การออกแบบและการอิมพลิเมนท์บัสระบบแบบอสมวารด้วยวิธีการเข้ารหัสหนึ่งในสี่

นางสาวกิตติมา ฐานพีรภัทร์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2553

A DESIGN AND IMPLEMENTATION OF ASYNCHRONOUS SYSTEM BUS USING 1-OF-4 DATA ENCODING

Miss Kittima Thanpeerapat

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic year 2010

หัวข้อวิทยานิพนธ์	การออกแบบและการอิมพลิเมนท์บัสระบบแบบอสมวาร ด้วยวิธีการเข้ารหัสหนึ่งในสี่
โดย	นางสาวกิตติมา ฐานพีรภัทร์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร. อาทิตย์ ทองทักษ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร. สาธิต วงศ์ประทีป)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร. อาทิตย์ ทองทักษ์)

..... กรรมการ
(รองศาสตราจารย์ ดร. ธาราทิพย์ สุวรรณศาสตร์)

..... กรรมการภายนอกมหาวิทยาลัย
(อาจารย์ ดร. ภัทร ลีลาพฤทธิ์)

กิตติมา ฐานพีรภัทร์: การออกแบบและการอิมพลิเมนท์บัสระบบแบบอสมวารด้วยวิธีการเข้ารหัสหนึ่งในสี่. (A DESIGN AND IMPLEMENTATION OF ASYNCHRONOUS SYSTEM BUS USING 1-OF-4 DATA ENCODING) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร.อาทิตย์ ทองทักษ์, 134 หน้า.

วิทยานิพนธ์ฉบับนี้เสนอการออกแบบและการอิมพลิเมนท์บัสระบบแบบอสมวารด้วยวิธีการเข้ารหัสหนึ่งในสี่ เพื่อพัฒนาประสิทธิภาพทางด้านการใช้พลังงานในการรับส่งข้อมูล และประมวลผลในงานวิจัยของวงจรอสมวาร โดยใช้รหัสหนึ่งในสี่เข้ารหัสในการรับส่งข้อมูลเพื่อลดพลังงานที่ใช้ในบัสระบบ ที่เชื่อมต่อกับไมโครโพรเซสเซอร์แบบอสมวาร หน่วยความจำแบบสมวาร และอุปกรณ์อินพุท/เอาท์พุท ซึ่งรองรับความสามารถในการเพิ่มความเร็วของบัสด้วยเทคนิคอินเตอร์รัพท์และดีเอ็มเอได้

งานวิจัยนี้ได้ออกแบบบัสระบบแบบอสมวารเข้ารหัสหนึ่งในสี่และองค์ประกอบคือไมโครโพรเซสเซอร์ และดีเอ็มเอ ซึ่งใช้งานข้อมูลบนบัสระบบร่วมกัน และทดสอบประสิทธิภาพของบัสระบบในด้านของการใช้พลังงาน ขนาดวงจร และความเร็วในการทำงาน จากผลการทดสอบประสิทธิภาพพบว่า บัสระบบสามารถทำงานร่วมกับองค์ประกอบได้อย่างถูกต้องบนเอฟพีจีเอ Xlinx SPARTAN-3E เบอร์ XC3S-500EFG320 เมื่อเปรียบเทียบบัสระบบเข้ารหัสหนึ่งในสี่กับบัสระบบเข้ารหัสรางคู่บนโครงสร้างเดียวกัน บัสระบบเข้ารหัสหนึ่งในสี่จะมีประสิทธิภาพดีกว่า นอกจากนี้จากผลการทดลองยังพบว่า วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่มีประสิทธิภาพดีในวงจรที่มีการคำนวณเป็นจำนวนคู่ (จำนวน 2n บิต) เช่น คำนวณครั้งละ 4 บิต คำนวณครั้งละ 8 บิต เป็นต้น และมีประสิทธิภาพดีที่สุดในวงจรที่มีการคำนวณครั้งละ 2 บิต อย่างไรก็ตาม วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่มีประสิทธิภาพที่ด้อยในวงจรที่มีการคำนวณครั้งละ 1 บิต

ภาควิชา .....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต.....กิตติมา ฐานพีรภัทร์.....

สาขาวิชา.....วิศวกรรมคอมพิวเตอร์.....ลายมือชื่อ อ. ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา ..... 2553.....

5070680621 : MAJOR COMPUTER ENGINEERING

KEYWORDS : ASYNCHRONOUS CIRCUIT / ASYNCHRONOUS SYSTEM BUS / 1-OF-4 CODE

KITTIMA THANPEERAPAT : A DESIGN AND IMPLEMENTATION OF ASYNCHRONOUS SYSTEM BUS USING 1-OF-4 DATA ENCODING. ADVISOR : ASST. PROF. ARTHIT THONGTAK, Ph.D., 134 pp.

This thesis proposes a design and implementation of asynchronous system bus using 1-of-4 data encoding for improving power performance of data transfer and processing fields in asynchronous circuit research. The 1-of-4 data encoding used for reducing power consumption in system bus, which is subsystem that connect to asynchronous microprocessor, synchronous memory and I/O device together and also support the interrupt and DMA technique.

This research presents a design of asynchronous system bus using 1-of-4 data encoding and components that are microprocessor and DMA, which share data on the system bus. And then test performance in terms of power, area and time. The performance test report is shown that the system bus and components operate correctly on Xilinx SPARTAN-3E XC3S-500EFG320 FPGA. When compares the 1-of-4 system bus to the dual-rail system bus in the same architecture, the 1-of-4 system bus appears to have the higher performance. Moreover, The test report is shown that the 1-of-4 function circuits can have the good performance in even-bits (2n-bits) computing per time such as compute 4-bits per time, compute 8-bits per time, and have the best performance in the circuit which is compute 2-bits per time. However, the 1-of-4 function circuits have the poor performance in the circuit which is compute 1-bit per time.

Department: ....Computer Engineering... Student's Signature: .....

Field of Study: ..Computer Engineering... Advisor' Signature: .....

Academic Year: .....2010.....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดีด้วยความกรุณาอย่างยิ่งของผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์ ผู้เป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้ให้คำแนะนำและเสนอแนะข้อคิดเห็นต่างๆในการทำวิจัยด้วยดีมาตลอด

ขอขอบพระคุณ รองศาสตราจารย์ ดร.สาธิต วงศ์ประทีป รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ อาจารย์ ดร.ภัทร ลีลาพฤทธิ์ และคณาจารย์ที่กรุณาให้คำปรึกษาเพื่อขัดเกลางานวิจัยให้มีความสมบูรณ์

ขอขอบคุณสมาชิกแลป DSEL ทุกคน ที่ให้คำปรึกษาและแนะนำสิ่งต่างๆที่เกี่ยวข้องกับงานวิจัยทั้งทางตรงและทางอ้อม

ท้ายที่สุดนี้ ผู้วิจัยขอกราบขอบพระคุณบิดา มารดา ที่สนับสนุน ห่วงใย และให้กำลังใจแก่ผู้วิจัยเสมอมา

สารบัญ

บทที่ หน้า

บทที่ หน้า

บทที่ หน้า

สารบัญตาราง

สารบัญภาพ

ภาพที่ หน้า

บทที่ 1

บทนำ

บทนำของงานวิจัยเรื่องการออกแบบและการอิมพลิเมนท์บัสระบบแบบอสมวารด้วยวิธีการเข้ารหัสหนึ่งในสี่ ประกอบด้วย ความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ ขอบเขตการดำเนินงาน ประโยชน์ที่คาดว่าจะได้รับ ขั้นตอนการดำเนินงาน ลำดับขั้นตอนในการเสนอการวิจัย และผลงานที่ตีพิมพ์จากงานวิจัย โดยมีรายละเอียดดังนี้

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันงานทางด้านดิจิทัลถูกพัฒนาเพิ่มขีดความสามารถมากขึ้น เพื่อตอบรับความต้องการของเทคโนโลยีตามยุคสมัย ซึ่งมีความซับซ้อนของงานมากขึ้นและต้องการความรวดเร็วในการทำงานสูงขึ้นตามไปด้วย การออกแบบวงจรดิจิทัลแบบอสมวาร (Asynchronous Digital Circuits) [1] สามารถรองรับข้อจำกัดในการออกแบบวงจรดังกล่าวได้ดีกว่าการออกแบบด้วยวงจรสมวารที่มีข้อจำกัดในการออกแบบสูงกว่า เนื่องจากวงจรสมวารเป็นวงจรที่ใช้การกำกับจังหวะของสัญญาณนาฬิกาเป็นตัวควบคุมจังหวะในการรับส่งข้อมูลในระบบ ให้มีการทำงานที่พร้อมเพรียงกัน ซึ่งหากวงจรมีความซับซ้อนและมีขนาดใหญ่ ระยะทางในวงจรที่เพิ่มมากขึ้นจะส่งผลให้เกิดความคลาดเคลื่อนของสัญญาณนาฬิกา (Clock Skew) กล่าวคือส่วน (Path) ที่อยู่ไกลกว่าจะได้รับสัญญาณจากสัญญาณนาฬิกาได้ช้ากว่าส่วนที่อยู่ใกล้ทำให้เกิดความคลาดเคลื่อนของจังหวะการทำงาน ส่งผลให้ทำงานผิดพลาดได้ แต่ในวงจรอสมวารไม่ต้องอาศัยสัญญาณนาฬิกาเป็นตัวควบคุมจังหวะของการรับส่งข้อมูลในระบบ แต่ใช้สัญญาณเฉพาะติดต่อภายในแต่ละวงจรแทน ข้อจำกัดในเรื่องขนาดของวงจรจึงไม่มีผลต่อการทำงานของวงจรอสมวาร สำหรับประสิทธิภาพทางด้านความเร็ว วงจรอสมวารมีมากกว่าวงจรสมวาร เนื่องจากวงจรสมวารถูกกำหนดให้ความเร็วในการทำงานของวงจรขึ้นอยู่กับสัญญาณนาฬิกาส่วนที่ทำงานช้าที่สุดเพื่อให้วงจรทำงานเข้าจังหวะกันได้อย่างถูกต้อง แต่วงจรอสมวารไม่ใช้สัญญาณนาฬิกา ความเร็วในการทำงานที่ได้จึงเท่ากับเวลาการทำงานของแต่ละขั้นตอนทำให้ทำงานได้รวดเร็วกว่า

นอกจากนี้ วงจรอสมวารยังมีข้อดีในด้านประหยัดพลังงาน โดยวงจรอสมวารใช้พลังงานในการทำงานน้อยกว่าวงจรสมวาร เนื่องจากวงจรสมวารนั้นใช้สัญญาณนาฬิกาในการ

ควบคุมการสลับไปมาของการรับส่งของข้อมูล ซึ่งจะเกิดการคายประจุและเก็บประจุสัญญาณตลอดทั้งวงจรตลอดเวลา แต่ในวงจรอสมวาร การเปลี่ยนแปลงสัญญาณรับส่งข้อมูลในส่วนต่างๆ นั้นไม่จำเป็นต้องเปลี่ยนแปลงตลอดทั้งวงจรตลอดเวลา จะมีการเปลี่ยนแปลงเฉพาะในส่วนที่เกี่ยวข้องกับการทำงานในแต่ละงานเท่านั้น ทำให้ใช้พลังงานน้อยกว่า การเปลี่ยนแปลงเฉพาะส่วนนี้ยังง่ายต่อการตรวจสอบหากเกิดข้อผิดพลาดในการทำงานขึ้น โดยสามารถตรวจสอบความผิดพลาดได้จากระบบตอบรับ (Acknowledgement Circuits) ของวงจร ทำให้ทราบได้ว่าเกิดความผิดพลาดขึ้นที่จุดใดของวงจรจากการทำงานของส่วนตอบรับได้ ช่วยประหยัดเวลาในการทดลองโดยไม่จำเป็นต้องตรวจหาความผิดพลาดจากวงจรทั้งหมด

การรับส่งข้อมูลระหว่างวงจรที่มีการทำงานซับซ้อนนั้น เส้นทางรับส่งข้อมูลย่อมมีความซับซ้อนเนื่องจากแต่ละวงจรมีความแตกต่างกันเพื่อรองรับการทำงานที่หลากหลาย การออกแบบบัส (Bus) ที่ใช้เป็นเส้นทางรับส่งข้อมูลระหว่างวงจรให้มีประสิทธิภาพจึงทำได้ยาก หากในระบบมีทั้งวงจรสมวารและอสมวารรวมกันจะทำให้การออกแบบยุ่งยากมากยิ่งขึ้นไปอีก เกิดปัญหาการใช้สายสัญญาณจำนวนมาก การใช้พลังงานอย่างสิ้นเปลือง และเกิดความยุ่งยากในการปรับปรุงเพื่อพัฒนาต่อไป

การออกแบบวงจรอสมวารไม่มีการใช้สัญญาณนาฬิกาควบคุมจังหวะในการรับส่งข้อมูล จึงใช้วิธีการเข้ารหัสข้อมูลเพื่อตรวจสอบการมาถึงของข้อมูล และสร้างจังหวะในการทำงานที่ถูกต้องให้กับวงจรอสมวาร อีกทั้งการเข้ารหัสข้อมูลยังช่วยให้สามารถตรวจสอบความผิดพลาดของข้อมูลที่ผ่านการเข้ารหัสแล้วได้ง่ายขึ้น การเข้ารหัสที่นิยม เช่น การเข้ารหัส[2]โดยใช้ข้อมูลรวมชุด (Bundle Data) การเข้ารหัสโดยใช้รหัสรางคู่ (Dual-rail Code) และการเข้ารหัสโดยใช้รหัสหนึ่งในสี่ (1-of-4 Code) ซึ่งเป็นรูปแบบการเข้ารหัสแบบ 1-of-N Code [3] ที่ใช้พลังงานในการเปลี่ยนสถานะของสัญญาณ (State Transition) น้อยกว่าการเข้ารหัสข้อมูลแบบอื่น

งานวิจัยนี้จึงมีวัตถุประสงค์เพื่อพัฒนาขอบเขตการวิจัยเกี่ยวกับวงจรอสมวาร (Asynchronous Circuits) ทางด้านการใช้พลังงานให้มีประสิทธิภาพในการรับส่งข้อมูล และประมวลผลวงจรแบบอสมวาร โดยศึกษาการออกแบบบัสระบบแบบอสมวาร (Asynchronous System Bus) โดยใช้รหัสหนึ่งในสี่เข้ารหัสในการรับส่งข้อมูลเพื่อเพิ่มประสิทธิภาพทางด้านการใช้พลังงานให้กับบัสระบบ [4] ที่เชื่อมต่อกับไมโครโพรเซสเซอร์แบบอสมวาร (Asynchronous

Processor) หน่วยความจำ และอุปกรณ์อินพุท/เอาท์พุท ซึ่งรองรับความสามารถในการเพิ่มความเร็วของบัสด้วยเทคนิคอินเตอร์รัพท์และดีเอ็มเอได้

1.2 วัตถุประสงค์

งานวิจัยนี้มีวัตถุประสงค์เพื่อศึกษาการออกแบบบัสระบบแบบอสมวารโดยใช้เทคนิคการเข้ารหัสหนึ่งในสี่ช่วยเพิ่มประสิทธิภาพทางด้านการใช้พลังงานให้กับบัสระบบ

1.3 ขอบเขตการดำเนินงาน

1. ออกแบบบัสระบบแบบอสมวารที่มีคุณสมบัติดังต่อไปนี้
   1.1 สายสัญญาณและเลขที่อยู่ขนาด 8 บิตเป็นอย่างน้อย
   1.2 เข้ารหัสโดยใช้รหัสหนึ่งในสี่
   1.3 สามารถเชื่อมต่อกับไมโครโพรเซสเซอร์แบบอสมวาร อุปกรณ์ต่อพ่วง และรองรับการใช้งานอินเตอร์รัพท์และอีเอ็มเอได้
2. สังเคราะห์วงจรบัสระบบผ่านโปรแกรม Xilinx ISE เป็นอย่างน้อย และจำลองการทำงานผ่านโปรแกรม ModelSim XE เป็นอย่างน้อย โดยกำหนดค่าความหน่วงเวลาด้วยตนเอง
3. อิมพลิเมนท์วงจรบัสระบบด้วยการโปรแกรมลงเอฟพีจีเอเป็นอย่างน้อย
4. ทดสอบประสิทธิภาพการทำงานของบัสระบบ ด้วยการวัดการใช้พลังงานของบัสเข้ารหัสรางคู่กับบัสเข้ารหัสหนึ่งในสี่ในระดับการเปลี่ยนสถานะของสัญญาณ และประมาณค่าพลังงานที่ถูกใช้ในการทำงานของบัสระบบ

1.4 ประโยชน์ที่คาดว่าจะได้รับ

ได้บัสระบบแบบอสมวารโดยใช้เทคนิคการเข้ารหัสหนึ่งในสี่ช่วยเพิ่มประสิทธิภาพทางด้านการใช้พลังงานให้กับบัสระบบ

1.5 ขั้นตอนการดำเนินงาน

1. ศึกษาการออกแบบวงจรอสมวาร
2. ศึกษาการออกแบบบัสระบบสำหรับคอมพิวเตอร์ ชนิดของบัส การทำงานของบัส การแลกเปลี่ยนข้อมูล
3. ศึกษาหลักการทำงานของดีเอ็มเอ และอินเตอร์รัพท์
4. ศึกษาวิธีการเชื่อมต่อระหว่างวงจรสมวาร และอสมวาร
5. ศึกษาการออกแบบบัสระบบแบบอสมวาร
6. ออกแบบบัสระบบแบบอสมวารที่ใช้การเข้ารหัสหนึ่งในสี่
7. เชื่อมต่อบัสกับองค์ประกอบวงจรเข้าด้วยกัน
8. สังเคราะห์วงจรที่ได้ออกแบบไว้ จำลองการทำงาน และอิมพลิเมนท์วงจรลงเอฟพีจีเอ
9. ทดสอบประสิทธิภาพของบัสระบบ ด้วยการวัดการใช้พลังงานของบัส
10. สรุปผลและเรียบเรียงวิทยานิพนธ์

1.6 ลำดับขั้นตอนในการเสนอการวิจัย

วิทยานิพนธ์นี้แบ่งเนื้อหาออกเป็น 7 บทดังนี้ บทที่ 1 เป็นบทนำซึ่งกล่าวถึงที่มาและความสำคัญของปัญหา รวมทั้งวัตถุประสงค์ของงานวิจัย บทที่ 2 สรุปทฤษฎีที่ใช้และงานวิจัยที่เกี่ยวข้อง บทที่ 3 เสนอวิธีการออกแบบบัสระบบแบบอสมวาร ซึ่งเข้ารหัสหนึ่งในสี่เพื่อลดการใช้พลังงานของบัสระบบ บทที่ 4 อธิบายการออกแบบดีเอ็มเอซึ่งเป็นองค์ประกอบที่ใช้ร่วมกับบัสระบบ บทที่ 5 อธิบายการออกแบบและสร้างสถาปัตยกรรมรวมถึงชุดคำสั่งของไมโครโปรเซสเซอร์เข้ารหัสหนึ่งในสี่ ซึ่งเป็นองค์ประกอบที่ใช้ร่วมกับบัสระบบเช่นกัน บทที่ 6 เสนอผลการทดลอง รวมทั้งประสิทธิภาพของบัสระบบที่ออกแบบ โดยเปรียบเทียบกับบัสระบบแบบอสมวารเข้ารหัสรางคู่ที่มีสถาปัตยกรรมเดียวกัน และบทที่ 7 เป็นบทสรุปการวิจัยและข้อเสนอแนะ

1.7 ผลงานที่ตีพิมพ์จากงานวิจัย

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ตีพิมพ์เป็นบทความทางวิชาการ ในหัวข้อ “Reducing Power Consumption in Asynchronous System Bus” โดย กิตติมา ฐานพีรภัทร์ และอาทิตย์ ทองทักษ์ ในงานประชุมวิชาการ “The 25th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC2010)” ณ โรงแรมแอม-บาสเดอร์ซิตี้จอมเทียน เมืองพัทยา จังหวัดชลบุรี ในระหว่างวันที่ 4-7 กรกฎาคม 25533

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

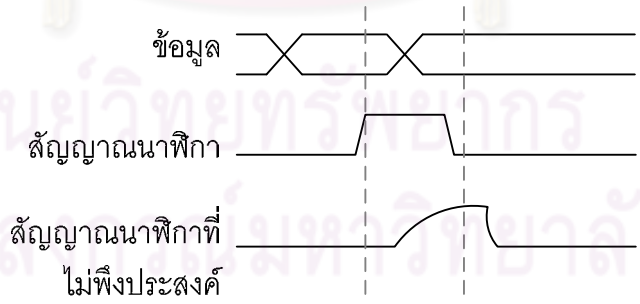
ในบทนี้กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง ซึ่งเป็นความรู้เบื้องต้นในการทำงานวิจัย แบ่งออกเป็น วงจรอสมวาร การเข้ารหัสในวงจรอสมวาร การออกแบบวงจรอสมวาร บัส ดีเอ็มเอ การใช้พลังงานของวงจร และงานวิจัยที่เกี่ยวข้อง โดยมีรายละเอียดดังต่อไปนี้

2.1 วงจรอสมวาร

วงจรสมวารมีลักษณะเด่นคือ การเปลี่ยนสถานะของวงจรต่างๆจะอาศัยสัญญาณนาฬิกาเป็นตัวกำกับจังหวะหรือกระตุ้นให้เกิดการเปลี่ยนแปลง แต่ในวงจรอสมวารนั้นลักษณะการทำงานจะแตกต่างออกไป กล่าวคือไม่ต้องอาศัยสัญญาณนาฬิกาในการกำกับจังหวะ ซึ่งจะทำให้เกิดข้อดี [5] ขึ้นดังต่อไปนี้

1. ปราศจากปัญหาสัญญาณนาฬิกาที่ผิดเพี้ยนไป

สัญญาณนาฬิกาที่ผิดเพี้ยน เนื่องจากการที่สัญญาณนาฬิกาเดินทางไปถึงจุดต่างๆในเวลาที่แตกต่างกันซึ่งจะเป็นไปตามหลักการของการมีสัญญาณนาฬิการ่วมกันของวงจรแบบสมวาร เกิดปัญหาความคลาดเคลื่อนของสัญญาณนาฬิกาที่ไม่พึงประสงค์ ตามรูปที่ 2.1



รูปที่ 2.1 สัญญาณนาฬิกาและสัญญาณนาฬิกาที่ไม่พึงประสงค์

จากรูปที่ 2.1 เมื่อเกิดสัญญาณนาฬิกาที่ไม่พึงประสงค์ขึ้น จะส่งผลกระทบให้จังหวะการทำงานในวงจรเกิดความคลาดเคลื่อน และเกิดความผิดพลาดในการทำงานตามมา

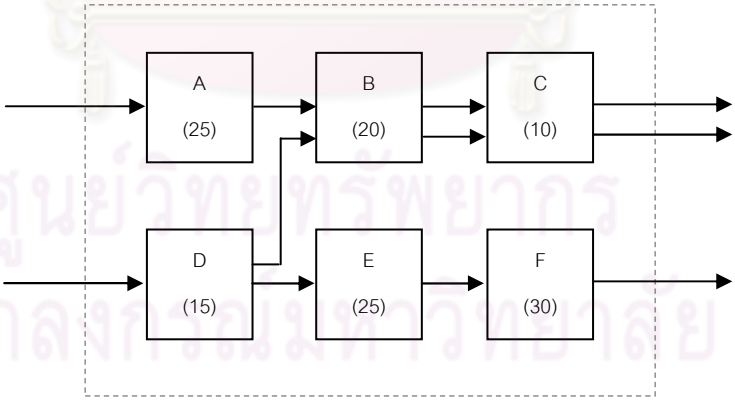
แต่ในวงจรแบบอสมวารจะไม่มีการจ่ายสัญญาณนาฬิกาเพื่อใช้เป็นตัวกำหนดการทำงาน จึงไม่ต้องกังวลเรื่องเวลาที่ผิดเพี้ยนไป

2. ใช้พลังงานต่ำ

วงจรมาตรฐานแบบสมวารใช้สัญญาณนาฬิกาที่มีการเปลี่ยนแปลงสถานะสลับไปมา ซึ่งจะต้องประจุสัญญาณและคายประจุสัญญาณอยู่ตลอดเวลาตลอดทั้งวงจร แม้ในส่วนของวงจรที่ไม่ได้ถูกเรียกใช้งาน ก็ยังคงได้รับสัญญาณนาฬิกากระตุ้นอย่างต่อเนื่อง ทำให้สิ้นเปลืองพลังงาน แต่ในวงจรแบบอสมวารจะมีการเปลี่ยนแปลงสัญญาณภายใน เมื่อวงจรส่วนนั้นถูกร้องขอให้ใช้งานเท่านั้น จึงใช้พลังงานต่ำกว่า

3. ประสิทธิภาพทางด้านความเร็วดีกว่าวงจรสมวาร

วงจรสมวารถูกกำหนดให้ความเร็วในการทำงานของวงจรขึ้นอยู่กับสัญญาณนาฬิกาส่วนที่ทำงานช้าที่สุดหรือวิถีวิกฤต (Critical Path) เพื่อให้วงจรทำงานเข้าจังหวะกันได้อย่างถูกต้อง แต่วงจรอสมวารไม่ใช้สัญญาณนาฬิกา ความเร็วในการทำงานที่ได้จึงเท่ากับเวลาการทำงานของแต่ละขั้นตอนทำให้ทำงานได้รวดเร็วกว่า ยกตัวอย่างการทำงาน ดังรูปที่ 2.2



รูปที่ 2.2 การใช้เวลาของงาน A B C D E และ F

จากรูปที่ 2.2 แสดงการทำงานที่ประกอบด้วย 6 ขั้นตอน คือขั้นตอน A B C D E และ F ซึ่งในวงจรสมวาร เวลาที่ใช้ในการทำงานแต่ละขั้นตอนจะเท่ากับขั้นตอนที่ใช้เวลาในการทำงานมากที่สุด โดยมีรายละเอียดดังนี้

เวลาในการทำของงานที่ 1 ซึ่งมีขั้นตอนจาก A → B → C = 30+30+30 = 90

เวลาในการทำของงานที่ 2 ซึ่งมีขั้นตอนจาก D → B → C = 30+30+30 = 90

เวลาในการทำของงานที่ 3 ซึ่งมีขั้นตอนจาก D → E → F = 30+30+30 = 90

รวมเวลาเฉลี่ยที่ใช้ในการทำงานทั้งหมด = (90+90+90) / 3 = 90

แต่ในวงจรอสมวาร เวลาที่ใช้ในการทำงานทั้งหมดจะเท่ากับเวลาที่ใช้ในการทำงานของแต่ละขั้นตอน นำมาหาค่าเฉลี่ย โดยมีรายละเอียดดังนี้

เวลาในการทำของงานที่ 1 ซึ่งมีขั้นตอนจาก A → B → C = 25+20+10 = 55

เวลาในการทำของงานที่ 2 ซึ่งมีขั้นตอนจาก D → B → C = 15+20+10 = 45

เวลาในการทำของงานที่ 3 ซึ่งมีขั้นตอนจาก D → E → F = 15+25+30 = 70

รวมเวลาเฉลี่ยที่ใช้ในการทำงานทั้งหมด = (55+45+70) / 3 = 56.67

ซึ่งจะเห็นได้ว่าเวลาที่ใช้ในวงจรอสมวารนั้น เวลาในการทำงานทั้งหมดของวงจรจะใช้เวลาน้อยกว่าวงจรสมวาร

2.2 การเข้ารหัสในวงจรอสมวาร

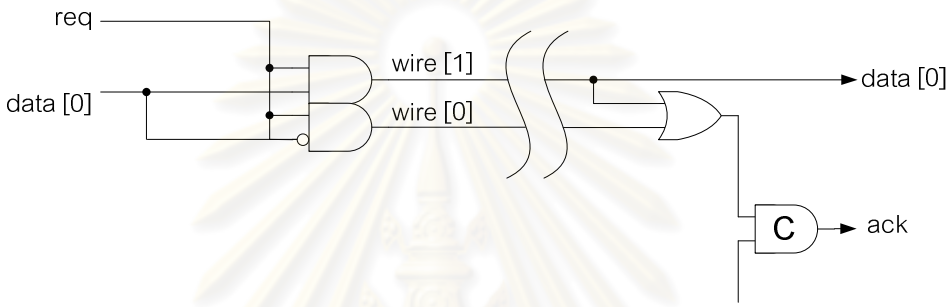
การออกแบบวงจรอสมวารไม่มีการใช้สัญญาณนาฬิกาควบคุมจังหวะในการรับส่งข้อมูล จึงใช้วิธีการเข้ารหัสข้อมูลเพื่อตรวจสอบการมาถึงของข้อมูล และสร้างจังหวะในการทำงานที่ถูกต้องให้กับวงจรอสมวาร นอกจากนี้การเข้ารหัสข้อมูลยังช่วยให้สามารถตรวจสอบความผิดพลาดของข้อมูลที่ผ่านการเข้ารหัสแล้วได้ง่ายขึ้น การเข้ารหัสที่นิยม เช่น การเข้ารหัสโดยใช้ข้อมูลรวมชุด การเข้ารหัสโดยใช้รหัสรางคู่ และการเข้ารหัสโดยใช้รหัสหนึ่งในสี่ โดยจะอธิบายเฉพาะการเข้ารหัสที่เกี่ยวข้องกับงานวิจัยชิ้นนี้ ดังนี้

2.2.1 รหัสรางคู่

การเข้ารหัสโดยใช้รหัสรางคู่จะดำเนินการเข้ารหัสข้อมูลทีละ 1 บิตดังตารางที่ 2.1 โดยรับส่งสัญญาณข้อมูล 1 บิตด้วยสายสายสัญญาณ 2 เส้น และตรวจสอบการมาถึงของสัญญาณข้อมูลด้วยสายสัญญาณตอบรับ (Acknowledge Signal) 1 เส้น ส่วนสัญญาณร้องขอ (Request Signal) จะถูกส่งรวมไปกับสัญญาณข้อมูล ดังรูปที่ 2.3

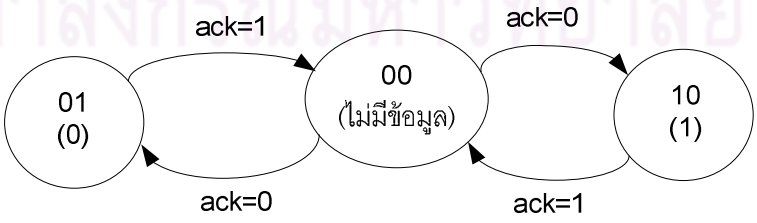
ตารางที่ 2.1 ค่าความจริงในสายสัญญาณรหัสรางคู่ที่ใช้เข้ารหัสข้อมูล 1 บิต

รหัสฐานสอง	รหัสรางคู่	
data[0]	wire[1]	wire[0]
0	0	1
1	1	0
ไม่มีข้อมูล	0	0



รูปที่ 2.3 วงจรแปลงค่าระหว่างรหัสฐานสองกับรหัสรางคู่ [6]

การใช้รหัสรางคู่กับสัญญาณอาณัติแบบ 4 ขั้น (4-Cycle Protocol, 4 Phase Protocol) [2] ซึ่งเป็นโพรโทคอลการเปลี่ยนสัญญาณแทนสัญญาณนาฬิกาในวงจรอสมวารแบบหนึ่ง มีหลักการทำงานคือ เมื่อภาคส่งทำการส่งข้อมูลไปยังภาครับเสร็จสิ้น สัญญาณตอบรับของภาครับจะมีค่าเป็น 1 เพื่อแสดงว่าได้รับข้อมูลครบแล้ว จากนั้นภาคส่งจะเปลี่ยนค่าในสายข้อมูลและค่าในสายสัญญาณตอบรับทั้งหมดของภาคส่งเป็น 0 (ไม่มีข้อมูล) เพื่อพร้อมสำหรับการรับข้อมูลใหม่ในครั้งถัดไป การทำงานรับส่งข้อมูล 1 บิตของรหัสรางคู่กับสัญญาณอาณัติแบบ 4 ขั้น เป็นดังรูปที่ 2.4



รูปที่ 2.4 การทำงานรับส่งข้อมูล 1 บิตของรหัสรางคู่กับสัญญาณอาณัติแบบ 4 ขั้น

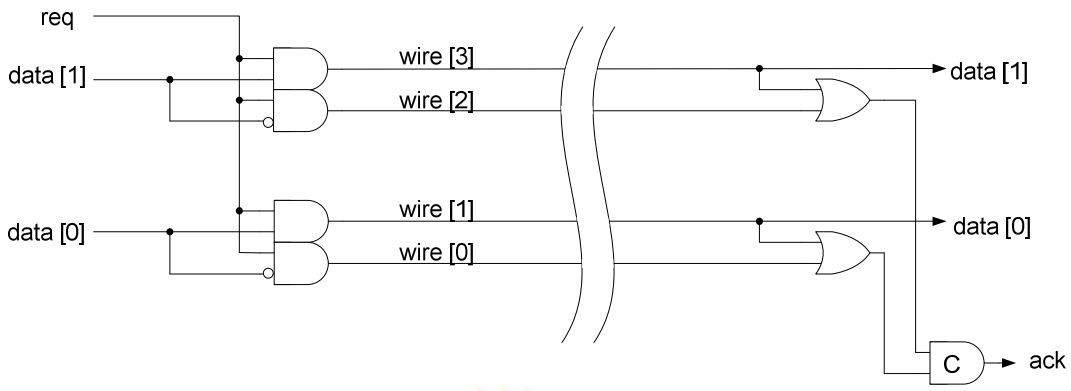
2.2.2 รหัสหนึ่งในสี่

การเข้ารหัสโดยใช้รหัสหนึ่งในสี่ เป็นรูปแบบการเข้ารหัสแบบ 1-of-N code ที่ใช้พลังงานในการเปลี่ยนสถานะของสัญญาณน้อยกว่าการเข้ารหัสข้อมูลแบบอื่น รหัสหนึ่งในสี่จะดำเนินการเข้ารหัสข้อมูลทีละ 2 บิต ดังตารางที่ 2.2 โดยรับส่งสัญญาณข้อมูล 2 บิตด้วยสายสายสัญญาณ 4 เส้น กล่าวคือ ข้อมูลรหัสฐานสองขนาด 2 บิต จะถูกเข้ารหัสกลายเป็นข้อมูลเข้ารหัสหนึ่งในสี่ขนาด 4 บิต (2 Binary Bits = 4 1-of-4 Bits) และตรวจสอบการมาถึงของสัญญาณข้อมูลด้วยสายสัญญาณตอบรับ 1 เส้นดังรูปที่ 2.5 (ข) ในรหัสรางคู่จะรับส่งสัญญาณข้อมูล 2 บิตด้วยสายสายสัญญาณ 4 เส้นเช่นกัน แต่ใช้สายสัญญาณตอบรับ 2 เส้น เพื่อตรวจสอบการมาถึงของสัญญาณข้อมูล 2 บิตดังรูปที่ 2.5 (ก) ดังนั้นรหัสหนึ่งในสี่จึงลดการใช้สายสัญญาณตอบรับลงได้ครึ่งหนึ่งเมื่อเทียบกับรหัสรางคู่ ส่วนสัญญาณร้องขอจะถูกส่งรวมไปกับสัญญาณข้อมูล

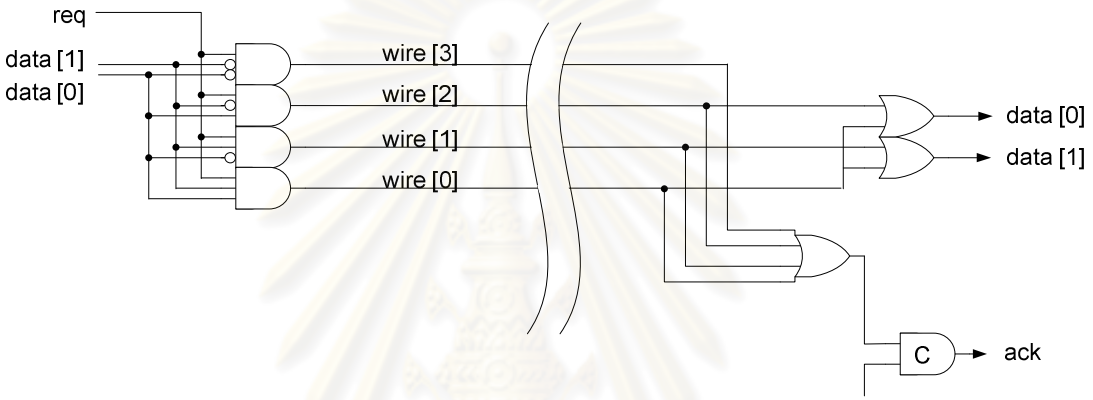
นอกจากนี้ รหัสหนึ่งในสี่มีข้อดีในเรื่องลดการใช้พลังงาน [7] จากตารางที่ 2.2 สายสัญญาณหนึ่งในสี่มีการเปลี่ยนแปลงค่าในสายสัญญาณเพื่อส่งข้อมูล 2 บิตแค่เพียงบิตเดียว ในขณะที่สายสัญญาณรหัสรางคู่เปลี่ยนแปลงค่าในสายสัญญาณ 2 บิต ดังนั้นรหัสหนึ่งในสี่กับสัญญาณอาณัติแบบ 4 ขั้น จึงลดพลังงานที่ใช้ในการเปลี่ยนสถานะของสัญญาณลงได้ครึ่งหนึ่งเมื่อเทียบกับ รหัสรางคู่กับสัญญาณอาณัติแบบ 4 ขั้น

ตารางที่ 2.2 ค่าความจริงในสายสัญญาณรหัสหนึ่งในสี่และรหัสรางคู่ที่ใช้เข้ารหัสข้อมูล 2 บิต

รหัสฐานสอง		รหัสหนึ่งในสี่				รหัสรางคู่			
data[1]	data[0]	wire[3]	wire[2]	wire[1]	wire[0]	wire[3]	wire[2]	wire[1]	wire[0]
0	0	1	0	0	0	0	1	0	1
0	1	0	1	0	0	0	1	1	0
1	0	0	0	1	0	1	0	0	1
1	1	0	0	0	1	1	0	1	0
ไม่มีข้อมูล		0	0	0	0	0	0	0	0



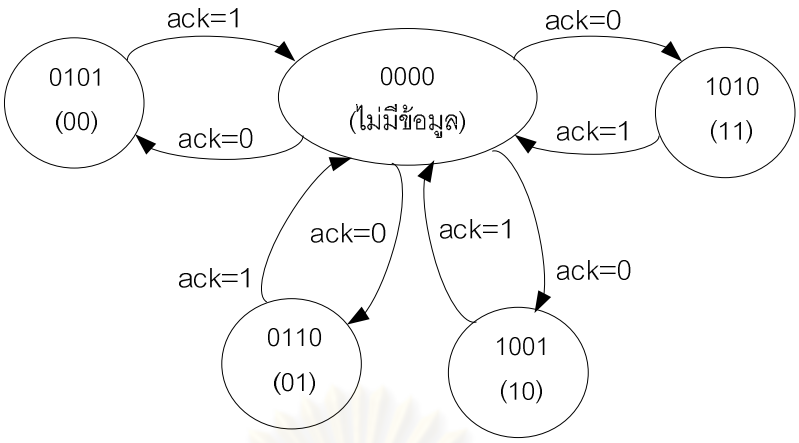
(ก) วงจรแปลงค่าระหว่างรหัสฐานสอง 2 บิตกับรหัสรางคู่



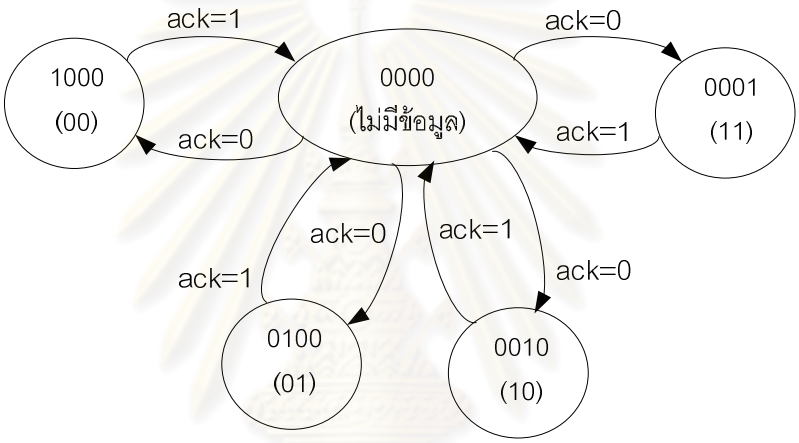
(ข) วงจรแปลงค่าระหว่างรหัสฐานสอง 2 บิตกับรหัสหนึ่งในสี่

รูปที่ 2.5 วงจรแปลงค่าระหว่างรหัสฐานสองกับรหัสรางคู่และรหัสหนึ่งในสี่ [6]

การใช้รหัสหนึ่งในสี่กับสัญญาณอาณัติแบบ 4 ขั้น มีหลักการทำงานเหมือนกับการใช้รหัสรางคู่กับสัญญาณอาณัติแบบ 4 ขั้น กล่าวคือเมื่อภาคส่งทำการส่งข้อมูลไปยังภาครับเสร็จสิ้น สัญญาณตอบรับของภาครับจะมีค่าเป็น 1 เพื่อแสดงว่าได้รับข้อมูลครบแล้ว จากนั้นภาคส่งจะเปลี่ยนค่าในสายข้อมูลและค่าในสายสัญญาณตอบรับทั้งหมดของภาคส่งเป็น 0 (ไม่มีข้อมูล) เพื่อพร้อมสำหรับการรับข้อมูลใหม่ในครั้งถัดไป การทำงานรับส่งข้อมูล 2 บิตของรหัสรางคู่และรหัสหนึ่งในสี่กับสัญญาณอาณัติแบบ 4 ขั้น เป็นดังรูปที่ 2.6



(ก) การทำงานรับส่งข้อมูล 2 บิตของรหัสรางคู่กับสัญญาณอาณัติแบบ 4 ขั้น



(ข) การทำงานรับส่งข้อมูล 2 บิตของรหัสหนึ่งในสี่กับสัญญาณอาณัติแบบ 4 ขั้น

รูปที่ 2.6 การทำงานรับส่งข้อมูล 2 บิตของรหัสรางคู่และรหัสหนึ่งในสี่
กับสัญญาณอาณัติแบบ 4 ขั้น

2.3 การออกแบบวงจรอสมวาร

การออกแบบวงจรอสมวาร ต้องคำนึงถึงแบบจำลองความหน่วง (Delay Model) เพื่อจัดการกับความหน่วงที่เกิดขึ้นกับวงจร เช่น ความหน่วงที่เกิดจากขั้นตอนการเจือสาร ความหน่วงจากอุณหภูมิ ความหน่วงจากแรงดันที่ป้อนให้กับวงจร เป็นต้น เพื่อให้วงจรทำงานได้อย่างถูกต้อง งานวิจัยนี้ใช้วิธีการออกแบบวงจรอสมวารด้วยแบบจำลองความหน่วงดังต่อไปนี้

2.3.1 การออกแบบวงจรควบคุมที่ไม่ขึ้นกับอัตราเร็วโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ [2]

กราฟบรรยายการเปลี่ยนสัญญาณ (Signal Transition Graph: STG) ถูกนำมาใช้สร้างวงจรควบคุมที่ไม่ขึ้นต่ออัตราเร็ว (Speed-independent Control Circuits) ซึ่งวงจรควบคุมดังกล่าว ใช้แบบจำลองความหน่วงที่ไม่ขึ้นต่ออัตราเร็ว กล่าวคือเป็นแบบจำลองความหน่วงที่ไม่มีการกำหนดค่าความหน่วงในสายสัญญาณ แต่จะกำหนดความหน่วงให้กับเกตในวงจร วิธีการออกแบบวงจรควบคุมที่ไม่ขึ้นต่ออัตราเร็วโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณมีขั้นตอนดังต่อไปนี้

1. สร้างแผนผังแสดงการเปลี่ยนแปลงสถานะลอจิกของอินพุทและเอาท์พุท ในแต่ละช่วงเวลา (Timing diagram)

2. สร้างกราฟบรรยายการเปลี่ยนสัญญาณให้สอดคล้องกับแผนผังในข้อ 1 โดยใช้เครื่องหมายบวก (+) เป็นสัญลักษณ์ของสัญญาณขาขึ้น หรือลอจิกมีค่าจาก 0 เป็น 1 ใช้เครื่องหมายลบ (-) เป็นสัญลักษณ์ของสัญญาณขาลง หรือลอจิกมีค่าจาก 1 เป็น 0 ใช้การขีดเส้นใต้ชื่อสัญญาณเพื่อบ่งบอกว่าสัญญาณดังกล่าวเป็นสัญญาณอินพุท และสัญญาณที่ไม่ได้ขีดเส้นใต้เป็นสัญญาณเอาท์พุท

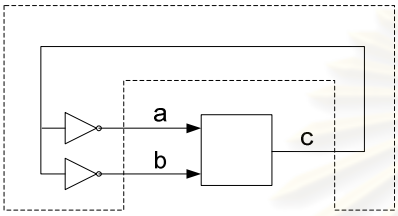
3. สร้างกราฟแสดงสถานะ (State Graph) จากกราฟในข้อ 2 โดยกำหนดให้สัญญาณขาขึ้นแทนด้วย 0* และสัญญาณขาลงแทนด้วย 1*

4. ใส่ค่าสัญญาณที่ได้จากกราฟในข้อ 3 ลงในแผนที่คาร์นอฟ (Karnaugh Map) และลดทอนสมการลอจิกโดยใช้พีชคณิตบูลีน (Boolean Algebra)

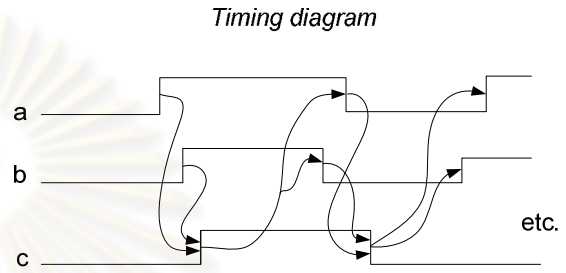
5. สร้างวงจรระดับเกตจากสมการลอจิกที่ลดทอนแล้วในข้อ 4

รูปที่ 2.7 แสดงการออกแบบอุปกรณ์ชนิดซีโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ ซึ่งมีขั้นตอนตามที่กล่าวไว้ กล่าวคือ สร้างแผนผังแสดงการเปลี่ยนแปลงสถานะลอจิกของอินพุทและเอาท์พุท ในแต่ละช่วงเวลา ได้ดังรูปที่ 2.7(ข) โดยมีพฤติกรรมการเปลี่ยนแปลงสถานะลอจิกคือ เมื่ออินพุท a และ b มีค่าลอจิกเดียวกัน คือ มีลอจิกเป็น 0 ทั้งคู่ หรือมีลอจิกเป็น 1 ทั้งคู่ ลอจิกของเอาท์พุท c จะมีค่าเท่ากันกับอินพุท a และ b แต่หากอินพุท a และ อินพุท b มีค่าลอจิกต่างกัน ค่าลอจิกของเอาท์พุท c จะไม่มีการเปลี่ยนแปลง (Next-state = Present-state)

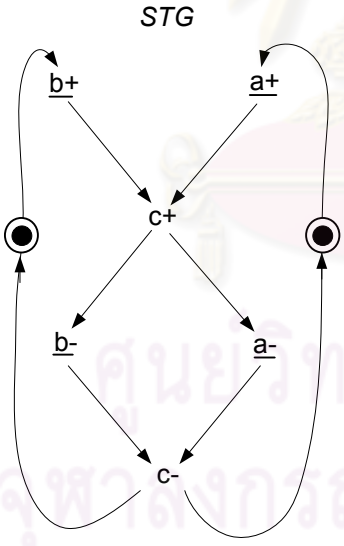
จากนั้นแปลงเป็นกราฟบรรยายการเปลี่ยนสัญญาณได้ดังรูปที่ 2.7(ค) และเขียนในรูปแบบของกราฟแสดงสถานะได้ดังรูปที่ 2.7(ง) โดยจะเขียนอยู่ในเทอมของค่า a b และ c ตามลำดับ เช่น 10*0 หมายถึง ค่า a เท่ากับ 1 ค่า b เปลี่ยนจาก 0 เป็น 1 และ ค่า c เท่ากับ 0 เป็นต้น เมื่อสรุปเป็นแผนที่คาร์นอฟจะได้ดังรูปที่ 2.7(จ) จากนั้นในขั้นตอนสุดท้ายจะแปลงสมการจากแผนที่คาร์นอฟเป็นวงจรระดับเกตของอุปกรณ์ซนิดซีได้ดังรูปที่ 2.7(ฉ) เป็นอันเสร็จสิ้นขั้นตอน



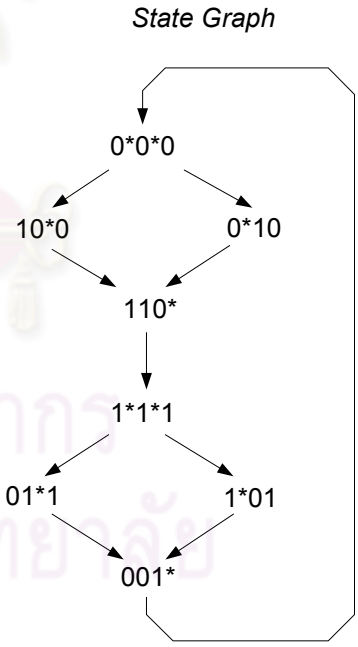
(ก) โครงร่างวงจร



(ข) แผนผังแสดงการเปลี่ยนแปลงสถานะลอจิกของอินพุทและเอาท์พุท ในแต่ละช่วงเวลา

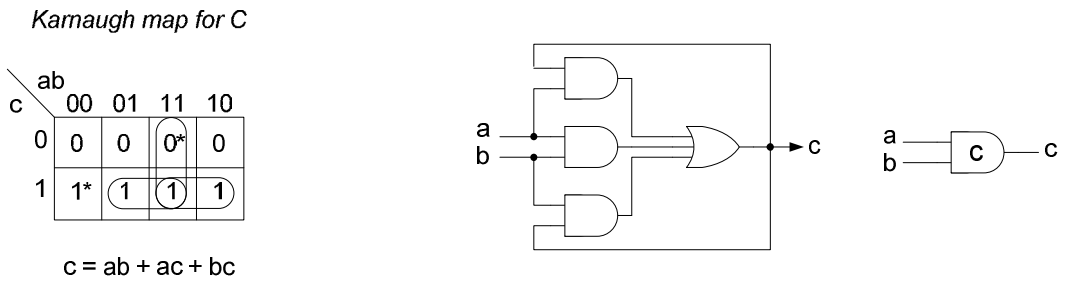


(ค) กราฟบรรยายการเปลี่ยนสัญญาณ



(ง) กราฟแสดงสถานะ

รูปที่ 2.7 การออกแบบอุปกรณ์ซนิดซีโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ



(จ) แผนที่คาร์นอฟ (ฉ) วงจรระดับเกต

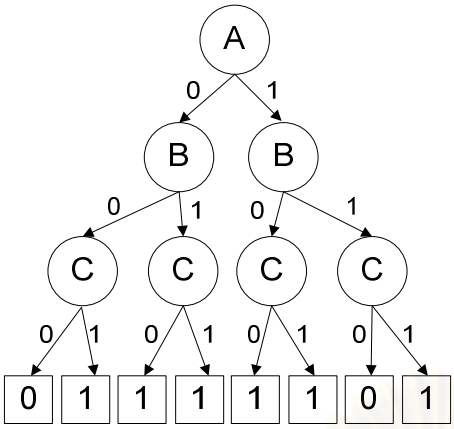
รูปที่ 2.7 การออกแบบอุปกรณ์ชนิดซีโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ (ต่อ)

