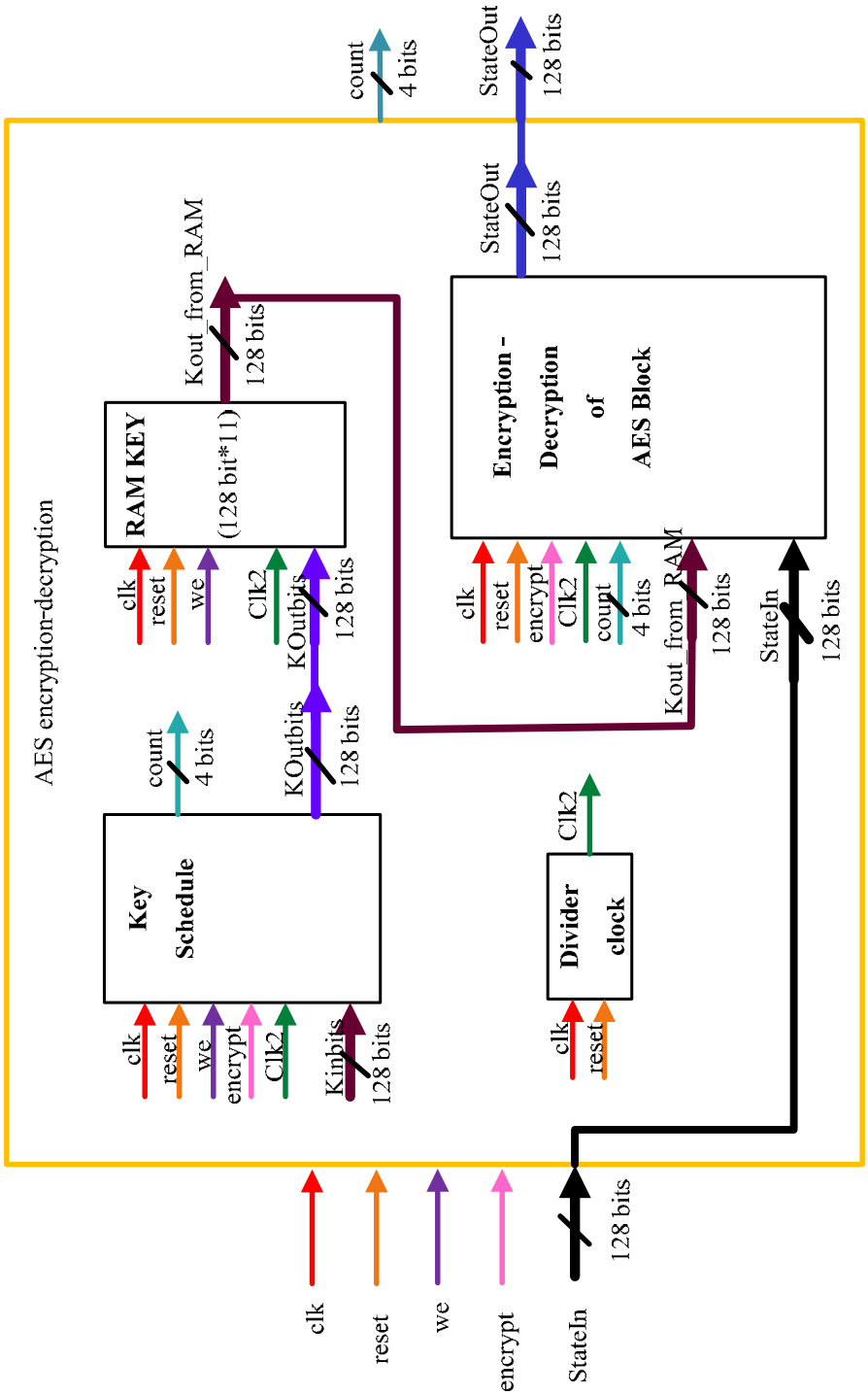
การเพิ่มจำนวนคีย์ที่ใช้ในการเข้ารหัสแบบ AES จะทำงานในระดับไบต์ข้อมูล เพื่อให้สอดคล้องกับการเข้ารหัสหลัก AES โดยมีอินพุตขนาด 128 บิตซึ่งจะป้อนเข้าสู่กระบวนการต่างๆ ดังนี้ Rotage Word, Sub-Word, Round Constant (Rcon) และ Word Column (W_column)



ภาพประกอบ 3-2 โครงสร้างการทำงานของการเพิ่มจำนวนคีย์

จากกระบวนการทำงานที่ได้กล่าวมานี้สามารถออกแบบสถาปัตยกรรมของระบบการเข้ารหัสข้อมูลแบบ AES ได้ดังภาพประกอบที่ 3-3 ซึ่งได้แสดงรายละเอียดการเข้ารหัสข้อมูลแบบ AES จะประกอบด้วยโมดูลการจัดการสัญญาณนาฬิกา โมดูลการเพิ่มจำนวนคีย์ หน่วยความจำเพื่อเก็บคีย์และโมดูลการเข้า/ถอดรหัสแบบ AES การทำงานเริ่มจากกำหนดค่าคีย์เริ่มต้นไว้กับระบบการเข้ารหัสนั้นๆ ซึ่งทำพร้อมกับการโปรแกรมข้อมูล การเพิ่มจำนวนคีย์จะรอให้มีการส่งของสัญญาณนาฬิกา(clk) และสัญญาณการเขียนค่าคีย์ลงหน่วยความจำ RAM (we) ส่วนสัญญาณ encrypt นั้นใช้ในการบอกการทำงานของ S-box เมื่อใช้วิธีกาลัวร์ฟิลด์ $GF(2^4)^2$ ค่าคีย์ที่ใช้ในแต่ละรอบของการเข้ารหัสจะอ่านจาก RAM_KEY โดยการเชื่อมโยงกันของสัญญาณนาฬิกาและการเข้ารหัสแบบ AES นี้ใช้ตัวนับโดยนับสัญญาณนาฬิกาเพื่อให้การทำงานสอดคล้องกันทั้งระบบ

จากสถาปัตยกรรมที่ได้ออกแบบพบว่ามีโมดูลหลัก 2 ตัวคือการเข้ารหัสแบบ AES และโมดูลการเพิ่มจำนวนคีย์ โดยการสร้างวงจรของทั้งสองโมดูลสามารถทำได้ 2 เทคนิคคือการใช้ $GF(2^4)^2$ และการใช้ตารางการเทียบค่า ดังนั้นจึงสามารถสร้างโมดูลการเข้ารหัสแบบ AES และโมดูลเพิ่มจำนวนคีย์ได้ทั้งหมด 4 รูปแบบ ซึ่งรายละเอียดของแต่ละโมดูลจะอธิบายไว้ในส่วนถัดไป นอกจากนี้ยังมีโมดูลสำคัญอื่นๆ ดังนี้



ภาพประกอบ 3-3 สถาปัตยกรรมของระบบการเข้ารหัส AES

3.1.1 การสร้างหน่วยความจำชนิด RAM

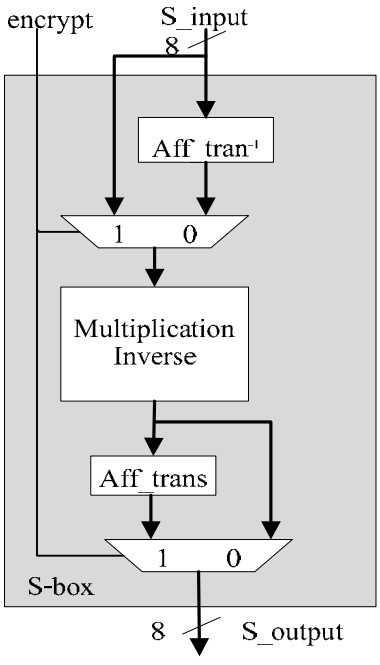
หน่วยความจำชนิด RAM ถูกสร้างขึ้นเพื่อใช้ในการเก็บคีย์ที่ถูกสร้างขึ้นใหม่ทุกรอบ ดังนั้นหน่วยความจำนี้จึงจะมีขนาด 128 × 11 บิต เพราะคีย์ตั้งต้นมีขนาด 128 บิตจำนวน 1 ชุด และจะเก็บคีย์ที่ได้จากการสร้างคีย์จำนวน 10 ชุด

3.1.2 การสร้างโมดูลจัดการสัญญาณนาฬิกา

สัญญาณนาฬิกาถูกนำเข้าสู่ระบบแล้วใช้วงจรหารความถี่ (Clock divider) โดยสัญญาณนาฬิกาหลัก (Global Clock) จะถูกนำไปใช้กับโมดูลการเข้ารหัส AES และทำการหารความถี่ด้วยวงจรหาร 2 เพื่อเป็นสัญญาณนาฬิกาให้กับตารางการเทียบค่า

3.2 การออกแบบวงจร S-box ด้วย $GF(2^4)^2$

วิทยานิพนธ์นี้ได้ทำการสร้างวงจร S-box จากสมการในงานวิจัย [14] แล้วลดรูปการใช้ XOR ซึ่งสามารถสร้างวงจร S-box ด้วย $GF(2^4)^2$ ได้ดังภาพประกอบที่ 3-4 ซึ่งประกอบด้วย 3 ฟังก์ชันหลักคือฟังก์ชันแอฟไฟน์ทรานสฟอร์ม (Affine Transform, Aff_trans) ฟังก์ชันอินเวิร์สแอฟไฟน์ทรานสฟอร์ม (Inverse Affine Transform, Aff_trans^{-1}) และฟังก์ชันมัลติพลิเคชันอินเวิร์ส (Multiplication Inverse) โครงสร้างนี้สามารถใช้งานได้ทั้งการเข้าและถอดรหัส โดยใช้สัญญาณ encrypt เป็นตัวควบคุม ถ้าเป็นการเข้ารหัสจะกำหนดให้ encrypt = 1 และการถอดรหัสจะกำหนดให้ encrypt = 0



ภาพประกอบ 3-4 องค์ประกอบภายในฟังก์ชัน S-box

3.2.1 ฟังก์ชันแอฟไฟน์ทรานสฟอร์ม

การประยุกต์ฟังก์ชันแอฟไฟน์ทรานสฟอร์ม (Aff_trans) จากสมการ (2-2) หน้า 12 ซึ่งจากเดิมใช้ XOR จำนวน 40 ตัว สามารถลดการใช้งาน XOR ลงโดยจับกลุ่มของข้อมูลเป็น 4 กลุ่ม ซึ่งสรุปผลการใช้งาน XOR ของวงจรแอฟไฟน์ทรานสฟอร์มพบว่าใช้ XOR ไปจำนวน 20 ตัว และ NOT อีก 4 ตัว โดยใช้สมการ (3-1)

$$q = aff\_trans(a) \tag{3-1}$$

โดยกำหนดให้ $a_A = a_0 \oplus a_1, a_B = a_2 \oplus a_3, a_C = a_4 \oplus a_5, a_D = a_6 \oplus a_7$

$$q_0 = \overline{a_0} \oplus a_C \oplus a_D$$
$$q_1 = \overline{a_5} \oplus a_A \oplus a_D$$
$$q_2 = a_2 \oplus a_A \oplus a_D$$
$$q_3 = a_7 \oplus a_A \oplus a_B$$
$$q_4 = a_4 \oplus a_A \oplus a_B$$
$$q_5 = \overline{a_1} \oplus a_B \oplus a_C$$
$$q_6 = \overline{a_6} \oplus a_B \oplus a_C$$
$$q_7 = a_3 \oplus a_C \oplus a_D$$

3.2.2 ฟังก์ชันอินเวิร์สแอฟไฟน์ทรานสฟอร์ม

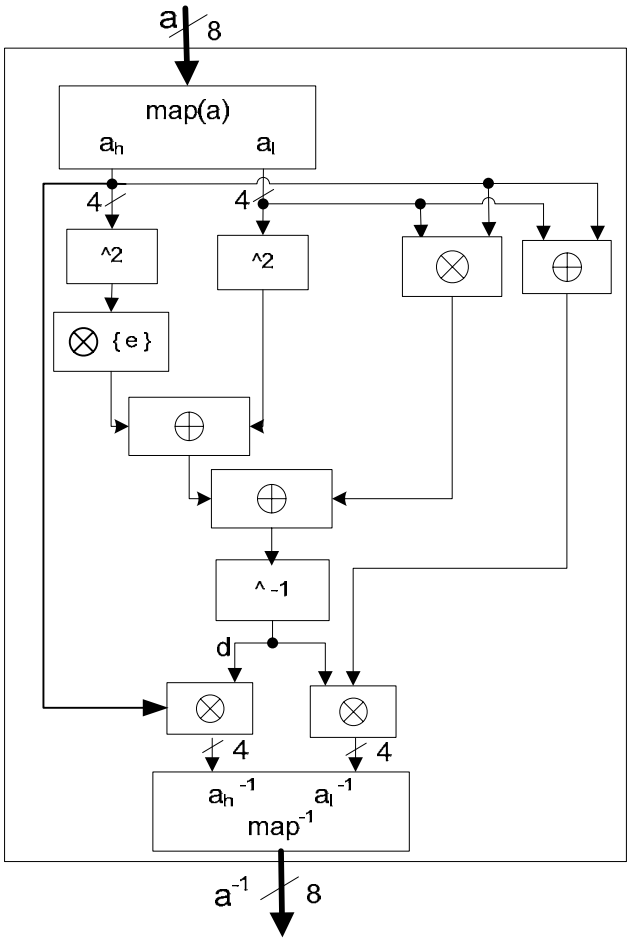
การประยุกต์ฟังก์ชันอินเวิร์สแอฟไฟน์ทรานสฟอร์ม (Aff_trans^{-1}) จากสมการ (2-5) หน้า 13 ซึ่งจะต้องใช้ XOR จำนวน 24 ตัว แต่สามารถลดการใช้งาน XOR โดยการจับคู่ค่าตัวแปร หรือค่าคงที่ ก็จะลด XOR เหลือเพียง 12 ตัว และใช้ NOT ไป 2 ตัวได้สมการการหาค่าอินเวิร์สแอฟไฟน์ทรานสฟอร์มดังสมการ (3-2)

$$q = aff\_trans^{-1}(a) \tag{3-2}$$

โดยกำหนดให้ $a_A = a_0 \oplus a_5, a_B = a_1 \oplus a_4, a_C = a_2 \oplus a_7, a_D = a_3 \oplus a_6$

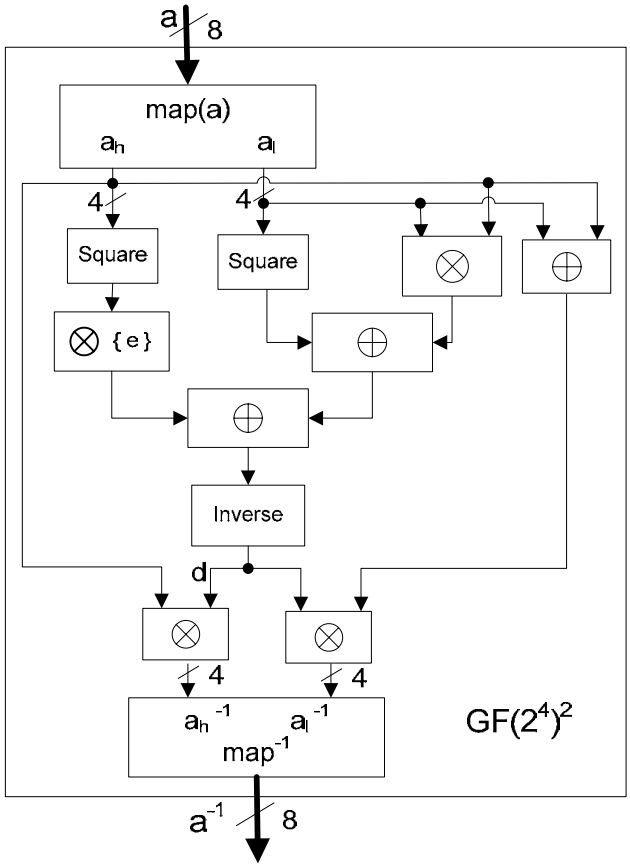
$$q_0 = \overline{a_5} \oplus a_C$$
$$q_1 = a_0 \oplus a_D$$
$$q_2 = \overline{a_7} \oplus a_B$$
$$q_3 = a_2 \oplus a_A$$
$$q_4 = a_1 \oplus a_D$$
$$q_5 = a_4 \oplus a_C$$
$$q_6 = a_3 \oplus a_A$$
$$q_7 = a_6 \oplus a_B$$

3.2.3 ฟังก์ชันมัลติพลิเคชันอินเวิร์ส



ภาพประกอบ 3-5 การทำงานของฟังก์ชันมัลติพลิเคชันอินเวิร์สใน S-box โดยใช้ $GF(2^4)^2$ ของ An ASIC Implementation of the AES SBoxes [14]

ภาพประกอบที่ 3-5 ฟังก์ชันมัลติพลิเคชันอินเวิร์ส [14] ซึ่งเป็นสถาปัตยกรรมแบบเดิมได้ถูกจัดสถาปัตยกรรมใหม่ โดยงานวิจัยนี้ได้สร้างฟังก์ชันมัลติพลิเคชันอินเวิร์สใหม่เป็นวงจรดังภาพประกอบที่ 3-6 โดยลดรูปการใช้ XOR ในขั้นตอนต่างๆ ซึ่งการประมวลผลจะเริ่มจากนำอินพุต a ขนาด 8 บิตมาผ่านฟังก์ชัน map(a) ทำให้สามารถแยกออกได้เป็น 2 พจน์คือ a_h และ a_l ที่มีขนาด 4 บิต แล้วผ่านการคำนวณยกกำลังสอง การคูณ การบวก การคูณด้วยค่าสัมประสิทธิ์คงที่ (e) และการ map^{-1} โดยที่แต่ละฟังก์ชันมีรายละเอียดการลดรูปของ XOR ในการทำงานดังต่อไปนี้



ภาพประกอบ 3-6 การทำงานของฟังก์ชันมัลติพลิเคชันอินเวิร์ส
ใน S-box โดยใช้ $GF(2^4)^2$ วิธีการลดรูปด้วย XOR

1. การบวก (Addition)

การบวกของโพลิโนเมียลสองเทอร์ม เป็นการนำสัมประสิทธิ์ที่ตรงกันมาบวกกัน ดังสมการ (3-3)

$$(a_h x + a_l) \oplus (b_h x + b_l) = (a_h \oplus b_h)x + (a_l + b_l) \tag{3-3}$$

2. การคูณ (Multiplication)

การคูณของโพลิโนเมียลสองเทอร์มเกี่ยวข้องกับการคูณอีลีเมนต์ใน $GF(2^4)$ โดยอีลีเมนต์ใน $GF(2^4)$ สมการ (3-4)

$$m_4(x) = x^4 + x + 1 \tag{3-4}$$

การคูณในกระบวนการ $GF(2^4)$ เมื่อ $a(x), b(x), q(x) \in GF(2^4)$ สมการที่ได้เป็นดังนี้

$$q(x) = a(x) \otimes b(x) = a(x) \cdot b(x) \ mod \ m_4(x) \tag{3-5}$$

โดยกำหนดให้ $a_A = a_0 \oplus a_3, a_B = a_2 \oplus a_3, a_C = a_1 \oplus a_2$

$$a_0 = (a_0 \wedge b_0) \oplus (a_3 \wedge b_1) \oplus (a_2 \wedge b_2) \oplus (a_1 \wedge b_3)$$
$$a_1 = (a_1 \wedge b_0) \oplus (a_A \wedge b_1) \oplus (a_B \wedge b_2) \oplus (a_C \wedge b_3)$$
$$a_2 = (a_2 \wedge b_0) \oplus (a_1 \wedge b_1) \oplus (a_A \wedge b_2) \oplus (a_B \wedge b_3)$$
$$a_3 = (a_3 \wedge b_0) \oplus (a_2 \wedge b_1) \oplus (a_1 \wedge b_2) \oplus (a_A \wedge b_3)$$

หากมีการคูณของอีลีเมนต์ใน GF(2^4) กับค่าคงที่ $\{e\}$ จะเลือกใช้สมการ (3-6) ซึ่งการคูณและการย้อนกลับใน GF(2^4) เป็นส่วนที่ซับซ้อนที่สุดของฟังก์ชัน

$$q = a \otimes \{e\} \tag{3-6}$$

โดยกำหนดให้ $a_A = a_0 \oplus a_3, a_B = a_2 \oplus a_3, a_C = a_1 \oplus a_2$

$$a_0 = a_2 \oplus a_1 \oplus a_3 \oplus a_3$$
$$a_1 = a_0 \oplus a_2 \oplus a_2$$
$$a_2 = a_A \oplus a_C \oplus a_2$$
$$a_3 = a_A \oplus a_B \oplus a_C$$

3. การยกกำลัง (Squaring)

การยกกำลังใน GF(2^4) เป็นการคูณแบบเฉพาะโดยใช้สมการ (3-7)

$$q(x) = a(x)^2 \ mod \ m_4(x), \ q(x), a(x) \in \text{GF}(2^4) \tag{3-7}$$

$$q_0 = a_0 \oplus a_2$$
$$q_1 = a_2$$
$$q_2 = a_1 \oplus a_3$$
$$q_3 = a_3$$

4. อินเวอร์ส (Inversion)

อินเวอร์ส a^{-1} (Inversion) ของอีลีเมนต์ $a \in$ GF(2^4) สามารถอธิบายโดยสมการ $a(x) \bullet a^{-1} mod \ m_4(x) = 1$ แสดงในสมการ (3-8)

$$q(x) = a(x)^{-1} \ mod \ m_4(x), \ q(x), a(x) \in \text{GF}(2^4) \tag{3-8}$$

โดยกำหนดให้ $a_{A01} = a_0 \hat{} a_1, a_{A02} = a_0 \hat{} a_2, a_{A03} = a_0 \hat{} a_3,$

$a_{B12} = a_1 \hat{} a_2, a_{B13} = a_1 \hat{} a_3, a_{C23} = a_2 \hat{} a_3,$

$a_{G1} = a_1 \oplus a_{C23} \oplus (a_{B12} \hat{} a_3)$

$q_0 = a_{G1} \oplus a_0 \oplus a_{A02} \oplus a_{B12} \oplus (a_0 \hat{} a_{B12})$

