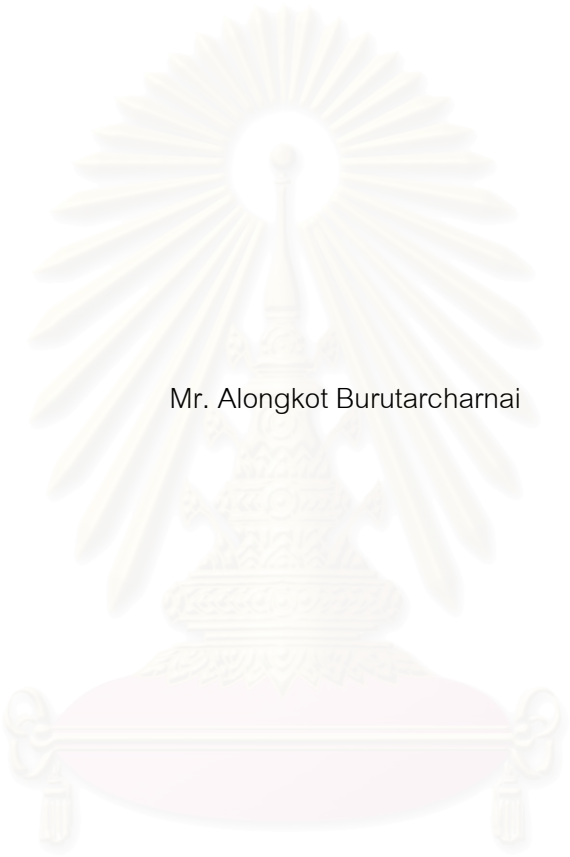
การออกแบบหน่วยประมวลผลฝังตัวสำหรับการใช้งานอินเทอร์เน็ตด้วยเกณฑ์วิธีไอซีเอ็มพี

นายอลงกต บุรุษอาชาไนย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2548
ISBN 974-53-2885-5

DESIGN OF AN EMBEDDED PROCESSOR FOR INTERNET APPLICATIONS WITH ICMP PROTOCOL

Mr. Alongkot Burutarcharnai



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University

หัวข้อวิทยานิพนธ์	การออกแบบหน่วยประมวลผลฝังตัวสำหรับการใช้งานอินเทอร์เน็ตด้วยเกณฑ์วิธีไอซีเอ็มพี
โดย	นายอลงกต บุรุษอาชาไนย
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ บุญชัย โสวรรณวณิชกุล)

..... อาจารย์ที่ปรึกษา
(รองศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. เขมทัต วิภาตะวนิช)

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

นายอลงกต บุรุษอาชาไนย : การออกแบบหน่วยประมวลผลฝังตัวสำหรับการใช้งานอินเทอร์เน็ตด้วยเกณฑ์วิธีไอซีเอ็มพี. (DESIGN OF AN EMBEDDED PROCESSOR FOR INTERNET APPLICATIONS WITH ICMP PROTOCOL) อ.ที่ปรึกษา : รศ.ดร. ประภาส จงสถิตย์วัฒนา, 74 หน้า. ISBN 974-53-2885-5.

วิทยานิพนธ์นี้เสนอวิธีการออกแบบหน่วยประมวลผลเพื่อใช้งานอินเทอร์เน็ตด้วยเกณฑ์วิธีไอซีเอ็มพีซึ่งเป็นพื้นฐานที่สำคัญในการติดต่อในระบบเครือข่ายโดยระบบฝังตัวประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ 3 ชิ้นคือ ตัวประมวลผล หน่วยความจำ และ ตัวควบคุมการติดต่อกับระบบเครือข่าย มีการพัฒนาหน่วยประมวลผลที่มีความเรียบง่าย และใช้ภาษาเครื่องที่ประหยัดด้วยหน่วยประมวลผลแบบแสตกขนาด 16 บิตเพราะข้อมูลต่างๆ ที่เข้ามาจากระบบเครือข่ายจะถูกประมวลผลได้เร็วขึ้นในระบบ 16 บิต มีการสร้างระบบ TCP/IP แสตกให้กับหน่วยประมวลผลสามระดับคือชั้นกายภาพ ชั้นเชื่อมโยงข้อมูล และ ชั้นเครือข่าย ระบบสามารถตอบสนองระบบเครือข่ายได้ดีตามมาตรฐานอีเทอร์เน็ตตามเกณฑ์วิธี ICMP และใช้ทรัพยากรน้อย

ภาควิชา.... วิศวกรรมคอมพิวเตอร์.....	ลายมือชื่อนิสิต.....อลงกต บุรุษอาชาไนย.....
สาขาวิชา....วิศวกรรมคอมพิวเตอร์.....	ลายมือชื่ออาจารย์ที่ปรึกษา.....ประภาส จงสถิตย์วัฒนา.....
ปีการศึกษา .....2548.....	ลายมือชื่ออาจารย์ที่ปรึกษาร่วม.....

4670600721 : MAJOR Computer Engineering

KEY WORD: EMBEDDED SYSTEMS / HARDWARE DESIGN / ICMP RESPONSE / TCP/IP / MICRO CONTROLLER / NETWORK INTERFACE

ALONGKOT BURUTARCHARNAI : DESIGN OF AN EMBEDDED FOR INTERNET APPLICATIONS WITH ICMP PROTOCOL. THESIS ADVISOR : ASSOC. PROF. PRABHAS CHONGSTITVATANA, 74 pp. ISBN 974-53-2885-5.

This thesis proposes a design of a processor for internet applications with ICMP protocol, which is the most important basis for internet connection. This embedded system composed of three electronics devices: processor, memory and network interface controller. A 16-bit simple stack processor was developed with a compact instruction set. The most incoming data can be processed faster if it can be processed in 16-bit form. Software for TCP/IP stack was developed with the first three layers namely, physical link layer, data link layer and network layer. The system can effectively response to Ethernet standard by ICMP and it uses resource efficiently.

Department..... Computer Engineering....	Student's signature... A. Burutarcharnai
Field of study.... Computer Engineering....	Advisor's signature..... P. Chongstitvatana
Academic year .....2005.....	Co-advisor's signature.....

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จออกมาได้ด้วยความกรุณาของ รศ.ดร.ประภาส จงสถิตย์วัฒนา อาจารย์ที่ปรึกษา ซึ่งนอกจากจะช่วยให้คำแนะนำและความคิดเห็นเกี่ยวกับงานวิจัยแล้ว ยังสอนความรู้อื่นๆ อีกมากมายซึ่งมีส่วนในการเปิดโลกทัศน์การเรียนรู้ของข้าพเจ้าเป็นอย่างมาก

ขอขอบคุณ ผู้ช่วยศาสตราจารย์ บุญชัย โสวรรณวณิชกุล ที่ให้ความรู้และคำปรึกษาทางด้านฮาร์ดแวร์ต่างๆ และขอขอบคุณ อาจารย์ เกริก ภิรมย์โสภาที่ช่วยให้คำปรึกษาทางอีเมล์เกี่ยวกับการทำงานในโครงงานต้นแบบ

ขอขอบคุณ ดร. ราชพร เขียนประสิทธ์ที่สนับสนุนทางด้านอุปกรณ์การเชื่อมต่อและให้คำปรึกษาเกี่ยวับการเชื่อมต่อต่างๆจนเสร็จสมบูรณ์

ขอขอบคุณเพื่อนๆ พี่ๆ และน้องๆ ทุกคนในภาควิชาที่ช่วยแก้ปัญหาเบ็ดเตล็ดต่างๆ ทั้งที่เกี่ยวกับงานวิจัยและนอกเหนือจากงานวิจัย

และเหนือสิ่งอื่นใดข้าพเจ้าคงไม่มีทางทำงานวิจัยนี้สำเร็จถ้าขาดพระคุณของคุณพ่อและคุณแม่ที่ให้กำเนิด อบรมเลี้ยงดู สอนความรู้ ให้โอกาส และห่วงใยดูแลข้าพเจ้าเสมอมา

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพ

รูปที่ หน้า

สารบัญภาพ (ต่อ)

รูปที่ หน้า

สารบัญตาราง

ตารางที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันเทคโนโลยีต่างๆ ได้มีการใช้ระบบอัตโนมัติมากขึ้นจึงทำให้เทคโนโลยีระบบฝังตัวได้มีอิทธิพลเข้ามาในชีวิตประจำวันเป็นอย่างมากไม่ว่าจะเป็นในระบบสื่อสาร ระบบการขับขี่ยานพาหนะหรือว่าอุปกรณ์ไฟฟ้า[1] และในปัจจุบันบริษัทอิเลคทรอนิกส์ส่วนใหญ่ ได้มีการออกแบบและพัฒนาระบบควบคุมขนาดเล็ก (MicroController) กันมากขึ้น แม้ว่าบริษัทเหล่านั้นจะพัฒนาระบบควบคุมต่างๆออกมาแต่ก็ยังไม่มีการพัฒนาระบบประมวลผลภายใน (Core CPU) ขึ้นมาใช้เอง ดังนั้นจึงมีแนวคิดที่จะออกแบบระบบประมวลผลขนาดเล็กขึ้นมาเพื่อใช้ในการพัฒนาระบบควบคุมขึ้นมาใช้เอง ซึ่งจะเป็นแรงกระตุ้นให้มีการพัฒนาและออกแบบวงจรใช้เองมากขึ้น เนื่องจากระบบควบคุมแบบฝังตัวเป็นระบบที่เหมาะสมในการควบคุมงานเฉพาะทาง แต่ระบบเหล่านี้มักเป็นระบบที่ไม่มีการสื่อสารกันกับอุปกรณ์ควบคุมภายนอก จึงทำให้การควบคุมและการใช้งานต้องกระทำผ่านเครื่องมือหรืออุปกรณ์เฉพาะของผู้ออกแบบหรืออุปกรณ์เฉพาะของผู้ออกแบบหรือผู้ผลิตแต่ละรายเท่านั้น

จากปัญหาและแรงกระตุ้นดังกล่าวจึงนำมาสู่แนวความคิดในการออกแบบหน่วยประมวลผลสำหรับระบบฝังตัว โดยประกอบด้วยการพัฒนาหน่วยประมวลผลแบบแสตกขนาด 16 บิต เพื่อเน้นสถาปัตยกรรมที่มีความซับซ้อนน้อย เพื่อสามารถที่จะสังเคราะห์ลงบนวงจรรวมจริงได้ เพราะสถาปัตยกรรมภายในแผงวงจรทดลองกับสถาปัตยกรรมของวงจรรวมเฉพาะงาน (ASIC) นั้นยังมีความแตกต่างกัน [2] หน่วยประมวลผลแบบแสตกนั้นมีความซับซ้อนน้อยกว่าหน่วยประมวลผลแบบเรจิสเตอร์ไม่ว่าจะเป็นแบบ CISC หรือ RISC สถาปัตยกรรมพื้นฐานที่สุดของหน่วยประมวลผลแบบแสตกไม่มีการใช้ฮาร์ดแวร์เก็บค่าต่างๆ ในการทำงานและยังสามารถรองรับการทำงานด้านการเรียกโปรแกรมย่อยได้ดีด้วย เพราะไม่ต้องไปสร้างส่วนที่เป็นแสตกขึ้นมาเพื่อรองรับการทำงานการเรียกโปรแกรมย่อยอีก และหน่วยประมวลผลแบบแสตกยังมีขนาดโปรแกรมที่เล็กกว่าโปรแกรมจากหน่วยประมวลผลแบบเรจิสเตอร์ด้วย เนื่องจากโปรแกรมที่เก็บนั้นไม่มีการเก็บค่าของตัวที่ถูกดำเนินงาน โดยหน่วยประมวลแบบแสตกจะทำงานกับหน่วยความจำโดยตรงจึงไม่ต้องใช้เก็บเรจิสเตอร์ในการคำนวณต่างๆ ซึ่งประโยชน์ของโปรแกรมขนาดเล็กนั้นยังทำให้ใช้พื้นที่ในการเก็บขนาดเล็กลงด้วยจึงมีพื้นที่หน่วยความจำเหลือในการประมวลผลการทำงานอย่างอื่นได้มากขึ้น [3]

การเชื่อมต่อกับระบบควบคุมแบบฝังตัวเข้าสู่ระบบเครือข่ายอินเทอร์เน็ตผ่านทางเครื่องบริการเว็บแบบฝังตัวนั้น มีปัจจัยพื้นฐานที่ต้องตระหนักถึงเช่น

- การสร้างระบบ TCP/IP
- ความเร็วในการตอบสนอง

และในการพัฒนาระบบควบคุมแบบฝังตัวทั้งระบบนั้นจะมีปัจจัยพื้นฐานที่ต้องคำนึงอยู่หลายประการอาทิเช่น

- สมรรถนะของหน่วยประมวลผล
- การเชื่อมต่อของหน่วยประมวลผลกับอุปกรณ์เชื่อมต่อระบบเครือข่าย

จากปัญหาและแนวคิดดังกล่าวจึงได้เกิดงานวิจัยนี้ขึ้นเพื่อพัฒนาระบบประมวลผลแบบฝังตัวเพื่อการใช้งานอินเทอร์เน็ต

1.2 วัตถุประสงค์

1. เพื่อพัฒนาหน่วยประมวลผลแบบแสตกที่ใช้ในระบบฝังตัว
2. พยายามลดชิ้นส่วนในการเชื่อมต่อให้น้อยที่สุดโดยพยายามย้ายส่วนที่ย้ายได้มากที่สุดให้เข้ามาอยู่ภายในวงจรที่สามารถสร้างขึ้นมาได้ เพื่อความประหยัดและความรวดเร็ว

1.3 ขอบเขตงานวิจัย

1. พัฒนาหน่วยประมวลผลแบบแสตก 16 บิตด้วยภาษาเวอร์ริล็อก
2. พัฒนาอุปกรณ์สื่อสารด้านระบบเครือข่ายโดยอาศัยมาตรฐาน IEEE802.3 อีเทอร์เน็ต 10BaseT ในการเชื่อมต่อเข้ากับระบบเครือข่ายโดยใช้หน่วยประมวลผลที่พัฒนาขึ้น
3. พัฒนา TCP/IP Stack เพื่อรองรับเกณฑ์วิธี ICMP แบบสื่อสารสองทางครึ่งอัตรา (half duplex) โดยทดสอบเฉพาะ PING (Packet Internet Grope) ซึ่งใช้เกณฑ์วิธี ICMP ชนิด 0 และชนิด 8

1.4 ขั้นตอนและวิธีดำเนินงานวิจัย

1. พัฒนาหน่วยประมวลผลให้ทำงานได้ในบอร์ดเอฟพีจีเอ และทำการทดสอบการทำงานของหน่วยประมวลให้ครบทุกคำสั่ง

2. สังเคราะห์วงจรลงบนบอร์ดเอฟพีจีเอ
3. ทำการพัฒนาตัวแปลภาษาให้มีความสัมพันธ์กับหน่วยประมวลผล
4. พัฒนาโปรแกรมเพื่อการสื่อสารภายในระบบเครือข่าย
5. ทดสอบการทำงานและวัดประสิทธิภาพ
6. สรุปผลการทดลองและเขียนวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถพัฒนาระบบประมวลผลแบบแสตกเหมาะกับการใช้งานอินเทอร์เน็ตสามารถพัฒนาระบบประมวลผลแบบแสตกเหมาะกับการใช้งานอินเทอร์เน็ต
2. สามารถพัฒนาระบบ TCP/IP ที่มีขนาดเล็กและเพียงพอต่อการใช้งานพื้นฐาน
3. ใช้เป็นแนวทางในการพัฒนาหน่วยประมวลผลที่ใช้ในระบบฝังตัว
4. เพิ่มทักษะในการออกแบบวงจรและสังเคราะห์ลงบนวงจรรวมขนาดเล็กเพื่อนำไปใช้ในวงจรขนาดใหญ่

1.6 ลำดับการจัดเรียงเนื้อหาในวิทยานิพนธ์

วิทยานิพนธ์นี้แบ่งเนื้อหาออกเป็น 6 บทดังนี้ บทที่ 1 เป็นบทนำซึ่งกล่าวถึงที่มาและความสำคัญของปัญหา รวมทั้งวัตถุประสงค์ของงานวิจัย บทที่ 2 สรุปงานวิจัยที่เกี่ยวข้องในด้านการลดขนาดโปรแกรม บทที่ 3 กล่าวถึงรายละเอียดของอุปกรณ์พื้นฐานโดยจะอธิบายการทำงานพื้นฐานของแสตก (Stack) และนำเสนอรายละเอียดของชุดคำสั่งและหน่วยประมวลผลและอุปกรณ์สื่อสารในเครือข่ายที่เลือกใช้ บทที่ 4 กล่าวถึงมาตรฐานเกณฑ์วิธีและทฤษฎีที่เกี่ยวข้องโดยรวมถึงรายละเอียดการเขียนโปรแกรม การทดลอง วิธีการทดลอง การใช้เครื่องมือต่างๆในการทดลอง และผลการทดลองแสดงไว้ในบทที่ 5 และสุดท้ายข้อสรุปจากการวิจัยถูกกล่าวถึงในบทที่ 6

1.7 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ได้รับการตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง "Design of an embedded TCP/IP internet appliance" โดย อลงกต บุรุษอาชาไนย ภานุพันธ์ นันทนาวุฒิ และประภาส จงสถิตย์วัฒนา ในงานประชุมวิชาการ NCSEC ครั้งที่ 8 ซึ่งจัดโดยมหาวิทยาลัยหอการค้าไทย กรุงเทพ ระหว่างวันที่ 27-28 ตุลาคม 2548

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ได้รับการตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง “A stack-based processor for resource efficient embedded system” โดย อลงกต บุรุษอาชาไนย ภานุพันธ์ นันทนาวุฒิ และประภาส จงสถิตย์วัฒนา ในงานประชุมวิชาการ TenCon ซึ่งจัดโดย IEEE ที่โรงแรมโลตัสบางสวนแก้วเชียงใหม่ ระหว่างวันที่ 21-24 พฤษภาคม 2547

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ได้รับการตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง “การออกแบบส่วนควบคุมที่ใช้สัญญาณนาฬิกาสองเฟสเพื่อเพิ่มประสิทธิภาพของหน่วยประมวลผลแบบแสตก” โดย อลงกต บุรุษอาชาไนย ภานุพันธ์ นันทนาวุฒิ และประภาส จงสถิตย์วัฒนา ในงานประชุมวิชาการ NCSEC ครั้งที่ 7 ซึ่งจัดโดยมหาวิทยาลัยสงขลานครินทร์ ที่โรงแรม เจ บี หาดใหญ่ สงขลา ระหว่างวันที่ 21-22 ตุลาคม 2547

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ได้รับการตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง “A fast instruction fetch unit for an embedded stack processor“ โดย อลงกต บุรุษอาชาไนย วิษณุ โคตรจรัส และประภาส จงสถิตย์วัฒนา ในงานประชุมวิชาการ ICT2004 ครั้งที่ 11 กรุงเทพ 2547

บทที่ 2

งานวิจัยที่เกี่ยวข้อง

บทนี้กล่าวถึงงานวิจัยที่เกี่ยวกับผลิตภัณฑ์ที่ถูกใช้เป็นเครื่องบริการเว็บและในโครงงานของพัฒนาระบบเครื่องบริการเว็บแบบฝังตัวที่สามารถจัดรูปแบบใหม่ได้นี้เป็นแรงกระตุ้นในการออกแบบและพัฒนาโครงงานนี้ขึ้นมาด้วย ซึ่งมุ่งหาวิธีการพัฒนาโปรแกรมประยุกต์และการเชื่อมต่อทางด้านอิเล็กทรอนิกส์ที่ใช้ในระบบฝังตัวที่มีอยู่ในปัจจุบันและออกแบบให้รองรับกับการใช้งานในอนาคตด้วย

2.1 การพัฒนาระบบเครื่องบริการเว็บแบบฝังตัวที่สามารถจัดรูปแบบใหม่ได้

งานวิจัยที่กล่าว[1] ถึงนี้มีการใช้ไมโครคอนโทรลเลอร์ขนาด 8 บิต ตระกูล 8051 เป็นตัวประมวลผลหลักและมีการเชื่อมต่อกับอุปกรณ์ควบคุมการเชื่อมต่อระบบเครือข่ายมาตรฐานตามอีเทอร์เน็ตรุ่น DP83902 ของบริษัทเนชั่นแนลเซมิคอนดักเตอร์ โดยที่โครงงานนี้มีการพัฒนาซอฟต์แวร์เพื่อใช้ในการเชื่อมต่อระบบเครือข่ายบนเกณฑ์วิธี TCP/IP เพื่อการทำเป็นเครื่องบริการเว็บ และมีการพัฒนาให้รองรับการทำงานทางด้าน TCP UPD ICMP HTTP และ PHP รวมถึงมีการสร้างระบบรักษาความปลอดภัยโดยการพิสูจน์ตัวจริงแบบเบื้องต้น (Basic Authentication)

2.2 AVR460: Embedded Web Server

โครงงานนี้เป็นผลิตภัณฑ์ทางการค้าเพื่อใช้ในระบบบ้านฉลาดเพื่อให้อุปกรณ์เครื่องใช้ต่างๆภายในบ้านสามารถติดต่อกับระบบเครือข่าย และสามารถรองรับการทำงานทางด้านต่างๆ ผ่านระบบเครือข่ายได้อย่างสมบูรณ์ โดยมี AVR® [4] เป็นหน่วยประมวลผลกลางในการพัฒนา โดย มีใช้ระบบการสื่อสารภายในกับหน่วยประมวลผลแบบ 8 บิต ผลิตภัณฑ์นี้ได้มีการสร้างระบบ TCP/IP และมีส่วนต่อประสานไว้ติดต่อกับอุปกรณ์ภายนอกไว้เรียบร้อยแล้วนอกจากนี้ยังมีการรองรับการรับ-ส่งเมล์ (SNMP) เทลเนท (Telnet) และ มีการรับส่งไฟล์ (FTP) อีกด้วยซึ่งโค้ดทั้งหมดถูกเขียนโดยภาษาซีซึ่งทำให้ผู้ใช้ง่ายต่อการพัฒนา

2.3 WWWpic2 (PIC)

โครงงานนี้มีการทำขึ้นเป็นงานอดิเรก มีการเชื่อมต่ออุปกรณ์ต่างๆ ภายในแผงวงจรอย่างง่ายๆ เพราะมีการใช้อุปกรณ์หลักๆ เพียง หน่วยประมวลผลตระกูล PIC กับ EEPROM เบอร์

24LC256 เท่านั้น [5] มีการสื่อสารกับระบบเครือข่ายผ่านสายอนุกรม จึงต้องสร้างระบบเพื่อให้รองรับการทำงานของการรับ-ส่งแบบอนุกรม ตามเกณฑ์วิธีอินเทอร์เน็ตแบบสายอนุกรม (SLIP) การรับ-ส่งของ IP การรับ-ส่งของ TCP และการดึงข้อมูลของ HTTP

2.4 Tiny

มีต้นกำเนิดและพัฒนามาจาก iPic 12C509A ขนาด 8 บิตจาก University of Massachusetts ซึ่งมีชื่อว่า "tinest" โดยมีการสร้างระบบ TCP/IP แบบแสตกและมีการใช้ความถี่สัญญาณนาฬิกาเพียง 4 เมกะเฮิรท์ และใช้เนื้อที่สร้างส่วนทีซีพีไอพีแสตกเพียง 256 ไบต์ [6] และมีการสร้างให้ระบบรองรับการทำงานของ TCP และ UPD ซึ่งใช้เพียงแค่ 79-90 คำสั่งเท่านั้น และสร้างให้ระบบรองรับการทำงานของ ICMP เพียงแค่ 14 คำสั่งและระบบที่สร้างขึ้นมีการรองรับ IPv4 ซึ่งใช้คำสั่งการทำงานอยู่ที่ 68-77 คำสั่งและมีการรองรับ HTTP เวอร์ชัน 1.0 ซึ่งตัวโปรแกรมเครื่องบริการเว็บทั้งหมดที่กล่าวมานี้สามารถเก็บลงบน EEPROM เบอร์ 24LC256 การเชื่อมต่อจะเชื่อมต่อกับภายนอกโดยใช้เกณฑ์วิธีอินเทอร์เน็ตแบบสายอนุกรม (SLIP) แต่มีปัญหาในการใช้งานคือ มีการเริ่มต้นการใช้งานที่ช้าเพราะมีการสร้างระบบ TCP/IP แบบแสตกในพื้นที่หน่วยความจำที่จำกัดแต่มีการส่งข้อมูลที่เร็วเพราะใช้การส่งข้อมูลขนาดเล็กๆ

2.5 Stanford Match Box Web Server - match box PC

เป็นการสร้างระบบเครื่องบริการเว็บบนคอมพิวเตอร์ส่วนบุคคลตระกูล 486 DIMM จากห้องวิจัยของมหาวิทยาลัยสแตนฟอร์ด[7] โดยใช้ระบบปฏิบัติการ Red Hat 5.2 ในการจัดการระบบต่างๆ ทั้งหมดแทนที่จะสร้างเพียงแต่ระบบทีซีพีไอพีอย่างเดียวซึ่งวิธีการสร้างแบบนี้เป็นวิธีการที่ง่ายๆ เหมือนกับการใช้โปรแกรม Apache เวอร์ชันเล็กๆ แต่ว่าสิ้นเปลืองทรัพยากรจำพวกหน่วยความจำในการเก็บข้อมูลค่อนข้างมาก แต่ก็แลกมาด้วยการทำงานบริการการเรียกร้องทางด้านเครือข่าย (requests) ต่างๆ ได้หลากหลายกว่าด้วย

2.6 Rt-Control uCsimm - 3.5 x 1 inches

ตัวบริการเว็บนี้ (uCsimm) มีการใช้หน่วยประมวลผลของโมโตโรล่าเบอร์ uC68EZ328 ใช้ QVGA ในการแสดงผล และมีการใช้ SPI Serial, Parallel I/Oในการเชื่อมต่อภายในและมาตรฐาน 10baseT ในการเชื่อมต่อกับเครือข่ายภายนอก ยังมีการรองรับการสื่อสารผ่านสายอนุกรมไว้ด้วย [8] และระบบการเชื่อมต่อทั้งหมดในแผงวงจรมีขนาด 3.5 x 1 นิ้ว และใช้ระบบปฏิบัติการ uClinux เพื่อสร้างเครื่องบริการเว็บขนาดเล็กด้วย

2.7 ADM6996 Ethernet Switch

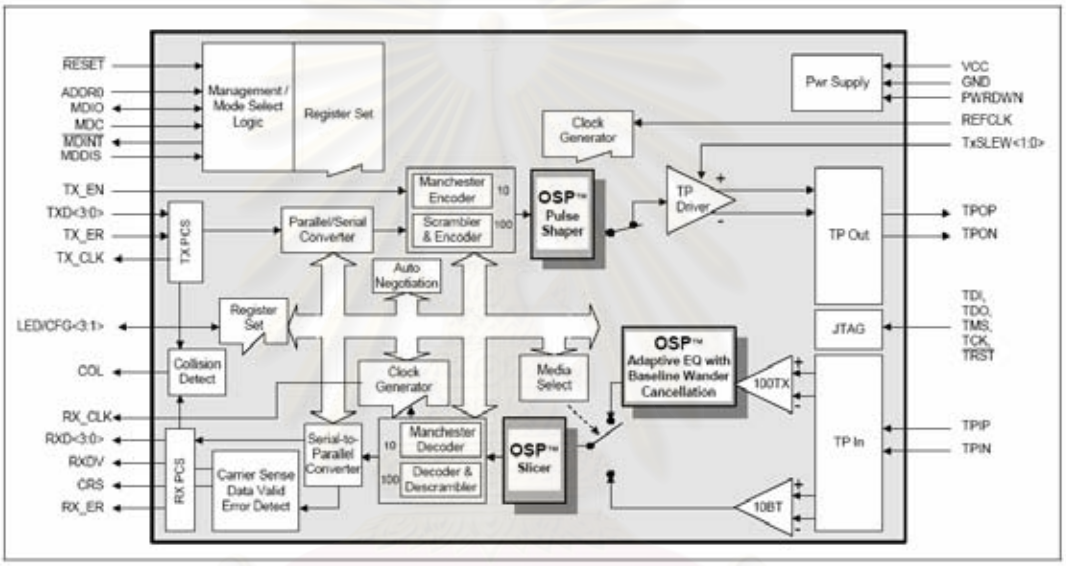
ในโครงงานนี้ได้ออกแบบอุปกรณ์ Switch 4 พอร์ต ขึ้นมาใช้เองโดยเป็นโครงงานของเนคเทค [9] โดยใช้หน่วยประมวลผลขนาด 16 บิตตระกูลอาร์ม9 รุ่น ADM6996 [10] มาเป็นหน่วยประมวลผลหลักซึ่งมีวงจรที่ใช้ในการสื่อสารกับอุปกรณ์ภายนอกตามมาตรฐานเตรียมไว้ให้แล้ว การเชื่อมต่อกับอุปกรณ์เกี่ยวกับระบบเครือข่ายจึงเชื่อมไม่ยุ่งยากและไม่ต้องสร้างวงจรที่ใช้ในการสื่อสารกับอุปกรณ์เชื่อมต่อเครือข่ายและโครงงานนี้ใช้ระบบปฏิบัติการ linux และใช้ภาษาซีในการพัฒนาและสื่อสารกับอุปกรณ์ควบคุมเครือข่ายตามาตรฐาน MII [11]