2.3.2 การออกแบบวงจรอสมวารโดยใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ [8]

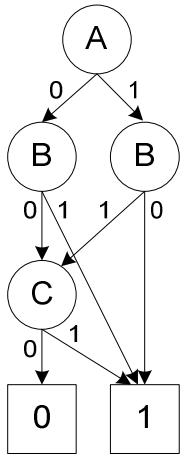
แผนภาพการตัดสินใจแบบทวิภาค (Binary Decision Diagram) เป็นแผนภาพที่ใช้อธิบายการทำงานของฟังก์ชันตรรกะ โดยแทนค่าตัวแปรในพีชคณิตบูลีนและเขียนแจกแจงเป็นโครงสร้างที่มีอันดับชั้น เพื่อช่วยในการออกแบบและสังเคราะห์วงจรที่มีขนาดใหญ่ โดยแผนภาพตัดสินใจแบบทวิภาคชนิดมีอันดับที่สามารถลดขนาดของแผนภาพลงได้จะเรียกว่า แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ (Reduced-Ordered-Binary Decision Diagram: ROBDD) วิธีการออกแบบวงจรอสมวารโดยใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ มีขั้นตอนดังต่อไปนี้

1. สร้างแผนภาพการตัดสินใจแบบทวิภาค โดยแทนค่าตัวแปรในพีชคณิตบูลีนและเขียนแจกแจงเป็นโครงสร้างที่มีอันดับชั้น
2. ลดทอนอันดับแผนภาพในข้อ 1 เพื่อเป็นการลดรูปวงจร
3. แปลงแผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับเป็นวงจรอสมวารระดับเกต

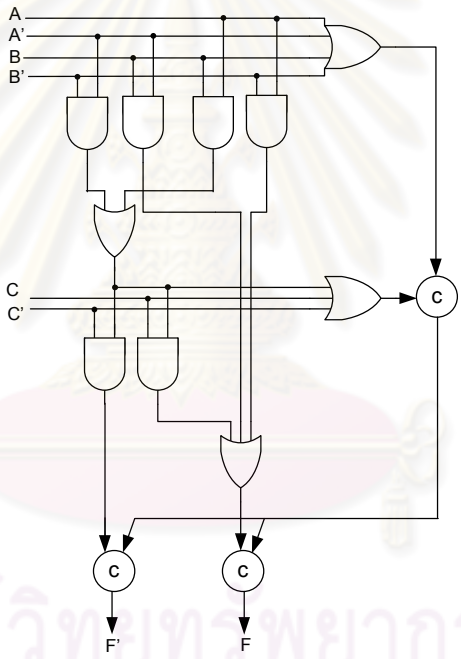
รูปที่ 2.8 แสดงการออกแบบวงจรรางคู่ของฟังก์ชัน F=AB'+A'B+C โดยใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ ซึ่งมีขั้นตอนตามที่กล่าวไว้ กล่าวคือ สร้างแผนภาพการตัดสินใจแบบทวิภาค โดยแทนค่าตัวแปรในฟังก์ชัน F=AB'+A'B+C และเขียนแจกแจงเป็นโครงสร้างที่มีอันดับชั้น ได้ดังรูปที่ 2.8(ก) โดยแต่ละกิ่งเป็นการแทนค่าของลอจิกในฟังก์ชัน



(ก) แผนภาพการตัดสินใจแบบทวิภาค



(ข) แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ

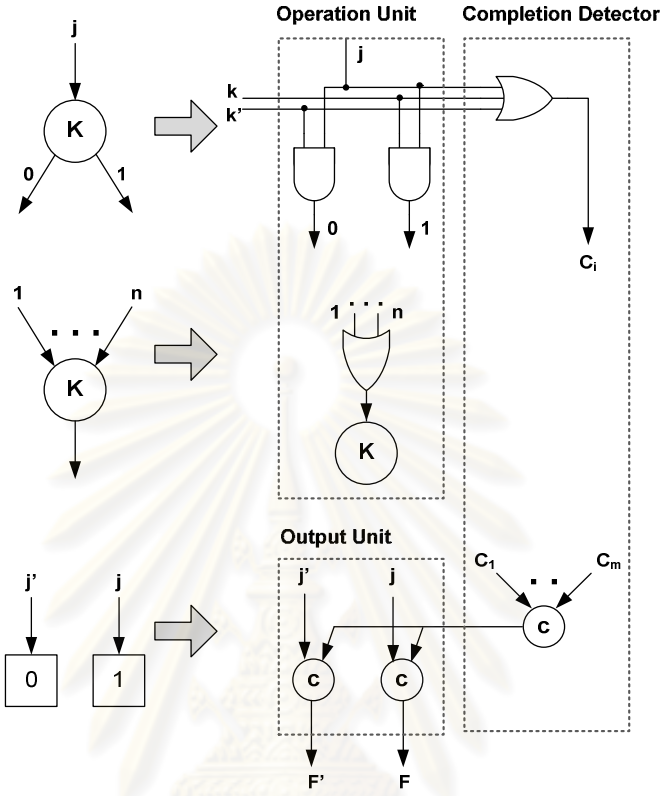


(ค) วงจรรางคู่

รูปที่ 2.8 การออกแบบวงจรรางคู่ของฟังก์ชัน F=AB'+A'B+C โดยใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ

เช่น กิ่งซ้ายสุดหมายถึง เมื่อค่าลอจิกของอินพุท A B และ C เท่ากับ 0 ค่าลอจิกของเอาท์พุท F จะมีค่าเท่ากับ 0 จากนั้นลดทอนอันดับแผนภาพดังกล่าวจนได้ดังรูปที่ 2.8(ข) ในขั้นตอนสุดท้ายจะแปลงแผนภาพที่ลดทอนแล้วเป็นวงจรรางคู่ของฟังก์ชัน F=AB'+A'B+C ได้ดังรูปที่ 2.8(ค) เป็นอัน

เสร็จสิ้นขั้นตอน โดยหลักการแปลงแผนภาพตัดสินใจแบบทวิภาคชนิดลดทอนอันดับเป็นวงจรรางคู่ แสดงดังรูปที่ 2.9



รูปที่ 2.9 การแปลงแผนภาพตัดสินใจแบบทวิภาคชนิดลดทอนอันดับเป็นวงจรรางคู่

2.4 บัส

บัส คือ ช่องทางที่ใช้ในการรับส่งข้อมูลไปยังอุปกรณ์ต่างๆในคอมพิวเตอร์ ที่เชื่อมต่อถึงกัน โดยบัสจะถูกใช้งานจากหลายอุปกรณ์ ดังนั้นการออกแบบบัสจึงต้องคำนึงถึงลำดับการเข้าเรียกใช้ของแต่ละอุปกรณ์ เนื่องจาก ณ เวลาหนึ่ง จะมีอุปกรณ์เพียงอุปกรณ์เดียวเท่านั้นที่เรียกใช้บัสได้ [9] ซึ่งบัสโดยทั่วไปจะสามารถรับส่งข้อมูลได้ทั้งสองทิศทาง (Bi-directional)

2.4.1 ส่วนประกอบของบัส

ส่วนประกอบของบัสประกอบด้วย บัสข้อมูล (Data Bus) บัสเลขที่อยู่ (Address Bus) และบัสควบคุม (Control Bus) โดยแต่ละส่วนมีหน้าที่การทำงาน [4] ดังต่อไปนี้

1. บัสข้อมูล

ทำหน้าที่รับส่งข้อมูลจากจุดหนึ่งไปยังจุดหนึ่งภายในบัส หากจำนวนเส้นของบัสข้อมูลมีมาก ความจุในการรับส่งข้อมูลต่อเวลาจะมีจำนวนมากตามไปด้วย ยกตัวอย่างเช่น บัสระบบแบบ 16 บิต หมายถึงบัสสามารถรับส่งข้อมูลได้ครั้งละ 16 บิต

2. บัสเลขที่อยู่

ใช้เก็บตำแหน่งที่ต้องการรับส่งข้อมูล โดยจะเก็บตำแหน่งของต้นทางที่จะส่งข้อมูลออกไป และตำแหน่งของปลายทางที่จะรับข้อมูลไปเก็บ ตำแหน่งที่ถูกอ้างถึงเช่น ตำแหน่งของหน่วยความจำ ตำแหน่งของช่องทางหรือพอร์ท (Port) ในอุปกรณ์อินพุท/เอาท์พุท (I/O)

3. บัสควบคุม

ทำหน้าที่ควบคุมสัญญาณที่กำหนดว่าจะให้บัสทำการอ่านหรือเขียนข้อมูลลงตามคำสั่งที่ได้รับ โดยทั่วไปมีหน้าที่ควบคุมการทำงาน ดังต่อไปนี้

การเขียนข้อมูลลงหน่วยความจำ (Memory Write): เป็นการเขียนข้อมูลลงในหน่วยความจำจากบัสตามตำแหน่งที่อยู่ที่ได้จากสายที่เลขที่อยู่

การอ่านข้อมูลจากหน่วยความจำ (Memory Read): เป็นการนำข้อมูลจากหน่วยความจำจากตำแหน่งที่อยู่ที่ส่งมาร้องขอ แล้วนำข้อมูลเข้าไปยังบัส

การเขียนอินพุท/เอาท์พุท (I/O Write): เป็นการนำข้อมูลในบัสเขียนลงตำแหน่งที่อยู่ของพอร์ทอินพุท/เอาท์พุท

การอ่านอินพุท/เอาท์พุท (I/O Read): เป็นการอ่านข้อมูลจากตำแหน่งที่อยู่ของพอร์ทอินพุท/เอาท์พุท แล้วนำข้อมูลที่อ่านได้ไปไว้ในบัส

สัญญาณตอบรับ (Acknowledge Signal): เป็นตัวบอกว่าข้อมูลได้รับเป็นที่เรียบร้อยแล้ว (ในกรณีการเขียน) และข้อมูลได้เข้าไปในบัสเรียบร้อยแล้ว (ในกรณีการอ่าน)

สัญญาณร้องขอ (Request Signal): เป็นสัญญาณที่อุปกรณ์แต่ละตัวใช้ในการร้องขอเพื่อใช้บัสในการรับส่งข้อมูล

การอนุญาตให้ใช้บัส (Bus Grant): เป็นผลมากจากการร้องขอ ควบคุมโดยตัวตัดสินใจให้อนุญาตให้ใช้บัส

การร้องขออินเตอร์รัพท์ (Interrupt Request): เป็นสัญญาณที่บอกว่าสัญญาณร้องขอในขณะนั้นกำลังรอการอนุญาตจากตัวตัดสินใจ (Bus Arbiter) อยู่

การตอบรับอินเตอร์รัพท์ (Interrupt Acknowledge): เป็นการตอบรับจากสัญญาณร้องขออินเตอร์รัพท์

สัญญาณนาฬิกา (Clock): ใช้ในการควบคุมลำดับการทำงานของวงจร (ในกรณีของวงจรอสมวารจะไม่ใช้สัญญาณนาฬิกาในการควบคุม)

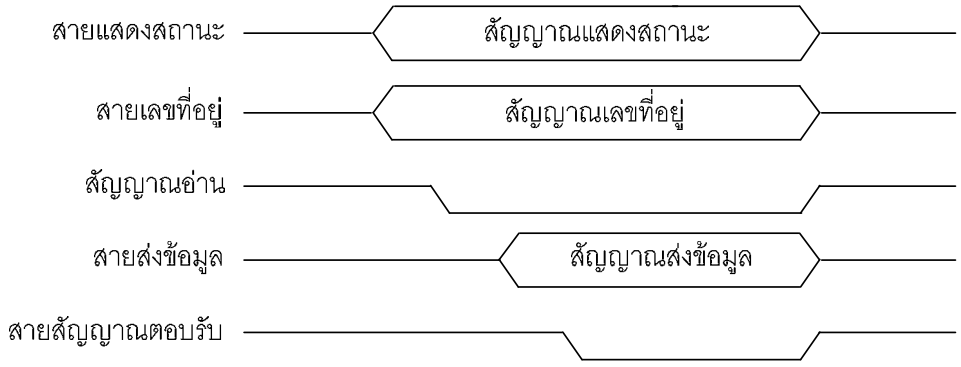
รีเซ็ต (Reset): เป็นการเริ่มต้นค่าใหม่

2.4.2 **ชนิดของบัส**

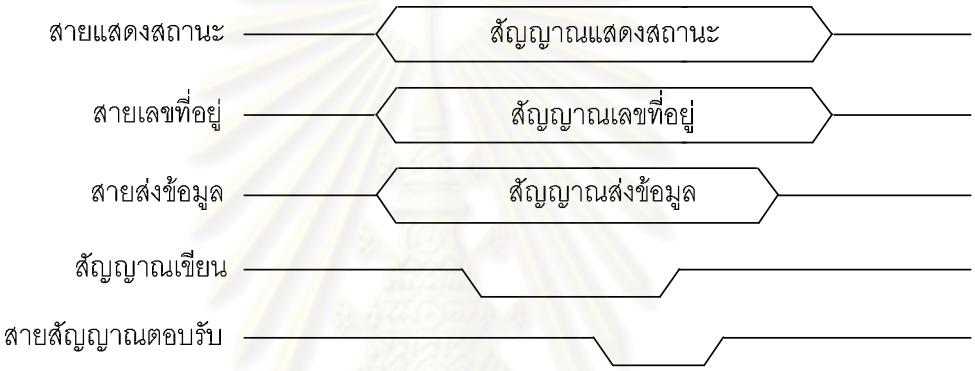
บัสแบ่งออกเป็น 2 ชนิด คือ บัสแบบเดดิเคต (Dedicated Bus) และบัสแบบมัลติเพล็กซ์ (Multiplex Bus) บัสแบบเดดิเคตประกอบด้วยสายรับส่งเลขที่อยู่ และสายรับส่งข้อมูลซึ่งจะแยกกันอย่างชัดเจน เกิดข้อดีคือสามารถรับส่งเลขที่อยู่พร้อมกับข้อมูลได้ในเวลาเดียวกัน แต่มีข้อเสียคือใช้สายสัญญาณจำนวนมาก บัสแบบมัลติเพล็กซ์จะใช้สายสัญญาณในการรับส่งเลขที่อยู่และข้อมูลร่วมกัน เกิดข้อดีคือใช้สายสัญญาณจำนวนน้อยกว่าแบบเดดิเคต แต่มีข้อเสียคือทำงานได้ช้ากว่าแบบเดดิเคต เนื่องจากไม่สามารถรับส่งเลขที่อยู่พร้อมกับข้อมูลได้ในเวลาเดียวกัน นอกจากนี้การออกแบบยังทำได้ยากกว่า

2.4.3 **หลักการการทำงานของบัส**

การทำงานของบัสแบบอสมวารมีจังหวะการอ่านข้อมูลดังรูปที่ 2.10 (ก) โดยไมโครโพรเซสเซอร์จะส่งสัญญาณเลขที่อยู่ และสัญญาณแสดงสถานะ (หรืออาจใช้สัญญาณร้องขอ) ป้อนเข้าบัส จากนั้นก็จะรอให้สัญญาณเลขที่อยู่ที่ป้อนเข้าบัสเสถียรแล้ว จึงป้อนสัญญาณอ่าน เพื่อสั่งให้บัสทำการอ่านข้อมูล จากนั้นเมื่อหน่วยความจำได้รับสัญญาณเลขที่อยู่ และสัญญาณอ่าน จะถอดรหัสสัญญาณเลขที่อยู่ แล้วจึงนำข้อมูลในตำแหน่งสัญญาณเลขที่อยู่ที่ได้รับป้อนเข้าสู่สัญญาณข้อมูลในบัส เมื่อข้อมูลในบัสเสถียรแล้ว ตัวควบคุมบัสก็จะส่งสัญญาณตอบรับเพื่อบอกว่าบัสได้รับข้อมูลแล้ว จากนั้นเมื่ออุปกรณ์ร้องขอได้อ่านข้อมูลออกไปจากบัสแล้ว อุปกรณ์นั้นก็จะปล่อยสัญญาณที่ร้องขอไว้ เป็นผลให้หน่วยความจำหยุดส่งข้อมูล ตัวควบคุมบัสจะหยุดส่งสัญญาณตอบรับ สุดท้ายทุกสัญญาณจะถูกปล่อยทั้งหมด



(ก) รอบการอ่านข้อมูลของบัสแบบอสมวาร



(ข) รอบการเขียนข้อมูลของบัสแบบอสมวาร

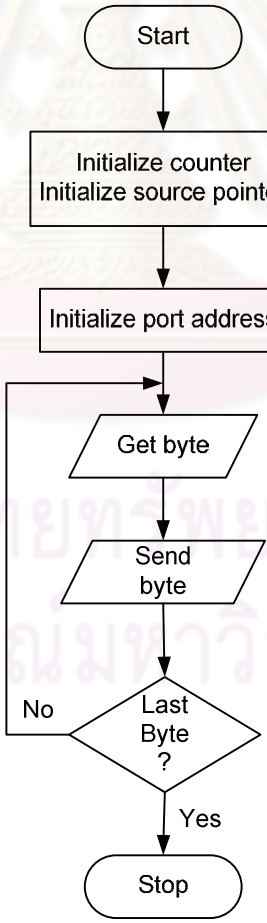
รูปที่ 2.10 รอบการอ่านเขียนข้อมูลของบัสแบบอสมวาร

สำหรับจังหวะการเขียนข้อมูลดังรูปที่ 2.10 (ข) โดยอุปกรณ์ที่ต้องการเขียนข้อมูลจะป้อนข้อมูลลงในบัส ในขณะเดียวกันไมโครโพรเซสเซอร์จะส่งสัญญาณเลขที่อยู่ และสัญญาณแสดงสถานะ (หรืออาจใช้สัญญาณร้องขอ) ป้อนเข้าบัส จากนั้นก็จะรอให้สัญญาณเลขที่อยู่และข้อมูลที่ป้อนเข้าบัสเสถียรแล้ว จึงป้อนสัญญาณเขียน เพื่อสั่งให้หน่วยความจำเขียนข้อมูลจากบัส เมื่อหน่วยความจำได้รับสัญญาณเขียนข้อมูลแล้ว ก็จะเริ่มทำการเขียนข้อมูลลงในตำแหน่งของสายสัญญาณเลขที่อยู่ที่ได้รับ เมื่อเขียนเสร็จแล้วก็จะส่งสัญญาณตอบรับกลับไป จากนั้นอุปกรณ์ที่ร้องขอเขียนข้อมูลก็จะปล่อยสัญญาณที่ร้องขอไว้ หน่วยความจำก็จะปล่อยสัญญาณตอบรับสุดท้ายทุกสัญญาณจะถูกปล่อยทั้งหมด

2.5 ดีเอ็มเอ

ไมโครโพรเซสเซอร์ทำหน้าที่ประมวลผลงานต่างๆที่มีความหลากหลาย รวมถึงงานที่ต้องรับส่งข้อมูลจากอุปกรณ์ต่อพ่วงภายนอกมาใช้ด้วย เช่น การโหลดโปรแกรมจากฮาร์ดดิสไปยังหน่วยความจำ เป็นต้น โดยไมโครโพรเซสเซอร์จะต้องคอยควบคุมการรับส่งข้อมูลจากอุปกรณ์ต่อพ่วง ซึ่งความเร็วในการทำงานของอุปกรณ์ต่อพ่วงต่ำกว่าไมโครโพรเซสเซอร์อยู่มาก ทำให้ต้องเสียเวลาคอยอุปกรณ์ ดีเอ็มเอ (Direct Memory Access: DMA) [26] สามารถแบ่งเบาภาระดังกล่าวได้ด้วยการทำหน้าที่ควบคุมการรับส่งข้อมูลของอุปกรณ์ต่อพ่วงโดยตรงแทนไมโครโพรเซสเซอร์ ส่งผลให้ไมโครโพรเซสเซอร์ไม่ต้องเสียเวลารอการติดต่อจากอุปกรณ์ต่อพ่วงและสามารถไปทำงานอื่นได้

2.5.1 หลักการทำงานของดีเอ็มเอ



รูปที่ 2.11 หลักการทำงานของดีเอ็มเอ

หลักการทำงานของดีเอ็มเอคือ ไมโครโพรเซสเซอร์จะส่งสัญญาณเริ่มต้นบอกให้ดีเอ็มเอรับผิดชอบการรับส่งข้อมูลจากอุปกรณ์ต่อพ่วง เมื่อมีงานที่เกี่ยวข้องกับอุปกรณ์ต่อพ่วงเข้ามา โดยสัญญาณเริ่มต้นประกอบด้วย การระบุตำแหน่งที่อยู่ที่ต้องการส่งข้อมูล และตำแหน่งที่อยู่ที่ต้องการรับข้อมูล (Initialize source pointer) การระบุพอร์ทของอุปกรณ์ที่ต้องการรับส่งข้อมูล (Initialize port address) รวมถึงจำนวนข้อมูลที่ต้องการเคลื่อนย้ายจากอุปกรณ์ (Initialize counter) จากนั้นไมโครโพรเซสเซอร์จะไปทำงานอื่นและปล่อยให้ดีเอ็มเอทำหน้าที่เคลื่อนย้ายข้อมูลกับอุปกรณ์ต่อพ่วงตามที่สัญญาณเริ่มต้นระบุ ดังแสดงในรูปที่ 2.11 [10] และเมื่อดีเอ็มเอทำงานเสร็จจะส่งสัญญาณอินเตอร์รัพท์ไปที่ไมโครโพรเซสเซอร์ เพื่อแจ้งผลเสร็จสิ้นของงานให้ไมโครโพรเซสเซอร์ทราบ

2.5.2 การเชื่อมต่อดีเอ็มเอ

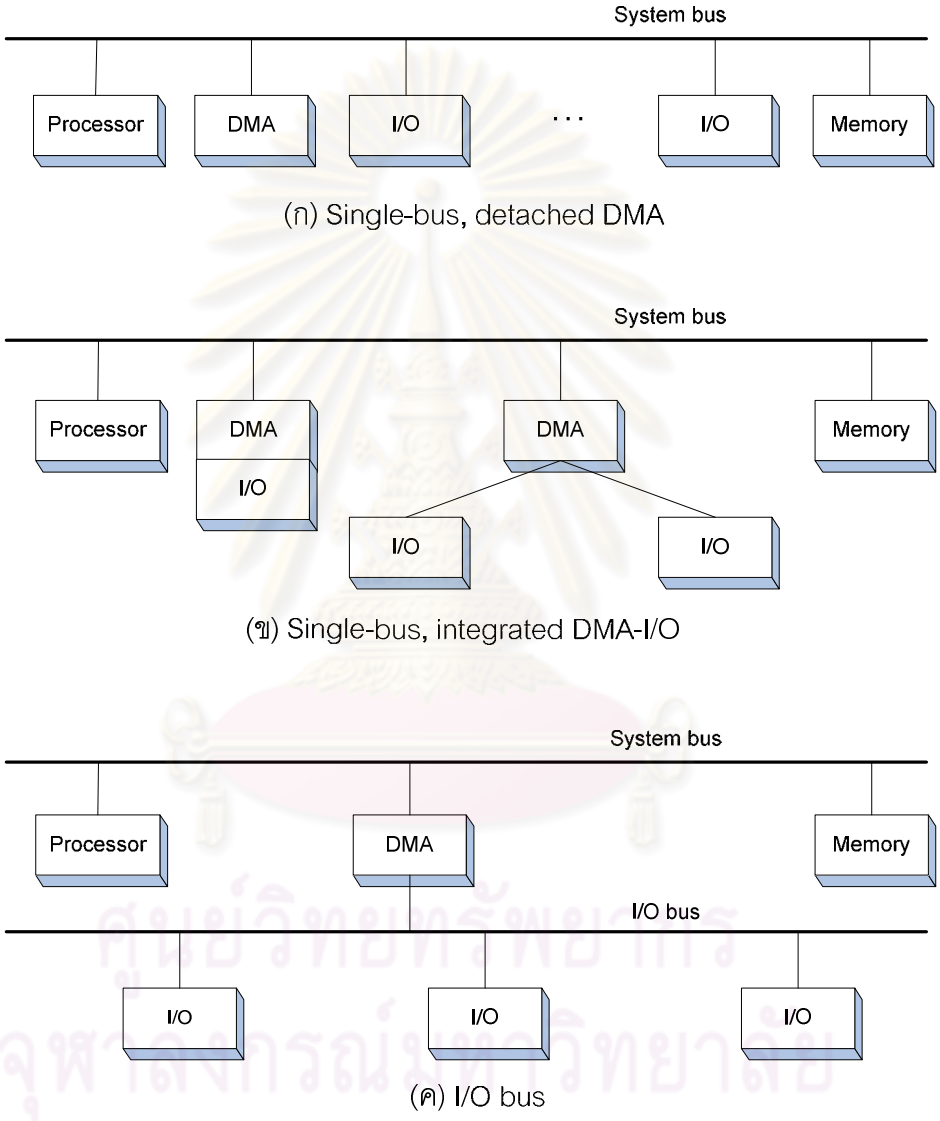
โครงสร้างของการเชื่อมต่อระหว่างดีเอ็มเอกับไมโครโพรเซสเซอร์ หน่วยความจำหลัก และอุปกรณ์ต่อพ่วง (Alternative DMA Configurations) เป็นไปได้หลายแบบดังแสดงในรูปที่ 2.12 คือ

1. Single-bus, detached DMA : ดีเอ็มเอ ไมโครโพรเซสเซอร์ หน่วยความจำหลัก และอุปกรณ์ต่อพ่วง เชื่อมต่อกันผ่านบัสระบบเพียงอย่างเดียว ข้อดีคือออกแบบง่ายและใช้ค่าใช้จ่ายในการสร้างต่ำ แต่มีข้อเสียคือเมื่อดีเอ็มเอทำงานรับส่งข้อมูลจากอุปกรณ์ต่อพ่วงซึ่งต้องใช้บัสระบบ จะส่งผลให้ไมโครโพรเซสเซอร์ไม่สามารถใช้งานบัสระบบเพื่อติดต่อกับหน่วยความจำในขณะนั้นได้

2. Single-bus, integrated DMA-I/O : ดีเอ็มเอ ไมโครโพรเซสเซอร์ หน่วยความจำหลัก เชื่อมต่อกันผ่านบัสระบบ โดยอุปกรณ์ต่อพ่วงจะเชื่อมต่อโดยตรงกับดีเอ็มเอ ข้อดีคือ เมื่อดีเอ็มเอทำงานรับส่งข้อมูลจากอุปกรณ์ต่อพ่วงไปยังอุปกรณ์ต่อพ่วงด้วยกันจะไม่ใช้บัสระบบ ส่งผลให้ไมโครโพรเซสเซอร์สามารถใช้งานบัสระบบเพื่อติดต่อกับหน่วยความจำในขณะนั้นได้ แต่มีข้อเสียคือ การเชื่อมต่อกันโดยตรงดังกล่าว เมื่อมีอุปกรณ์เชื่อมต่อจำนวนมากขึ้นจะทำให้ส่วนเชื่อมต่อเพิ่มจำนวนมากขึ้นตาม เกิดการสิ้นเปลืองทรัพยากรในระบบ

3. I/O bus : ดีเอ็มเอ ไมโครโพรเซสเซอร์ หน่วยความจำหลัก เชื่อมต่อกันผ่านบัสระบบ ส่วนอุปกรณ์ต่อพ่วงจะเชื่อมต่อกับดีเอ็มเอผ่านบัสสำหรับอุปกรณ์ต่อพ่วง เมื่อดีเอ็มเอ

ทำงานรับส่งข้อมูลจากอุปกรณ์ต่อพ่วงไปยังอุปกรณ์ต่อพ่วงด้วยกันจะไม่ใช้บัสระบบ ส่งผลให้ไมโครโพรเซสเซอร์สามารถใช้งานบัสระบบเพื่อติดต่อกับหน่วยความจำในขณะนั้นได้ เช่นเดียวกับแบบ Single-bus, integrated DMA-I/O แต่การเชื่อมต่อดีเอ็มเอกับอุปกรณ์ต่อพ่วงผ่านบัสสำหรับอุปกรณ์ต่อพ่วง จะช่วยลดโครงสร้างการเชื่อมต่อเมื่อมีอุปกรณ์จำนวนมากขึ้น จึงประหยัดทรัพยากรระบบได้มากกว่า



รูปที่ 2.12 การเชื่อมต่อดีเอ็มเอกับระบบ [11]

2.6 การใช้พลังงานของวงจร

พลังงานที่วงจรสูญเสียไปแบ่งออกเป็น พลังงานสแตติก (Static Power) และพลังงานไดนามิก (Dynamic power) ดังสมการที่ 2.1 [12] โดยพลังงานสแตติกสูญเสียจากการ

รั่วไหลของกระแสไฟฟ้าภายในตัวไดโอดหรือทรานซิสเตอร์ (Leakage Power) พลังงานไดนามิกสูญเสียจากการใช้พลังงานเปลี่ยนสถานะของสัญญาณภายในวงจร (Switching Power) เช่น เปลี่ยนสถานะสัญญาณจากสถานะ 0 เป็นสถานะ 1 หรือ จากสถานะ 1 เป็นสถานะ 0 เป็นต้น

$$P_{total} = P_{static} + P_{dynamic} \quad (2.1)$$

$$P_{total} = (P_{switching} + P_{short}) + P_{leakage}$$

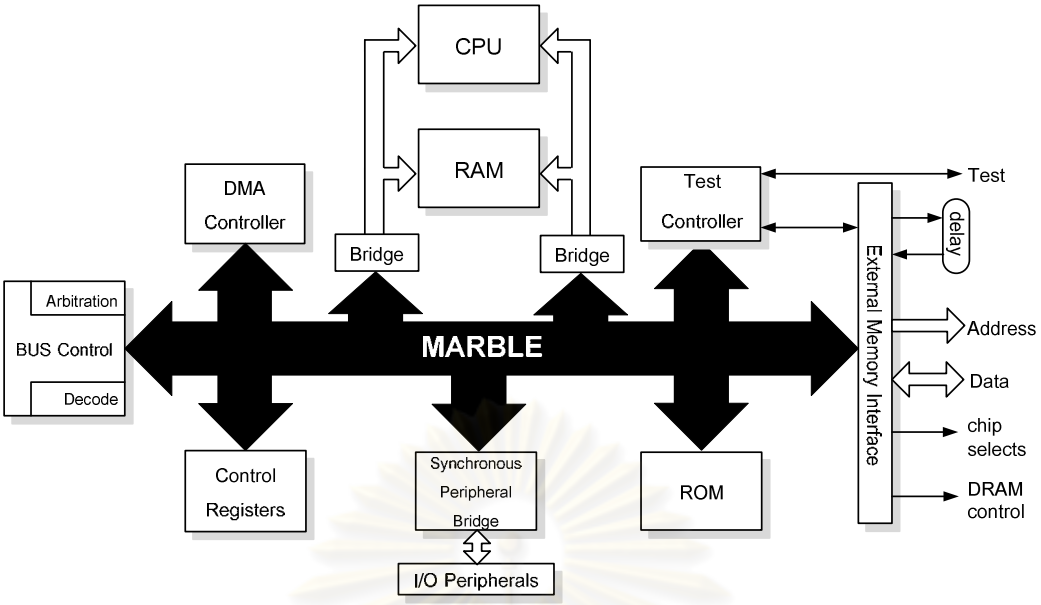
การออกแบบวงจรให้สูญเสียพลังงานในวงจรน้อยลง หรือให้วงจรใช้พลังงาน (Power Consumption) ต่ำ สามารถทำได้หลายแนวทาง เช่น การลดขนาดวงจร โดยใช้ไดโอดและทรานซิสเตอร์ลดลงเพื่อลดการสูญเสียพลังงานสแตติก การลดความถี่ของการเปลี่ยนสถานะสัญญาณภายในวงจรเพื่อลดการสูญเสียพลังงานไดนามิก เป็นต้น

2.7 งานวิจัยที่เกี่ยวข้อง

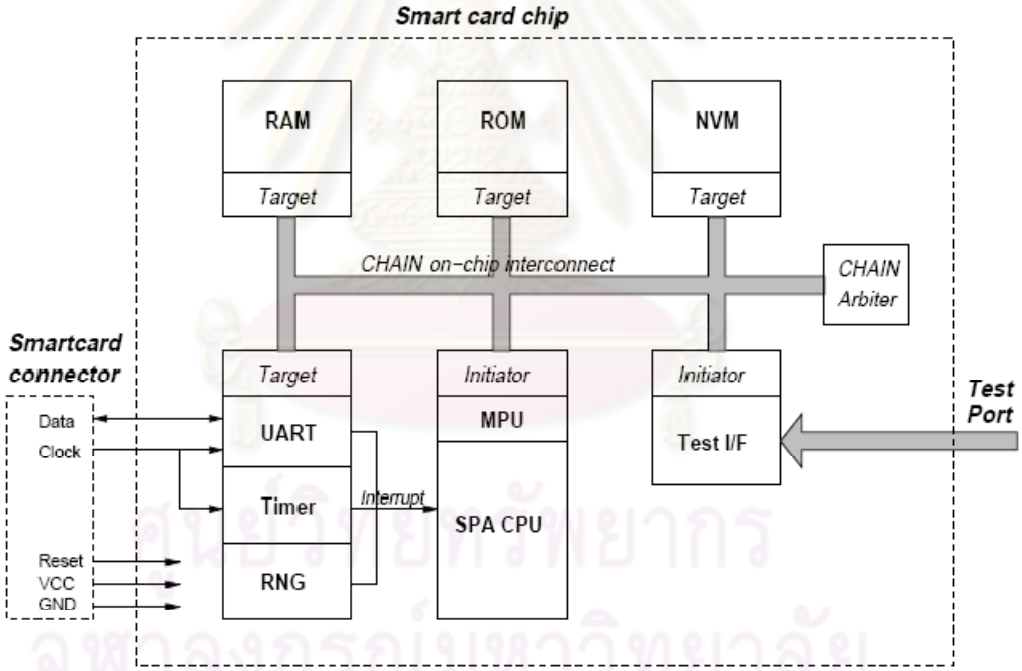
ในส่วนนี้จะอธิบายถึงงานวิจัยที่เกี่ยวข้องกับการออกแบบบัสระบบแบบอสมวาร โดยมีรายละเอียดดังนี้

2.7.1 งานวิจัยพัฒนาบัสระบบ MARBLE

งานวิจัย [7] พัฒนาบัสระบบ MARBLE (Manchester AsynchRnous Bus for Low Energy) [14] ซึ่งใช้เชื่อมต่อไมโครโพรเซสเซอร์ AMULET3i กับแกนหน่วยประมวลผล และตัวควบคุมดีเอ็มเอกับหน่วยความจำแรมรอมและอุปกรณ์อื่นๆดังรูปที่ 2.13 โดยพัฒนาปรับปรุงบัสระบบดังกล่าวให้ปริมาณงานที่ทำในหนึ่งหน่วยเวลา (Throughput) มีค่าสูงขึ้น และลดปัญหาความผิดพลาดของการรับส่งข้อมูลในสายสัญญาณ ที่เกิดจากการรบกวนของสายสัญญาณที่ทำงานอยู่ใกล้กัน (Crosstalk) ลงได้ โดยใช้การเข้ารหัสข้อมูลแบบหนึ่งในสี่ แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วง (Delay Insensitive: DI) สัญญาณอาณัติแบบ 4 ขั้น และโครงสร้างการทำงานแบบสายท่อแบบอสมวาร (Asynchronous Pipelines) ในการออกแบบ



รูปที่ 2.13 บัสระบบ MARBLE และองค์ประกอบวงจร



รูปที่ 2.14 บัสระบบ CHAIN และองค์ประกอบของวงจร

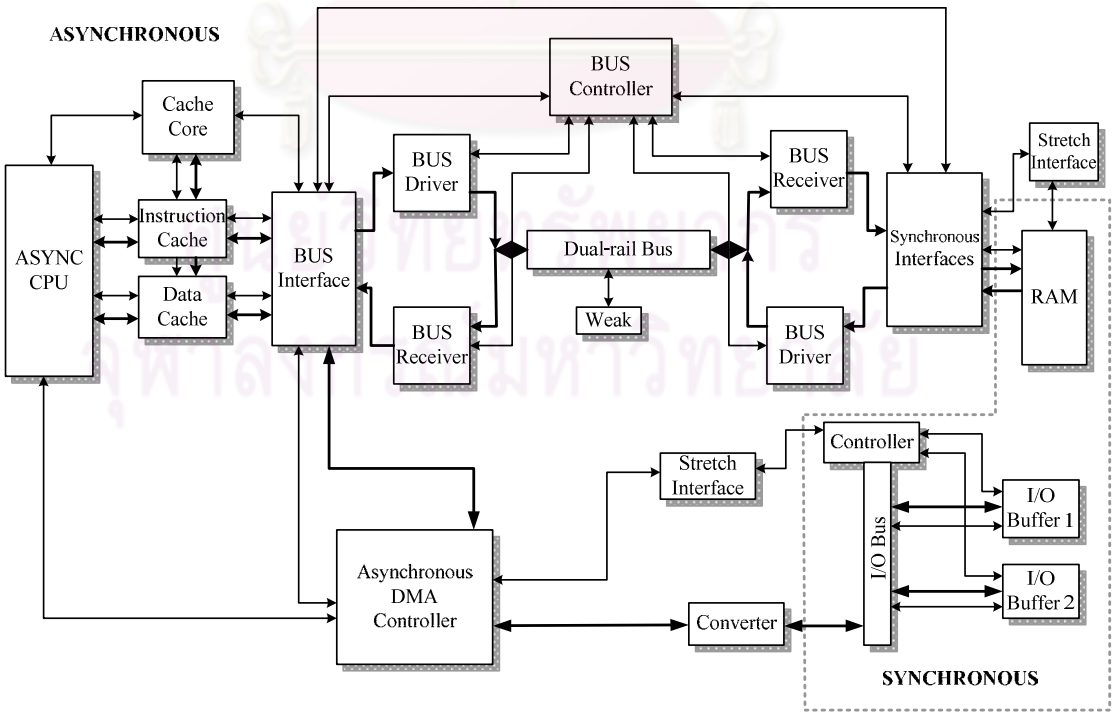
2.7.2 งานวิจัยบัสระบบ CHAIN

งานวิจัย [15] นำบัสระบบ CHAIN ซึ่งประยุกต์มาจากบัสระบบ MARBLE ในงานวิจัยชิ้นก่อน [7] ไปใช้กับโพรเซสเซอร์ SPA ซึ่งใช้งานบนชิปของสมาร์ตคาร์ด (Smartcard

Chip) พบว่ามีประสิทธิภาพด้านความถูกต้องของข้อมูลและความปลอดภัยของข้อมูลสูง ยากต่อการโจมตีระบบ โครงสร้างของบัสระบบ CHAIN และองค์ประกอบของวงจรดังกล่าวเป็นดังรูปที่ 2.14

2.7.3 งานวิจัยการออกแบบบัสสำหรับวงจรอสมวาร

งานวิจัยการออกแบบบัสระบบเพื่อเป็นต้นแบบสำหรับวงจรอสมวาร [4] ประกอบด้วย การออกแบบบัส อินเตอร์รัพท์ ดีเอ็มเอ และนำเสนอการเชื่อมต่อระหว่างวงจรสมวารและอสมวาร โดยบัสระบบที่ออกแบบใช้สายสัญญาณข้อมูลและสายเลขที่อยู่ขนาด 8 บิต เข้ารหัสโดยใช้รหัสรางคู่ ใช้แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วงชนิดเสมือน และสัญญาณอาณัติแบบ 4 ขั้นในการออกแบบ มีคุณสมบัติรับส่งข้อมูลได้ทั้งสองทิศทางและสามารถเชื่อมต่อกับวงจรสมวารและอสมวารได้ ส่วนขององค์ประกอบวงจรได้ดัดแปลงในส่วนคำสั่ง โครงสร้างและส่วนควบคุมของไมโครโพรเซสเซอร์แบบอสมวารที่เชื่อมต่อกับบัส เพื่อให้สามารถเรียกใช้งานบัสระบบแบบอสมวารได้ ดีเอ็มเอถูกออกแบบเป็นแบบสมวารและอสมวาร ส่วนควบคุมอินพุท/เอาท์พุทถูกออกแบบให้ควบคุมการรับส่งข้อมูลอินพุท/เอาท์พุทแบบสมวาร วงจรที่สมบูรณ์ของบัสระบบและองค์ประกอบวงจรเป็นดังรูปที่ 2.15



รูปที่ 2.15 บัสระบบแบบอสมวารและองค์ประกอบวงจร

งานวิจัยนี้มีแนวคิดในการนำการออกแบบบัสระบบเพื่อเป็นต้นแบบสำหรับวงจรอสมวาร [4] ดังกล่าว มาเป็นต้นแบบในการออกแบบบัสระบบ และใช้รหัสหนึ่งในสี่เข้ารหัสในการส่งข้อมูลเพื่อเพิ่มประสิทธิภาพทางด้านการใช้พลังงานให้กับบัสระบบ

บทที่ 3

การออกแบบบัสระบบแบบอสมวาร

ในบทนี้กล่าวถึงการออกแบบบัสระบบแบบอสมวาร ด้วยวิธีการเข้ารหัสหนึ่งในสี่ ซึ่งใช้เป็นช่องทางในการรับส่งข้อมูลไปยังอุปกรณ์ต่างๆที่เชื่อมต่อถึงกันในระบบ รายละเอียดของการออกแบบบัสระบบแบ่งออกเป็น คุณสมบัติของบัสระบบ โครงสร้างของบัสระบบ และการทำงานของบัสระบบ โดยมีรายละเอียดดังต่อไปนี้

3.1 คุณสมบัติของบัสระบบ

บัสระบบแบบอสมวารที่ออกแบบ มีคุณสมบัติโดยสรุปดังนี้

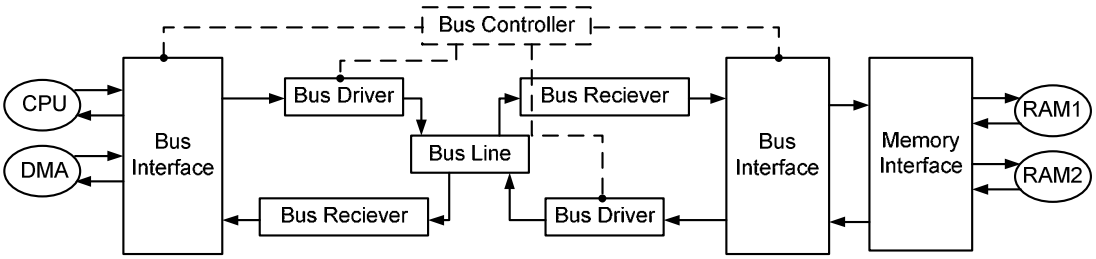
- ทำงานแบบมัลติเพล็กซ์ เพื่อลดจำนวนการใช้สายสัญญาณของบัส
- บัสมีความกว้าง 20'1of4* บิต ซึ่งเพิ่มจากบัสเวอร์ชันเดิม [4] ที่มีความกว้างเพียง16'dual-rail** บิต เพื่อรองรับการใช้งานรับส่งเลขที่อยู่ขนาด 20'1of4 บิต ซึ่งเพิ่มจากบัสเวอร์ชันเดิมที่รองรับการใช้งานรับส่งเลขที่อยู่เพียง 16'dual-rail บิต และรองรับการใช้งานรับส่งข้อมูลขนาด 16'1of4 บิต
- เชื่อมต่อกับไมโครโพรเซสเซอร์ ดีเอ็มเอ และหน่วยความจำขนาด 1Kx8bits จำนวน 2 ตัว เพื่อเพิ่มการติดต่อกับหน่วยความจำ จากบัสเวอร์ชันเดิม ที่เชื่อมต่อกับหน่วยความจำขนาด 256x8bits จำนวนเพียง 1 ตัว
- เข้ารหัสหนึ่งในสี่กับสัญญาณอาณัติแบบ 4 ขั้น แทนการเข้ารหัสรางคู่กับสัญญาณอาณัติแบบ 4 ขั้นของบัสเวอร์ชันเดิม เพื่อลดการเปลี่ยนสถานะสัญญาณของบัส
- ไม่ใช้สถานะไฮ-อิมพีแดนซ์ (High Impedance: Hi-Z) ในการทำงาน ใช้เพียงสัญญาณดิจิตอล คือ สถานะ 0 และสถานะ 1 เพื่อให้ทำงานแบบอสมวารได้

---

* '1of4 คือ จำนวนบิตของข้อมูลเข้ารหัสหนึ่งในสี่

** 'dual-rail คือ จำนวนบิตของข้อมูลเข้ารหัสรางคู่

3.2 โครงสร้างของบัสระบบ



รูปที่ 3.1 โครงสร้างของบัสระบบ

โครงสร้างของบัสระบบที่ออกแบบแสดงดังรูปที่ 3.1 ปรับปรุงจากบัสระบบเข้ารหัสรางคู่เวอร์ชันเดิม [4] ประกอบด้วย บัส ส่วนติดต่อกับไมโครโพรเซสเซอร์และดีเอ็มเอ (Bus Interface) ส่วนติดต่อกับหน่วยความจำแบบสมวาร (Memory Interface) และตัวควบคุมบัส (Bus Controller) โดยมีรายละเอียดของแต่ละส่วนประกอบดังนี้

3.2.1 บัส

บัสแบบอสมวารเข้ารหัสหนึ่งในสี่ประกอบด้วย ตัวขับบัส (Bus Driver) สายสัญญาณบัส (Bus Line) และตัวรับบัส (Bus Receiver) โดยปรับปรุงจากโครงสร้างบัสแบบอสมวารเข้ารหัสรางคู่เวอร์ชันเดิม [4] ดังรูปที่ 3.2(ก) เป็นเวอร์ชันปรับปรุงดังรูปที่ 3.2(ข) ซึ่งได้ลดรูปโครงสร้างบัสส่วนที่ซ้ำซ้อนออก จากนั้นปรับปรุงให้บัสเข้ารหัสหนึ่งในสี่แทนการเข้ารหัสรางคู่ได้ดังรูปที่ 3.2(จ) รายละเอียดของการออกแบบและปรับปรุงบัสทั้งหมด มีดังนี้

1. ตัวขับบัส: ออกแบบโดยใช้ลอจิกสามสถานะ (Tri-state Buffer) เช่นเดียวกับบัสเวอร์ชันเดิม ซึ่งลอจิกสามสถานะมีลักษณะการทำงานเหมือนสวิทช์ โดยเมื่อเปิดสวิทช์ (Enable=1) ลอจิกสามสถานะจะยอมให้ค่าลอจิกของอินพุทผ่านออกไปยังเอาท์พุท กล่าวคือค่าลอจิกของเอาท์พุทจะเท่ากับอินพุท ซึ่งมีความหมายว่าบัสถูกใช้งาน แต่เมื่อปิดสวิทช์ (Enable=0) ลอจิกสามสถานะจะตัดการเชื่อมต่อระหว่างอินพุทและเอาท์พุท กล่าวคือค่าลอจิกของเอาท์พุทจะมีค่าเท่ากับไฮ-อิมพีแดนซ์ ซึ่งมีความหมายว่าบัสว่าง ลอจิกสามสถานะสามารถต่อร่วมกันบนสายสัญญาณบัสเส้นเดียวได้ ซึ่งคุณสมบัตินี้ตรงกับความสามารถของบัสที่ต้องรองรับการใช้งาน

จากหลายอุปกรณ์ร่วมกันบนบัสเดียว ลอจิกสามสถานะจึงถูกใช้เป็นสวิทช์เปิด-ปิดให้สัญญาณของแต่ละอุปกรณ์ผ่านเข้าสู่บัสในการออกแบบทั่วไป

บัสเวอร์ชันเดิมยังประกอบด้วยส่วนตรวจสอบความพร้อมของสัญญาณก่อนเริ่มส่งข้อมูลเข้าบัส ซึ่งสามารถตัดออกได้ เนื่องจากบัสต้องใช้งานควบคู่กับส่วนพักข้อมูลต้นทางของบัส (Source Latch) ซึ่งทำหน้าที่เก็บข้อมูลก่อนเริ่มส่งเข้าสู่บัส ส่วนพักข้อมูลดังกล่าวประกอบด้วยส่วนตรวจสอบความพร้อมของสัญญาณเช่นกัน ดังนั้นในบัสเวอร์ชันปรับปรุงจึงตัดส่วนตรวจสอบความพร้อมของสัญญาณก่อนเริ่มส่งข้อมูลเข้าบัสออก เพื่อลดความซ้ำซ้อนของโครงสร้าง

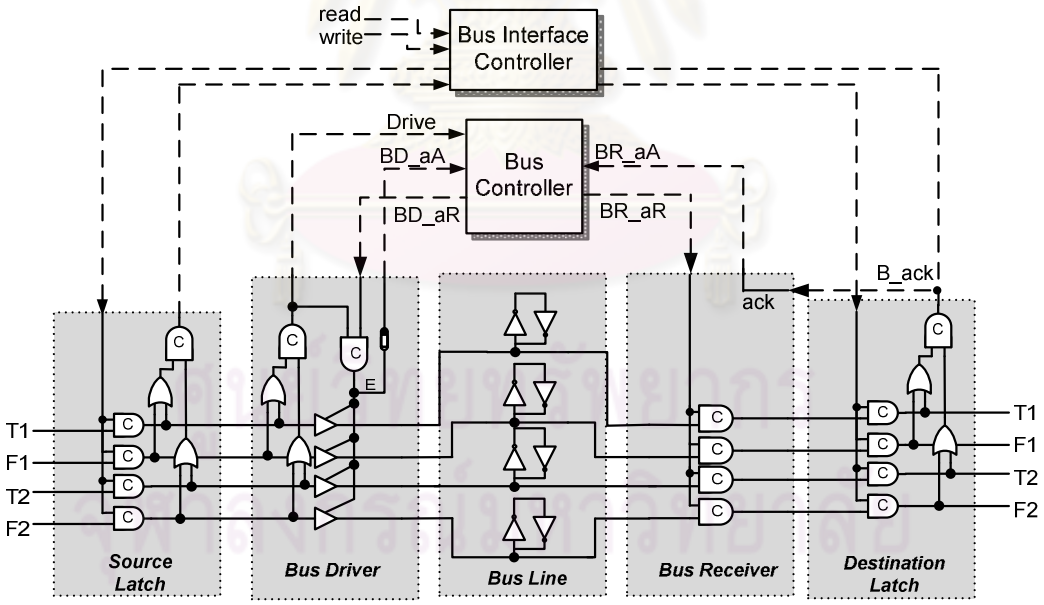
2. สายสัญญาณบัส: การทำงานของลอจิกสามสถานะจากตัวขับบัสทำให้เกิดค่าไฮ-อิมพีแดนซ์เมื่อบัสว่าง แต่ในโพรโทคอลแบบอสมวารใช้ลอจิก 0 และลอจิก 1 ในการติดต่อระหว่างวงจรอสมวารเท่านั้น บัสเวอร์ชันเดิมจึงใช้ Weak Inverter ซึ่งประกอบด้วยเกตผกผัน (Inverter) จำนวนสองตัวต่อกลับหัวกัน บนสายสัญญาณบัสแต่ละเส้น เพื่อเปลี่ยนสถานะไฮ-อิมพีแดนซ์เป็นลอจิก 0 แต่การออกแบบโดยใช้ Weak Inverter ดังกล่าวเป็นการออกแบบในระดับทรานซิสเตอร์ (Transistor Level) ซึ่งโปรแกรม Xilinx ที่งานวิจัยนี้ใช้งาน ออกแบบได้ในระดับเกต (Gate Level) เท่านั้น Weak Inverter จึงถูกตัดออกและแทนที่ด้วยแอนด์เกต (And-gate) ซึ่งเป็นการออกแบบระดับเกตแทน