$q_1 = a_{A01} \oplus a_{A02} \oplus a_{B12} \oplus a_3 \oplus a_{B13} \oplus (a_1 \hat{} a_{A03})$

$q_2 = a_{A01} \oplus a_{C23} \oplus a_{A02} \oplus a_{A03} \oplus (a_2 \hat{} a_{A03})$

$q_3 = a_{G1} \oplus a_{A03} \oplus a_{B13} \oplus (a_2 \hat{} a_3)$

อินเวอร์ชันของสองเทอร์มโพลิโนเมียลเหมือนกับอินเวอร์ชันของ GF(2^8) เป็นการคูณของสอง เทอร์มโพลิโนเมียลด้วยอินเวอร์ชันผลลัพธ์ (Inverse yields) ซึ่งเป็นอีลีเมนต์ของฟิลด์: $(a_h x + a_l) \otimes (a'_h x + a'_l) = \{0\}x + \{1\}$ โดย $a_h, a_l, a'_h, a'_l \in$ GF(2^4) ซึ่งเกณฑ์ของอินเวอร์ชันเมื่อ $d = \left(\left(a_h^2 \otimes \{e\}\right) \oplus (a_h \otimes a_l) \oplus a_l^2\right)^{-1}$ เป็นดังนี้

$$(a_h x + a_l)^{-1} = (a'_h x + a'_l) = (a_h \otimes d)x + (a_h \oplus a_l) \otimes d \quad (3\text{-}9)$$

5. สมการแยกกลุ่มบิต

ฟิลด์จำกัด GF(2^8) มีความสมมาตรกับฟิลด์จำกัด GF(2^4) ในรูปแบบ bijection ของอีลีเมนต์ $a \in$ GF(2^8) มีรูปแบบโพลิโนเมียลสองเทอร์ม $a_h x + a_l$ ซึ่ง $a_h, a_l \in$ GF(2^4) ซึ่งถูกเรียกใช้ในชื่อของฟังก์ชัน map โดยมีสมการดังนี้คือ

$$a_h x + a_l = map(a), a_h, a_l \in \text{GF}(2^4), a \in \text{GF}(2^8) \quad (3\text{-}10)$$

โดยกำหนดให้ $a_A = a_4 \oplus a_6$

$a_{l0} = a_{h0} \oplus a_0$

$a_{l1} = a_1 \oplus a_2$

$a_{l2} = a_1 \oplus a_7$

$a_{l3} = a_2 \oplus a_4$

$a_{h0} = a_A \oplus a_5$

$a_{h1} = a_{l2} \oplus a_A$

$a_{h2} = a_{h3} \oplus a_2 \oplus a_3$

$a_{h3} = a_5 \oplus a_7$

6. สมการรวมกลุ่มบิต

ส่วนสมการย้อนกลับ (map inversion หรือ map^{-1}) เป็นการเปลี่ยนโพลิโนเมียลสองเทอร์ม $a_h x + a_l$ ซึ่ง $a_h, a_l \in GF(2^4)$ ให้กลับเป็นอีลีเมนต์ $a \in GF(2^8)$ นั้นจะใช้สมการ (3-11)

$$a = map^{-1}(a_h x + a_l), a \in GF(2^8), a_h, a_l \in GF(2^4) \quad (3\text{-}11)$$

โดยกำหนดให้ $a_A = a_{l1} \oplus a_{h3}, a_B = a_{h0} \oplus a_{h1}$

$$a_0 = a_{l0} \oplus a_{h0}$$
$$a_1 = a_B \oplus a_{h3}$$
$$a_2 = a_A \oplus a_B$$
$$a_3 = a_B \oplus a_{l1} \oplus a_{h2}$$
$$a_4 = a_A \oplus a_B \oplus a_{l3}$$
$$a_5 = a_B \oplus a_{l2}$$
$$a_6 = a_A \oplus a_{l2} \oplus a_{l3} \oplus a_{h0}$$
$$a_7 = a_B \oplus a_{l2} \oplus a_{h3}$$

3.3 โครงสร้าง S-box

เนื่องจากวงจร S-box เป็นวงจรหลักของกระบวนการเข้ารหัสแบบ AES ดังนั้นจึงมุ่งเน้นเทคนิควิธีการออกแบบ S-box และจะทำการวิเคราะห์ความเร็วกับกำลังงานของวงจร โดยโครงสร้างของวงจร S-box ในวิทยานิพนธ์นี้ ได้ถูกออกแบบไว้ 2 รูปแบบได้แก่ วงจร S-box แบบไปป์ไลน์ และวงจร S-box แบบขนานเพื่อทำการวิเคราะห์เปรียบเทียบวงจร S-box ทั้งสองโครงสร้างโดยมีรายละเอียดดังต่อไปนี้

3.3.1 การออกแบบ S-box แบบไปป์ไลน์ (Pipelining Architecture)

การออกแแบการทำงานของไปป์ไลน์นั้นมีเป้าหมายคือการทำให้แต่ละสเตทมีความสมดุลกันซึ่งแต่ละขั้นตอนต้องใช้เวลาในการคำนวณที่สมดุลกันหรือเท่ากัน แต่ก็เป็นเพียงแนวทางในอุดมคติซึ่งเวลาที่สมดุลนั้น คือ เวลาต่อหนึ่งคำสั่งในการทำงานที่ไม่เป็นไปป์ไลน์หารด้วยจำนวนสเตทไปป์ไลน์ที่ต้องการ แต่ในความเป็นจริงการทำให้เกิดความสมดุลในด้านเวลาการประมวลผลของสเตทในไปป์ไลน์นั้นค่อนข้างยาก สเตทการทำงานของไปป์ไลน์จะไม่มีความสมดุลกัน เนื่องจากไปป์ไลน์เกี่ยวข้องกับเวลาของ Overhead ในการคำนวณต่อหนึ่งคำสั่งจึงไม่ต้องมีค่าน้อยที่สุด [11]

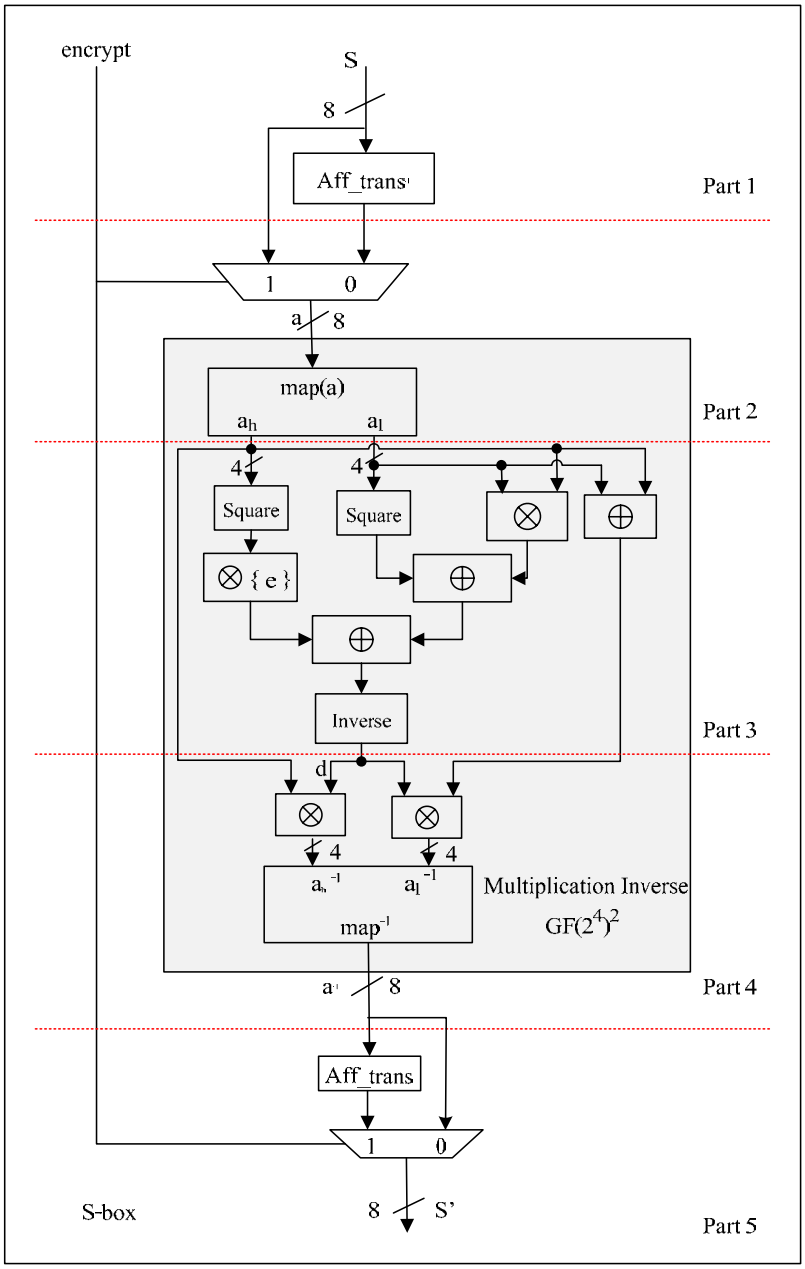
ข้อมูลอินพุตของการเข้ารหัสแบบ AES มีขนาด 128 บิต โดยแบ่งออกเป็นบล็อกย่อยๆ จำนวน 16 บล็อก ซึ่งแต่ละบล็อกจะเรียกว่า สเตท (State) ประกอบด้วยข้อมูลขนาด 8 บิต บรรจุอยู่ การออกแบบวงจร S-box แบบไปป์ไลน์นั้นมีจุดมุ่งหมายเพื่อให้สามารถใช้งานทรัพยากรได้อย่างเหมาะสม และเพิ่มความเร็วในการทำงานให้กับระบบ โดยวงจรที่ใช้ไปป์ไลน์จะใช้จำนวนของลอจิกเกตน้อยทำให้ช่วยประหยัดกำลังงาน (ลดตัวแปร C จากสมการการหาค่ากำลังงาน) ซึ่งผู้ออกแบบสามารถที่จะเพิ่มความเร็วของการทำงานของวงจรได้ด้วยวิธีการเพิ่มความถี่ของสัญญาณนาฬิกา ในวิทยานิพนธ์นี้ได้ออกแบบวงจร S-box แบบไปป์ไลน์ไว้ 2 แบบคือ

3.3.1.1 วงจร S-box แบบไปป์ไลน์ขนาด 8 บิต

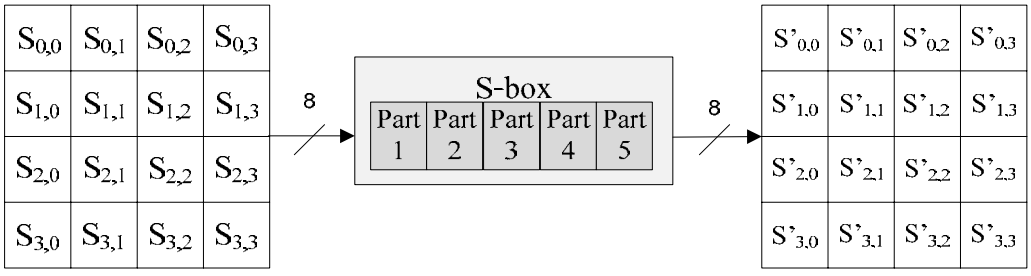
วงจร S-box ถูกแบ่งย่อยออกเป็น 5 ส่วนดังภาพประกอบที่ 3-7 โดยการทำงานจะรับอินพุตขนาด 128 บิต ผ่านฟังก์ชัน Subbyte และจะแบ่งอินพุตป้อนเข้าฟังก์ชันย่อยครั้งละ 8 บิต โดยป้อนข้อมูลจากสเตท $S_{0,0}$, $S_{0,1}$, $S_{0,2}$, $S_{0,3}$, $S_{1,0}$, $S_{1,1}$, $S_{1,2}$, $S_{1,3}$, …, $S_{3,2}$, $S_{3,3}$ ดังนั้นข้อมูลในแต่ละสเตทจะผ่านเข้าสู่ไปป์ไลน์ 5 ส่วนซึ่งจะรอจนครบ 5 รอบสัญญาณนาฬิกาจะได้ผลลัพธ์สเตทแรกออกมา และหลังจากนั้นในแต่ละรอบสัญญาณนาฬิกาจะมีผลลัพธ์ของสเตทถัดไปทยอยออกมาจนครบทั้ง 16 สเตทดังภาพประกอบที่ 3-8 ซึ่งจะใช้เวลาทั้งสิ้น 21 รอบสัญญาณนาฬิกา

3.3.1.2 วงจร S-box แบบไปป์ไลน์ขนาด 32 บิต

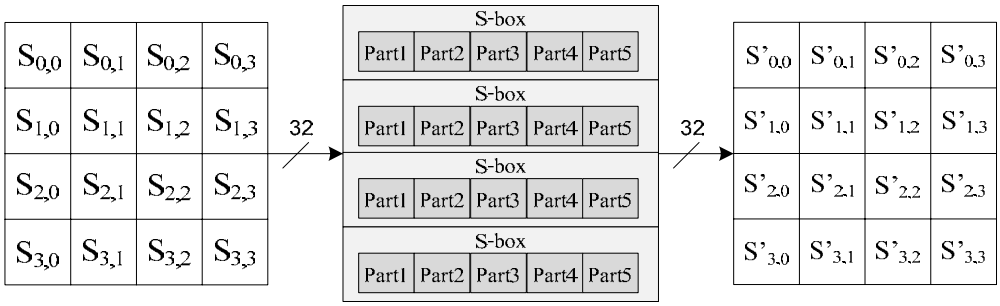
วงจร S-box แบบไปป์ไลน์ขนาด 32 บิตเป็นวงจร S-box ที่ถูกออกแบบเพื่อลดจำนวนของรอบสัญญาณนาฬิกา โดยจะนำข้อมูลเข้าสู่ไปป์ไลน์ครั้งละ 4 สเตทหรือทีละ 1 แถว ซึ่งจะใช้เวลาทำงานทั้งสิ้น 9 รอบสัญญาณนาฬิกา ดังภาพประกอบที่ 3-9



ภาพประกอบ 3-7 การทำงานของวงจร S-box แบบไปป์ไลน์



ภาพประกอบ 3-8 การทำงานของโครงสร้างการทำงานแบบไปป์ไลน์ขนาด 8 บิต



ภาพประกอบ 3-9 การทำงานของโครงสร้างการทำงานแบบไปป์ไลน์ขนาด 32 บิต

3.3.2 การออกแบบ S-box แบบขนาน (Parallel Architecture)

โครงสร้างวงจรแบบขนานสามารถทำงานได้เร็วกว่าแบบไปป์ไลน์ ดังนั้นงานวิจัยนี้จึงได้ออกแบบและสร้างวงจร S-box แบบขนานขึ้นเพื่อทำการเปรียบเทียบโดยออกแบบวงจรให้สามารถนำอินพุตขนาด 128 บิต เข้ามาประมวลผลฟังก์ชัน S-box ให้เสร็จภายในสัญญาณนาฬิกา 1 รอบสัญญาณนาฬิกา เพื่อให้สามารถลดความเร็วของสัญญาณนาฬิกาลงจึงประหยัดกำลังงาน (ลดความถี่ (α) จากสมการที่ (2-15) หน้า 24)

3.4 การออกแบบวงจร S-box ด้วยตารางการเทียบค่า

การสร้างวงจร S-box โดยใช้ตารางการเทียบค่าเป็นรูปแบบที่ใช้งานกันอย่างแพร่หลาย ซึ่งวิธีการทำงานแบบตารางการเทียบค่าได้กล่าวไว้ในบทที่ 2 เมื่อพิจารณาในส่วนของการใช้งานในระบบการเข้ารหัสข้อมูลแบบ AES นั้น จะสังเกตได้ว่าฟังก์ชัน S-box ถูกนำไปใช้งานถึงสองส่วน คือในส่วนการเข้ารหัสหลัก (core AES) และอีกส่วนคือการเพิ่มจำนวนคีย์ (Expaned Keys) ซึ่งในวิทยานิพนธ์นี้ได้นำตารางการเทียบค่าและกาลัวร์ฟิลด์มาใช้งานในการออกแบบวงจรการเข้ารหัสข้อมูลแบบ AES เพื่อหารูปแบบที่เหมาะสมต่อการใช้งานที่เน้นการใช้กำลังงานอย่างมีประสิทธิภาพ

บทที่ 4
การทดสอบและวิเคราะห์ประสิทธิภาพของวงจรการเข้ารหัสแบบ AES

ในบทนี้จะได้กล่าวถึงวิธีการทดสอบการทำงานและวิเคราะห์ประสิทธิภาพของวงจรการเข้ารหัสแบบ AES ซึ่งได้ออกแบบให้มีการประมวลผลโดยเลือกใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ และตารางการเทียบค่าใน S-box การทดสอบจะแบ่งออกเป็นการทดสอบความถูกต้องของในฟังก์ชัน S-box การทดสอบความถูกต้องของวงจรการเข้ารหัสแบบ AES ทั้งระบบ การวิเคราะห์และรายงานจำนวนทรัพยากรที่ใช้ในวงจร ความเร็ว รวมไปถึงการหาค่ากำลังงานที่สูญเสียในการเข้ารหัสแบบ AES

4.1 การทดสอบประสิทธิภาพของวงจร

ประสิทธิภาพในการทดสอบวงจรจะแสดงถึงความสามารถและคุณสมบัติของวงจรที่ได้ทำการออกแบบ ซึ่งโดยทั่วไปสามารถนำเสนอประสิทธิภาพของวงจรได้หลากหลายรูปแบบ ในวิทยานิพนธ์ฉบับนี้จะทำการวิเคราะห์ประสิทธิภาพของวงจร 3 รูปแบบได้แก่ การวิเคราะห์ประสิทธิภาพของปริมาณงานที่สามารถทำได้ต่อหนึ่งหน่วยเวลาหรือที่เรียกว่า Throughput การวิเคราะห์ประสิทธิภาพในการใช้ทรัพยากร (Resource Efficiency) และประสิทธิภาพในการใช้กำลังงาน (Power Efficiency)

4.1.1 ปริมาณงานต่อหนึ่งหน่วยเวลา (Throughput)

ในการวัดปริมาณงานต่อหนึ่งหน่วยเวลานั้นจะวัดจากจำนวนบิตของไซเฟอร์เท็ก (Ciphertext) ที่สามารถประมวลผลได้ต่อหนึ่งวินาที โดยการคำนวณนั้นจะใช้สมการดังนี้คือ

$$throughput = \frac{BS \times DB}{CC \times CP} \tag{4-1}$$

BS คือ จำนวนบิตข้อมูลต่อหนึ่งบล็อก

DB คือ บล็อกข้อมูลในเพลนเท็ก (Plaintext) ของการเข้ารหัสแบบ AES

CC คือ จำนวนของรอบสัญญาณนาฬิกา (clock cycle) ที่ถูกใช้ในการประมวลผลข้อมูล

และ CP คือ ระยะเวลาในหนึ่งรอบสัญญาณนาฬิกา

ซึ่งค่าปริมาณงานต่อหนึ่งหน่วยเวลายิ่งมากจะหมายถึงวงจรที่ออกแบบมีประสิทธิภาพดีสามารถทำได้รวดเร็ว

4.1.2 ประสิทธิภาพด้านการใช้งานทรัพยากร (Resource Efficiency)

ประสิทธิภาพด้านการใช้งานทรัพยากรนั้นวัดได้จากการใช้ทรัพยากรของวงจร นั่นคือจำนวนของลอจิกเกต (Slices) ที่นำไปใช้ประมวลผลการเข้ารหัสและถอดรหัสแบบ AES ซึ่งการวัดประสิทธิภาพด้านการใช้ทรัพยากรนั้น จะวัดในสัดส่วนที่เทียบกับปริมาณงานต่อหนึ่งหน่วยเวลากับจำนวนลอจิกเกตที่สูญเสียไปให้กับวงจร ซึ่งประสิทธิภาพด้านการใช้งานทรัพยากรที่ดีนั้นจะต้องมีค่ามากเพื่อให้วงจรมีประสิทธิภาพที่สูงด้วย

4.1.3 ประสิทธิภาพด้านการใช้กำลังงานของวงจร (Power efficiency)

ประสิทธิภาพด้านการใช้กำลังงานของวงจรสามารถคำนวณได้จากกำลังงานที่สูญเสียไปโดยวัดในรูปแบบของปริมาณงานต่อหนึ่งหน่วยเวลาเทียบกับกำลังงานที่ใช้ไปในวงจร เพื่อให้ประสิทธิภาพในด้านการใช้กำลังงานมีค่ามาก

ส่วนประกอบของวงจรนำมาซึ่งการออกแบบวงจรอย่างมีประสิทธิภาพ จนกระทั่งได้ระบบที่มีความน่าเชื่อถือนั้นส่งผลต่อค่ากำลังงานรวมในวงจร ในงานวิทยานิพนธ์นี้เลือกใช้ FPGA ในการทดสอบระบบดังนั้นการคำนวณกำลังงานรวมของ FPGA สามารถคำนวณได้จากสมการด้านล่างนี้

Total FPGA power = Device Static Power + Design Static Power + Dynamic Power (4-2)

การประมาณการค่ากำลังงานรวมของระบบนั้นเริ่มจากการจำลองอุณหภูมิ ค่าแรงดันไฟฟ้าและสภาพแวดล้อมโดยรอบ ซึ่งจะทำให้สามารถระบุค่าที่เหมาะสมสำหรับเซลล์ (Cells) ได้

Device Static Power เป็นการอ้างอิงถึงการรั่วไหลของกำลังงานของวงจรที่ไม่ได้มีการใช้งาน (Device Static) การรั่วไหลของกระแสภายในทรานซิสเตอร์ เมื่ออุปกรณ์ได้รับแรงดันไฟฟ้าและไม่ได้กำหนดค่าให้

Design Static Power แสดงถึงการใช้กำลังงานเพิ่มขึ้นเมื่ออุปกรณ์มีการกำหนดค่าให้และไม่มีการเปลี่ยนสถานะ กำลังงานแบบนี้คือ กำลังงานของอินพุต/เอาต์พุต (I/Os) การจัดการสัญญาณนาฬิกา ฯลฯ