จากหน่วยประมวลผลที่ยกตัวอย่างมาเห็นได้ว่าส่วนใหญ่แล้วสร้างเป็นแบบ 8 บิตอาจจะเพื่อความประหยัด ความง่ายในการทำการตลาดและการเชื่อมต่อระหว่างอุปกรณ์ควบคุมเครือข่ายเข้ากับหน่วยประมวลผลนั้นๆ แต่เนื่องจากการทำงานส่วนใหญ่ของ TCP/IP แบบแสตกจะทำงานได้เร็วขึ้นถ้าทำงานครั้งละ 16 บิตเช่นในการคำนวณค่าผลรวมตรวจสอบ (Check Sum) เช่นในหัวข้อ4.2 และจากโครงงานที่ยกมาเป็นตัวอย่างนี้เห็นได้ชัดว่าทุกโครงงานมีการใช้อุปกรณ์ทางอิเล็กทรอนิกส์เป็นจำนวนมากซึ่งตัวที่น้อยที่สุดเป็นของ WWWpic2 ที่ใช้อุปกรณ์พวกวงจรรวมหลักๆ เพียง 2 ตัว เนื่องจากเป็นการเชื่อมต่อกับสายอนุกรมจึงไม่ต้องมีการสร้างระบบ TCP/IP แบบแสตกที่ซับซ้อน จึงทำให้ลายวงจรไม่ซับซ้อนมากและข้อดีอีกประการหนึ่งในการออกแบบและสังเคราะห์หน่วยประมวลผลขึ้นมาใช้เองคือสามารถลดจำนวนอุปกรณ์ที่ใช้ในการเชื่อมต่อได้ เช่นมีการย้ายหน่วยความจำเข้าไปภายในหน่วยประมวลผลที่จะสังเคราะห์ขึ้นเป็นวงจรรวม การลดการใช้อุปกรณ์พวกNot Gate หรือทำการสังเคราะห์ระบบ TCP/IP แบบแสตกเข้าไปเป็นฮาร์ดแวร์ในหน่วยประมวลผลเพื่อเพิ่มความเร็วในการประมวลผล แต่เดิมได้แรงบันดาลใจจากโครงงานพัฒนาระบบเครื่องบริการเว็บแบบฝังตัวที่สามารถจัดรูปแบบใหม่ได้แต่เนื่องจากอุปสรรคในการหาอุปกรณ์บางชนิดเช่นหม้อแปลงที่ใช้ในการติดต่อกับระบบเครือข่ายหรือจะเป็นทางด้านระดับแรงดันไฟที่ไม่เท่ากันจึงทำให้ต้องหาอุปกรณ์เชื่อมต่อเครือข่ายมาใช้ใหม่และเนื่องจากโปรแกรมในโครงงานที่กล่าวมานี้แนบมาไว้ส่วนใหญ่จะมีใช้งานแบบ 16 บิตดังนั้นการออกแบบหน่วยประมวลผลให้ทำงานเป็นระบบ 16 บิตนั้นน่าจะเป็นวิธีการประหยัดการทำงานของหน่วยประมวลผลได้ดี

2.8 แนวคิดในการออกแบบระบบโดยรวม

การทำงานโดยรวมมีการใช้อุปกรณ์เชื่อมต่อเครือข่ายของบริษัทอินเทลรุ่น LXT972a เพราะใช้ศักย์ทางไฟฟ้าในระดับเดียวกันกับบอร์ดทดลอง FPGA ที่ 3.3 โวลต์แต่การสื่อสารจะต้องออกแบบใหม่เพราะวงจรรวมนี้สื่อสารผ่านมาตรฐาน MII ซึ่งสามารถลดการเชื่อมระหว่างขาของวงจรรวมเข้ากับบอร์ดทดลอง FPGA ได้เพราะมีการใช้งานตามรูปที่ 2.1 โดยที่จากเดิมในการ

เชื่อมต่อวงจรรวม DP83902A[12] เข้ากับบอร์ดทดลอง FPGA จะต้องเชื่อมต่อทั้งสายตำแหน่ง (address bus) 16 เส้นและสายข้อมูล (data bus) 16 เส้นและสายควบคุมอีก 22 เส้น รวมแล้วจะต้องต่อสายสัญญาณทั้งหมด 54 เส้นถึงจะทำงานได้ จึงได้เปลี่ยนมาใช้วงจรรวม LXT972a [13] แทนเพราะช่วยลดการเชื่อมต่อได้เหลือเพียง 22 เส้นเพราะเป็นตามมาตรฐาน MII จึงทำให้เชื่อมต่อทำได้ง่ายขึ้น ส่วนฝั่งการเชื่อมต่อกับสายคู่ไขว้ (Twisted pair) เพื่อเข้าสู่ระบบเครือข่ายจะทำผ่านหม้อแปลงสัญญาณที่เฉพาะไปในแต่ละรุ่นของวงจรรวมตามที่คู่มือระบุมาเท่านั้นโดยวงจรรวมนี้ใช้ PT163065 ซึ่งการเชื่อมต่อส่วนนี้สามารถทำได้ตามที่คู่มือระบุมา



รูปที่ 2.1 รูปแสดงการเชื่อมต่อและการทำงานภายในของ LXT972a

บทที่ 3

รายละเอียดของอุปกรณ์พื้นฐานและการออกแบบฮาร์ดแวร์

บทนี้กล่าวถึงความรู้พื้นฐานเกี่ยวกับการทำงานและการออกแบบระบบโดยรวมเพื่อทำงานในการสื่อสารในระบบเครือข่าย ซึ่งจะแบ่งออกเป็นการออกแบบหน่วยประมวลผลแบบแสตกซึ่งให้หน่วยประมวลผลทำงานแบบมีกลไกการทำงานพื้นฐานของชุดคำสั่งแบบแสตก ส่วนการรับข้อมูลทางเครือข่าย ส่วนส่งข้อมูลออกไปที่เครือข่าย ส่วนควบคุมการสื่อสารกับเครือข่าย ส่วนกำหนดค่าเริ่มต้นให้หน่วยความจำ และส่วนควบคุมการสื่อสารและการทำงานภายในวงจร

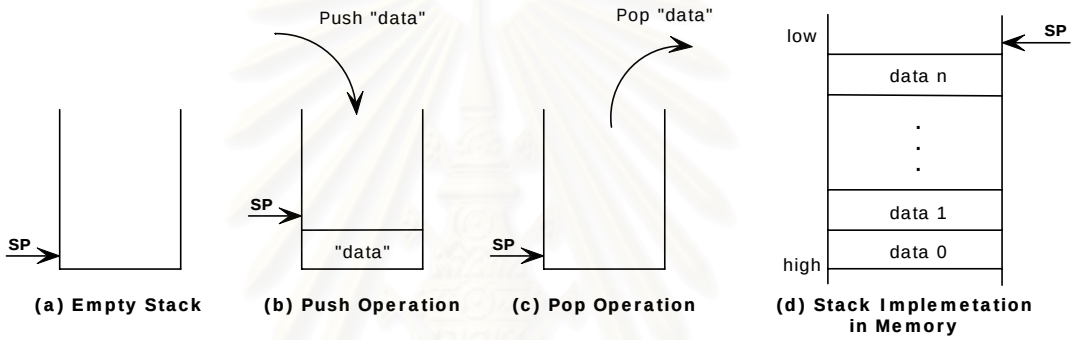
3.1 แสตกและการทำงานของหน่วยประมวลผล

การทำงานแบบแสตกเป็นพื้นฐานการทำงานในระบบคอมพิวเตอร์ ทำให้สามารถรองรับการทำงานของระบบคอมพิวเตอร์ได้เป็นอย่างดี และประหยัดการใช้ทรานซิสเตอร์ได้ และจากความประหยัดนี้ทำให้หลังจากที่สังเคราะห์แล้วจะได้ชิพที่มีบรรจุภัณฑ์ที่มีขาน้อยกว่าซึ่งทำให้ต้นทุนในการผลิตต่ำ และประโยชน์ของการใช้ทรานซิสเตอร์น้อยๆ ก็คือทำให้มีพื้นที่เหลือในวงจรรวมพอที่จะสังเคราะห์วงจรอื่นเข้าไปได้เช่นสังเคราะห์วงจรนับขนาด 16 บิต หรือ หน่วยความจำได้อีกทั้งขนาดของโปรแกรมที่เขียนอยู่ในรูปแบบของแสตกขนาด 16 บิตยังมีขนาดเล็ก [3] และเนื่องจากชุดคำสั่งของหน่วยประมวลผลแบบแสตกนั้นมีขนาดเล็กกว่าชุดคำสั่งแบบเรจิสเตอร์อยู่แล้ว จึงได้ขนาดโปรแกรมที่มีขนาดเล็ก ซึ่งจะมีตัวอย่างเช่นจาวาไบต์โค้ด

ตัวอย่าง ตัวประมวลผลแบบแสตกที่ผลิตเชิงพาณิชย์คือ RTX2000 [3] หน่วยประมวลผล RTX2000 เป็นหน่วยประมวลผลขนาด 16 บิตโดยมีการพัฒนามาจาก Novix NC4016 ที่มีการเพิ่มส่วนของหน่วยความจำแบบแสตก 256 ตัว ตัวคูณฮาร์ดแวร์ 16x16 บิตภายในการทำงานรอบเดียว และ มีการสร้างตัวนับเวลา 3 ตัว ด้วย และมีการเพิ่มการคำสั่งทำงานแบบพิเศษเพื่อเพิ่มประสิทธิภาพของคำสั่งกระโดดแบบมีเงื่อนไขคือคำสั่งที่ใช้ในการสลับไบต์ และมีการแบ่งหน่วยความจำออกเป็นหน้าสามารถยืดหน่วยความจำได้ถึง 32 พันคำ (K word) มีการแบ่งหน่วยความจำออกเป็นหลายส่วนคือ ส่วนของการเก็บโค้ดโปรแกรมและส่วนของการเก็บข้อมูล (สำหรับทำการดึงค่าและเก็บค่า) หน่วยความจำของผู้ใช้เพื่อเป็นฐานของตำแหน่ง (ใช้ในการยืดค่าเมื่อค่าที่อยู่ถูกส่งคือกลับมา Extending the value of the Return Stack address) เพราะว่าค่าที่คืนกลับมามีความยาว 21 บิตซึ่งข้อดีของการออกแบบนี้คือสามารถเรียกใช้โปรแกรมย่อยได้ทุกที่ในหน่วยความจำ รูปแบบคำสั่งของ RTX2000 จะใช้ความยาวบิต 16 บิตเท่ากันทุกคำสั่งในการ

ประมวลผลซึ่งทำให้คำสั่งที่เกี่ยวกับการกระโดดไปตำแหน่งอื่นสามารถทำได้เพียงแค่ 32 K Word เท่านั้นเพราะใช้บิตบนสุดในการบอกว่าเป็นการกระโดดแบบไหนซึ่งถ้าจะกระโดดไปให้ทั่วถึงทุกตำแหน่งในหน่วยความจำจะต้องทำการบวกค่าเข้าไปอีก

แสตก (Stack) คือโครงสร้างข้อมูลแบบหนึ่งที่มีลักษณะการจัดเก็บข้อมูลที่อนุญาตให้ข้อมูลที่เข้ามาทีหลังออกก่อน (Last in First out, LIFO) ดังรูปที่ 3.1(a) โครงสร้างของแสตกมีทางเข้าออกทางเดียว ทำให้ข้อมูลที่พึ่งเข้ามาในแสตกต้องออกไปก่อน ดังนั้นการทำงานกับแสตกจึงทำกับข้อมูลบนสุดของแสตกผ่านทางตัวดำเนินการพื้นฐานสองตัวได้แก่ตัวดำเนินการดัน (Push operation) และดึง (Pop operation) ซึ่งการทำงานของแสตกและตัวดำเนินการทั้งสองจะแสดงไว้ในรูปที่ 3.1(b) และ (c)



รูปที่ 3.1 แสตกและตัวดำเนินการของแสตก

การนำเอาแสตกมาใช้ในเป็นหน่วยความจำหลักในระบบคอมพิวเตอร์นั้นจะใช้หน่วยความจำเป็นพื้นที่เก็บข้อมูลของแสตกและการเข้าถึงข้อมูลบนแสตกกระทำผ่านการอ้างอิงด้วยตัวชี้แสตก (Stack pointer, SP) การดันข้อมูลลงแสตกคือการเขียนข้อมูลลงหน่วยความจำในตำแหน่งเลขที่อยู่ที่ SP ชี้อยู่ ส่วนการดึงข้อมูลเป็นการอ่านข้อมูลจากหน่วยความจำในเลขที่อยู่ที่ SP ชี้อยู่จะเห็นได้ว่ามีตัวที่ติดต่อกับหน่วยความจำเพียงตัวเดียว

โครงสร้างข้อมูลแบบแสตกมีบทบาทที่สำคัญมากกับการทำงานของคอมพิวเตอร์สองกลไกหลักได้แก่ กลไกในการหาค่านิพจน์ (Expression evaluation) และการจัดการโปรแกรมย่อย (Subroutine management) [20]

3.1.1 กลไกในการหาค่านิพจน์

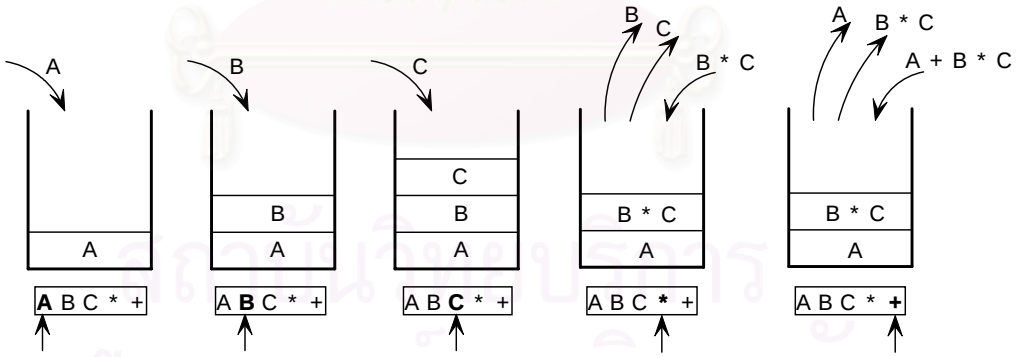
การหาค่านิพจน์ (Expression evaluation) ในหน่วยประมวลผลแบบแสตกนั้นจะต่างจากหน่วยประมวลผลแบบต่างๆ เพราะตัวชี้ตำแหน่งหน่วยความจำในการทำงานของหน่วยประมวลผลแบบแสตกมีเพียงตัวเดียวดังนั้นการเขียนสัญกรณ์จึงต่างจากสัญกรณ์ทั่วๆไป รูปแบบสัญกรณ์เติมกลาง (Infix notation) ดังเช่น

X = A + B * C

ไปเป็นแบบสัญกรณ์เติมท้าย(Postfix notation) โดยจะนำ B กับ C มาคูณกันก่อนแล้วค่อยนำผลลัพธ์มาบวกกับ A ซึ่งถ้าใช้การคำนวณแบบสัญกรณ์เติมกลาง ทำให้การคำนวณเข้าใจยากและซับซ้อน เนื่องจากโดยทั่วไปจะพิจารณาทีละตัวจากซ้ายไปขวาซึ่งทำให้คำตอบผิดเพราะจะเป็น X = (A + B) * C แต่ถ้าเปลี่ยนการคำนวณมาใช้แบบสัญกรณ์เติมหลังจะทำให้การคำนวณเป็นไปได้อย่างถูกต้องดังนี้

X = A B C * +

ซึ่งโครงสร้างข้อมูลแบบแสตกจะรองรับการคำนวณแบบสัญกรณ์เติมหลังได้เป็นอย่างดี แสดงได้ดังรูปที่ 3.2 การทำงานจะค่อยๆทำทีละตัวแปรในนิพจน์จากซ้ายไปขวาเช่นกัน โดยเมื่อเจอตัวแปรจะดันลงแสตกแต่ถ้าเจอตัวดำเนินการจะดึงข้อมูลออกจากแสตกมา2ตัวเพื่อคำนวณแล้วดันกลับลงไปใหม่ โดยการทำงานเมื่อเจอตัวแปร A ระบบจะดันลงแสตก เช่นเดียวกับเมื่อเจอตัว B และ C ระบบจะดันค่าลงแสตก แต่เมื่อตัวที่พิจารณาอยู่เป็นตัวดำเนินการคูณ ระบบจะดึงค่าสองค่าที่อยู่บนแสตก (B และ C) ออกมาและคูณค่าทั้งสองเข้าด้วยกัน หลังจากนั้นจะดันผลลัพธ์กลับลงแสตก หลังจากนั้นเมื่อเจอตัวดำเนินการบวกจะทำเช่นเดียวกับตัวดำเนินการคูณ ก็จะได้ผลลัพธ์ของนิพจน์นี้ออกมา



รูปที่ 3.2 การใช้แสตกในการคำนวณแบบสัญกรณ์เติมหลัง

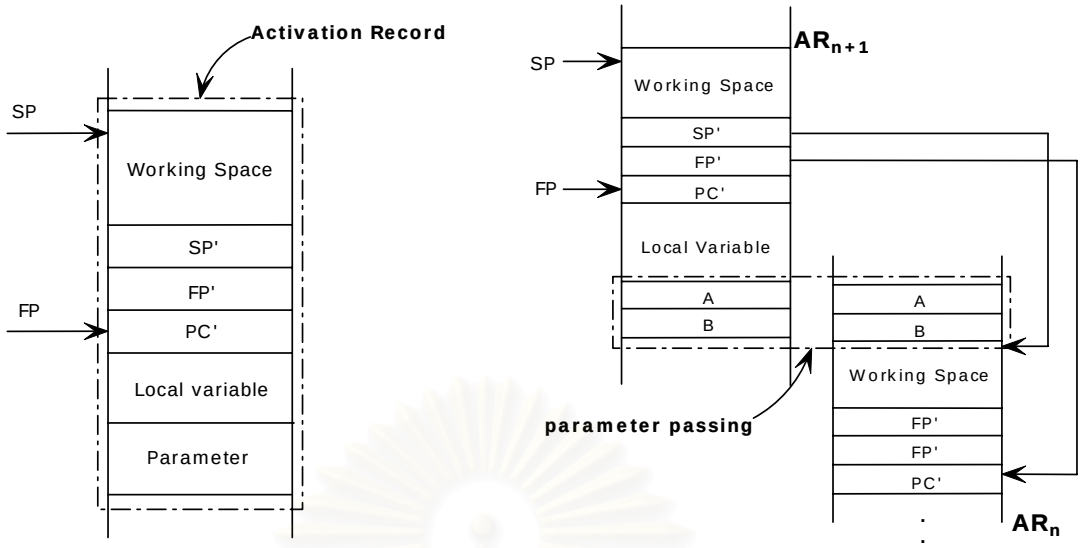
3.1.2 กลไกในการเรียกโปรแกรมย่อย

การเรียกโปรแกรมย่อยเป็นกลไกที่สำคัญในการพัฒนาโปรแกรมเพราะสามารถช่วยลดการเขียนโค้ดลงได้เยอะมากยิ่งกับโปรแกรมที่ส่วนใหญ่การทำงานมีรูปแบบการทำงานเหมือนกัน โดยทุกครั้งที่มีการเรียกโปรแกรมย่อยหน่วยประมวลผลที่ออกแบบมาจำเป็นต้องมีกลไกในการจัดการสิ่งต่างๆ ดังต่อไปนี้

1. จัดการให้สามารถกลับไปโปรแกรมหลักเพื่อทำงานต่อหลังจากเสร็จสิ้นโปรแกรมย่อย
2. การส่งพารามิเตอร์ (Parameter) จากโปรแกรมหลักไปยังโปรแกรมย่อย
3. จองพื้นที่และเข้าถึงตัวแปรเฉพาะที่ ที่อยู่ภายในโปรแกรมย่อย
4. จัดสรรพื้นที่แสตกที่ใช้ในโปรแกรมย่อยนั้นๆ ไม่ให้ไปซ้อนทับกับพื้นที่แสตกของโปรแกรมย่อยอื่น และจัดการคืนพื้นที่แสตกนั้นๆ หลังจากที่โปรแกรมย่อยทำงานเสร็จ

เทคนิคที่ใช้เรียกโปรแกรมย่อยต่างๆ นี้เรียกว่า "แอคทิเวชันเรคคอร์ด" (Activation record) ซึ่งเป็นโครงสร้างข้อมูลที่ทำงานบนแสตก โดยแบ่งออกเป็น 4 ส่วนสำหรับเก็บค่าการคำนวณต่างๆ และกำหนดขอบเขตพื้นที่ในแต่ละครั้งของการเรียกในแต่ละครั้ง ดังรูปที่ 3.3(a) ได้แก่ (1) เก็บสถานะการคำนวณ (Computation state) ซึ่งจะเก็บค่า PC, SP และ FP สำหรับใช้ในการกลับไปทำงานที่โปรแกรมหลัก (2) ค่าพารามิเตอร์ที่ส่งมาจากโปรแกรมหลัก (3) ตัวแปรเฉพาะที่ และ (4) พื้นที่แสตกที่ใช้ในการทำงานภายในโปรแกรมย่อย การกำหนดขอบแขตของแอคทิเวชันเรคอร์ดอาศัยตัวชี้สองตัว ได้แก่ ตัวชี้กรอบ (Frame pointer, FP) และตัวชี้แสตก (Stack pointer, SP) โดยที่ตัวชี้กรอบใช้ในการอ้างถึงแอคทิเวชันเรคคอร์ดนั้นๆ ส่วนตัวชี้แสตกใช้ในการเข้าถึงข้อมูลในแสตกที่อยู่ในแอคทิเวชันเรคคอร์ดนั้นๆ

กลไกการเรียกโปรแกรมย่อยกำหนดให้แอคทิเวชันเรคคอร์ดหนึ่งอันแทนการเรียกโปรแกรมย่อยหนึ่งครั้ง เมื่อมีการเรียกโปรแกรมย่อยทุกครั้งจะเกิดการสร้างแอคทิเวชันเรคคอร์ดขึ้นมาใหม่ดังในรูปที่ 3.3(b) ถ้ามีการเรียกโปรแกรมย่อยเกิดขึ้นขณะที่อยู่ที่ AR_n ระบบจะสร้าง AR_{n+1} ขึ้นใหม่ และเก็บสถานะการคำนวณซึ่งประกอบด้วยค่าเลขที่อยู่กลับ (Return Address) ของโปรแกรมย่อย เก็บค่าตัวชี้กรอบ (FP) และตัวชี้แสตก (SP) ของแอคทิเวชันเรคคอร์ดของโปรแกรมหลัก เมื่อโปรแกรมย่อยทำงานเสร็จ จะกลับมาทำงานในโปรแกรมหลัก ณ ตำแหน่งที่เก็บในค่าเลขที่อยู่กลับ และตำแหน่งของแอคทิเวชันเรคอร์ดของโปรแกรมหลัก (AR_n)



รูปที่ 3.3 โครงสร้างของแอคทิเวชันเรคคอร์ดและกลไกการเรียกโปรแกรมย่อย

การส่งพารามิเตอร์เข้าไปในโปรแกรมย่อยใช้การซ้อนทับกันของแอคทิเวชันเรคอร์ดทั้งสองอัน พารามิเตอร์ที่ถูกส่งมาจากแอคทิเวชันเรคอร์ดเดิมกลายมาเป็นตัวแปรเฉพาะที่ของแอคทิเวชันเรคอร์ดอันถัดไป ส่วนการอ้างถึงตัวแปรเฉพาะที่นั้น จะอ้างอิงกับตัวชี้เฟรม ทำให้สามารถจัดการกับตัวแปรเฉพาะที่ของโปรแกรมย่อยได้ง่ายและเป็นระบบ ในการเข้าไปทำงานในโปรแกรมย่อยนั้นจะไปตำแหน่งที่เก็บจำนวนทั้งพารามิเตอร์และตัวแปรเฉพาะที่ก่อน โดยที่จำนวนพารามิเตอร์จะเก็บค่าเป็นเลขบวก ส่วนจำนวนตัวแปรเฉพาะที่จะมีค่าเป็นลบ เช่นตัวอย่างการเรียกโปรแกรมย่อยบวกเลข

```
.c 0
  call main
  halt

fun add2 a b [ ]
  get a
  get b
  add
  retv
```

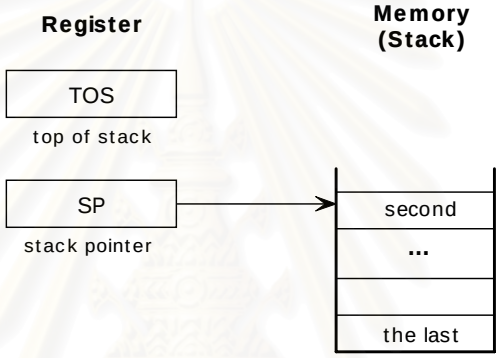
```
fun main [ ]
  lit 1
  lit 2      ; parameter passing
  call add2
  ret

.e
```

จากโค้ดภาษาแอสแซมบลีนี้เห็นได้ว่ามีส่วนที่ซ้อนกันอยู่สองตัวคือ a และ b ซึ่งแอคทิเวชันเรคอร์ดทำให้การจัดการกับโปรแกรมย่อยสะดวกขึ้น อีกทั้งยังสนับสนุนการเขียนโปรแกรมแบบเรียกซ้ำ (Recursive call) อีกด้วย

3.1.3 แสตกแคชชิ่ง

แสตกแคชชิ่ง (Stack caching) [20] เป็นหลักการหนึ่งที่ใช้ในการเพิ่มความเร็วการเข้าถึงข้อมูลในแสตก โดยมีแนวคิดมาจากการสังเกตพฤติกรรมการเข้าถึงข้อมูลในแสตก ข้อมูลที่จะถูกใช้ก่อนคือข้อมูลที่พึ่งเข้ามาเก็บในแสตก ดังนั้นการเก็บข้อมูลบนแสตกไว้ในเรจิสเตอร์ที่สามารถเข้าถึงได้เร็วกว่าหน่วยความจำแล้วจะลดเวลาที่ใช้ในการเข้าถึงแสตกได้ เนื่องจากเป็นการดึงข้อมูลมาเก็บไว้ก่อน

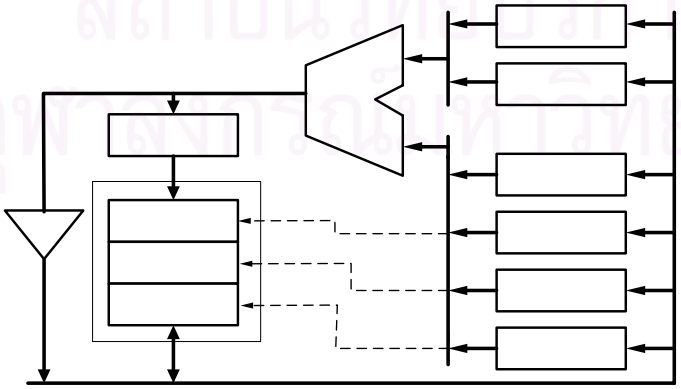
รูปที่ 3.4 แสดงหลักการของแสตกแคชชิ่งคือการนำเอาข้อมูลตัวบนสุดของแสตกมาเก็บไว้ในเรจิสเตอร์แทนการเก็บลงในหน่วยความจำที่เป็นแสตก และข้อมูลที่ถูกเรจิสเตอร์ SP ชี้อยู่เป็นข้อมูลตัวที่สองบนแสตก ซึ่งเป็นการลดการเข้าถึงข้อมูลแรกสุดในแสตกทำให้การทำงานเร็วขึ้น



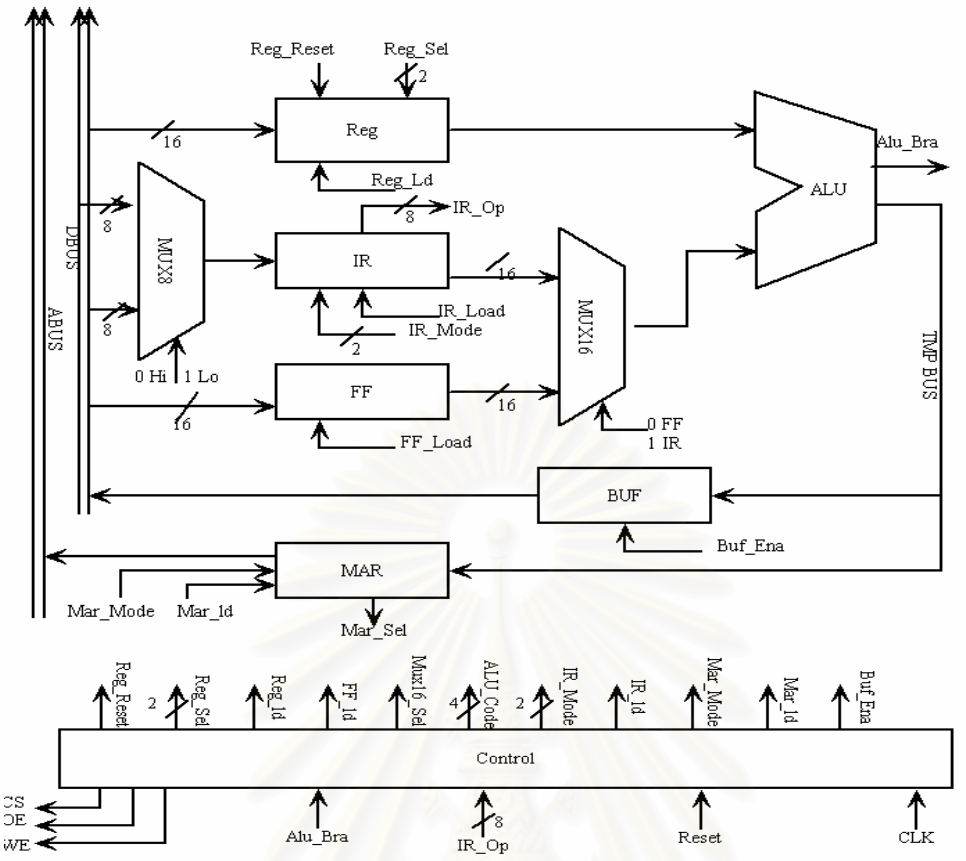
รูปที่ 3.4 หลักการของแสตกแคชชิ่ง

3.2 การออกแบบหน่วยประมวลผล

การออกแบบทางเดินข้อมูลให้หน่วยประมวลผล เน้นความเรียบง่ายตามรูปที่ 3.5 และ รูปที่ 3.6 ซึ่งจากจุดเด่นที่ว่าโครงสร้างเรียบง่ายที่เป็นจุดเด่นของหน่วยประมวลผลแบบแสตกนี้ [14] เพิ่มโอกาสสำเร็จในขั้นการสังเคราะห์บนวงจรรวม และมีการออกแบบชุดคำสั่งในหัวข้อ 3.3