เมื่อต่อขาที่หนึ่งของแอนด์เกตไว้บนสายสัญญาณบัส และต่อขาที่สองของแอนด์เกตร่วมกับขาสวิทช์ (Enable) ของลอจิกสามสถานะ ได้ดังรูปที่ 3.2(ค) ซึ่งหากพิจารณาจากบัสทิศทางเดียวซึ่งประกอบด้วยลอจิกสามสถานะตัวเดียว เมื่อขาสวิทช์ของลอจิกสามสถานะปิด (Enable=0) ซึ่งเป็นปัญหาคือทำให้เกิดค่าไฮ-อิมพีแดนซ์ที่เอาท์พุทของลอจิกสามสถานะ ขาสองของแอนเกตที่ต่อร่วมกับขาสวิทช์ของลอจิกสามสถานะ จะมีค่าลอจิกเท่ากับ 0 (เท่ากับEnable) เช่นกัน ดังนั้นแอนเกตจะรับค่าไฮ-อิมพีแดนซ์ที่เข้ามาทางขาหนึ่ง และเปลี่ยนเป็นค่าลอจิก 0 ออกไปทางขาเอาท์พุทของแอนเกต

ในทางปฏิบัติ บัสต้องรับส่งข้อมูลได้สองทิศทาง ซึ่งโครงสร้างบัสสองทิศทางจะประกอบด้วยลอจิกสามสถานะสองตัวต่อร่วมกัน เพื่อทำหน้าที่เป็นตัวขับบัสโดยให้สัญญาณจากสองทิศทาง คือทิศทางซ้ายและทิศทางขวา ผ่านเข้าสู่บัส ดังนั้นต้องเพิ่มแอนเกตเป็นสองตัวตามจำนวนของลอจิกสามสถานะ เพื่อแก้ปัญหาสถานะไฮ-อิมพีแดนซ์เช่นกัน โดยเป็นดังรูปที่ 3.2(ง) และรูปที่ 3.2(จ)

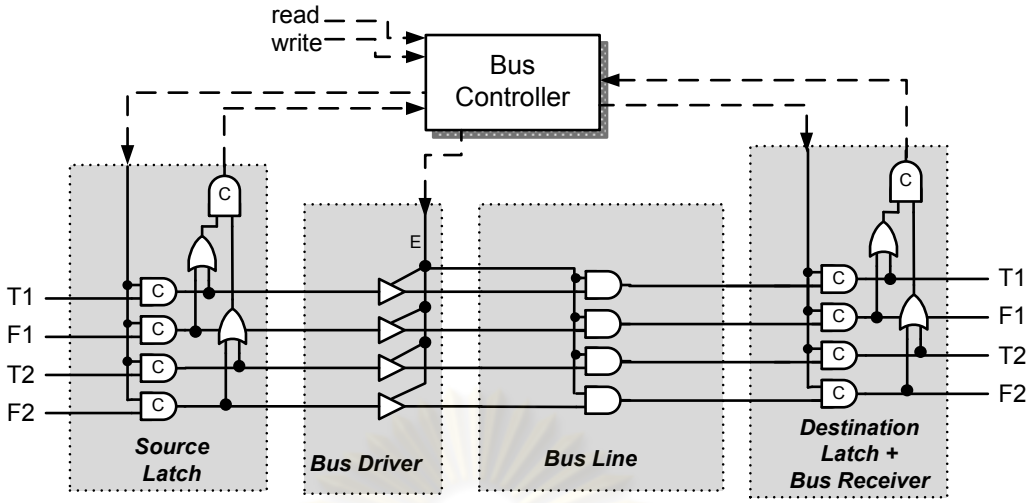
นอกจากนี้อาจต่อตัวต้านทานพูลดาวน์ (Pull-down Resistor) แทนการต่อแอนด์เกตบนสายสัญญาณบัสได้ เนื่องจากสามารถเปลี่ยนสถานะไฮ-อิมพีแดนซ์เป็นลอจิก 0 ได้เช่นกัน อีกทั้งมีขนาดเล็กกว่าแอนด์เกต แต่การออกแบบบัสซึ่งประกอบด้วยตัวต้านทานพูลดาวน์ดังกล่าวจะไม่สามารถทดลองบนเอฟพีจีเอ ที่ประกอบด้วยตัวต้านทานพูลอัพ (Pull-up Resistor) และพูลดาวน์เป็นโครงสร้างหลักภายในวงจร I/O Block ของเอฟพีจีเออยู่แล้วได้ ดังนั้นการออกแบบบัสระบบและทดลองผ่านเอฟพีจีเอ Xilinx SPARTAN 3E ในงานวิจัยนี้ จึงเลือกใช้แอนด์เกตต่อบนสายสัญญาณบัส

3. ตัวรับบัส: บัสเวอร์ชันเดิมใช้อุปกรณ์ชนิดซีเป็นตัวรับบัส โดยอุปกรณ์ชนิดซีจะเก็บค่าที่ส่งผ่านบัสไว้เพื่อส่งต่อไปยังปลายทางของบัส ซึ่งสามารถตัดอุปกรณ์ชนิดซีนี้ออกได้ เนื่องจากบัสต้องใช้งานควบคู่กับส่วนพักข้อมูลปลายทางของบัส (Destination Latch) ซึ่งทำหน้าที่เก็บข้อมูลที่บัสส่งเข้ามาเพื่อส่งต่อไปยังปลายทางเช่นกัน ส่วนพักข้อมูลดังกล่าวประกอบด้วยอุปกรณ์ชนิดซีเช่นกัน ดังนั้นในบัสเวอร์ชันปรับปรุงจึงตัดอุปกรณ์ชนิดซีในตัวรับบัสออก เพื่อลดความซ้ำซ้อนของโครงสร้าง

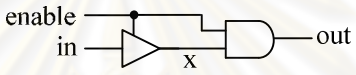


(ก) บัสเข้ารหัสรางคู่ทิศทางเดียวเวอร์ชันเดิมขนาด 4 บิต และส่วนพักข้อมูล

รูปที่ 3.2 การปรับปรุงโครงสร้างบัส

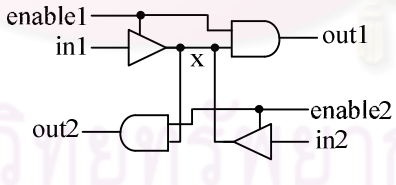


(ข) บัสเข้ารหัสรางคู่ทิศทางเดียวเวอร์ชันปรับปรุงขนาด 4 บิต และส่วนพักข้อมูล



Input		Inout	Output
enable	*in*	*x*	*out*
0	0	hi-Z	0
0	1	hi-Z	0
1	0	0	0
1	1	1	1

(ค) บัสทิศทางเดียวเวอร์ชันปรับปรุง และตารางค่าความจริง

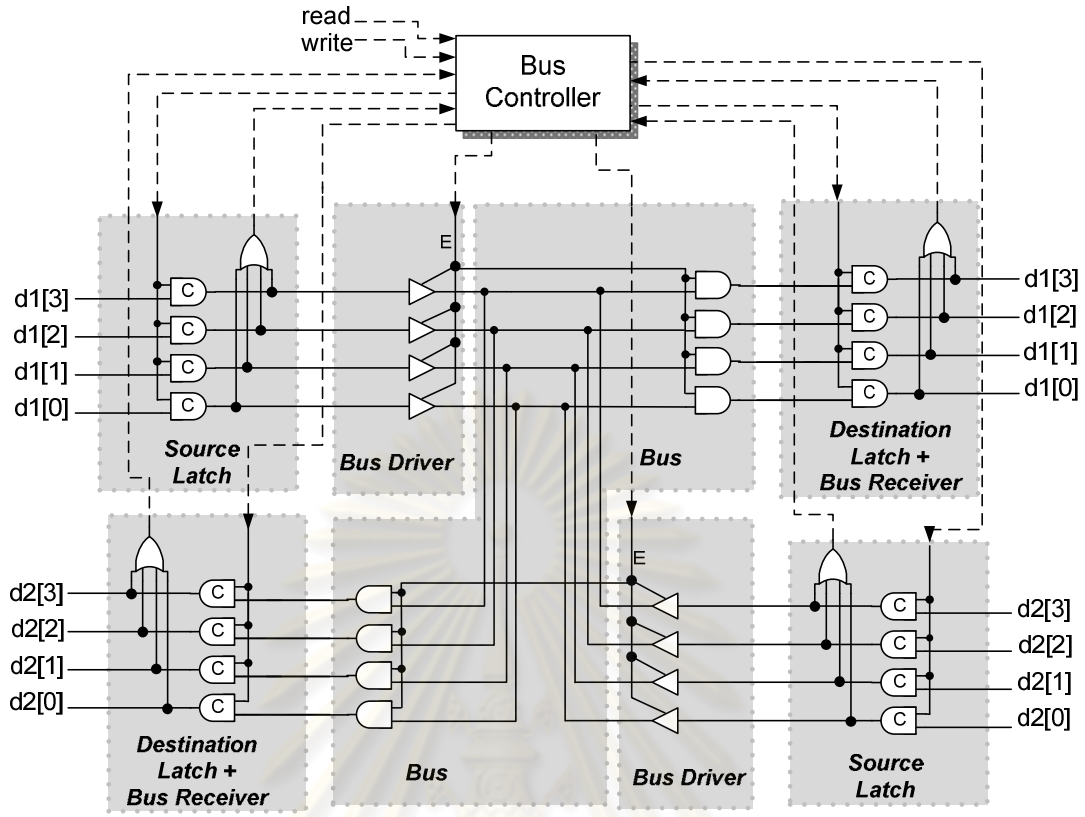


Input				Inout	Output	
enable1	*in1*	*enable2*	*in2*	*x*	*out1*	*out2*
0	0	0	0	hi-Z	0	0
0	1	0	1	hi-Z	0	0
1	0’	0	0’’	0’	0’	0
1	1’	0	1’’	1’	1’	0
0	0’	1	0’’	0’’	0	0’’
0	1’	1	1’’	1’’	0	1’’

หมายเหตุ: (’) หมายถึงค่าลอจิกจาก in1, (“) หมายถึงค่าลอจิกจาก in2

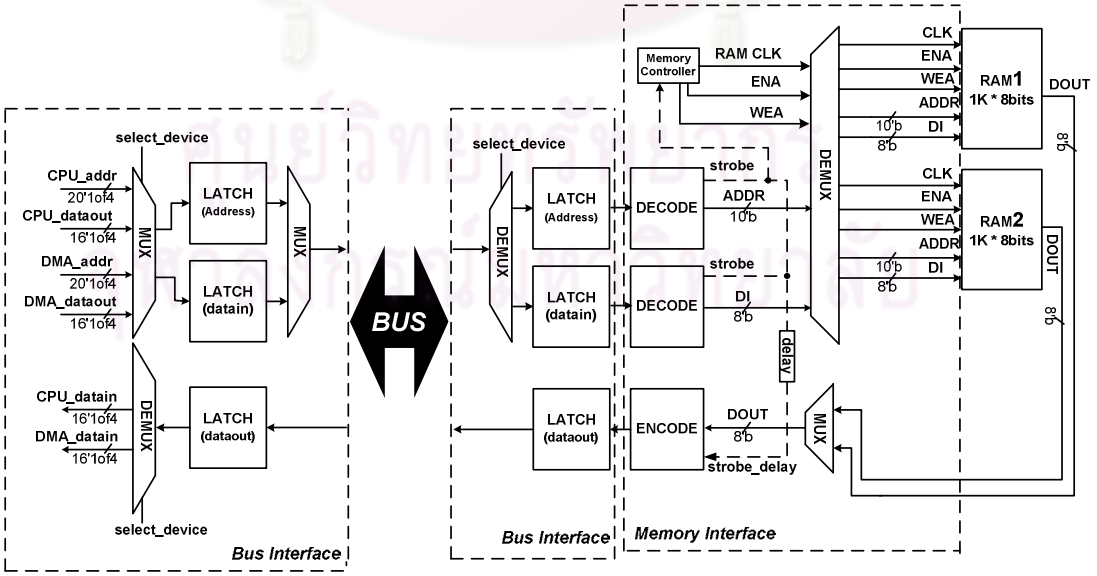
(ง) บัสสองทิศทางเวอร์ชันปรับปรุง และตารางค่าความจริง

รูปที่ 3.2 การปรับปรุงโครงสร้างบัส (ต่อ)



(จ) บัสเข้ารหัสหนึ่งในสี่สองทิศทางขนาด 4 บิต และส่วนพักข้อมูล

รูปที่ 3.2 การปรับปรุงโครงสร้างบัส (ต่อ)



รูปที่ 3.3 ส่วนติดต่อของบัส

3.2.2 ส่วนติดต่อของบัส

ส่วนติดต่อของบัส ใช้ติดต่อระหว่างบัสกับอุปกรณ์ที่ใช้บัส คือ ไมโครโพรเซสเซอร์ ดีเอ็มเอ และหน่วยความจำ ดังรูปที่ 3.3 ฝั่งซ้ายของบัสคือส่วนติดต่อของบัสกับไมโครโพรเซสเซอร์และดีเอ็มเอ ฝั่งขวาของบัสคือส่วนติดต่อของบัสกับหน่วยความจำแบบสมวาร มีรายละเอียดการออกแบบดังนี้

- **ส่วนติดต่อกับไมโครโพรเซสเซอร์และดีเอ็มเอ**

ส่วนติดต่อของบัสกับไมโครโพรเซสเซอร์และดีเอ็มเอ ประกอบด้วย

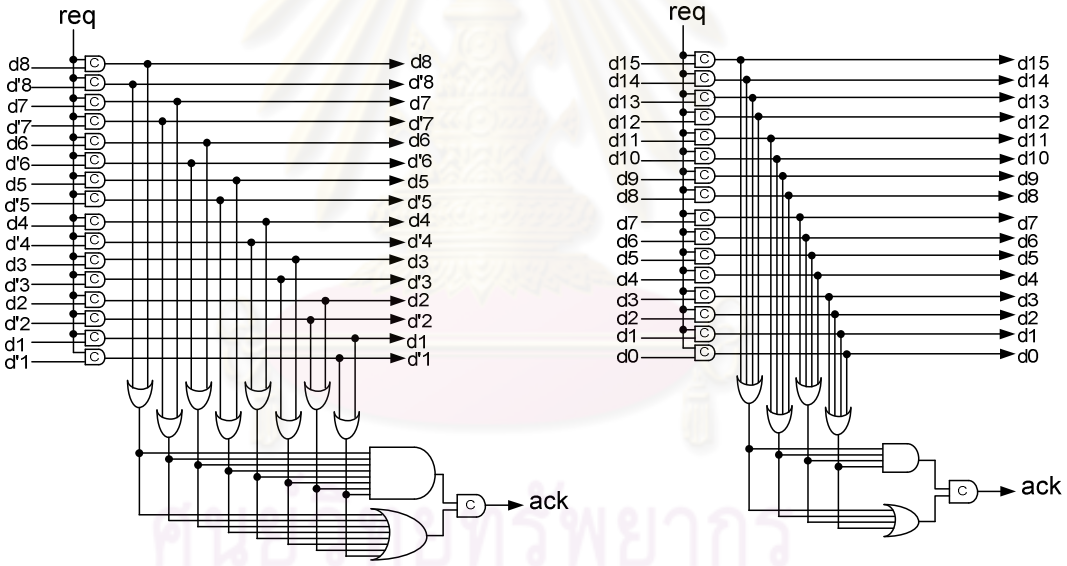
1. ตัวเลือกรวมสัญญาณ หรืออุปกรณ์สหสัญญาณ (Multiplexer) : ทำหน้าที่เลือกสัญญาณจากไมโครโพรเซสเซอร์ หรือดีเอ็มเอเข้าสู่บัส ตัวเลือกรวมสัญญาณมีโครงสร้างดังรูปที่ 3.4(ก) เมื่อมีสัญญาณเลือกข้อมูลอินพุท A (select A = 1) อินพุท A จะถูกเลือกไปเป็นเอาท์พุทของวงจร กล่าวคือ เอาท์พุท C จะมีค่าเท่ากับอินพุท A ในทางกลับกันเมื่อมีสัญญาณเลือกข้อมูลอินพุท B (select B = 1) อินพุท B จะถูกเลือกไปเป็นเอาท์พุทของวงจร กล่าวคือ เอาท์พุท C จะมีค่าเท่ากับอินพุท B

2. ตัวเลือกแยกสัญญาณ (Demultiplexer) : ทำหน้าที่เลือกสัญญาณออกจากบัสไปยังไมโครโพรเซสเซอร์ หรือดีเอ็มเอ ตัวเลือกแยกสัญญาณมีโครงสร้างดังรูปที่ 3.4(ข) เมื่อมีสัญญาณเลือกข้อมูลเอาท์พุท A (select A = 1) อินพุท C จะถูกเลือกไปเป็นเอาท์พุทของขา A กล่าวคือเอาท์พุท A จะมีค่าเท่ากับอินพุท C ในทางกลับกันเมื่อมีสัญญาณเลือกข้อมูลเอาท์พุท B (select B = 1) อินพุท C จะถูกเลือกไปเป็นเอาท์พุทของขา B กล่าวคือเอาท์พุท B จะมีค่าเท่ากับอินพุท C

(ก) ตัวเลือกรวมสัญญาณ (ข) ตัวเลือกแยกสัญญาณ

รูปที่ 3.4 ตัวรวมสัญญาณและตัวแยกสัญญาณ

3. ส่วนพักข้อมูล: ทำหน้าที่รับข้อมูลและเก็บข้อมูลไว้เพื่อเตรียมส่งต่อไปยังส่วนถัดไป โดยใช้สัญญาณร้องขอและสัญญาณตอบรับในวงจรอสมวารเป็นตัวควบคุมจังหวะการรับส่งข้อมูลในส่วนพักข้อมูล ส่วนพักข้อมูลมีโครงสร้างดังรูปที่ 3.5 [16] หากมีข้อมูลอินพุทเข้าและสัญญาณร้องขอมีค่าเป็น 1 ส่วนพักข้อมูลจะจำข้อมูล เมื่อจำข้อมูลครบแล้วสัญญาณตอบรับจะมีค่าเป็น 1 แต่หากไม่มีข้อมูลเข้าหรือข้อมูลว่างและสัญญาณร้องขอมีค่าเป็น 0 ส่วนพักข้อมูลจะลบข้อมูล เมื่อลบข้อมูลทั้งหมดแล้วสัญญาณตอบรับจะมีค่าเป็น 0 โครงสร้างของส่วนพักข้อมูลสำหรับข้อมูลเข้ารหัสรางคู่แสดงดังรูปที่ 3.5(ก) และสำหรับข้อมูลเข้ารหัสหนึ่งในสี่แสดงดังรูปที่ 3.5(ข) ซึ่งจะสังเกตได้ว่าส่วนพักข้อมูลสำหรับข้อมูลเข้ารหัสรางคู่มีขนาดใหญ่กว่าส่วนพักข้อมูลสำหรับข้อมูลเข้ารหัสหนึ่งในสี่



(ก) ส่วนพักข้อมูลเข้ารหัสรางคู่ขนาด 16 บิต (ข) ส่วนพักข้อมูลเข้ารหัสหนึ่งในสี่ ขนาด 16 บิต

รูปที่ 3.5 ส่วนพักข้อมูล

- **ส่วนติดต่อกับหน่วยความจำแบบสมวาร**

ส่วนติดต่อของบัสกับหน่วยความจำแบบสมวาร แบ่งออกเป็น ส่วนเลือกสัญญาณซึ่งประกอบด้วย ตัวเลือกรวมสัญญาณ ตัวเลือกแยกสัญญาณ และส่วนพักข้อมูล

เช่นเดียวกับส่วนติดต่อของบัสกับไมโครโพรเซสเซอร์และดีเอ็มเอดังเช่นอธิบายข้างต้น และนอกจากนี้ยังประกอบด้วยส่วนสำหรับติดติดต่อกับหน่วยความจำโดยเฉพาะ คือ วงจรเข้ารหัสและถอดรหัสข้อมูล วงจรสร้างสัญญาณควบคุมหน่วยความจำแบบสมวาร และวงจรหน่วงเวลา โดยมีรายละเอียดการออกแบบในส่วนดังกล่าวรวมทั้งตัวหน่วยความจำดังนี้

1. หน่วยความจำ

การออกแบบส่วนติดต่อของบัสกับหน่วยความจำแบบสมวาร ต้องพิจารณาคุณสมบัติของหน่วยความจำที่บัสจะติดต่อด้วย เพื่อสร้างส่วนติดต่อที่สอดคล้องกับการทำงานของหน่วยความจำ โดยหน่วยความจำแบบอสมวารที่ใช้กับบัสระบบนี้สร้างจากโปรแกรมย่อย Core Generator บนโปรแกรม Xlinx ISE 11 มีโครงสร้างดังรูปที่ 3.6 [13] และมีคุณสมบัติเช่นเดียวกับหน่วยความจำที่ใช้กับไมโครโพรเซสเซอร์เข้ารหัสรางคู่ [17] ดังนี้

<u>คุณสมบัติของหน่วยความจำ</u>

i) หน่วยความจำเป็นแบบสมวาร เก็บข้อมูลรหัสฐานสอง (Binary Code) สูงสุดได้ 1K*8bits กล่าวคือมี 2^{10} = 1024 ช่อง (ความกว้างเลขที่อยู่ขนาด 10'b* บิต) แต่ละช่องเก็บข้อมูลได้ช่องละ 8'b บิต (ความกว้างข้อมูลขนาด 8'b บิต)

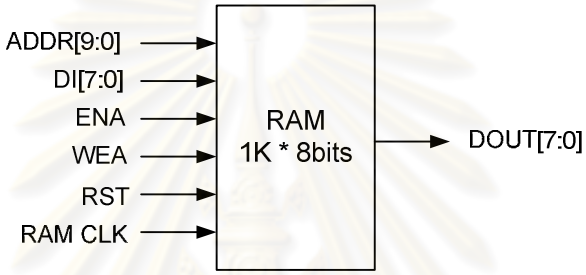
ii) ประกอบด้วยพอร์ทดังรูปที่ 3.6(ก) พอร์ทอินพุทคือ พอร์ท ADDR (Address) ขนาด 10'b บิต พอร์ท DI (Data Input) ขนาด 8'b บิต พอร์ท ENA (Enable) พอร์ท WEA (Write Enable) พอร์ท RST (Reset) พอร์ท CLK (Clock) พอร์ทเอาท์พุทคือ พอร์ท DOUT (Data Output) ขนาด 8'b บิต

iii) ทำงานเมื่อมีสัญญาณนาฬิกา (RAM CLK) ขอบขาขึ้นเข้ามา โดยรองรับการทำงานร่วมกับสัญญาณนาฬิกาความถี่สูงสุดได้ไม่เกิน 200 เมกกะเฮิร์ท (1 Clock Cycle Time $\geq$ 5 ns)

---

* 'b คือ จำนวนบิตของข้อมูลเข้ารหัสฐานสอง

iv) มีโหมดการทำงานเป็นแบบ WRITE FIRST Mode หรือ Transparent Mode ซึ่งมีลักษณะการทำงานคือ ขาเอาท์พุท DOUT ของหน่วยความจำ จะแสดงค่าข้อมูลปัจจุบันที่เก็บไว้ใน ตำแหน่งหน่วยความจำตามที่ขา ADDR ระบุไว้เสมอ และหน่วยความจำจะจำค่าข้อมูลใหม่จะจำค่าข้อมูลใหม่จากขา DI ต่อเมื่อสัญญาณอนุญาตให้เขียนข้อมูลคือ WEA มีค่าเท่ากับ 1 เท่านั้น โดยลักษณะการทำงานที่แสดงข้อมูลปัจจุบันเสมอนี้ ทำให้การตรวจสอบความถูกต้องของค่าภายในหน่วยความจำทำได้ง่าย



รูปที่ 3.6 หน่วยความจำ

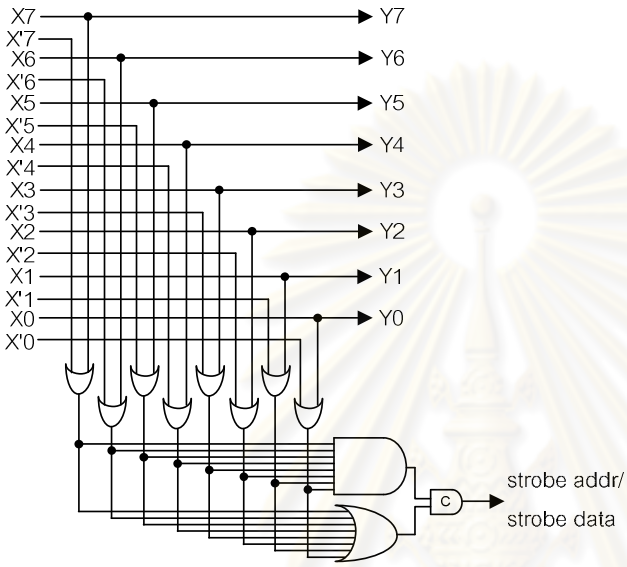
2. วงจรเข้ารหัสและถอดรหัสข้อมูล

การรับส่งข้อมูลจากบัสเข้ารหัสรางคู่เวอร์ชันเดิม ไปที่หน่วยความจำซึ่งเก็บข้อมูลในรูปแบบรหัสฐานสอง จำเป็นต้องมีวงจรถอดรหัส (Decode Circuit) ข้อมูลรางคู่จากบัส เป็นรหัสฐานสองเพื่อใช้บนหน่วยความจำ ดังรูปที่ 3.7(ก) [8] และวงจรเข้ารหัสฐานสองจากหน่วยความจำเป็นรหัสรางคู่เพื่อใช้บนบัส (Encode Circuit) ดังรูปที่ 3.7(ข) [8]

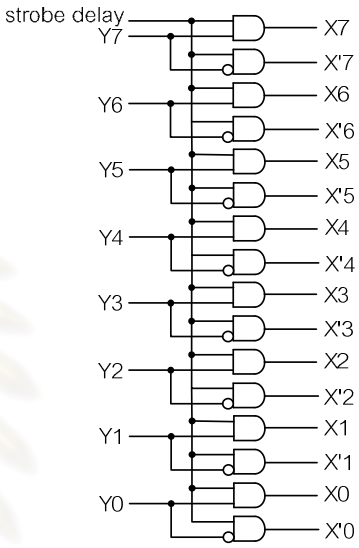
สำหรับเวอร์ชันปรับปรุง การรับส่งข้อมูลจากบัสเข้ารหัสหนึ่งในสี่เวอร์ชันปรับปรุงไปที่หน่วยความจำซึ่งเก็บข้อมูลในรูปแบบรหัสฐานสอง จำเป็นต้องมีวงจรถอดรหัสข้อมูลหนึ่งในสี่จากบัส เป็นรหัสฐานสองเพื่อใช้บนหน่วยความจำดังรูปที่ 3.7(ค) และวงจรเข้ารหัสข้อมูลฐานสองจากหน่วยความจำ เป็นรหัสหนึ่งในสี่เพื่อใช้บนบัสดังรูปที่ 3.7(ง) เช่นกัน

จากรูปที่ 3.7 ให้ค่า X เป็นรหัสรางคู่หรือรหัสหนึ่งในสี่ และให้ค่า Y เป็นรหัสฐานสอง สำหรับวงจรถอดรหัส เมื่อถอดรหัสข้อมูลสมบูรณ์แล้ว สัญญาณเอาท์พุท strobe addr

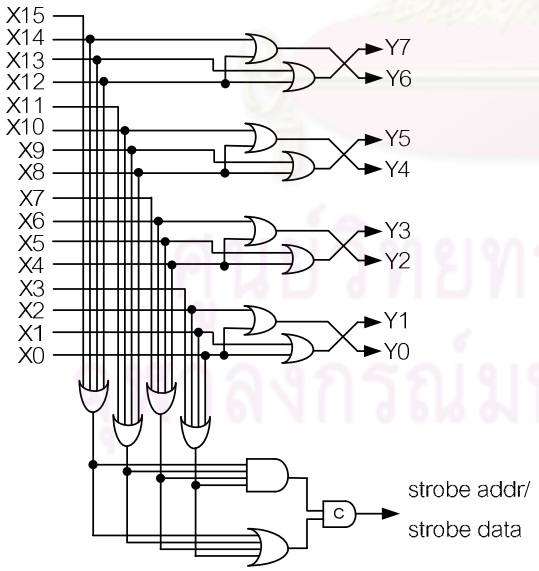
หรือ strobe data จะมีค่าเป็น 1 โดยออกแบบให้สัญญาณ strobe addr เป็นสัญญาณบ่งบอกว่าถอดรหัสเลขที่อยู่จากบัสสมบูรณ์แล้ว ส่วนสัญญาณ strobe data เป็นสัญญาณบ่งบอกว่าถอดรหัสข้อมูลจากบัสสมบูรณ์แล้ว และสำหรับวงจรเข้ารหัส เมื่อข้อมูลมีความสมบูรณ์พร้อมที่จะเข้ารหัสสู่บัส สัญญาณอินพุท strobe delay จะมีค่าเป็น 1



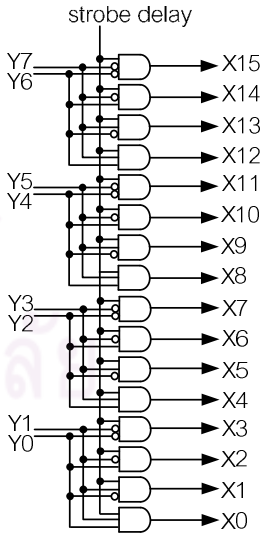
(ก) วงจรถอดรหัสรางคู่ขนาด 16 บิต



(ข) วงจรเข้ารหัสรางคู่ขนาด 16 บิต



(ค) วงจรถอดรหัสหนึ่งในสี่ขนาด 16 บิต



(ง) วงจรเข้ารหัสหนึ่งในสี่ขนาด 16 บิต

รูปที่ 3.7 วงจรเข้ารหัสและวงจรถอดรหัส

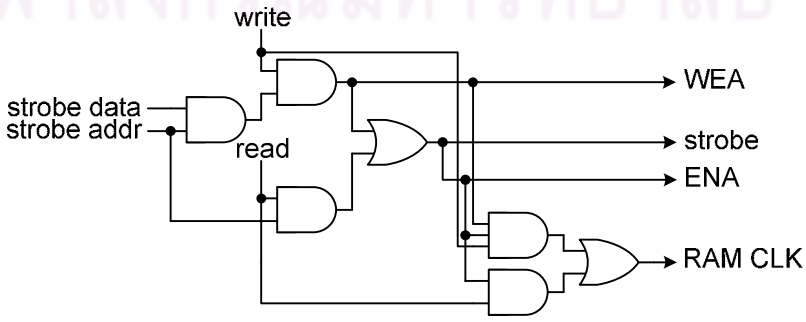
3. วงจรสร้างสัญญาณควบคุมหน่วยความจำแบบสมวาร

วงจรสร้างสัญญาณควบคุมหน่วยความจำแบบสมวาร (Memory Controller) ทำหน้าที่สร้างสัญญาณนาฬิกาและสัญญาณควบคุมคือ ENA และ WEA เข้าสู่หน่วยความจำแบบสมวาร บัสระบบแบบอสมวารจะติดต่อกับหน่วยความจำแบบสมวารได้ ต้องจำลองสัญญาณนาฬิกาจากสัญญาณภายในวงจรอสมวารขึ้นมาเพื่อใช้ติดต่อกับวงจรสมวาร โครงสร้างเป็นดังรูปที่ 3.8 โดยมีหลักการทำงานดังนี้

สำหรับคำสั่งเขียนข้อมูล เมื่อสัญญาณเขียนข้อมูล write เท่ากับ 1 และการถอดรหัสข้อมูลเลขที่อยู่รวมถึงการถอดรหัสข้อมูลมีความสมบูรณ์ คือ สัญญาณอินพุท strobe addr และ strobe data มีค่าเป็น 1 วงจรจะส่งสัญญาณเอาท์พุท RAM_CLK ENA WEA ไปที่หน่วยความจำแบบสมวารเพื่อกระตุ้นให้หน่วยความจำเริ่มต้นทำงานเขียนข้อมูล โดยสัญญาณ RAM_CLK นี้คือสัญญาณนาฬิกาที่จำลองขึ้นมาเพื่อใช้ติดต่อกับหน่วยความจำแบบสมวาร

สำหรับคำสั่งอ่านข้อมูล เมื่อสัญญาณอ่านข้อมูล read เท่ากับ 1 และการถอดรหัสข้อมูลเลขที่อยู่มีความสมบูรณ์ คือ สัญญาณอินพุท strobe addr มีค่าเป็น 1 วงจรจะส่งสัญญาณเอาท์พุท RAM_CLK ENA ไปที่หน่วยความจำแบบสมวารเพื่อกระตุ้นให้หน่วยความจำเริ่มต้นทำงานอ่านข้อมูล

นอกจากนี้วงจรจะส่งสัญญาณเอาท์พุท strobe ไปที่วงจรหน่วงเวลาเพื่อใช้เป็นสัญญาณเริ่มต้นการทำงาน ซึ่งจะอธิบายในหัวข้อถัดไป

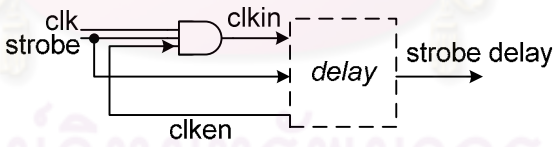


รูปที่ 3.8 วงจรสร้างสัญญาณควบคุมหน่วยความจำแบบสมวาร

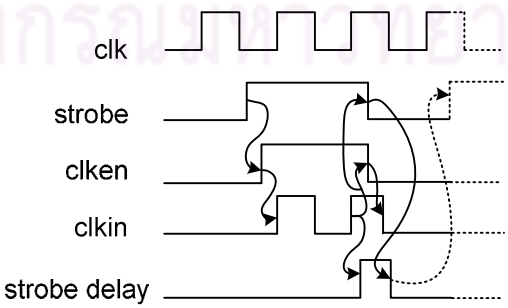
4. วงจรหน่วงเวลา

วงจรเข้ารหัสฐานสองเป็นรหัสหนึ่งในสี่ จะรับค่ารหัสฐานสองจากหน่วยความจำมาเข้ารหัส ซึ่งหากข้อมูลที่รับเข้ามาเข้ารหัสยังไม่สมบูรณ์ กล่าวคือเป็นข้อมูลที่หน่วยความจำยังเขียนหรืออ่านไม่เสร็จ จะทำให้การเข้ารหัสข้อมูลเกิดความผิดพลาด และส่งต่อค่าผิดพลาดดังกล่าวไปยังบัสและส่วนอื่นๆ วงจรหน่วงเวลาสามารถแก้ปัญหานี้ได้โดยหน่วงเวลารอให้หน่วยความจำเขียนหรืออ่านข้อมูลเสร็จสมบูรณ์ก่อน (Delay Time > Memory Access Time) [8] แล้วจึงค่อยส่งสัญญาณ strobe delay ซึ่งเป็นสัญญาณบ่งบอกว่าข้อมูลมีความพร้อมที่จะรับการเข้ารหัสแล้ว ไปที่วงจรเข้ารหัสเพื่อเริ่มต้นเข้ารหัสข้อมูล

วงจรหน่วงเวลาออกแบบโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณดังรูปที่ 3.9 โดยมีโครงร่างวงจรดังรูปที่ 3.9(ก) และมีพฤติกรรมวงจรดังรูปที่ 3.9(ข) โดยเมื่อมีค่าอินพุท strobe เท่ากับ 1 ซึ่งหมายถึงข้อมูลถูกถอดรหัสเข้าหน่วยความจำโดยสมบูรณ์และหน่วยความจำได้รับการกระตุ้นให้เริ่มต้นทำงาน วงจรจะหน่วงเวลา 1 ลูกคลื่นสัญญาณนาฬิกาที่ความถี่ไม่เกิน 400 เมกกะเฮิร์ท (1 Clock Cycle Time = ½ RAM Clock Cycle Time ≥ 2.5ns) เพื่อรอเวลา

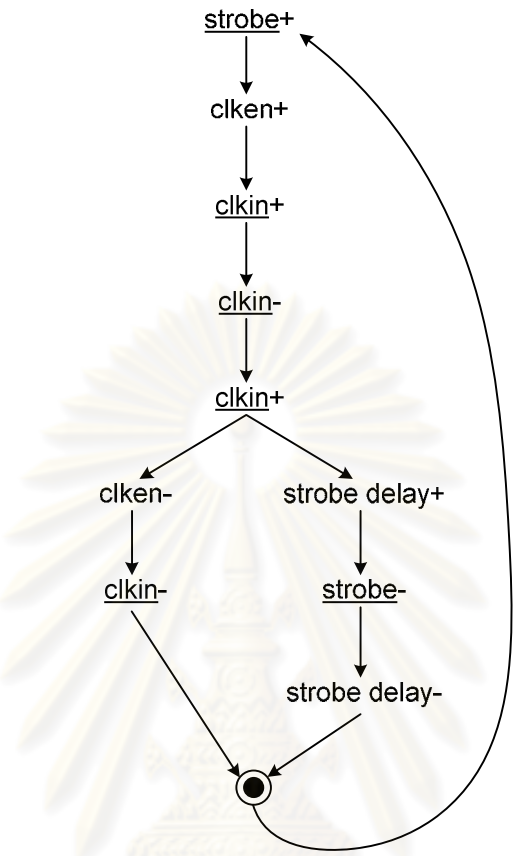


(ก) โครงร่างวงจร



(ข) แผนผังแสดงการเปลี่ยนแปลงสถานะลอจิกของอินพุทและเอาท์พุท ในแต่ละช่วงเวลา

รูปที่ 3.9 การออกแบบวงจรหน่วงเวลาโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ



(ค) กราฟบรรยายการเปลี่ยนสัญญาณ

รูปที่ 3.9 การออกแบบวงจรหน่วงเวลาโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ (ต่อ)

อย่างน้อย 5 ns ให้หน่วยความจำเขียนหรืออ่านข้อมูลจนเสร็จสมบูรณ์ แล้วจึงให้ค่าเอาท์พุท strobe delay เท่ากับ 1 เพื่อเข้ารหัสข้อมูลเข้าบัสต่อไป กราฟบรรยายการเปลี่ยนสัญญาณจากพฤติกรรมนี้แสดงดังรูปที่ 3.9 (ค)

หลังจากสร้างกราฟบรรยายการเปลี่ยนสัญญาณเสร็จสิ้นแล้ว ขั้นตอนของการสร้างวงจรต่อไปคือ สร้างกราฟแสดงสถานะ การพิจารณาค่าสัญญาณที่ได้จากกราฟแสดงสถานะลงในแผนที่คาร์นอฟ และการลดทอนสมการลอจิกโดยใช้ทฤษฎีบูลีน สามารถใช้โปรแกรม Petrify ช่วยในขั้นตอนเหล่านี้ได้ โดยโปรแกรม Petrify จะตรวจสอบและลดความซ้ำซ้อนของสถานะสัญญาณต่างๆในกราฟบรรยายการเปลี่ยนสัญญาณที่ออกแบบไว้ รวมถึงลดรูปสมการบูลีนด้วย ดาวน์โหลดโปรแกรมดังกล่าวได้ที่ http://www.lsi.upc.edu/~jordicf/petrify และดาวโหลดคู่มือ

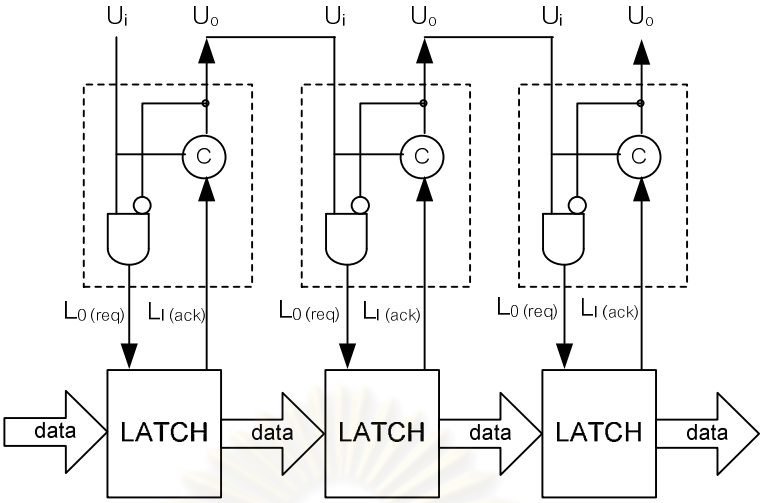
การใช้โปรแกรมได้ที่ http://www.cs.unc.edu/~montek/teaching/spring-04/petrify-tutorial.ps วิธีการใช้โปรแกรม Petrify ออกแบบวงจรหน่วงเวลา จะอธิบายในบทที่ 6 ต่อไป

3.2.3 ตัวควบคุมบัส

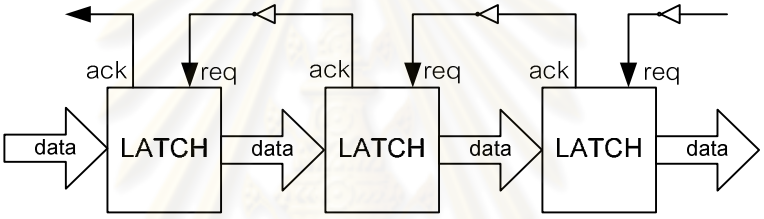
สัญญาณร้องขอและสัญญาณตอบรับในบัสระบบเวอร์ชันเดิม ถูกควบคุมด้วยอุปกรณ์ออโต้สวีป (Autosweeping Module : ASM) ดังรูปที่ 3.10(ก) [16] โดยใช้สัญญาณอาณัติแบบ 4 ขั้น สำหรับติดต่อรับส่งข้อมูลในแต่ละส่วน ซึ่งมีหลักการคือเมื่อตัวส่งข้อมูลต้องการส่งข้อมูล สัญญาณ Ui จะเปลี่ยนจาก 0 เป็น 1 เพื่อให้สัญญาณร้องขอของตัวรับข้อมูล คือ L0 เปลี่ยนจาก 0 เป็น 1 เพื่อเข้าสู่ขั้นทำงาน จากนั้นเมื่อตัวรับข้อมูลรับข้อมูลเสร็จสมบูรณ์ สัญญาณตอบรับของตัวรับข้อมูลคือ Li จะเปลี่ยนจาก 0 เป็น 1 ส่งผลให้ L0 เปลี่ยนจาก 1 เป็น 0 เพื่อเข้าสู่ขั้นว่าง จะเห็นได้ว่าการรับส่งข้อมูลจะมีการทำงานในขั้นทำงานและขั้นว่างสลับกัน ซึ่งเป็นไปตามรูปแบบของสัญญาณอาณัติแบบ 4 ขั้น

สัญญาณร้องขอและสัญญาณตอบรับในบัสระบบเวอร์ชันปรับปรุง ถูกควบคุมด้วยอุปกรณ์สายท่อ หรืออุปกรณ์ไปป์ไลน์ (Asynchronous Pipeline Module) [16] ดังรูปที่ 3.10(ข) ซึ่งใช้กับสัญญาณอาณัติแบบ 4 ขั้นเช่นกัน แต่มีขนาดวงจรเล็กกว่าอุปกรณ์ออโต้สวีป และใช้สัญญาณควบคุมน้อยกว่าอุปกรณ์ออโต้สวีปอีกด้วย อุปกรณ์ไปป์ไลน์มีหลักการคือ เมื่อตัวส่งข้อมูลต้องการส่งข้อมูล สัญญาณร้องขอของตัวรับข้อมูล คือ req จะเปลี่ยนจาก 0 เป็น 1 เพื่อเข้าสู่ขั้นทำงาน จากนั้นเมื่อตัวรับข้อมูลรับข้อมูลเสร็จสมบูรณ์ สัญญาณตอบรับของตัวรับข้อมูลคือ ack จะเปลี่ยนจาก 0 เป็น 1 ส่งผลให้ req เปลี่ยนจาก 1 เป็น 0 เพื่อเข้าสู่ขั้นว่าง

เส้นทางการรับส่งข้อมูลภายในบัส ถูกควบคุมโดยวงจรควบคุมซึ่งสร้างจากกราฟบรรยายการเปลี่ยนสัญญาณ และใช้โปรแกรม Petrify สร้างวงจร เช่นเดียวกับการออกแบบวงจรหน่วงเวลาที่ได้กล่าวไปแล้ว



(ก) Autosweeping Module : ASM



(ข) Asynchronous Pipeline Module

รูปที่ 3.10 ส่วนควบคุมการเปลี่ยนระดับสัญญาณร้องขอและสัญญาณตอบรับ

3.3 การทำงานของบัสระบบ

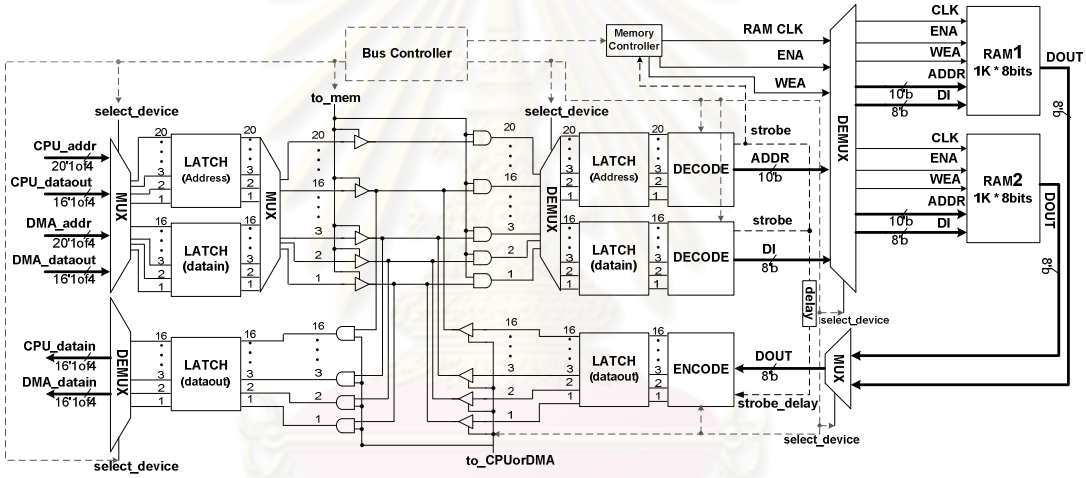
บัสระบบเข้ารหัสหนึ่งในสี่เวอร์ชันปรับปรุง มีโครงสร้างดังรูปที่ 3.11 โดยบัสระบบเชื่อมต่อไมโครโพรเซสเซอร์ ดีเอ็มเอ และหน่วยความจำเข้าด้วยกัน เพื่อรับส่งข้อมูลระหว่างอุปกรณ์ทั้ง 3 ตามคำสั่งที่ได้รับ คำสั่งที่เรียกใช้งานบัสระบบมีดังนี้

- คำสั่งโหลด (Load): ไมโครโพรเซสเซอร์จะส่งสัญญาณอ่าน และเลขที่อยู่ ผ่านบัสไปยังหน่วยความจำ เมื่ออ่านค่าจากหน่วยความจำเสร็จสิ้นจะส่งค่าที่อ่านได้ผ่านบัสกลับมายังไมโครโพรเซสเซอร์
- คำสั่งสโตร์ (Store): ไมโครโพรเซสเซอร์จะส่งสัญญาณเขียน เลขที่อยู่ และข้อมูล ผ่านบัสไปยังหน่วยความจำ เพื่อเขียนค่าลงหน่วยความจำ

- คำสั่งอิน (IN): แบ่งออกเป็นคำสั่งย่อยที่เรียกใช้งานบัสระบบคือ

- IN @K2, @K : ดีเอ็มเอจะส่งสัญญาณอ่าน และเลขที่อยู่ตำแหน่งที่ต้องการอ่าน ผ่านบัสไปยังหน่วยความจำต้นทาง เมื่ออ่านค่าจากหน่วยความจำต้นทางเสร็จสิ้นจะส่งค่าที่อ่านได้ผ่านบัสกลับมายังดีเอ็มเอ จากนั้นดีเอ็มเอจะส่งสัญญาณเขียน เลขที่อยู่ตำแหน่งที่ต้องการเขียน และข้อมูลที่อ่านได้ดังกล่าว ผ่านบัสไปยังหน่วยความจำปลายทาง เพื่อเขียนค่าลงหน่วยความจำปลายทาง

- IN I/O,@K : ดีเอ็มเอจะส่งสัญญาณอ่าน และเลขที่อยู่ตำแหน่งที่ต้องการอ่าน ผ่านบัสไปยังหน่วยความจำต้นทาง เมื่ออ่านค่าจากหน่วยความจำต้นทางเสร็จสิ้นจะส่งค่าที่อ่านได้ผ่านบัสกลับมายังดีเอ็มเอ เพื่อส่งต่อไปยังอุปกรณ์ต่อพ่วงปลายทาง



รูปที่ 3.11 บัสระบบเข้ารหัสหนึ่งในสี่ขนาด 10 บิต

- คำสั่งเอาท์ (OUT): แบ่งออกเป็นคำสั่งย่อยที่เรียกใช้งานบัสระบบคือ

- OUT @K, @K2 : ดีเอ็มเอจะส่งสัญญาณอ่าน และเลขที่อยู่ตำแหน่งที่ต้องการอ่าน ผ่านบัสไปยังหน่วยความจำต้นทาง เมื่ออ่านค่าจากหน่วยความจำต้นทางเสร็จสิ้นจะส่งค่าที่อ่านได้ผ่านบัสกลับมายังดีเอ็มเอ จากนั้นดีเอ็มเอจะส่งสัญญาณเขียน เลขที่อยู่ตำแหน่งที่ต้องการเขียน และข้อมูลที่อ่านได้ดังกล่าว ผ่านบัสไปยังหน่วยความจำปลายทาง เพื่อเขียนค่าลงหน่วยความจำปลายทาง

- OUT I/O,@K : ดีเอ็มเอจะรับค่าจากอุปกรณ์ต่อพ่วงไว้ และส่งสัญญาณเขียน เลขที่อยู่ตำแหน่งที่ต้องการเขียน และข้อมูลที่รับจากอุปกรณ์ต่อพ่วงดังกล่าว ผ่านบัสไปยังหน่วยความจำ เพื่อเขียนค่าลงหน่วยความจำ

จะเห็นได้ว่างานหลักของบัสระบบคือ อ่านข้อมูลจากหน่วยความจำและส่งต่อกับเขียนข้อมูลที่หน่วยความจำ การทำงานอ่านเขียนข้อมูลของบัสระบบดังกล่าว มีขั้นตอนการทำงานดังต่อไปนี้