Design Dynamic Power แสดงถึงการใช้กำลังงานที่เพิ่มขึ้นจากลอจิกที่ถูกใช้งานและเมื่ออุปกรณ์มีการเปลี่ยนสถานะหรือเรียกว่าเกิด switching activities

การทดสอบกำลังงานที่สูญเสียในการประมวลผลสามารถใช้โปรแกรม XPower Analyzer ซึ่งโปรแกรมนี้ จะคำนวณโดยใช้ข้อมูลของวงจรในการวิเคราะห์กำลังงาน โปรแกรมจะแบ่งชนิดของกำลังงานที่สูญเสียไปเป็น 2 ส่วนคือ

1. กำลังงานแบบไดนามิค (Dynamic Power) เป็นกำลังงานที่แปรผันโดยตรงกับการออกแบบ ซึ่งจะแสดงให้เห็นการเปลี่ยนแปลงของกำลังงาน เมื่อมีการเปลี่ยนแปลงของลอจิกที่ใช้งาน (switching user logic) และการเชื่อมต่อ (routing)

2. กำลังงานแบบควอสเซนต์ (Quiescent Power) หรือกำลังงานแบบสแตติก (Static Power) เป็นกำลังงานที่ได้ถูกใช้ไปโดยอุปกรณ์ซึ่งกำลังงานนี้จะสูญเสียไปกับการสร้างวงจรของลอจิกที่ใช้งานและไม่มีโหมดการสวิตซ์ชิงของลอจิก กำลังงานแบบสแตติกนี้ขึ้นกับการกำหนดสภาพแวดล้อมและองค์ประกอบของลอจิกในการออกแบบ

4.2 เครื่องมือในการทดสอบประสิทธิภาพของวงจร

4.2.1 อุปกรณ์ FPGA

อุปกรณ์ FPGA ที่เลือกใช้คือ Virtex–5 หมายเลข XC5VLX50 แพ็คเก็ตแบบมีจำนวน I/O ที่ ff–676 ที่ความเร็ว -3 โดยวงจรที่จะทดสอบบน FPGA นี้จะถูกออกแบบและพัฒนาด้วยภาษา VHDL โดยใช้โปรแกรม Simulation Xilinx ISE 10.1

4.2.2 โปรแกรม Simulation Xilinx ISE 10.1

โปรแกรม Simulation Xilinx ISE 10.1 เป็นโปรแกรมในการจำลองการทำงานของ FPGA ที่มีคุณภาพสูง เหมาะแก่การใช้งานในการศึกษาบนคอมพิวเตอร์ PC โปรแกรมนี้ถูกสร้างขึ้นโดยบริษัท Xilinx รองรับการออกแบบวงจรรวมและการทดสอบที่สมจริง ก่อนที่จะนำวงจรไปใช้งานจริงในรูปแบบของ ASIC โดยใช้ไฟล์ backanotation ที่ได้ไฟล์จากการสังเคราะห์ของโปรแกรม Xilinx

ในการทำงานวิทยานิพนธ์นี้นอกจากจะใช้โปรแกรม Xilinx ISE ในการสังเคราะห์วงจรเพื่อวัดประสิทธิภาพในการทำงานแล้วยังได้ใช้งานโปรแกรมย่อยใน Xilinx ISE เพิ่มเติมอีกด้วยนั่นคือ CORE Generator และ XPower Anlyzer การใช้งานโปรแกรมต่างๆ จะได้อธิบายในภาคผนวก ข และภาคผนวก ค ตามลำดับ

ต่อไปจะกล่าวถึงวิธีการทดสอบประสิทธิภาพของวงจรที่ได้ออกแบบนี้โดยจะทดสอบการทำงานของวงจรเรียงตามลำดับดังนี้ คือ

1. การทดสอบความถูกต้องของวงจรของ S-box ที่ใช้ $GF(2^4)^2$

2. การเปรียบเทียบประสิทธิภาพโครงสร้างการทำงานของ S-box โดยใช้กาลัวร์-ฟิลด์ $GF(2^4)^2$ ในแบบลูป ไปป์ไลน์และขนาน

3. การเปรียบเทียบประสิทธิภาพวงจร S-box แบบกาลัวร์ฟิลด์ $GF(2^4)^2$ วิธีการดั้งเดิมและวิธีการลดรูป XOR

4. การเปรียบเทียบประสิทธิภาพของวงจร S-box ระหว่างตารางการเทียบค่าและกาลัวร์ฟิลด์ $GF(2^4)^2$

5. การเปรียบเทียบประสิทธิภาพการทำงานของระบบการเข้ารหัสข้อมูลแบบ AES

โดยจะวิเคราะห์ประสิทธิภาพในการทำงานทั้งในเรื่องของปริมาณงานต่อหนึ่งหน่วยเวลา ประสิทธิภาพด้านการใช้งานทรัพยากร และประสิทธิภาพด้านการใช้กำลังงานของวงจร โดยรายละเอียดแสดงไว้ในหัวข้อที่ 4.1

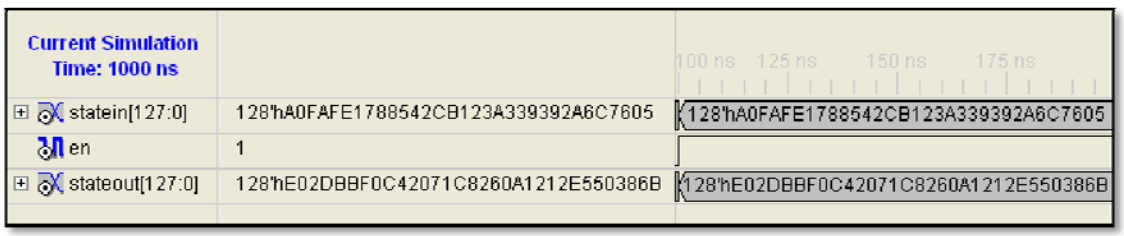
4.3 การทดสอบความถูกต้องของวงจรของ S-box ที่ใช้ $GF(2^4)^2$

ชุดทดสอบถูกสร้างขึ้นด้วยภาษา VHDL ในรูปแบบที่เรียกว่า Test Bench โดยกำหนดอินพุตข้อมูลให้กับวงจรที่ถูกสร้างขึ้น ผู้ออกแบบสามารถตรวจสอบผลการทำงานได้จากการจำลองการทำงานนี้ด้วยโปรแกรม Xilinx ISE ที่อยู่ในรูปแบบของ Waveforms เพื่อให้สะดวกการตรวจสอบผลการเข้ารหัสว่ามีความถูกต้องหรือไม่ โดยอินพุตที่ใช้ในการทดสอบยึดตามตัวอย่างที่แสดงในมาตรฐานของ FIP–197 และตรวจสอบการทำงานของอินพุตอื่นๆ ที่แตกต่างได้จากเว็บแอพพลิเคชันการเข้ารหัสแบบ AES ของมหาวิทยาลัยนิวเสาท์เว็ลล์ [15]

ผลการทดสอบความถูกต้อง แสดงให้เห็นจาก Waveforms ดังภาพประกอบ 4 – 1 โดยใช้อินพุตและเอาต์พุตที่มีขนาด 128 บิต แสดงเป็นเลขฐานสิบหกมีค่าดังนี้ คือ

ค่าอินพุต คือ “a0fafe1788542cb123a339392a6c7605”

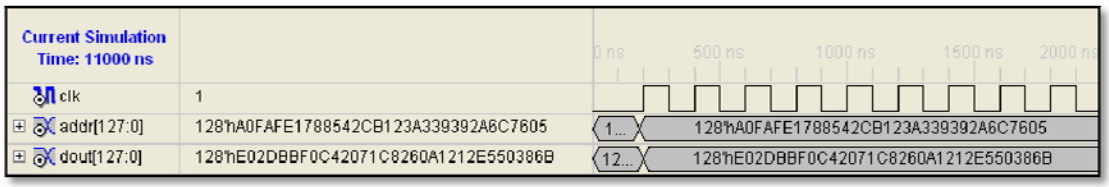
ค่าเอาต์พุต คือ “e02dbbf0c42071c8260a1212e550386b” (ตรงกับค่าที่ได้จาก web application http://www.unsw.adfa.edu.au)



ภาพประกอบ 4-1 ผลการจำลองการทำงานวงจร S-box ด้วยเทคนิคกาลัวร์ฟิลด์ $GF(2^4)^2$

วงจร S-box ที่ได้จากการสังเคราะห์วงจรเมื่อนำไปทดสอบการทำงานแสดงในภาพประกอบที่ 4-1 Waveforms แสดงอินพุตและเอาต์พุตของข้อมูลเมื่อผ่านกระบวนการทดสอบ

ความถูกต้องของการทำงานของ S-box ที่ได้ออกแบบว่าสามารถนำไปใช้งานต่อไปได้จริง นอกจากนี้ยังสามารถเปรียบเทียบกับการใช้เทคนิคตารางการเทียบค่า ซึ่งก็ได้ผลลัพธ์ที่ตรงกันดังแสดงในภาพประกอบ 4-2



ภาพประกอบ 4-2 ผลการจำลองการทำงานวงจร S-box ด้วยเทคนิคตารางการเทียบค่า

4.4 การเปรียบเทียบประสิทธิภาพโครงสร้างการทำงานของ S-box โดยใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ ในแบบลูป ไปป์ไลน์และขนาน

ในการสร้างวงจร S-box ด้วยการใช้ $GF(2^4)^2$ เป็นที่น่าสนใจเนื่องจากไม่จำเป็นที่จะต้องใช้งานหน่วยความจำที่จะส่งผลต่อการใช้กำลังงานและความเร็วในการประมวลผล เนื่องจากในการเข้ารหัสแบบ AES จำเป็นที่จะต้องใช้วงจร S-box เป็นจำนวนมาก ในส่วนนี้จึงได้ทำการวิเคราะห์เปรียบเทียบประสิทธิภาพของวงจร S-box โดยใช้ $GF(2^4)^2$ บนโครงสร้างการทำงานแบบต่างๆ เช่น โครงสร้างการทำงานของ S-box แบบลูป แบบไปป์ไลน์และแบบขนาน ซึ่งผลการสังเคราะห์วงจร S-box สามารถแสดงทรัพยากรที่ใช้ รวมถึงค่าหน่วงเวลา (latency) และกำลังงานไว้ในตารางที่ 4-1

ตารางที่ 4-1 เปรียบเทียบวงจร S-box แบบกาลัวร์ฟิลด์ $GF(2^4)^2$ ในรูปแบบโครงสร้างต่างๆ

	แบบลูป	**แบบไปป์ไลน์ 8 บิต**	**แบบไปป์ไลน์ 32 บิต**	**แบบขนาน**
Slice Logic Utilization:				
Number of Slice Registers:	148 out of 19200 0%	188 out of 19200 0%	354 out of 19200 0%	-
Number of Slice LUTs:	124 out of 19200 0%	113 out of 19200 0%	313 out of 19200 1%	944 out of 19200 4%
Number used as Logic:	124 out of 19200 0%	113 out of 19200 0%	313 out of 19200 1%	944 out of 19200 4%
รวมจำนวนแผ่นลอจิกที่ใช้งาน	272	301	667	944
Slice Logic Distribution:				
Number of fully used LUT-FF pairs:	19 out of 253 7%	51 out of 250 20%	193 out of 454 40%	0 out of 944 0%
Number of unique control sets:	2	2	2	0
IO Utilization:				
Number of IOs:	258	258	258	257
Specific Feature Utilization:				
Number of BUFG/BUFGCTRLs:	1 out of 32 3%	1 out of 32 3%	1 out of 32 3%	-

ตารางที่ 4-1 เปรียบเทียบวงจร S-box แบบกาลัวร์ฟิลด์ $GF(2^4)^2$ ในรูปแบบโครงสร้างต่างๆ (ต่อ)

	แบบลูป	**แบบไปป์ไลน์ 8 บิต**	**แบบไปป์ไลน์ 32 บิต**	**แบบขนาน**
Timing Summary:				
Minimum period:	5.094 ns	3.342 ns	3.346 ns	No path found
(Maximum Frequency)	96.321 MHz	99.222 MHz	298.846 MHz	No path found
Minimum input arrival time before clock:	5.543 ns	2.284 ns	1.723 ns	No path found
Maximum output required time after clock:	2.779 ns	2.779 ns	2.779 ns	10.849 ns
Maximum combinational path delay:	No path found	No path found	No path found	
Supply Power:				
Dynamic Power	5 mW	5 mW	7 mW	0.02 mW
Quiescent Power	378 mW	378 mW	378 mW	378 mW
Total Power	383 mW	383 mW	385 mW	378 mW

จำนวนของรอบสัญญาณนาฬิกาที่ใช้ในการประมวลผลนั้น แบบขนานใช้ไป 1 รอบสัญญาณนาฬิกาซึ่งใช้น้อยที่สุด ตามมาด้วยโครงสร้างแบบไปป์ไลน์ขนาด 32 บิตใช้ไป 11 รอบสัญญาณนาฬิกา ส่วนโครงสร้างแบบไปป์ไลน์แบบ 8 บิตใช้ไป 23 รอบสัญญาณนาฬิกาและท้ายสุดโครงสร้างแบบลูปใช้ไป 18 รอบสัญญาณนาฬิกา แต่โดยสรุประยะเวลาในการทำงานของกระบวนการ S-box นั้นสามารถคำนวณได้จากสมการดังต่อไปนี้

เวลาในการทำงานทั้งหมด = รอบสัญญาณนาฬิกา × T

โดยที่ T คือคาบสัญญาณนาฬิกา ดังนั้นโครงสร้างการทำงานแบบไปป์ไลน์ขนาด 8 บิต หนึ่งรอบสัญญาณนาฬิกาใช้เวลา 3.342 ns โครงสร้างการทำงานแบบไปป์ไลน์ขนาด 32 บิต หนึ่งรอบสัญญาณนาฬิกาใช้เวลา 3.346 ns โครงสร้างการทำงานแบบลูปหนึ่งรอบสัญญาณนาฬิกาใช้เวลา 5.094 ns และโครงสร้างการทำงานแบบขนานหนึ่งรอบสัญญาณนาฬิกาใช้เวลา 10.849 ns

สรุปได้ว่าวงจร S-box ที่ใช้ $GF(2^4)^2$ ด้วยโครงสร้างแบบขนานจะให้ผลการทำงานที่เร็วที่สุด รองลงมาเป็นโครงสร้างแบบไปป์ไลน์ขนาด 32 บิต โครงสร้างแบบไปป์ไลน์ขนาด 8 บิต และโครงสร้างแบบลูปตามลำดับ โดยแสดงการเปรียบเทียบความเร็วของการทำงานของวงจร S-box ได้ดังตารางที่ 4-2

ตารางที่ 4-2 เวลาในการทำงานของวงจร S-box ด้วย $GF(2^4)^2$ ในรูปแบบต่างๆ

รูปแบบของวงจร S-box ด้วย $GF(2^4)^2$	**เวลาในการทำงาน (ns)**	**การเปรียบเทียบกับแบบ loop sequence**
แบบขนาน	10.849	8.45 เท่า
แบบไปป์ไลน์ 32 บิต	36.806	2.49 เท่า
แบบไปป์ไลน์ 8 บิต	76.866	1.19 เท่า
แบบลูป	91.710	1.00 เท่า

เมื่อทำการวิเคราะห์ทรัพยากรที่ใช้ในวงจร S-box ในโครงสร้างแบบต่างๆ พบว่าจำนวนทรัพยากรที่ใช้งานบน FPGA หรือเรียกว่าแผ่นลอจิกซึ่งแบ่งออกเป็น 2 รูปแบบคือ

1) Slice utilization ซึ่งแสดงให้เห็นถึงการใช้ลอจิกเกตทั่วไปและรีจิสเตอร์
2) Slice distribution แสดงให้เห็นถึงการใช้งาน Flip Flop

ผลการสังเคราะห์วงจร S-box ที่ได้พบว่า โครงสร้างแบบลูปมีการใช้งานลอจิกเกตทั่วไป 124 ตัว รีจิสเตอร์ 148 ตัว และมี Flip Flop อีก 19 ตัว โครงสร้างแบบไปป์ไลน์ขนาด 8 บิตมี

การใช้งานลอจิกเกตทั่วไป รีจิสเตอร์และ Flip Flop เป็นจำนวน 113, 188 และ 51 ตัวตามลำดับ ขณะที่วงจรที่ใช้โครงสร้างแบบไปป์ไลน์ขนาด 32 บิตมีการใช้งานลอจิกเกตทั่วไป 313 ตัว รีจิสเตอร์ 354 ตัว และมี Flip Flop อีก 193 ตัว และท้ายสุดแสดงโครงสร้างแบบขนานใช้ลอจิกทั่วไปเพียง 944 ตัวเท่านั้น สามารถสรุปทรัพยากรที่ใช้งานในแต่ละรูปแบบของโครงสร้างวงจร S-box ดังตารางที่ 4-3

ตารางที่ 4-3 การใช้ทรัพยากรในโครงสร้างของวงจร S-box ด้วย $GF(2^4)^2$ แบบต่างๆ

รูปแบบของวงจร S-box ด้วย $GF(2^4)^2$	**Logic gate**		**Register**		**F/F**	
	จำนวน (ตัว)	**เปรียบเทียบกับแบบ loop sequence**	**จำนวน (ตัว)**	**เปรียบเทียบกับแบบ loop sequence**	**จำนวน (ตัว)**	**เปรียบเทียบกับแบบ loop sequence**
แบบขนาน	944	7.6	-	-	-	-
แบบไปป์ไลน์ 32 บิต	313	2.5	354	2.4	193	10.1
แบบไปป์ไลน์ 8 บิต	113	0.9	188	1.3	51	2.7
แบบลูป	124	1.0	148	1.0	19	1.0

เมื่อวิเคราะห์ข้อมูลที่ได้ แม้การใช้งานรูปแบบขนานจะใช้ปริมาณลอจิกเกตที่มากถึง 7.6 เท่าเมื่อเทียบกับการทำงานแบบลูป แต่ไม่มีการใช้งานส่วนของรีจิสเตอร์และ Flip Flop ซึ่งแบบลูปนอกจากจะใช้ลอจิกเกตแล้วยังใช้รีจิสเตอร์และ Flip Flop ด้วย หากพิจารณาการใช้ทรัพยากรการทำงานแบบลูปมีประสิทธิภาพดีที่สุดเมื่อเทียบกับแบบอื่นๆ แต่เมื่อพิจารณาด้านความเร็วแล้วพบว่าการทำงานแบบขนานดีที่สุด โดยพิจารณาได้จากปริมาณงานที่ได้ต่อหนึ่งหน่วยเวลา ดังแสดงในตารางที่ 4-4

ตัวอย่างวิธีการคิดค่าปริมาณงานต่อหนึ่งหน่วยเวลาในวงจร S-box รูปแบบขนาน คำนวณได้จากสมการที่ (4-1) ดังนี้คือ

$$throughput = \frac{BS \times DB}{CC \times CP}$$

$$= \frac{8 \text{ บิตข้อมูลต่อหนึ่งบล็อก} \times 16 \text{ บล็อกข้อมูล}}{1 \text{ รอบสัญญาณนาฬิกา} \times 10.849 \text{ ns}}$$

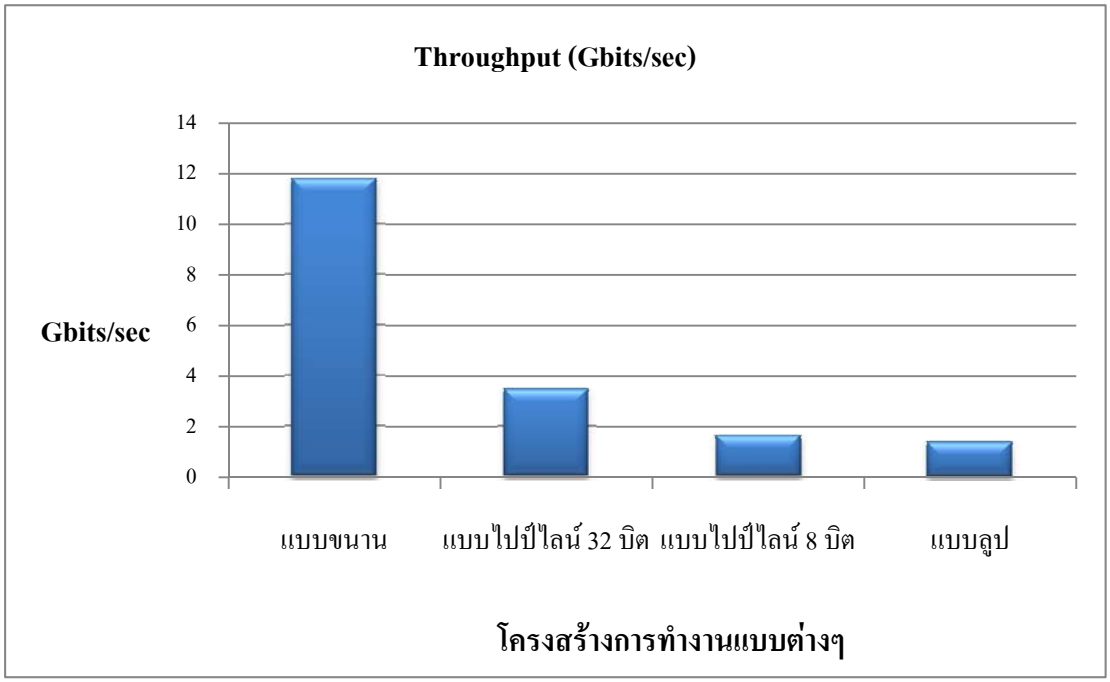
$$= \frac{128}{1 \text{ รอบสัญญาณนาฬิกา} \times 10.849 \text{ ns}}$$

$$= 11.7983 \text{ Gbps}$$

ตารางที่ 4-4 เปรียบเทียบประสิทธิภาพของวงจร S-box ที่ใช้
เทคนิคกาลัวร์ฟิลด์ $GF(2^4)^2$ ในโครงสร้างรูปแบบต่างๆ