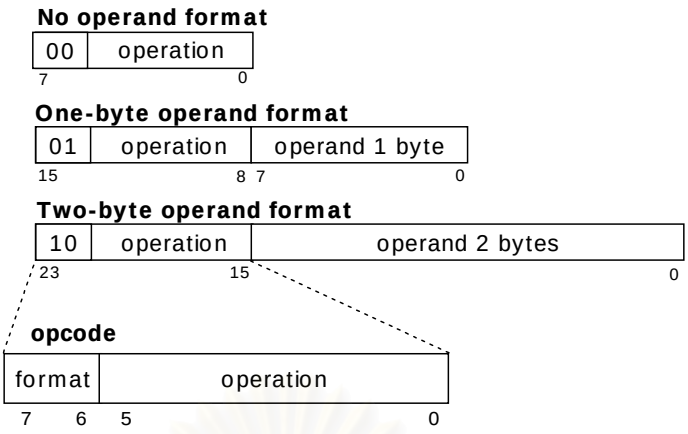
รูปที่ 3.5 แสดงการออกแบบหน่วยประมวลผลแบบแสตก



รูปที่ 3.6 แสดงวงจรภายในและวงจรควบคุมของหน่วยประมวลผลแบบแสตก

3.3 ชุดคำสั่งแบบแสตก SMC

ชุดคำสั่ง SMC เป็นชุดคำสั่งแบบแสตก (Stack-based instruction set) ประกอบไปด้วยคำสั่งแบบแสตกทั้งสิ้น 25 คำสั่ง รูปแบบของชุดคำสั่ง SMC มีด้วยกัน 3 รูปแบบดังรูปที่ 3.7 โดยแบ่งตามขนาดของตัวถูกดำเนินการ (Operand) ได้แก่คำสั่งที่ไม่มีตัวถูกดำเนินการ (No operand format) คำสั่งที่มีตัวถูกดำเนินการขนาดหนึ่งไบต์ (One-byte operand format) และคำสั่งที่มีตัวถูกดำเนินการขนาดสองไบต์ (Two-byte operand format) ไบต์แรกของทุกๆ รูปแบบของชุดคำสั่งจะเป็นรหัสดำเนินการ (Operation code: opcode) และ 2 บิตแรกของรหัสดำเนินการระบุถึงรูปแบบของคำสั่ง (Opcode type) นั้นๆ ดังตารางที่ 3.1



รูปที่ 3.7 รูปแบบของชุดคำสั่งแบบแสตก SMC

ตารางที่ 3.1 รายละเอียดของรูปแบบคำสั่งของ 2 บิตแรกในรหัสดำเนินการ

Opcode type	Format	Bytecode size (Byte)
00	No operand format	1
01	One-byte operand format	2
10	Two-byte operand format	3
11	Not defined	-

คำสั่งทั้ง 25 คำสั่งของชุดคำสั่ง SMC นั้นแบ่งออกเป็น 4 ประเภทด้วยกันได้แก่กลุ่มคำสั่งการคำนวณและตรรกะ (Arithmetic and logic instructions) กลุ่มคำสั่งย้ายข้อมูล (Data transfer instruction) กลุ่มคำสั่งควบคุม (Control flow instruction) และกลุ่มคำสั่งจัดการตัวแปรเฉพาะที่ (Local variable management)

รายละเอียดของการทำงานของคำสั่งแสดงไว้ในตารางที่ 3.4 การทำงานของคำสั่งจะใช้เรจิสเตอร์ที่จำเป็นทั้งสิ้น 5 ตัว ซึ่งตารางที่ 3.2 แสดงสัญลักษณ์และหน้าที่การทำงานของเรจิสเตอร์ต่างๆ เหล่านั้นไว้ การทำงานของคำสั่งใช้แนวคิดของแสตกแคชซึ่งในการเก็บข้อมูลบนสุดไว้ในเรจิสเตอร์ TOS เพื่อเพิ่มความเร็วในการทำงานของคำสั่ง

ตารางที่ 3.2 สัญลักษณ์และหน้าที่ของเรจิสเตอร์ที่ใช้ในชุดคำสั่ง SMC

สัญลักษณ์	หน้าที่การทำงานของเรจิสเตอร์
PC	ใช้ในการอ้างถึงคำสั่งในหน่วยความจำ (Program counter)
TOS	ใช้ในการเก็บข้อมูลบนสุดในแสตก (Top of stack register)
TMP	ใช้ในการคำนวณทั่วไป
SP	ตัวชี้แสตก (Stack pointer)
FP	ตัวชี้กรอบ (Frame pointer)

ในตารางที่ 3.4 การดำเนินการพื้นฐานของคำสั่งรหัสไบต์คือการดัน (Push) และดึง (Pop) ข้อมูลจากแสตกขึ้นมาประมวลผล โดยการทำงานของการดันและดึงแสดงได้ดังนี้

ตารางที่ 3.3 การดำเนินการดันและดึงของคำสั่งรหัสไบต์

Operator	**RTL Operation**
PUSH	MEM[SP] ← TOS; SP ← SP-1; TOS ← Data-to-push;
POP	Operate TOS as Pop-data; TOS ← MEM[SP+1]; SP ← SP + 1;

เมื่อคำสั่งดังกล่าวต้องการดันข้อมูลลงแสตก ต้องนำค่าที่อยู่ใน TOS ไปเก็บในแสตกส่วนที่เป็นหน่วยความจำก่อนแล้วจึงนำข้อมูลที่ต้องการดันถูกเก็บลงไปที่ TOS (จากหลักการแสตกแคชชิ่ง) และลดค่า SP ลงไปหนึ่งเพื่อชี้ไปยังแสตกช่องที่ว่างถัดไป

สำหรับการดึงข้อมูลนั้น ข้อมูลบนสุดของแสตกถูกเก็บไว้ใน TOS จึงสามารถดำเนินการกับข้อมูลดังกล่าวได้ทันที และหลังจากเสร็จสิ้นการดำเนินการแล้วต้องไปนำค่าที่อยู่ในแสตกอันดับถัดไปมาเก็บใน TOS แทนข้อมูลที่ถูกใช้ไป

ตารางที่ 3.4 รายละเอียดของชุดคำสั่งแบบแสตก SMC

Mnemonic	Operand	Description	Operation	Format
Arithmetic and logic instructions				
INC	-	Increment the top of stack	TOS ← TOS+1;	No operand
DEC	-	Decrement the top of stack	TOS ← TOS-1;	No operand
SHL	-	Shift the top of stack left one bit	TOS ← TOS<<1;	No operand
SHR	-	Shift the top of stack right one bit	TOS ← TOS>>1;	No operand
ADD	-	Add two data in the top of stack	TOS ← MEM[SP]+TOS; SP ← SP-1;	No operand
SUB	-	Sub two data in the top of stack	TOS ← MEM[SP]-TOS; SP ← SP-1;	No operand
AND	-	And two data in the top of stack	TOS ← MEM[SP]&TOS; SP ← SP-1;	No operand
OR	-	Or two data in the top of stack	TOS ← MEM[SP]| TOS; SP ← SP-1;	No operand
XOR	-	Xor two data in the top of stack	TOS ← MEM[SP]^TOS; SP ← SP-1;	No operand
LT	-	Less than comparison	if (TOS < MEM[SP]) TOS ← 1; SP ← SP-1;	No operand
LE	-	Less than or Equal comparison	if (TOS <= MEM[SP]) TOS ← 1; SP ← SP – 1;	NO operand
EQ	-	Equal comparison	if (MEM[SP] == TOS) TOS ← 1; SP ← SP-1;	No operand
Data transfer instructions				
LD	-	Load data from the memory	TOS ← MEM[TOS];	No operand
ST	-	Store top of stack to the memory	MEM[TOS] ← MEM[SP]; SP ← SP -1;	No operand
LIT	-	Push the literal into the top of stack	SP ← SP + 1; MEM[SP] ← TOS; TOS ← OPERAND	Two-byte

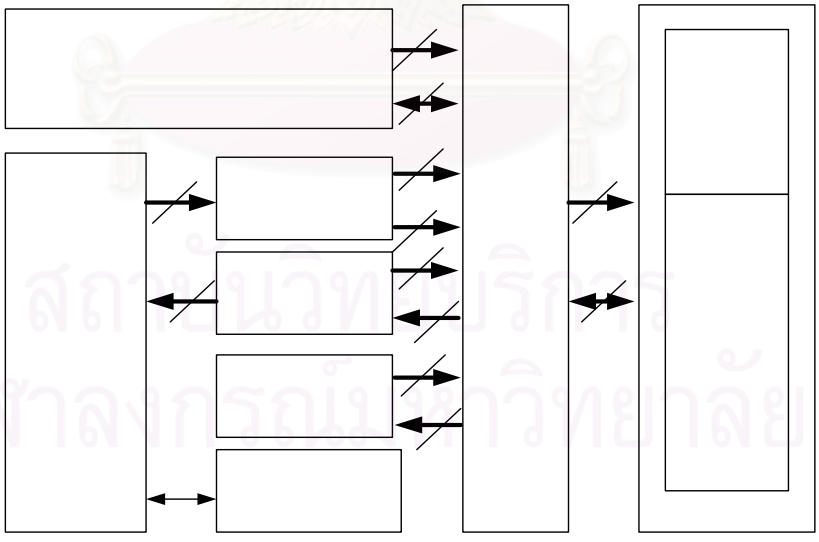
ตารางที่ 3.4 รายละเอียดของชุดคำสั่งแบบแสตก SMC (ต่อ)

Mnemonic	Operand	Description	Operation	Format
Control flow instructions				
JMP	RELATIVE	Unconditional jump	PC ← PC + OPERAND;	Two-byte
JT	RELATIVE	Jump if true (TOS = 1)	if(TOS == 1) PC ← PC+OPERAND; else PC ← PC+1; TOS ← MEM[SP]; SP ← SP-1;	Two-byte
JF	RELATIVE	Jump if fault (TOS != 1)	if(TOS == 0) PC ← PC+OPERAND; else PC ← PC+1; TOS ← MEM[SP]; SP ← SP-1;	Two-byte
CALL	ADS	Call subroutine	MEM[SP] ← PC; PC ← ADS; SP ← SP + 1; MEM[SP] ← TS; TS ← PC; PC ← oper; IR ← MEM[PC]; PC ← PC + 1; MEM[SP + IR] ← FP; FP ← SP + IR; IR ← MEM[PC]; PC ← PC + 1; MEM[FP + 1] ← SP + IR; SP ← FP + 1; MEM[SP + 1] ← TS;	Two-byte
RET	-	Return from subroutine	PC ← MEM[FP+2]; SP ← MEM[FP+1]; FP ← MEM[FP]; TOS ← MEM[SP]; SP ← SP – 1;	No operand
RETV	-	Return from subroutine with return value	PC ← MEM[FP+2]; SP ← MEM[FP+1]; Fp ← MEM[FP];	No operand
Local variable management				
GET	LOCAL	Get the local variable to the top of stack	SP ← SP+1; MEM[SP] ← TOS; TOS ← MEM[FP+OPERAND];	One-byte
PUT	LOCAL	Set the local variable with top of stack	MEM[FP+OPERAND] ← TOS; TOS ← MEM[SP]; SP ← SP-1;	One-byte

3.4 **ส่วนการออกแบบให้สื่อสารกับวงจรรวม** LXT972a

โดยปกติแล้ววงจรรวมหรือไมโครคอนโทรลเลอร์ระดับสูงหรือราคาแพงจะมีส่วนนี้ไว้ให้แล้วโดยจะเป็นชั้นของการสื่อสารของชั้น MAC [11] แต่ในแผงวงจรของวงจรรวมที่ใช้ทำการทดลองนั้นไม่มีส่วนนี้จึงต้องออกแบบขึ้นมาเอง โดยหัวข้อ 2.7 สามารถแบ่งการทำงานในการสื่อสารออกเป็น 3 ส่วนด้วยกันคือ ส่วนตั้งค่าให้กับเรจิสเตอร์ต่างๆ ในการสื่อสารกับเครือข่าย ส่วนรับข้อมูลจากเครือข่าย และ ส่วนส่งค่าออกไปที่เครือข่าย และ LXT972a สามารถทำงานอย่างอัตโนมัติเพียงแค่ป้อนสัญญาณนาฬิกาที่ความถี่ 25เข้าไปที่ขา REFCLK ของ LXT972a ก็สามารถทำงานได้แล้ว[13] แต่การทำงานจะไม่สามารถปรับเปลี่ยนความเร็วได้เพราะ LXT972a จะสื่อสารกับอุปกรณ์ต่างๆ ในระบบเครือข่ายแล้วนำมากำหนดความเร็วในการเชื่อมต่อเอง

และจากการเลือกใช้ LXT972a สามารถลดการเชื่อมต่อสายไฟลงได้ตามที่กล่าวไว้ในหัวข้อ 2.7 ซึ่งการเชื่อมต่อทั้ง 22 เส้นมีสายสัญญาณที่ต้องทำการเชื่อมต่อตามรูปที่ 2.1 โดยใช้ความกว้างของสายข้อมูลในการรับและส่งข้อมูลอย่างละ 4 เส้นและมีสัญญาณควบคุมอีกอย่างละ 3 เส้นเท่านั้น ดังนั้นการเชื่อมต่อจึงทำได้ง่ายขึ้นโดยการเชื่อมต่อระหว่างหน่วยประมวลผลเข้ากับ LXT972a ดังรูปที่ 3.8 ซึ่งมีตัวเลขกำกับตามสายสัญญาณใช้แสดงขนาดความกว้างของสายสัญญาณที่ใช้ในการสื่อสารซึ่งวงจรนี้ออกแบบไว้ด้วยโปรแกรม Protel รุ่น DXP โดยมีรูปวงจรที่ออกแบบการเชื่อมต่อระหว่างบอร์ดทดลอง FPGA เข้ากับบอร์ด LXT972a ตามภาคผนวก ง

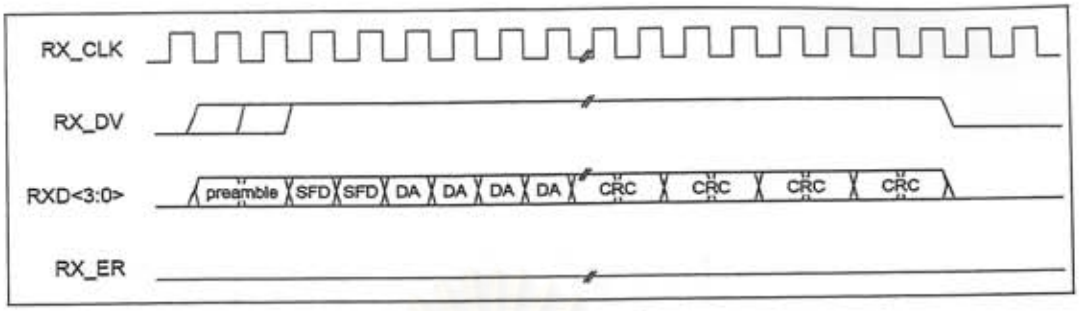


รูปที่ 3.8 แสดงการเชื่อมต่อของหน่วยประมวลผลกับ LXT972a

3.4.1 **ส่วนการออกแบบการรับข้อมูล**

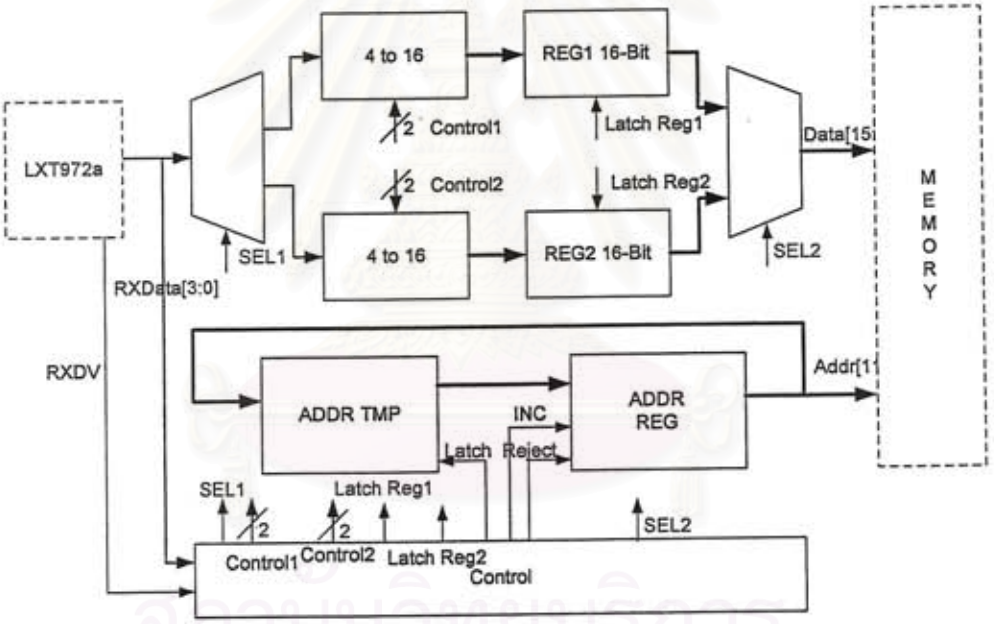
เนื่องจาก LXT972a จะทำการแปลงข้อมูลจากสัญญาณสัญญาแบบสายอนุกรมและอนาลอกสายเดียวมาเป็นสัญญาณแบบดิจิตอล 4 บิต และข้อมูลที่มานี้จะมาพร้อมๆ กับสัญญาณ

RxDv โดยมีจังหวะการทำงานในการรับข้อมูลตามสัญญาณ RX_CLK จาก LXT972a และข้อมูลโดยปกติมาพร้อมๆกับสัญญาณ RxDv การออกแบบจึงให้รอสัญญาณ RxDv ให้มีค่าเป็น 1 ตามรูปที่ 3.9



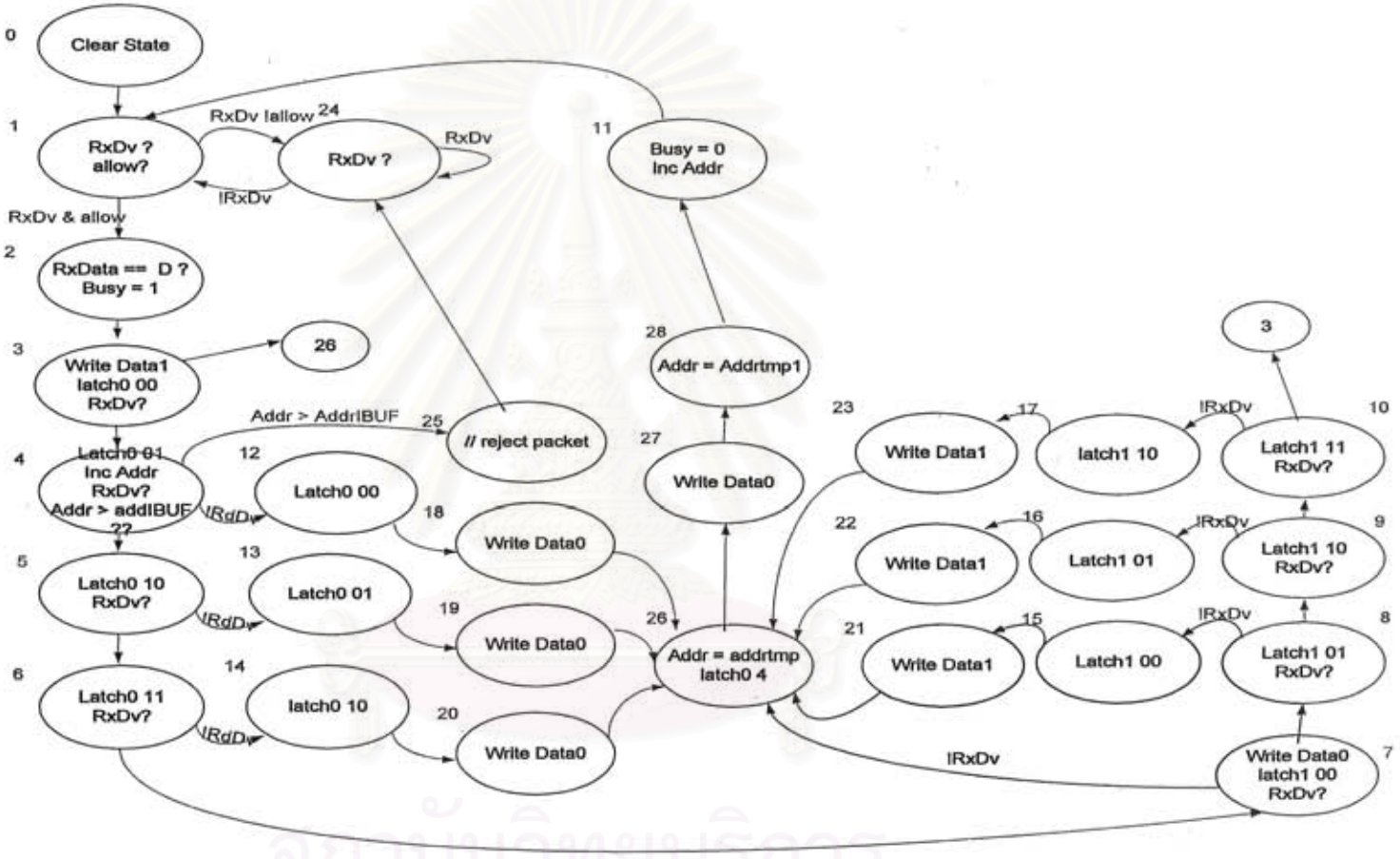
รูปที่ 3.9 แสดงข้อมูลที่รับได้มาจากเครือข่ายที่ความเร็ว 100เมกะบิต

โดยที่สัญญาณที่เข้ามาจากเครือข่ายจะมีค่าเป็น 5 และ D ตามลำดับ แล้วค่อยตามด้วยข้อมูลจริงๆในแต่ละชุดของข้อมูลชุดนั้นๆ โดยมีการออกแบบให้ส่วนรับข้อมูลมีลักษณะวงจรตามรูปที่ 3.10



รูปที่ 3.10 แสดงวงจรของส่วนรับข้อมูล

โดยใช้เรจิสเตอร์สองตัวในการเก็บข้อมูลที่เข้ามาเพื่อนำไปเขียนลงในหน่วยความจำและมีพฤติกรรมการทำงานตามรูปที่ 3.11

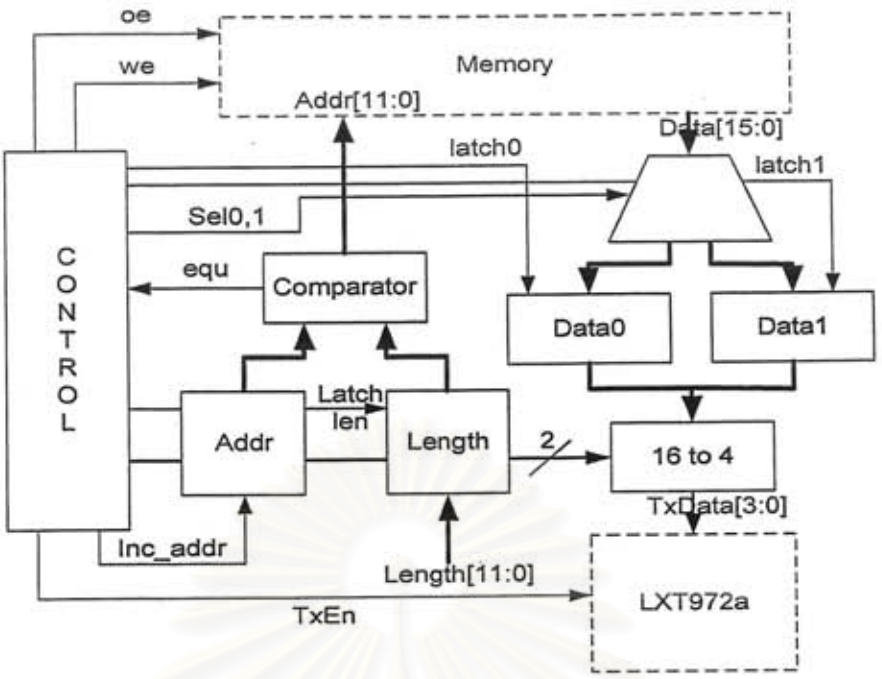


รูปที่ 3.11 แสดงการทำงานของส่วนรับข้อมูล

โดยส่วนใหญ่การทำงานหลักๆ จะใช้การทำงานหมายเลข 1 – 11 และ 26 – 28 เพราะข้อมูลที่รับเข้ามาส่วนใหญ่แล้วมีขนาดความกว้างเป็น Word หรือ Byteเท่านั้นแต่ได้มีการออกแบบในการทำงานหมายเลข 12 – 23 ไว้เพื่ออาจมีข้อมูลที่มีขนาดไม่ลงตัวตาม Word หรือ Byte และในส่วนการทำงานหมายเลข 26 – 28 จะเป็นการเขียนค่าความยาวไว้ที่ส่วนหัวของชุดข้อมูลชุดนั้นเพื่อความง่ายต่อการเขียนโปรแกรมของหน่วยประมวลผลในการละเลยชุดข้อมูลบางชุดเพราะการทำงานจะแค่เลื่อนตัวชี้ไปชี้ที่หัวของข้อมูลชุดใหม่เท่าเท่านั้น ซึ่งสามารถละเลยข้อมูลได้ทุกชนิดตามผู้เขียนโปรแกรมรู้จัก การทำงานหมายเลข 24 มีไว้เพื่อมีข้อมูลจากเครือข่ายเข้ามาในขณะที่หน่วยประมวลผลทำงานใดๆ อยู่ก็จะไม่รับข้อมูลชุดนั้นทั้งชุด แต่จากการทดลองพบว่าไม่เคยเกิดกรณีนี้ขึ้นเลย และในการออกแบบนี้ได้ออกแบบให้ใช้เรจิสเตอร์ในการเก็บค่าสองตัวเพื่อป้องกันการเขียนซ้ำเพราะว่าระหว่างการรับข้อมูลนั้นเครือข่ายไม่มีการหยุดพักเพื่อรอให้เขียนข้อมูลลงในหน่วยความจำโดยดูได้จากที่การทำงานที่ 3 – 6 จะเขียนข้อมูลที่ชื่อว่า Data1 ไปในหน่วยความจำและเก็บข้อมูลจาก LXT972a ไว้ใน Data0 ก่อน แล้วในการทำงานที่ 7 – 10 ก็จะเขียนข้อมูล Data0 และเก็บข้อมูลจาก LXT972a ไว้ใน Data1 และทำเช่นนี้ไปเรื่อยๆ จนกว่าจะข้อมูลจะหมดชุดและรอชุดต่อไปใหม่และในการจัดเรียงข้อมูลนั้นจะมีการจัดเรียงข้อมูลในแต่ละ word ของหน่วยความจำโดยมีการเรียงลำดับบิตเป็นเก็บครั้งแรกในตำแหน่งบิตที่ 11-8 แล้วตามด้วย 15-12 แล้ว 3-0 และ 7-4 ตามลำดับ และในการเก็บข้อมูลในส่วนของตัวรับข้อมูลจะใช้คิววนในการเก็บข้อมูลโดยจะเก็บอยู่ในช่วงที่หมายเลยแอดเดรสมีค่าเป็น E ในเลขฐาน 16 เท่านั้นเพื่อความง่ายต่อการออกแบบและพัฒนา และในการออกแบบมีการออกแบบสัญญาณแสดงสถานะว่างไว้ในกรณีที่ไม่มีข้อมูลเข้ามาจากเครือข่ายเลยสัญญาณนี้ก็จะให้ค่าเป็น 1 ออกไป ซึ่งค่านี้จะเทียบจากตัวชี้ของตัวเองก่อนที่จะเก็บข้อมูลในตำแหน่งถัดไปกับตัวชี้ของหน่วยประมวลผลโดยถ้ามีค่าเท่ากัน จะบอกว่าไม่มีข้อมูลรออยู่ในคิว หรือข้อมูลในคิวได้ผ่านการประมวลผลไปแล้ว ในการออกแบบได้คำนึงว่าอาจเกิดเหตุการณ์ข้อมูลเข้ามาระหว่างหน่วยประมวลผลทำงานก็จะไม่สามารถทำงานได้โดยจะติดอยู่ที่การทำงานที่ 2