<u>การอ่านข้อมูลของบัสระบบ</u>

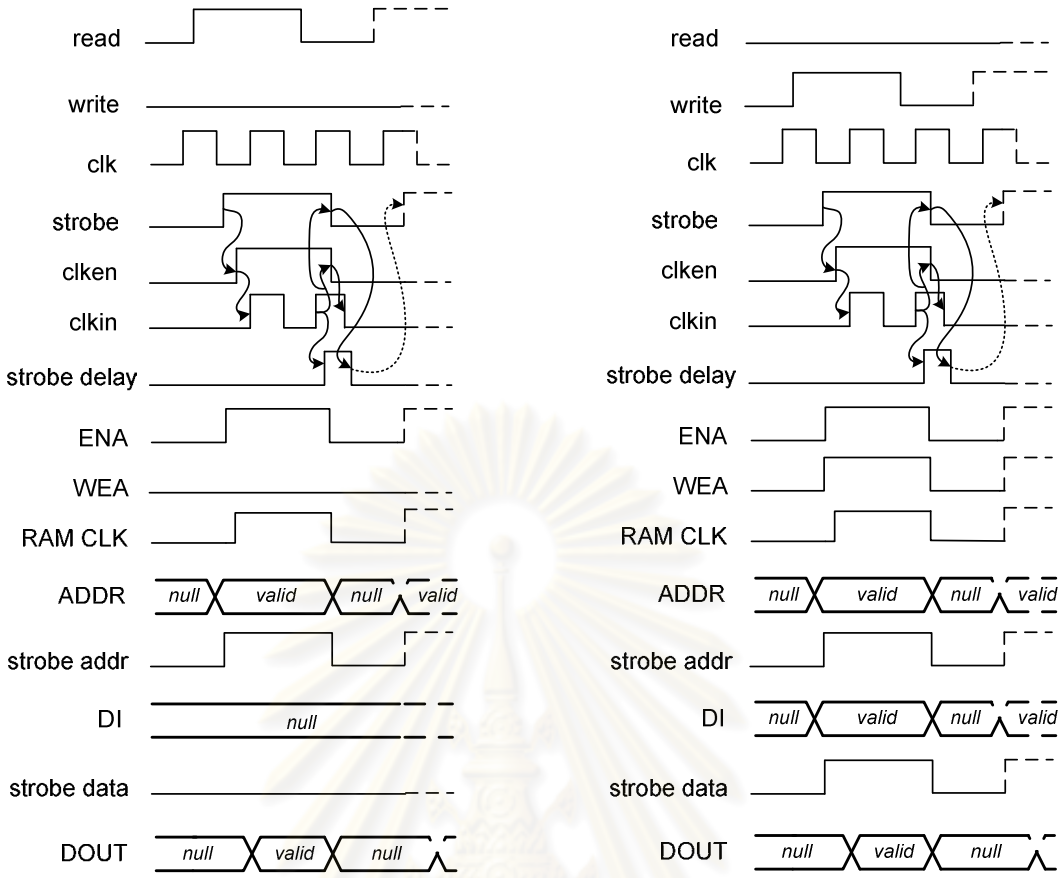
1. เมื่ออุปกรณ์ได้สิทธิ์ใช้งานบัสระบบ เส้นทางการรับส่งข้อมูลภายในบัสจะถูกกำหนดให้เชื่อมต่อระหว่างตัวส่งข้อมูลและตัวรับข้อมูลตามคำสั่งที่ได้รับ เมื่อบัสได้รับสัญญาณอ่านข้อมูล และเลขที่อยู่ ครบแล้วจะเข้าสู่ขั้นทำงาน โดยจะส่งค่าเลขที่อยู่จากตัวส่งข้อมูลผ่านบัสไปยังหน่วยความจำ

2. ส่วนติดต่อหน่วยความจำจะถอดรหัสค่าเลขที่อยู่จากบัสจนสมบูรณ์ (strobe addr = strobe = 1) แล้วจึงส่งสัญญาณควบคุมซึ่งประกอบด้วย สัญญาณเปิดการทำงานคือ ENA สัญญาณนาฬิกา คือ RAM_CLK และเลขที่อยู่ที่ถอดรหัสแล้วคือ ADDR ไปยังหน่วยความจำ

3. เมื่อหน่วงเวลารอให้อ่านข้อมูลจากหน่วยความจำเสร็จสิ้น (strobe delay = 1) ส่วนติดต่อหน่วยความจำจะเข้ารหัสข้อมูลที่อ่านได้จากหน่วยความจำคือ DOUT จนสมบูรณ์ และส่งต่อข้อมูลดังกล่าวผ่านบัสไปยังตัวรับข้อมูล จากนั้นสัญญาณควบคุมทั้งหมดจะเปลี่ยนเป็น 0 เพื่อเข้าสู่ขั้นว่าง

<u>การเขียนข้อมูลของบัสระบบ</u>

1. เมื่ออุปกรณ์ได้สิทธิ์ใช้งานบัสระบบ เส้นทางการรับส่งข้อมูลภายในบัสจะถูกกำหนดให้เชื่อมต่อระหว่างตัวส่งข้อมูลและตัวรับข้อมูลตามคำสั่งที่ได้รับ เมื่อบัสได้รับสัญญาณอ่านข้อมูล เลขที่อยู่ และข้อมูลที่ต้องการเขียนครบแล้วจะเข้าสู่ขั้นทำงาน โดยจะส่งค่าเลขที่อยู่และค่าข้อมูลจากตัวส่งข้อมูล ผ่านบัสไปยังหน่วยความจำ



(ก) อ่านข้อมูล (ข) เขียนข้อมูล

รูปที่ 3.12 แผนผังแสดงการเปลี่ยนแปลงสถานะลอจิกของส่วนติดต่อหน่วยความจำในคำสั่งอ่านและเขียนข้อมูล

2. ส่วนติดต่อหน่วยความจำจะถอดรหัสค่าเลขที่อยู่และค่าข้อมูลจากบัสจนสมบูรณ์ (strobe addr * strobe data = strobe = 1) แล้วจึงส่งสัญญาณควบคุมซึ่งประกอบด้วยสัญญาณอนุญาตเขียนข้อมูลคือ WEA สัญญาณเปิดการทำงานคือ ENA สัญญาณนาฬิกา คือ RAM_CLK เลขที่อยู่ที่ถอดรหัสแล้วคือ ADDR และข้อมูลที่ถอดรหัสแล้วคือ DI ไปยังหน่วยความจำ

3. เมื่อหน่วงเวลารอให้เขียนข้อมูลบนหน่วยความจำเสร็จสิ้น (strobe delay = 1) สัญญาณควบคุมทั้งหมดจะเปลี่ยนเป็น 0 เพื่อเข้าสู่ขั้นว่าง

บทที่ 4

การออกแบบดีเอ็มเอแบบอสมวาร

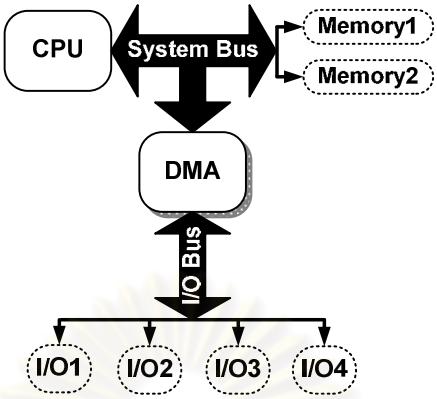
ในบทนี้กล่าวถึงการออกแบบดีเอ็มเอ ซึ่งทำงานร่วมกับบัสระบบเพื่อช่วยงานไมโครโพรเซสเซอร์ในการติดต่อกับหน่วยความจำและอุปกรณ์ต่อพ่วง แบ่งออกเป็น คุณสมบัติของดีเอ็มเอ โครงสร้างของดีเอ็มเอ บัสอุปกรณ์ต่อพ่วง และการทำงานของดีเอ็มเอ โดยมีรายละเอียดดังต่อไปนี้

4.1 คุณสมบัติของดีเอ็มเอ

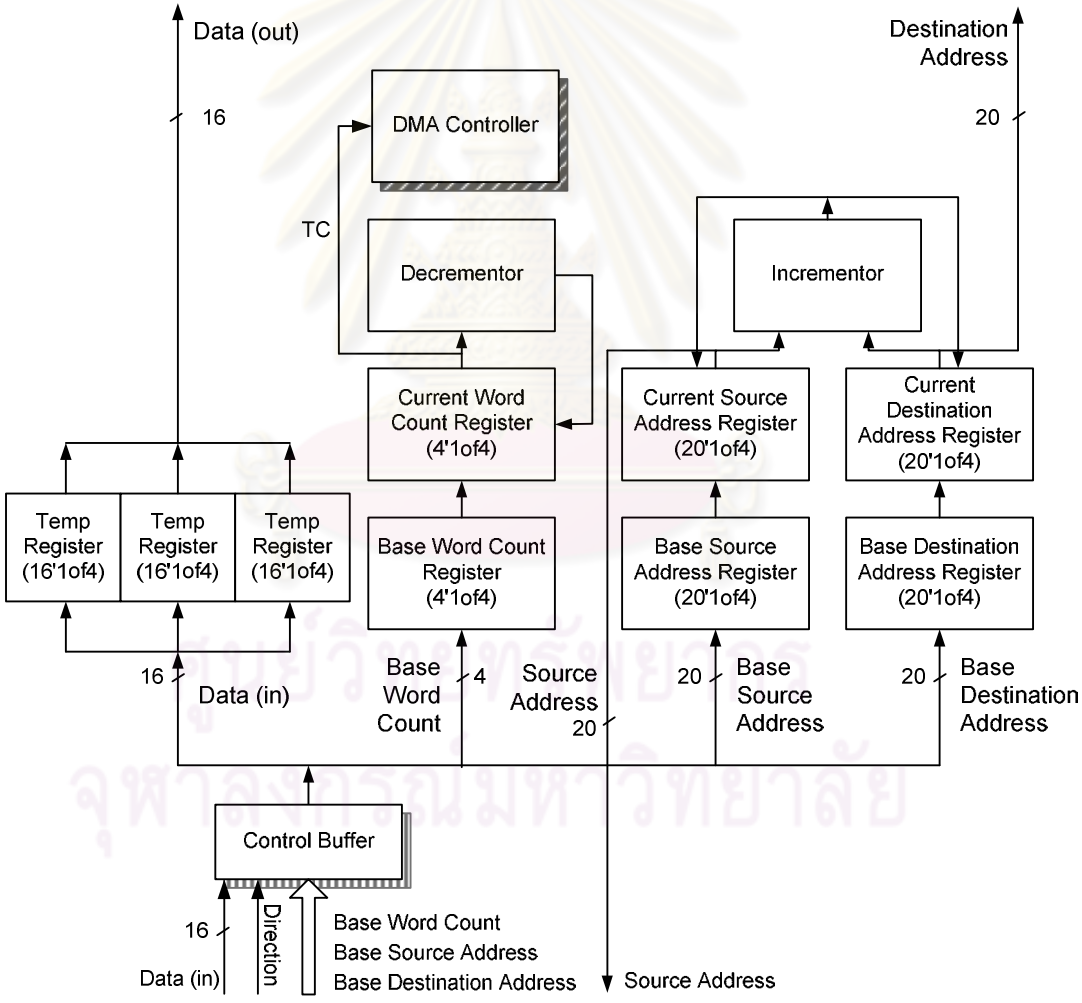
ดีเอ็มเอแบบอสมวารที่ออกแบบ มีคุณสมบัติโดยสรุปดังนี้

- สามารถรับส่งข้อมูลระหว่าง หน่วยความจำ และหน่วยความจำ, หน่วยความจำ และอุปกรณ์ต่อพ่วง, อุปกรณ์ต่อพ่วง และอุปกรณ์ต่อพ่วง เช่นเดียวกับดีเอ็มเอแบบอสมวารเวอร์ชันเดิม [4]
- รับส่งข้อมูลเข้ารหัสหนึ่งในสี่ได้สูงสุดครั้งละ 48 บิต และต่ำสุดครั้งละ 16 บิต เพื่อเพิ่มจำนวนการรับส่งข้อมูลจากดีเอ็มเอเวอร์ชันเดิม ที่รับส่งข้อมูลเข้ารหัสรางคู่ได้สูงสุดเพียงครั้งละ 16 บิต
- อ้างอิงตำแหน่งหน่วยความจำสูงสุดได้ 10'b บิต หรือ 1 กิโลบิต เพื่อรองรับการติดต่อกับหน่วยความจำขนาด 1K*8bits จากเดิมที่ดีเอ็มเอเวอร์ชันเดิมรองรับการติดต่อกับหน่วยความจำสูงสุดขนาดเพียง 256*8bits
- เชื่อมต่อกับบัสระบบ ซึ่งใช้ติดต่อกับหน่วยความจำได้สูงสุด 2 ตัว และเชื่อมต่อกับบัสอุปกรณ์ต่อพ่วง ซึ่งใช้ติดต่อกับอุปกรณ์ต่อพ่วงได้สูงสุด 4 ตัว เพื่อรองรับการใช้งานอุปกรณ์ต่อพ่วงได้จำนวนมากขึ้น โดยเพิ่มจากดีเอ็มเอเวอร์ชันเดิมที่เชื่อมต่อกับบัสระบบ ซึ่งใช้ติดต่อกับหน่วยความจำได้สูงสุดเพียง 1 ตัว และเชื่อมต่อกับบัสอุปกรณ์ต่อพ่วง ซึ่งใช้ติดต่อกับอุปกรณ์ต่อพ่วงได้สูงสุดเพียง 2 ตัว
- เข้ารหัสหนึ่งในสี่กับสัญญาณอาณัติแบบ 4 ขั้น แทนการเข้ารหัสรางคู่กับสัญญาณอาณัติแบบ 4 ขั้นของดีเอ็มเอเวอร์ชันเดิม เพื่อลดการเปลี่ยนสถานะสัญญาณของดีเอ็มเอ

4.2 โครงสร้างของดีเอ็มเอ



(ก) ตำแหน่งของดีเอ็มเอ



(ข) โครงสร้างของดีเอ็มเอ

รูปที่ 4.1 ตำแหน่งและโครงสร้างของดีเอ็มเอ

ดีเอ็มเอจะเชื่อมต่อกับ บัสระบบ และบัสอุปกรณ์ต่อพ่วง ดังรูปที่ 4.1(ก) โดยโครงสร้างของดีเอ็มเอที่ออกแบบแสดงดังรูปที่ 4.1(ข) ประกอบด้วย ส่วนของรีจิสเตอร์ (Register) บัฟเฟอร์ค่าควบคุม (Control Buffer) ตัวควบคุมดีเอ็มเอ (DMA Controller) วงจรเพิ่มค่า (Incrementor) และวงจรลดค่า (Decrementor) โดยมีรายละเอียดของแต่ละส่วนประกอบดังนี้

4.2.1 **รีจิสเตอร์ของดีเอ็มเอ**

รีจิสเตอร์ในดีเอ็มเอปรับปรุงมาจากดีเอ็มเอ 8237A [10] ประกอบด้วย รีจิสเตอร์ที่ใช้เก็บข้อมูลที่ต้องการทำดีเอ็มเอ และรีจิสเตอร์ที่ใช้เก็บค่าสำหรับควบคุมการทำงานของดีเอ็มเอ ดังนี้

1. รีจิสเตอร์ฐานเลขที่อยู่ต้นทาง (Base Source Address Register) : ใช้เก็บค่าเลขที่อยู่ต้นทาง ที่ใช้อ้างอิงไปยังตำแหน่งที่อยู่ของอุปกรณ์ต้นทาง ที่ต้องการส่งข้อมูล

2. รีจิสเตอร์ฐานเลขที่อยู่ปลายทาง (Base Destination Address Register) : ใช้เก็บค่าเลขที่อยู่ปลายทาง ที่ใช้อ้างอิงไปยังตำแหน่งที่อยู่ของอุปกรณ์ปลายทาง ที่ต้องการรับข้อมูล

3. รีจิสเตอร์ฐานตัวนับ (Base Word Count Register) : ใช้เก็บค่าจำนวนข้อมูลที่ต้องการรับส่งโดยมีหน่วยเป็นเวิร์ด (Word) กล่าวคือ ถ้าค่าในรีจิสเตอร์นี้คือ 1 นั่นหมายความว่าต้องการรับส่งข้อมูลจำนวน 1 เวิร์ด หรือ 16’1of4 บิตนั่นเอง โดยขนาดข้อมูลสูงสุดที่สามารถรับส่งได้ต่อครั้งคือ 3 เวิร์ด หรือ 48’1of4 บิต

4. รีจิสเตอร์เลขที่อยู่ต้นทางปัจจุบัน (Current Source Address Register) : ใช้เก็บค่าเลขที่อยู่ต้นทางปัจจุบัน ซึ่งอาจเป็นค่าเดิมจากรีจิสเตอร์ฐานเลขที่อยู่ต้นทาง หรืออาจเป็นค่าใหม่ที่เกิดจากการเพิ่มค่าเดิมในรีจิสเตอร์นี้ขึ้นไปหนึ่งค่าด้วยวงจรเพิ่มค่า ซึ่งจะเกิดขึ้นในคำสั่งเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำกับหน่วยความจำ (Memory-to-Memory Mode) ที่มีขนาดข้อมูลมากกว่า 1 เวิร์ด

5. รีจิสเตอร์เลขที่อยู่ปลายทางปัจจุบัน (Current Destination Address Register) : ใช้เก็บค่าเลขที่อยู่ปลายทางปัจจุบัน ซึ่งอาจเป็นค่าเดิมจากรีจิสเตอร์ฐานเลขที่อยู่ปลายทาง หรืออาจเป็นค่าใหม่ที่เกิดจากการเพิ่มค่าเดิมในรีจิสเตอร์นี้ขึ้นไปหนึ่งค่าด้วยวงจรเพิ่ม

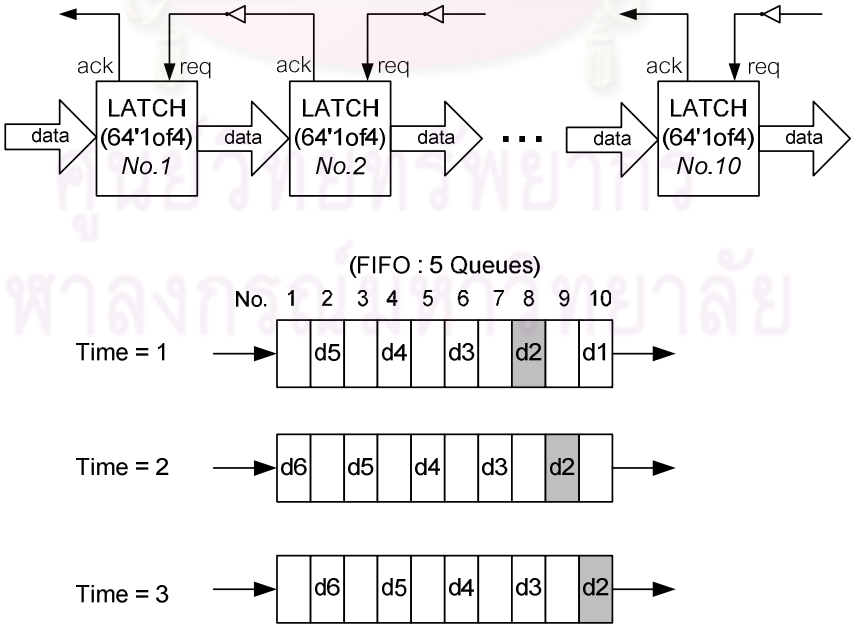
ค่า ซึ่งจะเกิดขึ้นในคำสั่งเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำกับหน่วยความจำ ที่มีขนาดข้อมูลมากกว่า 1 เวิร์ด

6. รีจิสเตอร์ตัวนับปัจจุบัน (Current Word Count Register) : ใช้เก็บค่าจำนวนข้อมูลที่ต้องการรับส่ง ณ ปัจจุบัน ซึ่งอาจเป็นค่าเดิมจากรีจิสเตอร์ฐานตัวนับ หรืออาจเป็นค่าใหม่ที่เกิดจากการลดค่าเดิมในรีจิสเตอร์นี้ลงมาหนึ่งค่าด้วยวงจรลดค่า เพื่อให้สอดคล้องกับจำนวนข้อมูลรับส่งที่เหลืออยู่

7. รีจิสเตอร์พักข้อมูล (Temp Register) : ใช้พักค่าข้อมูลที่รับมาจากอุปกรณ์ต้นทาง เพื่อส่งต่อไปยังอุปกรณ์ปลายทาง โดยใช้รีจิสเตอร์พักข้อมูลขนาด 16’1of4 บิต จำนวน 3 ตัวเพื่อรองรับการพักข้อมูลสูงสุดจำนวน 3 เวิร์ด หรือ 48’1of4 บิต ต่อการรับส่งหนึ่งครั้ง

4.2.2 บัฟเฟอร์ควบคุม

เมื่อไมโครโพรเซสเซอร์พบคำสั่งที่ต้องการเรียกใช้งานดีเอ็มเอ ซึ่งประกอบด้วยคำสั่งอิน (IN) หรือ คำสั่งเอาท์ (OUT) ไมโครโพรเซสเซอร์จะส่งค่าที่ได้จากการถอดรหัสคำสั่งดังกล่าวขนาด 64’1of4 บิต มาเก็บไว้ในบัฟเฟอร์ควบคุมนี้เพื่อรอส่งต่อไปยังดีเอ็มเอภายหลัง หากมีคำสั่งที่เรียกใช้ดีเอ็มเอเข้ามาอย่างต่อเนื่อง บัฟเฟอร์นี้จะเก็บค่าถอดรหัสการทำงานของแต่ละคำสั่งเรียงต่อกันไป โดยเก็บค่าถอดรหัสจากคำสั่งสูงสุดได้ 5 คำสั่ง เมื่ออุปกรณ์ที่ใช้รับส่งข้อมูล



รูปที่ 4.2 โครงสร้างของบัฟเฟอร์ควบคุม

ตามคำสั่งนั้นๆมีความพร้อมที่จะตอบสนองงานตามคำสั่ง ข้อมูลถอดรหัสจะถูกส่งออกจากบัฟเฟอร์นี้ไปยังดีเอ็มเอเพื่อเริ่มต้นการทำดีเอ็มเอ โดยคำสั่งที่เข้ามาก่อนจะได้ทำงานก่อนเรียงกันไปตามลำดับแบบไฟโฟ (First-In, First-Out: FIFO) โครงสร้างของบัฟเฟอร์ควบคุมประกอบด้วยส่วนพักข้อมูล หรือแลช ขนาด 64’1of4 บิต 10 ตัว แสดงดังรูปที่ 4.2 ซึ่งอธิบายการทำงานโดยให้ d1 ถึง d5 เป็นข้อมูล ซึ่ง d1 เป็นข้อมูลเข้ามาก่อนและตามด้วย d2 d3 d4 และd5 ตามลำดับ

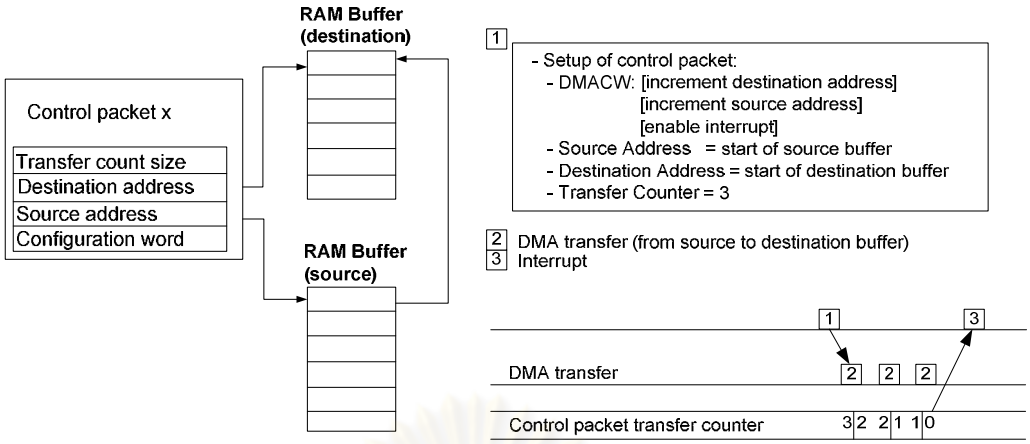
ค่าที่ได้จากการถอดรหัสคำสั่งขนาด 64’1of4 บิตนั้นประกอบด้วย ฐานเลขที่อยู่ต้นทางขนาด 20'1of4 บิต ฐานเลขที่อยู่ปลายทางขนาด 20'1of4 บิต ฐานตัวนับขนาด 4'1of4 บิต หมายเลขพอร์ทอุปกรณ์ต่อพ่วงต้นทางและปลายทางรวม 16’1of4 บิต และสัญญาณควบคุมขนาด 4'b บิต

4.2.3 ตัวควบคุมดีเอ็มเอ

ตัวควบคุมดีเอ็มเอทำหน้าที่รับส่งสัญญาณติดต่อระหว่างดีเอ็มเอกับไมโครโพรเซสเซอร์ และอุปกรณ์อื่นๆ รวมทั้งส่งสัญญาณควบคุมไปยังส่วนต่างๆของดีเอ็มเอเพื่อให้ดีเอ็มเอทำงานสอดคล้องตามคำสั่งที่ได้รับอย่างถูกต้อง โดยออกแบบสัญญาณควบคุมในจุดต่างๆตามลำดับการเปลี่ยนแปลงของสัญญาณด้วยกราฟบรรยายการเปลี่ยนสัญญาณ และใช้โปรแกรม Petrify Tool สร้างวงจรควบคุมจากกราฟบรรยายการเปลี่ยนสัญญาณดังกล่าวเช่นเดียวกับการออกแบบตัวควบคุมบัสระบบในบทที่ 3 ที่ได้กล่าวไปแล้ว

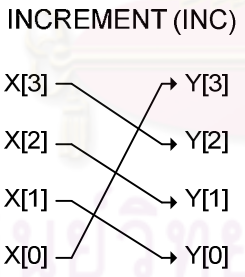
4.2.4 วงจรเพิ่มค่า

วงจรเพิ่มค่าทำหน้าที่เพิ่มค่าที่อยู่ในรีจิสเตอร์เลขที่อยู่ต้นทางปัจจุบัน และรีจิสเตอร์เลขที่อยู่ปลายทางปัจจุบันขึ้นทีละหนึ่งค่า เมื่อหมดรอบการทำงานแต่ละรอบ ซึ่งเป็นการทำงานตามลักษณะของคำสั่งเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำกับหน่วยความจำ (Basic RAM to RAM Transfer with DMA) ที่มีขนาดข้อมูลมากกว่า 1 เวิร์ด เพื่อส่งข้อมูลไปยังตำแหน่งต่อไปในหน่วยความจำอย่างต่อเนื่อง ดังรูปที่ 4.3 [19]



รูปที่ 4.3 การเคลื่อนย้ายข้อมูลด้วยดีเอ็มเอระหว่างหน่วยความจำกับหน่วยความจำ

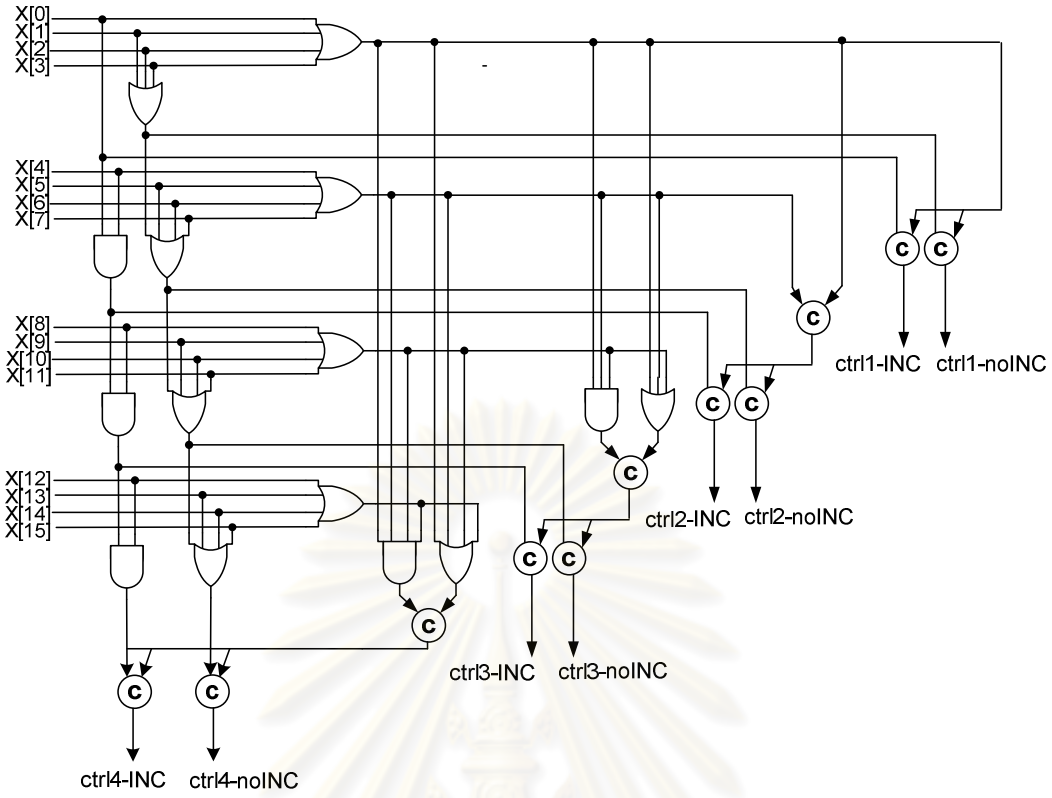
วงจรเพิ่มค่าเข้ารหัสหนึ่งในสี่ที่ออกแบบ รองรับการคำนวณข้อมูลเข้ารหัสหนึ่งในสี่ขนาด 20 บิตได้ มีโครงสร้างดังรูปที่ 4.4 สำหรับการออกแบบวงจรเลือกหลักซึ่งอยู่ในวงจรเพิ่มค่ามีเงื่อนซับซ้อน จึงใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับมาสร้างวงจรเลือกหลักเข้ารหัสหนึ่งในสี่ ได้ดังรูปที่ 4.4(ข) รายละเอียดการสร้างวงจรเข้ารหัสหนึ่งในสี่ด้วยแผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ ได้อธิบายไว้ในบทที่ 5



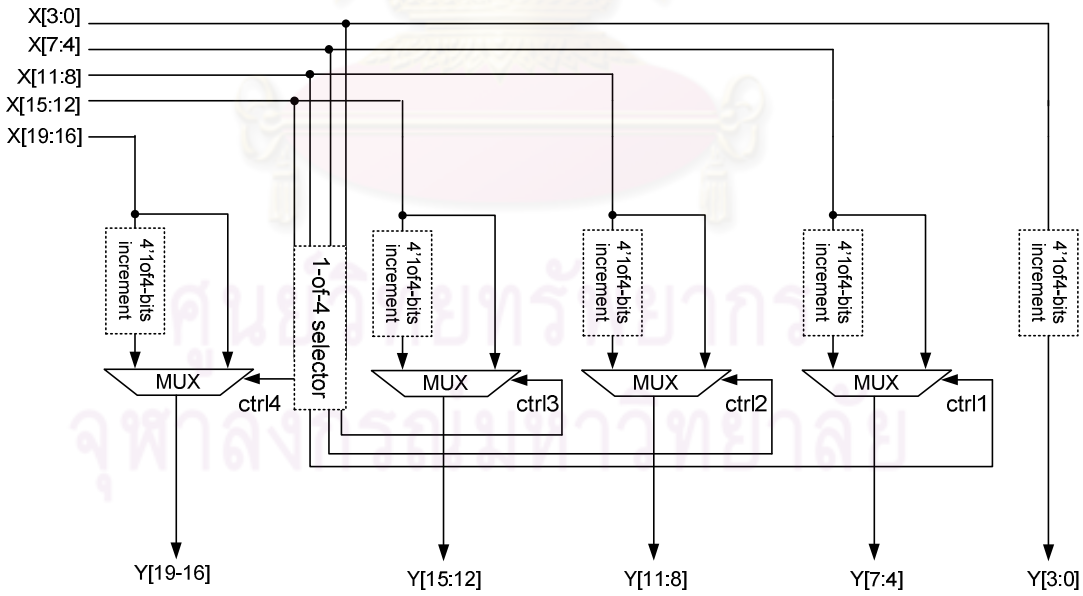
Input				Output			
X[3]	X[2]	X[1]	X[0]	Y[3]	Y[2]	Y[1]	Y[0]
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0
0	0	1	0	0	0	0	1
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0

(ก) วงจรเพิ่มค่าเข้ารหัสหนึ่งในสี่ขนาด 4’1of4 บิต (4’1of4-bits Increment)

รูปที่ 4.4 วงจรเพิ่มค่าเข้ารหัสหนึ่งในสี่



(ข) วงจรเลือกหลักสำหรับวงจรเพิ่มค่าเข้ารหัสหนึ่งในสี่ขนาด 20’1of4 บิต (1-of-4 Selector)



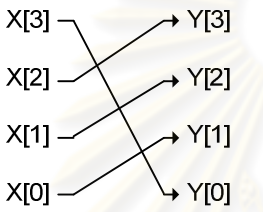
(ค) วงจรเพิ่มค่าเข้ารหัสหนึ่งในสี่ขนาด 20’1of4 บิต

รูปที่ 4.4 วงจรเพิ่มค่าเข้ารหัสหนึ่งในสี่ (ต่อ)

4.2.5 วงจรลดค่า

วงจรลดค่าทำหน้าที่ลดค่าจำนวนข้อมูลที่ต้องการรับส่งในรีจิสเตอร์ตัวนับปัจจุบันลงทีละหนึ่งค่า หรือลบหนึ่งเมื่อหมดรอบการทำงานแต่ละรอบ ซึ่งเป็นการทำงานตามลักษณะของคำสั่งเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำกับหน่วยความจำ ที่มีขนาดข้อมูลมากกว่า 1 เวิร์ด เพื่อตรวจสอบจำนวนคงเหลือของข้อมูลที่ต้องการรับส่ง โดยวงจรลดค่าที่ออกแบบรองรับการคำนวณข้อมูลเข้ารหัสหนึ่งในสี่ขนาด 4 บิตได้ โครงสร้างวงจรลดค่าเป็นดังรูปที่ 4.5

DECREMENT (DEC)



Input				Output			
X[3]	X[2]	X[1]	X[0]	Y[3]	Y[2]	Y[1]	Y[0]
1	0	0	0	0	0	0	1
0	1	0	0	1	0	0	0
0	0	1	0	0	1	0	0
0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0

รูปที่ 4.5 วงจรลดค่าเข้ารหัสหนึ่งในสี่ขนาด 4 บิต

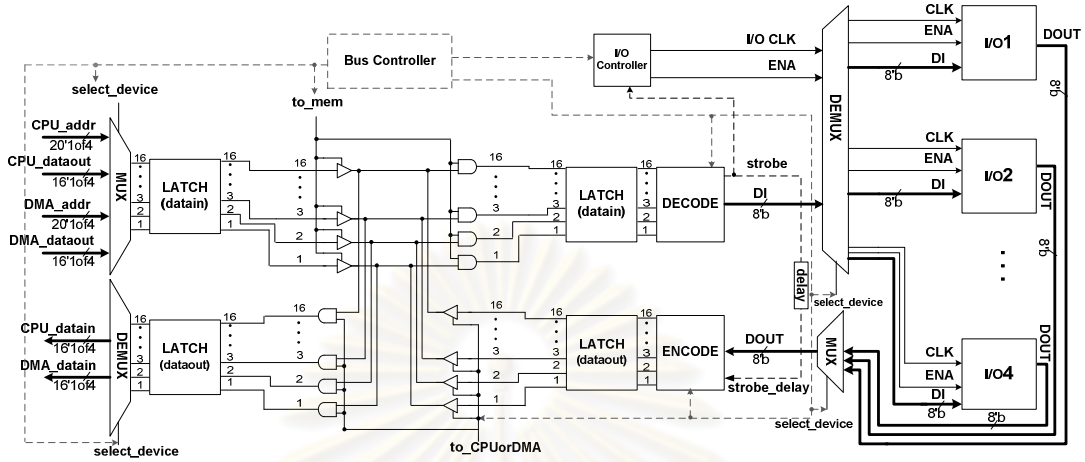
4.3 บัสอุปกรณ์ต่อพ่วง

ดีเอ็มเอ ไมโครโพรเซสเซอร์ และหน่วยความจำหลัก เชื่อมต่อกันผ่านบัสระบบในบทที่ 3 ส่วนอุปกรณ์ต่อพ่วงจะเชื่อมต่อกับดีเอ็มเอผ่านบัสสำหรับอุปกรณ์ต่อพ่วง ซึ่งติดต่อกับอุปกรณ์ต่อพ่วงได้สูงสุด 4 ตัว ข้อดีของการใช้บัสอุปกรณ์ต่อพ่วงคือ เมื่อดีเอ็มเอทำงานรับส่งข้อมูลจากอุปกรณ์ต่อพ่วงไปยังอุปกรณ์ต่อพ่วงด้วยกันจะไม่ใช้บัสระบบ แต่ใช้บัสอุปกรณ์ต่อพ่วงแทน ส่งผลให้ไมโครโพรเซสเซอร์สามารถใช้งานบัสระบบเพื่อติดต่อกับหน่วยความจำในขณะที่ดีเอ็มเอทำงานดังกล่าวได้ นอกจากนี้บัสอุปกรณ์ต่อพ่วงยังช่วยลดโครงสร้างการเชื่อมต่อเมื่ออุปกรณ์มีจำนวนมากขึ้นอีกด้วย

4.3.1 โครงสร้างของบัสอุปกรณ์ต่อพ่วง

บัสอุปกรณ์ต่อพ่วงมีโครงสร้างหลักเช่นเดียวกับบัสระบบในบทที่ 3 แต่ตัดส่วนพักเลขที่อยู่ (Address Latch) ส่วนติดต่อกับไมโครโพรเซสเซอร์ และส่วนติดต่อกับหน่วยความจำ

แบบสมวารออก และเพิ่มส่วนติดต่อกับอุปกรณ์ต่อพ่วงแบบสมวาร (I/O Interface) แทนบัสอุปกรณ์ต่อพ่วงแสดงดังรูปที่ 4.6



รูปที่ 4.6 บัสอุปกรณ์ต่อพ่วง

4.3.2 การทำงานของบัสอุปกรณ์ต่อพ่วง

บัสอุปกรณ์ต่อพ่วงทำหน้าที่รับส่งข้อมูลระหว่างดีเอ็มเอและอุปกรณ์ต่อพ่วงตามคำสั่ง ซึ่งคำสั่งที่เรียกใช้งานบัสอุปกรณ์ต่อพ่วงมีดังนี้

- คำสั่งอิน (IN): แบ่งออกเป็นคำสั่งย่อยที่เรียกใช้งานบัสระบบคือ

- IN I/O,@K : ดีเอ็มเอจะส่งสัญญาณอ่าน และเลขที่อยู่ตำแหน่งที่ต้องการอ่าน ผ่านบัสระบบไปยังหน่วยความจำต้นทาง เมื่ออ่านค่าจากหน่วยความจำต้นทางเสร็จสิ้นจะส่งค่าที่อ่านได้ผ่านบัสระบบกลับมายังดีเอ็มเอ ดีเอ็มเอจะส่งสัญญาณ moveout เข้าบัสอุปกรณ์ต่อพ่วงและส่งค่าที่อ่านได้ในเอ็มเอ ผ่านบัสอุปกรณ์ต่อพ่วง ไปยังอุปกรณ์ต่อพ่วงปลายทาง

- IN I/O,I/O : ดีเอ็มเอจะส่งสัญญาณ movein เข้าบัสอุปกรณ์ต่อพ่วง และบัสอุปกรณ์ต่อพ่วงจะรับข้อมูลจากอุปกรณ์ต่อพ่วงต้นทางส่งต่อมายังดีเอ็มเอ เมื่อดีเอ็มเอรับข้อมูลครบแล้วสัญญาณ movein จะเปลี่ยนเป็น 0 จากนั้นดีเอ็มเอจะส่งสัญญาณ moveout เข้าบัสอุปกรณ์ต่อพ่วงและส่งข้อมูลที่รับไว้ในดีเอ็มเอดังกล่าว ผ่านบัสอุปกรณ์ต่อพ่วง ไปยังอุปกรณ์ต่อพ่วงปลายทาง

- คำสั่งเอาท์ (OUT) I/O,@K : ดีเอ็มเอจะส่งสัญญาณ movein เข้าบัสอุปกรณ์ต่อพ่วง และบัสอุปกรณ์ต่อพ่วงจะรับข้อมูลจากอุปกรณ์ต่อพ่วงต้นทางส่งต่อมายังดีเอ็มเอ เมื่อดีเอ็มเอรับข้อมูลครบแล้วสัญญาณ movein จะเปลี่ยนเป็น 0 จากนั้นดีเอ็มเอจะส่งสัญญาณเขียนเลขที่อยู่ตำแหน่งที่ต้องการเขียน และข้อมูลที่รับจากอุปกรณ์ต่อพ่วงดังกล่าว ผ่านบัสระบบไปยังหน่วยความจำ เพื่อเขียนค่าลงหน่วยความจำ

จะเห็นได้ว่างานหลักของบัสอุปกรณ์ต่อพ่วงคือ รับส่งข้อมูลจากอุปกรณ์ต่อพ่วง การรับข้อมูลจากอุปกรณ์ต่อพ่วง มีขั้นตอนโดยสรุปคือ เมื่อบัสอุปกรณ์ได้รับสัญญาณรับข้อมูลคือ movein และรับข้อมูลจากอุปกรณ์ต่อพ่วงต้นทางครบแล้วจะเข้าสู่ขั้นทำงาน โดยจะส่งข้อมูลดังกล่าวผ่านตัวบัสอุปกรณ์ต่อพ่วงไปยังดีเอ็มเอ ส่วนการส่งข้อมูลไปยังอุปกรณ์ต่อพ่วง มีขั้นตอนโดยสรุปคือ เมื่อบัสอุปกรณ์ได้รับสัญญาณส่งข้อมูลคือ moveout และรับข้อมูลจากดีเอ็มเอครบแล้วจะเข้าสู่ขั้นทำงาน โดยจะส่งข้อมูลดังกล่าวผ่านตัวบัสอุปกรณ์ต่อพ่วงไปยังอุปกรณ์ต่อพ่วงปลายทาง เมื่อเสร็จสิ้นการทำงานสัญญาณควบคุมทั้งหมดจะเปลี่ยนเป็น 0 เพื่อเข้าสู่ขั้นว่าง

4.4 การทำงานของดีเอ็มเอ

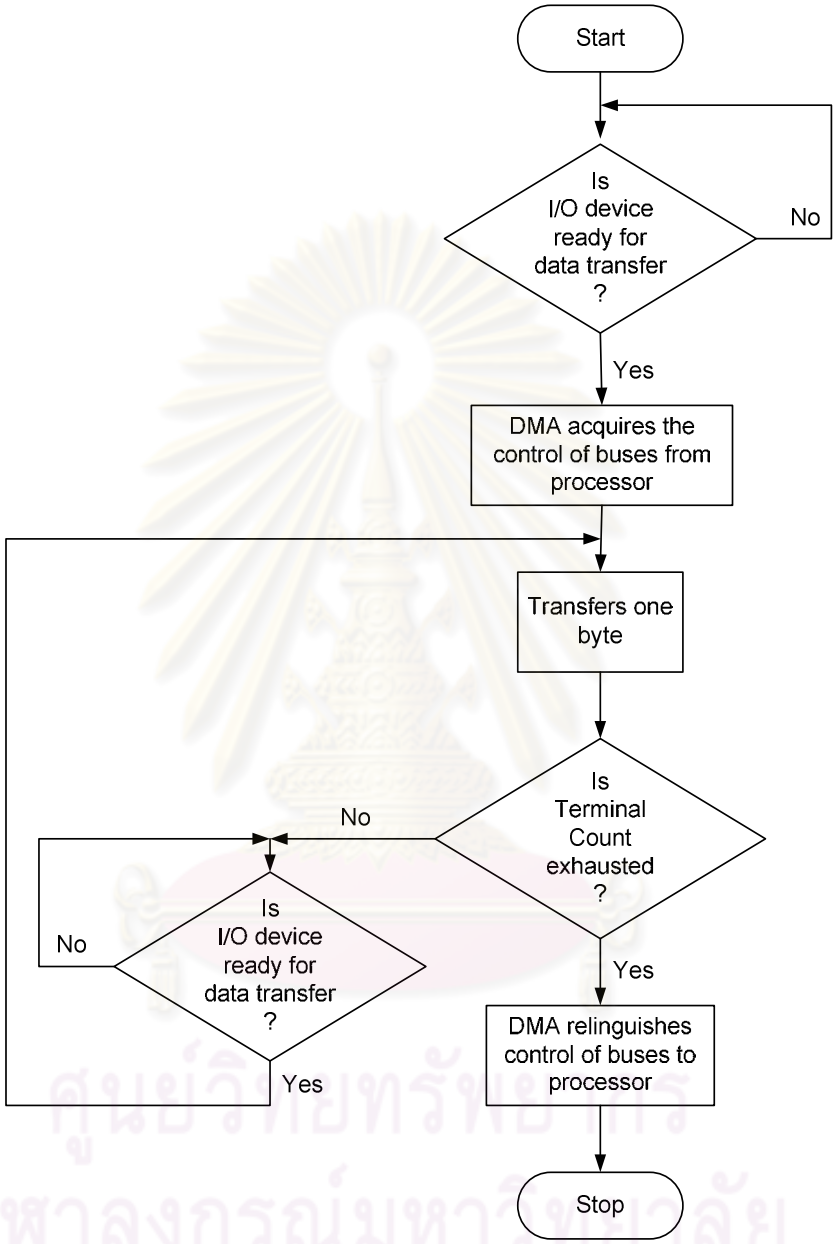
การทำดีเอ็มเอใช้รูปแบบการเคลื่อนย้ายข้อมูลแบบ Block Transfer Mode ดังรูปที่ 4.7 [10] โดยใช้จังหวะของสัญญาณติดต่อระหว่างดีเอ็มเอ ไมโครโพรเซสเซอร์ และหน่วยความจำดังรูปที่ 4.8 ซึ่งปรับปรุงมาจากดีเอ็มเอแบบอสมวาร [4] เป็นตัวควบคุมจังหวะในการทำงาน ซึ่งอธิบายขั้นตอนการทำงานของดีเอ็มเอ ได้ดังนี้

1. เมื่อไมโครโพรเซสเซอร์พบคำสั่งที่เรียกใช้งานดีเอ็มเอ ซึ่งประกอบด้วย คำสั่งอิน และคำสั่งเอาท์ จะส่งค่าถอดรหัสการทำงานจากคำสั่งดังกล่าว ไปที่บัฟเฟอร์ควบคุมของดีเอ็มเอ

2. ดีเอ็มเอตรวจสอบสถานะของอุปกรณ์ต้นทางที่ต้องการส่งข้อมูล และอุปกรณ์ปลายทางที่ต้องการรับข้อมูล ตามค่าถอดรหัสการทำงานที่เก็บไว้ในบัฟเฟอร์ควบคุมในข้อ 1 จากนั้น รอจนกว่าอุปกรณ์มีสถานะพร้อมเคลื่อนย้ายข้อมูล (I/O Ready = 1)

3. เมื่ออุปกรณ์ในข้อ 2 มีสถานะพร้อมทำงาน ดีเอ็มเอจะร้องขอใช้บัสระบบ (Bus Request: BR = 1) จากไมโครโพรเซสเซอร์และได้รับสิทธิ์ในการใช้บัสระบบ (Bus Grant:

BG = 1) จากไมโครโพรเซสเซอร์หากเป็นงานที่เกี่ยวข้องกับหน่วยความจำหลัก หรือได้สิทธิ์ในการใช้บัสอุปกรณ์ต่อพ่วงหากเป็นงานที่เกี่ยวข้องกับอุปกรณ์ต่อพ่วงเท่านั้น



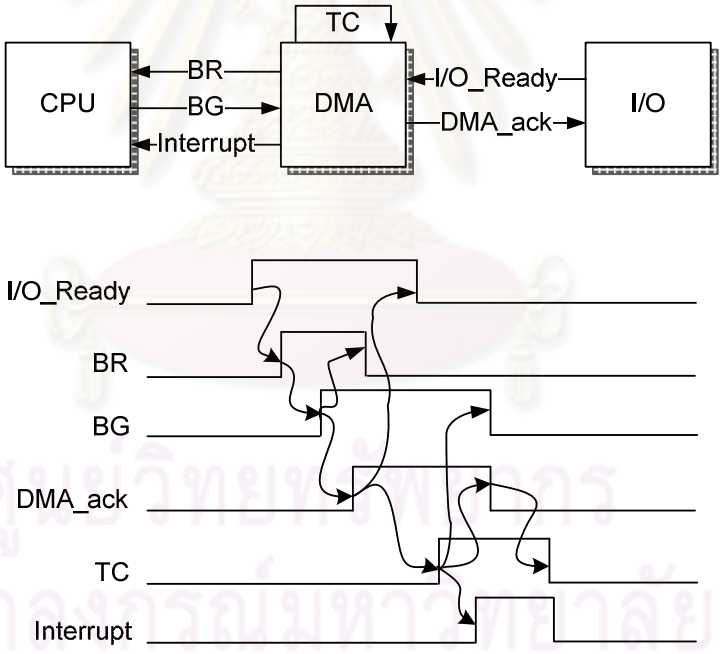
รูปที่ 4.7 การทำดีเอ็มเอแบบ Block Transfer Mode

4. ค่าถอดรหัสในข้อ 1 ถูกส่งมาที่ดีเอ็มเอเพื่อระบุทิศทาง ระบุที่อยู่ของอุปกรณ์ส่งข้อมูล ที่อยู่ของอุปกรณ์รับข้อมูล ข้อมูลที่ต้องการรับส่ง และจำนวนข้อมูลที่ต้องการรับส่ง จากนั้นดีเอ็มเอทำการรับส่งข้อมูลผ่านบัสที่ได้สิทธิ์ใช้งานในข้อ 3 (DMA ack = 1)

5. ดีเอ็มเอตรวจสอบว่าจำนวนข้อมูลที่รับส่งในข้อ 4 เท่ากับจำนวนข้อมูลที่ต้องการรับส่งหรือไม่ ถ้าไม่ให้ทำการส่งต่อไปจนครบ โดยหากเป็นคำสั่งเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำกับหน่วยความจำ ที่มีขนาดข้อมูลมากกว่า 1 เวิร์ด ดีเอ็มเอจะเพิ่มค่าที่อยู่ในรีจิสเตอร์เลขที่อยู่ต้นทางปัจจุบัน และรีจิสเตอร์เลขที่อยู่ปลายทางปัจจุบันขึ้นหนึ่งค่าในแต่ละรอบก่อนจะทำการส่งรอบต่อไป เพื่อส่งข้อมูลไปยังตำแหน่งต่อไปในหน่วยความจำอย่างต่อเนื่อง

การตรวจสอบจะเปรียบเทียบกับค่าในรีจิสเตอร์ตัวนับปัจจุบัน ซึ่งเก็บจำนวนเวิร์ดข้อมูลที่ต้องการส่งไว้ ทุกครั้งที่รับหรือส่งข้อมูลครบ 1 เวิร์ด ค่าในรีจิสเตอร์ดังกล่าวจะลดลง 1 ค่า หากค่าดังกล่าวมีค่าเป็น 0 นั่นหมายถึงการรับส่งข้อมูลครบตามจำนวนที่ต้องการแล้ว

6. เมื่อส่งข้อมูลครบแล้ว (TC = 1) ให้คืนสิทธิ์การใช้บัสระบบให้กับไมโครโพรเซสเซอร์หากเป็นงานที่เกี่ยวข้องกับหน่วยความจำหลัก หรือหยุดใช้บัสอุปกรณ์ต่อพ่วงหากเป็นงานที่เกี่ยวข้องกับอุปกรณ์ต่อพ่วงเท่านั้น และจบการทำดีเอ็มเอ (Interrupt = 1)