รูปแบบของวงจร S-box ด้วย $GF(2^4)^2$	**Throughput (Gbits/sec)**	**Area (slice)**	**Throughput/Area (Mbits/sec/slice)**
แบบขนาน	11.798	944	12.498
แบบไปป์ไลน์ขนาด 32 บิต	3.477	667	5.213
แบบไปป์ไลน์ขนาด 8 บิต	1.665	301	5.532
แบบลูป	1.395	272	5.131

เมื่อวิเคราะห์ประสิทธิภาพในการใช้งานแล้วพบว่าโครงสร้างแบบต่างๆ ในการออกแบบวงจร S-box โดยใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ พบว่าปริมาณงานต่อหน่วยเวลาของโครงสร้างการทำงานแบบขนานเป็นวิธีการที่ให้ประสิทธิภาพดีที่สุด ซึ่งให้ประสิทธิภาพได้มากถึง 11.798 Gbps



ภาพประกอบ 4-3 กราฟเปรียบเทียบประสิทธิภาพการทำงานของ S-box
โดยใช้เทคนิคกาลัวร์ฟิลด์ $GF(2^4)^2$ ในโครงสร้างแบบต่างๆ

ตารางที่ 4-5 เปรียบเทียบการใช้กำลังงานของเทคนิคการออกแบบวงจร S-box ด้วยกาลัวร์ฟิลด์ $GF(2^4)^2$ ในโครงสร้างรูปแบบต่างๆ

รูปแบบของวงจร S-box ด้วย $GF(2^4)^2$	**Supply Power (mW)**		
	Dynamic	**Quiescent**	**Total**
แบบขนาน	0.02	378	378
แบบไปป์ไลน์ขนาด 32 บิต	7	378	385
แบบไปป์ไลน์ขนาด 8 บิต	5	378	383
แบบลูป	5	378	383

จากตารางที่ 4-5 แสดงผลการทดลองเพื่อหาการใช้กำลังงานของวงจร S-box โดยเปรียบเทียบใน 4 กรณี ดังที่ได้กล่าวมาข้างต้นแล้ว พบว่า การใช้โครงสร้างการทำงานแบบขนานให้ผลของการใช้กำลังงานน้อยที่สุด 378 มิลลิวัตต์ การทำงานของส่วนอื่นๆ ไม่ว่าจะเป็นโครงสร้างการทำงานแบบลูปและโครงสร้างการทำงานแบบไปป์ไลน์จะใช้กำลังงานมากขึ้นเป็นลำดับ

โครงสร้างการทำงานแบบขนานมีข้อดีทั้งในส่วนของเวลาที่ใช้ในการประมวลผลที่น้อยกว่าโครงสร้างแบบอื่น และให้ปริมาณงานต่อหน่วยเวลาที่มากกว่าโครงสร้างแบบอื่นๆ แม้ว่าจะใช้แผ่นลอจิกที่มากกว่าก็ตาม แต่ด้วยเปอร์เซ็นต์ที่ไม่มากจึงได้เลือกโครงสร้างแบบขนานในการสร้างวงจร S-box ที่ใช้เทคนิคกาลัวร์ฟิลด์ $GF(2^4)^2$ ในการเข้ารหัสแบบ AES

4.5 การเปรียบเทียบประสิทธิภาพวงจร S-box แบบกาลัวร์ฟิลด์ $GF(2^4)^2$ วิธีการดั้งเดิมและวิธีการลดรูป XOR

การจัดรูปแบบการทำงานของฟังก์ชันมัลติพลิเคชันอินเวิร์สใน S-box โดยใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ นั้น วิธีการลดรูป XOR ใช้เวลาไปทั้งสิ้น 10.936 ns ซึ่งวิธีการดั้งเดิมที่สร้างโดยใช้ ASIC [14] ดีกว่า เพราะใช้เวลาไปเพียง 10.849 ns แต่เมื่อพิจารณาการใช้งานทรัพยากรนั้น พบว่า กระบวนการ S-box โดยใช้ $GF(2^4)^2$ ใช้วิธีการลดรูป XOR มีการใช้ลอจิกเกตจำนวนน้อยกว่า 4.91% รวมถึงด้านกำลังงานที่ใช้น้อยกว่า 19.04% ดังแสดงในตารางที่ 4-6

ตารางที่ 4-6 เปรียบเทียบวงจร S-box แบบกาลัวร์ฟิลด์ $GF(2^4)^2$ วิธีการดั้งเดิมและวิธีการลดรูป XOR

	S-box $GF(2^4)^2$ วิธีการดั้งเดิม [14] (ภาพประกอบ 3-5)	**S-box $GF(2^4)^2$ วิธีการลดรูป XOR (ภาพประกอบ 3-6)**
Slice Logic Utilization:		
Number of Slice LUTs:	976 out of 19200 5%	928 out of 19200 4%
IO Utilization:		
Number of IOs:	257	257
Number of bonded IOBs:	257 out of 400 64%	257 out of 400 64%
Timing Summary:		
Minimum period:	No path found	No path found
Minimum input arrival time before clock:	No path found	No path found
Maximum output required time after clock:	No path found	No path found
Maximum combinational path delay:	10.849 ns	10.936 ns
Supply Power:		
Dynamic Power	432 mW	278 mW
Quiescent Power	382 mW	381 mW
Total Power	814 mW	659 mW

4.6 การเปรียบเทียบประสิทธิภาพของวงจร S-box ระหว่างตารางการเทียบค่าและกาลัวร์ฟิลด์ $GF(2^4)^2$

จากการสร้างวงจร S-box ด้วยตารางการเทียบค่าและกาลัวร์ฟิลด์ $GF(2^4)^2$ ทั้งสองรูปแบบถูกนำไปสังเคราะห์วงจรบนเทคโนโลยี FPGA ซึ่งได้ผลการสังเคราะห์วงจรดังแสดงไว้ในตารางที่ 4-7 สามารถสรุปได้ว่าวงจร S-box ที่สร้างจากโดยใช้กระบวนการทางคณิตศาสตร์กาลัวร์-ฟิลด์ $GF(2^4)^2$ ด้วยวิธีการลดรูป XOR นั้นใช้จำนวนลอจิกเกตที่สูงกว่าวงจร S-box แบบใช้ตารางการเทียบค่า เนื่องจากแผ่นลอจิกเกตถูกนำไปใช้ในการคำนวณค่าแทนการใช้หน่วยความจำ RAM จึงทำให้ผลของการใช้งานแผ่นลอจิกในการสังเคราะห์วงจรสูงกว่านั่นเอง แต่ใช้กำลังงานรวมน้อยกว่าการสังเคราะห์วงจรแบบตารางการเทียบค่า

สำหรับเวลาในการประมวลผลพบว่าวงจร S-box ที่สร้างจาก $GF(2^4)^2$ ซึ่งเป็นลักษณะวงจรแบบ combinational มีค่าหน่วงเวลาประมาณ 10.85 ns ในขณะที่วงจร S-box ที่สร้างแบบตารางการเทียบค่าจะใช้เวลาทำงานตามสัญญาณนาฬิกาเนื่องจากเป็นโครงสร้างแบบหน่วยความจำ พบว่าสามารถทำได้ที่ความถี่สูงสุด 414.216 MHz และมีค่าหน่วงเวลาประมาณ 2.414 ns ในขณะที่การใช้กำลังงานของ S-box ทั้ง $GF(2^4)^2$ และตารางการเทียบค่าแทบไม่มีความแตกต่างซึ่งอยู่กันที่ประมาณ 378 และ 402 mW ตามลำดับ

ตารางที่ 4-7 เปรียบเทียบวงจร S-box ที่ได้จากการสังเคราะห์ระหว่างเทคนิคการใช้ตารางการเทียบค่า กับ กาลัวร์ฟิลด์ $GF(2^4)^2$

	Galois Fields GF $(2^4)^2$	**ตารางการเทียบค่า (LUT)**
Slice Logic Utilization:		
Number of Slice Registers:	-	384 out of 19200 2%
Number used as Logic:	944 out of 19200 4%	129 out of 19200 0%
IO Utilization:		
Number of IOs:	257	258
Number of bonded IOBs:	257 out of 400 64%	257 out of 400 64%
Specific Feature Utilization:		
Number of Block RAM/FIFO:	-	16 out of 32 50%
Number using Block RAM only:	-	16
Number of BUFG/BUFGCTRLs:	-	1 out of 32 3%
Timing Summary:		
Minimum period:	-	2.414 ns
(Maximum Frequency)	-	(414.216MHz)
Minimum input arrival time before clock:	No path found	1.781ns
Maximum output required time after clock:	No path found	2.775ns
Maximum combinational path delay:	10.849 ns	No path found
Supply Power		
Dynamic Power	0.02 mW	24 mW
Quiescent Power	378 mW	378 mW
Total Power	378 mW	402 mW

4.7 เปรียบเทียบประสิทธิภาพการทำงานของระบบการเข้ารหัสข้อมูลแบบ AES

การทำงานในส่วนนี้จะอธิบายให้เห็นถึงผลลัพธ์จากการแยกส่วนการทำงานเพื่อวิเคราะห์รูปแบบของการออกแบบเพื่อให้ได้ประสิทธิภาพดีที่สุด โดยจะแบ่งได้ดังนี้

1. จำลองการทำงานของการเข้ารหัสหลักแบบ AES
2. จำลองการทำงานของการสร้างคีย์
3. การรวมระบบการเข้ารหัสแบบ AES

4.7.1 จำลองการทำงานของการเข้ารหัสหลักแบบ AES

การเข้ารหัสข้อมูลแบบ AES นี้จะประกอบด้วย 4 ส่วนหลักๆ คือ SubByte ShiftRow MixColumn และ AddRoundKey การทำงานได้อธิบายไว้ในบทที่ 3 และผลการทดสอบประสิทธิภาพระบบการเข้ารหัสหลักแบบ AES โดยแสดงผลการทดสอบดังตารางที่ 4–8

การทำงานของฟังก์ชันการเข้ารหัสหลักนั้นจากทฤษฎีที่ได้กล่าวไว้ข้างต้น มีการทำงานจำนวน 10 รอบ และจากผลการทดสอบข้างต้น ได้สรุปการเลือกใช้การทำงานของฟังก์ชัน S-box โดยใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ บนโครงสร้างแบบขนาน เนื่องจากมีประสิทธิภาพในการทำงานดีที่สุด และเมื่อนำ S-box จากวิธีการดังกล่าวมาใช้กับกระบวนการเข้ารหัสหลัก พบว่าการใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ นั้น ใช้ทรัพยากรมากกว่าการใช้ตารางการเทียบค่าหลายเท่าตัว แต่ใช้กำลังงานน้อยกว่าการใช้ตารางการเทียบค่า

ตารางที่ 4-8 เปรียบเทียบการใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ และตารางการเทียบค่า
บนการเข้ารหัสแบบ AES

	กาลัวร์ฟิลด์ $GF(2^4)^2$	**ตารางการเทียบค่า (LUT)**
Slice Logic Utilization:		
Number of Slice Registers:	384 out of 19200 1%	512 out of 19200 2%
Number used as Logic:	3213 out of 19200 16%	893 out of 19200 4%
รวมจำนวนแผ่นลอจิกที่ใช้งาน	3597	1405
Slice Logic Distribution:		
Number of LUT Flip Flop pairs used:	3213	1003
Number of fully used LUT-FF pairs:	384 out of 3213 11%	402 out of 1003 40%
Number of unique control sets:	2	132
IO Utilization:		
Number of IOs:	391	258
Number of bonded IOBs:	390 out of 400 97%	258 out of 400 64%
Specific Feature Utilization:		
Number of Block RAM/FIFO:	1 out of 32 3%	8 out of 32 25%
Number of BUFG/BUFGCTRLs:		6 out of 32 18%
Timing Summary:		
Minimum period:	7.099 ns	No path found
(Maximum Frequency)	(140.855 MHz)	
Minimum input arrival time before clock:	7.609 ns	2.483 ns
Maximum output required time after clock:	2.775 ns	2.989 ns
Maximum combinational path delay:	No path found	No path found
Supply Power:		
Dynamic Power	11 mW	15 mW
Quiescent Power	378 mW	378 mW
Total Power	389 mW	393 mW

4.7.2 จำลองการทำงานของการเพิ่มจำนวนคีย์

ผลการทดสอบประสิทธิภาพการเพิ่มจำนวนคีย์เพื่อใช้ในการเข้ารหัสข้อมูลแบบ AES นี้ประกอบด้วย 4 ส่วนในการทำงานได้แก่ Rotage Word, Sub-Word, Round Constant (Rcon) และ Word Column (W_column) ซึ่งได้อธิบายไว้ในบทที่ 3 แต่เนื่องจากงานวิทยานิพนธ์นี้ให้ความสนใจกับวงจร S-box ซึ่งภายในกระบวนการสร้างคีย์ได้นำ S-box มาใช้งานในส่วนของ SubWord ซึ่งภายในมีการทำงานของ S-box จำนวน 4 ชุด โดยใช้เทคนิค 2 รูปแบบ คือ แบบใช้ตารางการเทียบค่าและแบบใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ ผลการทดสอบแสดงได้ดังตารางที่ 4-9

การสร้างคีย์ใช้ในการคำนวณทั้งหมด 10 รอบเช่นเดียวกันกับการเข้ารหัสหลักแบบ AES แต่ฟังก์ชัน S-box ที่ใช้นั้นมีเพียง 4 ชุดหรือเท่ากับ 32 บิต ซึ่งการทำงานของการเพิ่มจำนวนคีย์จะใช้ทรัพยากรน้อยกว่าการเข้ารหัส AES ดังนั้นประสิทธิภาพที่ได้จากการทดสอบ พบว่าการใช้เทคนิคกาลัวร์ฟิลด์ $GF(2^4)^2$ ยังคงไม่ใช้งาน BRAM แต่เทคนิคการใช้ตารางการเทียบค่าใช้เวลาน้อยกว่าและยังใช้กำลังงานน้อยกว่าเช่นกัน ดังนั้นการทำงานโดยใช้เทคนิคกาลัวร์ฟิลด์ $GF(2^4)^2$ เหมาะกับวงจรที่ซับซ้อนและต้องการการประมวลผลสูง

ตารางที่ 4-9 เปรียบเทียบเทคนิคการใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ และ
ตารางการเทียบค่าบนวงจรเพิ่มจำนวนคีย์

	กาลัวร์ฟิลด์ $GF(2^4)^2$	**ตารางการเทียบค่า (LUT)**
Slice Logic Utilization:		
Number of Slice Registers:	288 out of 19200 1%	162 out of 19200 0%
Number used as Logic:	1174 out of 19200 6%	395 out of 19200 2%
รวมจำนวนแผ่นลอจิกที่ใช้งาน	1462	557
Slice Logic Distribution:		
Number of LUT Flip Flop pairs used:	1174	397
Number of fully used LUT-FF pairs:	288 out of 1174 24%	160 out of 397 40%
Number of unique control sets:	1	1
IO Utilization:		
Number of IOs:	261	265
Number of bonded IOBs:	261 out of 400 65%	262 out of 400 65%
Specific Feature Utilization:		
Number of Block RAM/FIFO:	-	2 out of 32 6%
Number of BUFG/BUFGCTRLs:	1 out of 32 3%	2 out of 32 6%
Timing Summary:		
Minimum period:	5.626 ns	2.756 ns
(Maximum Frequency)	(177.756 MHz)	(362.856 MHz)
Minimum input arrival time before clock:	2.717 ns	1.917 ns
Maximum output required time after clock:	5.365 ns	4.987 ns
Maximum combinational path delay:	No path found	No path found
Supply Power:		
Dynamic Power	6 mW	2 mW
Quiescent Power	378 mW	378 mW
Total Power	384 mW	380 mW

4.7.3 จำลองการทำงานของระบบการเข้ารหัสแบบ AES

การเข้ารหัสแบบ AES ดังที่ได้กล่าวไว้ว่ามีสองส่วนที่ใช้งาน S-box คือ ในส่วนของการเข้ารหัสหลักซึ่งอยู่ในส่วนของ Subbyte และส่วนของการเพิ่มจำนวนคีย์ จึงได้ระบุชื่อเพื่อให้เข้าใจตรงกันดังนี้

1. รูปแบบการทำงานแบบ LL (LUTs & LUTs)

เป็นรูปแบบการทำงานที่เลือกใช้ตารางการเทียบค่าทั้งในส่วนของการเพิ่มจำนวนคีย์ (Expaned Keys) และอีกส่วนคือวงจรเข้ารหัสแบบ AES (Core AES)

2. รูปแบบการทำงานแบบ LG (LUTs & $GF(2^4)^2$

เป็นรูปแบบการทำงานที่เลือกใช้ตารางการเทียบค่าในส่วนของการเพิ่มจำนวนคีย์ (Expaned Keys) และรูปแบบการทำงานที่เลือกใช้ $GF(2^4)^2$ ในส่วนของวงจรเข้ารหัสแบบ AES (Core AES)

3. รูปแบบการทำงานแบบ GL ($GF(2^4)^2$ & LUTs)

เป็นรูปแบบการทำงานที่เลือกใช้ $GF(2^4)^2$ ในส่วนของการเพิ่มจำนวนคีย์ (Expaned Keys) และรูปแบบการทำงานที่เลือกใช้ตารางการเทียบค่าในส่วนของวงจรเข้ารหัสแบบ AES (Core AES)

4. รูปแบบการทำงานแบบ GG ($GF(2^4)^2$ & $GF(2^4)^2$)

เป็นรูปแบบการทำงานที่เลือกใช้ $GF(2^4)^2$ ทั้งในส่วนของการเพิ่มจำนวนคีย์ (Expaned Keys) และอีกส่วนคือวงจรเข้ารหัสแบบ AES (Core AES)

งานวิทยานิพนธ์นี้ได้ออกแบบและพัฒนาวงจรที่ใช้ในกระบวนการเข้ารหัสข้อมูลแบบ AES เสร็จสิ้นและได้ทดสอบประสิทธิภาพของวงจรรวมทั้งหมดซึ่งให้ผลดังตารางที่ 4-10 และตารางที่ 4-11

ตารางที่ 4-10 ประสิทธิภาพด้านการใช้ทรัพยากรของการเข้ารหัสข้อมูลแบบ AES แบบต่างๆ

รูปแบบของวงจรการเข้ารหัสแบบ AES	**ชื่อย่อ**	**จำนวนแผ่นลอจิกที่วงจรใช้ (slices)**	**ปริมาณงานต่อหน่วยเวลา (Mbps)**	**ประสิทธิภาพของวงจร(Mbps/slices)**
LUTs ใน Key และ AES	LL	4325	1251	0.289
LUTs ใน Key และ $GF(2^4)^2$ ใน AES	LG	7125	320	0.045
$GF(2^4)^2$ ใน Key และ LUTs ใน AES	GL	4532	898	0.198
$GF(2^4)^2$ ใน Key และ AES	GG	7862	1184	0.150

ตารางที่ 4-11 ประสิทธิภาพด้านการใช้กำลังงานของการเข้ารหัสข้อมูลแบบ AES แบบต่างๆ

ชื่อย่อรูปแบบของวงจรการเข้ารหัสแบบ AES	**กำลังงานที่วงจรใช้ (mW)**	**ประสิทธิภาพด้านกำลังงานของวงจร (Mbps/mW)**
LL	575	2.175
LG	553	0.578
GL	439	2.045
GG	408	2.901

จากตารางที่ 4-10 แสดงประสิทธิภาพของการเข้ารหัสข้อมูลแบบ AES ในโครงสร้างการทำงานของ S-box ที่เลือกใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ กับตารางเทียบค่าใน 4 รูปแบบคือ LL, LG, GL และ GG เนื่องจาก S-box ถูกเรียกใช้งานมากในส่วนของการเข้ารหัสหลัก จึงทำให้รูปแบบ LG และ GG ซึ่งใช้กาลัวร์ฟิลด์ $GF(2^4)^2$ นั้นมีการใช้แผ่นลอจิกหรือทรัพยากรค่อนข้างสูง คือ 7,125 และ 7,862 ตามลำดับ ซึ่งสอดคล้องกับผลที่ได้จากหัวข้อที่ 4.7.1

ส่วนประสิทธิภาพด้านกำลังงานของการเข้ารหัสข้อมูลแบบ AES นั้นรูปแบบ GG ใช้กำลังงานน้อยที่สุดเพียง 408 mW และเมื่อเปรียบเทียบประสิทธิภาพของปริมาณงานต่อหน่วยเวลาต่อกำลังงานที่วงจรใช้ พบว่า รูปแบบ GG ยังคงให้ประสิทธิภาพสูงในด้านกำลังงานของวงจรที่ 2.901 Mbps/mW ซึ่งสูงกว่าการทำงานรูปแบบอื่นๆ

4.8 การเปรียบเทียบประสิทธิภาพเมื่อนำไปใช้งานบนเครือข่ายเซนเซอร์ไร้สาย

ฮาร์ดแวร์ที่ใช้งานจริงในการรับส่งข้อมูลเช่น CC2420 ของบริษัท TI มีการพัฒนาวงจรการเข้ารหัสอยู่บนชิป CC2420 ถึงแม้จะสะดวกต่อการใช้งาน แต่ก็มีข้อจำกัดในด้านการใช้งานที่จะต้องติดต่อผ่านไมโครคอนโทรลเลอร์ ทำให้เสียเวลาในการทำงานไปส่วนหนึ่งซึ่งการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์และ CC2420 จะต้องทำผ่านทางพอร์ตอนุกรม SPI (Serial Peripheral Interface) ที่ความเร็ว 4 MHz ในขณะที่อัตราการส่งข้อมูลแบบไร้สายจะอยู่ที่ 250 kbps ทำให้จำเป็นที่จะต้องมีการสร้างที่พักข้อมูลหรือ Buffer ขึ้นมารองรับ

นอกจากนี้ในงาน [2] สามารถวัดความเร็วในการทำงานของการเข้ารหัสของโมดูล CC2420 ได้ โดยวัดจากขาเอาต์พุตของ MSP430 ด้วยออสซิลโลสโคป ตามขั้นตอนการทำงานของการเข้ารหัสใน CC2420 ตามที่แสดงในตารางที่ 4-12 มีดังนี้