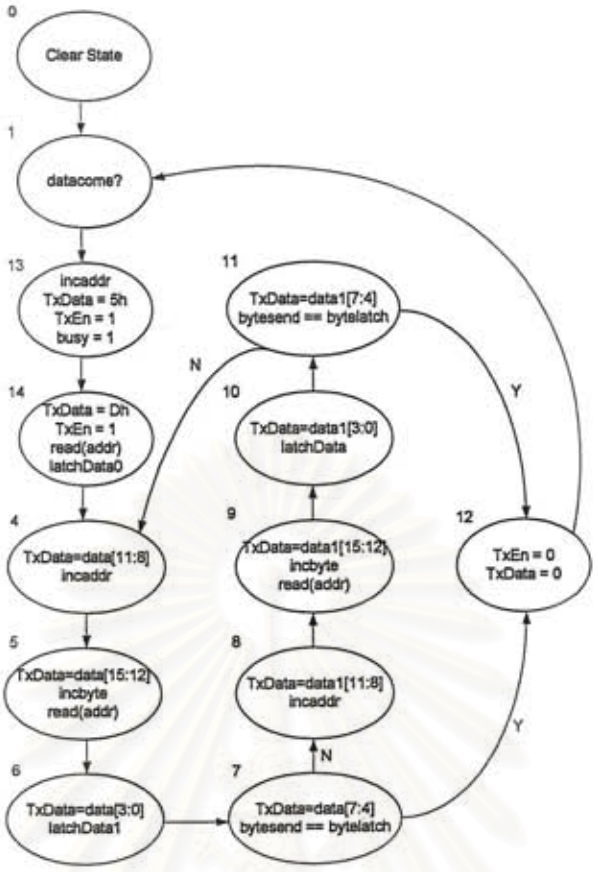
3.4.2 **ส่วนการออกแบบการส่งข้อมูล**

ในการออกแบบวงจรการส่งข้อมูล มีการออกแบบให้ใช้เรจิสเตอร์สองตัวเช่นเดียวกันกับส่วนรับข้อมูลโดยมีรูปวงจรตามรูปที่ 3.12 เมื่อต้องการส่งข้อมูลที่เตรียมพร้อมไว้แล้วในหน่วยความจำที่ออกแบบให้มีลักษณะการเรียงข้อมูลเช่นเดียวกันกับในส่วนรับข้อมูลคือเป็นแบบคิววนในช่วงที่ขึ้นต้นด้วยค่า F ในเลขฐาน 16 แล้ว มีการทำงานตามรูปที่ 3.13 ซึ่งการทำงานของการเรียงลำดับข้อมูลในหน่วยความจำจะต้องสัมพันธ์กับส่วนรับข้อมูลด้วยเพราะลำดับการรับข้อมูลไม่ได้มีเป็นทั้งเป็นรูปแบบ Big Endian หรือ Little Endian ตามในส่วนรับข้อมูล ดังนั้นการ



รูปที่ 3.12 แสดงวงจรของส่วนรับข้อมูล

นำข้อมูลออกจากหน่วยความจำจึงมีการเรียงลำดับการนำข้อมูลจากหน่วยความจำออกเป็นลำดับบิตที่ 11-8 ออกก่อนเป็นลำดับแรกแล้วตามด้วยแล้วค่อยตามด้วยบิตที่ 15-12 และบิต 3-0 และ 7-4 เป็นลำดับสุดท้ายและก่อนหน้าที่จะส่งชุดข้อมูลที่หน่วยประมวลผลเตรียมไว้ในแถวลำดับตัวพักข้อมูลขาออกก็จะส่งค่า 5 และ D ออกไปพร้อมๆกับ TxEn เป็น 1 ในส่วนหัว โดยการส่งข้อมูลนี้จะมีสัญญาณความถี่นาฬิกา TxClk ที่มาจาก LXT972a แล้วค่อยส่งข้อมูลในหน่วยความจำในส่วนตัวพักข้อมูลออกโดยส่งไปตามจำนวนคำที่หน่วยประมวลผลแจ้งให้ส่งออกไปในตำแหน่ง DFFFh โดยการเขียนข้อมูลลงไปในส่วนนี้จะกระตุ้นให้ส่วนส่งข้อมูลทำงาน

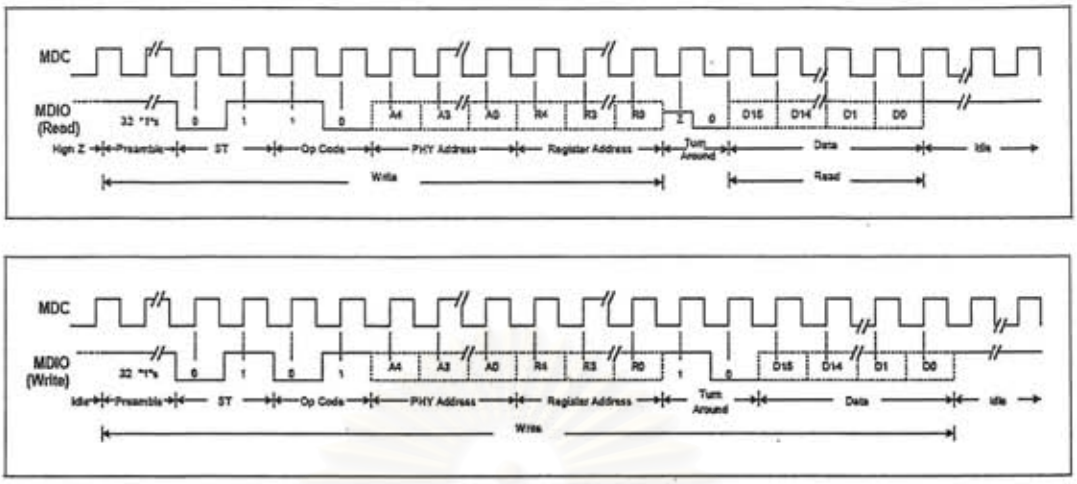


รูปที่ 3.13 แสดงส่วนการส่งข้อมูลทางเครือข่าย

3.4.3 ส่วนการตั้งค่าต่าง ๆให้เรจิสเตอร์ภายใน LXT972a

กำหนดการทำงานต่างๆ เช่นไม่ต้องทำการต่อรองแบบอัตโนมัติ (auto- negotiation) ให้ทำงานที่ความเร็ว 10 เม็กกะบิตอย่างเดียวและอื่นๆ โดยที่การสื่อสารให้ทำงานทั้งหมดตามมาตรฐาน MII เพราะในการสื่อสารที่ความเร็ว10 เม็กกะบิตีรูปแบบที่แตกต่างจากการสื่อสารที่ความเร็ว 100 เม็กกะบิตนส่วนของการส่งข้อมูลในส่วนหัวแต่โครงสร้างองข้อมูลต่างๆ จะคล้ายกัน โดยมีรูปแบบการสื่อสารการควบคุมเรจิสเตอร์ของอุปกรณ์ LXT972a มีรูปแบบการสื่อสารตามรูปที่ 3.14 โดยการทำงานนี้ให้เพื่อปรับให้ทำงานที่ความเร็ว 10 เม็กกะบิตเท่านั้นและไม่ให้ทำงานแบบต่อรองอัตโนมัติ และจากการสื่อสารแบบนี้ตามที่คู่มือของ LXT972a ระบุไว้ว่าสามารถทำงานที่ความถี่สูงสุดได้ที่ความถี่ 8 เมกะเฮิรตซ์และทำให้สามารถคำนวณได้ว่าสื่อสารสูงสุดได้ที่ความถี่ 62.5 กิโลเฮิรตซ์ (8เมกะเฮิรตซ์ / 64) และจากความต้องการดังกล่าวจะปรับเพียงแค่ที่เรจิสเตอร์หมายเลข 0 หมายเลขเดียวก็เพียงพอในการทำงานและการกำหนดค่าในเรจิสเตอร์นี้ต้องทำภายหลังจากเริ่มจ่ายไฟหรือกดสวิตซ์เริ่มต้นการทำงานภายในเวลา 300 ไมโครวินาทีเท่านั้น ซึ่งหลังจากการตั้งค่าให้กับ LXT972a แล้วจะให้ทำการเขียนค่าจาก

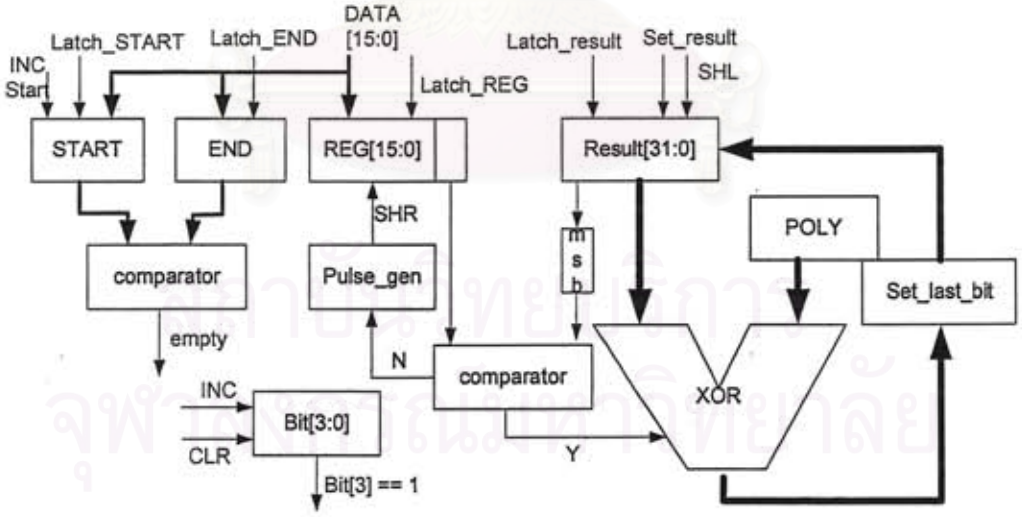
หน่วยความจำภายในวงจรรวม Spartan3 เข้าไปใน หน่วยความจำบนบอร์ดทดลอง FPGA เพื่อจะได้มีพื้นที่หน่วยความจำเพิ่มมากขึ้นจาก 10K มาเป็น 64K



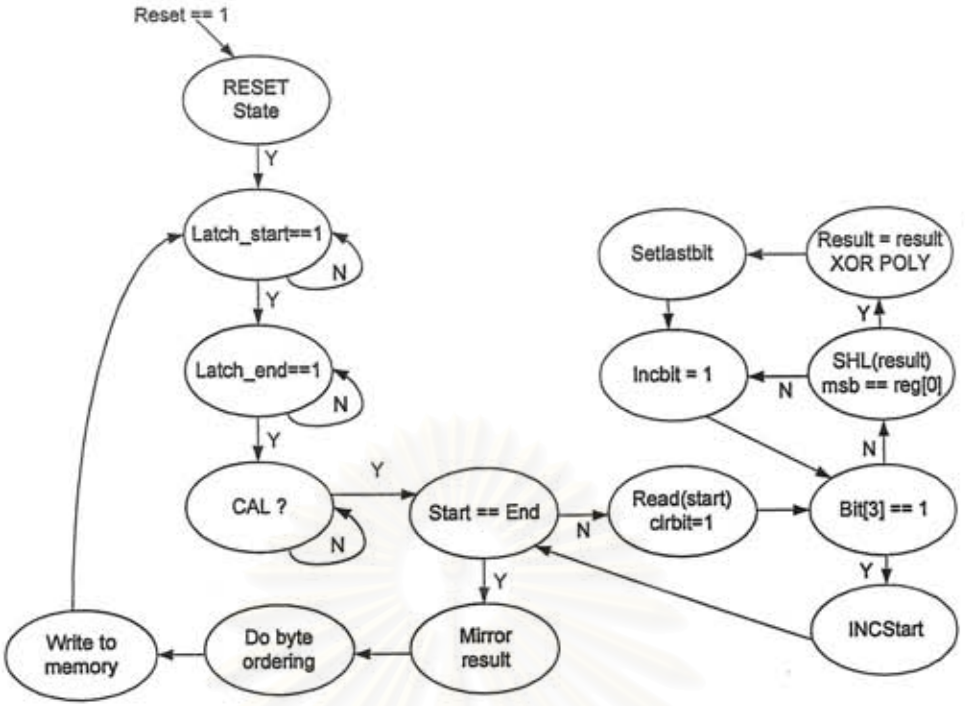
รูปที่ 3.14 แสดงการสื่อสารเพื่ออ่าน เขียนเรจิสเตอร์ควบคุมภายใน LXT972a

3.4.4 การออกแบบส่วน CRC32

การสื่อสารในเครือข่ายปกติจะมีข้อมูลขนาด 16 บิตสองชุดตามท้ายในแต่ละแพคเก็จซึ่งจริงๆ แล้วเป็น CRC ขนาด 32 บิต [16] และในการออกแบบส่วนนี้ถือเป็นวงจรพิเศษที่ออกแบบขึ้นมาใช้ในระบบ เนื่องจากไม่ต้องการเข้าไปแก้ไขในส่วนของหน่วยประมวลผลให้เป็น 32 บิต หรือทำงานได้ทั้ง 16 บิตและ 32 บิตโดยมีวงจรดังรูปที่ 3.15 โดยมีส่วนที่เป็น 32 บิตอยู่ 3 ตัว คือ Result POLY และ XOR และมีการทำงานของวงจรตามรูปที่ 3.16 โดยการทำงานจะเริ่มต้นจาก



รูปที่ 3.15 แสดงวงจรการคำนวณ CRC32



รูปที่ 3.16 แสดงการทำงานในการคำนวณค่า CRC32

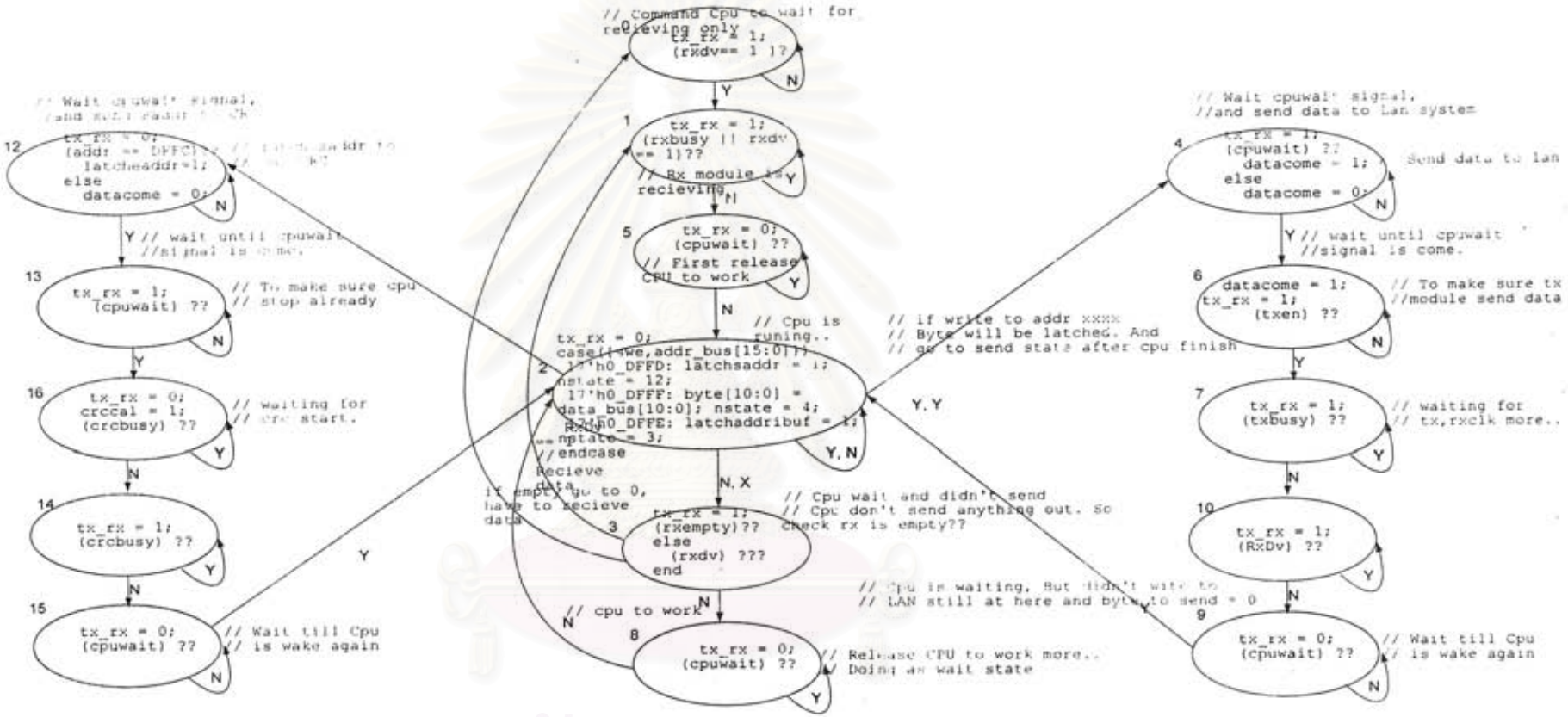
รับค่า SADDR จากสายข้อมูล (data_bus) มาเก็บไว้ และรอรับค่า EADDR เพื่อเริ่มทำการคำนวณค่า CRC ตามพฤติกรรมที่ออกแบบไว้และข้อมูลที่ทำการคำนวณจะเก็บไว้ในส่วนของ OBUF ซึ่งเมื่อคำนวณเสร็จแล้วก็จะเขียนเข้าไปต่อท้ายของข้อมูลที่จะส่งโดยส่วนส่งข้อมูลซึ่งข้อมูลที่จะส่งมีรูปแบบที่จะมีส่วนต้นแล้วตามด้วยข้อมูลจริงๆ และส่วน CRC [15] และในการสร้างวงจรคำนวณค่า CRC32 ออกแบบให้ ฮาร์ดแวร์ทำงานเพื่อความเร็วในการประมวลผลข้อมูลเพราะสามารถเพิ่มความถี่สัญญาณนาฬิกาให้ไม่เกินความถี่ที่ใช้ติดต่อกับหน่วยความจำ ซึ่งผิดกับการสร้างส่วนคำนวณเข้าไปให้ซอฟต์แวร์ทำงานเพราะจะใช้เวลาในการคำนวณมากกว่า

3.4.5 การออกแบบการสื่อสารภายในทั้งหมด

เนื่องจากวงจรทั้งหมดไม่ได้ออกแบบมาให้มีการใช้ตัวพักข้อมูลจึงมีการออกแบบตัวควบคุมการทำงานของส่วนของวงจรทั้ง 3 วงจรในข้างต้น โดยให้รับข้อมูลจากเครือข่ายแล้วนำไปเขียนในหน่วยความจำ แล้วค่อยปล่อยให้หน่วยประมวลผลทำงานโดยหน่วยประมวลผลจะนำข้อมูลที่ได้รับนี้ไปประมวลผลต่างๆ และดูว่าจะส่งข้อมูลหรือไม่ ถ้าส่งหน่วยประมวลผลจะเขียนความยาวของข้อมูลของชุดนั้นๆ ที่เตรียมไว้ในหน่วยความจำในช่วงที่ระบุไปที่ตำแหน่ง DFFFh และเมื่อส่งข้อมูลเสร็จก็จะปล่อยให้หน่วยประมวลผลทำงานต่อไป แต่หน่วยประมวลผลจะห้ามประมวลผล และถ้าส่วนรับข้อมูลแจ้งสถานะว่าไม่มีข้อมูลในหน่วยความจำ หน่วยประมวลผลจะ

ถูกสั่งให้คอย เพราะการทำงานต่างๆ ไม่ว่าจะโดยหน่วยประมวลผล โดยส่วนรับข้อมูล หรือโดยส่วนส่งข้อมูล การทำงานจะต้องติดต่อกับหน่วยความจำโดยมีการออกแบบให้ทำงานตามรูปที่ 3.17 จะเป็นภาพรวมของการทำงานต่างๆ โดยมีการให้สถานการณ์ทำงานต่างๆ ไว้กับอุปกรณ์นั้นๆ ก่อนเช่นให้หน่วยประมวลผลรอก็จะให้ค่า Tx_Rx เป็น 1 เตรียมไว้ก่อนเมื่อหน่วยประมวลผลมาถึงที่ขั้นรอ(ไม่มีการติดต่อกับหน่วยความจำขณะนั้น)และได้รับสถานะนี้เข้าไปก็ทำให้หน่วยประมวลผลหยุดรอ เมื่อเปลี่ยนสถานะ Tx_Rx หน่วยประมวลผลก็จะทำงานต่อจากที่ทำงานไว้เดิม และในการทำงานทั้งหมดนี้ในแต่ละส่วนจะใช้สัญญาณนาฬิกาที่ ความถี่ต่างกันทั้งหมดโดยที่ตัวควบคุมการทำงานทั้งหมด จะใช้ความถี่ที่ 50 เมกะเฮิรตซ์ส่วนในส่วนรับข้อมูลก็ทำงานตามความถี่สัญญาณนาฬิกา TxClk และ RxClk ที่ได้รับมาจาก LXT972a และความถี่สัญญาณนาฬิกา 6.25 เมกะเฮิรตซ์ที่ใช้ในการป้อนให้หน่วยประมวลผล

ในการทดสอบการทำงานของทุกๆส่วนที่ออกแบบมานั้นจะผ่านการทดสอบเพื่อเพิ่มความมั่นใจโดยการใช้โปรแกรมโมเดลซิมในเบื้องต้นและทดสอบพฤติกรรมในการเขียนภาษาเวอร์รีล็อก แต่การทดสอบนี้ยังไม่เพียงพอเพราะหลายๆ วงจรสามารถทำงานได้อย่างถูกต้องแม่นยำโดยใช้โปรแกรมโมเดลซิมเพื่อตรวจสอบแต่ไม่สามารถทำงานจริงได้ในบอร์ดทดลอง อาจเป็นเพราะขาดความรู้ความเข้าใจในการใช้เทคนิคของเครื่องจำลองการทำงานอย่าถ่องแท้จึงทำให้ได้ผลการทำงานในเชิงพฤติกรรมถูกต้องแต่แผนภูมิเวลาก็ยังคลาดเคลื่อนอยู่ ซึ่งปัญหาเหล่านี้เกิดจากผู้พัฒนาที่ขาดความเข้าใจในการใช้เครื่องมืออย่างถ่องแท้ หรือขาดความเข้าใจอย่างแท้จริงในการเขียนภาษาเวอร์รีล็อกด้วยโครงงานนี้จึงได้ใช้เครื่องวิเคราะห์ตรรกะ เข้ามาเพื่อจับสัญญาณในเชิงดิจิตอลและใช้ออสซิโลสโคปในการจับสัญญาณไฟฟ้าและสัญญาณในเครือข่ายทั้งหลาย



รูปที่ 3.17 การควบคุมการทำงานการรับ - ส่งข้อมูลและหน่วยประมวลผล

บทที่ 4

มาตรฐานเกณฑ์วิธี ทฤษฎีที่เกี่ยวข้อง และการออกแบบซอฟต์แวร์

บทนี้กล่าวถึงการพัฒนาระบบ TCP/IP แบบแสตกโดยมีโครงในการพัฒนามาจากภาษาแอสเซมบลีของไมโครคอนโทรเลอร์ 8051 ในโครงงานการพัฒนาระบบเครื่องบริการเว็บที่สามารถจัดรูปลักษณ์ใหม่ได้ [1] มาเปลี่ยนเป็นภาษาแอสเซมบลีของหน่วยประมวลผลแสตก เพื่อความรวดเร็วและไม่ยากในการพัฒนา แต่เนื่องจากหน่วยประมวลผล 8051 ที่ใช้ในโครงงานนั้นเป็นหน่วยประมวลผลแบบ 8 บิต ส่วนหน่วยประมวลผลที่ออกแบบมาเป็นแบบ 16 บิตจึงต้องมีการเปลี่ยนโครงสร้างของข้อมูลในการติดต่อระหว่างอุปกรณ์ทางด้านเครือข่ายกับหน่วยประมวลผลแบบแสตก ในการออกแบบนี้จะออกแบบให้รองรับถึง ICMP[17] ก็เพียงพอต่อการสื่อสารในเครือข่ายแล้ว แต่การออกแบบก็จะอิงตามTCP/IP แสตก เพราะมีความซับซ้อนน้อยกว่าการสร้างชั้นต่างๆ ของ OSI 7^{th} - Layer ซึ่งข้อมูลที่ได้รับจากหน่วยเชื่อมต่อเครือข่ายนั้นก็จะได้ข้อมูลที่เข้ามาในรูปแบบของข้อมูลที่แท้จริงที่ได้รับการห่อหุ้มมาจากชั้นต่างๆ เช่นกัน

4.1 การสร้างระบบทีซีพีไอพีแสตก

ข้อมูลที่ผ่านเข้ามาจาก LXT972a จะผ่านการระบบทีซีพีไอพีแบบแสตกที่สามารถรองรับการทำงาน TCP UDP ICMP [16][17] เนื่องจากระบบที่สร้างขึ้นมา มีการใช้หน่วยความจำที่จำกัดและในการสร้างให้หน่วยประมวลผลสามารถรองรับการทำงานที่กล่าวมานั้นก็เพียงพอต่อการสื่อสารทางด้านเครือข่ายซึ่งมีรายละเอียดในการสร้างเป็นไปตามชั้นทั้ง 7 ของ OSI หรือ TCP/IP แสตก[16] ตามรูปที่ 4.1 ซึ่งเป็นมาตรฐานในการสื่อสารในระเครือข่ายทั่วๆไปแต่ข้อมูลที่ได้รับจาก LXT972a นี้จะผ่านเข้ามาในชั้นที่ 2 เลยหมายความว่าข้อมูลที่ได้รับหลังจะผ่านการคลี่ออกมาบางส่วนแล้ว โดยมีรายละเอียดอย่างย่อในแต่ละชั้นดังนี้

Layer	ISO/OSI	TCP/IP (Internet)
7	Application	Telnet, FTP, SMTP, HTTP, DNS, BOOTP,DHCP,SNMP
6	Representation	
5	Session	
4	Transport	TCP , UDP
3	Network	IP , ICMP , IGMP
2	Data Link	Device Driver and Interface
1	Physical	Media

รูปที่ 4.1 แสดง OSI -7 layers เปรียบเทียบกับ TCP/IP แสตก

1. Physical layer

ชั้นนี้จะเป็นคำนึงถึงการสื่อสารในแต่ละบิตของวงจรเพื่อให้แน่ใจได้ว่าการรับหรือส่งข้อมูลนั้นเป็น 1 หรือเป็น 0 แน่ๆโดยการใช้ระดับแรงดันของสัญญาณไฟฟ้าและมีช่วงเวลาของในการรับ-ส่งของแต่ละบิตมีอยู่ในระดับไมโครวินาทีเท่านั้น

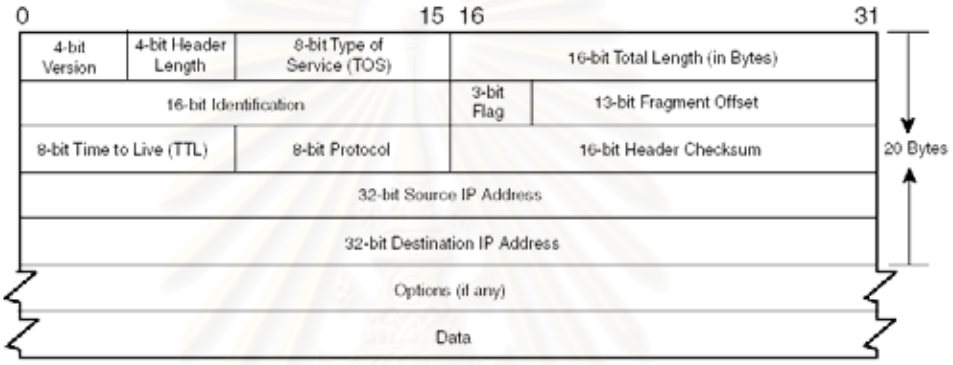
2. Data link layer

เป็นชั้นถัดมาจากชั้นที่ 1 โดยจะมีการทำงานโดยแบ่งข้อมูลออกเป็นเฟรมๆ เพื่อในการหลีกเลี่ยงปัญหาของข้อผิดพลาดต่างๆในการรับข้อมูลของอุปกรณ์ซึ่งในชั้นนี้จะมีการทำวิเคราะห์ขอบเขตของแต่ละเฟรม ตัดสินใจความถูกต้องของแต่ละเฟรมและมีการวางระเบียบของระยะเวลาการมาถึงของเฟรมต่างๆ

3. Network layer เปรียบเทียบได้กับ IP layer ใน TCP/IP แสตก

เป็นชั้นที่มีความซับซ้อนมาก โดยจะมีการบริการการสื่อสารของข้อมูลระหว่างสองแม่ข่ายโดยที่ไม่ขึ้นอยู่กับอุปกรณ์ของแม่ข่ายนั้นๆ เช่นการเชื่อมต่อที่ง่ายที่สุดของ 2 สถานีจะเป็นการเชื่อมต่อโดยตรงซึ่งข้อมูลนั้นจะมาได้จากเครือข่ายที่ต่างทางกันแต่เข้ามาทางเกตเวย์เดียวกันซึ่งในชั้นนี้จะต้องมีการตอบสนองสิ่งต่างๆ เช่น สถานที่ทำการ มีการซ่อมแซมมาหรือไม่ และจุดสิ้นสุด