รูปที่ 4.8 การติดต่อของสัญญาณควบคุมระหว่างดีเอ็มเอกับระบบ

ตารางที่ 4.1 ค่าภายในรีจิสเตอร์และอินพุทเอาท์พุทของดีเอ็มเอสำหรับคำสั่ง OUT3 @20,@10

Register	Times								
	t1	t2	t3	t4	t5	t6	t7	t8	t9
Base Source Address	10								null
Base Destination Address	20								
Base Word Count	3								
Current Source Address	null	10	11	12	null				
Current Destination Address		null				20	21	22	
Current Word Count		3	2	1	0	null	3	2	1	0	null
Temp1		25							null
Temp2		null	80						
Temp3		null	null	72					
I/O DATA									
DMA input	null	25	80	72	null				null
DMA output		null				25	80	72	

ตารางที่ 4.1 แสดงค่าของรีจิสเตอร์ภายในดีเอ็มเอ ซึ่งตอบสนองงานตามคำสั่งตัวอย่างที่กำหนด คือเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำกับหน่วยความจำ โดยมีรายละเอียดคือ เคลื่อนย้ายข้อมูลขนาด 3 เวิร์ด จากหน่วยความจำต้นทางตำแหน่งที่ 10 ไปยังหน่วยความจำปลายทางตำแหน่งที่ 20 โดยค่าข้อมูลเวิร์ดแรกที่เก็บไว้ในหน่วยความจำหลักตำแหน่งที่ 10 คือ 25 ค่าข้อมูลเวิร์ดต่อไปคือ 80 และ 72 เก็บไว้ในตำแหน่งถัดไปจากเวิร์ดแรกคือตำแหน่งที่ 11 และ 12 ตามลำดับ

บทที่ 5

การออกแบบไมโครโพรเซสเซอร์แบบอสมวาร

ในบทนี้กล่าวถึงการออกแบบไมโครโพรเซสเซอร์แบบอสมวารเข้ารหัสหนึ่งในสี่ ซึ่งเป็นองค์ประกอบที่ได้ครอบครองการใช้งานบัสระบบ โดยใช้งานบัสระบบเพื่อตอบสนองงานตามคำสั่งที่ต้องการติดต่อรับส่งข้อมูลกับหน่วยความจำ รายละเอียดของการออกแบบไมโครโพรเซสเซอร์มีดังต่อไปนี้

5.1 คุณสมบัติของไมโครโพรเซสเซอร์

ไมโครโพรเซสเซอร์แบบอสมวารที่ออกแบบ มีคุณสมบัติโดยสรุปได้ดังนี้

- มีขนาด 8 บิต หรือ 16’1of4 บิต เช่นเดียวกับไมโครโพรเซสเซอร์เข้ารหัสรางคู่ [4,17]
- เข้ารหัสหนึ่งในสี่กับสัญญาณอาณัติแบบ 4 ขั้น แทนการเข้ารหัสรางคู่กับสัญญาณอาณัติแบบ 4 ขั้นของไมโครโพรเซสเซอร์เวอร์ชันเดิม [4,17] เพื่อลดการเปลี่ยนสถานะสัญญาณของไมโครโพรเซสเซอร์
- รองรับการบริการอินเตอร์รัพท์ และรองรับการทำงานร่วมกับบัสระบบและดีเอ็มเอ เช่นเดียวกับไมโครโพรเซสเซอร์เวอร์ชันเดิม [4] เพื่อให้การทำงานร่วมกับอุปกรณ์ต่อพ่วงมีความรวดเร็วยิ่งขึ้น
- ติดต่อกับอุปกรณ์อินพุท/เอาท์พุทแบบ I/O-mapped I/O หรือ Peripheral-mapped แทนการติดต่อกับอุปกรณ์อินพุท/เอาท์พุทแบบ Memory-mapped I/O ในไมโครโพรเซสเซอร์เวอร์ชันเดิม [4] เพื่อให้สามารถใช้หน่วยความจำได้ทุกตำแหน่งเลขที่อยู่
- มีจำนวนคำสั่ง 27 คำสั่ง ซึ่งเพิ่มจากไมโครโพรเซสเซอร์เวอร์ชันเดิม [17] ที่มีจำนวนคำสั่งเพียง 16 คำสั่ง

5.2 ชุดคำสั่งและรหัสดำเนินการ

ไมโครโปรเซสเซอร์ที่ออกแบบ ประกอบด้วยชุดคำสั่ง (Instruction Set) เดิมจากไมโครโพรเซสเซอร์ 8 บิตเข้ารหัสรางคู่ [17] และชุดคำสั่งซึ่งออกแบบเพิ่มเติมสำหรับเรียกใช้งานดีเอ็มเอ และการบริการอินเตอร์รัพท์ ชุดคำสั่งและรหัสดำเนินการทั้งหมดแสดงในภาคผนวก ก. โดยชุดคำสั่งทั้งหมดนั้นแบ่งออกเป็น 3 กลุ่ม มีจำนวนคำสั่งรวม 27 คำสั่ง รายละเอียดมีดังต่อไปนี้

1. กลุ่มคำสั่งเคลื่อนย้ายข้อมูล (Data Transfer Operation) ใช้สำหรับงานเคลื่อนย้ายข้อมูลระหว่างไมโครโพรเซสเซอร์ หน่วยความจำ และอุปกรณ์ต่อพ่วง ประกอบด้วยคำสั่งโหลด (Load: LD) คำสั่งสโตร์ (Store: ST) คำสั่งอิน(IN) และคำสั่งเอาท์ (OUT)

2. กลุ่มคำสั่งควบคุมการทำงาน (Program and Machine Control Operation) ใช้สำหรับควบคุมการทำงานตามเงื่อนไขที่กำหนดไว้ ประกอบด้วยคำสั่งกระโดดแบบไร้เงื่อนไข (Jump: JMP) คำสั่งกระโดดเมื่อค่าในรีจิสเตอร์ตัวสะสม (Accumulator Register: ACC, A) เท่ากับศูนย์ (Jump Zero: JZ) คำสั่งกระโดดเมื่อค่าในรีจิสเตอร์ตัวสะสมไม่เท่ากับศูนย์ (Jump not Zero: JNZ) คำสั่งกระโดดเมื่อค่าในรีจิสเตอร์ตัวสะสมมีตัวทด (Jump Carry: JC) คำสั่งกระโดดเมื่อค่าในรีจิสเตอร์ตัวสะสมไม่มีตัวทด (Jump not Carry: JNC) คำสั่งเรียกใช้งานโปรแกรมย่อย (CALL) คำสั่งกลับไปใช้งานโปรแกรมหลัก (Return: RET) และคำสั่งกลับจากงานบริการอินเตอร์รัพท์ (Return from Interrupt: RETI)

3. กลุ่มคำสั่งดำเนินการทางคณิตศาสตร์และตรรกะ (Arithmetic and Logic Operation) ประกอบด้วยคำสั่งสำหรับงานดำเนินการทางคณิตศาสตร์คือ คำสั่งบวก (Addition: ADD) คำสั่งลบ (Subtract: SUB) คำสั่งเพิ่มค่าหนึ่งค่า (Increment: INC) คำสั่งลดค่าหนึ่งค่า (Decrement: DEC) คำสั่งคูณ (Multiplicand: MUL) และประกอบด้วยคำสั่งสำหรับงานดำเนินการทางตรรกะ คือ คำสั่งเลื่อนทุกบิตไปทางซ้ายแบบไม่คิดเครื่องหมาย (Shift Logical Left: SLL) คำสั่งเลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมาย (Shift Logical Right: SLR) คำสั่งเลื่อนทุกบิตไปทางซ้ายแบบคิดเครื่องหมาย (Shift Arithmetic Left: SAL) คำสั่งเลื่อนทุกบิตไปทางขวาแบบคิดเครื่องหมาย (Shift Arithmetic Right: SAR) คำสั่งกระทำลอจิกแอนด์ (Logical And: AND) คำสั่งกระทำลอจิกออร์ (Logical Or: OR) คำสั่งกระทำลอจิกเอ็กคลูซีฟออร์ (Logical Exclusive-or: XOR) คำสั่งกระทำบิตต่อบิตแอนด์ (Bitwise And: ANDB) คำสั่งกระทำบิตต่อบิตออร์ (Bitwise Or: ORB) และคำสั่งกระทำบิตต่อบิตเอ็กคลูซีฟออร์ (Bitwise Exclusive-or: XORB)

5.3 โครงสร้างของไมโครโพรเซสเซอร์

โครงสร้างของไมโครโพรเซสเซอร์เข้ารหัสหนึ่งในสี่ ปรับปรุงจากไมโครโพรเซสเซอร์เข้ารหัสรางคู่[17] ประกอบด้วย ส่วนอ่านคำสั่ง (Fetch Unit) ส่วนแปลความหมายของคำสั่ง (Decode Unit) ส่วนประมวลผล (Execute Unit) หน่วยคำนวณทางคณิตศาสตร์และตรรกะ (Arithmetic and Logic Units: ALU) ส่วนเขียนผลลัพธ์ (Write Result Unit) ส่วนบริการอินเตอร์รัพท์ (Interrupt Unit) และส่วนควบคุม (Control Unit) โดยมีรายละเอียดดังต่อไปนี้

5.3.1 ส่วนอ่านคำสั่งและส่วนแปลความหมายของคำสั่ง

ส่วนอ่านคำสั่ง ทำหน้าที่อ่านคำสั่งที่ไมโครโพรเซสเซอร์ได้รับมา และส่งต่อไปยังส่วนแปลความหมายของคำสั่ง เพื่อแปลความหมายคำสั่งออกเป็นขั้นตอนการทำงานย่อย จากนั้นส่งค่าที่ต้องใช้สำหรับการทำงานย่อยดังกล่าวไปยังส่วนประมวลผล เพื่อประมวลผลต่อไป ขั้นตอนการทำงานของส่วนอ่านคำสั่งและส่วนแปลความหมายของคำสั่ง เป็นดังรูปที่ 5.1 โดยมีขั้นตอนการทำงานดังนี้

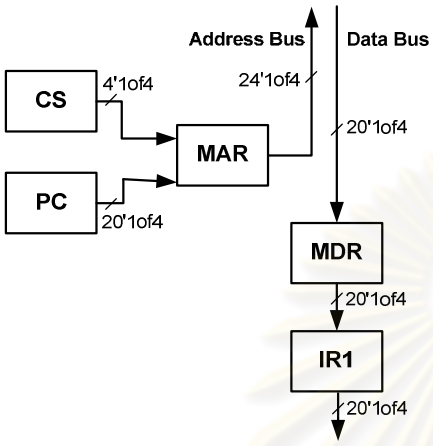
ขั้นตอนที่ 1 เป็นดังรูป 5.1(ก) โดยค่าของรีจิสเตอร์ CS (Code Segment) และ PC (Program Counter) จะถูกเก็บไว้ที่รีจิสเตอร์ MAR (Memory Address Register) เพื่อชี้ตำแหน่งของคำสั่งที่ต้องการอ่าน จากนั้นอ่านรหัสดำเนินการของคำสั่ง 10'b บิตแรกเข้ามาเก็บไว้รีจิสเตอร์ MDR (Memory Data Register) และส่งต่อไปยังรีจิสเตอร์ IR1 (Instruction Register) เพื่อแปลความหมายของคำสั่งว่ามีความยาวของรหัสดำเนินการสูงสุดเท่าไหร่ และต้องอ่านข้อมูลจากหน่วยความจำหรือไม่ โดยแบ่งรูปแบบและการทำงานของคำสั่งออกเป็น

- คำสั่งที่มีความยาวของรหัสดำเนินการเท่ากับ 10'b บิต การทำงานเพียงขั้นตอนที่ 1 ก็สามารถอ่านรหัสดำเนินการได้ครบ จึงไม่ต้องดำเนินการขั้นตอนต่อไป
- คำสั่งที่มีความยาวของรหัสดำเนินการเท่ากับ 20'b บิต เมื่อเสร็จสิ้นขั้นตอนที่ 1 แล้ว จะทำขั้นตอนที่ 2 ต่อไป ดังรูปที่ 5.1(ข) คืออ่านรหัสดำเนินการอีก 10'b บิตถัดไปที่เหลือเข้ามาให้ครบ โดยเพิ่มค่าในรีจิสเตอร์ PC ขึ้นหนึ่งค่า และใช้ค่าในรีจิสเตอร์ CS และ PC เก็บไว้

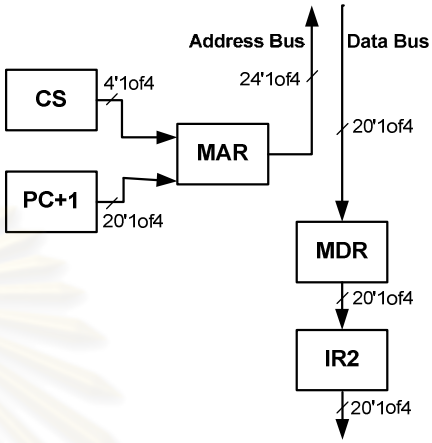
ที่รีจิสเตอร์ MAR เพื่อชี้ไปยังตำแหน่งถัดไปของรหัสดำเนินการที่เหลือ จากนั้นอ่านรหัสดำเนินการที่เหลืออีก 10'b บิตเข้ามาเก็บไว้ในรีจิสเตอร์ MDR และส่งต่อไปยังรีจิสเตอร์ IR2

- คำสั่งที่มีความยาวของรหัสดำเนินการ 30'b บิต เมื่อเสร็จสิ้นขั้นตอนที่ 1 และขั้นตอนที่ 2 แล้ว จะทำขั้นตอนที่ 3 ต่อไป ดังรูปที่ 5.1(ค) คืออ่านรหัสดำเนินการอีก 10'b บิตถัดไปที่เหลือเข้ามาให้ครบ โดยเพิ่มค่าในรีจิสเตอร์ PC ขึ้นหนึ่งค่า และใช้ค่าในรีจิสเตอร์ CS และ PC เก็บไว้ที่รีจิสเตอร์ MAR เพื่อชี้ไปยังตำแหน่งถัดไปของรหัสดำเนินการที่เหลือ จากนั้นอ่านรหัสดำเนินการที่เหลืออีก 10'b บิตเข้ามาเก็บไว้ในรีจิสเตอร์ MDR และส่งต่อไปยังรีจิสเตอร์ IR3
- คำสั่งที่ต้องอ่านข้อมูลในหน่วยความจำ จากตำแหน่งซึ่งเป็นค่าคงที่ที่กำหนดขึ้นเองโดยไม่ผ่านดีเอ็มเอ เช่น LD A, @K เป็นต้น หลังจากทำขั้นตอนที่ 1 ถึง 3 ตามเงื่อนไขที่เหมาะสมแล้ว จะทำตามขั้นตอนที่ 4 ต่อไป ดังรูปที่ 5.1(ง) โดยใช้ค่าในรีจิสเตอร์ DS (Data Segment) และ IR2 เก็บไว้ที่รีจิสเตอร์ MAR เพื่อชี้ไปยังตำแหน่งของหน่วยความจำที่ต้องการอ่านข้อมูล จากนั้นอ่านข้อมูลจำนวน 8'b บิตจากหน่วยความจำ ผ่านบัสระบบเข้ามาเก็บไว้ในรีจิสเตอร์ MDR และส่งต่อไปยังรีจิสเตอร์ OP (Operand)
- คำสั่งที่ต้องอ่านข้อมูลในหน่วยความจำ จากตำแหน่งที่ค่าในรีจิสเตอร์ R0 หรือ R1 กำหนดโดยไม่ผ่านดีเอ็มเอ เช่น LD A, @Reg เป็นต้น หลังจากทำขั้นตอนที่ 1 ถึง 3 ตามเงื่อนไขที่เหมาะสมแล้ว จะทำตามขั้นตอนที่ 5 ต่อไป ดังรูปที่ 5.1(จ) โดยใช้ค่าในรีจิสเตอร์ DS และ R0 หรือ R1 เก็บไว้ที่รีจิสเตอร์ MAR เพื่อชี้ไปยังตำแหน่งของหน่วยความจำที่ต้องการอ่านข้อมูล จากนั้นอ่านข้อมูลจำนวน 8'b บิตจากหน่วยความจำ ผ่านบัสระบบเข้ามาเก็บไว้ในรีจิสเตอร์ MDR และส่งต่อไปยังรีจิสเตอร์ OP
- คำสั่งที่ต้องข้ามไปอ่านคำสั่งใหม่เข้ามา เช่น JMP CALL JZ JNZ เป็นต้น หลังจากทำขั้นตอนที่ 1 และ 2 แล้ว หากเป็นคำสั่งข้ามไปอ่านคำสั่งใหม่แบบไม่มีเงื่อนไข เช่น JMP CALL จะทำตามขั้นตอนที่ 6 ต่อไป แต่หากเป็นคำสั่งแบบมีเงื่อนไข เช่น JZ JNZ จะต้องรอผลลัพธ์การเปรียบเทียบเงื่อนไข จากค่าของแฟลกรีจิสเตอร์ ในส่วนประมวลผล โดยถ้าไม่เป็นไปตามเงื่อนไข จะสิ้นสุดการทำงานของคำสั่งนี้ แต่ถ้าเป็นไปตามเงื่อนไข จะทำตามขั้นตอนที่ 6 ต่อไปเช่นกัน ขั้นตอนที่ 6 เป็นดังรูปที่ 5.1(ฉ) คือส่วนอ่านคำสั่งจะอ่านคำสั่งใหม่ที่ตำแหน่งใหม่เข้ามาโดยใช้ค่าในรีจิสเตอร์ CS และ IR2 เก็บไว้ที่รีจิสเตอร์ MAR เพื่อชี้ตำแหน่งของคำสั่งใหม่ที่ต้องการอ่าน จากนั้นอ่านรหัสดำเนินการของคำสั่งใหม่ 10'b บิตแรกเข้ามาเก็บไว้รีจิสเตอร์ MDR และส่ง

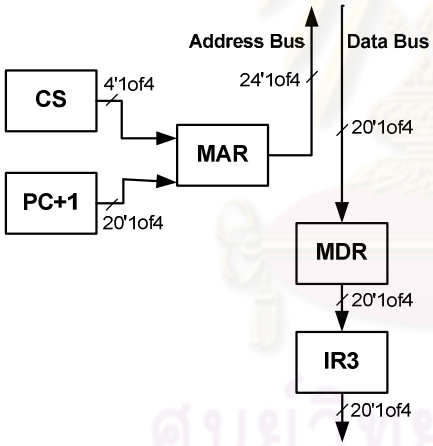
ต่อไปยังรีจิสเตอร์ IR1 เพื่อแปลความหมายของคำสั่งใหม่ว่ามีความยาวของรหัสดำเนินการสูงสุดเท่าไหร่ และต้องอ่านข้อมูลจากหน่วยความจำหรือไม่ จากนั้นทำงานตามความหมายของคำสั่งในขั้นตอนที่ 2 ถึง 6 ตามที่แปลได้



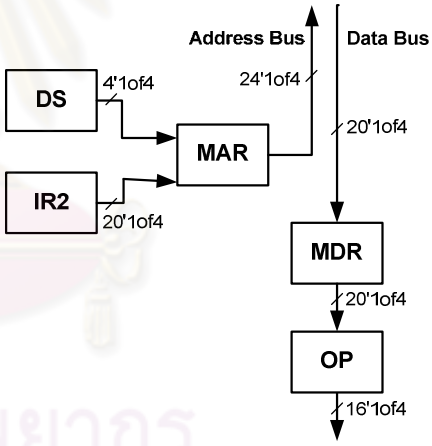
(ก) ขั้นตอนที่ 1



(ข) ขั้นตอนที่ 2

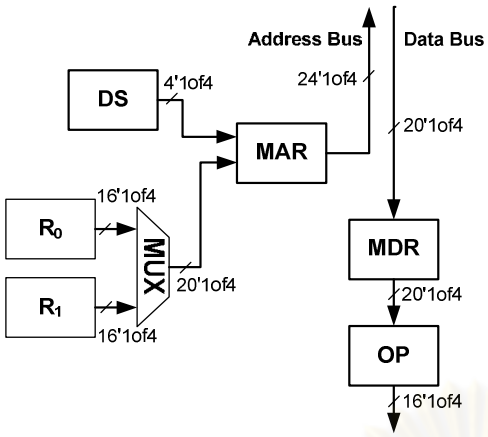


(ค) ขั้นตอนที่ 3

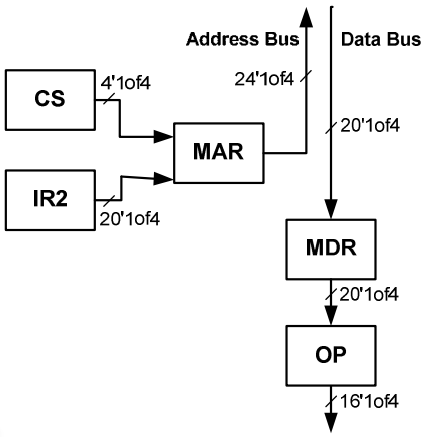


(ง) ขั้นตอนที่ 4

รูปที่ 5.1 ขั้นตอนการทำงานของส่วนอ่านคำสั่ง และส่วนแปลความหมายของคำสั่ง



(จ) ขั้นตอนที่ 5



(ฉ) ขั้นตอนที่ 6

รูปที่ 5.1 ขั้นตอนการทำงานของส่วนอ่านคำสั่ง และส่วนแปลความหมายของคำสั่ง (ต่อ)

หลังจากแปลความหมายของคำสั่งและทำงานในขั้นตอนที่ 1 ถึง 6 ตามเงื่อนไขของคำสั่งแล้ว จะสิ้นสุดการทำงานในส่วนนี้ และไปดำเนินการในส่วนประมวลผลต่อไป

5.3.2 **ส่วนประมวลผลและหน่วยคำนวณทางคณิตศาสตร์และตรรกะ**

ส่วนประมวลผล มีโครงสร้างวงจรดังรูปที่ 5.2 ทำหน้าที่ส่งค่ามาประมวลผลที่หน่วยคำนวณทางคณิตศาสตร์และตรรกะ หรือ ALU ในกลุ่มคำสั่งดำเนินการทางคณิตศาสตร์และตรรกะ แล้วเก็บผลลัพธ์ไว้ที่รีจิสเตอร์ตัวสะสม หรือ รีจิสเตอร์ ACC นอกจากนี้ ส่วนประมวลผลยังทำหน้าที่ส่งผ่านค่าในกลุ่มคำสั่งเคลื่อนย้ายข้อมูล สำหรับค่าในแฟลกรีจิสเตอร์คือ ZF (Zero Flag) CF (Carry Flag) OF (Overflow Flag) และ EI (Enable Interrupt Flag) จะถูกนำไปใช้พิจารณาเงื่อนไขในคำสั่งกลุ่มควบคุมการทำงาน โดยความหมายของค่าในแฟลกรีจิสเตอร์แต่ละแบบมีดังต่อไปนี้

ZF = 0 หมายความว่า ค่าในรีจิสเตอร์ ACC มีค่าไม่เท่ากับ 0

ZF = 1 หมายความว่า ค่าในรีจิสเตอร์ ACC มีค่าเท่ากับ 0

CF = 0 หมายความว่า ค่าในรีจิสเตอร์ ACC ไม่มีตัวทด

CF = 1 หมายความว่า ค่าในรีจิสเตอร์ ACC มีตัวทด

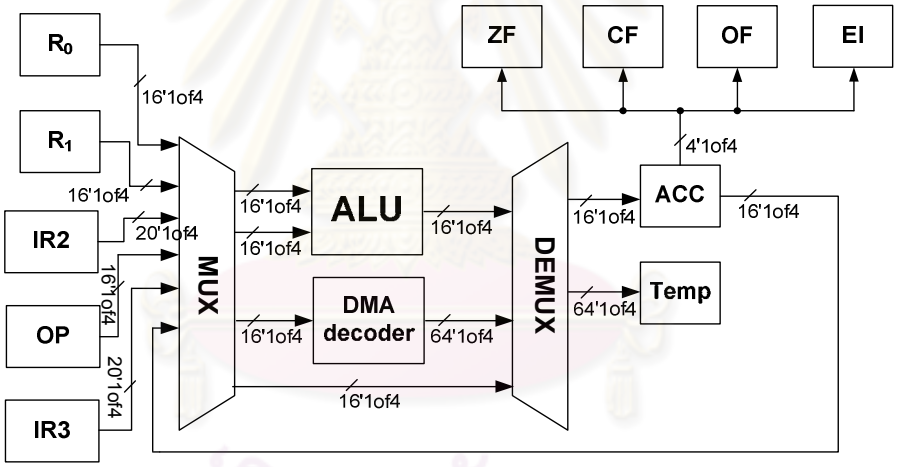
OF = 0 หมายความว่า ค่าในรีจิสเตอร์ ACC มีค่าไม่เกินจำนวนบิตสูงสุด

OF = 1 หมายความว่า ค่าในรีจิสเตอร์ ACC มีค่าเกินจำนวนบิตสูงสุด

EI = 0 หมายความว่า ค่าในรีจิสเตอร์ ACC ยังไม่สมบูรณ์และยังทำงานตามคำสั่งไม่เสร็จสิ้น ไม่อนุญาตให้ตอบสนองงานบริการอินเตอร์รัพท์

EI = 1 หมายความว่า ค่าในรีจิสเตอร์ ACC สมบูรณ์และเสร็จสิ้นการทำงานตามคำสั่งแล้ว อนุญาตให้ตอบสนองงานบริการอินเตอร์รัพท์ได้

จากรูปที่ 5.2 หากข้อมูลที่ใช้การประมวลผลคือค่าคงที่ที่กำหนดขึ้นเอง (#K) ส่วนประมวลผลจะนำค่าจากรีจิสเตอร์ IR2 มาประมวลผล หากข้อมูลที่ใช้ประมวลผลคือค่าที่อ่านมาจากรีจิสเตอร์ทั่วไป R0 หรือ R1 (Reg) ส่วนประมวลผลจะนำค่าจากรีจิสเตอร์ R0 หรือ R1 มาประมวลผล หากข้อมูลที่ใช้ประมวลผลคือค่าที่อ่านมาจากหน่วยความจำ (@K หรือ @Reg) ส่วนประมวลผลจะนำค่าจากรีจิสเตอร์ OP มาประมวลผล หากเป็นคำสั่งที่เรียกใช้ดีเอ็มเอ ส่วนประมวลผลจะนำค่าจากรีจิสเตอร์ IR2 และ IR3 มาประมวลผล ตัวอย่างการทำงานของส่วนประมวลผลในกลุ่มคำสั่งต่างๆมีดังนี้

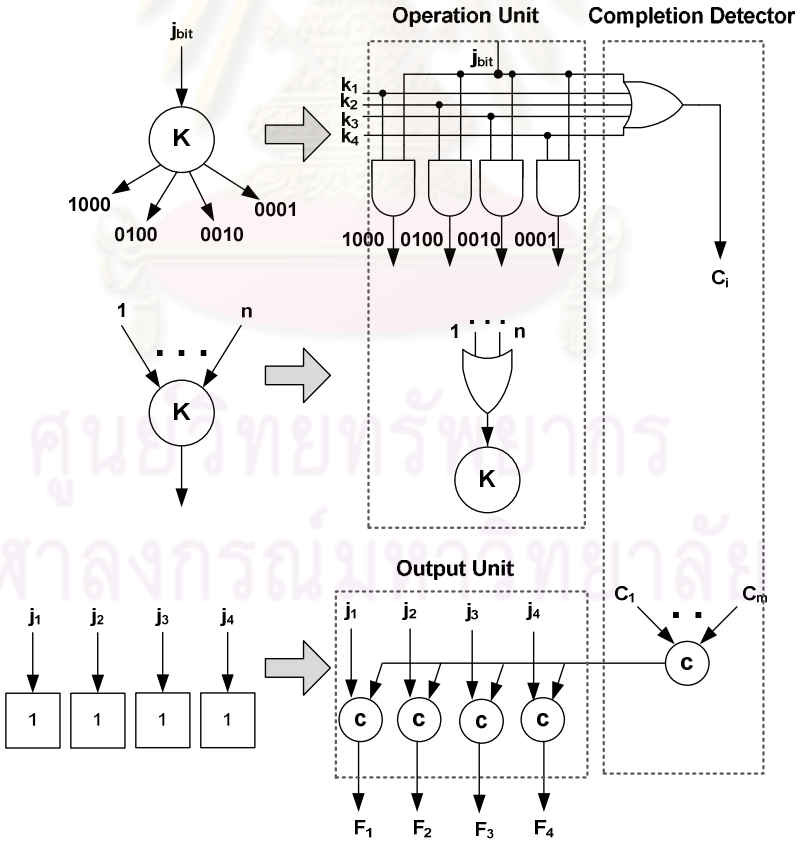


รูปที่ 5.2 ส่วนประมวลผล

- การทำงานในกลุ่มคำสั่งดำเนินการทางคณิตศาสตร์ เช่น การทำงานในคำสั่ง ADD A, @K มีขั้นตอนคือ ส่วนประมวลผลจะนำค่าจากรีจิสเตอร์ ACC และ รีจิสเตอร์ OP ไปประมวลผลที่วงจรบวกภายใน ALU และเก็บผลลัพธ์ที่ได้จากการประมวลผลไว้ในรีจิสเตอร์ ACC
- การทำงานในกลุ่มคำสั่งดำเนินการทางตรรกะ เช่น การทำงานในคำสั่ง SLL A มีขั้นตอนคือ ส่วนประมวลผลจะนำค่าจากรีจิสเตอร์ ACC ไปประมวลผลที่วงจรเลื่อนทุกบิตไปทางซ้ายแบบไม่คิดเครื่องหมายภายใน ALU และเก็บผลลัพธ์ที่ได้จากการประมวลผลไว้ในรีจิสเตอร์ ACC

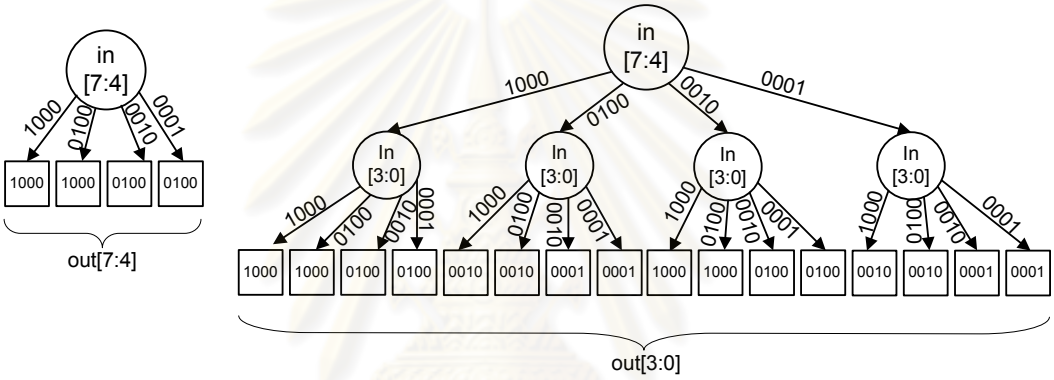
- การทำงานในกลุ่มคำสั่งเคลื่อนย้ายข้อมูล เช่น การทำงานในคำสั่ง LD A, #K มีขั้นตอนคือ ส่วนประมวลผลจะนำค่าจากรีจิสเตอร์ IR2 ส่งผ่านไปเก็บที่รีจิสเตอร์ ACC โดยไม่ผ่าน ALU สำหรับการทำงานในกลุ่มคำสั่งเคลื่อนย้ายข้อมูลซึ่งมีการเรียกใช้ดีเอ็มเอ เช่น การทำงานในคำสั่ง IN @K2,@K มีขั้นตอนคือ ส่วนประมวลผลจะนำค่าจากรีจิสเตอร์ IR2 และ IR3 ไปประมวลผลที่วงจรถอดรหัสคำสั่งดีเอ็มเอ (DMA Decoder) และเก็บผลลัพธ์ที่ได้จากการถอดรหัสไว้ในรีจิสเตอร์ Temp

- การทำงานในกลุ่มควบคุมการทำงาน เช่น การทำงานในคำสั่ง JZ Address มีขั้นตอนคือ ส่วนประมวลผลจะตรวจสอบค่าในแฟลกรีจิสเตอร์ ZF โดยหาก ZF มีค่าเท่ากับ 1 จะกลับไปทำงานในขั้นตอนที่ 6 ของส่วนอ่านคำสั่งและส่วนแปลของคำสั่ง แต่หาก ZF มีค่าเท่ากับ 0 จะสิ้นสุดการทำงานของคำสั่งนี้

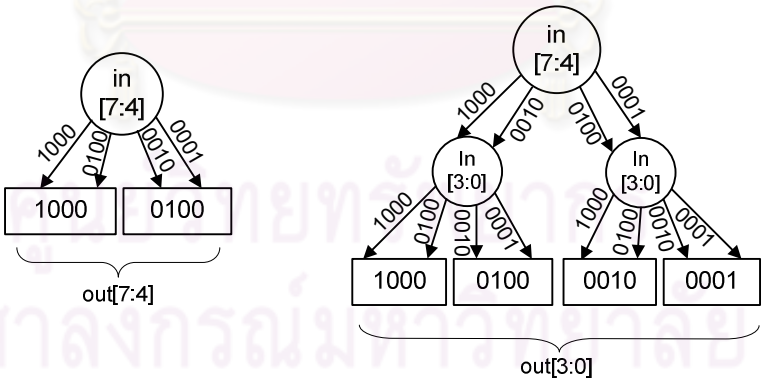


รูปที่ 5.3 การแปลงแผนภาพตัดสินใจแบบทวิภาคชนิดลดทอนอันดับเป็นวงจรหนึ่งในสี่

วงจรฟังก์ชันภายใน ALU ประกอบด้วย วงจรบวก วงจรลบ วงจรเพิ่มค่าหนึ่งค่า วงจรลดค่าหนึ่งค่า วงจรคูณ วงจรเลื่อนทุกบิตไปทางซ้ายแบบไม่คิดเครื่องหมาย วงจรเลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมาย วงจรเลื่อนทุกบิตไปทางซ้ายแบบคิดเครื่องหมาย วงจรเลื่อนทุกบิตไปทางขวาแบบคิดเครื่องหมาย วงจรกระทำลอจิกแอนด์ วงจรกระทำลอจิกออร์ วงจรกระทำลอจิกเอ็กคลูซีฟออร์ วงจรกระทำบิตต่อบิตแอนด์ วงจรกระทำบิตต่อบิตออร์ และวงจรกระทำบิตต่อบิตเอ็กคลูซีฟออร์ วงจรดังกล่าวทั้งหมดเข้ารหัสหนึ่งในสี่ การออกแบบวงจรใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับหรือ ROBDD ในการออกแบบ โดยหลักการแปลงแผนภาพ ROBDD เป็นวงจรหนึ่งในสี่แสดงดังรูปที่ 5.3

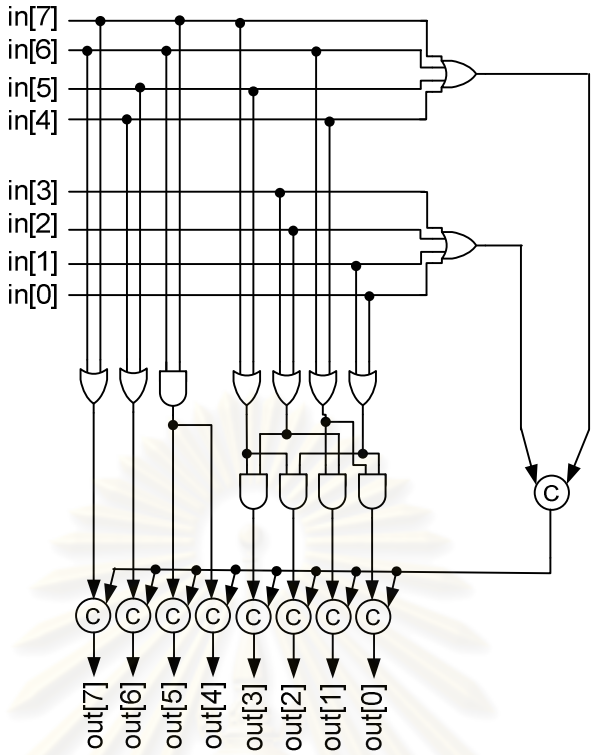


(ก) แผนภาพการตัดสินใจแบบทวิภาค



(ข) แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ

รูปที่ 5.4 การออกแบบวงจรหนึ่งในสี่ของฟังก์ชันเลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมาย ขนาด 8 บิต โดยใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ

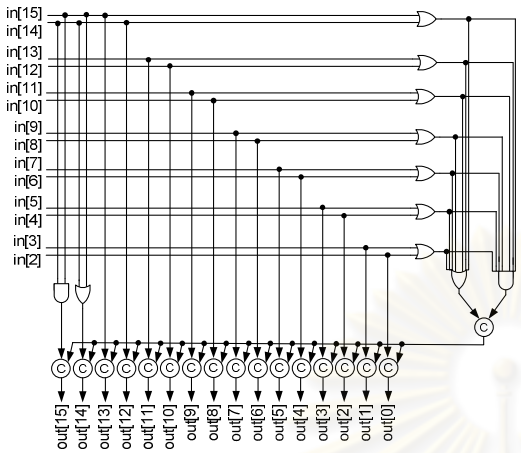


(ค) วงจรหนึ่งในสี่

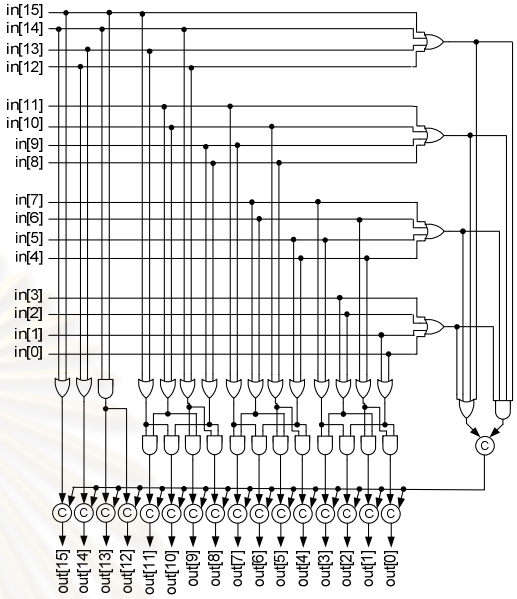
รูปที่ 5.4 การออกแบบวงจรหนึ่งในสี่ของฟังก์ชันเลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมาย ขนาด 8 บิต โดยใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ (ต่อ)

รูปที่ 5.4 แสดงการออกแบบวงจรหนึ่งในสี่ของวงจรฟังก์ชันเลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมายขนาด 8 บิต โดยใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ ซึ่งมีขั้นตอนตามขั้นตอนการออกแบบวงจรใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับในบทที่ 2 กล่าวคือ สร้างแผนภาพการตัดสินใจแบบทวิภาค โดยแทนค่าตัวแปรในฟังก์ชันเลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมาย และเขียนแจกแจงเป็นโครงสร้างที่มีอันดับชั้น ได้ดังรูปที่ 5.4(ก) โดยแต่ละกิ่งเป็นการแทนค่าของลอจิกในฟังก์ชัน เช่น ในวงจรที่สร้างเอาท์พุท out บิตที่ 3 ถึง 0 กิ่งซ้ายสุดของแผนภาพวงจรดังกล่าวหมายถึง เมื่อค่าลอจิกของอินพุท in บิตที่ 7 ถึง 4 มีค่าเท่ากับ 1000 และ อินพุท in บิตที่ 3 ถึง 0 มีค่าเท่ากับ 1000 แล้วค่าลอจิกของเอาท์พุท out บิตที่ 3 ถึง 0 จะมีค่าเท่ากับ 1000 เป็นต้น จากนั้นลดทอนอันดับแผนภาพดังกล่าวจนได้ดังรูปที่ 5.4(ข) ในขั้นตอนสุดท้ายจะแปลงแผนภาพที่ลดทอนแล้วเป็นวงจรหนึ่งในสี่ของฟังก์ชัน

เลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมาย ตามหลักการในรูปที่ 5.3 จนได้เป็นวงจรเข้ารหัสหนึ่งในสี่ดังรูปที่ 5.4(ค) เป็นอันเสร็จสิ้นขั้นตอน



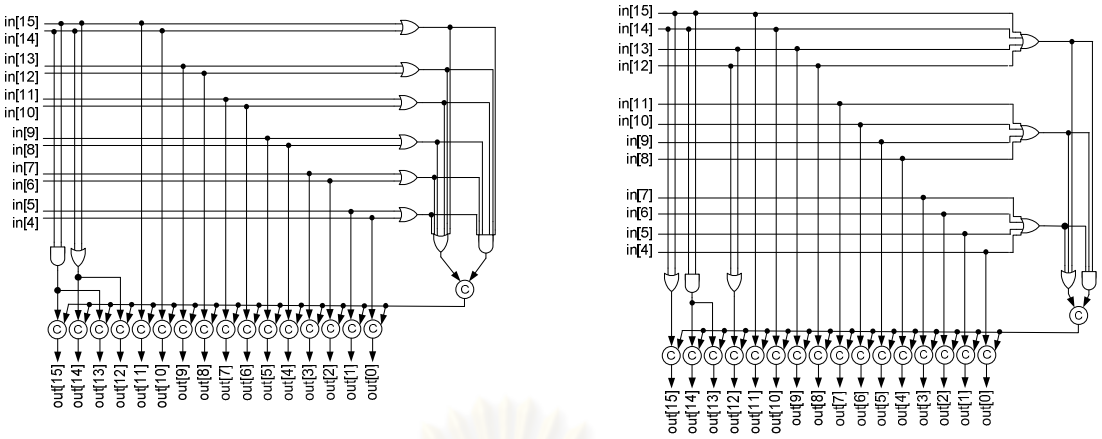
(ก) วงจรเลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมายเข้ารหัสรางคู่



(ข) วงจรเลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมายเข้ารหัสหนึ่งในสี่

รูปที่ 5.5 วงจรเลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมายเข้ารหัสรางคู่และเข้ารหัสหนึ่งในสี่ขนาด 16 บิต

วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ที่สร้างจากแผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ มีขนาดใหญ่กว่าวงจรฟังก์ชันเข้ารหัสรางคู่ซึ่งสร้างจากวิธีเดียวกัน ดังแสดงในรูปที่ 5.5 จะพบว่าวงจรเลื่อนทุกบิตไปทางขวาแบบไม่คิดเครื่องหมายแบบเข้ารหัสรางคู่ มีขนาดเล็กกว่าแบบเข้ารหัสหนึ่งในสี่ ทั้งนี้หากเป็นวงจรฟังก์ชันที่มีการคำนวณมากกว่าครั้งละ 1 หลัก เช่น วงจรเลื่อนทุกบิตไปทางขวาครั้งละ 2 บิตแบบไม่คิดเครื่องหมาย ซึ่งคำนวณครั้งละ 2 หลัก วงจรดังกล่าวแบบเข้ารหัสหนึ่งในสี่จะมีขนาดเล็กลง และมีขนาดใกล้เคียงกับวงจรแบบเข้ารหัสรางคู่ ดังแสดงในรูปที่ 5.6

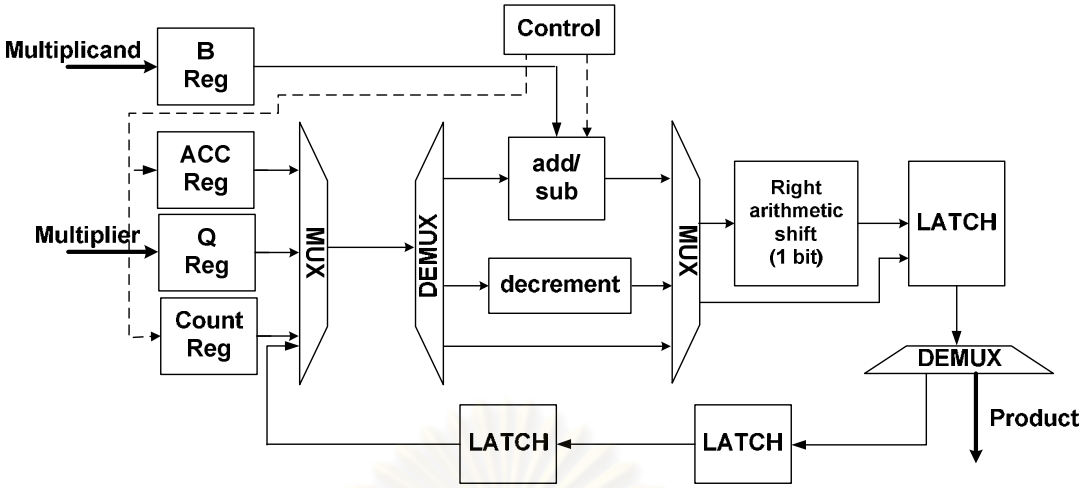


(ก) วงจรเลื่อนทุกบิตไปทางขวาครั้งละ 2 บิต แบบไม่คิดเครื่องหมายเข้ารหัสรางคู่

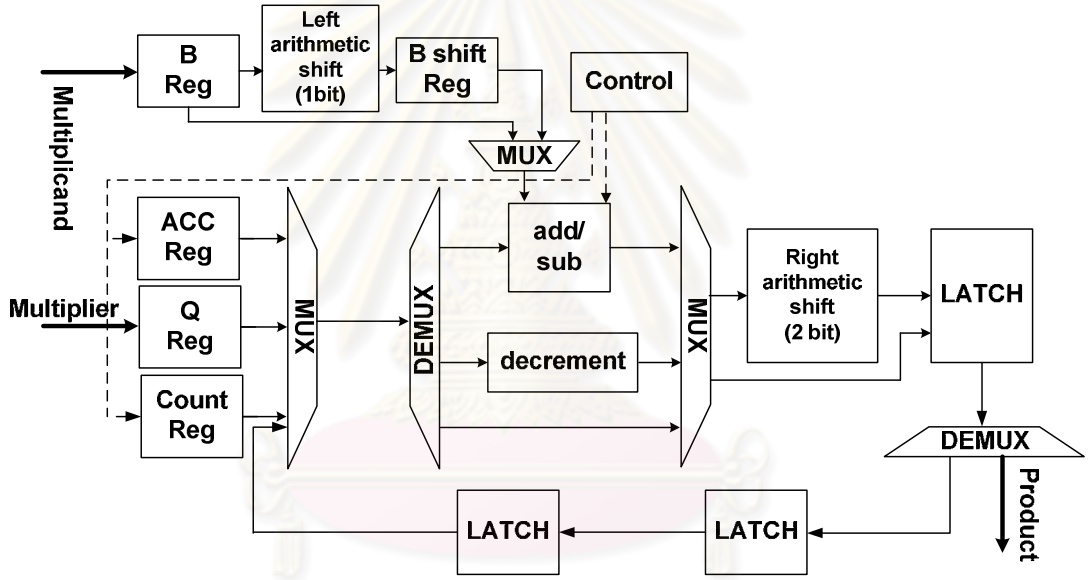
(ข) วงจรเลื่อนทุกบิตไปทางขวาครั้งละ 2 บิต แบบไม่คิดเครื่องหมายเข้ารหัสหนึ่งในสี่

รูปที่ 5.6 วงจรเลื่อนทุกบิตไปทางขวาครั้งละ 2 บิตแบบไม่คิดเครื่องหมาย เข้ารหัสรางคู่และเข้ารหัสหนึ่งในสี่ขนาด 16 บิต

ดังนั้นวงจรคูณเข้ารหัสหนึ่งในสี่ จะมีประสิทธิภาพดีขึ้นเมื่อเป็นวงจรคูณครั้งละ 2 หลักซึ่งประกอบด้วยวงจรเลื่อนทุกบิตไปทางขวาครั้งละ 2 บิตแบบไม่คิดเครื่องหมายเป็นส่วนประกอบ อีกทั้งหลักการทำงานของวงจรคูณครั้งละ 2 หลัก [25] จะทำงานจำนวน n/2 รอบเพื่อคูณตัวตั้ง (Multiplicand) และตัวคูณ(Multiplier) จำนวน n บิตออกมาเป็นผลคูณ (Product) ที่สมบูรณ์ แต่วงจรคูณครั้งละ 1 หลัก ต้องทำงานจำนวน n รอบเพื่อคูณตัวตั้งและตัวคูณจำนวน n บิตออกมาเป็นผลคูณที่สมบูรณ์ ในงานวิจัยนี้จึงออกแบบวงจรคูณบูทอัลกอลิทึม (Booth Algorithm) ครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ดังรูปที่ 5.7(ก) และพัฒนาเป็นวงจรคูณบูทอัลกอลิทึมครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ (1-of-4 Radix-4 Booth Multiplier with 1-of-4 Function Unit) ดังรูปที่ 5.7(ข) เพื่อให้การคำนวณข้อมูลเข้ารหัสหนึ่งในสี่ที่ใช้ในงานวิจัยมีประสิทธิภาพดีขึ้น โดยจะทดลองประสิทธิภาพของวงจรคูณทั้งสองแบบ เพื่อหาข้อสรุปทางด้านประสิทธิภาพที่ชัดเจนต่อไปในบทที่ 6



(ก) วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่



(ข) วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่

รูปที่ 5.7 วงจรคูณครั้งละ 1 หลักและวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่

หลักการทำงานของวงจรวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ที่ออกแบบ จะพิจารณาเงื่อนไขผ่านบิตที่เรียงต่อกันไปของตัวคูณจนครบทุกบิต ตามหลักการคูณของบูทอัลกอลิทึม โดยเมื่อเข้ารหัสหนึ่งในสี่แล้ว เงื่อนไขที่พิจารณาจะเป็นดังตารางที่ 5.1

ตารางที่ 5.1 เงื่อนไขการคูณครั้งละ 2 หลักของบูทอัลกอลิทึมเข้ารหัสหนึ่งในสี่

m_{i+1}	mi	m_{i-1}	เงื่อนไข	กระทำ
1000		01	เงื่อนไขที่ 5	ไม่ทำอะไร
1000		10	เงื่อนไขที่ 1	บวกด้วยตัวตั้ง
0100		01	เงื่อนไขที่ 1	บวกด้วยตัวตั้ง
0100		10	เงื่อนไขที่ 2	เลื่อนซ้ายตัวตั้ง 1 บิต, บวกด้วยตัวตั้ง
0010		01	เงื่อนไขที่ 4	เลื่อนซ้ายตัวตั้ง 1 บิต, ลบด้วยตัวตั้ง
0010		10	เงื่อนไขที่ 3	ลบด้วยตัวตั้ง
0001		01	เงื่อนไขที่ 3	ลบด้วยตัวตั้ง
0001		10	เงื่อนไขที่ 5	ไม่ทำอะไร

ตัวอย่างการคูณรหัสหนึ่งในสี่ 1000_1000_0100_0100 (5) ด้วยจำนวน 0010_0100_0001_0001 (-97) ตามเงื่อนไขการคูณของบูทอัลกอลิทึมเข้ารหัสหนึ่งในสี่ จากตารางที่ 5.1 เป็นดังนี้ โดยเมื่อเปรียบเทียบกับการคูณรหัสฐานสองแล้ว การคูณรหัสหนึ่งในสี่จำนวน 16 บิตจะเท่ากับคูณข้อมูลฐานสองขนาด 8 บิต (8'b บิต =16'1of4 บิต) จึงแบ่งการทำงานออกเป็น 4 รอบ (8/2=4)