1. เขียนข้อมูลไปยัง RAM ของ CC2420
2. การใช้คำสั่งในการเข้ารหัสของ CC2420
3. รอให้โมดูลการเข้ารหัสประมวลผลเสร็จเรียบร้อย โดย requesting status byte
4. อ่านค่าจาก RAM ของ CC2420

ตารางที่ 4-12 เวลาในแต่ละขั้นตอนการทำงานของ CC2420 [2]

ขั้นตอนการทำงานของ CC2420	**เวลา (µs)**
เขียนข้อมูลไปยัง RAM ของ CC2420	94.40
การใช้คำสั่งในการเข้ารหัสของ CC2420	6.40
รอให้โมดูลการเข้ารหัสประมวลผลเสร็จเรียบร้อย โดย requesting status byte	18.40
อ่านค่าจาก RAM ของ CC2420	102.40
รวมเวลาในการทำงาน	**221.60**

จากผลการวิเคราะห์ข้างต้นพบว่ากระบวนการเข้ารหัสด้วยฮาร์ดแวร์บน CC2420 มีความเร็วในการทำงานระดับหน่วยไมโครวินาที (µs) ในขณะที่การเข้ารหัสด้วยซอฟต์แวร์ประมวลผลบนไมโครคอนโทรลเลอร์ทั่วไปใช้ความเร็วในการทำงานระดับหน่วยมิลลิวินาที (ms) ซึ่งแตกต่างกันในระดับ 1,000 เท่าโดยประมาณ แต่ข้อจำกัดของวงจรการเข้ารหัสบน CC2420 คือความเร็วในการเข้ารหัสข้อมูลยังช้า เมื่อเทียบกับวงจรการเข้ารหัสที่ได้พัฒนาในงานนี้ ซึ่งมีความเร็วอยู่ในระดับนาโนวินาที (ns) ดังแสดงในตารางที่ 4-13

ตารางที่ 4-13 ประสิทธิภาพและเวลารวมของการเข้ารหัสข้อมูลแบบ AES แบบต่างๆ

ชื่อย่อรูปแบบของวงจรการเข้ารหัสแบบ AES	ปริมาณงานต่อหน่วยเวลา (Mbps)	Minimum period (ns)	Maximum Frequency (MHz)	จำนวนรอบนาฬิกา (clock cycles) เข้า/ถอดรหัส	เวลารวม (ns) (เข้า/ถอดรหัส)
LL	1251	1.894	527.856	52/50	102.276/94.700
LG	320	7.679	130.217	52/50	399.308/383.950
GL	898	5.697	175.528	52/50	296.244/284.850
GG	1184	8.312	120.308	13/23	108.056/191.176

ดังนั้นการเข้ารหัสข้อมูลบนฮาร์ดแวร์ที่ออกแบบนี้จะช่วยลดค่าหน่วงเวลาที่อาจจะเกิดขึ้นเมื่อมีการนำไปประยุกต์ใช้งานบนเครือข่ายเซนเซอร์ไร้สาย ซึ่งโดยหลักการของระบบเครือข่ายเซนเซอร์ไร้สายต้องการส่งข้อมูลแบบเรียลไทม์ (Real-time) เพื่อนำข้อมูลที่ได้ไปวิเคราะห์หรือแจ้งเตือนภัย ดังนั้นข้อมูลที่สำคัญที่ต้องการเข้ารหัสจะต้องมีการหน่วงเวลาน้อยหรือไม่มีการหน่วงเวลา หากนำไปใช้ในระบบเครือข่ายเซนเซอร์ไร้สาย มิเช่นนั้นอาจจะต้องให้เกิดอันตรายถึงแก่ชีวิตได้ เช่นการนำไปประยุกต์ใช้ส่งสัญญาณชีพของผู้ป่วยในสถานพยาบาล การส่งสัญญาณเตือนภัยจากภัยพิบัติธรรมชาติ เป็นต้น นอกจากนี้แล้วในด้านของการใช้กำลังงานพบว่าวงจรที่ได้พัฒนายังมีประสิทธิภาพของการใช้กำลังงานที่สูงอีกด้วย เหมาะในการนำไปประยุกต์ใช้งานบนเครือข่ายเซนเซอร์ไร้สายที่มีข้อจำกัดในเรื่องของแหล่งจ่ายกำลังงาน

จากผลการทดลองของวิทยานิพนธ์นี้สามารถวิเคราะห์การทำงานได้ว่า ในการประมวลผล โปรแกรมสามารถระบุค่าหน่วงเวลาได้และแสดงค่าสัญญาณนาฬิกาสูงสุด (Maximum Frequency) ที่วงจรสามารถใช้ได้ ซึ่งเวลารวมในการเข้าและการถอดรหัสนั้นหน่วยของเวลาอยู่ในระดับนาโนวินาที คือ ในช่วง 95-400 ns ซึ่งน้อยกว่าการทำงานของฮาร์ดแวร์โมดูล CC2420 ที่ใช้เวลาในการเข้ารหัส 221.60 μs (ตารางที่ 4-13) คิดเป็น 550 เท่าโดยประมาณ (เมื่อใช้เวลา 400 ns) และยังน้อยกว่าการทำงานของ CC2420 บนอุปกรณ์จริง [3] อย่างเช่น MICAz จะใช้เวลาในการเข้ารหัส 323.834 μs และ Tmote SKY จะใช้เวลาในการเข้ารหัส 2509.543 μs (ซึ่งได้กล่าวไว้ในตารางที่ 1-3 บทที่ 1)

ซึ่งก็ตรงกับการใช้งานในการสื่อสารบนเครือข่ายเซนเซอร์ไร้สายจาก [16] พบว่าการทำงานในการส่ง/รับข้อมูลนั้นอยู่ใน physical layer ซึ่งโครงสร้างของแพ็กเก็ตจะอยู่ในช่วง

ความยาวที่ 0-127 ไบต์ และในงานประยุกต์เกี่ยวกับ home application เช่น การ monitoring และการควบคุมด้านความปลอดภัย การควบคุมการใช้งานหลอดไฟ เครื่องปรับอากาศ หรือระบบไฟฟ้าตลอดจนงานในรูปแบบอื่นๆ นั้น ขนาดปกติของข้อมูลที่ถูกใช้จะอยู่ในช่วง 30-60 ไบต์ แต่ก็ขึ้นอยู่กับชนิดของงานอีกด้วย หากเป็นงานเกี่ยวกับเกมหรือการเชื่อมต่อกับคอมพิวเตอร์จะส่งผลต่อขนาดของข้อมูล ซึ่งจะมีขนาดใหญ่ขึ้นไปด้วย จากอัตราการส่งผ่านข้อมูลแล้วแพ็คเก็ตสูงสุดจะใช้ระยะเวลา 4.25 มิลลิวินาทีสำหรับ 2.4 GHz สรุปได้ว่าโครงสร้างแพ็กเก็ตสูงสุดในชั้น physical layer ในมาตรฐานของ IEEE 802.15.4 มีขนาด 133 ไบต์ หรือเท่ากับ 1064 บิต นั้นส่งข้อมูลใช้เวลา 4.25 มิลลิวินาที ดังนั้นข้อมูลถูกส่งด้วยความเร็ว 250.35 kbps ตามการส่งข้อมูลแบบ IEEE 802.15.4 ในส่วนของวิทยานิพนธ์ฉบับนี้ ผลจากการจำลองการทำงานเมื่อนำมาคำนวณปริมาณงานต่อหน่วยเวลา มีค่าโดยประมาณอยู่ในช่วง 320-1250 Mbps ซึ่งสามารถรับส่งข้อมูลได้เร็วกว่าถึง 1280-5000 เท่า ในด้านกำลังงานที่ใช้อยู่ในช่วง 408-575 mW หรือใช้พลังงานในช่วง 44.09-220.82 nJ ดังแสดงในตารางที่ 4-14

ตารางที่ 4-14 กำลังงานและพลังงานของวงจรการเข้ารหัสข้อมูลแบบ AES แบบต่างๆ

ชื่อย่อรูปแบบของวงจรการเข้ารหัสแบบ AES	**กำลังงานที่วงจรใช้ (Power; mW)**	**เวลารวม (ns) (เข้า/ถอดรหัส)**	**Energy ที่วงจรใช้ (nJ) Energy = Power * Time**
LL	575	102.28/94.70	58.81/54.45
LG	553	399.31/383.95	220.82/212.32
GL	439	296.24/284.85	130.05/125.05
GG	408	108.06/191.18	44.09/78.00

หากเปรียบเทียบประสิทธิภาพการนำไปใช้งานจริง แบตเตอร์รี่ยี่ห้อ Energizer [17] ขนาด AA จำนวนสองก้อน หนึ่งก้อนมีค่า Nominal Voltage เป็น 1.5 volts และ Nominal IR เป็น 150 ถึง 300 mΩ การ discharge ที่ 0.8 volts พบว่าสามารถใช้งานได้โดยประมาณ 100 ชั่วโมง ดังนั้น Power (W) ของถ่าน AA $= IV = I^2R = V^2/R$

$$= \frac{2 \text{ ก้อน} \times (1.5 \text{ V})^2}{150\ m\Omega}$$

= 30 W (ที่ 150 mΩ) หรือ 15 W (ที่ 300 mΩ)

แบตเตอร์รี่ AA จำนวน 2 ก้อนมีกำลังงานรวมเท่ากับ 15-30 W และจากตารางที่ 4-14 พบว่าการเข้ารหัสแบบ GG ใช้กำลังงานไป 408 mW ซึ่งหากนำใช้งานจริงอยู่ที่ 36.76 เท่า (ที่ 300 mΩ) หรือ 73.53 เท่า (ที่ 150 mΩ)

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

ในบทนี้จะกล่าวถึงการสรุปผลและข้อเสนอแนะที่ได้จากการดำเนินการทำวิทยานิพนธ์ ตลอดจนปัญหาและอุปสรรคที่เกิดขึ้นขณะทำวิทยานิพนธ์ และท้ายที่สุดจะกล่าวถึงข้อเสนอแนะแก่ผู้สนใจที่จะนำวิทยานิพนธ์ชิ้นนี้เพื่อพัฒนาต่อไป

5.1 สรุปผล

งานวิจัยชิ้นนี้นำเสนอการออกแบบวงจรการเข้ารหัสข้อมูลแบบ AES เพื่อให้มีประสิทธิภาพสูงและสามารถพัฒนาต่อเพื่อใช้ในงานเครือข่ายเซนเซอร์ไร้สาย โดยจำลองวงจรที่ได้ออกแบบด้วยการสังเคราะห์วงจรของโปรแกรม Xilinx ISE 10.1 ปรับปรุงการทำงานของวงจรในส่วนของฟังก์ชัน S-box ซึ่งเป็นฟังก์ชันที่มีการเรียกใช้งานมากที่สุด อีกทั้งยังเป็นฟังก์ชันที่มีขั้นตอนการคำนวณที่ซับซ้อน การพัฒนานี้เพื่อให้ได้วงจรที่มีประสิทธิภาพสูงนั้น ต้องคำนึงถึงการใช้ทรัพยากร เวลาการประมวลผล ตลอดจนการใช้กำลังงานให้น้อยที่สุด

แนวคิดและวิธีการออกแบบวงจร S-box ของการเข้ารหัสข้อมูลแบบ AES นั้น วิธีการอย่างง่ายที่นิยมใช้คือ การใช้ตารางการเทียบค่าซึ่งจะต้องอ้างอิงถึง 2 ตาราง เนื่องจากตารางหนึ่งต้องใช้สำหรับการเข้ารหัส และอีกตารางหนึ่งต้องใช้สำหรับการถอดรหัส แต่ในวิทยานิพนธ์นี้ได้ใช้การคำนวณด้วยกาลัวร์ฟิลด์ $GF(2^4)^2$ ในวงจร S-box ซึ่งวงจรที่สร้างขึ้นนี้สามารถใช้ได้ทั้งเข้ารหัสและถอดรหัส

นอกจากนี้ยังได้เสนอแนวคิดเพื่อลดการใช้ XOR และรูปแบบโครงสร้างดังที่ได้กล่าวในบทที่ 3 เพื่อให้สามารถนำไปใช้ประโยชน์และเลือกประยุกต์ใช้กับงานที่แตกต่างตามความเหมาะสม ซึ่งงานวิทยานิพนธ์นี้นำเสนอประสิทธิภาพด้านการใช้กำลังงานซึ่งการเข้ารหัสและการถอดรหัสโดยใช้การคำนวณด้วยกาลัวร์ฟิลด์ $GF(2^4)^2$ นั้นให้ปริมาณงานต่อหน่วยเวลาที่ 1.184 Gbps (แสดงในตารางที่ 4-9) นอกจากนั้นยังใช้กำลังงานที่น้อยกว่าแบบตารางการเทียบค่าคือใช้กำลังงานอยู่ที่ 408 mW (แสดงในตารางที่ 4-10)

สรุปได้ว่าวงจรการเข้ารหัสแบบ AES ที่ได้ออกแบบนี้เหมาะกับงานเครือข่ายเซนเซอร์ไร้สายที่ต้องการความเร็วในการประมวลผลในด้านความปลอดภัยโดยใช้การเข้ารหัสแบบ AES ที่มีขนาดคีย์ 128 บิตและงานประยุกต์ที่ข้อมูลเป็นหัวใจสำคัญเช่น ด้านการแพทย์ [18] เนื่องจากพารามิเตอร์ที่ใช้ในการเก็บข้อมูลทางการแพทย์เป็นข้อมูลส่วนบุคคล และรายละเอียด

บางอย่าง เช่นน้ำหนัก อายุ โรคประจำตัว ส่งผลต่อการให้ยา เช่นอุปกรณ์ชื่อว่า "*iMedTracker*" ที่ช่วยในการเตือนผู้ป่วยในการรับประทานยา และช่วยระมัดระวังในเรื่องการทานยาผิด นอกจากนั้นยังมีข้อมูลของตัวตรวจจับเพื่อเก็บข้อมูลในการรักษา ซึ่งส่วนใหญ่ส่งผลถึงชีวิตของผู้ป่วยด้วย เช่น สัญญาณอัตราการเต้นของหัวใจ ความดันโลหิต เป็นต้น

5.2 ผลที่ได้จากการทำวิทยานิพนธ์ชุดนี้

ผู้ทำวิทยานิพนธ์ได้ศึกษาและวิเคราะห์การออกแบบวงจรการเข้ารหัสข้อมูลแบบ AES โดยเน้นการปรับปรุง S-box ซึ่งมีจุดประสงค์เพื่อได้วงจรการเข้ารหัสและถอดรหัสที่ใช้กำลังงานต่ำและมีประสิทธิภาพสูงสำหรับงานเครือข่ายเซนเซอร์ไร้สาย โดยการนำแนวคิดการคำนวณด้วยกาลัวร์ฟิลด์ $GF(2^4)^2$ มาใช้พัฒนาวงจรการเข้ารหัสนี้บนอุปกรณ์ที่สามารถโปรแกรมได้ (FPGA) บรรยายด้วยภาษา VHDL โดยใช้เทคนิคโครงสร้างแบบต่างๆ มาเปรียบเทียบการทำงาน (รายละเอียดกล่าวไว้ในบทที่ 3) ซึ่งการออกแบบและนำวงจรมาทดสอบประสิทธิภาพในด้านความถูกต้องของการทำงานของวงจรการเข้ารหัสข้อมูล ทดสอบความเร็วสูงสุดที่วงจรสามารถใช้งานได้ รวมถึงทดสอบหาประสิทธิภาพของการใช้ทรัพยากรและกำลังงานที่สูญเสียในการทำงานวงจร ทำให้สรุปผลจากการสังเคราะห์วงจรได้ว่าความเร็วในการเข้ารหัสข้อมูลมีความเหมาะสม สามารถนำไปใช้งานได้จริงบนเครือข่ายเซนเซอร์ไร้สาย เพราะมีความเร็วในการทำงานอยู่ที่ 320-1250 Mbps มีความถูกต้องของการเข้ารหัสและใช้กำลังงานน้อยอยู่ที่ 408 mW จึงสามารถนำไปใช้บนงานเครือข่ายเซนเซอร์ไร้สายได้

ผลที่ได้รับจากวิทยานิพนธ์ชุดนี้คือ ได้แนวคิดในการออกแบบวงจร โดยการนำอัลกอริทึมที่มีอยู่มาประยุกต์ใช้งาน เพื่อปรับปรุง เปลี่ยนแปลงให้วงจรเหมาะสมต่อการประยุกต์ใช้งาน โดยเฉพาะวงจรการเข้ารหัสแบบ AES นอกจากนั้นยังได้ทราบถึงการออกแบบวงจรเพื่อให้ประหยัดกำลังงาน และสามารถนำไปประยุกต์ใช้งานเพื่อเพิ่มประสิทธิภาพในการเข้ารหัสในงานเครือข่ายเซนเซอร์ไร้สาย

5.3 ปัญหาและอุปสรรค

ปัญหาและอุปสรรคในการดำเนินวิทยานิพนธ์นี้ ในด้านความรู้ผู้ทำวิทยานิพนธ์นั้นต้องศึกษาการเข้ารหัสข้อมูล เพื่อวิเคราะห์หาความเหมาะสมในการออกแบบวงจร รวมทั้งการศึกษาอัลกอริทึมของวงจรการเข้ารหัส ซึ่งพื้นฐานงานในด้านนี้ผู้ทำวิทยานิพนธ์มีความรู้อยู่น้อยมาก ส่วนด้านการออกแบบวงจรนั้น ผู้ทำวิทยานิพนธ์มีพื้นฐานอยู่บ้าง ซึ่งใช้ระยะเวลาในการศึกษาเกี่ยวกับการเข้ารหัสข้อมูลอยู่พอสมควร รวมไปถึงการหาเทคนิคในการออกแบบวงจรและวิธีใน

การแก้ปัญหา นอกจากนั้นด้านการเขียนรายงานเชิงวิชาการที่ไม่ดีเท่าที่ควร จึงต้องมีการปรับปรุงแก้ไขบ่อยครั้ง ส่วนปัญหาด้านการทำงานของวิทยานิพนธ์นั้น ได้วางแผนในการทำงานเป็นลำดับแยกงานเป็นส่วนๆ งานก็ดำเนินไปด้วยความราบรื่น แต่เมื่อนำงานมารวมเข้าด้วยกันเกิดปัญหาจึงได้ขอคำปรึกษาจากอาจารย์ จึงได้ทราบว่าปัญหาเกิดจากการทำงานไม่สัมพันธ์สอดคล้องของสัญญาณนาฬิกา จึงทำให้เสียเวลาในการแก้ไข

5.4 ข้อเสนอแนะ

งานวิทยานิพนธ์ชิ้นนี้นำเสนอการเข้ารหัสข้อมูลแบบ AES และเน้นการพัฒนาและออกแบบในส่วนของวงจร S-box ซึ่งเป็นเพียงบางส่วนของวงจรการเข้ารหัสข้อมูลแบบ AES ผู้ที่สนใจนำวิทยานิพนธ์นี้ไปใช้งานหรือศึกษาต่อเนื่อง สามารถปรับปรุง S-box ให้ดีขึ้นรวมทั้งฟังก์ชันอื่นๆ ของวงจรให้ดีขึ้นได้อีกด้วย ข้อจำกัดของวงจรที่ได้นำเสนอในวิทยานิพนธ์นี้คือ ข้อมูลลับและคีย์ลับยังคงมีขนาดเพียง 128 บิต และคีย์จะยังฝังตัวหรือกำหนดคีย์ไว้ให้เฉพาะสำหรับตัวอุปกรณ์หนึ่งๆ เท่านั้น

บรรณานุกรม

[1] T. Kavitha, and D. Sridharan, “Security Vulnerabilities In Wireless Sensor Networks: A Survey,” *Journal of Information Assurance and Security 5*, pp. 031-044, 2010.

[2] Shammi Didla et al., “Optimizing AES for embedded devices and wireless sensor networks,” in *Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*, Innsbruck, Austria, 2008.

[3] M. Healy, T. Newe, and E. Lewis., “Efficiently securing data on a wireless sensor network,” *Journal of Physics*: Conference Series, 76(1), 2007.

[4] Tanzilur Rahman et al., “Design of a High Throughput 128-bit AES (Rijndael Block Cipher),” in *Proceedings of the International Multi Conference of Engineers and Computer Scientists 2010, IMECS2010*, Vol II, 2010.

[5] Jason Van Dyken et al., "FPGA schemes for minimizing the power-throughput trade-off in executing the Advanced Encryption Standard algorithm," *Journal of Systems Architecture 56*, Volume 56, page 116-123, 2010.

[6] Rais, M.H. and Qasim, S.M., “Resource Efficient Implementation of S-box base on reduced residue of prime numbers using Virtex-5 FPGA,” in *Proceedings of the world congress on Engineering 2010* Vol II, WCE 2010, London, U.K., 2010.

[7] J. L´azaro et al., “AES –Galois Counter Mode Encryption/Decription FPGA Core for Industrial and Residential Gigabit Ethernet communications,” *Reconfigurable Computing: Architectures, Tools and Applications Lecture Notes in Computer Science*, Volume 5453/2009, pp. 312-317, 2009.

[8] National Institute of Standards and Technology (NIST): Advanced Encryption Standard (AES), FIPS-197, [Online]. Available: http://csrc.nist.gov/publications/PubsFIPS.html

[9] Abhijit Das,C. E. Veni Madhavan, “Public-Key Cryptography: Theory and Practice,” Pearson Education Inc., 2009.

[10] Zainalabedin Navabi, “Embedded core design with FPGAs,” New York : MC Graw –Hill, 2007.

[11] John L. Hennessy et al., “Computer architecture: a quantitative approach,” 4th ed., Morgan Kaufmann Publisher, 2007.

[12] Steve Kilts. “Advanced FPGA design: Architecture, Implementation, and Optimization,” John Wiley & Sons, Inc. , 2007.

[13] Troy Scott and Lattice, “Power consumption reduction for portable FPGA designs”, [Online].Available: http://www.latticesemi.com/documents/doc28425x79.pdf?jsessionid=f030ea0b31700a3eaf2846506f155f3281f9

[14] Johannes Wolkerstorfer , Elisabeth Oswald , Mario Lamberger, “An ASIC Implementation of the AES SBoxes,” in *Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology*, pp.67-78, 2002.