การติดต่อระหว่างสองแม่ข่ายนั้นๆ ซึ่งก็จะขึ้นอยู่กับปัญหาของการกำหนดเลขที่อยู่ ปัญหาเส้นทาง และการป้องกันปัญหาคอขวด โดยข้อมูลที่ได้รับเข้ามาจะต้องผ่านชั้นนี้ก่อนที่จะผ่านเข้าไปในชั้นของชั้นการสื่อสารจึงมีการรองรับบริการ ICMP และ ARP เพราะหน่วยประมวลผลไม่จำเป็นจะต้องทำการวิเคราะห์ข้อมูลที่ส่งเข้ามาจะเป็นเพียงการตรวจดูว่ามีการร้องเรียกชนิดนั้นๆ หรือไม่เท่านั้นเอง เช่น มีสัญญาณร้องเรียกมาเป็นชนิด ICMP ก็จะเพียงแค่ตอบกลับไปเท่านั้นโดยไม่จำเป็นจะต้องผ่านขั้นตอนอื่นอีก พอข้อมูลที่ผ่านจากชั้นนี้ไปก็จะเข้าสู่ชั้นการขนส่งรูปแบบข้อมูลในชั้นนี้เป็นไปตามรูปที่ 4.2



รูปที่ 4.2 รูปแสดงข้อมูลในชั้นไอพี

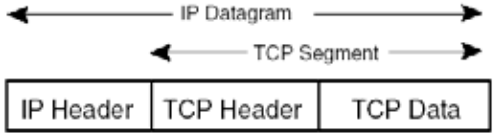
4. Transport Layer เปรียบเทียบได้กับ TCP หรือ UDP ใน TCP/IP STACK

ภารกิจหลักจะเป็นการซ่อนลักษณะของเครือข่ายทั้งหมดจากชั้นที่อยู่ข้างบนชั้นนี้และมีการทำให้การส่งข้อมูลโปร่งใสดังนั้นทุกๆ การให้ความหมายของเกณฑ์วิธีในชั้นนี้จะใช้ในการสื่อสารระหว่างสองเครือข่ายเท่านั้นจะไม่จำเป็นต้องสร้างเข้าไปในทุกๆ เครื่องคอมพิวเตอร์ที่ผ่านในระบบเครือข่าย ดังนั้นนักพัฒนาระบบจึงสนใจเฉพาะการสื่อสารข้อมูลในชั้นนี้เป็นต้นไป จะแบ่งออกเป็น 2 แบบคือ UDP และ TCP เป็นไปตามรูปที่ 4.3 และ รูปที่ 4.5 ตามลำดับ ส่วนรูปที่ 4.4 จะเป็นเสมือนภาพรวมของกลุ่มข้อมูลทั้งหมดในชั้น IP และชั้นการการขนส่ง

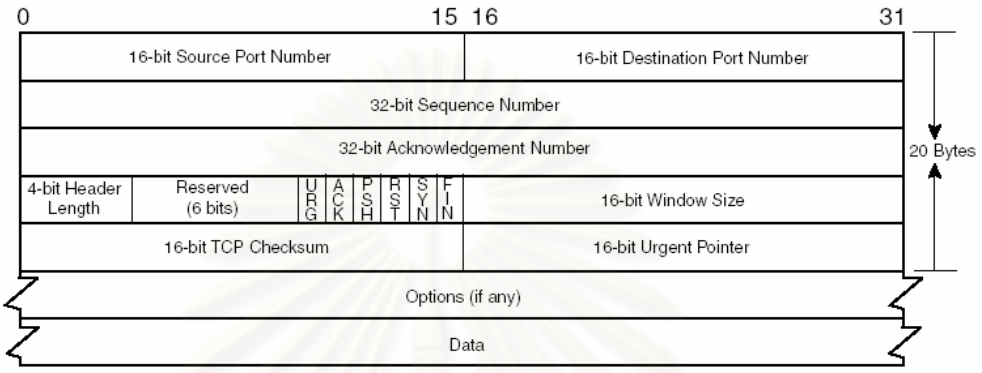
0 31

Source Port	Destination Port
UDP Length	UDP Checksum

รูปที่ 4.3 แสดงรายละเอียดกลุ่มข้อมูลแบบ UDP



รูปที่ 4.4 รูปแสดงส่วนใหญ่ๆของกลุ่มข้อมูลทั้งในชั้นของ IP และชั้น TCP



รูปที่ 4.5 แสดงรายละเอียดกลุ่มข้อมูลแบบ TCP

5. Session layer

เป็นไปตามชื่อคือในชั้นนี้จะเป็นการระบุระยะเวลาอย่างเป็นตรรกะในการติดต่อระหว่างผู้ใช้ และจุดประสงค์ของชั้นนี้ เพื่อบริการการติดต่อแบบ ให้ผู้ใช้ติดต่อแบบสมดุล(*user-oriented*)

6. Presentation Layer

ชั้นนี้จะตระหนักถึงรูปแบบของข้อมูลที่แลกเปลี่ยนกันซึ่งจะมีการบริการกลุ่มของการบริการข้อมูลที่เปลี่ยนแปลง รูปแบบและรวมไปถึงการส่งข้อมูล เช่น ผู้ใช้คนหนึ่งใช้รหัสแอสกี้ (ASCII) ส่วนอีกคนหนึ่งใช้รหัสเอบซิดิก (EBCDIC) ซึ่งในชั้นนี้ก็จะทำการเปลี่ยนแปลงให้เหมือนกันยิ่งไปกว่านั้นในชั้นนี้ยังสามารถที่จะใช้วิธีการบีบอัดข้อความหรือว่าการเข้ารหัสข้อมูลด้วย

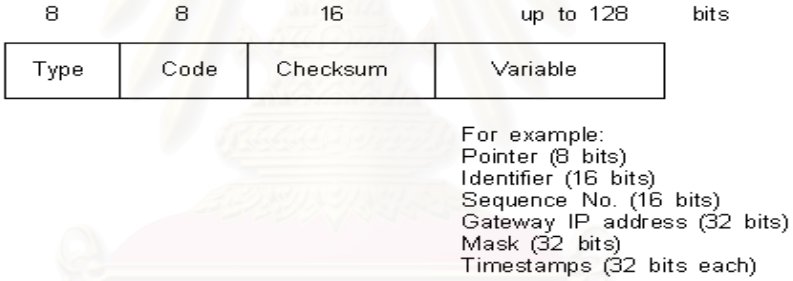
7. Application Layer

เป็นชั้นสูงที่สุดที่จะอ้างถึงสภาพแวดล้อมต่างๆที่ผู้ใช้จะดำเนินการและสื่อสารและในชั้นนี้จะมีการเก็บฟังก์ชันการจัดการและให้ประโยชน์เชิงกลเพื่อรองรับโปรแกรมประยุกต์แบบกระจายซึ่งเกณฑ์วิธีที่บริการก็จะมีเช่น ftp mail telnet โดยการทำงานของชั้นที่สามารถปรับเปลี่ยนและ

จัดการต่างๆ ได้จะอยู่ในชั้นของเครือข่ายเปรียบเทียบได้กับชั้นไอพีและชั้นขนส่งเปรียบเทียบได้กับชั้นทีซีพีหรือยูดีพีโดยจะมีการรองรับการทำงานด้านต่างๆ

4.1.1 การทำงาน PING

จากวัตถุประสงค์จะทำการพัฒนาให้ถึงการตอบสนองตามเกณฑ์วิธี ICMP เท่านั้นเพราะเพียงพอต่อการรับ-ส่งข้อมูลของระบบเครือข่ายแล้ว แต่ในการตอบสนองตามเกณฑ์วิธี ICMP นั้นต้องมีการตอบสนองรูปแพคเก็จรูปแบบ ARP ก่อน เพราะการทำงาน PING จะเริ่มต้นด้วยการส่งแพคเก็จ ARP ออกมาเพื่อหาที่อยู่ของจุดหมายเพื่อเก็บหมายเลข MAC ก่อนจึงค่อยส่งแพคเก็จรูปแบบ ICMP ตามออกมา และมาตรฐานในชั้นไอพีเท่านั้นโดยเกณฑ์วิธีICMP นั้นจะใช้ในในความผิดพลาดและการควบคุมข้อความต่างๆ ภายในชั้นไอพีซึ่งเกณฑ์วิธีนี้จะให้ข้อมูลออกมาเมื่อไม่เป็นไปตามแผนเช่น ในการส่งข้อมูลอาจมีข้อมูลจะต้องส่งข้อมูลออกเป็นกลุ่มข้อมูลแล้วค่อยส่งออกไปซึ่งก็มีโอกาสที่กลุ่มข้อมูลก่อนหน้าที่ส่งออกไปจะสูญหายได้ซึ่งข้อความ ICMP จะส่งผลของข้อความก่อนหน้าออกไปด้วย โดยมีโครงสร้างที่แตกต่างกันขึ้นอยู่กับรูปแบบของแต่ละเกณฑ์วิธีและโครงสร้างของข้อความก็จะขึ้นอยู่กับชนิดตามรูปที่ 4.6



รูปที่ 4.6 แสดงรูปของข้อมูลชั้น IP

จะมีรูปแบบทั้งหมดตามนี้

- Type0(Echo Reply) จะใช้ในการตอบจากสถานีสุดท้ายซึ่งก็คือการตอบเมื่อมีข้อความ Type8 ส่งมาซึ่งในส่วนของส่วนที่เปลี่ยนแปลงได้ (Variable) จะมีตัวชี้(Identifier)ยาว 16 บิตและตัวเลขลำดับ(Sequence Number)ยาว 16 บิต การทำงานจะทำโดยตรวจสอบตัวชี้ถ้าเหมือนกันก็จะทำตอบกลับไปและหมายเลขลำดับก็จะเพิ่มขึ้นอีกหนึ่งทุกๆการส่งกลับไป

- Type3(Destination Unreachable) จะบอกเมื่อเกิดปัญหาขึ้นระหว่างการส่งกลุ่มข้อมูลซึ่งจะมีปัญหาลักษณะต่างๆกันโดยจะแสดงในส่วนของรหัสอยู่ 5 แบบคือ

- Code0 (Net Unreachable) จะถูกส่งโดยอุปกรณ์บอกเส้นทาง(router)ไปที่เครื่องให้บริการถ้าตัวบอกทางไม่รู้เส้นทางที่ระบบเครือข่ายเรียกร้อง
- Code1 (Host Unreachable) จะถูกส่งโดยอุปกรณ์บอกเส้นทางไปที่เครื่องให้บริการถ้าสามารถเห็นการเรียกร้องจากเครือข่ายแต่ไม่ใช่จากสถานีที่หมาย
- Code2 (Protocol Unreachable) จะให้รหัสนี้ออกมาเมื่อกลุ่มข้อมูลส่งไปได้แต่ไม่สามารถทำงานด้าน UDP หรือ TCP
- Code3 (Port Unreachable) จะเกิดในกรณีที่ข้อมูลไปถึงสถานีปลายทางและทำงานด้าน TCP/IP ได้แต่จะทำได้เฉพาะบริการที่ระบุเท่านั้นเช่นในเครื่องบริการเว็บจะต้องมีการระบุพอร์ตที่จะเข้าถึงให้แน่นอนจึงสามารถทำงานได้
- Code4 (Cannot Fragment) ส่งโดยอุปกรณ์บอกเส้นทางและในกลุ่มข้อมูลจะมีบิตที่บอกห้ามแบ่งข้อมูลออก(DF-Do not fragment)อยู่ในส่วนหัวของข้อมูล ซึ่งถ้าเป็น 1 แสดงว่าห้ามแบ่ง
- Code5 (Source Route Failed) ก็เป็นอีกหนึ่งทางเลือกที่มีไว้ให้

- Type4(Source Quench) จะเกิดเมื่อสถานีส่งทางส่งกลุ่มข้อมูลเร็วกว่าสถานีรับมาก
- Type5(Redirect) เมื่อสถานีส่งพบว่ามีเส้นทางอื่นที่ดีกว่าโดยตรวจสอบได้จากจากเกตเวย์และจะให้รหัสต่างๆออกมาตามนี้
  - Code0 ส่งกลุ่มข้อมูลออกไปจากเครือข่าย
  - Code1ส่งกลุ่มข้อมูลออกไปจากสถานีส่ง
  - Code2ส่งกลุ่มข้อมูลออกไปจากและรูปแบบการให้บริการ
  - Code3ส่งกลุ่มข้อมูลออกไปจากสถานีส่งและรูปแบบการให้บริการ

  และ64บิตของส่วนที่เปลี่ยนแปลงได้จะใช้สำหรับหมายเลขไอพีของเกตเวย์
- Type8 (Echo Request) ใช้สำหรับตรวจสอบการเชื่อมต่อและความหนาแน่น (Ping-Packet Internet Groper) รายละเอียดจะเหมือนกับรูปแบบ0 (Type0)
- Type11 (Time Exceeded) ข้อมูลถูกทิ้งเมื่อส่งออกไปเป็นระยะเวลานานซึ่งเวลาจะถูกตั้งไว้ในช่องเวลาที่อยู่รอด (TTL-Time To Live) ในหัวของไอพีและจะมีส่วนของรหัสที่ใช้ในการบอกสถานะว่าเกินเวลาหรือยังถ้าเกินรหัสจะเป็น 0 และเป็น 1 ถ้ายังไม่เกินเวลา

- Type12 (Parameter Problem) จะบอกค่าพารามิเตอร์ที่ผิดพลาดในกลุ่มข้อมูล(รหัส0)จะมีช่องตัวชี้ขนาด 8 บิตอยู่ในส่วนที่เปลี่ยนแปลงได้ของกลุ่มข้อมูล ICMP ซึ่งตัวชี้นี้จะชี้โดยใช้ 8 บิตภายในส่วนหัวของ IP

- Type13 (Timestamp request) ให้ข้อมูลเวลาของการเดินทางครบรอบจากสถานีส่งจนถึงสถานีรับที่ระบุไว้ซึ่งในส่วนที่เปลี่ยนแปลงได้ประกอบไปด้วย ช่องที่มีขนาด 16 บิต 2 ช่องและช่องที่มีขนาด 32 บิต 3 ช่องคือ ช่องตัวชี้(Identifier) ช่องหมายเลขลำดับ ช่องเวลาปั๊มเริ่มต้น ช่องเวลาปั๊มตอนรับ และช่องเวลาปั๊มตอนส่ง โดยเวลาจะอยู่ในรูปมิลิวินาทีนับจากเที่ยงคืน

- Type14 (Timestamp reply) ให้ข้อมูลของเวลาการเดินทางเหมือนกับรูปแบบที่13

- Type15 (Information Request) รูปแบบนี้จะอนุญาตให้สถานีส่ง เรียนส่วนของระบบเครือข่ายของหมายเลข IP โดยการส่งข้อความกับที่อยู่ของสถานีส่งในส่วนหัวของ IP และใส่เลข 0 เข้าไปในช่องสถานีปลายทางซึ่งรูปแบบนี้จะใช้ความยาว 16 บิตเป็นตัวชี้และหมายเลขลำดับ

- Type16 (Information Reply) เป็นการตอบโดยการห่อส่วนของเครือข่ายเข้าไป

- Type17 (Address mask request) เรียกร้องสำหรับการเก็บตัวพรางเครือข่ายย่อย (subnet mask) ไว้ใช้

- Type18 (Address mask response) ตอบด้วยตัวพรางเครือข่ายย่อยที่ใช้

ซึ่งการออกแบบการทดลองจะใช้การการร้องเรียกกับการตอบกลับก็พอคือชนิด 0 และ ชนิด 8

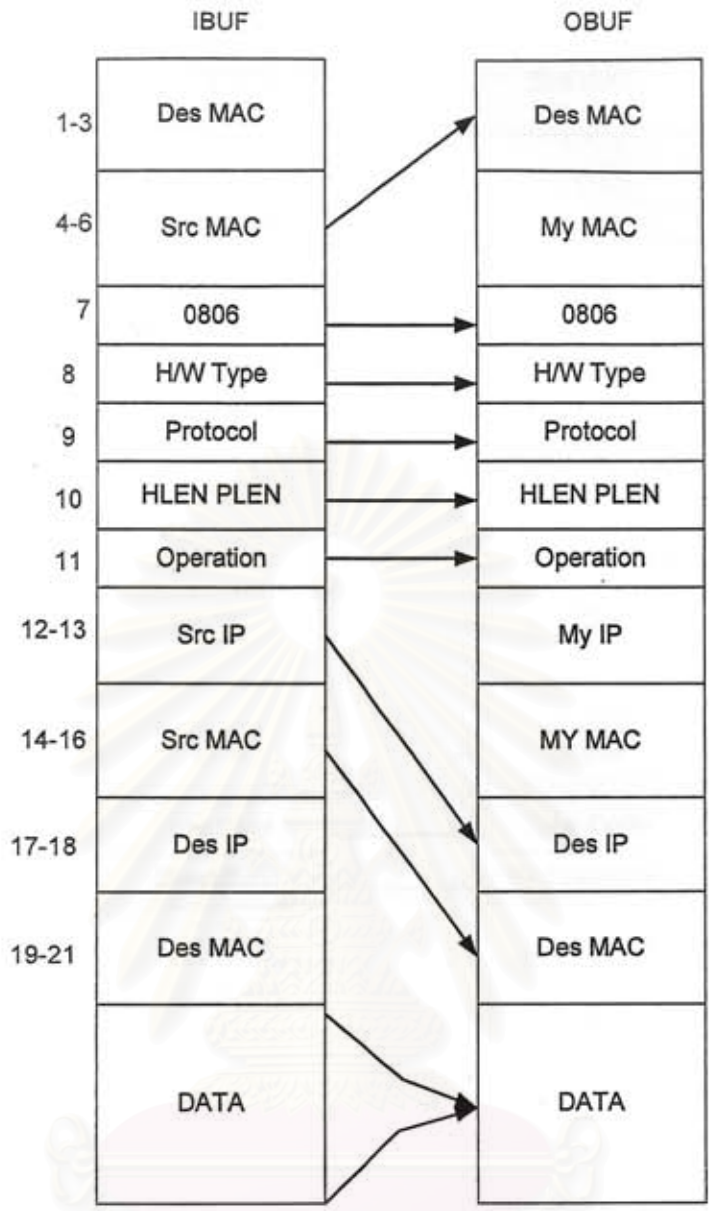
4.2 ออกแบบการทำงานส่วนซอฟต์แวร์

การทำงานจะอ้างอิงจากโครงงานวิจัยการพัฒนาระบบเครื่องบริการเว็บแบบฝังตัวที่สามารถจัดรูปแบบใหม่ได้[1][18] [19] โดยการทำงานจะเริ่มต้นเมื่อหน่วยประมวลผลรับข้อมูลเข้ามาแล้วหน่วยประมวลผลจะเริ่มนำข้อมูลเข้ามาตรวจสอบว่าเป็นชนิดที่ต้องการหรือไม่ ถ้าไม่ใช่ชนิดที่ต้องการก็จะปฏิเสธข้อมูลชุดนั้นไปโดยการเลื่อนตัวชี้ไปชี้ที่ชุดถัดไปและจากการออกแบบในส่วน 3.4.4 หน่วยประมวลผลจะต้องรอทางด้านฮาร์ดแวร์โดยห้ามทำงานอะไรเพราะการทำงานคำสั่งรอทางด้านซอฟต์แวร์ต้องมีการติดต่อหน่อยความจำ ถ้ายังไม่มีข้อมูลเข้ามาจากเครือข่าย ดังนั้นการเขียนโปรแกรมจึงไม่ต้องคำนึงถึงเรื่องการรอข้อมูลจึงสามารถเขียนโปรแกรมให้ทำงานได้ตามปกติเลย และสาเหตุที่ออกแบบให้ทั้งระบบทำงานแบบนี้เพราะการตรวจสอบสถานะของหน่วยประมวลผลจะต้องติดต่อกับหน่วยความจำซึ่งอาจเป็นช่วงเดียวกันกับที่มีการรับหรือส่ง

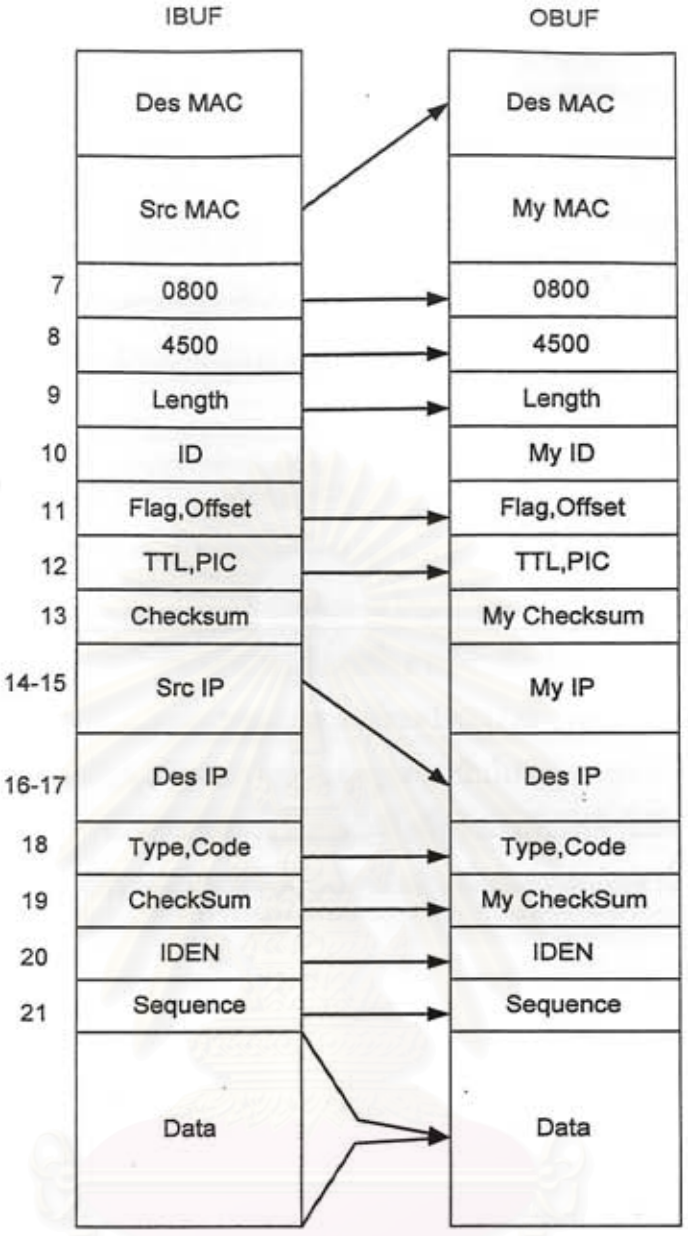
ข้อมูลก็ได้จึงต้องออกแบบตามนั้น ดังนั้นการออกแบบซอฟต์แวร์จึงเขียนโปรแกรมได้ง่ายๆ ตามรูปที่ 4.7 โดยการทำงานทั่วๆไปของระบบเครือข่ายเมื่อมีการเรียกร้องจากจุดไหนก็ตามในเครือข่ายจะมีการส่งแพคเก็จ ARP ไปถามที่อยู่ก่อนซึ่งเมื่อได้รับแพคเก็จ ARP เข้ามาแล้วก็จะต้องตอบกลับไปด้วยแพคเก็จ ARP เช่นกันซึ่งถ้าไม่ตอบกลับจะทำให้อุปกรณ์จัดเส้นทางตัดอุปกรณ์นี้ออกไปจากระบบและการตรวจสอบจะตรวจสอบที่ตำแหน่งที่ 7 ในข้อมูลชุดนั้นถ้ามีค่า 0806h ก็แสดงว่าเป็นแพคเก็จแบบ ARP และที่ตำแหน่งเดียวกันนี้ถ้ามีค่าเป็น 0800h จะหมายความว่าเป็นแพคเก็จแบบ ICMP ตาม รูปที่ 4.8 และ รูปที่ 4.9 ตามลำดับ เช่นการทำงานของคำสั่ง PING จะเริ่มจากการส่งแพคเก็จ ARP ออกไปก่อนแล้วรอการตอบกลับพร้อมๆ กับส่งแพคเก็จ ICMP ออกไปซึ่งเป็นอันจบการสื่อสาร และจากรูปที่ 4.8 และ รูปที่ 4.9 แสดงลูกศรที่ทำการเคลื่อนย้ายข้อมูลในส่วน IBUF ที่ส่วน OBUF แต่จะตรวจสอบก่อนว่าชุดข้อมูลที่เข้ามานั้นเป็นชุดข้อมูลของตัวเองหรือไม่โดยการตรวจสอบที่ค่าในตำแหน่งที่17 และ18 ในรูปที่ 4.8 แล้วค่อยตอบกลับไปถ้าใช่ แต่ถ้าไม่ใช่ก็จะข้ามข้อมูลชุดนั้นไป โดยที่การจัดเรียงข้อมูลที่จะทำการส่งจะเหลื่อมกันอยู่เล็กน้อยเพราะที่จริงแล้วที่ตำแหน่ง 0 ไม่มีความยาวของชุดข้อมูลเก็บไว้แต่ในการออกแบบส่วนรับข้อมูลออกแบบไว้ให้เพิ่มส่วนนี้เข้าไปเองตามที่กล่าวไว้แล้วในข้อ3.4.1 ซึ่งจะเป็นแบบนี้เพียงครั้งแรกและต่อไปข้อมูลใน OBUF จะไม่เว้นช่องว่างเลยและการจัดเรียงข้อมูลจะเป็นไปตามรูปที่ 4.10 โดยจะมีการเหลื่อมกันเล็กน้อยเพราะที่ช่องว่างของช่องแรกเป็นของ IBUF นั้นเป็นความยาวของชุดข้อมูลที่สร้างขึ้นมาเอง(ไม่ได้รวมอยู่ในชุดข้อมูล)ดังนั้นตอนส่งข้อมูลจึงไม่ต้องส่งไปด้วย เพราะในส่วนส่งข้อมูลออกแบบจะส่งแบบต่อเนื่องกันไปไม่สามารถเว้นได้โดยเริ่มส่งจากตำแหน่งที่ F001 และวนไปเรื่อยๆภายในตำแหน่งที่ขึ้นต้นด้วย F การจัดเรียงชุดข้อมูลจึงมีลักษณะเป็นไปตามรูปที่ 4.10 ดังนั้นเมื่อต้องการเลื่อนตำแหน่งที่ชี้ในส่วน IBUF จะต้องบวกเพิ่มไปอีก 2 แต่ในขณะที่ ส่วน OBUF จะบวกเพิ่มอีกแค่ 1 ก็พอ หลังจากที่ออกแบบไว้แล้วก็นำไปเขียนโปรแกรมภาษาแอสแซมบลีของหน่วยประมวลผลแบบแสตก

และในส่วน ICMP_RSP สร้างวงจรย่อยเพื่อคำนวณค่า Checksum ซึ่งค่านี้ไม่เหมือนกับค่าที่ใช้ทั่วไป [16] แต่จะมีการแก้ไขเล็กน้อยเช่น มีการลบค่าลงไปหนึ่งในส่วนของ Checksum ส่วนหัว และ มีการลบค่า Checksum หลังจากคำนวณลงไปอีก 5 ในส่วนของตัวข้อมูลซึ่งค่านี้จะต้องถูกต้องทั้งหมดถ้าไม่เช่นนั้นแล้วหน่วยประมวลผลจะไม่แสดงข้อมูลการตอบรับทำให้ดูเหมือนว่ารับสื่อสารไม่ได้

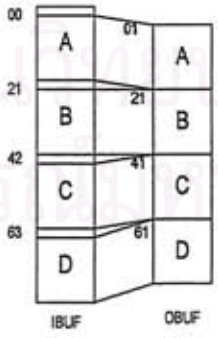
รูปที่ 4.7 แสดงการทำงานโดยรวมของซอฟต์แวร์



รูปที่ 4.8 แสดงรูปแบบชุดข้อมูลแบบ ARP ที่ความกว้างข้อมูล 16 บิต



รูปที่ 4.9 แสดงรูปแบบชุดข้อมูลแบบ ICMP ที่ความกว้างข้อมูล 16 บิต



รูปที่ 4.10 แสดงการจัดเรียงข้อมูลในหน่วยความจำของส่วนรับข้อมูลและส่วนส่งข้อมูล

บทที่ 5

การทดลอง

บทนี้กล่าวถึงการทดลองและผลที่ได้จากการทดลองของทุกๆ ส่วนที่ออกแบบมาทั้งหมด มีจุดประสงค์เพื่อทวนสอบการทำงานในแต่ละส่วนให้ชัดเจนพร้อมทั้งจะได้รู้เวลาที่ใช้ในแต่ละส่วนและวัดผลการทำงานในด้านต่างๆ