เริ่มต้นจากเติมลอจิก 0 ที่บิตขวาสุดของตัวคูณ (บิตขวาสุดที่เติมมีเพียงบิตเดียวจึงเข้ารหัสบิตขวาสุดเป็นรหัสรางคู่ เนื่องจากการเข้ารหัสข้อมูลบิตเดียวด้วยรหัสหนึ่งในสี่จะทำให้วงจรที่ได้มีขนาดใหญ่ดังที่กล่าวไปแล้ว นอกเหนือจากนี้เป็นรหัสหนึ่งในสี่ทั้งหมด) จากนั้นพิจารณาบิตตัวคูณรวมทั้งบิตที่เติมเข้ามา และกระทำตามเงื่อนไข

<u>รอบที่ 1</u> 0010_0100_0001_**0001 | 01** (กระทำตามเงื่อนไขที่ 3)

1000_1000_1000_1000_1000_1000_1000_1000 ลบ

<u>1000_1000_0100_0100</u> ได้

<u>0001_0001_0001_0001_0001_0001_0010_0001</u>

จากนั้นเลื่อนขวาบิตตัวคูณรวมบิตที่เติมเข้ามา 2 บิต เพื่อตรวจสอบเงื่อนไขในบิตถัดไป ได้เป็น 1000_0010_0100_0001 | 10

<u>รอบที่ 2</u> 1000_0010_0100_**0001 | 10** (กระทำตามเงื่อนไขที่ 5)

<u>0001_0001_0001_0001_0001_0001_0010_0001</u>

จากนั้นเลื่อนขวาบิตตัวคูณรวมบิตที่เติมเข้ามา 2 บิต เพื่อตรวจสอบเงื่อนไขในบิตถัดไป ได้เป็น 1000_1000_0010_0100 | 10

รอบที่ 3 1000_1000_0010_**0100 | 10** (กระทำตามเงื่อนไขที่ 2)

0001_0001_0001_0001_0001_0001_0010_0001 บวก

1000_1000_0010_0010 ได้

1000_1000_1000_1000_0010_0100_0010_0001

จากนั้นเลื่อนขวาบิตตัวคูณรวมบิตที่เติมเข้ามา 2 บิต เพื่อตรวจสอบเงื่อนไขในบิตถัดไป ได้เป็น 1000_1000_1000_0010 | 01

รอบที่ 4 1000_1000_1000_**0010 | 01** (กระทำตามเงื่อนไขที่ 4)

1000_1000_1000_1000_0010_0100_0010_0001 ลบ

1000_1000_0010_0010 ได้

0001_0001_0001_0010_1000_0100_0010_0001 (-485)

ดังนั้น คูณรหัสหนึ่งในสี่ 1000_1000_0100_0100 (5) ด้วยจำนวน

0010_0100_0001_0001 (-97) ได้ผลลัพธ์เท่ากับ

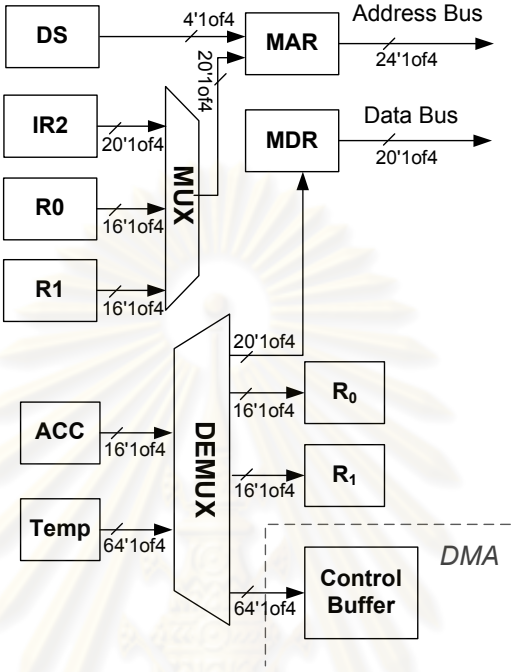
0001_0001_0001_0010_1000_0100_0010_0001 (-485)

5.3.3 ส่วนเขียนผลลัพธ์

ส่วนเขียนผลลัพธ์ ทำหน้าที่รับผลลัพธ์จากส่วนประมวลผลมาเขียนลงที่ รีจิสเตอร์หรือหน่วยความจำ ตามคำสั่ง โดยถ้าเป็นคำสั่งที่เรียกใช้งานดีเอ็มเอ ส่วนเขียนผลลัพธ์จะเขียนผลลัพธ์ลงที่บัฟเฟอร์ควบคุมของดีเอ็มเอซึ่งออกแบบไว้ในบทที่ 4 และดีเอ็มเอจะทำงานตามค่าในบัฟเฟอร์ควบคุมต่อไป โดยไม่ต้องเป็นภาระให้กับไมโครโพรเซสเซอร์ วงจรส่วนเขียนผลลัพธ์แสดงดังรูปที่ 5.8 ตัวอย่างการทำงานของส่วนเขียนผลลัพธ์ในคำสั่งต่างๆมีดังนี้

- คำสั่งที่ตัวรับผลลัพธ์เป็นรีจิสเตอร์ทั่วไป R0 หรือ R1 ส่วนเขียนผลลัพธ์จะนำค่าจากรีจิสเตอร์ ACC ไปเขียนผลลัพธ์ลงที่รีจิสเตอร์ทั่วไป R0 หรือ R1
- คำสั่งที่ตัวรับผลลัพธ์เป็นหน่วยความจำ และกำหนดให้เขียนผลลัพธ์ลงที่ตำแหน่งซึ่งเป็นค่าคงที่ที่กำหนดขึ้นเอง ส่วนเขียนผลลัพธ์จะใช้ค่าในรีจิสเตอร์ DS และ IR2 เก็บไว้

ที่รีจิสเตอร์ MAR เพื่อชี้ไปยังตำแหน่งที่ต้องการเขียนข้อมูลในหน่วยความจำ จากนั้นนำผลลัพธ์จากรีจิสเตอร์ ACC จำนวน 16’1of4 บิตเก็บไว้ที่ MDR และส่งต่อไปยังบัสระบบเพื่อเขียนผลลัพธ์ดังกล่าวไว้ในหน่วยความจำ



รูปที่ 5.8 ส่วนเขียนผลลัพธ์

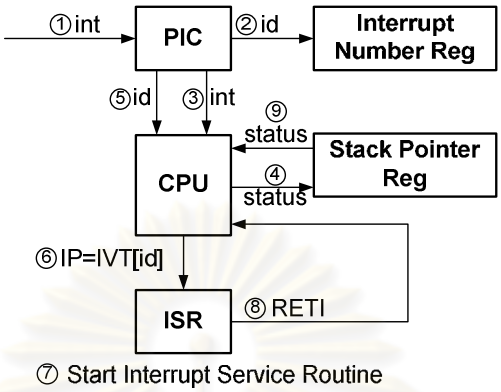
- คำสั่งที่ตัวรับผลลัพธ์เป็นหน่วยความจำ และกำหนดให้เขียนผลลัพธ์ลงที่ตำแหน่งที่ค่าในรีจิสเตอร์ R0 หรือ R1 กำหนด ส่วนเขียนผลลัพธ์จะใช้ค่าในรีจิสเตอร์ DS และ R0 หรือ R1 เก็บไว้ที่รีจิสเตอร์ MAR เพื่อชี้ไปยังตำแหน่งที่ต้องการเขียนข้อมูลในหน่วยความจำ จากนั้นนำผลลัพธ์จากรีจิสเตอร์ ACC จำนวน 16’1of4 บิตเก็บไว้ที่ MDR และส่งต่อไปยังบัสระบบเพื่อเขียนผลลัพธ์ดังกล่าวไว้ในหน่วยความจำ

- คำสั่งที่เรียกใช้งานดีเอ็มเอ ส่วนเขียนผลลัพธ์จะนำค่าจากรีจิสเตอร์ Temp ไปเขียนผลลัพธ์ลงที่บัฟเฟอร์ควบคุมของดีเอ็มเอ เพื่อใช้เป็นค่าเริ่มต้นการทำงานของดีเอ็มเอต่อไป

5.3.4 **ส่วนบริการอินเตอร์รัพท์**

ส่วนบริการอินเตอร์รัพท์ปรับปรุงจากส่วนบริการอินเตอร์รัพท์ในไมโครโพรเซสเซอร์เวอร์ชันเดิม [4] ทำหน้าที่รองรับการใช้งานอินเตอร์รัพท์ โดยมีหลักการคือ เมื่อมี

สัญญาณอินเตอร์รัพท์จากอุปกรณ์ต่อพ่วงเข้ามา CPU จะถูกขัดจังหวะการทำงานเดิม ให้ไปทำงานใหม่ตามที่กำหนดก่อน และเมื่อทำงานดังกล่าวเสร็จสิ้น CPU จะกลับไปทำงานเดิมต่อไป



รูปที่ 5.9 ขั้นตอนบริการอินเตอร์รัพท์

รูปที่ 5.9 แสดงขั้นตอนการทำงานของส่วนบริการอินเตอร์รัพท์ ซึ่งมีดังต่อไปนี้

1. เมื่อมีสัญญาณอินเตอร์รัพท์ (int) จากอุปกรณ์เข้ามา จะส่งสัญญาณอินเตอร์รัพท์ไปที่ PIC เพื่อตรวจสอบว่าเป็นสัญญาณอินเตอร์รัพท์จากอุปกรณ์หมายเลข (id) ใด
2. ส่งค่าหมายเลขของอุปกรณ์ ไปเก็บไว้ที่รีจิสเตอร์ Interrupt Number
3. ส่งค่าสัญญาณอินเตอร์รัพท์ไปที่ CPU
4. CPU เก็บสถานะ (Status) ของการทำงานปัจจุบันใส่ SP (Stack Pointer)
5. PIC ส่งค่าหมายเลขของอุปกรณ์มาที่ CPU
6. CPU คำนวณตำแหน่งของงาน (Instruction Pointer: IP) ใหม่ที่ต้องไปทำ ซึ่งเป็นงานของอินเตอร์รัพท์ (Interrupt Service Routine: ISR) โดยตรวจสอบตำแหน่งจากตารางตำแหน่งอินเตอร์รัพท์ (Interrupt Vector Table: IVT)
7. CPU เริ่มทำงานของอินเตอร์รัพท์
8. เมื่อสิ้นสุดงานอินเตอร์รัพท์ จะส่งสัญญาณ RETI (Return from Interrupt) ไปที่ CPU
9. CPU ดึงสถานะเก่าในข้อ 4. จาก SP คืนมา และทำงานต่อ

5.3.5 ส่วนควบคุม

ส่วนควบคุมไมโครโพรเซสเซอร์ทำหน้าที่ควบคุมส่วนต่างๆของไมโครโพรเซสเซอร์ให้ทำงานร่วมกันตามคำสั่งที่ได้รับได้อย่างถูกต้อง ส่วนควบคุมออกแบบโดยใช้อุปกรณ์ไปป์ไลน์ควบคุมสัญญาณร้องขอและสัญญาณตอบรับ เส้นทางการรับส่งข้อมูลภายในไมโครโพรเซสเซอร์ถูกควบคุมโดยวงจรควบคุมซึ่งสร้างจากกราฟบรรยายการเปลี่ยนสัญญาณ และใช้โปรแกรม Petrify สร้างวงจร เช่นเดียวกับการออกแบบตัวควบคุมบัสในบทที่ 3 ที่ได้กล่าวไปแล้ว

5.4 การทำงานของไมโครโพรเซสเซอร์

ส่วนอ่านคำสั่งของไมโครโพรเซสเซอร์จะอ่านคำสั่งจากแคชเก็บคำสั่ง (Instruction Cache) จากนั้นส่วนแปลความหมายของคำสั่งจะแปลความหมายของคำสั่งว่าต้องทำงานใดบ้าง แล้วส่งค่าไปประมวลผลที่ส่วนประมวลผลตามงานของคำสั่งที่แปลไว้ เมื่อประมวลผลเสร็จสิ้น ส่วนเขียนผลลัพธ์จะเขียนผลลัพธ์ที่ได้จากการประมวลผลไว้ที่รีจิสเตอร์ หรือหน่วยความจำ หรือบัฟเฟอร์ควบคุมของดีเอ็มเอ ซึ่งเป็นการสิ้นสุดการทำงานของคำสั่ง จากนั้นไมโครโพรเซสเซอร์จะเริ่มการทำงานของคำสั่งใหม่ โดยอ่านคำสั่งใหม่ที่ตำแหน่งถัดไปในแคชเก็บคำสั่ง และทำงานในคำสั่งถัดไปตามขั้นตอนที่ได้กล่าวมาอย่างต่อเนื่อง

บทที่ 6

การทดลองบัสระบบ

ในบทนี้กล่าวถึงการการทดลองบัสระบบ ซึ่งประกอบด้วย วัตถุประสงค์ของการทดลอง วิธีการทดลอง การทดลอง และผลการทดลอง โดยทำการทดลองวัดประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่ และวงจรคูณเข้ารหัสหนึ่งในสี่ภายในหน่วยดำเนินการทางคณิตศาสตร์และตรรกะ ด้านการใช้พลังงาน ขนาดวงจรที่ใช้ และความเร็วในการทำงาน สุดท้ายจะโปรแกรมวงจรบัสระบบเข้ารหัสหนึ่งในสี่ลงเอฟพีจีเอ เพื่อทดลองการทำงานของบัสระบบร่วมกับองค์ประกอบ รายละเอียดของแต่ละส่วนมีดังต่อไปนี้

6.1 วัตถุประสงค์ของการทดลอง

1. ทดลองบัสระบบเข้ารหัสหนึ่งในสี่ เพื่อตรวจสอบการทำงานให้มีความถูกต้อง และวัดประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่เทียบกับบัสเข้ารหัสรางคู่ ในด้านของการใช้พลังงาน ขนาดวงจรที่ใช้ และความเร็วในการทำงาน

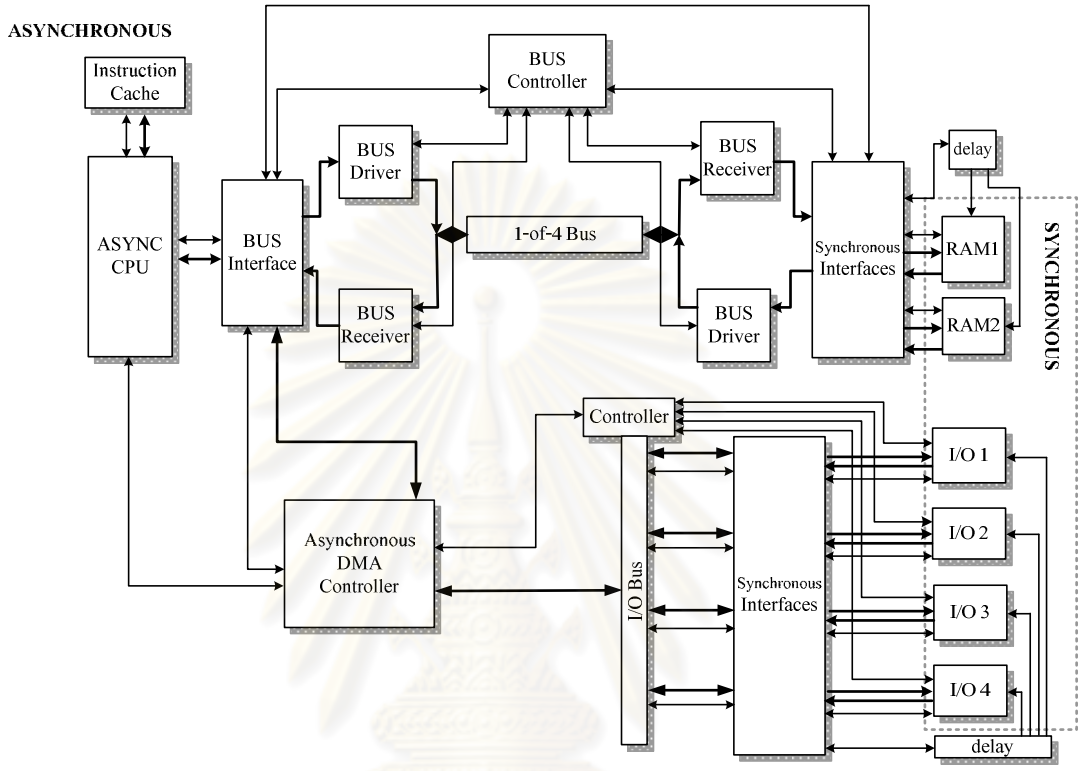
2. ทดลองวงจรคูณเข้ารหัสหนึ่งในสี่ภายในหน่วยดำเนินการทางคณิตศาสตร์และตรรกะ เพื่อตรวจสอบการทำงานให้มีความถูกต้อง และวัดประสิทธิภาพของวงจรคูณใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ เปรียบเทียบกับวงจรคูณใช้วงจรฟังก์ชันเข้ารหัสรางคู่ ในด้านของการใช้พลังงาน ขนาดวงจรที่ใช้ และความเร็วในการทำงาน

3. ทดลองบัสระบบร่วมกับองค์ประกอบ เพื่อตรวจสอบการทำงานร่วมกันให้มีความถูกต้องบนเอฟพีจีเอ

6.2 วิธีการทดลอง

วิธีการทดลอง แบ่งออกเป็นการสังเคราะห์วงจรบัสระบบแบบอสมวารเข้ารหัสหนึ่งในสี่และองค์ประกอบ ซึ่งแสดงดังรูปที่ 6.1 ที่ได้ออกแบบในบทที่ 3 4 และ 5 ผ่านโปรแกรม Xilinx ISE 11.1 [21] สำหรับวงจรควบคุมที่ออกแบบไว้จะสร้างวงจรโดยใช้โปรแกรม Petrify 4.2 แล้วจึงค่อยนำมาสังเคราะห์วงจร เมื่อสังเคราะห์วงจรเสร็จสิ้น จะนำวงจรที่สังเคราะห์มาอิมพลิเมนท์ลงบนเอฟพีจีเอ โดยใช้โปรแกรม Xilinx ISE 11.1 เช่นกัน แล้วจึงดูผลการทำงานของ

วงจรด้วยการจำลองการทำงานอิงเวลาด้วยโปรแกรม ModelSim XE 6.4b [22] สุดท้ายจะนำวงจรที่ทำงานถูกต้องแล้ว ไปโปรแกรมลงบนเอฟพีจีเอโดยใช้โปรแกรม Xilinx ISE 11.1 รายละเอียดของแต่ละขั้นตอนมีดังต่อไปนี้



รูปที่ 6.1 บัสระบบเข้ารหัสหนึ่งในสี่และองค์ประกอบ

6.2.1 การสร้างวงจรโดยใช้โปรแกรม Petrify 4.2

ตัวอย่างการใช้โปรแกรม Petrify [18] ออกแบบวงจรหน่วงเวลา เป็นดังนี้

i) สร้างไฟล์นามสกุล .g ไว้ในโฟลเดอร์ bin ซึ่งอยู่ในโฟลเดอร์ของโปรแกรม Petrify และตั้งชื่อไฟล์ตามต้องการ ในตัวอย่างนี้ใช้ชื่อ delay.g

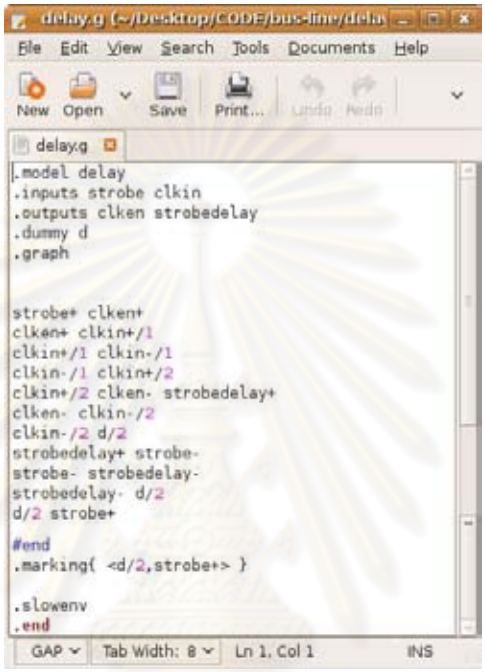
ii) เขียนภาษาอธิบายพฤติกรรมของวงจรตามกราฟ STG ไว้ในไฟล์ .g ในข้อ i. ในตัวอย่างนี้เขียนอธิบายพฤติกรรมตามกราฟ STG ในรูปที่ 3.9(ค) จากบทที่ 3 ไว้ที่ไฟล์ delay.g ได้ดังรูปที่ 6.2(ก)

iii) ใช้คำสั่งสร้างไฟล์ .csc และไฟล์ .eqn ผ่าน terminal โดยพิมพ์

./petrify ชื่อไฟล์.g –cg –atopt –egn ชื่อไฟล์.eqn –o ชื่อไฟล์.csc ↵ enter

ใช้คำสั่งสร้างไฟล์ .ps จากไฟล์ .g และไฟล์ .csc ผ่าน terminal โดยพิมพ์

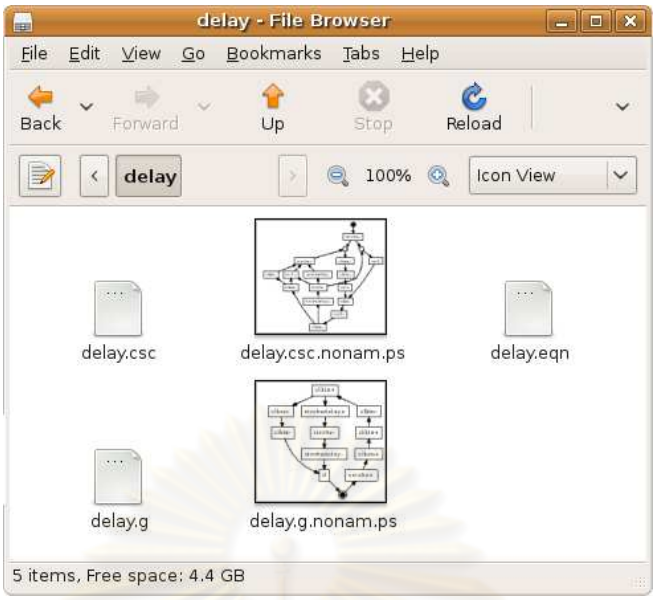
./draw_astg -nonames -noinfo -bw ชื่อไฟล์.g -o ชื่อไฟล์.g.nonam.ps ↵ enter ./draw_astg -nonames -noinfo -bw ชื่อไฟล์.csc -o ชื่อไฟล์.csc.nonam.ps ↵ enter



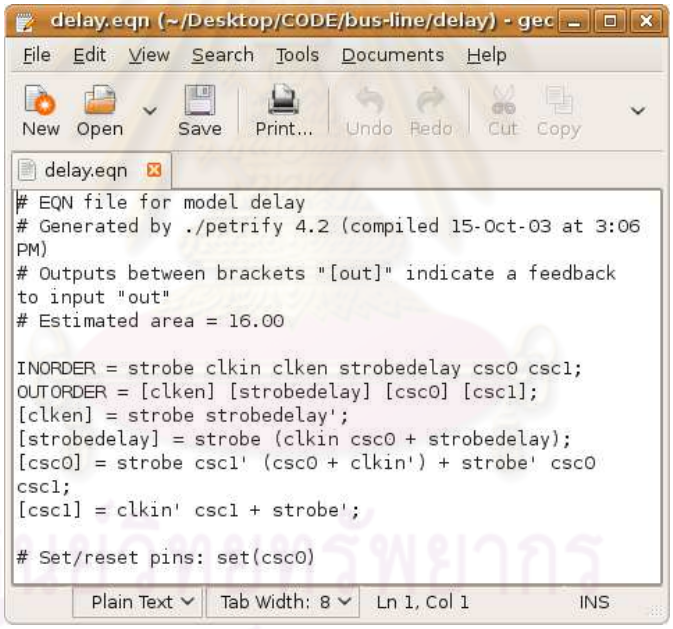
(ก) เขียนภาษาอธิบายพฤติกรรมของวงจรตามกราฟ STG

(ข) ใช้คำสั่งสร้างไฟล์วงจร

รูปที่ 6.2 การออกแบบวงจรหน่วงเวลาโดยใช้โปรแกรม Petrify

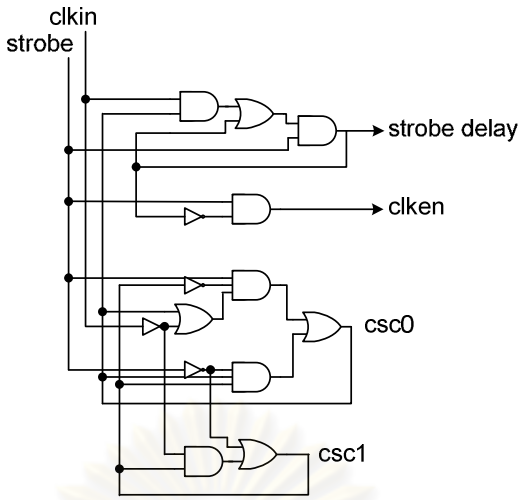


(ค) ไฟล์วงจรที่ได้จากโปรแกรม Petrify



(ง) ไฟล์สมการบูลีน (.eqn)

รูปที่ 6.2 การออกแบบวงจรหน่วงเวลาโดยใช้โปรแกรม Petrify (ต่อ)



(จ) วงจรระดับเกตซึ่งแปลงจากไฟล์สมการบูลีน

รูปที่ 6.2 การออกแบบวงจรหน่วงเวลาโดยใช้โปรแกรม Petrify (ต่อ)

ในตัวอย่างนี้พิมพ์คำสั่งดังกล่าวได้เป็น

./petrify delay.g –cg –atopt –egn delay.eqn –o delay.csc ↵ enter ดังรูปที่ 6.2 (ข)

./draw_astg -nonames -noinfo -bw delay.g -o delay.g.nonam.ps ↵ enter

./draw_astg -nonames -noinfo -bw delay.csc -o delay.csc.nonam.ps ↵ enter

iv) หลังจากใช้คำสั่งในข้อ iii. จะได้ไฟล์เพิ่มขึ้นมาในโฟลเดอร์ bin ประกอบด้วย

ไฟล์ .csc คือไฟล์แก้ไขความซ้ำซ้อนของสถานะบนกราฟ STG (Complete State Coding: CSC) โดยจะสร้างสัญญาณ csc เพิ่มเพื่อให้สถานะบนกราฟ STG ไม่ซ้ำกัน

ไฟล์ .eqn คือไฟล์สมการบูลีนที่มีพฤติกรรมตามกราฟ STG

ไฟล์ delay.g.nonam.ps คือไฟล์ภาพ STG ของไฟล์ .g

ไฟล์ delay.csc.nonam.ps คือไฟล์ภาพ STG ของไฟล์ .csc

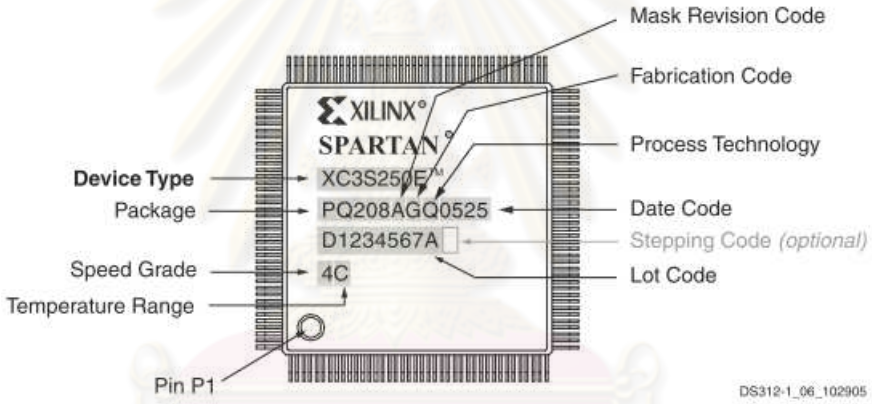
ในตัวอย่างนี้จะได้ไฟล์ในโฟลเดอร์ bin ประกอบด้วย delay.csc delay.eqn delay.g.nonam.ps delay.csc.nonam.ps ดังรูปที่ 6.2(ค)

v) เขียนวงจรจากสมการบูลีน .eqn ในตัวอย่างนี้จะเขียนวงจรจากไฟล์สมการบูลีน delay.eqn ในรูปที่ 6.2(ง) ได้เป็นวงจรหน่วงเวลาดังรูปที่ 6.2(จ) โดยอาจพิจารณาเพิ่มเติมสัญญาณ reset ไว้ในวงจรตามความเหมาะสม

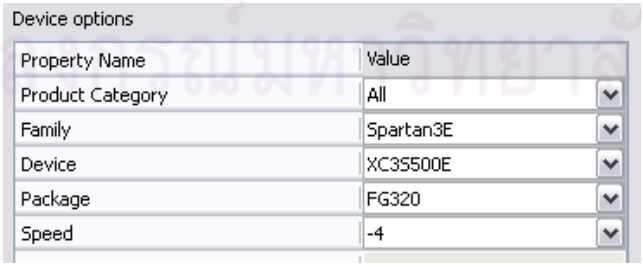
6.2.2 การสังเคราะห์วงจรด้วยโปรแกรม Xilinx ISE 11.1

1. เปิดไฟล์โปรเจควงจรที่ต้องการสังเคราะห์ ซึ่งออกแบบด้วยภาษาเวอริลอคไว้ โดยเปิดโปรแกรม Xilinx เลือกเมนู Open Project จากนั้นเลือกโปรเจคที่ต้องการเปิดแล้วกด Open

2. กำหนดอุปกรณ์เอฟพีจีเอที่ต้องการใช้สังเคราะห์และอิมพลิเมนท์วงจร โดยเลือกเมนู Project>Design Properties และทำการกำหนดคุณสมบัติของอุปกรณ์ซึ่งสามารถสังเกตได้จากชิพบนบอร์ด ตัวอย่างชิพและคุณสมบัติแสดงดังรูปที่ 6.3 [23] สำหรับงานวิจัยนี้ใช้อุปกรณ์เอฟพีจีเอ Xlinx SPARTAN-3E เบอร์ XC3S500EFG320 ซึ่งกำหนดคุณสมบัติของอุปกรณ์ได้ดังรูปที่ 6.4

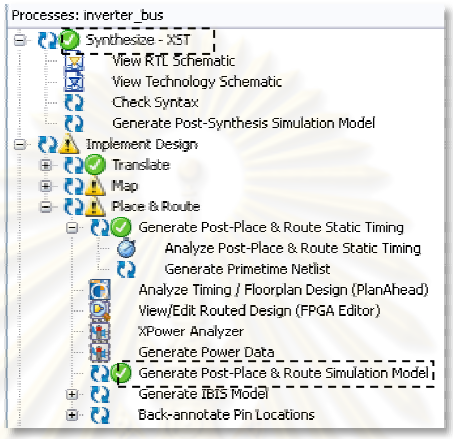


รูปที่ 6.3 ชิพบนบอร์ดเอฟพีจีเอ SPARTAN

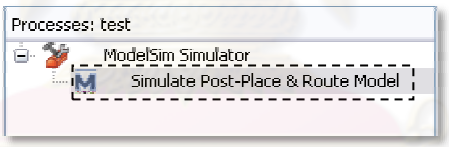


รูปที่ 6.4 การกำหนดคุณสมบัติของอุปกรณ์เอฟพีจีเอ Xlinx SPARTAN-3E เบอร์ XC3S500EFG320

3. สังเคราะห์วงจรที่ต้องการ โดยการคลิกเลือกไฟล์วงจรที่ต้องการสังเคราะห์ (.v) ซึ่งเรียงลำดับชั้นอยู่ภายในโปรเจคในหน้าต่าง Hierarchy ด้านซ้ายมือ แล้วดับเบิ้ลคลิกที่ Synthesis-XST ในหน้าต่าง Processes ด้านซ้ายมือ ดังแสดงในรูปที่ 6.5(ก) โปรแกรมจะเริ่มต้นสังเคราะห์วงจร และแสดงผลการสังเคราะห์ในหน้าต่าง Console หากการสังเคราะห์สมบูรณ์จะแสดงข้อความ Process "Synthesis" completed successfully



(ก) ตัวเลือกสังเคราะห์และอิมพลิเมนท์วงจร



(ข) ตัวเลือกจำลองการทำงานแบบอิงเวลา

รูปที่ 6.5 ตัวเลือกสังเคราะห์ อิมพลิเมนท์และจำลองการทำงานแบบอิงเวลา ในโปรแกรม Xlinx ISE

6.2.3 การอิมพลิเมนท์วงจรด้วยโปรแกรม Xilinx ISE 11.1

เมื่อสังเคราะห์วงจรสมบูรณ์แล้ว จึงทำการอิมพลิเมนท์โดยคลิกที่ Implement Design>Place&Route และดับเบิ้ลคลิกที่ Generate Post-Place & Route Simulation Model ในหน้าต่าง Processes ด้านซ้ายมือ ดังแสดงในรูปที่ 6.5(ก) โปรแกรมจะเริ่มต้นอิมพลิเมนท์วงจรแบบ Place and Route ซึ่งนำผลของค่าเวลาล่าช้า (Delay Time) ต่างๆในวงจรที่ออกแบบมาคำนวณร่วมกับการจัดวางเกตและเชื่อมสายบนอุปกรณ์เอฟพีจีเอ ผลการอิมพลิเมนท์จะแสดง

ในหน้าต่าง Console หากการอิมพลิเมนท์สมบูรณ์จะแสดงข้อความ Process "Generate Post-Place & Route Simulation Model" completed successfully

6.2.4 การจำลองการทำงานแบบอิงเวลาด้วยโปรแกรม ModelSim XE 6.4b

1. เมื่ออิมพลิเมนท์วงจรสมบูรณ์แล้ว จึงเลือกไฟล์ที่ใช้ทดลองการทำงานของวงจรซึ่งออกแบบด้วยภาษาเวอริลอคไว้ โดยเลือก Source for: Place & Route Simulation จากนั้นคลิกขวาในหน้าต่าง Hierarchy เลือก Add Source เลือกไฟล์ที่ใช้ทดลองที่ต้องการเปิดแล้วกด Open

2. จำลองการทำงานแบบอิงเวลา โดยการคลิกเลือกไฟล์ที่ใช้ทดลองวงจรที่เลือกในข้อ 1 ซึ่งแสดงในหน้าต่าง Hierarchy ด้านซ้ายมือ แล้วคลิกที่ ModelSim Simulator ในหน้าต่าง Processes ด้านซ้ายมือ และดับเบิ้ลคลิกที่ Simulate Post-Place & Route Model ดังแสดงในรูปที่ 6.5(ข) โปรแกรมจะเริ่มต้นจำลองการทำงานแบบอิงเวลา และเปิดโปรแกรม ModelSim ขึ้นมาโดยอัตโนมัติผลการจำลองการทำงานจะแสดงในหน้าต่าง wave ของโปรแกรม ModelSim

ผลจำลองการทำงานของวงจรย่อยอาจมีความถูกต้อง แต่เมื่อนำวงจรย่อยมาเชื่อมต่อกันเป็นวงจรขนาดใหญ่ขึ้น ผลจำลองการทำงานอาจไม่ถูกต้องตามพฤติกรรมวงจรที่ได้ออกแบบไว้ได้ เนื่องจากวงจรขนาดใหญ่จะถูกทำการสังเคราะห์และอิมพลิเมนท์วงจรใหม่ทั้งหมด กล่าวคือสังเคราะห์และอิมพลิเมนท์วงจรย่อยใหม่ทั้งหมดด้วย ซึ่งอาจทำให้วงจรย่อยมีรูปแบบการจัดวางเกตและเชื่อมสายบนอุปกรณ์เอฟพีจีเอเปลี่ยนไปจากเดิม ส่งผลกระทบถึงค่าเวลาล่าช้าในเกตและสาย ทำให้วงจรย่อยทำงานผิดพลาดได้ ดังนั้นจึงต้องแยกโครงสร้างที่ได้จากการสังเคราะห์และอิมพลิเมนท์วงจรย่อยที่มีความถูกต้องไว้ไม่ให้ถูกแก้ไข (Design Preservation) และนำโครงสร้างวงจรย่อยเหล่านั้นมาเชื่อมต่อกันเป็นวงจรขนาดใหญ่ภายหลัง

วิธีการแยกโครงสร้างดังกล่าวสามารถใช้คำสั่ง New Partition ในโปรแกรม Xilinx แยกโครงสร้างได้ โดยคลิกขวาที่ไฟล์วงจรย่อยที่ต้องการแยกแล้วเลือก New Partition จะปรากฏสัญลักษณ์  หน้าไฟล์วงจรย่อยที่ต้องการแยกโครงสร้าง และสัญลักษณ์  หน้าไฟล์วงจรขนาดใหญ่ซึ่งประกอบด้วยวงจรย่อยที่ต้องการแยกโครงสร้าง แล้วจึงเข้าสู่ขั้นตอนอิมพลิเมนท์และจำลองการทำงานอิงเวลาอีกครั้ง เพื่อตรวจสอบความถูกต้องของวงจรขนาดใหญ่ภายหลังการใช้คำสั่งแยกโครงสร้าง

6.2.5 การโปรแกรมวงจรลงเอฟพีจีเอด้วยโปรแกรม Xilinx ISE 11.1

ขั้นตอนการโปรแกรมลงเอฟพีจีเอประกอบด้วย

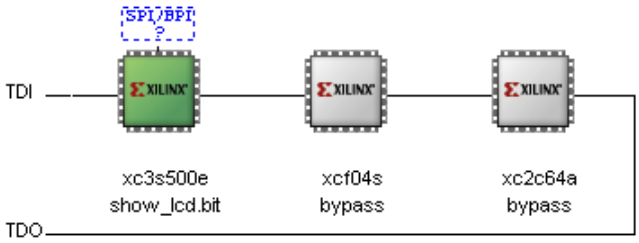
1. อิมพลิเมนท์ไฟล์ที่ต้องการโปรแกรมลงเอฟพีจีเอแบบ Post-Place & Route ตามขั้นตอนข้อ 6.2.3 และดับเบิ้ลคลิกที่เมนู Generate Program File ในหน้าต่าง Process เพื่อสร้างไฟล์ .bit สำหรับดาวน์โหลดลงเอฟพีจีเอ

2. ดับเบิ้ลคลิกที่เมนู User Constraints>I/O Planing (PlanAhead) ในหน้าต่าง Process จะเปิดโปรแกรม Plan Ahead 11.1 ขึ้นมา ให้เชื่อมต่อพอร์ทของเอฟพีจีเอเข้ากับ พอร์ทอินพุทหรือเอาท์พุทของวงจรที่ออกแบบไว้ โดยคลิกที่พอร์ทของวงจร และแก้ไขค่าที่หน้าต่าง I/O Properties ในส่วนของ Site ให้เป็นชื่อพอร์ทของเอฟพีจีเอที่ต้องการ เช่น พอร์ทของสวิทช์ พอร์ทของจอแอลซีดี เป็นต้น แล้วกด Apply ดังรูปที่ 6.6 เมื่อกำหนดพอร์ทครบแล้ว ให้กด save แล้วปิดโปรแกรม Plan Ahead

3. คลิกที่ไฟล์ในข้อ 1. แล้วดับเบิ้ลคลิกเมนู Configure Target Device> Manage Configuration Project (iMPACT) ในหน้าต่าง Process ของโปรแกรม Xilinx จะเปิดโปรแกรม ISE iMPACT ขึ้นมา ให้ดับเบิ้ลคลิกที่ Boundary Scan และคลิกขวาที่พื้นที่ว่างสีขาวด้านซ้ายมือเลือก Initialize Chain จะปรากฏอุปกรณ์ขึ้นดังรูปที่ 6.7 ให้กด Yes จากนั้นจะมีหน้าต่าง Assign New Configuration File ขึ้นมา ให้เลือกไฟล์ .bit ที่ได้จากข้อ 1. แล้วกด OK

4. สุดท้ายคลิกขวาที่อุปกรณ์เอฟพีจีเอที่ต้องการโปรแกรมวงจร จากที่ปรากฏดังรูปที่ 6.7 แล้วเลือก Program เพื่อเริ่มการโปรแกรมวงจรลงเอฟพีจีเอ เมื่อโปรแกรมเสร็จสมบูรณ์จะแสดงข้อความว่า Program Succeeded

รูปที่ 6.6 กำหนดพอร์ทเอฟพีจีเอ



รูปที่ 6.7 โปรแกรมลงเอฟพีจีเอ

6.3 การทดลองและผลการทดลอง

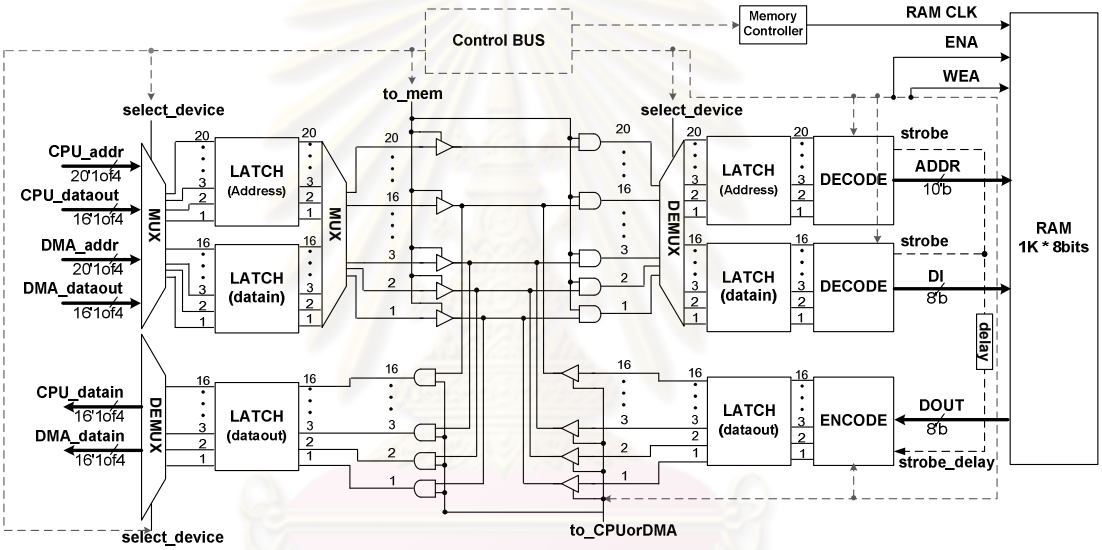
ทำการทดลองวัดประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่ และวงจรคูณเข้ารหัสหนึ่งในสี่ภายในหน่วยดำเนินการทางคณิตศาสตร์และตรรกะ ด้านการใช้พลังงาน ขนาดวงจรที่ใช้ และความเร็วในการทำงาน สุดท้ายจะโปรแกรมวงจรบัสระบบเข้ารหัสหนึ่งในสี่ลงเอฟพีจีเอ เพื่อทดลองการทำงานของบัสระบบร่วมกับองค์ประกอบ รายละเอียดของแต่ละการทดลองและผลการทดลองมีดังต่อไปนี้

6.3.1 การทดลองวัดประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่

วัดประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่ผ่านผลจำลองการทำงานอิงเวลา โดยเปรียบเทียบกับประสิทธิภาพของบัสเข้ารหัสรางคู่ที่ใช้โครงสร้างเดียวกันกับบัสเข้ารหัสหนึ่งในสี่เวอร์ชันปรับปรุงในบทที่ 3 แต่ปรับปรุงส่วนเข้ารหัส ส่วนถอดรหัส ส่วนพักข้อมูลให้ทำงานกับข้อมูลเข้ารหัสรางคู่ได้ บัสทั้งสองที่นำมาทดลองเชื่อมต่อกับหน่วยความจำหนึ่งตัวดังรูปที่ 6.8 วิธีการทดลองคือให้บัสระบบทั้งสองทำงานในคำสั่งเดียวกันคือ LD ST IN และOUT ซึ่งเป็นคำสั่งที่เรียกใช้งานบัสระบบทั้งสิ้น จากนั้นนำผลการทดลองการทำงานมาเปรียบเทียบในด้านของพลังงาน ขนาดวงจร และความเร็ว แล้วจึงสรุปผลการทดลอง คำสั่งที่ใช้ในการทดลองประกอบไปด้วยคำสั่งรับส่งข้อมูลผ่านบัสระบบจำนวน 8 คำสั่ง รายละเอียดของคำสั่งมีดังนี้

```
0       ORG     0000H
        LD      A,#FF       ;Reg A = FF
1       ST      A,@3FF      ; M[3FF] = Reg A
        LD      A,#7F       ; Reg A = 7F
```

```
2       ST      A,@1FF      ; M[1FF] = Reg A
3       LD      A,@3FF      ; Reg A = M[3FF]
4       LD      A,@1FF      ; Reg A = M[1FF]
5       OUT     4,@03F      ; M[3F] = I/O[4]  (I/O[4] = 3F)
6       OUT     3,@01F      ; M[1F] = I/O[3]  (I/O[3] = 1F)
7       IN      1,@03F      ; I/O[1] = M[3F]
8       IN      2,@01F      ; I/O[2] = M[1F]
        END
```



รูปที่ 6.8 บัสระบบเชื่อมต่อกับหน่วยความจำหนึ่งตัวที่ใช้ทดลอง

- **เปรียบเทียบการเปลี่ยนสถานะสัญญาณของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่**

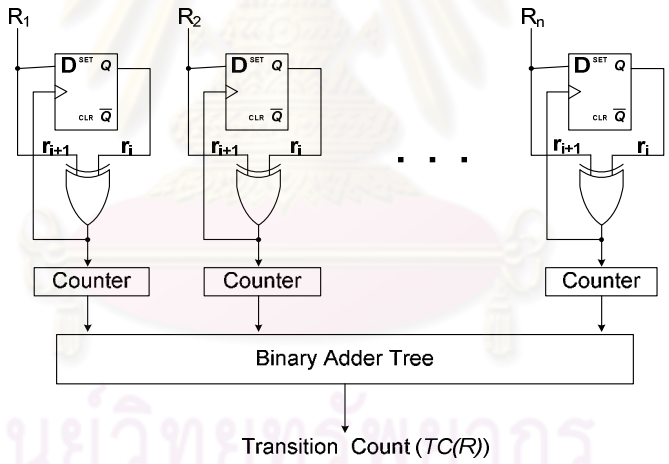
การเปรียบเทียบการเปลี่ยนสถานะสัญญาณ ทำได้โดยนับจำนวนการเปลี่ยนสถานะของสัญญาณ (Transition Count) จากสถานะ 0 เป็นสถานะ 1 หรือจากสถานะ 1 เป็นสถานะ 0 ซึ่งสามารถคำนวณได้จากสมการที่ 6.1 โดยให้ $TC(R)$ คือ จำนวนการเปลี่ยนสถานะสัญญาณของวงจร R ที่ประกอบด้วยสายสัญญาณ R_1 ไปจนถึง R_n โดย n คือจำนวนของ

สายสัญญาณ r_i คือสัญญาณก่อนหน้าของสายสัญญาณ R_n และ r_{i+1} คือสัญญาณถัดไปของสายสัญญาณ R_n

$$TC(R) = \sum_{i=1}^{n}(r_i \oplus r_{i+1}) \quad (6.1)$$

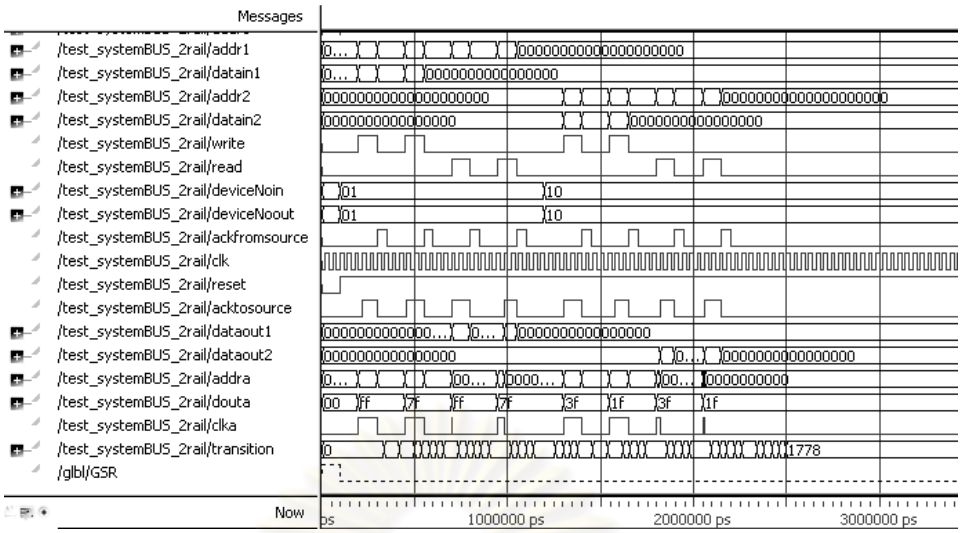
หากพิจารณาจากสายสัญญาณหนึ่งสาย จะพบว่าถ้าสัญญาณก่อนหน้าและสัญญาณถัดไปมีสถานะไม่เหมือนกัน ซึ่งหมายถึงมีการเปลี่ยนสถานะของสัญญาณในสายสัญญาณ จำนวนค่าการเปลี่ยนสถานะของสัญญาณหนึ่งสายจะถูกนับเพิ่มขึ้น 1 ค่า และเมื่อรวมจำนวนการเปลี่ยนสถานะสัญญาณของทุกสายสัญญาณในวงจร จะได้จำนวนการเปลี่ยนสถานะสัญญาณทั้งหมดของวงจร

วงจรนับจำนวนการเปลี่ยนสถานะของสัญญาณ (Transition Count Circuit) [20] ประกอบด้วย ดีฟลิปฟล็อป (D-Flip Flop) เอ็กคลูซีฟออร์เกต (Exclusive-Or Gate) วงจรนับ (Counter) วงจรบวกแบบแผนภาพต้นไม้ (Binary Adder Tree) มีโครงสร้างดังรูปที่ 6.9



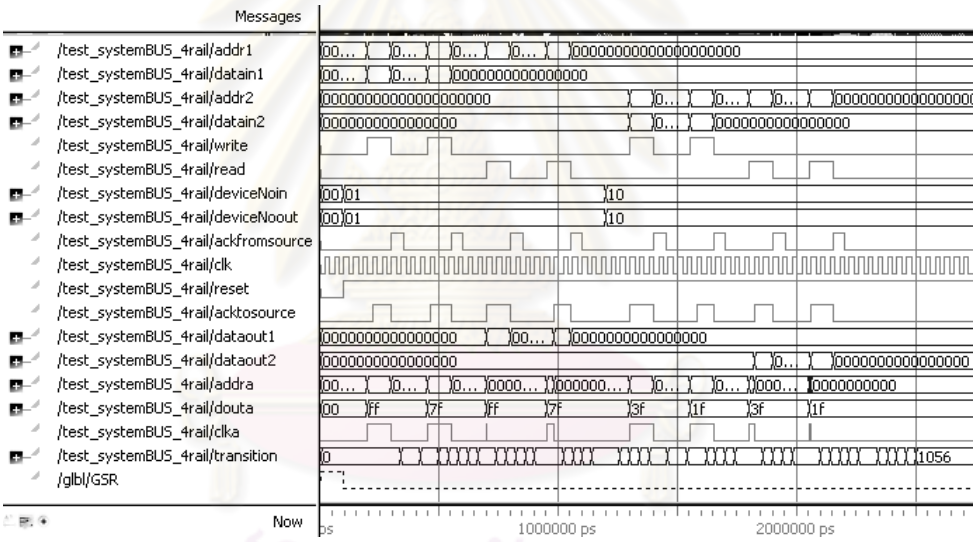
รูปที่ 6.9 วงจรนับจำนวนการเปลี่ยนสถานะของสัญญาณ