[15] AES Calculator of University of New South Wales, [Online]. Available: http://www.unsw.adfa.edu.au

[16] Anna Hac. “Wireless Sensor Network Design,” John Wiley & Sons, Ltd., 2003.

[17] Energizer Holding, Inc., “Product Datasheet Alkaline: ENERGIZER E91,” No. EBC - 8905AP-A [Online]. Available: http://data.energizer.com/PDFs/E91_AP.pdf

[18] Terrance J. Dishongh Michael McGrath. “Wireless Sensor Networks for Healthcare Applications,” Boston : Artech House, 2010.

[19] Xilinx®Inc. (2008), “Core Generator Guide,” [Online]. Available: http://www.xilinx.com/itp/xilinx6/books/docs/cgn/cgn.pdf

[20] Xilinx®Inc. (2008), “Xilinx Power Tool Web Page,” [Online]. Available: http://www.xilinx.com/products/technology/power/index.htm

ภาคผนวก

ภาคผนวก ก
ตารางค่าการใช้งาน S-box และ Inverse S-box

ตารางการเทียบค่าการใช้งาน S-box

ตารางการเทียบค่าการใช้งาน S-box ที่แสดงในตารางที่ ก-1 นั้นแสดงการเข้ารหัสของตาราง S-box เท่านั้น หากจะถอดรหัสจะต้องใช้ค่าจากตารางที่ ก-2

ตารางที่ ก-1 ตารางการเทียบค่าการใช้งาน S-box สำหรับการเข้ารหัส

Y / X	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
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

ตัวอย่างการอ่านค่าการเข้ารหัสด้วยตาราง S-box เช่น {a9} ค่านี้จะแทนด้วยสัญลักษณ์ {XY} ดังนั้น X = a และ Y = 9 การเปิดตาราง ให้แนวตั้งคือค่าของ X และแนวนอนคือค่าของ Y หากจะหาค่า S-box ของ {a9} จะมีค่าเป็น {d3}

และหากต้องการถอดรหัส ก็จะใช้ตารางที่ ก-2 ตัวอย่างเช่น {d5} ค่านี้จะแทนด้วยสัญลักษณ์ {XY} ดังนั้น X = d และ Y = 5 การเปิดตาราง ให้แนวตั้งคือค่าของ X และแนวนอนคือค่าของ Y หากจะหาค่า S-box ของ {d5} จะมีค่าเป็น {b5}

ตารางที่ ก-2 ตารางการเทียบค่าการใช้งาน S-box สำหรับการถอดรหัส

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

ภาคผนวก ข
การใช้งานโปรแกรม Xilinx Core Generator

การใช้งานโปรแกรม Xilinx Core Generator

โปรแกรม Xilinx CORE Generator [19] เป็นโปรแกรมที่ใช้ในการสร้าง RAM ในการเก็บข้อมูลและใช้ในการสร้างตาราง Lookup ในการใช้งานตาราง S-box และ Inverse S-box การใช้โปรแกรม Xilinx Core generator ในการสร้างตารางและบรรจุค่าข้อมูลต่างๆลงในตาราง ซึ่งอาจจะสร้างไว้เพื่อรองรับข้อมูลที่โปรแกรมจะเก็บค่า หรืออาจจะต้องการการบรรจุค่าในตารางนั้นจำเป็นจะต้องใช้ ไฟล์ .coe ซึ่งในที่นี้ได้บันทึกค่าข้อมูลลงในไฟล์.coe เพื่อใช้ในการทำงานของฟังก์ชัน S-box แสดงดังภาพประกอบที่ 4–1

วิธีการสร้างไฟล์งาน .coe

การสร้างไฟล์งาน .coe นั้นจะต้องกำหนดส่วนของตัวเลขที่จะใช้ในรูปแบบใด เช่น ฐานสอง ฐานสิบหก เป็นต้น จากภาพประกอบใช้ค่าเลขฐานสิบหก จากนั้นก็กำหนดค่าของข้อมูลโดยแต่ละค่าที่บรรจุ จะถูกคั่นด้วยเครื่องหมายจุลภาคหรือลูกน้ำ (,) และจะปิดท้ายข้อมูลด้วยเครื่องหมายอัฒภาค (;) ดังภาพประกอบ ข-1

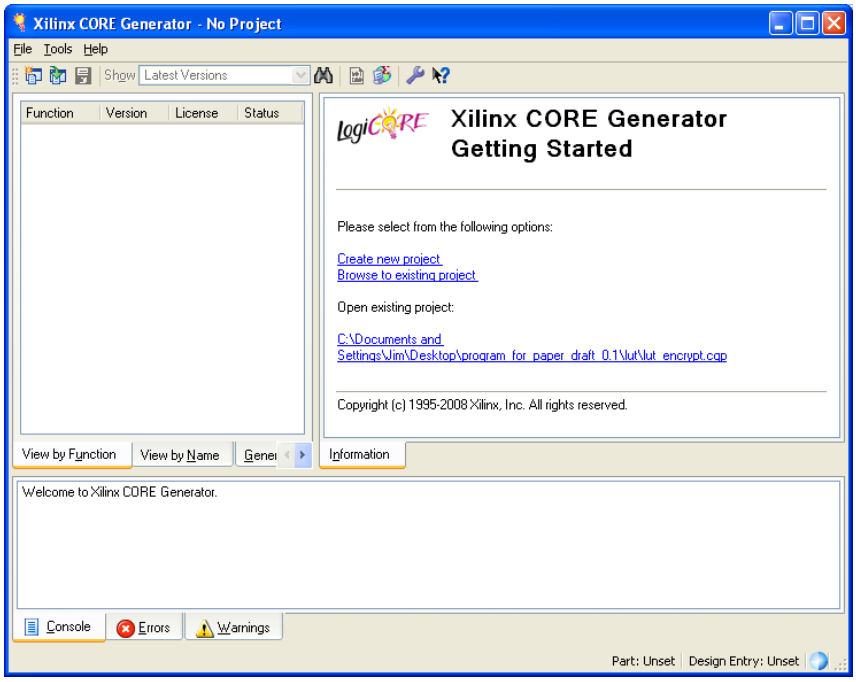
LUT_TB.coe

```
MEMORY_INITIALIZATION_RADIX=16;
MEMORY_INITIALIZATION_VECTOR=
63, 7C, 77, 7B, F2, 6B, 6F, C5,30, 01, 67, 2B, FE, D7, AB, 76,
CA, 82, C9, 7D, FA, 59, 47, F0,AD, D4, A2, AF, 9C, A4, 72, C0,
B7, FD, 93, 26, 36, 3F, F7, CC,34, A5, E5, F1, 71, D8, 31, 15,
04, C7, 23, C3, 18, 96, 05, 9A,07, 12, 80, E2, EB, 27, B2, 75,
09, 83, 2C, 1A, 1B, 6E, 5A, A0,52, 3B, D6, B3, 29, E3, 2F, 84,
53, D1, 00, ED, 20, FC, B1, 5B,6A, CB, BE, 39, 4A, 4C, 58, CF,
D0, EF, AA, FB, 43, 4D, 33, 85,45, F9, 02, 7F, 50, 3C, 9F, A8,
51, A3, 40, 8F, 92, 9D, 38, F5,BC, B6, DA, 21, 10, FF, F3, D2,
CD, 0C, 13, EC, 5F, 97, 44, 17,C4, A7, 7E, 3D, 64, 5D, 19, 73,
60, 81, 4F, DC, 22, 2A, 90, 88,46, EE, B8, 14, DE, 5E, 0B, DB,
E0, 32, 3A, 0A, 49, 06, 24, 5C,C2, D3, AC, 62, 91, 95, E4, 79,
E7, C8, 37, 6D, 8D, D5, 4E, A9,6C, 56, F4, EA, 65, 7A, AE, 08,
BA, 78, 25, 2E, 1C, A6, B4, C6,E8, DD, 74, 1F, 4B, BD, 8B, 8A,
70, 3E, B5, 66, 48, 03, F6, 0E,61, 35, 57, B9, 86, C1, 1D, 9E,
E1, F8, 98, 11, 69, D9, 8E, 94,9B, 1E, 87, E9, CE, 55, 28, DF,
8C, A1, 89, 0D, BF, E6, 42, 68,41, 99, 2D, 0F, B0, 54, BB, 16;
```

ภาพประกอบ ข-1 ตารางการเทียบค่า ในรูปแบบไฟล์ .coe

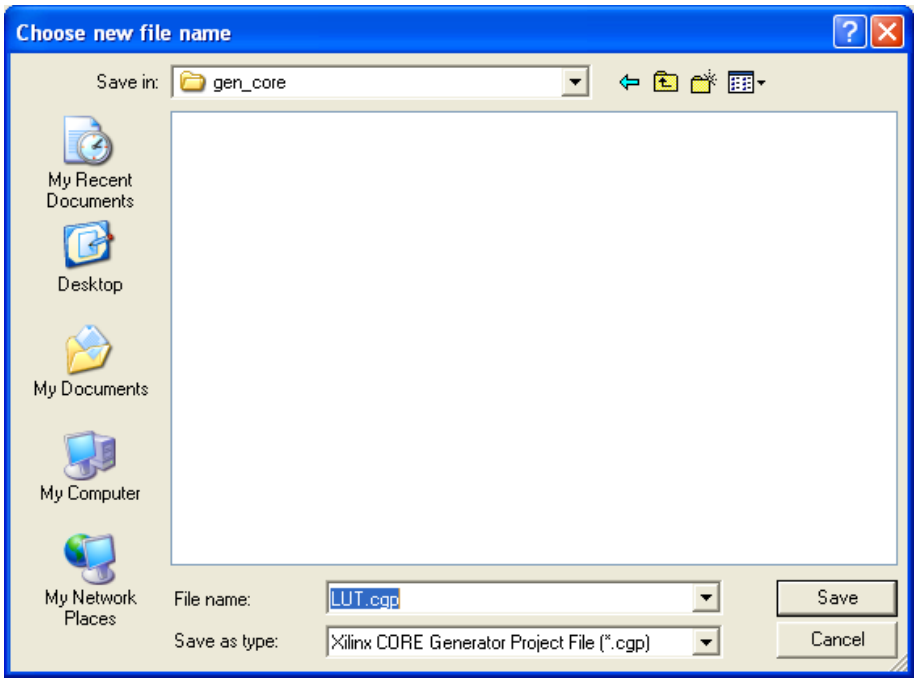
วิธีการใช้งานโปรแกรม Xilinx Core Generator

1. เปิดโปรแกรมโดยไปที่ Xilinx ISE Design Suite 10.1 >> ISE >> Accessories >> CORE Generator จะได้โปรแกรมดังภาพประกอบ ข-2



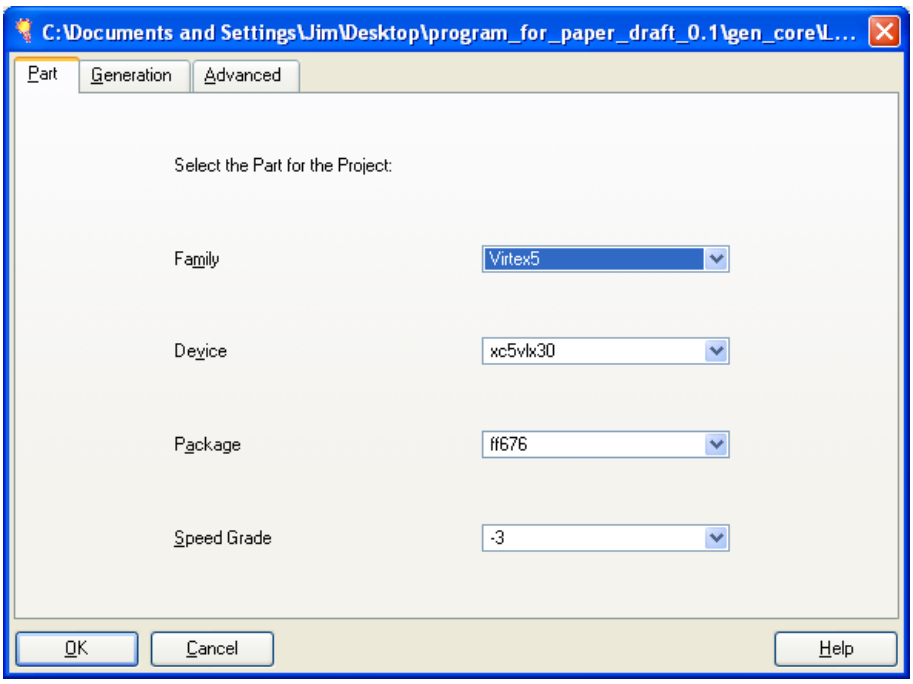
ภาพประกอบ ข-2 โปรแกรม Xilinx CORE Generator

2. เลือกเมนู File >> New Project >> ตั้งชื่อไฟล์ .cgp แล้วกดsave แสดงในภาพประกอบ ข-3



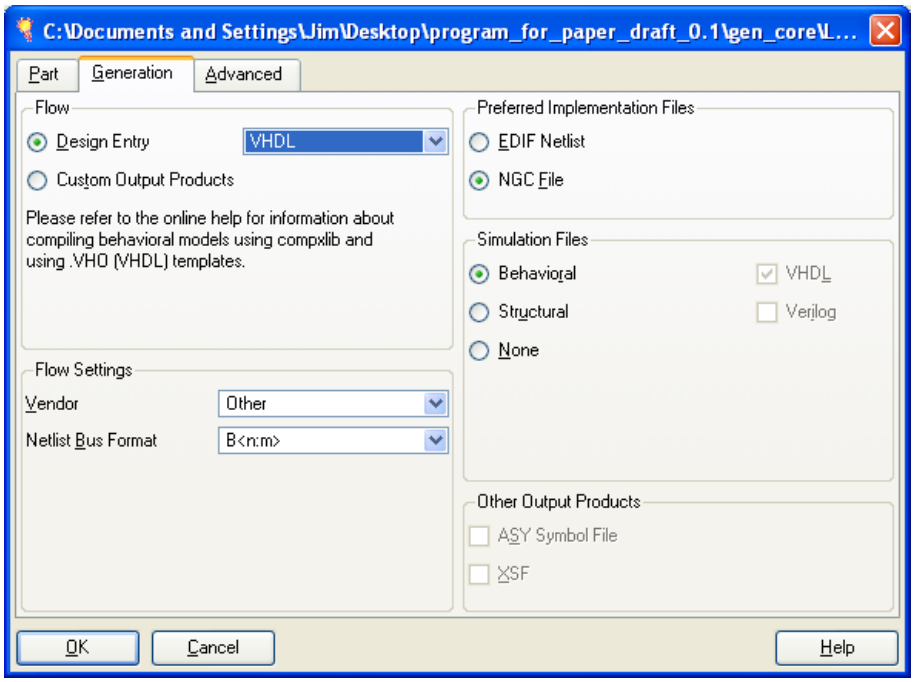
ภาพประกอบ ข-3 การตั้งชื่อไฟล์ใหม่

3. ภาพประกอบ ข-4 แสดงการเลือกอุปกรณ์ที่จะใช้ทดสอบ ในที่นี้ใช้ FPGA ตระกูล Virtex5 xc5vlx30 แพ็กเก็ต ff676 ความเร็วที่ -3



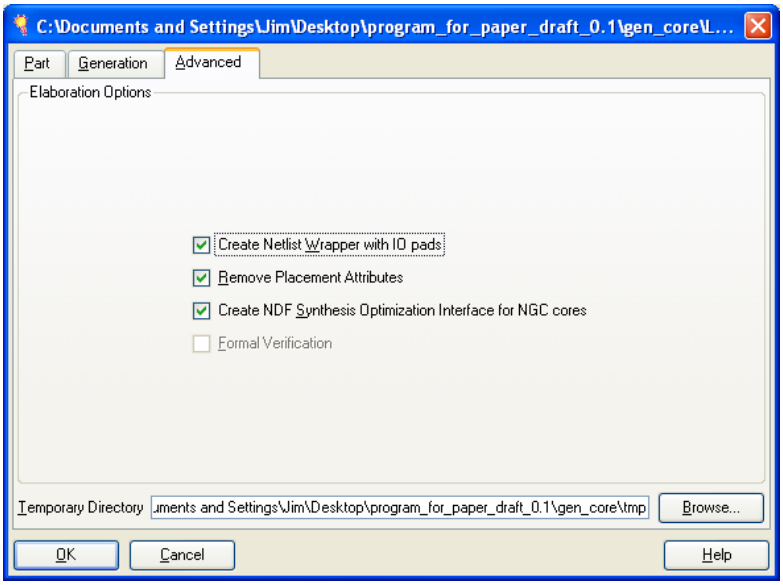
ภาพประกอบ ข-4 การเลือกอุปกรณ์ทดสอบในส่วนของ Part

4. จากนั้นกดแท็ป Generation ดังภาพประกอบ ข-5 เพื่อเลือกภาษาในการอธิบายโปรแกรม และสร้างไฟล์ NGC รวมทั้งการสังเคราะห์วงจรแบบ Behavioral



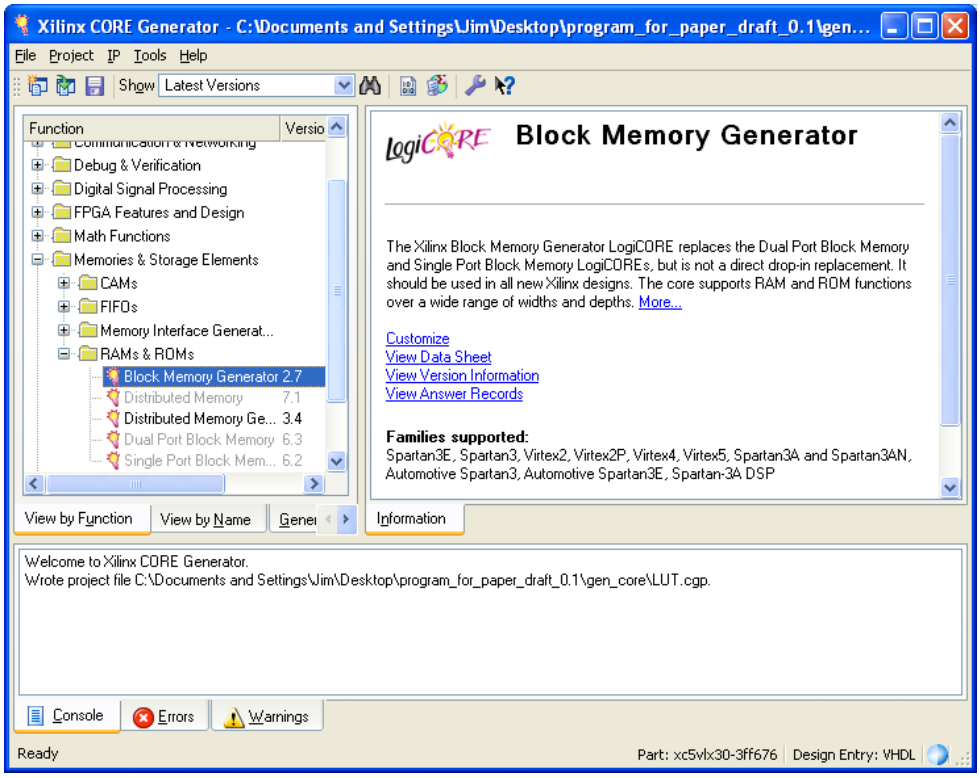
ภาพประกอบ ข-5 การเลือกอุปกรณ์ทดสอบในส่วนของ Generation

5. ในแท็ป Advance ดังภาพประกอบ ข-6 นั้น แสดงส่วนของรายละเอียดเพิ่มเติม การสร้าง Netlist Wrapper, การลบค่า Attribute , การสร้าง NDF ไฟล์



ภาพประกอบ ข-6 การเลือกอุปกรณ์ทดสอบในส่วนของ Advance

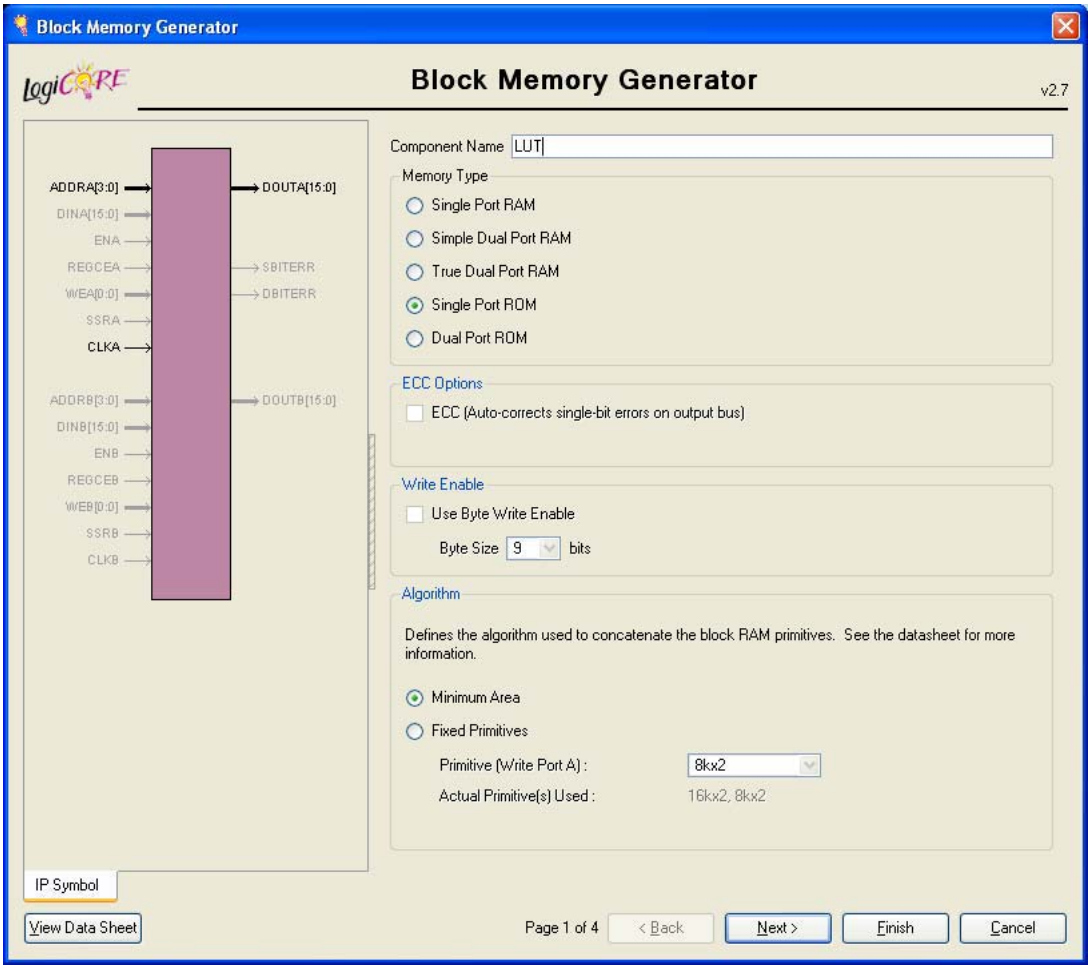
6. เมื่อคลิ๊ก ok จะได้โปรแกรมดังแสดงในภาพประกอบ ข-7