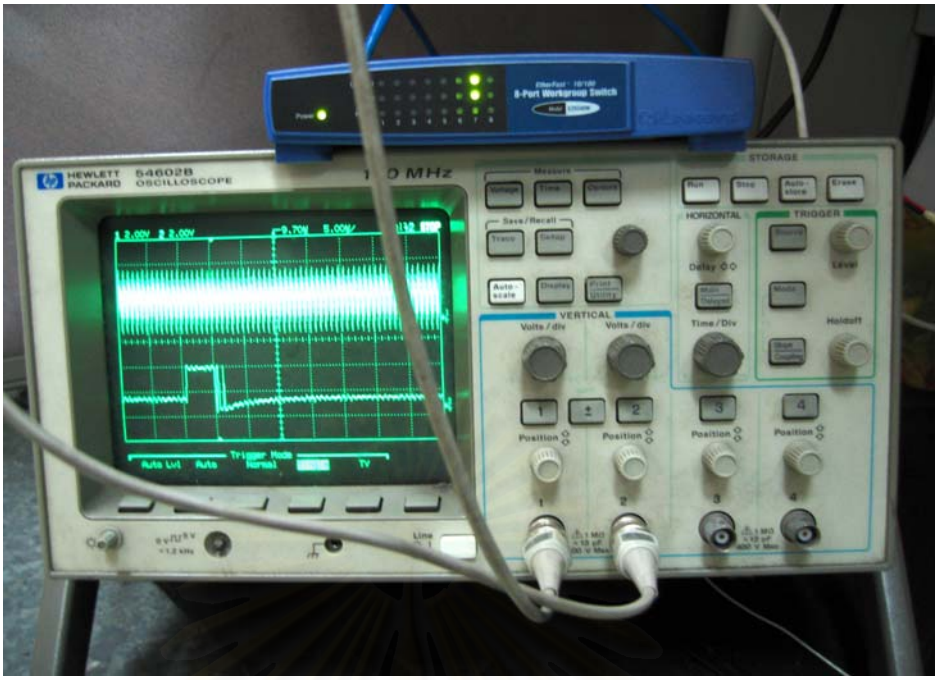
การทดลองทั้งหมดจะทำการทดลองทั้งบนการจำลอง (Simulation) และทดสอบการทำงานจริง ในแต่ละส่วนด้วย เครื่องออสซิโลสโคป โปรแกรมดักจับแพคเก็จ Ethereal[23] และเครื่องวิเคราะห์ตรรกะรุ่น 1670G ของบริษัท Agilent โดยการทดลองทั้งหมดจะสลับกันไปตามสถานการณ์ เช่นบางกรณีต้องการทดสอบการติดต่อสื่อสารก็จำเป็นต้องทดลองด้วยอุปกรณ์จริง หรือการทดสอบการอ่าน-เขียนเพื่อควบคุมเรจิสเตอร์ของ LXT972a เพราะการทดลองบนการจำลองข้อดีคือทำให้รู้ได้ว่าเขียนโปรแกรมเพื่อควบคุมถูกต้องเป็นไปตามคาดหรือไม่แต่การทำงานจริงอาจทำงานไม่ได้เพราะการเชื่อมต่อทางกายภาพไม่ดี หรือว่าแรงดันไฟฟ้าไม่ถึง ความยาวเส้นลวดยาวเกินไปซึ่งทำให้เกิดปัญหาเกี่ยวกับเวลาในการสื่อสาร ซึ่งเหตุการณ์เหล่านี้ไม่สามารถคาดเดาได้จากการจำลอง

5.1 การออกแบบการทดลอง

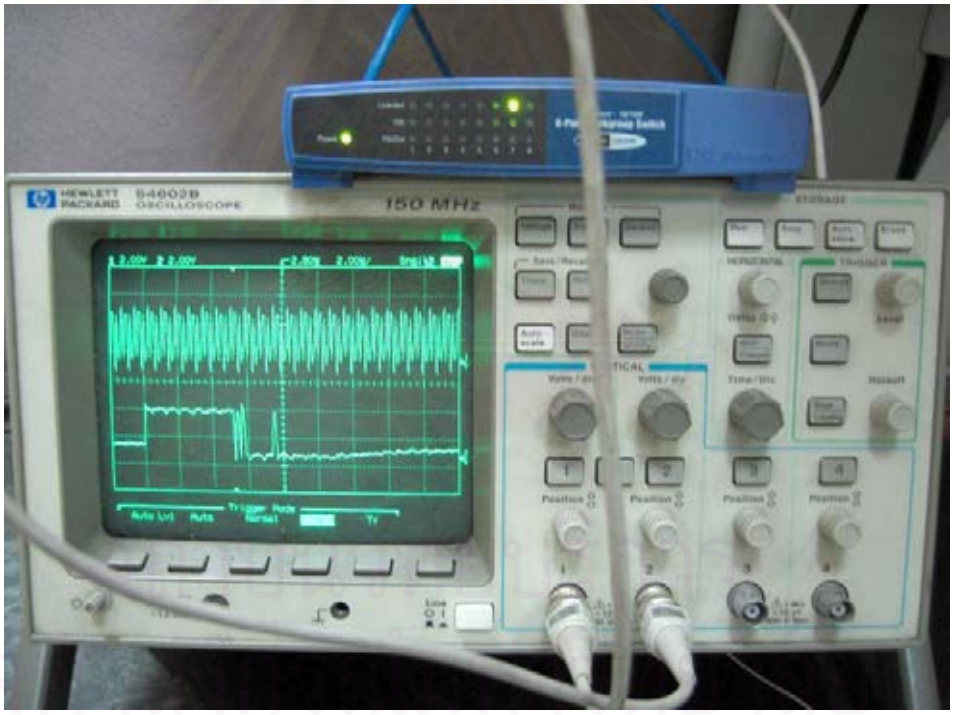
หัวข้อนี้แสดงถึงการออกแบบการทดลองในแต่ละส่วนของโครงงานรวมถึงการใช้เครื่องมือต่างๆ ในการวัด โดยในส่วนแรกจะใช้ออสซิโลสโคปในการวัดสัญญาณไฟฟ้าต่าง หรือสัญญาณในสายแลนต่างๆ เพื่อให้เกิดความผิดพลาดน้อยที่สุดในการเชื่อมต่อเพราะความผิดพลาดในกรณีนี้หลายๆครั้งเป็นเหตุให้อุปกรณ์อิเลคทรอนิกส์เสียหายได้ง่าย โดยค่อยๆวัดสัญญาณที่ละเส้นๆ ไปเรื่อยๆ จนครบหมดทุกสัญญาณ และก่อนหน้านั้นตอนวางอุปกรณ์อิเลคทรอนิกส์ที่เป็นแบบ Surface Mount การวางจะต้องตรวจสอบการเชื่อมต่อกับบอร์ดด้วยแว่นขยายหรือโอห์มมิเตอร์ เพราะหลายครั้งที่ดูเหมือนต่อแล้วแต่จริงๆยังต่อไม่สนิทหรืออาจไม่ต่อเลยก็ได้ซึ่งเป็นสาเหตุให้วงจรทำงานไม่ได้หรือเกิดการลัดวงจรขึ้นได้ และในส่วนอื่นจะเลือกใช้เครื่องมือที่มีอยู่ให้เหมาะสม โดยแบ่งออกเป็นหัวข้อย่อยต่างๆ ดังนี้

5.1.1 การทดสอบส่วน MII

ส่วนนี้ไม่มีความจำเป็นต้องสร้างขึ้นมาเพราะ LXT972a สามารถทำงานแบบต่อรองอัตโนมัติแต่จะไม่สามารถเลือกรูปแบบการทำงานต่างๆได้จึงต้องสร้างขึ้นมาเพื่อให้สามารถควบคุมการทำงานต่างๆได้ให้สะดวกขึ้น โดยในส่วนนี้ใช้ออสซิโลสโคป และเครื่องวิเคราะห์ตรรกะในการทดสอบการทำงาน แต่ที่จริงแล้วใช้เพียงแค่ออสซิโลสโคปในการวัดก็พอ แต่ในขณะเขียนโปรแกรมทดสอบต้องการดูพฤติกรรมของการออกแบบด้วยจึงมีการใช้เครื่องวิเคราะห์ตรรกะเข้ามาดูในการเปลี่ยนขั้นตอนการทำงานต่างๆ เพื่อความแม่นยำมากขึ้นในการทดสอบแต่การใช้เครื่องวิเคราะห์ตรรกะนั้นไม่เพียงพอในการดูค่าต่างๆ ของเรจิสเตอร์ภายใน LXT972a เพราะเครื่องมือชนิดนี้ไม่ได้แสดงสัญญาณเป็นศักย์ไฟฟ้าแต่แสดงเพียงสถานะ 0 หรือ 1 ซึ่งในช่วงแรกที่ควบคุมการอ่าน-เขียนเรจิสเตอร์ภายใน LXT972a จะดูไม่ออกว่าขณะที่อ่านค่าจากเรจิสเตอร์นั้นมีค่าเป็น 0 หรือค่า อิมพิแดนซ์สูง(hi-impedance)ึ่งหมายความว่าไม่สามารถติดต่อได้ดังรูปที่ 5.1 ขณะที่ออสซิโลสโคป จะแสดงถึงสถานะว่ามีสามารถติดต่อได้ดัง รูปที่ 5.2 ซึ่งแสดงให้เห็นว่าไม่สามารถให้ LXT972a ทำงานที่ความเร็วที่กำหนดได้โดยความเร็วที่กำหนดให้กับอุปกรณ์นี้สามารถสังเกตได้ที่สถานะของอุปกรณ์ทางด้านเครือข่ายจำพวก HUB หรือ SWITCHแล้วแต่จะเลือกใช้ แต่ในขณะที่รูปที่ 5.1 ไม่สามารถควบคุมความเร็วในการสื่อสารในเครือข่ายได้และระบบจะติดต่อแบบต่อรองอัตโนมัติและเลือกทำงานที่ความเร็วสูงสุดที่รองรับได้ แต่จากรูปที่ 5.2 ควบคุมให้อุปกรณ์ LXT972a ทำงานที่ความเร็ว 10 เมกะเฮิรตซ์อย่างเดียวทำให้เวลาต่อสายไฟเข้าไปที่อุปกรณ์สวิตซ์จะไม่ทำงานที่ความเร็ว 100 เมกะเฮิรตซ์ เพราะจากเดิมถ้าไม่ได้ควบคุมเรจิสเตอร์ตำแหน่งที่ 0 ให้ทำงานแบบ 10 เมกะเฮิรตซ์ LXT972a จะทำงานแบบต่อรองอัตโนมัติและเลือกที่ความเร็วสูงสุดคือ 100 เมกะเฮิรตซ์ ซึ่งสามารถดูได้จากการแจ้งเตือนสถานะบนอุปกรณ์สวิตซ์ได้โดยตรงตามที่สถานะของอุปกรณ์นั้นแจ้งไว้



รูปที่ 5.1แสดงการติดต่อเพื่ออ่านค่าที่ล้มเหลว

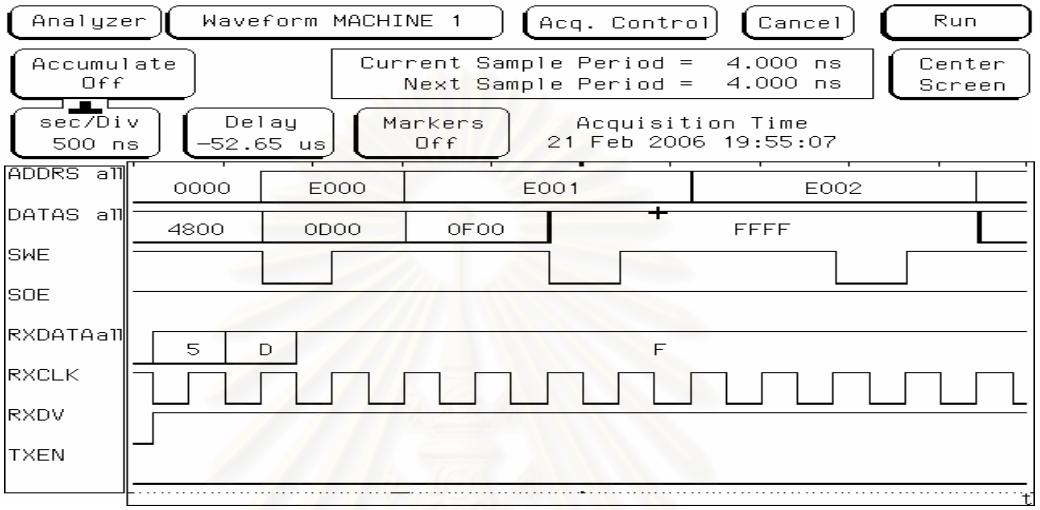


รูปที่ 5.2แสดงการติดต่อเพื่อควบคุม LXT972a สำเร็จ

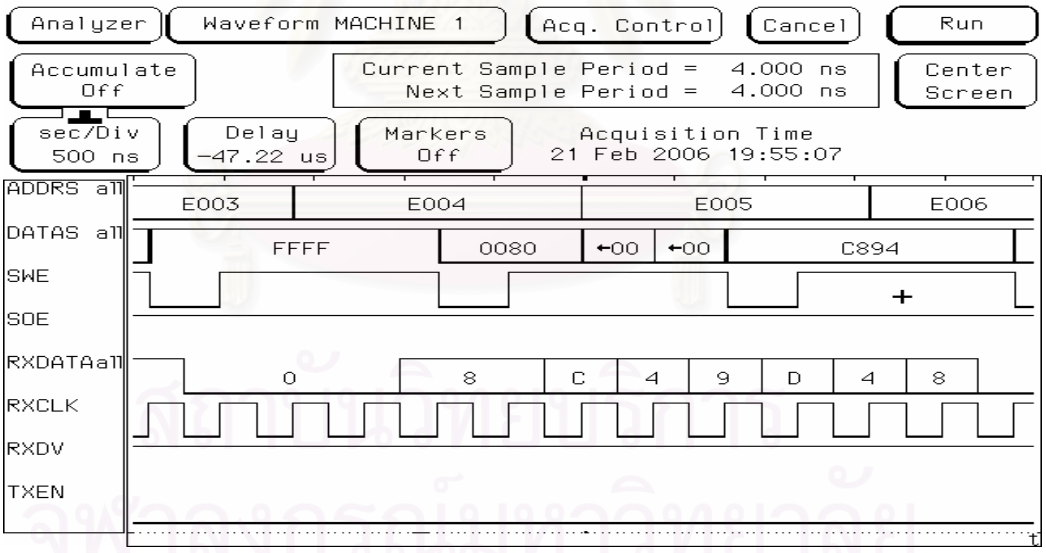
5.1.2 การทดสอบส่วนรับข้อมูลจากเครือข่าย

ในส่วนนี้จะทำงานตามที่กล่าวไว้แล้วในหัวข้อ 3.4.1 และจากการวางอุปกรณ์ต่างๆลงในบอร์ดทดลองที่แก้ไขจนมั่นใจแล้วว่าการเชื่อมต่อนั้นติดต่อถึงกันอย่างแน่นหนาและครบถ้วนถึงกันหมดทุกเส้นแล้ว และเนื่องจากส่วนรับข้อมูลนี้สามารถทำงานได้ด้วยตัวเองจึงไม่ต้องสร้างส่วน

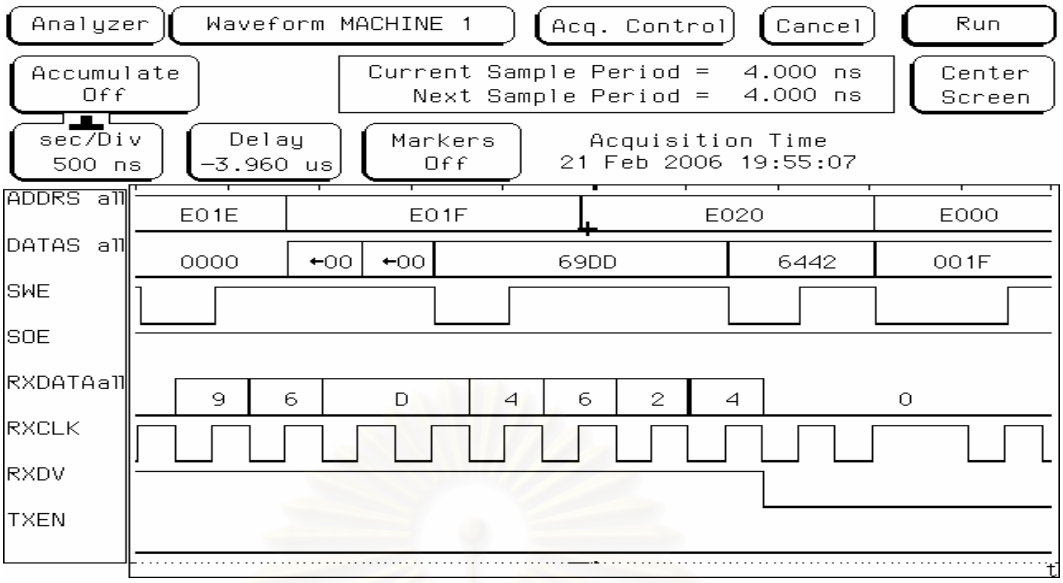
ควบคุมมาควบคุมการทำงานในการเขียนข้อมูลที่ได้รับเข้าไปในหน่วยความจำ แล้วหลังจากข้อมูลแต่ละแพคเก็จหมดก็ให้หน่วยประมวลผลทำงานต่อจากเดิมที่ทำงานค้างไว้ ดังนั้นการทดสอบจึงใช้เครื่องวิเคราะห์ตรรกะในการดูการจัดเรียงข้อมูลที่เข้ามา(จาก 4 บิตไปเป็น 16 บิต) และการเขียนไปที่ตำแหน่งต่างๆ ในหน่วยความจำ ดังรูปที่ 5.3 รูปที่ 5.4 และ รูปที่ 5.5 แสดงการรับข้อมูลในกรณีต่างๆ



รูปที่ 5.3 รูปแสดงการรับข้อมูลในส่วนหัว



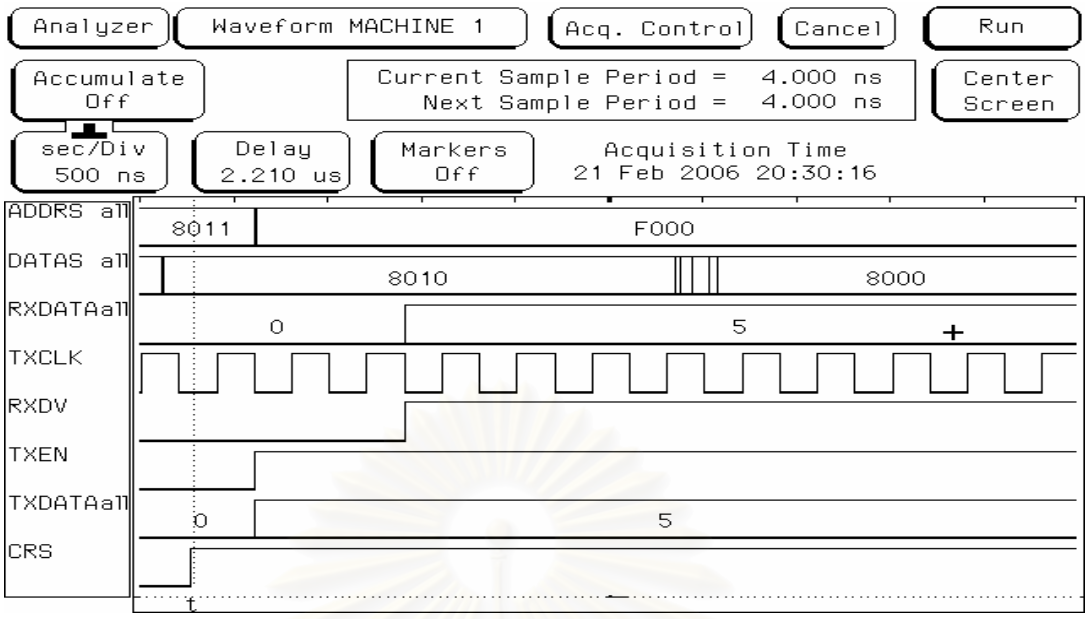
รูปที่ 5.4 แสดงการแปลงข้อมูลที่เข้ามาจากเครือข่ายลงในหน่วยความจำ



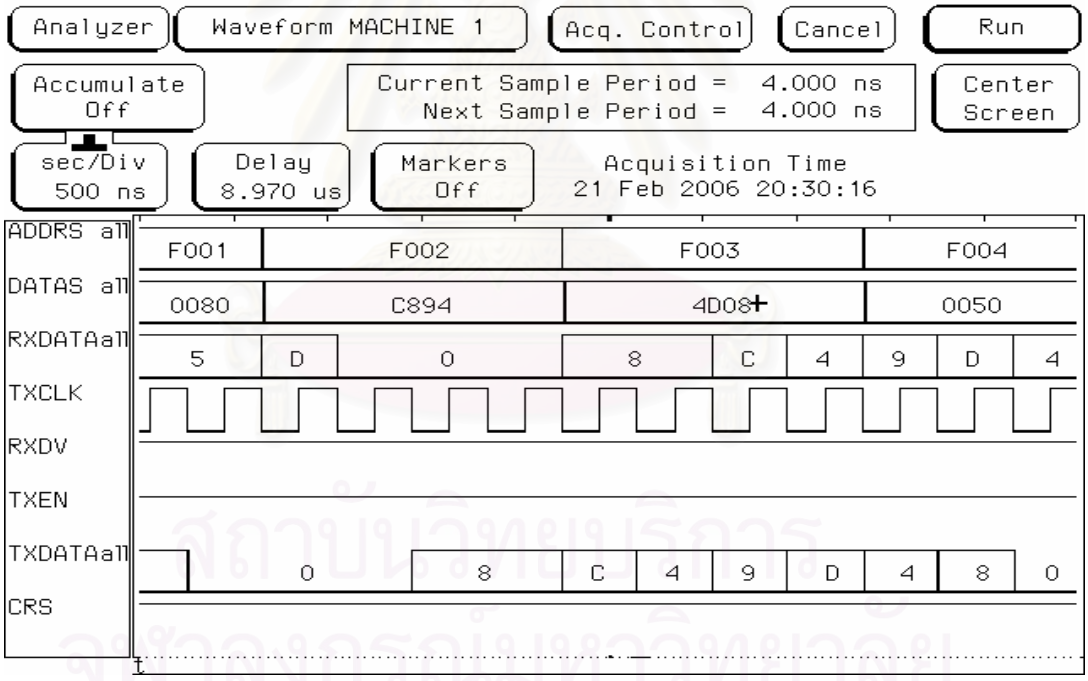
รูปที่ 5.5 แสดงการรับข้อมูลส่วน CRC พร้อมทั้งเขียนความยาวไว้ที่ส่วนหน้าของชุดข้อมูล

5.1.3 การทดสอบส่วนส่งข้อมูลเข้าไปในเครือข่าย

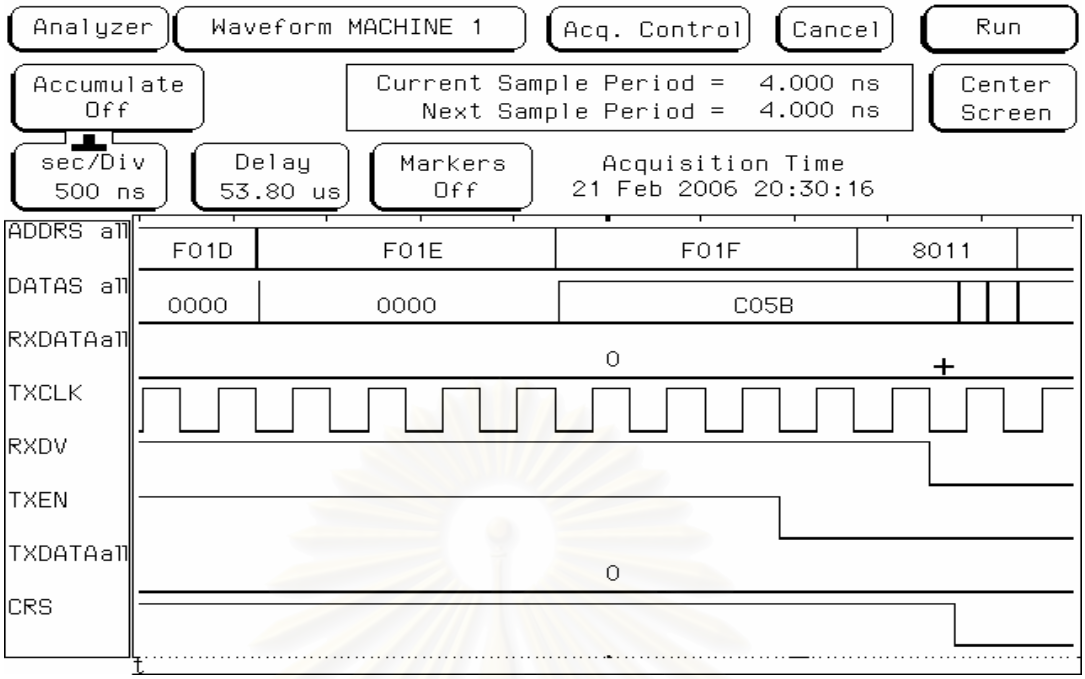
ในส่วนนี้เช่นเดียวกันกับหัวข้อ 5.1.2 การทดสอบจึงไม่ยุ่งยากนักเพียงแค่ส่วนส่งข้อมูลไม่สามารถส่งข้อมูลออกได้ด้วยตัวเองจึงต้องสร้างตัวควบคุมตัวส่งข้อมูลขึ้นมาเพื่อส่งค่าความยาวและส่งสัญญาณให้ส่งข้อมูลก็จะส่งข้อมูลได้แล้ว ซึ่งก็เหมือนกับการเขียนความยาวไปที่ตำแหน่ง DFFFh ในส่วนของระบบทั้งหมดก็จะมีฮาร์ดแวร์พิเศษเพื่อทำหน้าที่นี้ให้เพื่อลดการทำงานของหน่วยประมวลผล และ LXT972a สามารถปรับให้ทำงานได้ในแบบส่งข้อมูลกลับเข้ามา (Loop Back) คือส่งข้อมูลอะไรออกไปทางขา Tx ข้อมูลที่ส่งไปนั้นก็จะเข้ามาทางขา Rx เพื่อความสะดวกในการแก้ไขจึงปรับให้ LXT972a ทำงานแบบส่งข้อมูลกลับ โดยมีผลออกมาตามรูปที่ 5.6 รูปที่ 5.7 และ รูปที่ 5.8



รูปที่ 5.6 แสดงการส่งข้อมูลในส่วนหัว



รูปที่ 5.7 แสดงการจัดเรียงการส่งข้อมูลจากหน่วยความจำออกไปที่ LXT972a



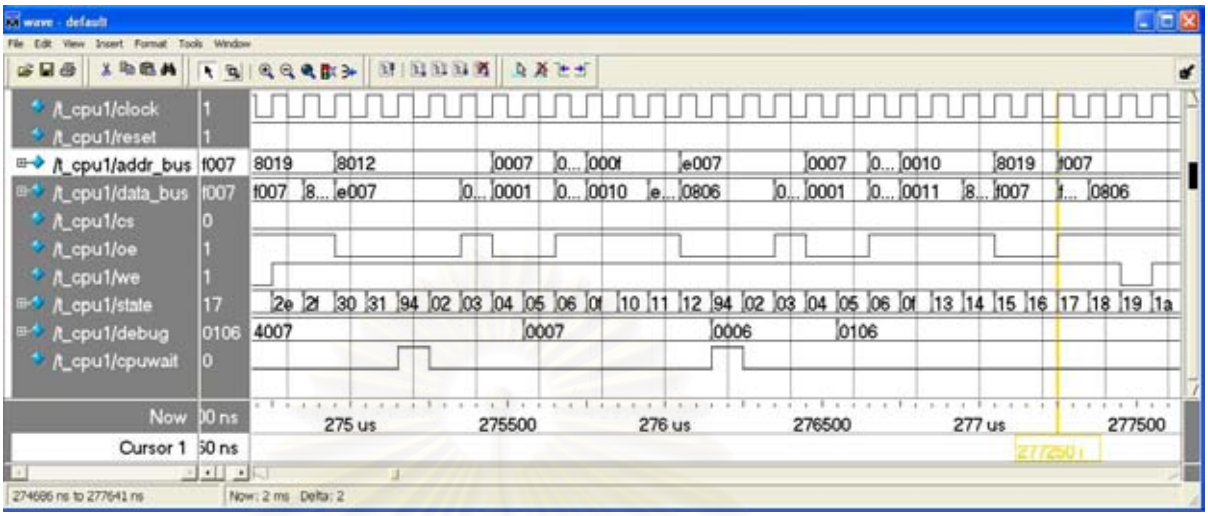
รูปที่ 5.8 แสดงการส่งข้อมูลสุดท้ายของแพคเก็จ

ซึ่งการส่งข้อมูลนี้สามารถปรับเปลี่ยนได้ตามที่ระบุไว้ในคู่มือโดยที่รูปที่แสดงนี้แสดงการทำงานที่ความถี่ 10 เม็กกะบิตต่อวินาทีแต่ถ้าปรับให้ทำงานที่ความถี่ 100 เม็กกะบิตต่อวินาทีก็จะไม่สามารถทำงานในรูปแบบการทำงานนี้ได้เพราะอุปกรณ์ LXT972a นี้ไม่สามารถทำงานแบบวนกลับ (loop back) ได้ที่ความเร็วนั้น และเพื่อความมั่นใจอาจใช้ออสซิโลสโคปในการวัดที่ข้อมูลขาออกทางที่ต่อกับสายคู่ไขว้เพื่อความถูกต้องได้อีกด้วย