เมื่อนำสายสัญญาณในบัสระบบทั้งหมดต่อกับวงจรนับจำนวนการเปลี่ยนสถานะของสัญญาณ จะได้เอาท์พุทเป็นจำนวนการเปลี่ยนสถานะสัญญาณของบัสระบบ ผลการจำลองการทำงานแบบอิงเวลาของการทดลองนี้กับ 8 คำสั่งที่ได้กล่าวไว้ข้างต้นและจำนวนการเปลี่ยนสถานะสัญญาณจากหน้าต่าง wave ของโปรแกรม ModelSim แสดงดังรูปที่ 6.10



(ก) ผลการจำลองการทำงานแบบอิงเวลาและ

จำนวนการเปลี่ยนสถานะสัญญาณของบัสระบบเข้ารหัสรางคู่



(ข) ผลการจำลองการทำงานแบบอิงเวลาและ

จำนวนการเปลี่ยนสถานะสัญญาณของบัสระบบเข้ารหัสหนึ่งในสี่

รูปที่ 6.10 ผลการจำลองการทำงานแบบอิงเวลาของบัสระบบและจำนวนการเปลี่ยน

สถานะสัญญาณของบัสระบบ

ตารางที่ 6.1 สรุปจำนวนการเปลี่ยนสถานะสัญญาณของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่ของการทำงานทั้ง 8 คำสั่ง โดยบัสระบบเข้ารหัสรางคู่มีการเปลี่ยนสถานะสัญญาณรวม 1,778 ครั้ง และบัสระบบเข้ารหัสหนึ่งในสี่มีการเปลี่ยนสถานะสัญญาณรวม

1,056 ครั้ง โดยเมื่อพิจารณาจะพบว่า บัสระบบเข้ารหัสหนึ่งในสี่ลดการเปลี่ยนสถานะของสัญญาณลงได้ 40.61% โดยประมาณ เมื่อเปรียบเทียบกับบัสระบบเข้ารหัสรางคู่

ตารางที่ 6.1 การเปลี่ยนสถานะสัญญาณของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่

Bus	Transition count		
	Control	Address/Data	Total
Dual-rail Bus	334	1444	1778
1-of-4 Bus	334	722	1056

- **เปรียบเทียบพลังงานที่ถูกใช้โดยประมาณในการทำงานของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่**

ประมาณค่าพลังงานที่ถูกใช้ในการทำงาน ของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่บนเอฟพีจีเอXlinx SPARTAN-3E เบอร์ XC3S500EFG320 ผ่านโปรแกรม XPower Analyzer 11 มีขั้นตอนการใช้โปรแกรมดังกล่าวดังนี้

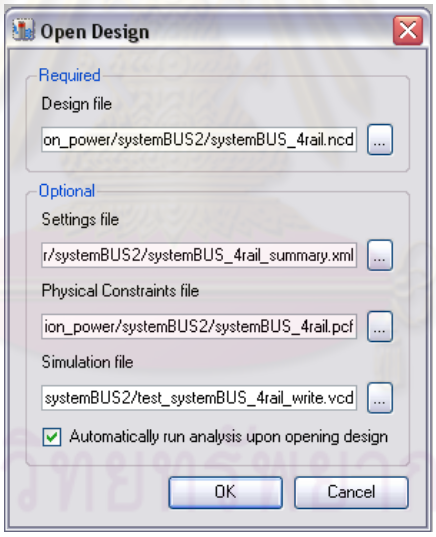
1. แก้ไขไฟล์ที่ใช้ทดลองเพื่อเพิ่มคำสั่งคำนวณพลังงาน โดยเปิดไฟล์ที่ใช้ทดลองวงจรในหัวข้อที่ 6.2.4 ด้วยโปรแกรม Xilinx จากนั้นแก้ไขไฟล์ดังกล่าวโดยเพิ่มบรรทัดโค้ดคำนวณพลังงาน แทรกไว้ระหว่างบรรทัด initial begin กับ // Initialize Inputs ได้ดังนี้

```
initial begin
$dumpfile("ชื่อไฟล์เอาท์พุท.vcd");
$dumpvars(1, ชื่อโมดูลที่ต้องการคำนวณพลังงาน.uut);
// Initialize Inputs
```

2. จำลองการทำงานแบบอิงเวลาโดยมีขั้นตอนเช่นเดียวกับที่อธิบายในหัวข้อที่ 6.2.4 คือคลิกเลือกไฟล์ที่ใช้ทดลองวงจรที่แก้ไขในข้อ 1 ซึ่งแสดงในหน้าต่าง Hierarchy ด้านซ้ายมือ แล้วคลิกที่ ModelSim Simulator ในหน้าต่าง Processes ด้านซ้ายมือ และดับเบิ้ลคลิกที่ Simulate Post-Place & Route Model โปรแกรมจะเริ่มต้นจำลองการทำงานแบบอิงเวลาและเปิดโปรแกรม ModelSim ขึ้นมาโดยอัตโนมัติ เมื่อจำลองการทำงานแบบอิงเวลาเสร็จสิ้น

ในโฟลเดอร์ที่อยู่ของไฟล์ที่ใช้ทดลองจะมีไฟล์นามสกุล .vcd ชื่อเดียวกับที่กำหนดไว้ในโค้ดข้อ 1 เพิ่มขึ้นมา

3. เปิดโปรแกรม Xpower Analyzer เลือกเมนู File>Open Design และเลือกเปิดไฟล์นามสกุล .ncd .xml .pcf และไฟล์นามสกุล .vcd ที่ได้จากข้อ 2 ใส่ลงในช่องของ Design file, Setting file, Physical Constraints file และSimulation file ตามลำดับ เพื่อใช้ในการคำนวณพลังงาน ดังแสดงในรูปที่ 6.11 จากนั้นกด OK โปรแกรมจะเริ่มคำนวณพลังงานจากไฟล์ที่เลือกไว้ เมื่อคำนวณเสร็จสิ้นจะขึ้นข้อความว่า Design 'ชื่อไฟล์.ncd' opened successfully ในหน้าต่าง Progress และสรุปผลการใช้พลังงานแบบย่อในแท็บ Table View นอกจากนี้ยังสามารถดูรายงานแบบสมบูรณ์ (Advance Report) ได้ โดยเลือกที่เมนู Tools>Generate Advance Report โปรแกรมจะสร้างไฟล์รายงานแบบสมบูรณ์นามสกุล .pwr ขึ้นมาภายในโฟลเดอร์ที่อยู่ของไฟล์ที่ใช้ทดลอง



รูปที่ 6.11 ไฟล์ที่ใช้ในการคำนวณพลังงานบนโปรแกรม XPower Analyzer

ผลรายงานแบบสมบูรณ์ที่ได้จากโปรแกรม XPower Analyzer ซึ่งประมาณค่าพลังงานที่ถูกใช้ในการทำงานทั้ง 8 คำสั่งที่ได้กล่าวไว้ข้างต้น ของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่ แสดงดังรูปที่ 6.12

Power summary		I(mA)	P(mW)
Total estimated power consumption			86.63
Total Vccint	1.20V	26.48	31.77
Total Vccaux	2.50V	18.10	45.25
Total Vcco25	2.50V	3.84	9.61
	BRAM		0.00
	Clocks		0.01
	IO		4.97
	Logic		0.12
	Signals		0.47
Quiescent Vccint	1.20V	25.88	31.05
Quiescent Vccaux	2.50V	18.00	45.00
Quiescent Vcco25	2.50V	2.00	5.00
Package power limits, ambient	25C		2873.56
	250 LFM		3640.78
	500 LFM		3865.98
	750 LFM		4032.26

(ก) รายงานพลังงานที่ถูกใช้ในการทำงานของบัสระบบเข้ารหัสรางคู่

Power summary		I(mA)	P(mW)
Total estimated power consumption			85.98
Total Vccint	1.20V	26.32	31.58
Total Vccaux	2.50V	18.09	45.23
Total Vcco25	2.50V	3.67	9.17
	BRAM		0.00
	Clocks		0.01
	IO		4.48
	Logic		0.08
	Signals		0.37
Quiescent Vccint	1.20V	25.87	31.04
Quiescent Vccaux	2.50V	18.00	45.00
Quiescent Vcco25	2.50V	2.00	5.00
Package power limits, ambient	25C		2873.56
	250 LFM		3640.78
	500 LFM		3865.98
	750 LFM		4032.26

(ข) รายงานพลังงานที่ถูกใช้ในการทำงานของบัสระบบเข้ารหัสหนึ่งในสี่

รูปที่ 6.12 ผลรายงานของโปรแกรม XPower Analyzer

จากผลรายงานของโปรแกรม XPower Analyzer ในรูปที่ 6.12(ก) คำนวณพลังงานโดยประมาณที่ใช้ในการทำงานของบัสระบบเข้ารหัสรางคู่ได้ดังนี้

คำนวณพลังงานสแตติก :

$P_{Static} = (VI)_{Quiescent\ Vccint} + (VI)_{Quiescent\ Vccaux} + (VI)_{Quiescent\ Vcco25}$

$P_{Static} = (1.20 \times 25.88) + (2.50 \times 18.00) + (2.50 \times 2.00)$

$P_{Static} = 31.05 + 45.00 + 5.00$

$P_{Static} = 81.05\ mW$

คำนวณพลังงานไดนามิก :

$P_{Dynamic} = P_{BRAM} + P_{Clocks} + P_{IO} + P_{Logic} + P_{Signals}$

$P_{Dynamic} = 0.00 + 0.01 + 4.97 + 0.12 + 0.47$

$P_{Dynamic} \approx 5.58$ mW

คำนวณพลังงานรวม :

$P_{Total} = P_{Static} + P_{Dynamic}$

$P_{Total} = 81.05 + 5.58$

$P_{Total} = 86.63$ mW

จากผลรายงานของโปรแกรม XPower Analyzer ในรูปที่ 6.12(ข) คำนวณพลังงานโดยประมาณที่ใช้ในการทำงานของบัสระบบเข้ารหัสหนึ่งในสี่ได้ดังนี้

คำนวณพลังงานสแตติก :

$P_{Static} = (VI)_{Quiescent\ Vccint} + (VI)_{Quiescent\ Vccaux} + (VI)_{Quiescent\ Vcco25}$

$P_{Static} = (1.20 \times 25.87) + (2.50 \times 18.00) + (2.50 \times 2.00)$

$P_{Static} = 31.04 + 45.00 + 5.00$

$P_{Static} = 81.04$ mW

คำนวณพลังงานไดนามิก :

$P_{Dynamic} = P_{BRAM} + P_{Clocks} + P_{IO} + P_{Logic} + P_{Signals}$

$P_{Dynamic} = 0.00 + 0.01 + 4.48 + 0.08 + 0.37$

$P_{Dynamic} = 4.94$ mW

คำนวณพลังงานรวม :

$P_{Total} = P_{Static} + P_{Dynamic}$

$P_{Total} = 81.04 + 4.94$

$P_{Total} = 85.98$ mW

จากการคำนวณการใช้พลังงานดังแสดงข้างต้น สรุปพลังงานโดยประมาณที่ใช้ในการทำงานของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่ได้ดังตารางที่ 6.2 โดยเมื่อพิจารณาจะพบว่า บัสระบบเข้ารหัสหนึ่งในสี่ลดการใช้พลังงานลงได้ 0.75% ต่อวินาทีโดยประมาณ เมื่อเปรียบเทียบกับบัสระบบเข้ารหัสรางคู่

ตารางที่ 6.2 พลังงานที่ถูกใช้โดยประมาณของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่

Bus	Estimated Power Consumption (mW)		
	Static Power	Dynamic Power	Total Power
Dual-rail Bus	81.05	5.58	86.63
1-of-4 Bus	81.04	4.94	85.98

- **เปรียบเทียบขนาดวงจรของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่**

เปรียบเทียบขนาดวงจรของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่จากผลประมาณค่าการใช้อุปกรณ์บนเอฟพีจีเอ (Device Utilization Summary) ของวงจรในโปรแกรม Xilinx มีขั้นตอนคือ คลิกที่ไฟล์วงจรที่ต้องการประมาณค่าการใช้อุปกรณ์ จากนั้นดับเบิ้ลคลิกที่ Design Summary/Reports ในหน้าต่าง Processes ค่าประมาณของการใช้อุปกรณ์จะแสดงในแท็บ Design Summary บนตาราง Device Utilization Summary (Estimated Values) โดยประกอบไปด้วยจำนวนสไลด์ (Slice) จำนวนตารางค้นหาแบบสี่อินพุท (4 Input LUTs) จำนวนพอร์ทอินพุทเอาท์พุท (Bonded IOBs) ที่ถูกใช้ไปบนเอฟพีจีเอ

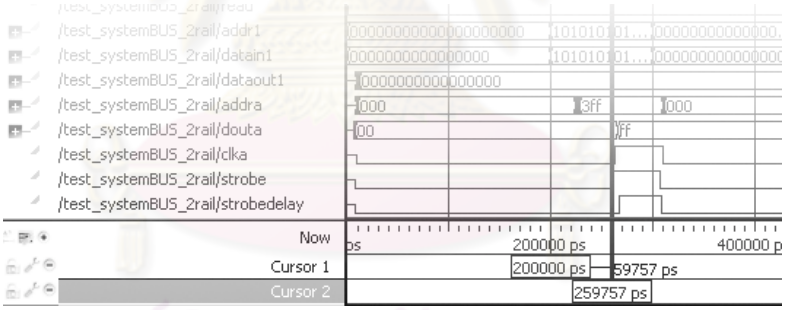
ตารางที่ 6.3 แสดงค่าการใช้อุปกรณ์ของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่ โดยบัสระบบเข้ารหัสรางคู่ใช้สไลด์จำนวน 221 ตัว ตารางค้นหาแบบสี่อินพุทจำนวน 388 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุทรวม 142 ตัว บัสระบบเข้ารหัสหนึ่งในสี่ใช้สไลด์จำนวน 203 ตัว ตารางค้นหาแบบสี่อินพุทจำนวน 359 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุท รวม 142 ตัว โดยเมื่อพิจารณาจะพบว่า บัสระบบเข้ารหัสหนึ่งในสี่ลดขนาดวงจรลงได้ 7.72% โดยประมาณ เมื่อเปรียบเทียบกับบัสระบบเข้ารหัสรางคู่

ตารางที่ 6.3 การใช้อุปกรณ์ของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่

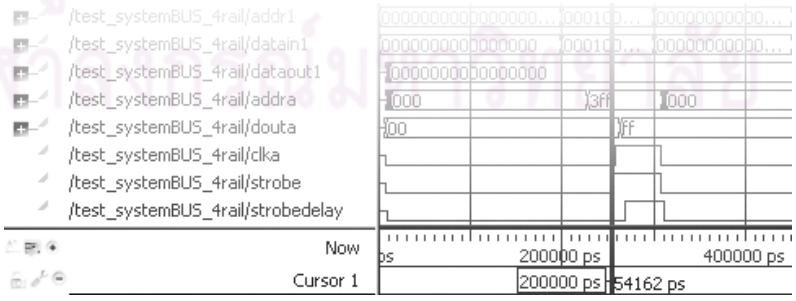
Bus	Device Utilization Summary		
	Slices	4 input LUTs	bonded IOBs
Dual-rail Bus	221	388	142
1-of-4 Bus	203	359	142

- **เปรียบเทียบความเร็วของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่**

เปรียบเทียบความเร็วของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่ โดยวัดความเร็วในการทำงานของบัสทั้งสองซึ่งแสดงบนผลการจำลองการทำงานแบบอิงเวลาที่หน้าต่าง wave ของโปรแกรม ModelSim เวลาที่ใช้ในการทำงานในคำสั่ง ST A,@3FF แบบไม่รวมเวลาอ้างอิงหน่วยความจำ ของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่ แสดงดังรูปที่ 6.13



(ก) เวลาที่ใช้ในการทำงานในคำสั่งสโตร์ของบัสระบบเข้ารหัสรางคู่



(ข) เวลาที่ใช้ในการทำงานในคำสั่งสโตร์ของบัสระบบเข้ารหัสหนึ่งในสี่

รูปที่ 6.13 ผลการจำลองการทำงานแบบอิงเวลาและเวลาที่ใช้ในการทำงานในคำสั่งสโตร์ของบัสระบบ

เวลาที่ใช้ในการทำงานของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่ของการทำงานทั้ง 8 คำสั่งแบบไม่รวมเวลาอ้างอิงหน่วยความจำ ในหน่วยนาโนวินาที (ns) แสดงดังตารางที่ 6.4 โดยบัสระบบเข้ารหัสรางคู่ทำงานในคำสั่ง ST ใช้เวลาโดยเฉลี่ย 60 นาโนวินาที คำสั่ง LD ใช้เวลาโดยเฉลี่ย 52 นาโนวินาที คำสั่ง OUT ใช้เวลาโดยเฉลี่ย 60 นาโนวินาที คำสั่ง IN ใช้เวลาโดยเฉลี่ย 50 นาโนวินาที ในขณะที่บัสระบบเข้ารหัสหนึ่งในสี่ทำงานในคำสั่ง ST ใช้เวลาโดยเฉลี่ย 53 นาโนวินาที คำสั่ง LD ใช้เวลาโดยเฉลี่ย 50 นาโนวินาที คำสั่ง OUT ใช้เวลาโดยเฉลี่ย 53 นาโนวินาที คำสั่ง IN ใช้เวลาโดยเฉลี่ย 48 นาโนวินาที โดยเมื่อพิจารณาจะพบว่า บัสระบบเข้ารหัสหนึ่งในสี่ลดเวลาในการทำงานลงได้ 7.42% โดยประมาณ เมื่อเปรียบเทียบกับบัสระบบเข้ารหัสรางคู่

ตารางที่ 6.4 เวลาที่ใช้ในการทำงานของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่

Instructions	Estimated Read/Write Cycle Time (ns)	
	Dual-rail Bus	1-of-4 Bus
ST A,@3FF	59.76	54.17
ST A,@1FF	59.20	52.71
LD A,@3FF	52.14	50.53
LD A,@1FF	50.96	49.08
OUT 4,@3F	59.18	53.10
OUT 3,@1F	61.00	53.30
IN 1,@3F	49.34	48.50
IN 2,@1F	51.17	48.50

- **สรุปผลการทดลองบัสระบบเข้ารหัสหนึ่งในสี่**

บัสระบบเข้ารหัสหนึ่งในสี่มีประสิทธิภาพดีกว่าบัสระบบเข้ารหัสรางคู่ กล่าวคือ บัสระบบเข้ารหัสหนึ่งในสี่มีจำนวนการเปลี่ยนสถานะของสัญญาณต่ำกว่า จึงใช้พลังงานในการเปลี่ยนสถานะของสัญญาณต่ำกว่า นอกจากนี้ยังใช้พลังงานรวมในการทำงานต่ำกว่า มีขนาดวงจรเล็กกว่า และใช้เวลาในการทำงานน้อยกว่าบัสระบบเข้ารหัสรางคู่ บนโครงสร้าง

เดียวกัน สรุปประสิทธิภาพได้ดังตารางที่ 6.5 จำนวนสัญลักษณ์ ✓ แสดงถึงประสิทธิภาพ โดยสัญลักษณ์ ✓✓ หมายถึงประสิทธิภาพดีที่สุด

ตารางที่ 6.5 ประสิทธิภาพของบัสเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่

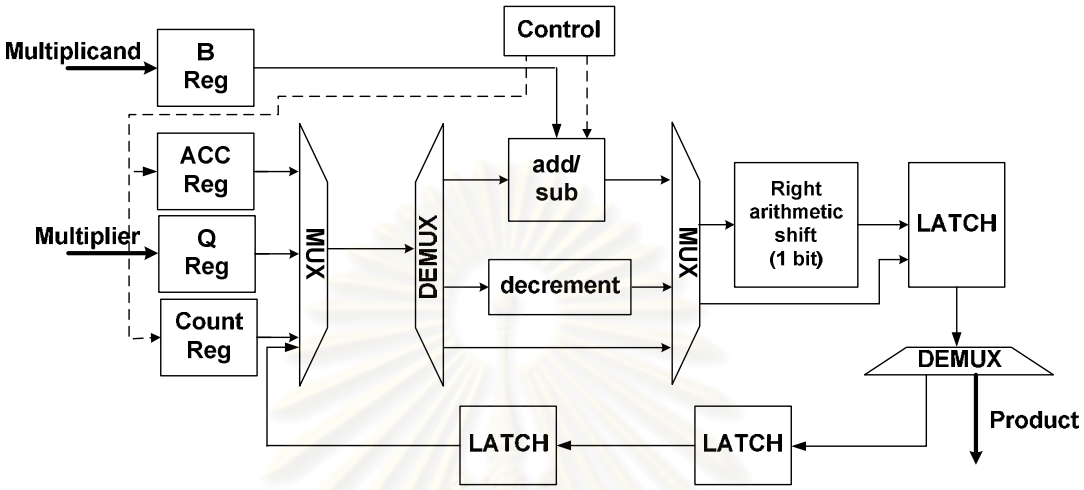
Efficiency	Bus	
	Dual-rail Bus	1-of-4 Bus
Transition Count	✓	✓✓
Power Consumption	✓	✓✓
Circuit Size	✓	✓✓
Read/Write Cycle Time	✓	✓✓

6.3.2 การทดลองวัดประสิทธิภาพของหน่วยคำนวณทางคณิตศาสตร์และตรรกะเข้ารหัสหนึ่งในสี่

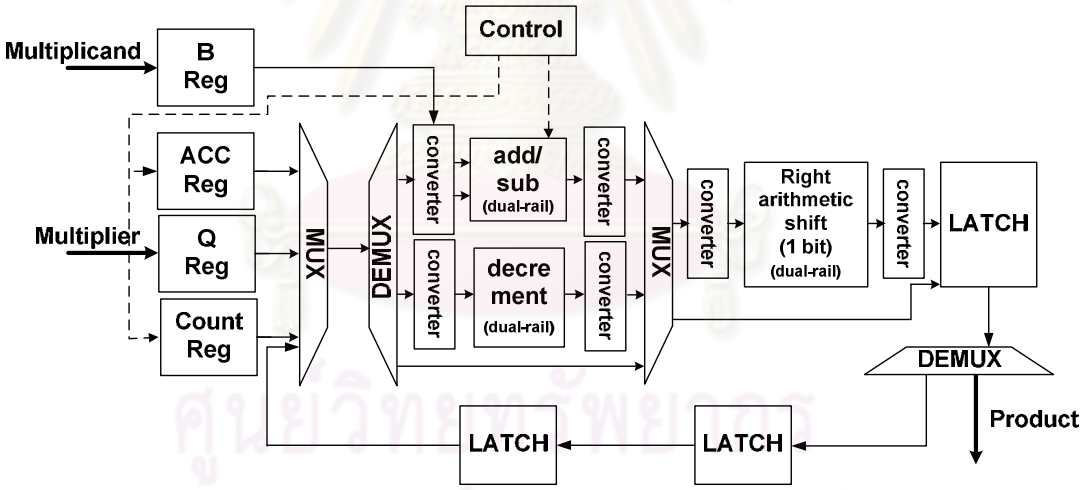
วัดประสิทธิภาพของหน่วยคำนวณทางคณิตศาสตร์และตรรกะผ่านการทดลองการทำงานของวงจรคูณบูทอัลกอลิทึมเข้ารหัสหนึ่งในสี่ ซึ่งประกอบด้วยส่วนคำนวณคือ วงจรบวก วงจรลบ วงจรลดค่า และส่วนตรรกะคือ วงจรเลื่อนบิต โดยวัดประสิทธิภาพการทำงานในด้านของการใช้พลังงาน ขนาดวงจร และความเร็ว แล้วจึงสรุปผลการทดลอง คำสั่งที่ใช้ในการทดลองมีทั้งหมด 4 คำสั่งคือ คูณค่า -34 ด้วยค่า -34 คูณค่า 125 ด้วยค่า 0 คูณค่า 15 ด้วยค่า -1 และคูณค่า 19 ด้วยค่า 30

จากการออกแบบส่วนฟังก์ชัน (Function Unit) หรือส่วนหน่วยคำนวณทางคณิตศาสตร์และตรรกะในบทที่ 5 จะพบว่าวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ที่สร้างจากแผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ มีขนาดใหญ่กว่าวงจรฟังก์ชันเข้ารหัสรางคู่ ในขณะที่ประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่สูงกว่าบัสระบบเข้ารหัสรางคู่ในทุกด้านดังที่อธิบายในหัวข้อที่ 6.3.4 การออกแบบวงจรแบบผสมจึงมีความน่าสนใจเนื่องจากใช้ข้อได้เปรียบของรหัสหนึ่งในสี่และรหัสรางคู่มาออกแบบ กล่าวคือ ออกแบบส่วนรับส่งข้อมูล (Data Path) ให้เข้ารหัสหนึ่งในสี่ และออกแบบส่วนฟังก์ชันให้เข้ารหัสรางคู่ การทดลองในหัวข้อนี้จึงนำวงจรแบบผสมมาร่วมทดลองด้วย โดยปรับปรุงวงจรคูณบูทอัลกอลิทึมเข้ารหัสหนึ่งในสี่ ซึ่งเดิมใช้

ฟังก์ชันเข้ารหัสหนึ่งในสี่ ให้ใช้ฟังก์ชันเข้ารหัสรางคู่แทน วงจรคูณบูทอัลกอลิทึมที่นำมาทดลองประสิทธิภาพของหน่วยคำนวณทางคณิตศาสตร์และตรรกะมีดังนี้

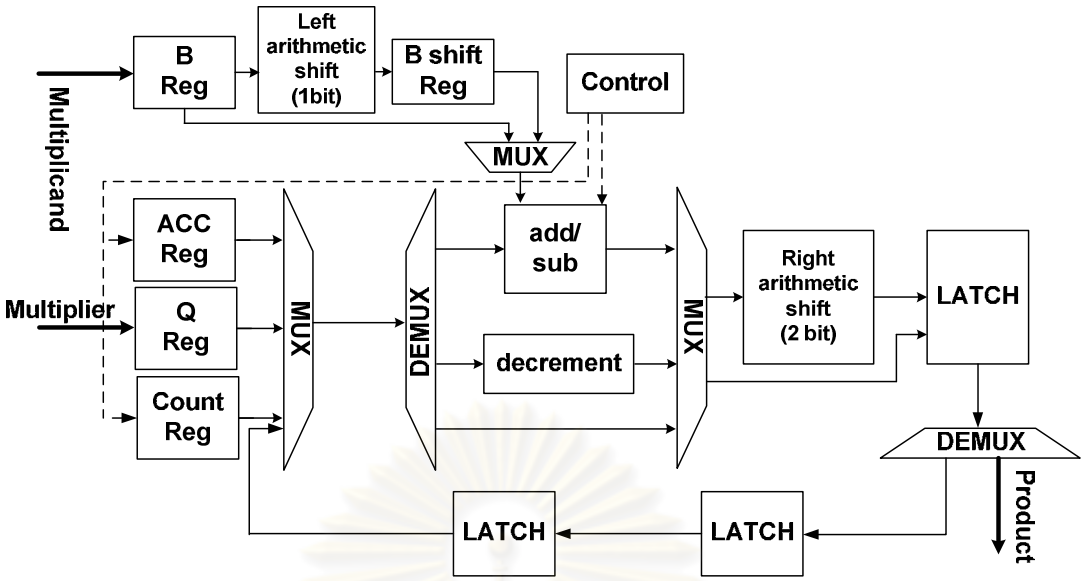


(ก) วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่

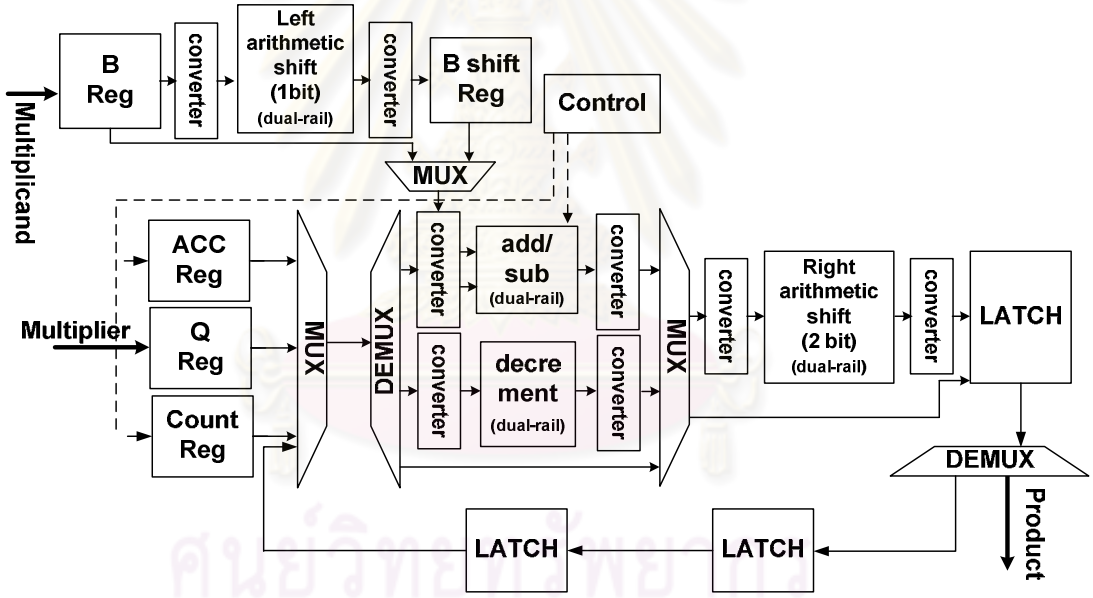


(ข) วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่

รูปที่ 6.14 วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่และใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่



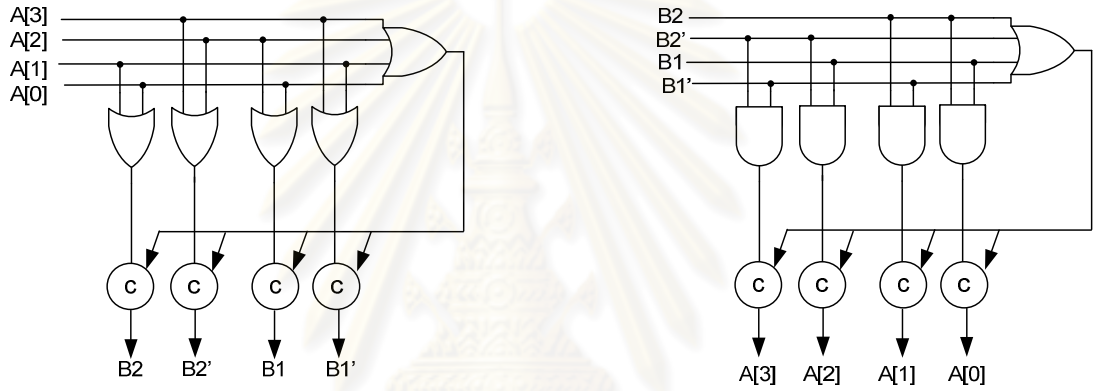
(ก) วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่



(ข) วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่

รูปที่ 6.15 วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่และใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่

- วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ (1-of-4 Radix-2 Booth Multiplier with 1-of-4 Function Unit) ดังรูปที่ 6.14(ก)
- วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ (1-of-4 Radix-2 Booth Multiplier with Dual-rail Function Unit) ดังรูปที่ 6.14(ข)
- วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ (1-of-4 Radix-4 Booth Multiplier with 1-of-4 Function Unit) ดังรูปที่ 6.15(ก)
- วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ (1-of-4 Radix-4 Booth Multiplier with Dual-rail Function Unit) ดังรูปที่ 6.15(ข)



(ก) วงจรแปลงค่ารหัสรางคู่เป็นรหัสหนึ่งในสี่ (ข) วงจรแปลงค่ารหัสหนึ่งในสี่เป็นรหัสรางคู่

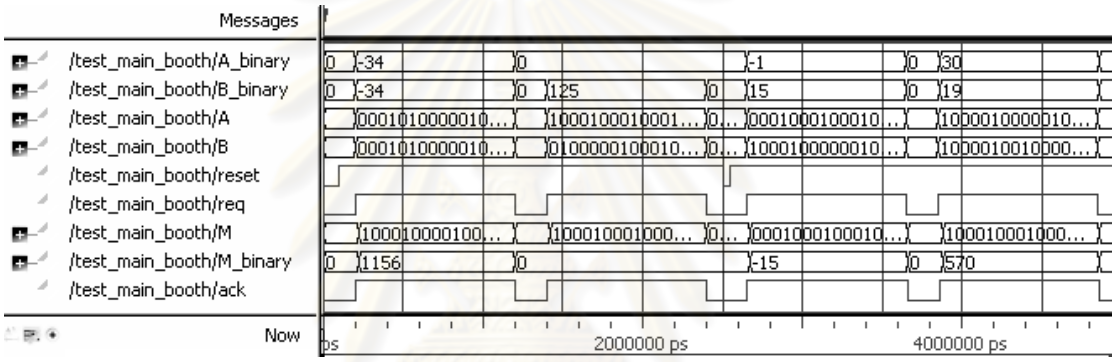
รูปที่ 6.16 วงจรแปลงค่าระหว่างรหัสรางคู่กับรหัสหนึ่งในสี่

ตารางที่ 6.6 รหัสรางคู่และรหัสหนึ่งในสี่

1-of-4 Code				Dual-rail Code			
A[3]	A[2]	A[1]	A[0]	B2	B2'	B1	B1'
1	0	0	0	0	1	0	1
0	1	0	0	0	1	1	0
0	0	1	0	1	0	0	1
0	0	0	1	1	0	1	0

วงจรแบบผสมมีส่วนรับส่งข้อมูลเข้ารหัสหนึ่งในสี่ และส่วนฟังก์ชันเข้ารหัสรางคู่ ดังนั้นจึงต้องใช้วงจรแปลงค่าดังรูปที่ 6.16 แปลงค่าระหว่างรหัสรางคู่กับรหัสหนึ่งในสี่ในตารางที่ 6.6 เพื่อให้ส่วนรับส่งข้อมูลและส่วนฟังก์ชันสามารถทำงานร่วมกันได้

ผลการจำลองการทำงานแบบอิงเวลาของวงจรคูณเข้ารหัสหนึ่งในสี่กับ 4 คำสั่งที่ได้กล่าวไว้ข้างต้น แสดงดังรูปที่ 6.17 ซึ่งผลการทำงานมีความถูกต้อง โดยมีพอร์ท B เป็นตัวตั้ง (Multiplicand) พอร์ท A เป็นตัวคูณ (Multiplier) และพอร์ท M เป็นผลลัพธ์ (Product) ซึ่งเข้ารหัสหนึ่งในสี่ทั้งหมด ส่วนพอร์ท B_binary A_binary และ M_binary จะแสดงค่าเลขฐานสิบของพอร์ท B A และ M ตามลำดับ เพื่อสะดวกต่อการอ่านค่าและตรวจสอบความถูกต้อง



รูปที่ 6.17 ผลการจำลองการทำงานแบบอิงเวลาของวงจรคูณเข้ารหัสหนึ่งในสี่

- **เปรียบเทียบการเปลี่ยนสถานะสัญญาณของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่**

เปรียบเทียบการเปลี่ยนสถานะสัญญาณของวงจรคูณ โดยใช้วงจรนับจำนวนการเปลี่ยนสถานะของสัญญาณในการทดลอง เช่นเดียวกันกับการทดลองบัสระบบ สรุปจำนวนการเปลี่ยนสถานะสัญญาณของวงจรคูณทั้ง 4 แบบกับการทำงานทั้ง 4 คำสั่ง ได้ดังตารางที่ 6.7 โดยวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่มีการเปลี่ยนสถานะสัญญาณรวม 5,444 ครั้ง วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่มีการเปลี่ยนสถานะสัญญาณรวม 8,004 ครั้ง วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่มีการเปลี่ยนสถานะสัญญาณรวม 2,990 ครั้ง วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่มีการเปลี่ยนสถานะสัญญาณรวม 4,534 ครั้ง โดยเมื่อ

พิจารณาจะพบว่า วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ลดการเปลี่ยนสถานะของสัญญาณลงได้ 31.98% โดยประมาณ เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ และวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ลดการเปลี่ยนสถานะของสัญญาณลงได้ 34.05% โดยประมาณ เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่

ตารางที่ 6.7 การเปลี่ยนสถานะสัญญาณของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่

Multiplicand x Multiplier	Transition count			
	1-of-4 Radix-2 Booth Multiplier		1-of-4 Radix-4 Booth Multiplier	
	with dual-rail function unit	with 1-of-4 function unit	with dual-rail function unit	with 1-of-4 function unit
-34 x -34	2,076	1,388	1,194	770
125 x 0	1,926	1,334	1,052	716
15 x -1	1,976	1,352	1,144	752
19 x 30	2,026	1,370	1,144	752
total	8,004	5,444	4,534	2,990

- **เปรียบเทียบขนาดวงจรของของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่**

เปรียบเทียบขนาดวงจรของวงจรคูณ จากผลประมาณค่าการใช้อุปกรณ์เช่นเดียวกันกับการทดลองบัสระบบ ค่าการใช้อุปกรณ์ของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่ แสดงดังตารางที่ 6.8 โดยวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ใช้สไลด์จำนวน 478 ตัว ตารางค้นหาแบบสี่อินพุท จำนวน 791 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุทรวม 107 ตัว วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ใช้สไลด์จำนวน 589 ตัว ตารางค้นหาแบบสี่อินพุท จำนวน 898 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุทรวม 107 ตัว วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัส

หนึ่งในสี่ใช้สไลด์จำนวน 443 ตัว ตารางค้นหาแบบสี่อินพุท จำนวน 709 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุทรวม 75 ตัว วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ใช้สไลด์จำนวน 671 ตัว ตารางค้นหาแบบสี่อินพุท จำนวน 1,007 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุทรวม 75 ตัว โดยเมื่อพิจารณาจะพบว่า วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ลดขนาดวงจรลงได้เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ โดยลดจำนวนการใช้สไลด์ลง 18.85% ลดจำนวนการใช้ตารางค้นหาแบบสี่อินพุท 11.92% และใช้จำนวนพอร์ทอินพุทเอาท์พุทเท่ากัน และวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ลดขนาดวงจรลงได้เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ โดยลดจำนวนการใช้สไลด์ลง 33.98% ลดจำนวนการใช้ตารางค้นหาแบบสี่อินพุท 29.59% และใช้จำนวนพอร์ทอินพุทเอาท์พุทเท่ากัน

ตารางที่ 6.8 การใช้อุปกรณ์ของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่

Multiplier (1-of-4 encoding)	Device Utilization Summary					
	with dual-rail function unit			with 1-of-4 function unit		
	4 input LUTs	Slices	bonded IOBs	4 input LUTs	Slices	bonded IOBs
Radix-2 Booth Multiplier	898	589	107	791	478	107
Radix-4 Booth Multiplier	1,007	671	75	709	443	75

- **เปรียบเทียบความเร็วของวงจรของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่**

เปรียบเทียบความเร็วของวงจรคูณ โดยวัดความเร็วในการทำงานของวงจรคูณทั้งสี่แบบ ซึ่งแสดงบนผลการจำลองการทำงานแบบอิงเวลาเช่นเดียวกันกับการทดลองบัสระบบ เวลาที่ใช้ในการทำงานของวงจรคูณกับการทำงานทั้ง 4 คำสั่งในหน่วยนาโนวินาที (ns) แสดงดังตารางที่ 6.9 โดยวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ใช้เวลารวม

ประมาณ 742 นาโนวินาที วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ใช้เวลารวมประมาณ 668 นาโนวินาที วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ใช้เวลารวมประมาณ 397 นาโนวินาที และวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ ใช้เวลารวมประมาณ 457 นาโนวินาที โดยเมื่อพิจารณาจะพบว่า วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ใช้เวลาในการทำงานเพิ่มขึ้น 10.96% โดยประมาณ เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ และวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ลดเวลาในการทำงานลงได้ 13.24% โดยประมาณ เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่

ตารางที่ 6.9 เวลาที่ใช้ในการทำงานของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่

Multiplicand x Multiplier	Estimated computation time (ns)			
	1-of-4 Radix-2 Booth Multiplier		1-of-4 Radix-4 Booth Multiplier	
	with dual-rail function unit	with 1-of-4 function unit	with dual-rail function unit	with 1-of-4 function unit
-34 x -34	167.69	187.13	127.75	113.28
125 x 0	162.47	181.46	100.63	83.70
15 x -1	169.49	186.00	110.40	94.00
19 x 30	168.78	187.09	118.69	105.91
total	668.43	741.68	457.47	396.89

- **สรุปผลการทดลองของวงจรคูณเข้ารหัสหนึ่งในสี่**

วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่มีประสิทธิภาพดีที่สุดเมื่อเปรียบเทียบกับวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ และวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่ง

ในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ กล่าวคือ ใช้พลังงานในการเปลี่ยนสถานะของสัญญาณต่ำที่สุด มีขนาดวงจรเล็กที่สุด และใช้เวลาในการทำงานน้อยที่สุด บนโครงสร้างเดียวกัน สรุปประสิทธิภาพได้ดังตารางที่ 6.10 จำนวนสัญลักษณ์ ✓ แสดงถึงประสิทธิภาพ โดยสัญลักษณ์ ✓✓✓✓ หมายถึงประสิทธิภาพดีที่สุด

ตารางที่ 6.10 ประสิทธิภาพของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่

Efficiency	Multiplier			
	1-of-4 Radix-2 Booth Multiplier		1-of-4 Radix-4 Booth Multiplier	
	with dual-rail function unit	with 1-of-4 function unit	with dual-rail function unit	with 1-of-4 function unit
Transition Count (Dynamic Power)	✓	✓✓	✓✓✓	✓✓✓✓
Circuit Size (Area)	✓✓	✓✓✓	✓	✓✓✓✓
Computation Time (Time)	✓✓	✓	✓✓✓	✓✓✓✓

ผลการทดลองนี้แสดงให้เห็นว่า วงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ซึ่งเป็นวงจรเข้ารหัสหนึ่งในสี่ล้วน จะมีประสิทธิภาพสูงขึ้นทั้งทางด้านพลังงานที่ใช้ในการเปลี่ยนสถานะของสัญญาณ ขนาดวงจร และความเร็ว หากเป็นการคำนวณครั้งละ 2 หลัก แต่วงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้ฟังก์ชันเข้ารหัสรางคู่ซึ่งเป็นวงจรผสม จะมีประสิทธิภาพทางด้านความเร็วดีกว่าวงจรเข้ารหัสหนึ่งในสี่ล้วน หากเป็นฟังก์ชันคำนวณทีละ 1 หลัก เนื่องจากวงจรฟังก์ชันที่มีการคำนวณทีละหนึ่งหลักของรหัสหนึ่งในสี่ มีความซับซ้อนกว่าของรหัสรางคู่ อย่างไรก็ตาม วงจรผสมเป็นการออกแบบที่ไม่คุ้มค่า เนื่องจากต้องเพิ่มวงจรแปลงข้อมูลระหว่างรหัสหนึ่งในสี่และรหัสรางคู่ในระบบ ทำให้สิ้นเปลืองอุปกรณ์มากขึ้น ส่งผลให้วงจรมีขนาดใหญ่กว่าการออกแบบวงจรด้วยการเข้ารหัสหนึ่งในสี่ล้วน อีกทั้งยังใช้พลังงานในการเปลี่ยนสถานะสัญญาณสูงกว่า

เมื่อพิจารณาประสิทธิภาพของรหัสหนึ่งในสี่เทียบกับการเข้ารหัสแบบอื่น จะสรุปได้ดังตารางที่ 6.11 โดยให้ n เป็นจำนวนบิตของรหัสฐานสอง ซึ่งมีค่าเป็นจำนวนนับ เมื่อพิจารณาจะพบว่าวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ ซึ่งใช้สายสัญญาณ 4 เส้นในการเข้ารหัสฐานสองจำนวน 2 บิต จึงมีประสิทธิภาพดีเมื่อใช้กับงานที่มีการคำนวณเป็นจำนวนคู่ เช่น คำนวณครั้งละ 2 บิต คำนวณครั้งละ 4 บิต คำนวณครั้งละ 8 บิต เป็นต้น โดยจะมีประสิทธิภาพดีที่สุดเมื่อใช้กับงานที่มีการคำนวณครั้งละ 2 บิต

ตารางที่ 6.11 ประสิทธิภาพของรหัสหนึ่งในสี่เทียบกับการเข้ารหัสแบบอื่น

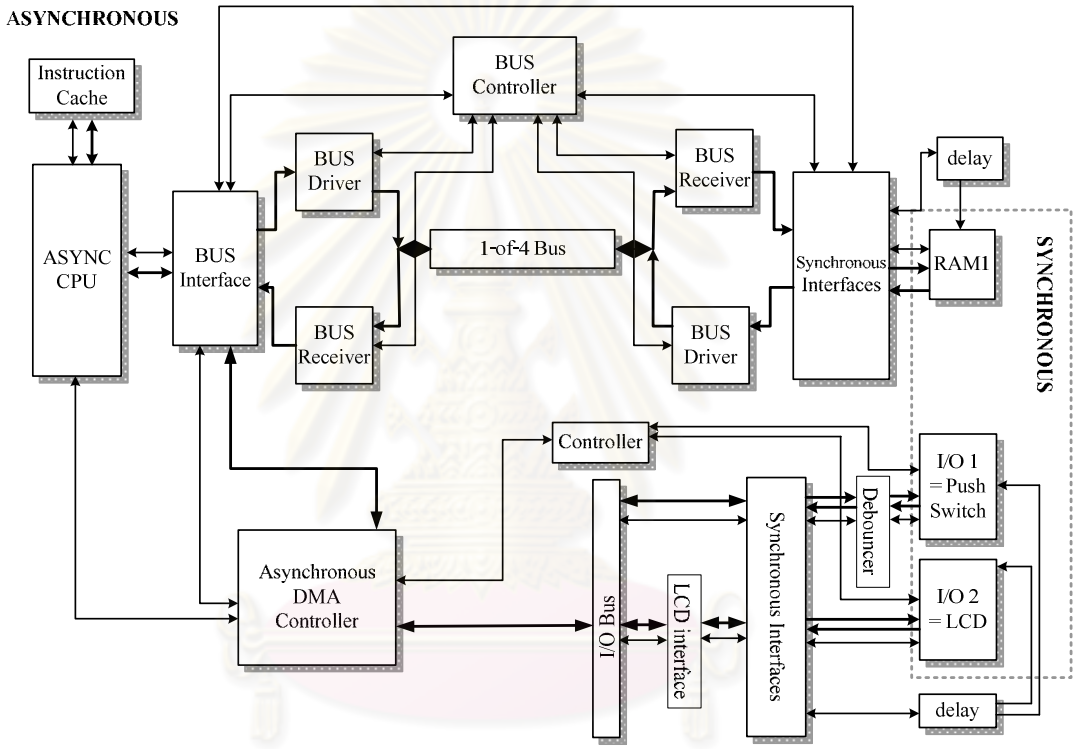
Code (1-of-2^n)	Wires (2^n wires)	Encoding (n bits)	Efficiency (Estimated)			
			Best	Good	Moderate	Poor
1-of-2 (dual-rail)	2	1	1 bit	n bit	-	-
1-of-4	4	2	2 bits	2n bits	2n+1 bits	<2 bits
1-of-8	8	3	3 bits	3n bits	3n+1 bits 3n+2 bits	<3 bits
1-of-16	16	4	4 bits	4n bits	4n+1 bits 4n+2 bits 4n+3 bits	<4 bits
1-of-32	32	5	5 bits	5n bits	5n+1 bits 5n+2 bits 5n+3 bits ⋮ 5n+4 bits	<5 bits
1-of-64	64	6	6 bits	6n bits	6n+1 bits 6n+2 bits 6n+3 bits ⋮ 6n+5 bits	<6 bits

6.3.3 โปรแกรมวงจรบัสระบบเข้ารหัสหนึ่งในสี่ลงเอฟพีจีเอ

ทดลองการทำงานของบัสระบบเข้ารหัสหนึ่งในสี่ร่วมกับองค์ประกอบ คือ ไมโครโพรเซสเซอร์ ดีเอ็มเอ และอุปกรณ์ต่อพ่วงคือ สวิทช์แบบกดติดปล่อยดับ และแอลซีดีบนเอฟพีจีเอ ด้วยการโปรแกรมวงจรลงเอฟพีจีเอ Xlinx SPARTAN-3E เบอร์ XC3S500EFG320 โดยเชื่อมต่อสวิทช์แบบกดติดปล่อยดับไว้กับพอร์ทอุปกรณ์ต่อพ่วงที่ 1 ของดีเอ็มเอ (I/O[1]) เชื่อมต่อแอลซีดีไว้กับพอร์ทอุปกรณ์ต่อพ่วงที่ 2 ของดีเอ็มเอ (I/O[2]) แล้วสร้างส่วนติดต่อกับสวิทช์แบบกดติดปล่อยดับคือ ตัวกันผลการเด้ง (Debouncer) และส่วนติดต่อกับแอลซีดี (LCD Interface) ดังรูปที่ 6.18 จากนั้นเขียนโปรแกรมให้เกิดการอินเตอร์รัพท์เมื่อสวิทช์แบบกดติดปล่อยดับถูกกด โดยจะอินเตอร์รัพท์ไปทำโปรแกรมย่อยคือส่งค่าตัวอักษรคำว่า 1-of-4 Bus! ออกจอแอลซีดี รายละเอียดของคำสั่งที่ใช้ในการทดลองมีดังนี้

```
0       ORG   0000H
// เก็บค่าตัวอักษรไว้ในหน่วยความจำ
1       LD    A,#31    ; Reg A = 31
2       ST    A,@1     ; M[1] = Reg A
3       LD    A,#2D    ; Reg A = 2D
4       ST    A,@2     ; M[2] = Reg A
5       LD    A,#6F    ; Reg A = 6F
6       ST    A,@3     ; M[3] = Reg A
7       LD    A,#66    ; Reg A = 66
8       ST    A,@4     ; M[4] = Reg A
9       LD    A,#2D    ; Reg A = 2D
10      ST    A,@5     ; M[5] = Reg A
11      LD    A,#34    ; Reg A = 34
12      ST    A,@6     ; M[6] = Reg A
13      LD    A,#20    ; Reg A = 20
14      ST    A,@7     ; M[7] = Reg A
15      LD    A,#42    ; Reg A = 42
16      ST    A,@8     ; M[8] = Reg A
17      LD    A,#75    ; Reg A = 75
18      ST    A,@9     ; M[9] = Reg A
19      LD    A,#73    ; Reg A = 73
20      ST    A,@A     ; M[A] = Reg A
21      LD    A,#21    ; Reg A = 21
22      ST    A,@B     ; M[2] = Reg A
// รับข้อมูลจากสวิทช์กดติดปล่อยดับ
23      OUT   1,@C     ; M[C] = I/O[1]
                         (I/O[1] = 1)
// เกิดการอินเตอร์รัพท์ไปทำโปรแกรมย่อย
คือส่งค่าตัวอักษรจากหน่วยความจำออก
จอ LCD
INT:
24      IN    2,@1     ; I/O[2] = M[1]
```

```
25    IN    2,@2    ; I/O[2] = M[2]      31    IN    2,@8    ; I/O[2] = M[8]
26    IN    2,@3    ; I/O[2] = M[3]      32    IN    2,@9    ; I/O[2] = M[9]
27    IN    2,@4    ; I/O[2] = M[4]      33    IN    2,@A    ; I/O[2] = M[A]
28    IN    2,@5    ; I/O[2] = M[5]      34    IN    2,@B    ; I/O[2] = M[B]
29    IN    2,@6    ; I/O[2] = M[6]      35    END
30    IN    2,@7    ; I/O[2] = M[7]
```