ภาพประกอบ ข-7 การเลือกฟังก์ชันในโปรแกรม Xilinx CORE Generator

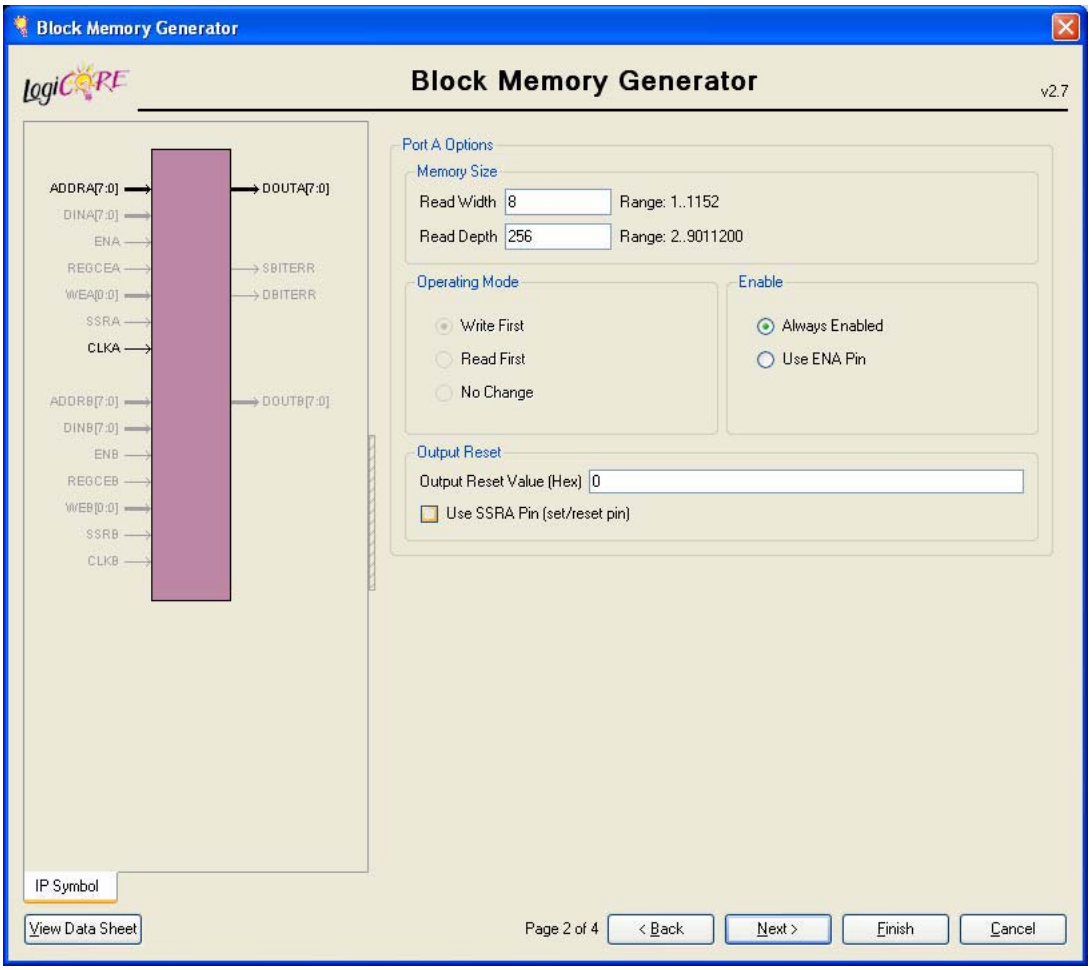
7. จากโปรแกรม Xilinx Core Generator ในส่วนของฟังก์ชันการใช้งาน บล็อกด้านซ้ายของโปรแกรม จะมีฟังก์ชันให้เลือกหลากหลาย ทั้งการติดต่อสื่อสารของเน็ตเวิร์ก การดีบักโปรแกรม ฟังก์ชันคณิตศาสตร์ การประมวลผลด้วยสัญญาณ ตลอดจนการสร้าง Memory and Storage Elements

8. ในงานชิ้นนี้เลือกใช้การสร้าง RAMs and ROMs ซึ่งใช้ตัวสร้างบล็อกหน่วยความจำ (Block Memory Generator) เวอร์ชัน 2.7 เมื่อดับเบิ้ลคลิ๊กจะได้โปรแกรมดังภาพประกอบ ข-8



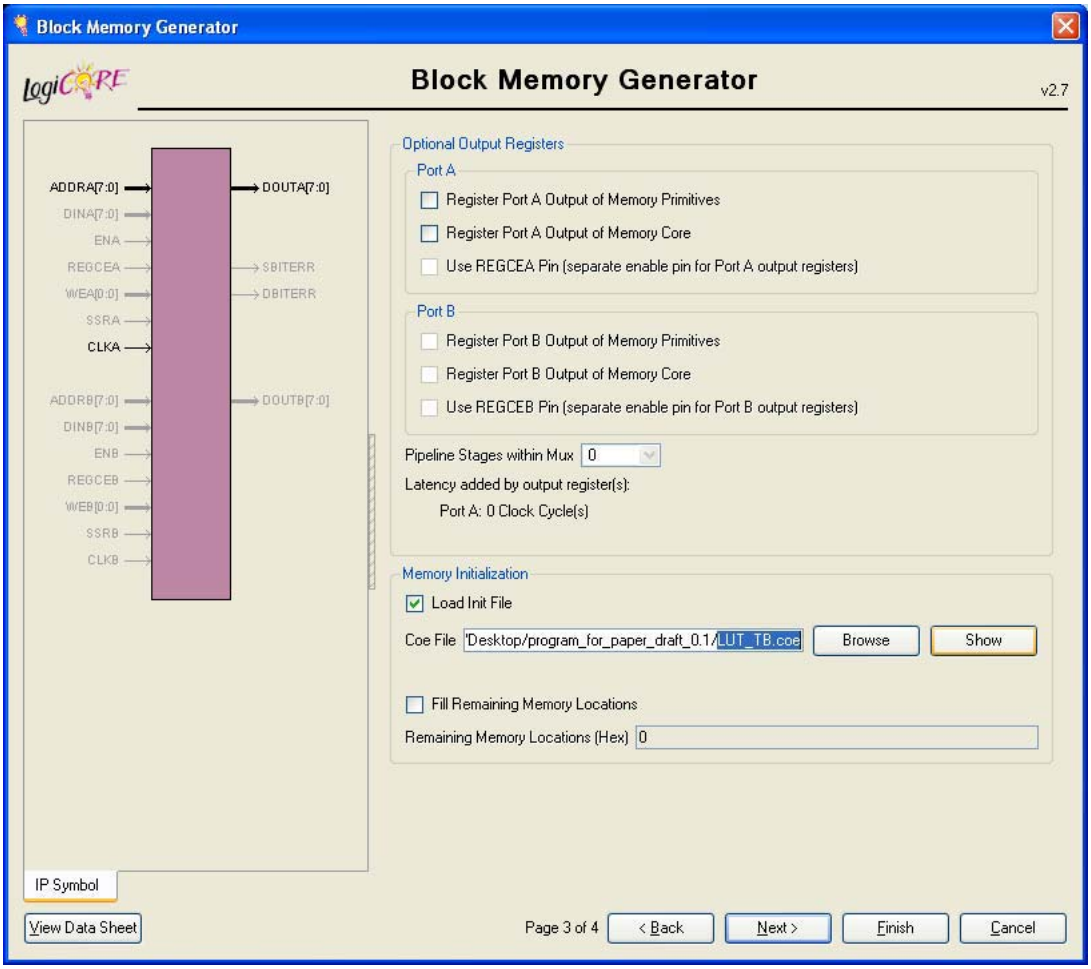
ภาพประกอบ ข-8 การสร้าง Block Memory

9. ตั้งชื่อคอมโพแนน แล้วเลือกประเภทของหน่วยความจำที่ต้องการ ในที่นี้เลือกการใช้งานแบบ Single Port ROM คลิ๊ก Next เพื่อเลือกขนาดของหน่วยความจำ โดยให้ความกว้างของหน่วยความจำ 8 บิตบรรจุข้อมูล 256 ชุด ดังแสดงในภาพประกอบ ข-9

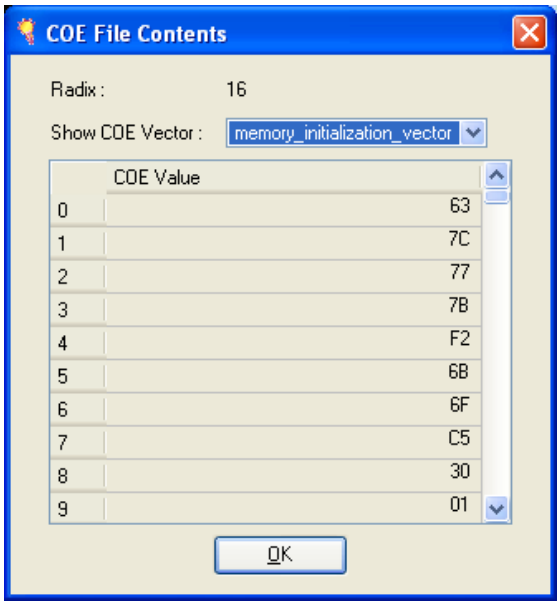


ภาพประกอบ ข-9 การกำหนดค่าให้กับหน่วยความจำ

10. จากนั้นเลือกโหลดไฟล์ค่าตั้งต้น (ไฟล์.coe) ให้กับหน่วยความจำในภาพประกอบ ข-10 และสามารถแสดงผลของค่าที่โหลดได้ในภาพประกอบ ข-11

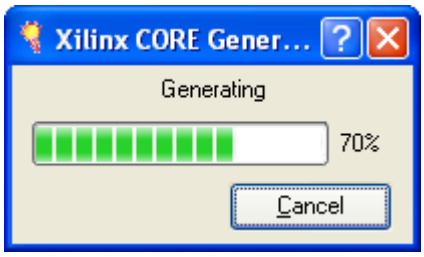


ภาพประกอบ ข-10 การโหลดค่าไฟล์ตั้งต้น (ไฟล์.coe)



ภาพประกอบ ข-11 ค่าที่บรรจุในไฟล์ .coe

11. เสร็จสิ้นการกำหนดค่าแล้วให้กด Finish และรอเพื่อให้ Xilinx Core Generator ทำการสร้างหน่วยความจำให้ ดังแสดงในภาพประกอบ ข-12 เมื่อสร้างเสร็จจะแสดงไฟล์ที่สร้างให้ในโฟล์เดอร์ที่ระบุในตอนแรกเริ่ม



ภาพประกอบ ข-12 โปรแกรมเพื่อให้ Xilinx Core Generator กำลังทำงาน

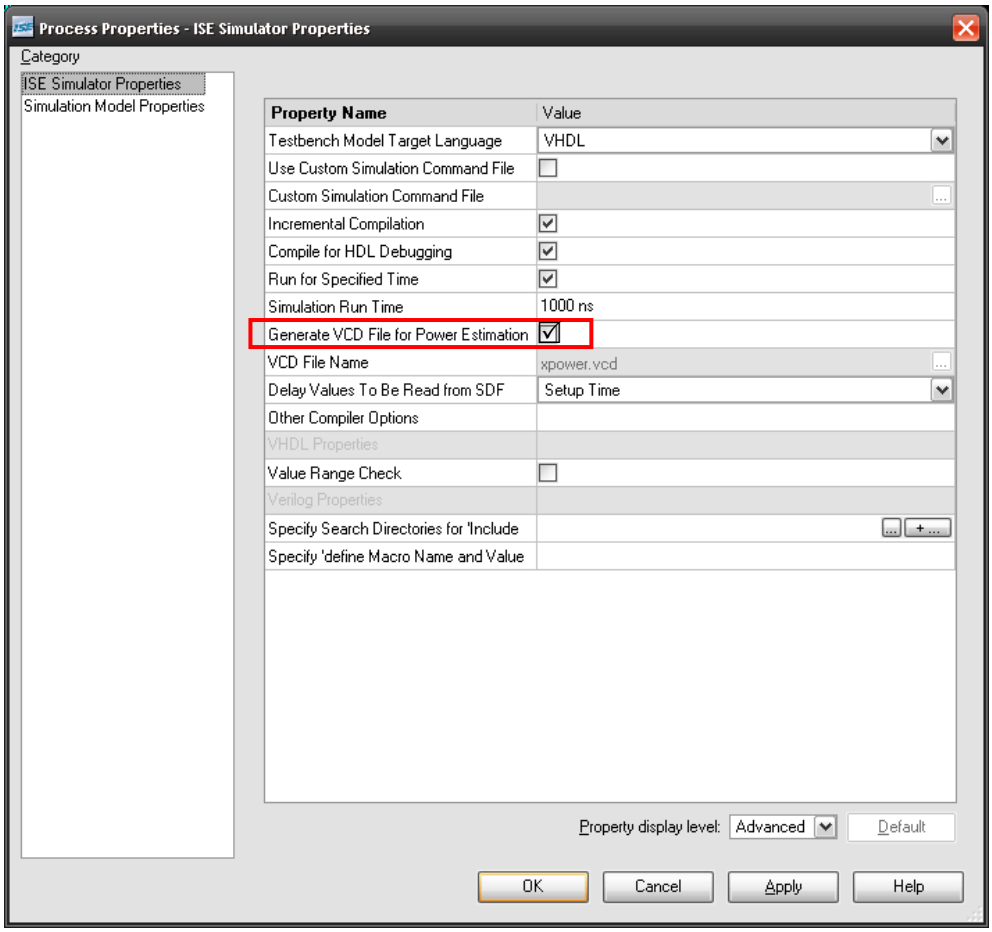
12. การใช้งานหน่วยความจำที่สร้างขึ้นนี้ สามารถใช้ไฟล์ .vhd ที่สร้างไปใช้งานได้เลย โดยเรียกใช้เหมือนเป็นคอมโพแนนหนึ่งๆ

ภาคผนวก ค
การใช้งานโปรแกรม XPower Anlyzer

การใช้งานโปรแกรม XPower Anlyzer

โปรแกรม XPower Anlyzer [20] เป็นโปรแกรมที่ใช้ในการคำนวณกำลังงานที่ใช้ในวงจร สามารถใช้โปรแกรมตามขั้นตอนดังนี้

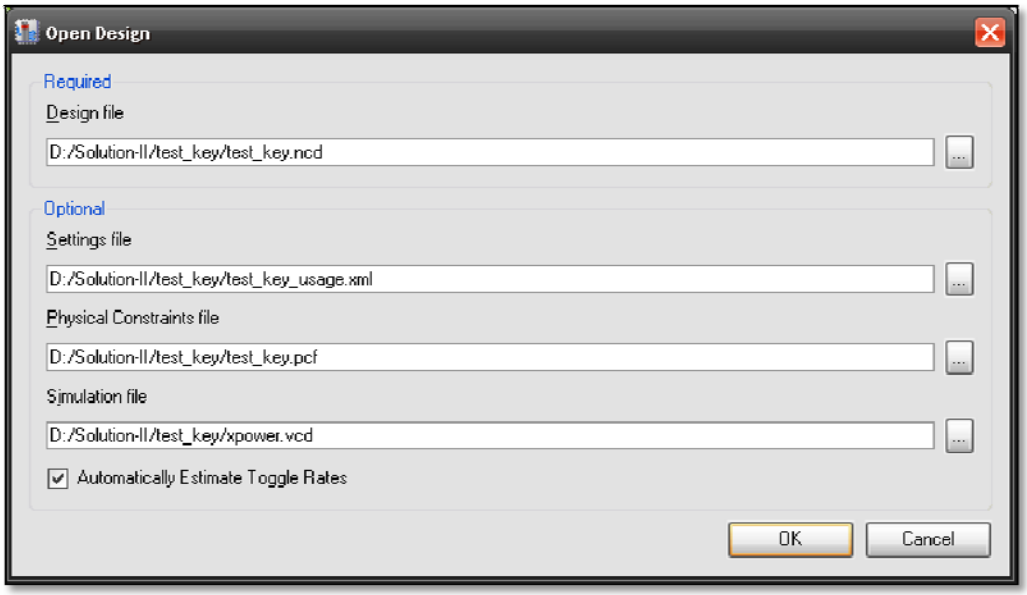
1. การทำงานของโปรแกรม XPower จะทำงานสอดคล้องกับการสังเคราะห์วงจรจากโปรแกรม Xilinx ดังนั้นในการออกแบบโปรแกรมในขั้นเริ่มแรกจะต้องตั้งค่าเริ่มต้น โดยไปที่โปรแกรม Xilinx จากนั้นเปิดโปรเจ็คที่ต้องการแล้วไปที่ Sources >> Post-Route Simulation เลือกโปรเจ็คที่เรา generate ไว้ จากนั้นไปที่ Processes >> Xilinx ISE Simulator >> Simulate Post-Place &Route Mode >> click ขวา >>เลือก Properties จะได้โปรแกรมการตั้งค่า properties ของโปรแกรมดังภาพประกอบ ค-1 โดยเลือกในส่วนของ Generate VCD file for Power Estimation เพื่อให้สามารถคำนวณการใช้กำลังงานได้ถูกต้อง



ภาพประกอบ ค-1 การตั้งค่าโปรแกรม Xilinx ส่วนของ ISE Simulator Properties

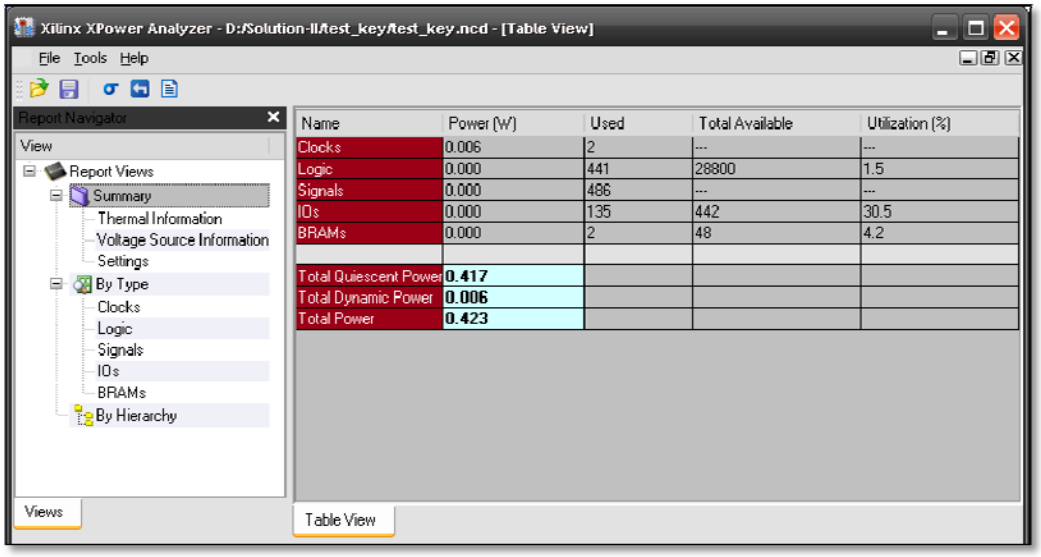
2. จากนั้น save โปรเจ็คแล้วจึง Synthesize and Implement ใหม่ แล้วก็จะได้ไฟล์ .VCD ซึ่งการทำงานของโปรแกรม Xilinx จะต้องสังเคราะห์วงจรจนกระทั่งการ post-route เสร็จสิ้น

3. จากนั้นเปิดโปรแกรม XPower เพื่อหากำลังงานที่ใช้ในวงจร เลือกชนิดไฟล์ต่างๆ ตั้งแต่ไฟล์ที่ได้ออกแบบไว้จะแสดงในชื่อไฟล์ .ncd ตลอดจนทางเลือกอื่นๆ ไม่ว่าจะเป็นการกำหนดค่าของไฟล์งานแสดงในชื่อไฟล์ .xml ข้อจำกัดทางกายภาพของไฟล์แสดงในชื่อไฟล์ .pcf และไฟล์ที่ได้จากการสังเคราะห์วงจรแสดงในชื่อไฟล์ .vcd การเลือกไฟล์แสดงในภาพประกอบ ค-2 แล้วกด ok โปรแกรมก็จะคำนวณค่ากำลังงานที่ใช้



ภาพประกอบ ค-2 การเลือกไฟล์ที่จะใช้สำหรับโปรแกรม XPower

4. เมื่อโปรแกรมคำนวณค่ากำลังงานที่ใช้เสร็จ จะแสดงให้เห็นดังภาพประกอบ ค-3



ภาพประกอบ ค-3 แสดงผลการคำนวณค่ากำลังงานของโปรแกรม XPower

ภาคผนวก ง
ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์

2011 INTERNATIONAL CONFERENCE ON EMBEDDED SYSTEMS AND INTELLIGENT TECHNOLOGY (ICESIT2011)

FEB 9, 2011 - FEB 11, 2011, at PHUKET, THAILAND


2011 INTERNATIONAL CONFERENCE
ON EMBEDDED SYSTEMS
AND INTELLIGENT TECHNOLOGY
ICESIT 2011
International Conference
on Embedded Systems
and Intelligent Technology
9 - 11 February 2011,
Prince of Songkla University,
Phuket Campus, Thailand

Energy Efficiency AES Encryption using Galois Fields

W.Wittayapanpracha and W.Suntiamorntut
Ubiquitous Networked Embedded System (UbiNES)
Department of Computer Engineering, Faculty of Engineering,
Prince of Songkla University, Hat Yai, Songkhla 90112 Thailand
Email W.wittayapanpracha@gmail.com, wannarat@coe.psu.ac.th

***Abstract*- Advanced Encryption Standard (AES) has been widely used in low-cost embedded applications. This paper analyses two implementations of S-box computation with lookup tables and Galois Field (GF). S-box development with GF has been proposed in order to avoid using table lookup and to allow the parallelism. This paper presents an efficient S-box with the multiplication of elements in $GF(2^4)^2$. The circuits have been implemented on FPGA. From the simulation, we found that S-box with GF technique gives a better power efficiency compared with LUT structure.**

Keyword- Galois Field, S-box computation, AES, FPGA

I. INTRODUCTION

Encryption is the important method for confidential protection. Many encryption algorithms such as DES, 3DES, AES and RSA, have been proposed. Most encryption algorithms are high speed and high cost suitable for high computation applications only. Therefore, AES became a good candidate for embedded applications such as wireless devices and PDA. In order to achieve the goal of low-power applications, AES development has to concern with energy-awareness.