5.1.4 การทดสอบส่วนหน่วยประมวลผล

เป็นการทดสอบก่อนที่นำไปใช้งานจริงด้วยโปรแกรมโมเดลซิม ซึ่งเป็นเครื่องจำลองที่มาพร้อมกันกับเครื่องมือที่ใช้พัฒนาโครงงาน แล้วค่อยการทดสอบการทำงานจริงบนบอร์ดทดลองซึ่งการทดสอบการทำงานด้วยเครื่องจำลองนั้นอาจยังไม่เพียงพอ และจากการเขียนโปรแกรมที่ผ่านๆ มาพบว่าหลายๆครั้งที่โปรแกรมสังเคราะห์วงจรจะตัดส่วนของโปรแกรมบางส่วนออกไปเองเนื่องจากเขียนโปรแกรมไม่รัดกุมพอจึงอาจทำงานได้แต่ในเครื่องจำลองเท่านั้นแต่การทดสอบในเครื่องจำลองก็สามารถดูพฤติกรรมการทำงานของหน่วยประมวลผลว่าทำงานตามที่ต้องการหรือไม่ ทดสอบการเขียนโปรแกรมว่าสามารถเข้าถึงตำแหน่งต่างๆภายในหน่วยความจำได้ตามต้องการหรือไม่เพราะในการใช้เครื่องวิเคราะห์ตรรกะ นั้นเก็บข้อมูลได้ไม่ทั้งหมดจะเก็บได้เป็นช่วงข้อมูลซึ่งหลายๆ ครั้งยาวไม่พอและการเลื่อนไปดูการทำงานที่เวลาต่างๆ ยากกกว่าใช้เครื่องจำลองอีกด้วยแต่ข้อดีที่ได้คือแม่นยำกว่ามากและในการทดสอบด้วยเครื่องจำลองมีการสร้างรูปแบบ

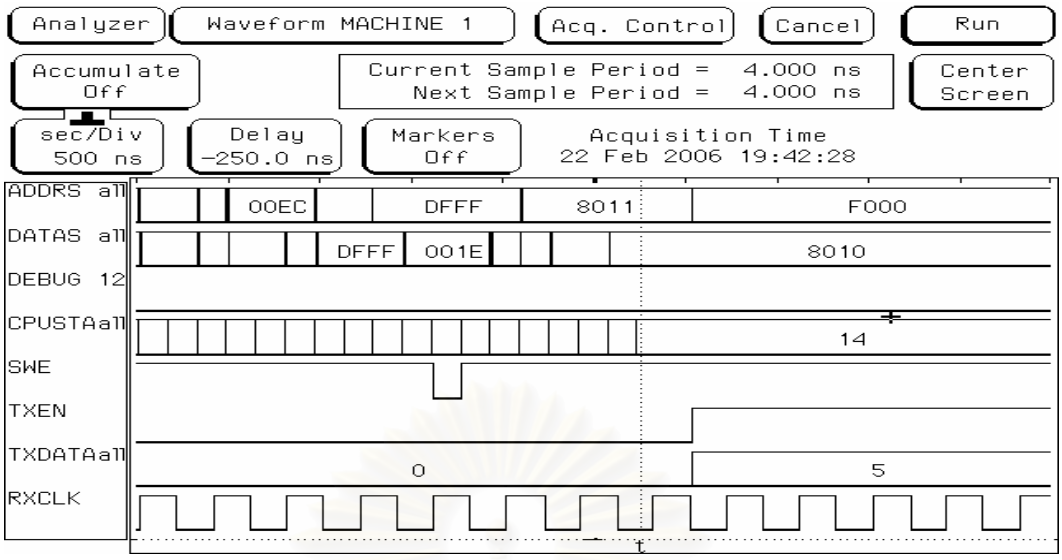
ข้อมูลไว้ตามแพคเก็จที่เข้ามาในหน่วยความจำตำแหน่งที่ระบุไว้ด้วยตามรูปที่ 5.9 ที่แสดงการทำงานบางส่วนของหน่วยประมวลผล



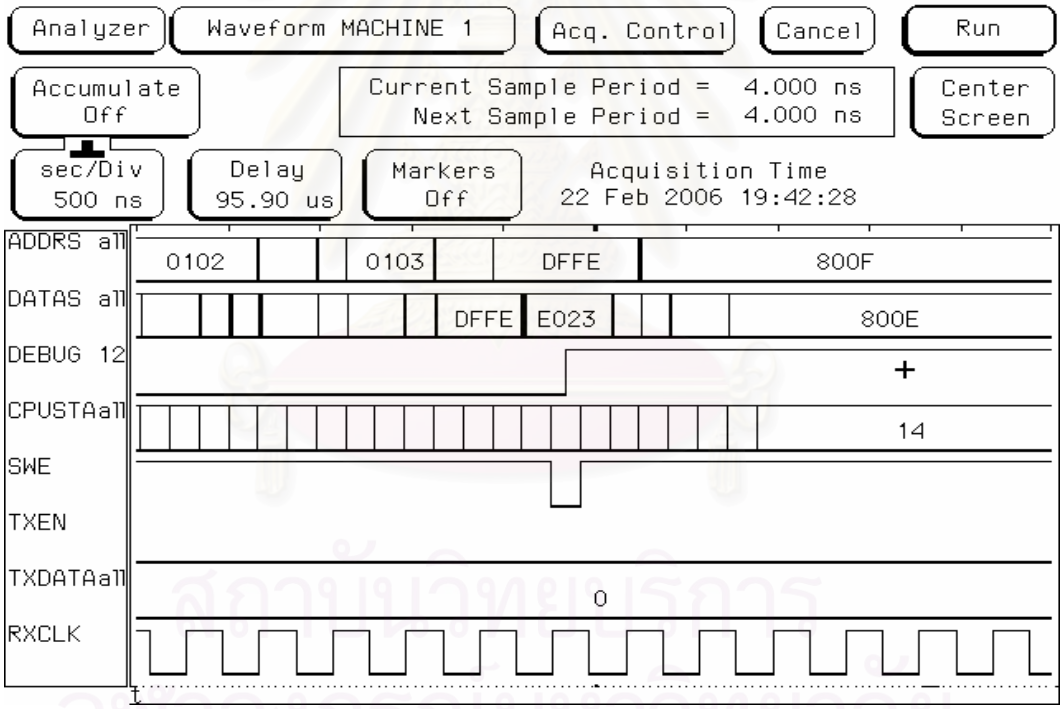
รูปที่ 5.9 แสดงแผนภูมิเวลาของหน่วยประมวลผลในการจัดเรียงข้อมูลเพื่อการส่งออก

5.1.5 การทดสอบส่วนการควบคุมการทำงานส่วนรับ-ส่วนส่ง-หน่วยประมวลผล

อย่างที่กล่าวไว้แล้วในหัวข้อ 3.4.4 โดยสามารถทดสอบได้ทั้งการใช้เครื่องจำลองหรือใช้เครื่องวิเคราะห์ตรรกะเพราะการทำงานส่วนนี้เป็นเสมือนการจัดลำดับการทำงานให้ส่วนต่างๆ และส่วนนี้ไม่สามารถทำงานได้ด้วยตัวเองเพราะต้องรอข้อมูลจากเครือข่าย ในส่วนของเครื่องจำลองต้องสร้างตัวกำเนิดสัญญาณต่างๆ ขึ้นมาเสมือนมีชุดข้อมูลเข้ามาจากเครือข่ายดังเช่นรูปที่ 5.9 จะได้เห็นได้ว่าจะมีสัญญาณ cpuwait เกิดขึ้นตลอดเวลาก่อนที่จะดึงคำสั่งชุดใหม่เข้ามาประมวลผลใหม่เพราะที่จุดนี้ (cpuwait = 1) หน่วยประมวลผลเองจะไม่ติดต่อหน่วยความจำ เมื่อหน่วยประมวลผลเรียงข้อมูลเรียบร้อยแล้ว ก็จะส่งข้อมูลออกไปโดยเขียนจำนวนข้อมูลไปตำแหน่งที่ระบุไว้ตามหัวข้อ 3.4.2 และมีผลการทำงานตามรูปที่ 5.10 และ รูปที่ 5.11 ที่เป็นการแสดงการควบคุมการส่งข้อมูลและหยุดรอให้หน่วยประมวลทำงานตามรูปที่ 5.10 ส่วนในรูปที่ 5.11 เป็นการแสดงการทำงานเมื่อหน่วยไม่มีข้อมูลจากเครือข่ายที่ยังไม่ถูกประมวลผลในส่วน IBUFจากรูปที่ 5.10 และ รูปที่ 5.11 สัญญาณ DEBUG12 จะเป็นสัญญาณแสดงสถานะว่างของ IBUF ตามที่ระบุไว้ในหัวข้อ 3.4.4 และสาเหตุที่ให้ออกแบบให้ทำเช่นนี้เพราะว่าพฤติกรรมของเครือข่ายนั้นเป็นพฤติกรรมที่คาดเดาไม่ได้จึงมีโอกาสที่ข้อมูลจะเข้ามาขณะที่หน่วยประมวลผลทำงานอยู่หรือหน่วยประมวลผลทำงานในขณะที่มีข้อมูลเข้ามาทางเครือข่ายและเขียนลงหน่วยความจำพอดีซึ่งทั้งสองกรณีก็จะแย่งกันใช้หน่วยความจำทั้งสิ้นและขณะนี้หน่วยประมวลผลก็ไม่มีงานรอให้ทำนอกจากประมวลผลข้อมูลทางเครือข่ายที่เป็นงานหลัก



รูปที่ 5.10 แสดงการส่งข้อมูลออกทางเครือข่ายและให้หน่วยประมวลผลรอ



รูปที่ 5.11แสดงการเลื่อนตัวชี้ข้อมูลขาเข้าเพื่อให้หน่วยประมวลผลรอ

5.2 ผลการทดลอง

หัวข้อนี้แสดงผลการทดลองที่ได้จากการทดสอบจริงโดยแบ่งออกเป็น 2 ส่วน เป็นผลการใช้ทรัพยากรของระบบ และเวลาในการตอบสนองของแต่ละแพคเก็จ โดยที่ในการทดสอบทางด้านเวลาในการตอบสนองมีการตั้งระบบเครือข่ายขึ้นมาใช้ในวงเครือข่ายภายในเองเพื่อเลี่ยงปัญหาต่างๆที่เกิดจากแพคเก็จไม่พึงประสงค์ที่เข้ามาทางเครือข่ายข้างนอก ระบบที่สร้างขึ้นมานี้

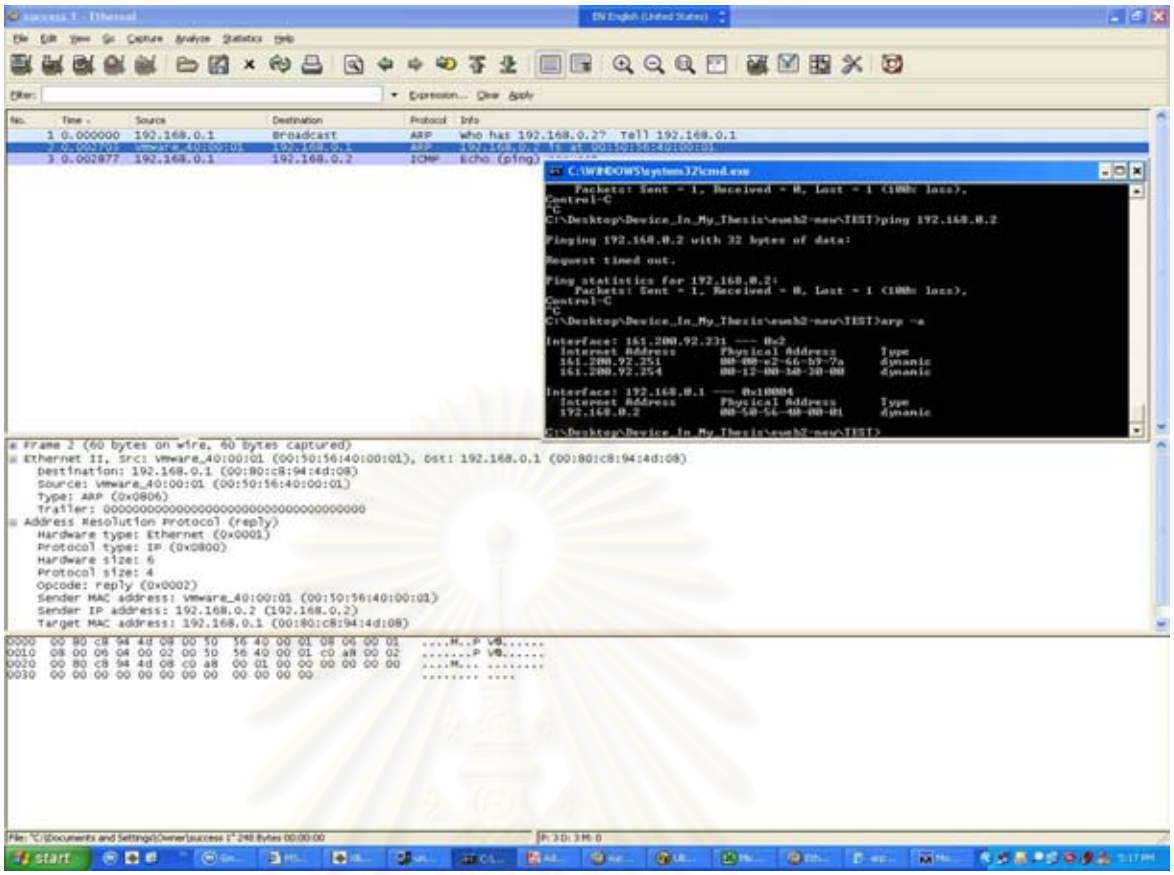
ประกอบด้วย บอร์ดทดลองและบอร์ด FPGA ที่ทำหน้าที่เป็นตัวตอบกลับ ICMP เครื่องคอมพิวเตอร์โดยทำหน้าที่สังเคราะห์หน่วยประมวลผลและระบบต่างๆที่จะทำการทดสอบลงบนบอร์ด FPGA และอุปกรณ์เครือข่าย Switch หรือ HUB ซึ่งหลังจากที่สร้างระบบขึ้นมาแล้วก็ได้ทำการทดสอบต่างๆทีละส่วนตามที่กล่าวมาอย่างถูกต้องแล้ว ก็เริ่มทำการทดสอบระบบโดยรวม

การทำงานในส่วนหน่วยประมวลผลจะวัดเวลาในการทำงานตั้งแต่ข้อมูลเข้ามาจนส่งข้อมูลคืนออกไป ซึ่งโดยปกติทำโดยเมื่อไม่มีข้อมูลเข้ามาในหน่วยความจำช่วงที่เตรียมไว้ของประมวลผล หน่วยประมวลผลจะไม่เริ่มทำงานหรือไม่ทำงานต่อจะต้องหยุดรอไว้ก่อน ดังนั้นการทดสอบการทำงานจะวัดโดยดูเวลาหลังจากข้อมูลเข้ามาทางเครือข่ายถูกเขียนลงในหน่วยความจำเรียบร้อยแล้วครบเฟรม จนกระทั่งตอบกลับออกไปตามชนิดของข้อมูลที่เข้ามานั้นๆ ตามที่ได้เสนอไว้ใน 4.2 และจากการทำงานของหน่วยประมวลผลที่ความถี่ 6.25 เมกะเฮิรตซ์ พบว่าใช้เวลาในการตอบสนองแต่ละชนิดข้อมูลดังตารางที่ 5.1 ซึ่งก็เป็นเหตุเป็นผลที่สมควรเพราะแพคเก็จ ICMP นั้นยาวกว่าและต้องผ่านการเรียกใช้โปรแกรมย่อยถึง 3 ครั้งแต่การทำงานตามที่ออกแบบมานี้เวลาที่เสียไปส่วนใหญ่เกิดจากการรอข้อมูลที่เข้ามาจากเครือข่าย

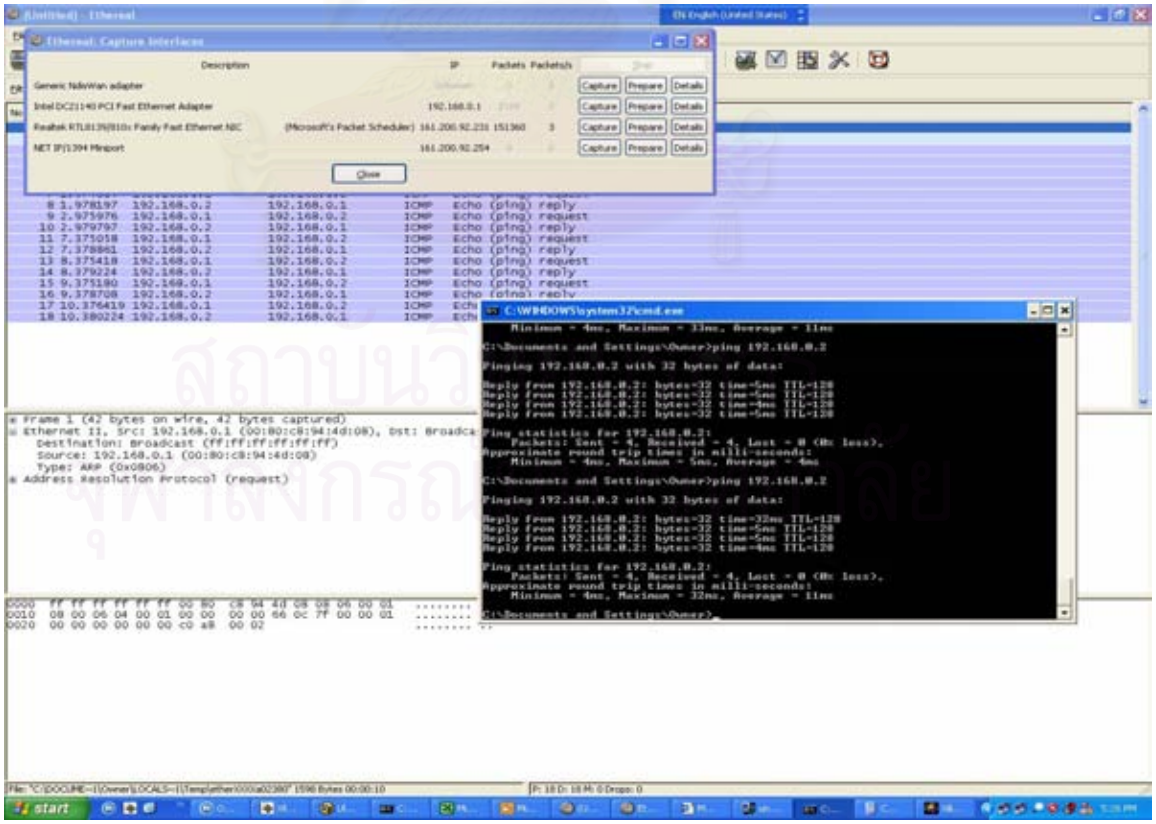
ตารางที่ 5.1 แสดงเวลาการตอบสนองของหน่วยประมวลผลตามชนิดข้อมูลที่เข้ามา

ชนิดแพคเก็จที่เข้ามา	เวลาในการตอบสนอง (ไมโครวินาที)
ARP	332
ICMP	2407

ในการทดลองโดยใช้คำสั่ง Ping จากเครื่องคอมพิวเตอร์ส่วนบุคคลไปที่วงจรที่ออกแบบพบว่าวงจรที่ออกแบบสามารถตอบสนองได้เป็นอย่างดีตามรูปที่ 5.12 และรูปที่ 5.13 ส่วนที่คอมพิวเตอร์ส่วนบุคคลใช้โปรแกรม Ethereal ช่วยจับข้อมูลที่สื่อสารเพราะก่อนที่จะทำงานได้อย่างสมบูรณ์นั้นที่หน้าจอหลังจากคำสั่ง PING จะไม่ปรากฏข้อความใดๆ ทำให้ดูเหมือนว่าไม่สามารถติดต่อได้ทั้งๆที่ติดต่อได้แต่มีบางส่วนที่ผิดพลาดไปเช่นการคำนวณค่า Checksum ผิดพลาด แต่ทั้งนี้ทั้งนั้นข้อมูลในส่วน หมายเลขปลายทาง หมายเลขต้นทางและหมายเลข CRC จะต้องถูกต้องไม่เช่นนั้นระบบปฏิบัติการจะปฏิเสธข้อมูลนั้นๆ ทิ้งไปโดยโปรแกรม Ethereal ก็ไม่แสดงข้อมูลด้วยเพราะตามที่คู่มือของโปรแกรมบอกมาว่าโปรแกรมนี้จะอยู่หลังการทำงานของระบบปฏิบัติการอีกชั้นจึงไม่สามารถแสดงข้อมูลได้แต่สามารถสื่อสารได้ เช่นในรูปที่ 5.12 ที่การคอมพิวเตอร์ส่วนบุคคลสามารถรับได้แต่แพคเก็จแบบ ARP เพราะเกิดความผิดพลาดในส่วนของ ICMP อยู่เล็กน้อย



รูปที่ 5.12 แสดงการติดต่อด้วยแพคเก็จ ARP สำเร็จ



รูปที่ 5.13 แสดงการสื่อสารด้วยคำสั่ง Ping สำเร็จ

และในตารางที่ 5.2 แสดงการใช้ทรัพยากรในโมดูลต่างๆ ที่สร้างขึ้นมา เห็นได้ว่าส่วนของการกำหนดค่าเริ่มต้นของหน่วยความจำในหน่วยประมวลผลจะใช้ทรัพยากรมากเพราะมีการสร้างหน่วยความจำขึ้นมาภายในขนาด 216 กิโลบิตคือใช้ทั้งหมดของหน่วยความจำภายในที่มีให้ของวงจรรวม Spartan 3 เพื่อนำไปใช้เขียนเป็นโปรแกรมลงในหน่วยความจำประเภทแรมบนบอร์ด FPGA แต่ในความเป็นจริงแล้วโปรแกรมที่ทำทั้งหมดสามารถให้ค่าเริ่มต้นได้ด้วยการใช้หน่วยความจำประเภทรอมในการเก็บข้อมูลข้อมูลส่วนนี้จึงถือว่าไม่ได้เป็นจำนวนเรจิสเตอร์ที่ใช้จริงในการสังเคราะห์วงจร และในส่วนควบคุมการทำงานไม่สามารถสังเคราะห์ให้เห็นความถี่ที่สังเคราะห์ที่แท้จริงได้เพราะไม่สามารถแยกออกมาสังเคราะห์เป็นส่วนๆ ได้ แต่การทำงานในการทดลองจริงนั้นใช้ความถี่สัญญาณนาฬิกาที่ 50 เมกะเฮิรตซ์ป้อนให้กับระบบทั้งหมดก็ยังคงทำงานได้อย่างดีอยู่

ตารางที่ 5.2 แสดงการใช้ทรัพยากรในแต่ละโมดูล

โมดูล	จำนวนเกทสมมูลที่ใช้	ความเร็วที่สังเคราะห์ได้
หน่วยประมวลผล	6,485	56.376 MHz
ส่วนรับข้อมูล	2,568	143.596MHz
ส่วนส่งข้อมูล	1,194	181.439MHz
ส่วนคำนวณค่า CRC	2,869	91.724MHz
ส่วนกำหนดค่าเริ่มต้นของหน่วยความจำในหน่วยประมวลผล	XXX(524,875)	107.921MHz
ส่วนควบคุมการทำงาน	981	XXX (32.614MHz)
ระบบโดยรวม	14,097(538,972)	32.463MHz

5.3 **สรุปผลการทดลอง**

จากการทดลอง ผลการตอบสนองพบว่าสามารถตอบสนองได้เป็นอย่างดีและแต่ละส่วนสามารถทำงานได้อย่างถูกต้องและประหยัดทรัพยากร [25] และสามารถพัฒนาให้มีการทำงานอื่นนอกเหนือจากระบบที่ออกแบบในปัจจุบัน เช่นในการรับข้อมูลสามารถออกแบบให้ไม่รับข้อมูลที่ไม่ใช่ของตัวเองได้ ซึ่งไม่ต้องให้หน่วยประมวลผลต้องมาจัดการกับแพคเก็จที่ไม่ใช่ของตัวเอง หรือ

สามารถสังเคราะห์วงจรเฉพาะกิจเพื่อทำงานพิเศษบางชนิดที่หน่วยประมวลผลไม่สามารถทำงานได้หรือสามารถทำงานได้แต่ไม่รวดเร็ว

เมื่อเปรียบเทียบกับวิธีการต่างๆที่หน่วยประมวลผลอื่นใช้ ก็เห็นได้ว่าเหมือนกัน แต่การสร้างส่วนต่างๆขึ้นมาเองนั้นสามารถทำงานอื่นได้อย่างดี เช่นในการสร้างคุณสมบัติที่อุปกรณ์ชนิดนั้นๆไม่มีลงไปได้ ดังเช่นการสร้างตัวติดต่อ MII ที่ไม่มีให้ใช้วงจรรวม Spartan3 แต่จะมีให้ในตระกูลอื่นที่แพงกว่าเช่น Vertex4 หรือ ไมโครคอนโทรเลอร์ระดับสูงอย่าง ARM9 เป็นต้น

บทที่ 6

สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

วิทยานิพนธ์นี้นำเสนอวิธีการออกแบบหน่วยประมวลผลและอุปกรณ์ภายนอก รวมทั้งการพัฒนาโปรแกรมขึ้นมาใช้เองโดยมีจุดประสงค์ให้วงจรที่ตอบสนองทางด้าน ICMP ที่เป็นประโยชน์อย่างยิ่งในการพัฒนาระบบควบคุมแบบฝังตัวทั่วไป เนื่องจากสามารถประยุกต์ดัดแปลงให้เข้ากับอุปกรณ์ต่างๆได้ง่าย ผลพลอยได้คือได้วงจรขนาดเล็กเพื่อใช้เป็นระบบฝังตัว จุดประสงค์ที่ต้องการคือวงจรมีขนาดเล็กจึงและโปรแกรมให้มีขนาดเล็กด้วย เพราะหน่วยประมวลผลมีหน่วยความจำน้อย ซึ่งการลดขนาดของหน่วยความจำที่ใช้นี้ลงได้ก็จะทำให้วงจรรวมใช้พื้นที่น้อยซึ่ง ส่งผลให้ต้นทุนในการผลิตชิปลดลงด้วย ซึ่งการออกแบบให้ใช้พื้นที่ในวงจรรวมน้อยนั้นมีทรัพยากรเหลือที่จะสังเคราะห์อุปกรณ์อื่นๆ ภายนอกหน่วยประมวลผลเพิ่มเติมลงไปได้อีกด้วย

ระบบตอบสนอง PING ในวิทยานิพนธ์นี้ใช้ทรัพยากรจำนวน 14097 เกทสมมูลและหน่วยประมวลผลทำงานที่ความถี่ 6.25 เมกะเฮิรตซ์และเวลาตอบสนองที่ 4 มิลิวินาที

6.2 การประยุกต์การใช้งาน

ผู้ใช้งานสามารถพัฒนาโปรแกรมต่อเพื่อใช้ทำงานในแบบอื่นๆ ได้เพราะในเกณฑ์วิธี ICMP นั้นไม่จำกัดอยู่เพียงแค่คำสั่ง PING รูปแบบเดียวแต่ในการใช้คำสั่ง PING นั้นสามารถใช้การส่งเรียกร้องและตอบสนองแบบอื่นได้เช่นมีการส่งข้อความติดไปด้วยซึ่งเพียงเท่านี้ก็สามารถนำไปใช้ในการควบคุมวงจรง่ายๆ ได้บ้างแล้วและส่วนต่อๆไปจะเป็นงานที่เน้นหนักทางด้านซอฟต์แวร์มากกว่า ที่จะพัฒนาทางด้านฮาร์ดแวร์

6.3 แนวทางในการปรับปรุงและข้อเสนอแนะ

สามารถทำให้ระบบอัตโนมัติมากขึ้นเช่นในส่วนรับ-ส่งข้อมูลอาจสังเคราะห์หน่วยความจำเข้าไปให้ทำงานได้แบบ DMA จะได้ไม่ต้องเสียเวลารอระหว่างการรับ-ส่งข้อมูลทางด้านเครือข่าย หรือในการพัฒนาหน่วยประมวลผลให้มีสมรรถนะเพิ่มขึ้นก็สามารถทำได้ [27] [28] และเนื่องจากที่เคยเสนอไปว่าภาษา ระดับสูงกับภาษาแอสแซมบลีของหน่วยประมวลผลมีช่องว่างระหว่างภาษาน้อย จึงเป็นแนวทางอันดีที่จะพัฒนาด้วยภาษาระดับสูงเพราะหลังจากที่ผ่านตัวแปลภาษามาแล้วก็ยังได้ภาษาเครื่องที่มีขนาดไม่ต่างจากที่เขียนเป็นภาษาแอสแซมบลีด้วยตัวเอง