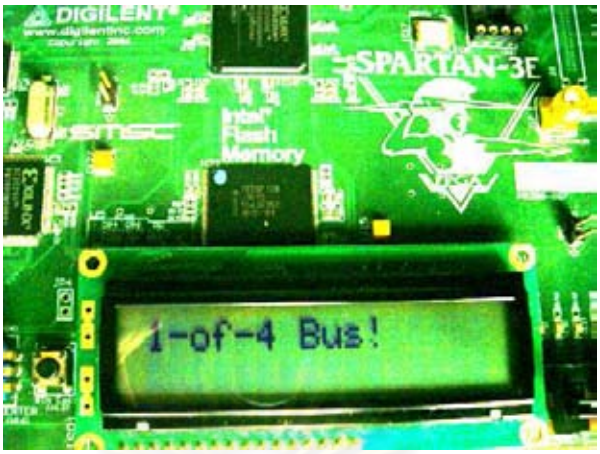


รูปที่ 6.18 บัสระบบเข้ารหัสหนึ่งในสี่ องค์ประกอบ

ส่วนติดต่อกับสวิทซ์แบบกดติดปล่อยดับ และส่วนติดต่อกับแอลซีดี

ผลการทดลองการทำงานของบัสระบบเข้ารหัสหนึ่งในสี่ร่วมกับองค์ประกอบ

พบว่าวงจรทำงานได้ถูกต้องบนเอฟพีจีเอ ดังแสดงในรูปที่ 6.19



รูปที่ 6.19 ผลการทดลองของบัสระบบเข้ารหัสหนึ่งในสี่ร่วมกับองค์ประกอบบนเอฟพีจีเอ

6.3.4 สรุปผลการทดลองทั้งหมด

สรุปผลการทดลองทั้งหมดได้ดังตารางที่ 6.12 จำนวนสัญลักษณ์ ✓ แสดงถึงประสิทธิภาพ โดยสัญลักษณ์ ✓✓ หมายถึงประสิทธิภาพดีที่สุด

ตารางที่ 6.12 สรุปผลการทดลองทั้งหมด

Circuits	Efficiency					
	Transition Count		Circuit Size		Computation Time	
	Dual-rail	1-of-4	Dual-rail	1-of-4	Dual-rail	1-of-4
Bus	✓	✓✓	✓	✓✓	✓	✓✓
Function Unit(compute 2-bits per time)	✓	✓✓	✓	✓✓	✓	✓✓
Function Unit(compute 1-bits per time)	✓	✓✓	✓✓	✓	✓✓	✓

จากตารางที่ 6.12 บัสระบบเข้ารหัสหนึ่งในสี่มีประสิทธิภาพดีกว่าบัสระบบเข้ารหัสรางคู่ บนโครงสร้างเดียวกัน สำหรับวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ จะมีประสิทธิภาพสูงกว่าวงจรเข้ารหัสรางคู่ทั้งทางด้านพลังงาน ขนาดวงจร และความเร็ว หากเป็นการคำนวณครั้งละ 2 หลัก แต่หากเป็นฟังก์ชันคำนวณทีละ 1 หลัก วงจรฟังก์ชันเข้ารหัสรางคู่จะมีประสิทธิภาพทางด้านความเร็วดีกว่าวงจรเข้ารหัสหนึ่งในสี่ เนื่องจากวงจรฟังก์ชันที่มีการคำนวณทีละหนึ่งหลักของรหัส

หนึ่งในสี่ มีความซับซ้อนกว่าของรหัสรางคู่ ดังนั้นวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่จึงมีประสิทธิภาพดีเมื่อใช้กับงานที่มีการคำนวณเป็นจำนวนคู่ เช่น คำนวณครั้งละ 2 บิต คำนวณครั้งละ 4 บิต คำนวณครั้งละ 8 บิต เป็นต้น โดยจะมีประสิทธิภาพดีที่สุดเมื่อใช้กับงานที่มีการคำนวณครั้งละ 2บิต

บทที่ 7

สรุปผลการวิจัยและข้อเสนอแนะ

ในบทนี้ประกอบด้วย การสรุปผลวิจัยและข้อเสนอแนะ โดยสรุปรายละเอียดของการออกแบบ การทดลอง และผลการทดลองของบัสระบบแบบอสมวารเข้ารหัสหนึ่งในสี่กับองค์ประกอบ จากนั้นเสนอข้อเสนอแนะที่ได้จากการทำวิจัยนี้ในตอนท้าย รายละเอียดมีดังต่อไปนี้

7.1 สรุปผลการวิจัย

การพัฒนาวงจรอสมวารมีความน่าสนใจเนื่องจาก วงจรอสมวารไม่ใช้สัญญาณนาฬิกาควบคุมในการรับส่งข้อมูล แต่ใช้วิธีการเข้ารหัสข้อมูลเพื่อตรวจสอบการมาถึงของข้อมูลและสร้างจังหวะในการทำงานที่ถูกต้องให้กับวงจรอสมวารแทน การทำงานที่เป็นอิสระจากสัญญาณนาฬิกานี้ส่งผลให้วงจรอสมวารมีประสิทธิภาพดีกว่าวงจรสมวารทางด้านความเร็ว และการใช้พลังงานที่ต่ำกว่า โดยการเข้ารหัสข้อมูลโดยใช้รหัสหนึ่งในสี่ เป็นรูปแบบการเข้ารหัสแบบ 1-of-N Code ที่ใช้พลังงานในการเปลี่ยนสถานะของสัญญาณน้อยกว่าการเข้ารหัสข้อมูลแบบอื่น สำหรับการออกแบบบัสที่ใช้เป็นเส้นทางรับส่งข้อมูลระหว่างวงจรให้มีประสิทธิภาพทำได้ยาก เนื่องจากการรับส่งข้อมูลระหว่างวงจรมีความแตกต่างกันเพื่อรองรับการทำงานที่หลากหลาย หากในระบบมีทั้งวงจรสมวารและอสมวารรวมกัน จะทำให้การออกแบบยุ่งยากมากยิ่งขึ้นไปอีก เกิดปัญหาการใช้สายสัญญาณจำนวนมาก การใช้พลังงานอย่างสิ้นเปลือง และเกิดความยุ่งยากในการปรับปรุงเพื่อพัฒนาต่อไป

งานวิจัยนี้จึงมีวัตถุประสงค์เพื่อพัฒนาขอบเขตการวิจัยเกี่ยวกับวงจรอสมวารทางด้านการใช้พลังงานให้มีประสิทธิภาพในการรับส่งข้อมูล และประมวลผลวงจรแบบอสมวาร โดยศึกษาการออกแบบบัสระบบแบบอสมวาร โดยใช้รหัสหนึ่งในสี่เข้ารหัสในการรับส่งข้อมูลเพื่อเพิ่มประสิทธิภาพทางด้านการใช้พลังงานให้กับบัสระบบที่เชื่อมต่อกับไมโครโพรเซสเซอร์แบบอสมวาร หน่วยความจำ และอุปกรณ์อินพุท/เอาท์พุท ซึ่งรองรับความสามารถในการเพิ่มความเร็วของบัสด้วยเทคนิคอินเตอร์รัพท์และดีเอ็มเอได้ โดยคุณสมบัติของบัสระบบ ดีเอ็มเอ และหน่วยประมวลผล เปรียบเทียบกับงานวิจัยเวอร์ชันเดิม [4] เป็นดังตารางที่ 7.1

การออกแบบบัสแบบอสมวารเข้ารหัสหนึ่งในสี่ในงานวิจัยนี้ ได้ปรับปรุงโครงสร้างจากบัสระบบรางคู่จากงานวิจัยออกแบบบัสสำหรับวงจรอสมวาร [4] เดิม โดยเปลี่ยนการเข้ารหัสบัสระบบ จากรหัสรางคู่เป็นรหัสหนึ่งในสี่ และตัดโครงสร้างส่วนที่มีความซ้ำซ้อนจากบัสระบบเดิมออก รวมถึงเปลี่ยนอุปกรณ์ Weak Inverter ที่ใช้กำจัดสัญญาณไฮ-อิมพีแดนซ์จากลอจิกสามสถานะบนสายสัญญาณบัส เป็นอุปกรณ์แอนเกตแทน เพื่อให้สอดคล้องกับข้อจำกัดซึ่งเป็นการออกแบบระดับเกต เข้ารหัสหนึ่งในสี่กับสัญญาณอาณัติแบบ 4 ขั้น นอกจากนี้ หากไม่ได้ออกแบบวงจรบัสบนเอฟพีจีเอที่ประกอบด้วยตัวต้านทานพูลอัพ และพูลดาวน์เป็นโครงสร้างหลักภายในวงจร I/O Block ของเอฟพีจีเอ จะสามารถต่อตัวต้านทานพูลดาวน์ แทนการต่อแอนด์เกตบนสายสัญญาณบัส เพื่อลดขนาดวงจรของสายสัญญาณบัสได้ บัสระบบแบบอสมวารที่ออกแบบมีคุณสมบัติโดยสรุปคือ เข้ารหัสหนึ่งในสี่กับสัญญาณอาณัติแบบ 4 ขั้น รับส่งข้อมูลได้ทั้งสองทิศทาง ทำงานแบบมัลติเพล็กซ์ มีความกว้าง 20’1of4 บิต รองรับการใช้งานรับส่งเลขที่อยู่ขนาด 20’1of4 บิต และรับส่งข้อมูลขนาด 16’1of4 บิต เชื่อมต่อกับไมโครโพรเซสเซอร์ ดีเอ็มเอ และหน่วยความจำจำนวน 2 ตัว โครงสร้างของบัสระบบประกอบด้วย บัส ส่วนติดต่อกับไมโครโพรเซสเซอร์และดีเอ็มเอ ส่วนติดต่อกับหน่วยความจำแบบสมวาร และตัวควบคุมบัส

ตารางที่ 7.1 คุณสมบัติโดยสรุปของบัสระบบ ดีเอ็มเอ และหน่วยประมวลผล

คุณสมบัติ	งานวิจัยเวอร์ชันเดิม [4]	งานวิจัยนี้
รูปแบบการเข้ารหัสข้อมูล	เข้ารหัสรางคู่	เข้ารหัสหนึ่งในสี่
ขนาดของบัสระบบ	8 บิต	10 บิต
ขนาดของดีเอ็มเอ	8 บิต	24 บิต
ขนาดของหน่วยประมวลผล	8 บิต	8 บิต
จำนวนชุดคำสั่ง	16	27
จำนวนหน่วยความจำที่รองรับ	1	2
จำนวนอุปกรณ์ต่อพ่วงที่รองรับ	2	4
รูปแบบการติดต่อกับอุปกรณ์ต่อพ่วง	Memory-mapped I/O	I/O-mapped I/O
รูปแบบการออกแบบ ALU	Dual-rail BDD	1-of-4 BDD

การออกแบบตัวควบคุมสัญญาณร้องขอและสัญญาณตอบรับ บนบัสระบบและองค์ประกอบคือ ดีเอ็มเอ และไมโครโพรเซสเซอร์ จะใช้อุปกรณ์ไปป์ไลน์ซึ่งมีขนาดเล็กกว่าแทนอุปกรณ์ออโต้สวีปที่มีขนาดใหญ่กว่าในบัสระบบเวอร์ชันเดิม สำหรับวงจรควบคุมระบบและ

องค์ประกอบ สร้างจากกราฟบรรยายการเปลี่ยนสัญญาณ โดยใช้แบบจำลองความหน่วงที่ไม่ขึ้นต่ออัตราเร็ว และใช้โปรแกรม Petrify Tool ช่วยสร้างวงจรให้สมบูรณ์ สำหรับวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ในองค์ประกอบของบัสระบบ สร้างจากแผนภาพการตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ โดยแทนค่าตัวแปรในพีชคณิตบูลีนและเขียนแจกแจงเป็นโครงสร้างที่มีอันดับชั้นเพื่อช่วยในการออกแบบและสังเคราะห์วงจรที่มีขนาดใหญ่

การออกแบบองค์ประกอบซึ่งเป็นส่วนที่ใช้งานบัส แบ่งออกเป็น การออกแบบดีเอ็มเอ โดยออกแบบดีเอ็มเอเป็นแบบอสมวารเข้ารหัสหนึ่งในสี่ เชื่อมต่อระหว่างบัสระบบกับบัสอุปกรณ์ต่อพ่วง เพื่อทำหน้าที่เป็นตัวกลางรับส่งข้อมูลระหว่างอุปกรณ์ถึงอุปกรณ์โดยตรง การออกแบบองค์ประกอบอีกส่วนหนึ่งคือ การออกแบบไมโครโพรเซสเซอร์ โดยออกแบบไมโครโพรเซสเซอร์ขนาด 8 บิต เข้ารหัสหนึ่งในสี่กับสัญญาณอาณัติแบบ 4 ขั้น ซึ่งสามารถรองรับการทำงานร่วมกับบัสระบบ ดีเอ็มเอ และการบริการอินเตอร์รัพท์ได้

การอิมพลิเมนท์วงจรบัสระบบแบบอสมวารเข้ารหัสหนึ่งในสี่ ด้วยการโปรแกรมลงเอฟพีจีเอเพื่อตรวจสอบความถูกต้องในการทำงานนั้นมีความยุ่งยาก เนื่องจากการสร้างวงจรบนวงจรรวมของเอฟพีจีเอจะสุ่มพื้นที่ในการสร้างวงจรขึ้นเอง จึงไม่สามารถกำหนดตำแหน่งของเกตภายในวงจรให้เป็นไปตามต้องการได้ ระยะทางของเกตภายในวงจรที่ถูกปรับเปลี่ยนจากการสุ่มนั้น จะทำให้ค่าเวลาล่าช้า หรือ Delay Time ในวงจรแตกต่างไปจากที่ออกแบบไว้ อาจส่งผลให้วงจรอสมวารทำงานผิดพลาดได้ กล่าวคือ ผลจำลองการทำงานของวงจรย่อยอาจมีความถูกต้อง แต่เมื่อนำวงจรย่อยมาเชื่อมต่อกันเป็นวงจรขนาดใหญ่ขึ้น ผลจำลองการทำงานอาจไม่ถูกต้องตามพฤติกรรมวงจรที่ได้ออกแบบไว้ได้ เนื่องจากวงจรขนาดใหญ่จะถูกทำการสังเคราะห์และอิมพลิเมนท์วงจรใหม่ทั้งหมด รวมถึงสังเคราะห์และอิมพลิเมนท์วงจรย่อยใหม่ทั้งหมดด้วย ซึ่งอาจทำให้วงจรย่อยมีตำแหน่งการจัดวางเกตและเชื่อมสายบนอุปกรณ์เอฟพีจีเอเปลี่ยนไปจากเดิม ส่งผลกระทบถึงค่าเวลาล่าช้าในเกตและสาย ทำให้วงจรย่อยทำงานผิดพลาดได้ ดังนั้นจึงต้องแยกโครงสร้างที่ได้จากการสังเคราะห์และอิมพลิเมนท์วงจรย่อยที่มีความถูกต้องไว้ไม่ให้ถูกแก้ไข และนำโครงสร้างวงจรย่อยเหล่านั้นมาเชื่อมต่อกันเป็นวงจรขนาดใหญ่ภายหลัง

การทดลองการทำงานของบัสระบบเข้ารหัสหนึ่งในสี่ จะวัดประสิทธิภาพของบัสระบบโดย วัดการใช้พลังงาน วัดขนาดวงจร วัดความเร็วในการทำงาน และตรวจสอบความถูกต้องในการทำงาน โดยพบว่า บัสระบบสามารถทำงานร่วมกับองค์ประกอบได้อย่างถูกต้อง โดยบัสระบบเข้ารหัสหนึ่งในสี่มีประสิทธิภาพดีกว่าบัสระบบเข้ารหัสรางคู่ บนโครงสร้างเดียวกัน สำหรับวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ จะมีประสิทธิภาพสูงกว่าวงจรเข้ารหัสรางคู่ ทั้งทางด้านพลังงานที่ใช้เปลี่ยนสถานะของสัญญาณ ขนาดวงจร และความเร็วในการทำงาน หากเป็นการคำนวณครั้งละ 2 หลัก แต่หากเป็นฟังก์ชันคำนวณทีละ 1 หลัก วงจรฟังก์ชันเข้ารหัสรางคู่จะมีประสิทธิภาพทางด้านความเร็วดีกว่าวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ ดังนั้นวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่จึงมีประสิทธิภาพดีเมื่อใช้กับงานที่มีการคำนวณเป็นจำนวนคู่ โดยจะมีประสิทธิภาพดีที่สุดเมื่อใช้กับงานที่มีการคำนวณครั้งละ 2 บิต

ผลการทดลองการทำงานของบัสระบบเข้ารหัสหนึ่งในสี่ตามที่ได้กล่าวไปนั้น เป็นผลการทดลองที่อ้างอิงจากการออกแบบวงจรบนเอฟพีจีเอ Xlinx SPARTAN-3E เบอร์ XC3S-500EFG320 โดยเอฟพีจีเอดังกล่าวประกอบด้วย [24] CLB (Configuration Logic Block) ซึ่งเป็นวงจรรวมที่สามารถปรับเปลี่ยนโครงแบบให้มีการทำงานสอดคล้องกับพฤติกรรมของวงจรตามที่ออกแบบไว้ได้ เมื่อเอฟพีจีเอทำงานจะสร้างสัญญาณนาฬิกาไปกระตุ้นให้ฟลิฟฟลอปภายใน CLB ทำงาน การทำงานของเอฟพีจีเอที่จำเป็นต้องอาศัยสัญญาณนาฬิกานี้ ส่งผลให้วงจรอสมวารที่ออกแบบและทดลองบนเอฟพีจีเอไม่สามารถทำงานแบบอสมวารอย่างแท้จริงได้เนื่องจากยังมีสัญญาณนาฬิกาเกี่ยวข้อง การวัดประสิทธิภาพของวงจรอสมวารโดยมีเอฟพีจีเอซึ่งทำงานแบบสมวาร (Synchronous FPGA) มาเกี่ยวข้อง จะทำให้ประสิทธิภาพที่แท้จริงของวงจรอสมวารถูกลดทอนลงไป

7.2 ข้อเสนอแนะ

การทดลองวงจรอสมวารบนเอฟพีจีเอแบบสมวารนั้น ทำให้ประสิทธิภาพที่แท้จริงของวงจรอสมวารถูกลดทอนลงไป ในอนาคตหากมีเอฟพีจีเอแบบอสมวารแพร่หลาย การทดลองวงจรอสมวารบนเอฟพีจีเอแบบอสมวาร (Asynchronous FPGA) จึงมีความน่าสนใจ เนื่องจากสามารถวัดประสิทธิภาพที่แท้จริงของวงจรอสมวารได้มากขึ้น เพราะเป็นอิสระต่อสัญญาณนาฬิกาบนเอฟพีจีเอแบบสมวาร

รายการอ้างอิง

[1] Davis, A. ,and Nowick, S.M. An Introduction to Asynchronous Circuit Design. Department of Computer Science University of Utah Technical report, September 1997.

[2] Sparsø, J. ,and Furber, S. Principles of asynchronous circuit design – A systems perspective. pp.9-14,87. Kluwer Academic Publishers, 2001.

[3] Verhoeff, T. Delay-insensitive Codes-an Overview. Department of Mathematics and Computing Science Eindhoven University of Technology, Springer Berlin / Heidelberg, 1987.

[4] Sufian, S. A Design of System Bus for Asynchronous Circuits. Master's thesis, Department of Computer Engineering Faculty of Engineering Chulalongkorn University, 2006.

[5] Hauck, S. Asynchronous Design Methodologies: An Overview. Proceedings of the IEEE, pp. 69-93 Vol. 83 No. 1. January 1995.

[6] Ginosar, R. Asynchronous Design and Synchronization. [Online]. 2009. Available from: http://webee.technion.ac.il/courses/048878.htm [2011,March 8]

[7] Bainbridge, W.J. ,and Furber, S.B. Delay Insensitive System-on-Chip Interconnect using 1-of-4 Data Encoding. Proceedings Async 2001 IEEE Computer Society Press , March 2001.

[8] Takamura, A., Kuwako, M., Ueno, Y., Kagotani, H. ,and Nanya, T. TITAC : Design of a Quasi-Delay-Insensitive Microprocessor. IEEE Design & Test of Computers, pp. 58-59,61. Summer 1994.

[9] Miller, R.E. Combinational Circuit. IEEE Transactions on Computers, Volume 1 of Switching Theory. 1965.

[10] Godse, A. P. ,and Godse, D.A. Microprocessor – I. First Edition chapter 8 pp.2,10. Technical Publications, 2008.

[11] Stallings, W. Computer organization and architecture: designing for performance. Seventh edition pp.223. Prentice Hall, 2006.

[12] Lewis, M.G. Lower Power Asynchronous Digital Signal Processing. Doctoral dissertation, Department of Computer Science Faculty of Engineering and Physical Sciences University of Manchester, 2000.

[13] Xilinx, Inc. Application Note: Spartan-3 FPGA Family. Using Block RAM in Spartan-3 Generation FPGAs. [Online]. 2005. Available from: http:// www.xilinx.com [2011, March 8]

[14] Bainbridge, W.J. ,and Furber, S.B. Asynchronous Macrocell Interconnect using MARBLE. Proc. Async'98 San Diego, pp. 122-132. April 1998.

[15] Plana, L.A., Riocreux, P.A., Bainbridge, W.J., Bardsley, A., Garside, J.D. ,and Temple, S. SPA - A Synthesisable Amulet Core for Smartcard Applications. Proceedings of Async'2002 Manchester, pp. 201-210. April 2002.

[16] Takamura, A. and others. TITAC–2 : An asynchronous 32-bit microprocessor based on Scalable-Delay-Insensitive model. In Proc. International Conf. Computer Design (ICCD'97), pp. 288–294. MIT Press, October 1997.

[17] Ruangsinsup, P. Design of 8-bit scalable-delay-Insensitive microprocessor using FPGA. Master's thesis, Department of Computer Engineering Faculty of Engineering Chulalongkorn University, 2001.

[18] Cortadella, J. The Department of Computer Architecture (DAC) Universitat Politècnica de Catalunya Spain. Petrify : a tutorial for the designer of asynchronous circuits. [Online]. (no year given). Available from: http://www.cs.unc.edu/~montek/teaching/spring-04.htm [2011, March 8]

[19] Mangino, J. Texas Instruments Incorporated. Using DMA with High Performance Peripherals to Maximize System Performance. [Online]. 2007. Available from: http://focus.ti.com/lit/wp/spna105/spna105.pdf [2011, March 8]

[20] Saxena, N.R. ,and Robinson, J.P. Syndrome and Transition Count are Uncorrelated. IEEE Transactions on Information Theory archive, IEEE Press Piscataway NJ USA, pp. 64. 1988.

[21] Xilinx, Inc. ISE Web pack 11.1. Xilinx Inc. : Xilinx Inc. 2009.

[22] Xilinx, Inc. ModelSim XE 6.4b. Xilinx Inc. : Xilinx Inc. 2009.

[23] Xilinx, Inc. Spartan-3 Generation FPGA User Guide. Package Marking. [Online]. 2009. Available from: http:// www.xilinx.com [2011, March 8]

[24] Xilinx, Inc. XA Spartan-3E Automotive FPGA Family Data Sheet. Architectural Overview. [Online]. 2009. Available from: http:// www.xilinx.com [2011, March 8]

[25] Petrov, P. Project II: Implementation of a Booth Multiplier. ENEE 359V: Advanced Digital Design with HDLs, Department of Computer Science University of Marryland, pp.1-3, Fall 2009.

[26] Hennessy, J.L. ,and Patterson, D.A. Computer Architecture : A Quantitative approach, pp.700-701. Morgan Kaifmann Publishers is an Imprint of Elsevier, 2007.

ภาคผนวก

ภาคผนวก ก

ชุดคำสั่งและรหัสดำเนินการ

Description	Mnemonic code

DATA TRANSFER

1 **LD** = Load

A, #K
(A <- K)

9	8	7...	1	0	9	8	7	6	5	4	3	2	1	0
0	0	K			0	0	0	0	0	0	0	0	0	1

A, Reg
(A <- Reg)

9...	3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 … 0			reg addr		0	0	0	1	0	0	0	0	0	1

A, @K
(A <- Mem[K])

9...	3	2	1	0	9	8	7	6	5	4	3	2	1	0
mem addr					0	0	1	0	0	0	0	0	0	1

A, @Reg
(A <- Mem[Reg])

9...	3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 … 0			reg addr		0	0	1	1	0	0	0	0	0	1

2 **ST** = Store

A, Reg
(Reg <- A)

9...	3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 … 0			reg addr		0	1	0	0	0	0	0	0	1	0

A, @K
(Mem[K] <- A)

9...	3	2	1	0	9	8	7	6	5	4	3	2	1	0
mem addr					1	0	0	0	0	0	0	0	1	0

A, @Reg
(Mem[Reg] <- A)

9...	3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 … 0			reg addr		1	1	0	0	0	0	0	0	1	0

3 **IN** = Input from

@K2, @K
(Mem2[K2] <- Mem[K])

9...	3	2	1	0	9...	3	2	1	0	9	8	7	6	5	4	3	2	1	0
mem addr2					mem addr					0	0	1	0	0	0	0	0	1	1

@K2, @K
(Mem2[K2] <- Mem[K];
Mem2[K2+1] <- Mem[K+1])

9...	3	2	1	0	9...	3	2	1	0	9	8	7	6	5	4	3	2	1	0
mem addr2					mem addr					0	1	0	0	0	0	0	0	1	1

@K2, @K
(Mem2[K2] <- Mem[K];
Mem2[K2+1] <- Mem[K+1];
Mem2[K2+2] <- Mem[K+2])

9...	3	2	1	0	9...	3	2	1	0	9	8	7	6	5	4	3	2	1	0
mem addr2					mem addr					1	0	0	0	0	0	0	0	1	1

I/O,@K
(I/O <- Mem[K])

9...	3	2	1	0	9...	3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 … 0		I/O no.			mem addr					0	0	1	0	1	0	0	0	1	1

I/O,I/O
(I/O <- I/O)

9...	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 … 0		I/O no.		I/O no.				0	0	0	0	1	0	0	1	0	0
		destination		source													

Description	Mnemonic code

4 **OUT** = Output from

@K, @K2
(Mem[K] <- Mem2[K2])

9... 3 2 1 0	9... 3 2 1 0	9 8 7 6 5 4 3 2 1 0
mem addr	mem addr2	0 0 1 0 0 0 0 1 0 1

@K, @K2
(Mem[K] <- Mem2[K2];
Mem[K+1] <- Mem2[K2+1])

9... 3 2 1 0	9... 3 2 1 0	9 8 7 6 5 4 3 2 1 0
mem addr	mem addr2	0 1 0 0 0 0 0 1 0 1

@K, @K2
(Mem[K] <- Mem2[K2];
Mem[K+1] <- Mem2[K2+1];
Mem[K+2] <- Mem2[K2+2])

9... 3 2 1 0	9... 3 2 1 0	9 8 7 6 5 4 3 2 1 0
mem addr	mem addr2	1 0 0 0 0 0 0 1 0 1

I/O,@K
(Mem[K] <- I/O)

9... 3 2 1 0		9... 3 2 1 0	9 8 7 6 5 4 3 2 1 0
0 ... 0	I/O no.	mem addr	1 0 0 0 1 0 0 1 0 1

PROGRAM AND MACHINE CONTROL

5 **JMP** = Unconditional Jump

Address
(PC <- Address)

9... 3 2 1 0	9 8 7 6 5 4 3 2 1 0
addr	0 0 0 0 0 0 0 1 1 0

6 **JZ** = Jump on Zero

Address
(PC <- Address if Z = 1)

9... 3 2 1 0	9 8 7 6 5 4 3 2 1 0
addr	0 0 0 0 0 0 0 1 1 1

7 **JNZ** = Jump on Not Zero

Address
(PC <- Address S if Z = 0)

9... 3 2 1 0	9 8 7 6 5 4 3 2 1 0
addr	0 0 0 1 0 0 0 1 1 1

8 **JC** = Jump on Carry

Address
(PC <- Address if C = 1)

9... 3 2 1 0	9 8 7 6 5 4 3 2 1 0
addr	0 0 1 0 0 0 0 1 1 1

9 **JNC** = Jump on Not Carry

Address
(PC <- Address if C = 0)

9... 3 2 1 0	9 8 7 6 5 4 3 2 1 0
addr	0 0 1 1 0 0 0 1 1 1

10 **CALL** = Call

Address
(SP <- PC; PC <- Address)

9... 3 2 1 0	9 8 7 6 5 4 3 2 1 0
addr	0 0 0 0 0 0 1 0 0 0

11 **RET** = Return

9 8 7 6 5 4 3 2 1 0
0 0 0 0 0 0 1 0 0 1

(PC <- SP)

Description	Mnemonic code

12 **RETI** = Return from Interrupt

9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	1

(PC <- SP)

ARITHMETIC

13 **ADD** = Add

A, #K
(A <- A + K)

9	8	7 ... 1	0	9	8	7	6	5	4	3	2	1	0
0	0	K		0	0	0	0	0	0	1	1	0	0

A, Reg
(A <- A + Reg)

9 ... 3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 ... 0	0	reg addr		0	0	0	1	0	0	1	1	0	0

A, @K
(A <- A + Mem[K])

9 ... 3	2	1	0	9	8	7	6	5	4	3	2	1	0
mem addr				0	0	1	0	0	0	1	1	0	0

A, @Reg
(A <- A + Mem[Reg])

9 ... 3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 ... 0	0	reg addr		0	0	1	1	0	0	1	1	0	0

14 **SUB** = Substract

A, #K
(A <- A – K)

9	8	7 ... 1	0	9	8	7	6	5	4	3	2	1	0
0	0	K		0	0	0	0	0	0	1	1	0	1

A, Reg
(A <- A – Reg)

9 ... 3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 ... 0	0	reg addr		0	0	0	1	0	0	1	1	0	1

A, @K
(A <- A - Mem[K])

9 ... 3	2	1	0	9	8	7	6	5	4	3	2	1	0
mem addr				0	0	1	0	0	0	1	1	0	1

A, @Reg
(A <- A - Mem[Reg])

9 ... 3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 ... 0	0	reg addr		0	0	1	1	0	0	1	1	0	1

15 **INC** = Increment

A, #K
(A <- K +1)

9	8	7 ... 1	0	9	8	7	6	5	4	3	2	1	0
0	0	K		0	0	0	0	0	0	1	1	1	0

A, Reg
(A <- Reg + 1)

9 ... 3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 ... 0	0	reg addr		0	0	0	1	0	0	1	1	1	0

A, @K
(A <- Mem[K] + 1)

9 ... 3	2	1	0	9	8	7	6	5	4	3	2	1	0
mem addr				0	0	1	0	0	0	1	1	1	0

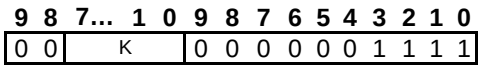
A, @Reg
(A <- Mem[Reg] + 1)

9 ... 3	2	1	0	9	8	7	6	5	4	3	2	1	0
0 ... 0	0	reg addr		0	0	1	1	0	0	1	1	1	0

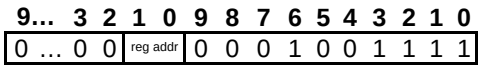
Description	Mnemonic code

16 **DEC** = Decrement

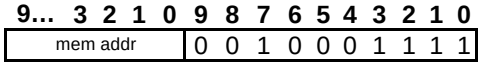
A, #K
(A <- K -1)



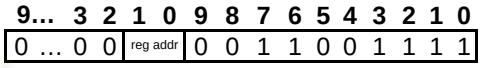
A, Reg
(A <- Reg - 1)



A, @K
(A <- Mem[K] - 1)

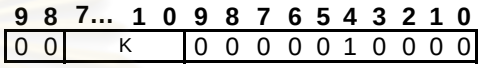


A, @Reg
(A <- Mem[Reg] - 1)

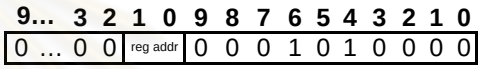


17 **MUL** = Multiplication

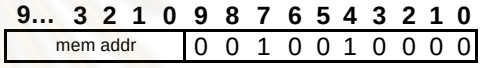
A, #K
(A <- A*K)



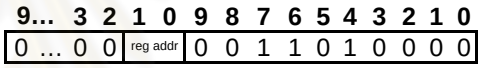
A, Reg
(A <- A*Reg)



A, @K
(A <- A*Mem[K])



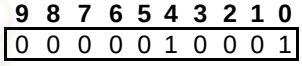
A, @Reg
(A <- A*Mem[Reg])



LOGIC

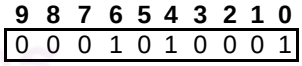
18 **SLL** = Shift Logical Left

A
(A <- A[MSB-1..0] & '0')



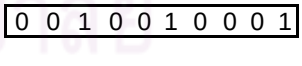
19 **SLR** = Shift Logical Right

A
(A <- '0' & A[MSB..1])



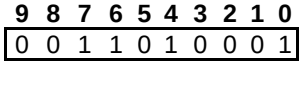
20 **SAL** = Shift Arithmetic Left

A
(A <- A[MSB-1..0] & 'A[LSB]')



21 **SAR** = Shift Arithmetic Right

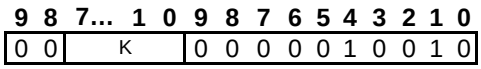
A
(A <- 'A[MSB]' & A[MSB..1])



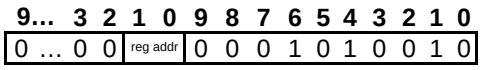
Description	Mnemonic code

22 **AND** = Logical And

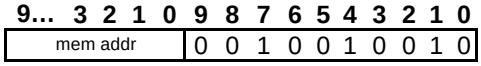
A, #K
(A <- A AND K)



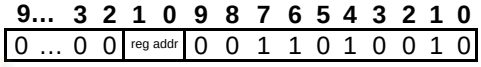
A, Reg
(A <- A AND Reg)



A, @K
(A <- A AND Mem[K])

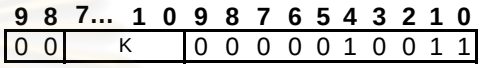


A, @Reg
(A <- A AND Mem[Reg])

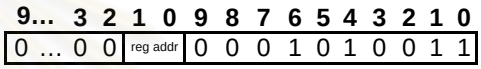


23 **OR** = Logical Or

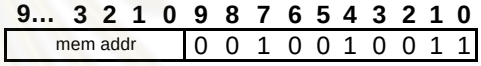
A, #K
(A <- A OR K)



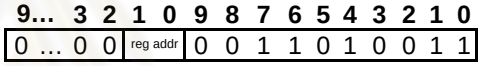
A, Reg
(A <- A OR Reg)



A, @K
(A <- A OR Mem[K])

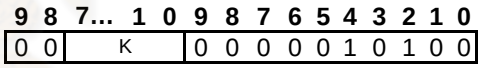


A, @Reg
(A <- A OR Mem[Reg])

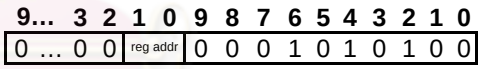


24 **XOR** = Logical Exclusive or

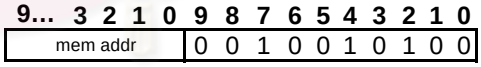
A, #K
(A <- A XOR K)



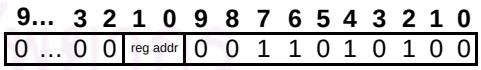
A, Reg
(A <- A XOR Reg)



A, @K
(A <- A XOR Mem[K])

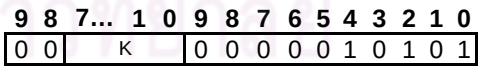


A, @Reg
(A <- A XOR Mem[Reg])

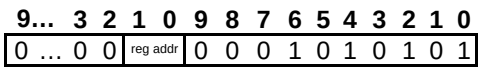


25 **ANDB** = Bitwise And

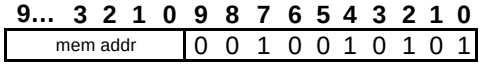
A, #K
(A <- A AND K)



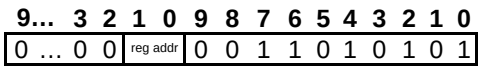
A, Reg
(A <- A AND Reg)

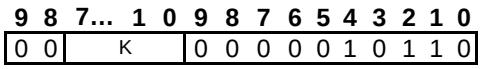
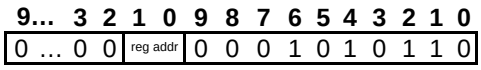
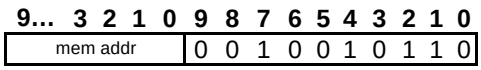
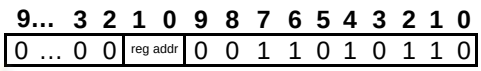
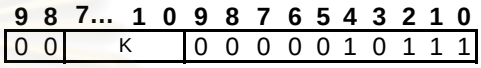
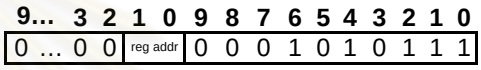
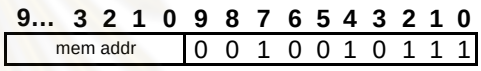
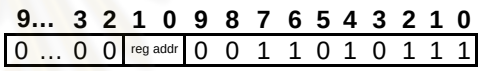


A, @K
(A <- A AND Mem[K])



A, @Reg
(A <- A AND Mem[Reg])



Description	Mnemonic code
26 **ORB** = Bitwise Or	
A, #K (A <- A OR K)	
A, Reg (A <- A OR Reg)	
A, @K (A <- A OR Mem[K])	
A, @Reg (A <- A OR Mem[Reg])	
27 **XORB** = Bitwise Exclusive or	
A, #K (A <- A XOR K)	
A, Reg (A <- A XOR Reg)	
A, @K (A <- A XOR Mem[K])	
A, @Reg (A <- A XOR Mem[Reg])	

หมายเหตุ: ความหมายของสัญลักษณ์มีดังนี้

A หมายถึง รีจิสเตอร์ตัวสะสม

K หมายถึง ค่าคงที่

Reg หมายถึง รีจิสเตอร์ทั่วไป ('01' คือรีจิสเตอร์ R0, '10' คือรีจิสเตอร์ R1)

Address หมายถึง ตำแหน่งอ้างอิงในหน่วยความจำสำหรับโปรแกรม

หมายถึง การอ้างอิงค่าคงที่

@หมายถึง การอ้างอิงค่าในหน่วยความจำสำหรับข้อมูล

ภาคผนวก ข

คำศัพท์ที่ใช้ในวิทยานิพนธ์

	คำศัพท์ภาษาไทย	คำศัพท์ภาษาอังกฤษ
ก	กราฟบรรยายการเปลี่ยนสัญญาณ	Signal Transition Graph (STG)
	กราฟแสดงสถานะ	State Graph
	กลุ่มคำสั่งควบคุมทำงาน	Program and Machine Control Operation
	กลุ่มคำสั่งเคลื่อนย้ายข้อมูล	Data Transfer Operation
	กลุ่มคำสั่งดำเนินการทางคณิตศาสตร์ และตรรกะ	Arithmetic and Logic Operation
	เกตผกผัน	Weak Inverter
ข	ข้อมูลรวมชุด	Bundle Data
	เขียนข้อมูลลงหน่วยความจำ	Memory Write
	เขียนอินพุท/เอาท์พุท	I/O Write
ค	ความคลาดเคลื่อนของสัญญาณนาฬิกา	Clock Skew
	ค่าเวลาล่าช้า	Delay Time
	คำสั่งกระโดดแบบไร้เงื่อนไข	Jump (JMP)
	คำสั่งกระโดดเมื่อค่าในรีจิสเตอร์ตัวสะสม เท่ากับศูนย์	Jump Zero (JZ)
	คำสั่งกระโดดเมื่อค่าในรีจิสเตอร์ตัวสะสม มีตัวทด	Jump Carry (JC)
	คำสั่งกระโดดเมื่อค่าในรีจิสเตอร์ตัวสะสม ไม่เท่ากับศูนย์	Jump not Zero (JNZ)
	คำสั่งกระโดดเมื่อค่าในรีจิสเตอร์ตัวสะสม ไม่มีตัวทด	Jump not Carry (JNC)
	คำสั่งกระทำบิตต่อบิตออร์	Bitwise or (ORB)

คำศัพท์ภาษาไทย	คำศัพท์ภาษาอังกฤษ
คำสั่งกระทำบิตต่อบิตเอ็กคลูซีฟออร์.....	Bitwise exclusive-or (XORB)
คำสั่งกระทำบิตต่อบิตแอนด์.....	Bitwise and (ANDB)
คำสั่งกระทำลอจิกออร์.....	Logical or (OR)
คำสั่งกระทำลอจิกเอ็กคลูซีฟออร์.....	Logical exclusive-or (XOR)
คำสั่งกระทำลอจิกแอนด์.....	Logical and (AND)
คำสั่งกลับจากงานบริการอินเตอร์รัพท์.....	Return from Interrupt (RET)
คำสั่งกลับไปใช้งานโปรแกรมหลัก.....	Return (RET)
คำสั่งคูณ.....	Multiplicand (MUL)
คำสั่งบวก.....	Addition (ADD)
คำสั่งเพิ่มค่าหนึ่งค่า.....	Increment (INC)
คำสั่งเรียกใช้งานโปรแกรมย่อย.....	CALL
คำสั่งลดค่าหนึ่งค่า.....	Decrement (DEC)
คำสั่งลบ.....	Subtract (SUB)
คำสั่งเลื่อนทุกบิตไปทางขวา แบบคิดเครื่องหมาย.....	Shift Arithmetic Right (SAR)
คำสั่งเลื่อนทุกบิตไปทางขวา แบบไม่คิดเครื่องหมาย.....	Shift Logical Right (SLR)
คำสั่งเลื่อนทุกบิตไปทางซ้าย แบบคิดเครื่องหมาย.....	Shift Arithmetic Left (SAL)
คำสั่งเลื่อนทุกบิตไปทางซ้าย แบบไม่คิดเครื่องหมาย.....	Shift Logical Left (SLL)
คำสั่งสโตร์.....	Store
คำสั่งโหลด.....	Load
คำสั่งอิน.....	IN
คำสั่งเอาท์.....	OUT

	คำศัพท์ภาษาไทย	คำศัพท์ภาษาอังกฤษ
	แคชเก็บคำสั่ง	Instruction Cache
ง	งานของอินเตอร์รัพท์	Interrupt Service Routine (ISR)
จ	จำลองการทำงานแบบอิงเวลา	Simulate
ช	ชิปของสมาร์ตคาร์ด	Smartcard Chip
	ชุดคำสั่ง	Instruction Set
	ใช้พลังงาน	Power Consumption
ด	ดีเอ็มเอ	Direct Memory Access (DMA)
ต	ตอบรับอินเตอร์รัพท์	Interrupt Acknowledge
	ตัวกันผลการเด้ง	Debouncer
	ตัวขับบัส	Bus Driver
	ตัวควบคุมดีเอ็มเอ	DMA Controller
	ตัวควบคุมบัส	Bus Controller
	ตัวคูณ	Multiplier
	ตัวตั้ง	Multiplicand
	ตัวตัดสินใจ	Bus Arbiter
	ตัวต้านทานพูลดาวน์	Pull-down Resistor
	ตัวต้านทานพูลอัพ	Pull-up Resistor
	ตัวรับบัส	Bus Receiver
	ตัวเลือกแยกสัญญาณ	Demultiplexer
	ตัวเลือกรวมสัญญาณ	Multiplexer
	ตารางค้นหาแบบสี่อินพุท	4 Input LUTs
	ตารางตำแหน่งอินเตอร์รัพท์	Interrupt Vector Table (IVT)
	ตำแหน่งของคำสั่ง	Instruction Pointer (IP)
น	นับจำนวนการเปลี่ยนสถานะ ของสัญญาณ	Transition Count

	คำศัพท์ภาษาไทย	คำศัพท์ภาษาอังกฤษ
บ	บัฟเฟอร์ค่าควบคุม	Control Buffer
	บัส	Bus
	บัสข้อมูล	Data Bus
	บัสควบคุม	Control Bus
	บัสแบบเดดิเคต	Dedicated Bus
	บัสแบบมัลติเพล็กซ์	Multiplex Bus
	บัสระบบแบบอสมวาร	Asynchronous System Bus
	บัสเลขที่อยู่	Address Bus
	บูทอัลกอลิทึม	Booth Agorithm
	แบบจำลองความหน่วง	Delay Model
	แบบจำลองความหน่วงที่ไม่ไว	
	ต่อความหน่วง	Delay Insensitive (DI)
ป	ประมาณค่าการใช้อุปกรณ์บนเอฟพีจีเอ	Device Utilization Summary
ผ	ผลลัพธ์	Product
	แผนที่คาร์นอฟ	Karnaugh Map
	แผนภาพตัดสินใจแบบทวิภาค	Binary Decision Diagram
	ชนิดมีการลดทอนอันดับ	Reduced-Ordered-Binary Decision Diagram (ROBDD)
พ	พลังงานที่ใช้เปลี่ยนสถานะของสัญญาณ	Switching Power
	พลังงานไดนามิก	Dynamic Power
	พลังงานไฟฟ้ารั่วไหล	Leakage Power
	พลังงานสแตติก	Static Power
	พอร์ทอินพุทเอาท์พุท	Bonded IOBs
	พีชคณิตบูลีน	Boolean Algebra
ฟ	ไฟโฟ	First-in-first-out (FIFO)

	คำศัพท์ภาษาไทย	คำศัพท์ภาษาอังกฤษ
ม	ไมโครโพรเซสเซอร์แบบอสมวาร	Asynchronous Processor
ร	รหัสฐานสอง	Binary Code
	รหัสฐานสองจำนวน n บิต	n'b
	รหัสรางคู่	Dual-rail Code
	รหัสหนึ่งในสิบหก	1-of-16 Code
	รหัสหนึ่งในสี่	1-of-4 Code
	รหัสหนึ่งในสี่จำนวน n บิต	n'1of4
	ร้องขออินเตอร์รัพท์	Interrupt Request
	ระดับเกต	Gate Level
	ระดับทรานซิสเตอร์	Transistor Level
	ระบบตอบรับ	Acknowledgement Circuits
	รีจิสเตอร์	Register
	รีจิสเตอร์ฐานตัวนับ	Base Word Count Register
	รีจิสเตอร์ฐานเลขที่อยู่ต้นทาง	Base Source Address Register
	รีจิสเตอร์ฐานเลขที่อยู่ปลายทาง	Base Destination Address Register
	รีจิสเตอร์ตัวนับปัจจุบัน	Current Word Count Register
	รีจิสเตอร์ตัวสะสม	Accumulator Register (ACC, A)
	รีจิสเตอร์พักข้อมูล	Temp Register
	รีจิสเตอร์เลขที่อยู่ต้นทางปัจจุบัน	Current Source Address Register
	รีจิสเตอร์เลขที่อยู่ปลายทางปัจจุบัน	Current Destination Address Register
	รีเซ็ต	Reset
ล	ลอจิกสามสถานะ	Tri-state Buffer
	แลตช์	Latch
ว	วงจรควบคุมที่ไม่ขึ้นต่ออัตราเร็ว	Speed-independent Control Circuits

คำศัพท์ภาษาไทย	คำศัพท์ภาษาอังกฤษ
วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ ใช้ฟังก์ชันเข้ารหัสรางคู่	1-of-4 Radix-4 Booth Multiplier with Dual-rail Function Unit
วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่	1-of-4 Radix-4 Booth Multiplier with 1-of-4 Function Unit
วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ ใช้ฟังก์ชันเข้ารหัสรางคู่	1-of-4 Radix-2 Booth Multiplier with Dual-rail Function Unit
วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่	1-of-4 Radix-2 Booth Multiplier with 1-of-4 Function Unit
วงจรถอดรหัสคำสั่งดีเอ็มเอ	DMA Decoder
วงจรเพิ่มค่า	Incrementor
วงจรลดค่า	Decrementor
วงจรเลือกหลักสำหรับวงจรเพิ่มค่าเข้ารหัสหนึ่งในสี่	1-of-4 Selector
วงจรสมวาร	Synchronous Circuit
วงจรสร้างสัญญาณควบคุมหน่วยความจำ	Memory Controller
วงจรอสมวาร	Asynchronous Circuits
วิถีวิกฤต	Critical Path
เวลาของสัญญาณนาฬิกาหนึ่งลูก	Clock Cycle Time
เวลาในเข้าถึงหน่วยความจำ	Memory Access Time

	คำศัพท์ภาษาไทย	คำศัพท์ภาษาอังกฤษ
ส	สถานะ	Status
	สถานะของสัญญาณ	State Transition
	สไลด์	Slice
	ส่วน	Path
	ส่วนเขียนผลลัพธ์	Write Result Unit
	ส่วนควบคุม	Control Unit
	ส่วนติดต่อกับแอลซีดี	LCD Interface
	ส่วนติดต่อกับอุปกรณ์ต่อพ่วง	I/O Interface
	ส่วนติดต่อของบัส	Bus Interface
	ส่วนติดต่อของหน่วยความจำ	Memory Interface
	ส่วนบริการอินเตอร์รัพท์	Interrupt Unit
	ส่วนประมวลผล	Execute Unit
	ส่วนแปลความหมายของคำสั่ง	Decode Unit
	ส่วนพักข้อมูล	Latch
	ส่วนพักข้อมูลต้นทาง	Source Latch
	ส่วนพักข้อมูลปลายทาง	Destination Latch
	ส่วนฟังก์ชัน	Function Unit
	ส่วนรับส่งข้อมูล	Data Path
	ส่วนอ่านคำสั่ง	Fetch Unit
	สังเคราะห์	Synthesis
	สัญญาณตอบรับ	Acknowledge Signal
	สัญญาณนาฬิกา	Clock
	สัญญาณร้องขอ	Request Signal
	สัญญาณอาณัติแบบ 4 ขั้น	4-Cycle Protocol, 4 Phase Protocol
	สายท่อแบบอสมวาร	Asynchronous Pipelines

	คำศัพท์ภาษาไทย	คำศัพท์ภาษาอังกฤษ
	สายสัญญาณบัส	Bus Line
ห	หน่วยคำนวณทางคณิตศาสตร์และตรรกะ	Arithmetic and Logic Units (ALU)
อ	อนุญาตให้ใช้บัส	Bus Grant
	อ่านข้อมูลจากหน่วยความจำ	Memory Read
	อ่านอินพุท/เอาท์พุท	I/O Read
	อิมพลิเมนท์	Implement
	อุปกรณ์ต่อพ่วง	I/O
	อุปกรณ์ไปป์ไลน์	Asynchronous Pipeline Module
	อุปกรณ์สหสัญญาณ	Multiplexer
	อุปกรณ์ออโต้สวีป	Autosweeping Module (ASM)
	เอฟพีจีเอแบบสมวาร	Synchronous FPGA
	เอฟพีจีเอแบบอสมวาร	Asynchronous FPGA
	แอนด์เกต	And-gate
ฮ	ไฮ-อิมพีแดนซ์	High Impedance (Hi-Z)

ประวัติผู้เขียนวิทยานิพนธ์

นางสาว กิตติมา ฐานพีรภัทร์ เกิดเมื่อวันที่ 17 เมษายน พ.ศ. 2528 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสยาม ในปีการศึกษา 2549 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ปีการศึกษา 2550