The circuit design in real-time system has to pay attention to low-power, optimize circuit size, and least processing time. The implementation of AES on hardware is preferred more than software because the computation in hardware is faster than the implementation on software [1].

Circuit design is implemented by ASIC which consumes the high cost. A new revolution, Field Programmable Gate Array technology (FPGA), is easy to learn, low cost, and very flexible. The developer can use hardware description language in order to implement and test on it. Therefore, FPGA is an alternative implementation to evaluate the design techniques for a particular circuit.

AES consists of several function blocks. S-box computation is an important module due to it consumes resources. This operating has to perform every round. The main contribution of this paper is to present the performance and energy efficiency of two AES circuits, one is Look-Up Table (LUT) and the other is GF structure. The use of GF, binary extension fields, is to reduce the number of LUT or memory storage.

We summary the AES algorithm and the related works about GF for AES algorithm in section II and III, respectively. The AES circuits using both LUT and GF are evaluated on FPGA simulation. The performance and energy efficiency are shown in section IV. The discussion and conclusion of our work are described in section V and VI, respectively.

II. OVERVIEW OF AES ALGORITHM

AES algorithm has published by the National Institute of Standards and Technology (NIST) in 2001[3]. This algorithm is flexible, it consumes low resource and encrypt/decrypt module performs high speed. The length of symmetric key varies from 128, 192, and 256 bits. And the numbers of processing round are 10, 12, and 14 rounds that depend on security strength and resources. AES will use array sized 4x4 byte called *states*. Each state is denoted with $S_{r,c}$. The "r" denotes row and "c" denote column as shown in Figure 1.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Figure 1 Data state position.

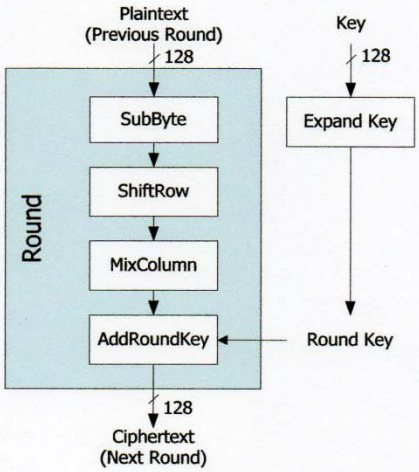


Figure 2 Round operation for 128 bits key AES encryption.

A. AES algorithm

Data state as shown in Figure 1 consists of 16 positions running from $S_{0,0}$ to $S_{3,3}$. The operation of AES algorithm has separated into 4 parts as shown in Figure 2 and enumerated each part as following:

1) Substitute Bytes Transformation

This function block is called *S-box* or *SubByte*. It has implemented by using Look-Up tables. The multiplication in Galois Fields, GF (2^8) is calculated using pre-computed Look-Up tables and some numbers of bitwise-XOR. The substitute data is hexadecimal number as shown in Table1 and represented by $S_{r,c} = \{X_x Y_y\}$ = {data from X cross Y}. For example , $S_{1,2} = \{5_x 3_y\}$ = {ed}

TABLE I
SUBSTITUTE TABLE (S-BOX TABLE).

$S_{r,c}$ = {XY}		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

2) ShiftRow Transformation

This function block is called ShiftRow. The data byte in a row is left rotated while the first row (Row 0) does not move. Second row, Row1 moves 1 byte. Third row, Row2 moves 2 bytes and final row, Row3 move 3 bytes. This operation shows in Figure 3.

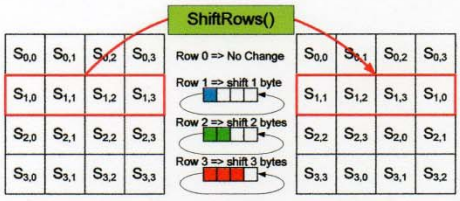


Figure 3 ShiftRows transformation operation.

3) MixColumn Transformation

MixCoulmn operation can be described in mathematical equation1 by multiplication matrix data with polynomial GF(2^8) and modulo X^4+1 with constant polynomial $a(x)$. It is called Xtimes function.

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (1)$$

Result from operation is $s'_{r,c}$ as following.

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

4) AddRoundKey Transformation

AddRoundKey operation is bitwise-XOR operation between data state ($S_{r,c}$) and key column (KC) from key expansion shows in Figure 4. The key expansion is explained in the next section.

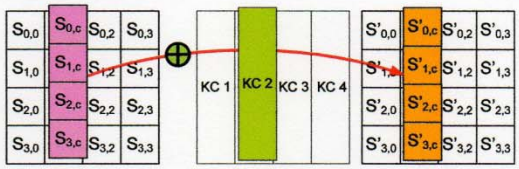


Figure 4 AddRoundKeys transformation operation.

B. Key Expansion

Key Expansion is function of generated key value in each round. The expansion of the input key into expand key block in Figure 2. Key input state takes a 4-byte input word through Rotate function. Rotate function takes cyclic permutation when KeyIn is $\{a0,a1,a2,a3\}$ output of Rotate Function is $\{a1,a2,a3, a0\}$ and send to S-box function, as discussed in section III. Rcon function uses round constant value XOR with last word key, the round constant is "00", "01", "02", "04", "08", "10", "20", "40", "80", "1b", "36", "00". Output from Rcon function through to WordColumn function, XOR with KeyIn then keep KeyOut.

In this paper, we show 128 bits – key encryption. The key and data are random to test. We are interest in SubByte and Key Expansion, which are used substitution box (S-box). Other function blocks are used methods mentioned above.

C. Related Works

As explained above, the subbyte function and key expansion function require S-box to encrypt plaintexts. The S-box is used for nine times in subbyte function and one time in key expansion function. Several papers apply any techniques in AES in order to reduce memory and decrease power usage.

Ref.[4] used GF(2^4)^2 technique to implement S-box and Key Expansion in pipelining structure. Ref.[5] presented Direct Optimized Routing with Key Storage (DOR + K) scheme, S-boxes used lookup table (LUTs).

III. GALOIS FIELDS FOR AES ALGORITHM

S-box is the function that used the mathematical transform data to other data. This architecture of SubByte is composed of 2 function blocks: multiplicative inversion in term of GF(2^8) and affine transforms [2], that shows in

Figure 5. We used $GF(2^4)^2$ to replace $GF(2^8)$ in the multiplicative inversion that explained in topic B.

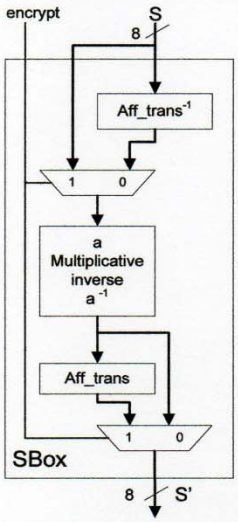


Figure 5 Inner SubByte transformation operation.

This system design shows in Figure 5 that controlled by encrypt signal to choose encryption or decryption. When encrypt = 1, work flow start when input bypass aff_trans^{-1} go to multiplicative inversion and aff_trans. Decryption, when encrypt = 0, the aff_trans-1 function opened and close aff_trans then delay for encryption is similar to decryption because circuit is likely complex.

A. $GF(2^8)$ expanded by $GF(2^4)$

We implement multiplicative inverse with simple logic circuit as affine transform which shows our architecture Figure 6 and then explain as follows each mathematic in function blocks;

$GF(2^8)$ expanded by $GF(2^4)$ in multiplicative inversion show in Figure 6. $GF(2^8)$ replaced by GF(2) and show in element of bytes which is best for hardware implementation. $GF(2^8)$ can apply to $GF((2^4)^2)$ when $a \in GF(2^8)$ that coefficient linear polynomial in $GF(2^4)$ that calculated using (2) moreover show in bijection $[a_h, a_l]$.

$$a \cong a_h x + a_l\,,\ a \in GF(2^8), \text{when } a_h, a_l \in GF(2^4) \quad (2)$$

Addition: Addition of two-term polynomial is addition of together coefficients in coordinate that calculated using (3).

$$(a_h x + a_l) \oplus (b_h x + b_l) = (a_h \oplus b_h)x + (a_l + b_l) \quad (3)$$

Multiplication: Multiplication of two-term polynomial that calculated using (4) and (5).

$$m_4(x) = x^4 + x + 1 \quad (4)$$

$$q(x) = a(x) \otimes b(x) = a(x) \cdot b(x) \bmod m_4(x), \text{ when } a(x), b(x), q(x) \in GF(2^4) \quad (5)$$

Squaring: Squaring in $GF(2^4)$ that calculated using (6).

$$q(x) = a(x)^2 \bmod m_4(x), \text{when } q(x), a(x) \in GF(2^4) \quad (6)$$

Multiplication constant: Multiplication $GF(2^4)$ with constant {e} that calculated using (7).

$$q = a \otimes \{e\} \quad (7)$$

Inverse: Inverse a^{-1} element $a \in GF(2^4)$ describes in equation $a(x) \bullet a^{-1} \bmod m_4(x) = 1$ and that calculated using (8).

$$q(x) = a(x)^{-1} \bmod m_4(x), \text{ when } q(x), a(x) \in GF(2^4) \quad (8)$$

Inversion of two-term polynomial used multiplication by inverse yields which are field element:

$$(a_h x + a_l) \otimes (a'_h x + a'_l) = \{0\}x + \{1\} \quad \text{when } a_h, a_l, a'_h, a'_l \in GF(2^4)$$

$$(a_h x + a_l)^{-1} = (a'_h x + a'_l) = (a_h \otimes d)x + (a_h \oplus a_l) \otimes d \quad (9)$$
$$d = \left((a_h^2 \otimes \{e\}) \oplus (a_h \otimes a_l) \oplus a_l^2\right)^{-1}$$

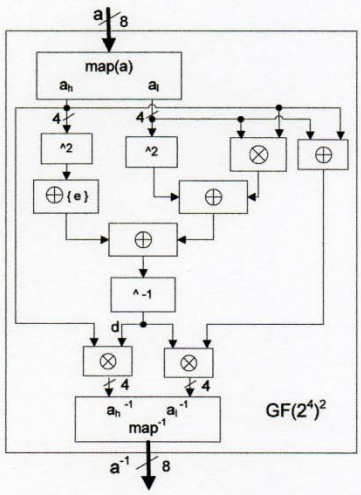


Figure 6 Inner multiplicative inversion operation

Finite fields $GF(2^8)$ is symmetry with finite fields $GF(2^4)$ in bijection. Element $a \in GF(2^8)$ that show in two-term polynomial $a_h x + a_l$ when $a_h, a_l \in GF(2^4)$ that call map function in (10).

$$a_h x + a_l = map(a), a_h, a_l \in GF(2^4), a \in GF(2^8) \quad (10)$$
$$i = a_1 \oplus a_7,\ j = a_5 \oplus a_7,\ k = a_4 \oplus a_6$$
$$a_{l0} = k \oplus a_0 \oplus a_5$$
$$a_{l1} = a_1 \oplus a_2$$
$$a_{l2} = i$$
$$a_{l3} = a_2 \oplus a_4$$

$a_{h0} = k \oplus a_5$
$a_{h1} = i \oplus k$
$a_{h2} = j \oplus a_2 \oplus a_3$
$a_{h3} = j$

Inversion of map (map^{-1}) is change two-term polynomial $a_h x + a_l$ when $a_h, a_l \in GF(2^4)$ to element $a \in GF(2^8)$ that call inv_map function in (11).

$$a = map^{-1}(a_h x + a_l), a \in GF(2^8), a_h, a_l \in GF(2^4) \quad (11)$$

$i = a_{l1} \oplus a_{h3}, j = a_{h0} \oplus a_{h1}$
$a_0 = a_{l0} \oplus a_{h0}$
$a_1 = j \oplus a_{h3}$
$a_2 = i \oplus j$
$a_3 = j \oplus a_{l1} \oplus a_{h2}$
$a_4 = i \oplus j \oplus a_{l3}$
$a_5 = j \oplus a_{l2}$
$a_6 = i \oplus a_{l2} \oplus a_{l3} \oplus a_{h0}$
$a_7 = j \oplus a_{l2} \oplus a_{h3}$

B. *Affine transformation and Inverse affine transformation*

The simple mathematic in function blocks are used in S-box, that are adapted to affine transformation and inverse affine transformation that calculated using (12) and (13). Which we are used affine transformation and inverse affine transformation referred by ref. [2] to implement the circuits in our work.

$$q = aff\_trans(a) \quad (12)$$

$i = a_0 \oplus a_1, j = a_2 \oplus a_3,$
$k = a_4 \oplus a_5, l = a_6 \oplus a_7$
$q_0 = \overline{a_0} \oplus k \oplus l$
$q_1 = \overline{a_5} \oplus i \oplus l$
$q_2 = a_2 \oplus i \oplus l$
$q_3 = a_7 \oplus i \oplus j$
$q_4 = a_4 \oplus i \oplus j$
$q_5 = \overline{a_1} \oplus j \oplus k$
$q_6 = \overline{a_6} \oplus j \oplus k$
$q_7 = a_3 \oplus k \oplus l$

$$q = aff\_trans^{-1}(a) \quad (13)$$

$i = a_0 \oplus a_5, j = a_1 \oplus a_4,$
$k = a_2 \oplus a_7, l = a_3 \oplus a_6$
$q_0 = \overline{a_5} \oplus k$
$q_1 = a_0 \oplus l$
$q_2 = \overline{a_7} \oplus j$
$q_3 = a_2 \oplus i$
$q_4 = a_1 \oplus l$
$q_5 = a_4 \oplus k$
$q_6 = a_3 \oplus i$
$q_7 = a_6 \oplus j$

IV. Experimental Results

A. *Test bed*

The system design of AES algorithm is implemented in VHDL language, simulated on Xilinx ISE 10.1, and designed tool by Virtex-5 device XC5VLX50 package ff676 and speed -3. The simulation random 128 bits plaintexts and 128 bits key to evaluate performance.

In this paper designed for 128 bits data when AES system starts the data plaintext and the key are through as the same time. This AES is loop architecture, that use counter for count round. In each round, key expansion generate key round send to system for change data and hard to known the plaintext. When finish each round, the output send to input again and counter count to the next round. This AES used 128 bits key then counter count 10 rounds and loop is 10 times.

B. *Performance Metrics*

1) Throughput: It is measurement the number of cipher bits in the duration time that calculated using (14).

$$throughput = \frac{BS \times DB}{CC \times CP} \quad (14)$$

When BS is the number of bits per block, DB denotes data block of plaintext for AES encryption, CC is the number of clock cycle that represent the processing time, and CP denotes time period per clock. The best circuit should produce the high throughput.

2) Resource efficiency: It indicates worth of slices (refer to resources) in the term of throughput that measured by ratio of throughput and number of slices. Any circuit take resource equal with other circuit and provide the higher throughput, we conclude that it is the high efficiency.

3) Power efficiency: It indicates worth of power usage in term of throughput that measure by ratio of throughput and power usage. As resource efficiency, when the throughput per power usage is high, we conclude that our system design is high power efficiency.

C. *Scenario I*

The AES algorithm uses S-box in 2 modules: encryption core and key expansion. The encryption core transforms plaintext with S-box and key expansion generates round key S-box. In table I show performance of simulation results when we substitute the encryption and key expansion with LUTs (denoted with L) and Galois fields (denoted with G).

TABLE II
PERFORMANCE OF GF IN AES ALGORITHM

type	Slices	Throughput (Mbps)	Efficiency (Mbps/Slices)	Power (mW)	Power Efficiency (Mbps/mW)
LL	4325	1251	0.289	575	2.175
LG	7125	320	0.045	553	0.578
GL	4532	898	0.198	439	2.045
GG	7862	1184	0.150	408	2.901

- LG means that key expansion with LUTs and encryption with Galois fields
- L = LUTs, G = Galois Fields

The results in table II show that the simple AES algorithm (LL) use the resource less than GG 44.98% but produce throughput more than GG only 5.66% and take the power up to 40.93%. In power efficiency perspective, GG gives the efficiency more than LL up to 33.38%.

We conclude that S-box in AES algorithm is implemented with Galois fields both encryption core and key expansion gives the highest power efficiency when we compare with other mode which composed of LUTs and Galois fields. However, the new technique provides throughput less than LUTs only 5.66%.

D. Scenario II

As explained in section II, many techniques are proposed to improve performance of AES on FPGA. In this section, we compare our technique (LL and GG) with related works. In table III, all our techniques are represented in technique name followed with (1) that used in loop architecture. While results that used 4 types and pipeline architecture from Tanzilure Rahman and et al [4] are shown in (2) and Jason Van Dyken and et al [5] are represented in (3) that used LUTs in loop architecture.

TABLE III
PERFORMANCE COMPARISON OF
OUR TECHNIQUE AND RELATED WORKS

	Slice	Throughput (Mbps)	Efficiency (Mbps/Slices)	Power (mW)	Power Efficiency (Mbps/mW)
LL(1)	4325	1251.00	0.289	575.00	2.175
LG(1)	7125	320.00	0.045	553.00	0.578
GL(1)	4532	898.00	0.198	439.00	2.045
GG(1)	7862	1184.00	0.151	408.00	2.901
LL(2)	7606	2190.00	0.288	n/a	n/a
LG(2)	6659	1110.00	0.167	n/a	n/a
GL(2)	6385	2320.00	0.363	n/a	n/a
GG(2)	5207	1110.00	0.213	n/a	n/a
LL(3)	2439	349.70	0.143	778.84	0.449

- L = LUTs, G=Galois Fields
- 1) is our technique, 2) pipeline architecture [4], 3) Loop architecture [5]

When we compare the throughput between [4] and our technique we found that our technique provide throughput less than ref. [4] because the they applied pipeline and Galois field to improve performance of AES on FPGA. However, Cost of resources in ref. [4] higher than our technique because they use more slices.

In energy perspective, we found that the Galois fields minimize memory and power usage compare to LUTs technique in ref. [5]. Although cost of Galois fields is higher than LUTs to 222.34% but throughput is increased up to 238.58% and energy is decreased up to 47.65%.

V. DISCUSSION

The performance evaluation in term of resource efficiency indicates worth of resource cost. The S-box implementation with Galois field use more slices because more requirement of logic gate for real time S-box generation. As result, the slice of LUTs technique is less than Galois field. On the other hand, the cost of Galois field technique more than LUTs.

TABLE IV
TOP 3 OF BEST RESOURCE EFFICIENCY

	Slices	Throughput	Resource Eff.	Power	Power Eff.
1	LL(3)	GL(2)	GL(2)	GG(1)	GG(1)
2	LL(1)	LL(2)	LG(1)	GL(1)	LL(1)
3	GL(1)	LL(1)	LL(2)	LG(1)	GL(1)

In table IV, our throughput less than Galois field technique in ref. [4] because ref [4] improve throughput with pipeline technique in order to decrease number of clock cycle that bring to the increasing throughput rate.

Because ref [4] does not discuss energy issue then we need to ignore this technique and we discuss and compare to ref. [5] only. By simulation, our approach consumes power less than LUTs and produces the higher throughput.

VI. CONCLUSION

This paper has implemented S-box of AES with Galois field technique and simulate to show that encryption and key expansion are implemented with Galois field consume energy less than LUTs. And it brings to the high energy efficiency.

However, our technique produces lower throughput when compare with Galois field and pipeline technique.

For future works, we should improve throughput while our technique still consume low power like this paper.

REFERENCES

[1] M Healy, T Newe and E Lewis, Efficiently security data on a wireless sensor network, Sensor and their Applications XIV (SENSORS07).IOP Publishing Ltd,2007.

[2] Johnnes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger, An ASIC Implementation of the AES SBoxes, CT-RSA 2002, LNCS 2271, pp. 67-78, 2002.

[3] National Institute of Standards and Technology (NIST): Advanced Encryption Standard (AES), FIPS-197, 2001.

[4] Tanzilur Rahman, Shengyi Pan, Qi Zhang, Design of a High Throughput 128-bit AES (Rijndael Block Cipher),Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 Vol II, IMECS 2010.

[5] Jason Van Dyken, José G. Delgado-Frias, FPGA schemes for minimizing the power-throughput trade-off in executing the Advanced Encryption Standard algorithm, Journal of Systems Architecture, Volume 56 , Issue 2-3 (February 2010),Pages: 116-123

ประวัติผู้เขียน

ชื่อ สกุล นางสาววรรณจันทน์ วิทยาพันธ์ประชา

รหัสประจำตัวนักศึกษา 5110120038

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2551

ทุนการศึกษา (ที่ได้รับในระหว่างการศึกษา)

- ทุนศิษย์ก้นกุฏิ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ประจำปีการศึกษา 2551

การตีพิมพ์เผยแพร่ผลงาน

- W.Wittayapanpracha and W.Suntiamorntut, "*Energy Efficiency AES Encryption using Galois Fields*," In Proceedings of 4th International Conference on Embedded System and Intelligent Technology 2011 (ICESIT2011), Phuket, Thailand, 12th – 14th May 2010, pages 154-158.