รายการอ้างอิง

1. เกริก ภิรมย์โสภา. (2543). การพัฒนาระบบเว็บเซิร์ฟเวอร์แบบฝังตัวที่สามารถจัดรูปลักษณ์ใหม่ได้. วิทยานิพนธ์ปริญญามหาบัณฑิต. ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย.
2. Zuchowski, P. S., Reynolds, C. B., Grupp R. J., Davis S. G., Cremen B., and Troxel B., A Hybrid ASIC and FPGA Architecture. Proceeding of SIGDA, IEEE,2002
3. Philip K. Jr., Stack computer the new wave. (n.p.).1989
4. Atmel Corporation. AVR-460[Online]. (n.d.). Available from: http://www.atmel.com[2004, March 20]
5. Vincent S. WWWpic2[Online]. (n.d.). Available from: http://www.kyllikki.org[2004, March 20]
6. Berkley University. Tiny web server[Online]. (n.d.). Available from: http://www.cs.berkley.edu[2004, March 19]
7. Standford University, Stanford Match Box[Online]. (n.d.). Available from: http://www-ccs.cs.umass.edu[2004, March 19]
8. uClinux, Rt-uCsimm[Online]. (n.d.). Available from: http://www.uclinux.org[2004, March 19]
9. ปิติพันธุ์ หล่อจีระชุณห์กุล, 2548. ADM6996 Ethernet Switch[Online]. (n.d.). Available from: http://www.nectec.or.th[2005, January 30]
10. Furber S., (1996). ARM system architecture, Boston: Addison-Wesley.
11. Medial Independent Interface, IEEE 802.3 – 2000 section 22; 2000.
12. Maxim Corporation, DS80C390[Online]. (n.d.). Available from: http://www.dalsemi.com[2004, March 19]
13. Intel Corporation, LXT972a[Online]. (n.d.). Available from: http://www.intel.com[2005, January 30]
14. Burutarchanai, A., Nanthanavoot, P., Aporntewan, C., and Chongstitvatana, P., "A stack-based processor for resource efficient embedded systems", Proceeding of IEEE TENCON 2004, 21-24 November 2004, Thailand.
15. Page M., SuperMac[Online]. (n.d.).2004. Available from:

http://www.sonyoxford.co.uk[2006, March 3]

16. Stevens W, Wright G., TCP/IP Illustrated. Volume2. Canada. : Addison-Wesley Publishing Company,1995
17. Rhys H., ICMP[Online]. (n.d.). Available from: http://www.rhyshaden.com/icmp.htm[2005, January 19]
18. Burutarchanai, A., Nanthanavoot, P. and Chongstitvatana, P., "Design of an embedded TCP/IP internet appliance", 8th National Computer Science and Engineering Conference, Bangkok, Thailand, October 27-28, 2005.
19. Intel Corporation, MCS 51 MICROCONTROLLER FAMILY USER'S MANUAL. February 1994
20. ภานุพันธ์ นันทนาวุฒิ. (2547). การลดขนาดโปรแกรมในระบบฝังตัวโดยใช้วงจรแปลงหรัส. วิทยานิพนธ์ปริญญามหาบัณฑิต. ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย.
21. Burutarchanai, A., Kotrajaras, V. and Chongstitvatana, P., "A fast instruction fetch unit for an embedded stack processor", Proceedings of International Conference on Information and Communication Technologies (ICT 2004), 18-19 November, 2004. Thailand.
22. Burutarchanai, A., and Chongstitvatana, P., "Design of a two-phased clocked control unit for performance enhancement of a stack processor", National Computer Science and Engineering Conference, Thailand, 21-22 Sept. 2004, pp.114-119.
23. Ethereal Corporation. Ethereal[Online]. (n.d.) Available from: http://www.ethereal.com[2005, December 1]

ภาคผนวก

ภาคผนวก ก

รายละเอียดภาษาส้ม

จุดประสงค์ของการพัฒนาภาษาส้ม นั้นเริ่มมาจากการพัฒนาภาษาคอมพิวเตอร์เพื่อใช้ในการศึกษาวิชาการออกแบบภาษาคอมพิวเตอร์และการพัฒนาโปรแกรมแปลภาษา (Compiler) ภาษาส้มจึงถูกออกแบบมาให้สามารถเข้าใจความหมายได้ง่าย

ตัวดำเนินการ (Operator) ของภาษาส้มมีลักษณะเป็นสัญกรณ์เติมกลาง (Infix notation) และมีรูปแบบของไวยากรณ์จำนวนน้อย ทำให้สามารถทำความเข้าใจและเรียนรู้ภาษานี้ได้ไม่ยาก

พื้นฐานการทำงานในภาษาส้มจะอยู่ที่นิพจน์ (Expression) โดยในแต่ละนิพจน์จะคืนค่าการทำงานของแต่ละนิพจน์ออกมา ภาษาส้มมีคำสงวนอยู่ 6 คำได้แก่ **to, if, else, while, for** และ **case** และมีตัวดำเนินการดังแสดงไว้ในตารางที่ ก.1

ตารางที่ ก.1 ตัวดำเนินการของภาษาส้ม

Operation	Operator
Arithmetic operation	+, -, *, /, %
Bitwise operation	& , | , ~ , ^
Logic operation	== , != , < , <= , >= , >
Shift operation	>>, <<
Assignment	=
Array declaration	array

ภาษาส้มมีตัวแปรอยู่ 3 ประเภทได้แก่ ตัวแปรส่วนกลาง (Global variable) ตัวแปรเฉพาะที่ (Local variable) และตัวแปรอาร์เรย์ (Array variable) โดยที่ตัวแปรส่วนกลางจะถูกประกาศไว้ในส่วนแรกสุดของโปรแกรมนอกการประกาศฟังก์ชัน ซึ่งตัวแปรส่วนกลางนี้จะถูกเรียกใช้ได้ในทุกฟังก์ชันในโปรแกรม ส่วนตัวแปรเฉพาะที่นั้นจะถูกเรียกใช้ได้ภายในฟังก์ชันที่ถูกประกาศไว้เท่านั้น และตัวแปรอาร์เรย์จะมีลักษณะการประกาศคล้ายตัวแปรส่วนกลางต่างกันที่การประกาศอาร์เรย์ต้องใช้ตัวประกาศอาร์เรย์ในการประกาศ

ตัวอย่างโปรแกรมภาษาส้มในการหาคำตอบของปัญหาหอคอยฮานอย

```
num = array 4          //global variable an array

// define function "mov" with 3 arguments and one
local var
to mov n from t | other =
  if n == 1
    num:from = num:from - 1
    num:t = num:t + 1

  else
    other = 6 - from - t
    mov n-1 from other
    mov 1 from t
    mov n-1 other t

// interactive mode
  disk = 3
  num:0 = 0
  num:1 = disk
  num:2 = 0
  num:3 = 0
  mov disk 1 3
```

จากโปรแกรมตัวอย่างพบว่ามีการประกาศตัวแปรอาร์เรย์ 1 ตัวคือ num โดยเป็นตัวแปรอาร์เรย์ขนาด 4 ช่อง

```
num [ 0 ] = 0
```

จะหมายถึงการอ้างให้ตัวแปร num ตัวที่ 0 มีค่าเท่ากับ 0 นั่นเอง

นอกจากนั้นในโปรแกรมตัวอย่างจะมีการประกาศฟังก์ชันที่ชื่อว่า mov โดยการประกาศฟังก์ชันจะมีการประกาศดังนี้

```
to function_name [para1, para2, ..] | [local1, local2..] =
```

โดยจะใช้คำสงวน to เป็นตัวเริ่มการประกาศฟังก์ชันแล้วตามด้วยชื่อฟังก์ชัน หลังจากนั้นจะเป็นการประกาศพารามิเตอร์ เมื่อประกาศพารามิเตอร์เสร็จแล้วจะใช้เครื่องหมาย | ในการคั่นเพื่อประกาศตัวแปรเฉพาะที่ต่อไป หลังจากการประกาศตัวแปรเฉพาะที่แล้วจะตามด้วยเครื่องหมาย **"="** แล้วจึงเป็นโปรแกรมภายในฟังก์ชันนั้นๆ

ในโปรแกรมภาษาส้มนี้จะมีการใช้ย่อหน้า เป็นการกำหนดขอบเขตของโปรแกรมและตัวแปรเฉพาะที่ภายในโปรแกรม ดังเช่นในโปรแกรมตัวอย่างนั้นที่ฟังก์ชัน mov นั้นหลังเครื่องหมาย **"="** เพื่อเป็นการบอกว่าเป็นจุดเริ่มของขอบเขตฟังก์ชัน mov นั่นเอง

ภาษาส้มรองรับโปรแกรมแบบเวียนเกิด (Recursive) ดังเห็นได้จากโปรแกรมตัวอย่างที่เป็นการหาคำตอบของปัญหาหอคอยฮานอยจะเห็นโปรแกรมตัวอย่างดังกล่าวถูกเขียนขึ้นในลักษณะของโปรแกรมเรียกซ้ำ

โดยสามารถเขียนไวย์กรณ์ของภาษาส้มออกมาได้ดังนี้

```
Notation:
* zero or more times
[..] optional
' constant symbol

Grammar:
toplevel -> 'to fundef | ex
fundef -> iden args '= ex
args ->     iden*  ['| iden* ]

ex → '{ ex* '} | ex1
ex1 →
  'if ex0 ['else ex ] |
  'while ex0 ex |
  var '= ex0 |
  ex0
ex0 → term term*
term →
  number |
  var [ '[ ex0'] ] |
  fun ex0* |
  '! ex0 |
  'array ex0 |
  '( ex0 ')

bop -> '+ | '- | '* | '/ | '& | '| | '^ | '== | '!= |
'< | '<= | '>= | '> | '% | '<< | '>>
```

ภาคผนวก ข

โปรแกรมทดสอบการทำงานหน่วยประมวลผล

ในบทนี้จะแสดงต้นฉบับโปรแกรมเพื่อทดสอบการทำงานทางด้านต่างๆ ของหน่วยประมวลผลโดยมีรูปแบบมาจากแสตนฟอร์ดทั้ง 7 โปรแกรมในรูปแบบภาษาส้ม

ข.1 โปรแกรม hanoi

```
num = array 4

to mov n from t2 | other =
  if n == 1
    num[from] = num[from] - 1
    num[t2] = num[t2] + 1
    print from space
    print t2 nl
  else
    other = 6 - from - t2
    mov n-1 from other
    mov 1 from t2
    mov n-1 other t2

to main | disk =
  disk = 3
  num[0] = 0
  num[1] = disk
  num[2] = 0
  num[3] = 0
  mov disk 1 3
```

ข.2 โปรแกรม quick

```
N = 20
a = array N

to inita | i =
  for i 0 N-1
    a[i] = N - i

to show | i =
  for i 0 N-1
    print a[i] space
  nl

to swap i j | t =
  t = a[i]
  a[i] = a[j]
  a[j] = t

to partition p r | x i j flag =
  x = a[p]
  i = p - 1
  j = r + 1
  flag = 1
  while flag
    j = j - 1
    while a[j] > x
      j = j - 1
    i = i + 1
    while a[i] < x
      i = i + 1
    if (i < j) swap i j else flag = 0
  j

to quicksort p r | q =
  if p < r
    q = partition p r
    quicksort p q
    quicksort q+1 r

to main =
  inita
  show
  quicksort 0 (N - 1)
  show
```

ข.3 โปรแกรม bubble

```
maxdata = 4
data = array maxdata

to init | i =
  for i 0 maxdata-1
    data[i] = maxdata - i

to show | i =
  for i 0 maxdata-1
    print data[i] space
  nl

to swap a b | t =
  t = data[a]
  data[a] = data[b]
  data[b] = t

to sort | i j =
  for i 0 maxdata-1
    for j 0 maxdata-2
      if data[j+1] < data[j]
        swap j j+1

to main =
  init
  show
  sort
  show
```

ข.4 โปรแกรม matmal

```
N = 4
a = array 16
b = array 16
c = array 16

// using shift and double add
// d is a * 2^n, accumulate is s
// s partial sum, d double

to mul3 a b | aa bb s c d =
  s = 0
  d = a
  if b < 0 c = 0-b else c = b
  while c > 0
    if c & 1 s = s + d
    d = d + d
    c = c >> 1
  if b < 0 s = 0-s
  s

// do (i,j) with i*N + j
to index i j = (mul3 i N) + j

to inita | i j =
  for i 0 N-1
    for j 0 N-1
       a[index i j] = i

to initb | i j =
  for i 0 N-1
    for j 0 N-1
      b[index i j] = j

to matmul | i j s k =
  for i 0 N-1
    for j 0 N-1
      s = 0
      for k 0 N-1
        s = s + (mul3 a[index i k] b[index k j])
      c[index i j] = s

to show | i j =
  for i 0 N-1
    for j 0 N-1
      print c[index i j] space
    nl

to main =
 inita initb matmul show
```

ข.5 โปรแกรม sieve

```
N = 500
a = array (N + 1)

// using shift and double add
// d is a * 2^n, accumulate is s
// s partial sum, d double

to mul3 a b | aa bb s c d =
  s = 0
  d = a
  if b < 0 c = 0-b else c = b
  while c > 0
    if c & 1 s = s + d
    d = d + d
    c = c >> 1
  if b < 0 s = 0-s
  s

to show | cnt last i =
  cnt = 0
  last = 0
  for i 2 N
    if a[i]
      print i space
      last = i
      cnt = cnt + 1
  nl print cnt space print last nl

to sieve | p j q =
  p = 2
  while (mul3 p p) <= N
    j = p + p
    while  j <= N
      a[j] = 0
      j = j + p
    p = p + 1
    while a[p] == 0
      p = p + 1

to main | i =
  a[1] = 0
  for i 2 N
    a[i] = 1
  sieve
  show
```

ข.6 โปรแกรม perm

```
N = 4
val = array 4
id = 0 - 1

to writeperm | i =
  for i 0 N-1
    print val[i] space
  nl

to visit k | t =
  id = id + 1
  val[k] = id
  if id == (N - 1)
    writeperm
  for t 0 N-1
    if val[t] == 0 visit t
  id = id - 1
  val[k] = 0

to main | i =
  for i 0 N-1
    val[i] = 0
  visit 0
```

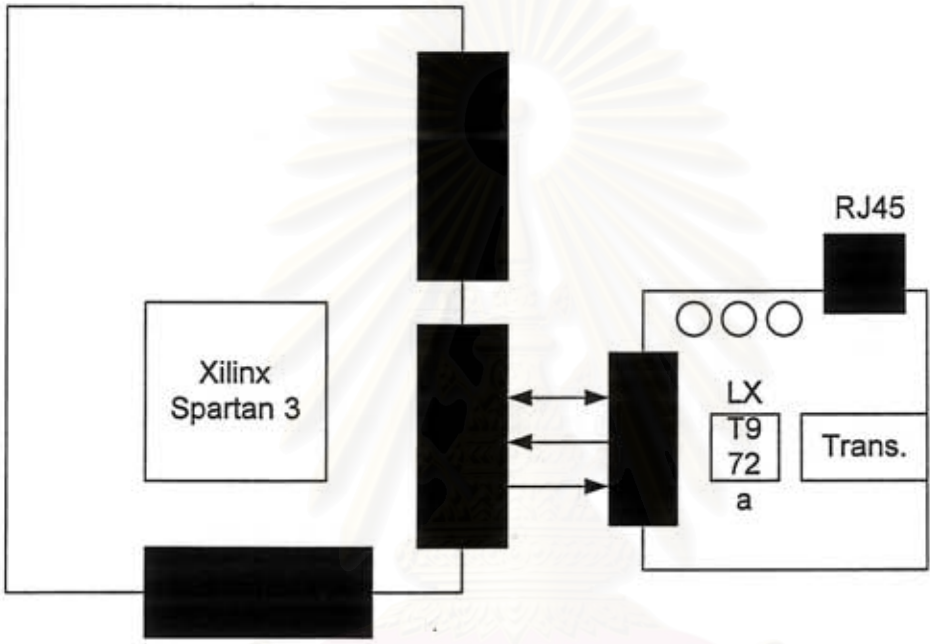
ข.7 โปรแกรม queen

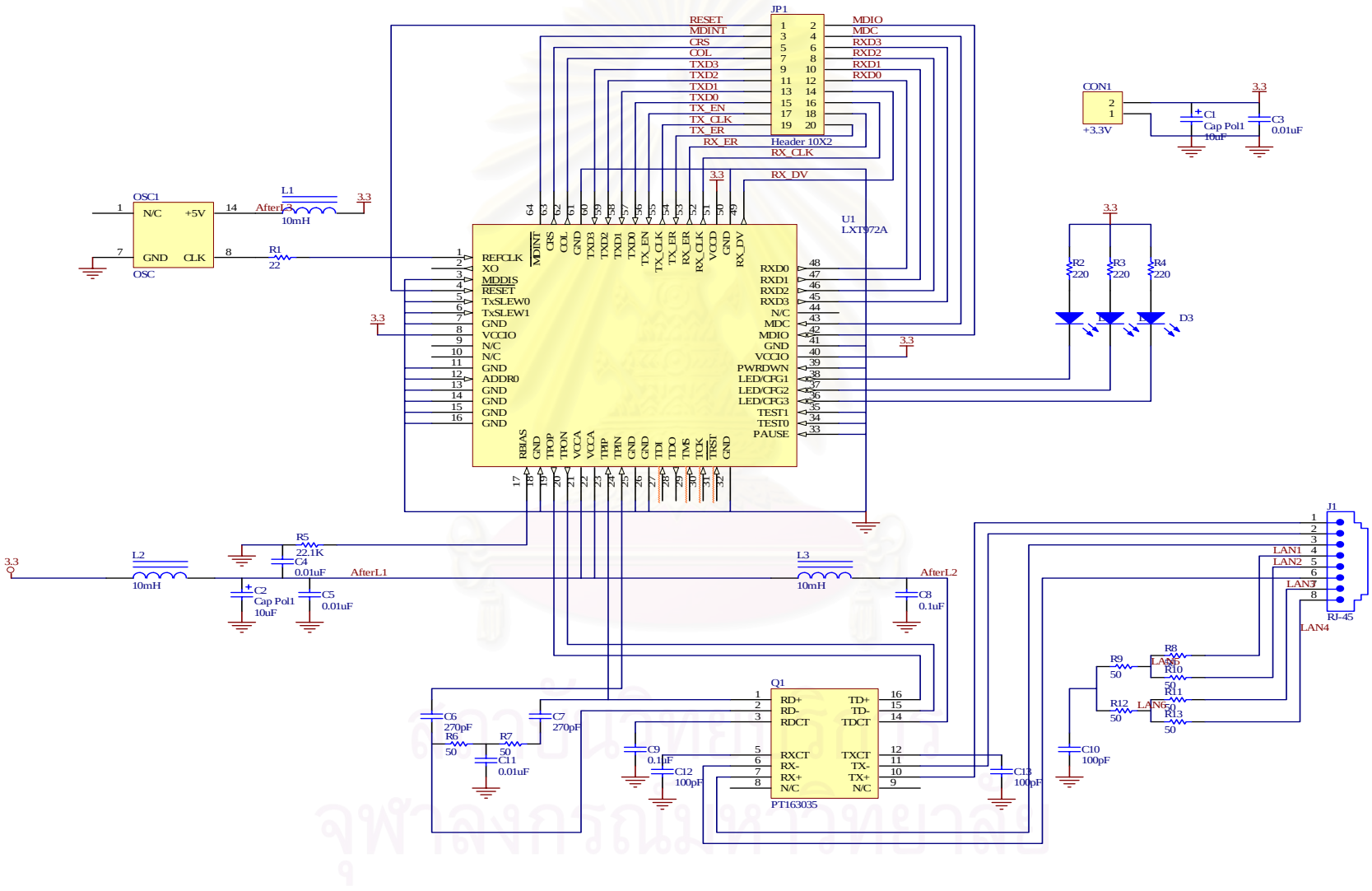
```
// Q = 8       size of Q x Q board
// Z = Q+1 = 9       empty
// D = Q+Q-1 = 15  size of diagonal

Q = 8
Z = 9
D = 15
soln = 0
col = array Q
d45 = array D
d135 = array D
queen = array Q

to find level | i =
  if level == Q
    soln = soln + 1
  else
    for i 0 Q-1
      if (col[i] >= level) &
      (d45[level+i] >= level) &
      (d135[level+Q-1-i] >= level)
          queen[level] = i
          col[i] = level
          d45[level+i] = level
          d135[level+Q-1-i] = level
          find level+1
          col[i] = Z
          d45[level+i] = Z
          d135[level+Q-1-i] = Z

to main | i =
  for i 0 Q-1
    col[i] = Z
  for i 0 D-1
    d45[i] = Z
    d135[i] = Z
  find 0
  print soln nl
```

ภาคผนวก ค

วงจรออกแบบเพื่อใช้ในการเชื่อมต่อเข้ากับบอร์ด FPGA

ในส่วนนี้ออกแบบโดยโปรแกรม Protel โดยพยามให้มีการเชื่อมต่อภายนอกให้น้อยที่สุดเพื่อลดความผิดพลาดให้โดยมีรูปการเชื่อมต่อดังนี้




JP1
Header 10X2
CON1
+3.3V
C1
Cap Pol1
10uF
C3
0.01uF
U1
LXT972A
OSC1
OSC
L1
10mH
R1
22
L2
10mH
L3
10mH
R5
22.1K
C4
0.01uF
C2
Cap Pol1
10uF
C5
0.01uF
C8
0.1uF
R2
220
R3
220
R4
220
D3
J1
RJ-45
LAN1
LAN2
LAN3
LAN4
Q1
PT163035
C6
270pF
R6
50
R7
50
C11
0.01uF
C7
270pF
C9
0.1uF
C12
100pF
C10
100pF
C13
100pF
R8
50
R9
50
R10
50
R11
50
R12
50
R13
50

ภาคผนวก ง

โปรแกรมที่ใช้ทำงานในการตอบสนอง PING ของหน่วยประมวลผล

```
.c 0
  call main
fun movstr src des len [ i ]
  lit 0
  put i
movstrloop:
  get des   ; address
  get src
  ld        ;get   value   from
src[0]
  st        ;  store  data  from
src into des[0]
  get src   ;--
  inc       ; move ptr to next
packet
  put src   ;--
  get des   ;--
  inc       ;  next  empty  for
send
  put des   ;--
  get i
  inc
  put i
  get i
  get len
  eq
  jf movstrloop
  ret
fun main [ ibuf obuf type len
mac0 mac1 mac2 ip0 ip1 myid tmp
addribuf bytesend saddr eaddr ]
;  setup  every  value  will  be
used
  lit 57344     ; E000
  put ibuf
  lit 61441     ; F001
  put obuf
  lit 80        ;00 50
  put mac0
  lit 22080     ;56 40
  put mac1
  lit 1             ;00 01
  put mac2
  lit 49320     ;192 168 -> C0A8
  put ip0
  lit 2             ;000 002 -> 0002
  put ip1
  lit 57343     ;DFFFh
  put bytesend
  lit 57342     ;DFFEh
  put addribuf
  lit 57341
  put saddr ;saddr to
  lit 57340
  put eaddr ;eaddr to
  lit 0
  put tmp   ; Clear tmp value
mainloop:
  get ibuf
  lit 7
  add
  ld        ; get ibuf[7]
  put type  ; type = ibuf[7]
CheckType:
  get type
  lit 2054      ; 0806h
  eq
  jf not_arp
 arp_rsp:
  get ibuf  ;  set  IBUF[1]to  be
Src
  inc
  get obuf  ;  set  OBUF[1]to  be
Des
  get ibuf
  ld        ; get lenght
  put len
  get len   ;  save  length  for
next using
  call movstr     ; copy all of
data
;copy SrcMAC to DesMAC
  get obuf
  lit 3
  add       ; obuf[4]
  get obuf  ; obuf[1]
  lit 3     ; len = 3
  call movstr
;--------------------  Copy  All
data form IBUF ot OBUF already
  get obuf
```

```
  lit 10      ; obuf[11]
  add
  lit 2
  st  ; Set operation to reply
;------ OBUF[11]=2 reply
  get obuf
  lit 11
  add       ; obuf[12]
  put tmp
  get tmp   ; tmp = obuf[12]
  get obuf
  lit 16
  add       ; obuf[17]
  lit 5           ; length = 5
  call movstr     ; cooy DesMAC
and DesIP
;----- Make DESIP, DESMAC
;create MyIP, MyMAC
  get tmp
  get mac0  ; obuf[12]
  st        ; obuf[12]=mac0
  get tmp
  inc
  put tmp   ; obuf[13]
  get tmp
  get mac1
  st        ; obuf[13]=mac1
  get tmp
  inc
  put tmp
  get tmp
  get mac2
  st        ; obuf[14]=mac2
  get tmp
  inc
  put tmp
  get tmp
  get ip0
  st
  get tmp
  inc
  get ip1
  st
;copy mymac
  get obuf
  lit 11
  add       ; obuf[12]
  get obuf
  lit 3
  add       ; obuf[4]
  lit 3           ; length = 3
  call movstr
 ;---------------- do CRC
  get saddr
  get obuf
  st
  get eaddr
  get obuf
  get len
  dec
  add
  st

;-------- Copy My mac to 4, 5, 6
;Send data
  get bytesend
  get len
  inc
  st
;Store IBUF and OBUF to next
packet
  get len
  get obuf
  add
  inc
  inc       ;;;;success already
  put obuf  ;           obuf[0] =
obuf[length] because obuf don't
have lenght byte so skip one
byte
  get len
  inc
  get ibuf
  add
  inc
  put ibuf  ;           ibuf[0] =
ibuf[length + 1] ; ibuf hold new
pointer already
;Set addribuf to make rxempty
  get addribuf
  get ibuf
  inc
  inc
  st
  jmp mainloop


not_arp:    ;  Continue  check
Are packet ICMP??
;  halt     ;  But  now  Don't
create it.
;Check  another  type  is  it  a
PING??
  get type
  lit 2048    ; put type 0800h
  eq
  jf not_packet
;Ping is coming
;  Copy  all  IBUF  data  to  OBUF
same
  get ibuf
  inc       ; set source to be
ibuf[1]
  get obuf  ;  set  destination
to be obuf[1]
  get ibuf
  ld        ; get length
```

```
  call movstr
;Swap MAC address
  get obuf
  lit 3
  add       ; obuf[4]
  get obuf
  lit 3
  call movstr     ; SRC -> DES
  ; insert with my mac
  get obuf
  lit 3
  add
  put tmp   ; tmp = obuf[4]
  get tmp
  get mac0
  st        ; obuf[4] = mac0
  get tmp
  inc
  put tmp   ; tmp = obuf[5]
  get tmp
  get mac1
  st        ; obuf[5] = mac1
  get tmp
  inc       ; tmp = obuf[6]
  get mac2
  st        ; obuf[6] = mac2
;Swap IP Already.
  get obuf
  lit 13
  add
  get obuf
  lit 15
  add
  lit 2     ; move 2 byte of IP
  call movstr
  get obuf
  lit 13
  add
  put tmp
  get tmp
  get ip0
  st
  get tmp
  inc
  get ip1
  st
;Swap IP Already
;Do Checksum
  get obuf
  lit 12
  add       ; obuf[13] = 0
  lit 0
  st        ;  Set  chksum  byte
to be 0
;Calculate CheckSum within this
range and store in OBUF[13]
  get obuf
  lit 12
  add
  get obuf
  lit 7
  add
  get obuf
  lit 16
  add
  call checksum
  st

;Change type field in obuf[18] to
be response
  get obuf
  lit 17     ;;;; Modify from 18
  add
  ;put tmp
  ;get tmp
  lit 0000
  st        ; Set Type and Code
to 0000, reply

;  Calculate  CheckSum  in  this
range again and store in OBUF[19]

  get obuf
  lit 18
  add
  get obuf
  lit 19
  add       ; address start
  get obuf
  lit 36
  add
  call checksum
  lit 4
  sub
  st  ; set Chksum to obuf[19]

  ;---------------- do CRC
  get saddr
  get obuf
  st
  get eaddr
  get obuf
  get ibuf
  ld
  put len
  get len
  dec
  add
  st

;Send Data
  get bytesend
  get len
  inc
  st        ; send data
```

```
;Change  OBUF  to  next  packet
same
  get obuf
  get len
  add       ;     success
  inc
  inc
  put obuf
;Change IBUF to next packet
  get len
  inc
  ;inc
  get ibuf
  add
  inc
  put ibuf

;Make rx_empty move addribuf
  get addribuf
  get ibuf
  inc
  st
;  jmp  to  main  loop  data  was
sent
  jmp mainloop

not_packet:  ;  this  routine'll
ignore packet in IBUF only
  get ibuf
  get ibuf
  ld
  inc
  add
  inc
  put ibuf
;move addribuf to make rx_empty
same
  get addribuf
  get ibuf
  inc
  inc
  st
;  jmp  to  main  loop  data  was
sent
  jmp mainloop

fun checksum start end [ sum ]
  lit 0
  put sum   ; clear  Sum  before
calculate
chksumloop:
  get start
  ld
  get sum
  add
  put sum
  get start
  get end
  eq
  jt compliment
  get start
  inc
  put start
  jmp chksumloop
compliment:
  get sum
  not
  dec
  dec
  retv


e
```

ประวัติผู้เขียนวิทยานิพนธ์

นายอลงกต บุรุษอาชาไนย เกิดเมื่อวันที่ 28 มกราคม พ.ศ. 2525 ที่โรงพยาบาลพญาไท จังหวัดกรุงเทพฯ สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2546 และเข้าศึกษาในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ปีการศึกษา 2546

ในระหว่างการศึกษาได้รับรางวัลรองชนะเลิศการแข่งขันการประกวดการออกแบบวงจรรวมแห่งชาติครั้งที่ 3 (The National IC Design Contest 2003: NIC2003) ประเภทความคิดสร้างสรรค์ ในระดับบัณฑิตศึกษา