การออกแบบบัสระบบสำหรับวงจรอสมวาร

นายสุฟียัน สูเด็ง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2549

A DESIGN OF SYSTEM BUS FOR ASYNCHRONOUS CIRCUITS

Mr. Sufian Sudeng

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 2006

หัวข้อวิทยานิพนธ์	การออกแบบบัสระบบสำหรับวงจรอสมวาร
โดย	นายสุฟียัน สูเด็ง
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร.อาทิตย์ ทองทักษ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร.ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(รองศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา)

..... อาจารย์ที่ปรึกษา

(อาจารย์ ดร.อาทิตย์ ทองทักษ์)

..... กรรมการ

(นายชำนาญ ปัญญาใส)

..... กรรมการ

(รองศาสตราจารย์ ดร.สาธิต วงศ์ประทีป)

สุฟียัน สูเด็ง : การออกแบบบัสระบบสำหรับวงจรอสมวาร. (A DESIGN OF SYSTEM BUS FOR ASYNCHRONOUS CIRCUITS) อ. ที่ปรึกษา : ดร. อาทิตย์ ทองทักษ์, 134 หน้า.

วิทยานิพนธ์ฉบับนี้นำเสนอการออกแบบบัสระบบเพื่อเป็นต้นแบบสำหรับวงจรอสมวาร ประกอบด้วย การออกแบบบัส อินเตอร์รัพท์ ดีเอ็มเอ และ นำเสนอการเชื่อมต่อระหว่างวงจรอสมวารและอสมวาร, ระบบที่ออกแบบประกอบด้วยไมโครโพรเซสเซอร์แบบอสมวาร ที่ดัดแปลงในส่วน คำสั่ง, โครงสร้าง และส่วนควบคุม ระบบบัสแบบอสมวาร ดีเอ็มเอ ซึ่งออกแบบเป็นวงจรสมวารและอสมวาร ส่วนควบคุมอินพุต/เอาท์พุต สำหรับควบคุมการรับส่งข้อมูลของอินพุต/เอาท์พุตแบบสมวาร

บัสแบบอสมวารออกแบบโดยแบ่งออกเป็น 6 ส่วน ประกอบด้วย ส่วนเชื่อมต่อบัส, ตัวขับบัส, สายสัญญาณบัส, ตัวควบคุมบัส, ตัวรับบัส และส่วนเชื่อมต่อวงจรสมวาร ส่วนเชื่อมต่อบัส ออกแบบสำหรับเชื่อมต่อระหว่างบัส ไมโครโพรเซสเซอร์ และดีเอ็มเอ ตัวขับบัส ออกแบบให้ทำงานในลักษณะสวิทช์ที่ใช้ในการเปิดปิดสัญญาณเข้าสู่บัส เกตผกผัน ต่อกับสายสัญญาณบัสเพื่อให้บัสใช้กับวงจรอสมวารได้ ตัวรับบัส ใช้เพื่อรับสัญญาณจากบัสและส่งต่อไปยังปลายทาง. ตัวควบคุมบัส ออกแบบโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ ใช้ในการควบคุมการทำงานของบัส

ดีเอ็มเอ ออกแบบเป็นทั้งวงจรสมวาร และวงจรอสมวาร ดีเอ็มเอแบบอสมวารออกแบบโดยใช้เครื่องจักรสถานะ ส่วนดีเอ็มเอแบบอสมวาร ออกแบบโดยใช้การเข้ารหัสแบบโครงสร้าง ส่วนควบคุมอินพุต/เอาท์พุต ออกแบบเพื่อควบคุมการติดต่อกับอินพุต/เอาท์พุตส่วนที่เป็นวงจรสมวารทุกชนิด เช่น จอแสดงผล, แป้นพิมพ์, ปุ่มต่าง ๆ เป็นต้น

วงจรที่ออกแบบทดสอบโดยการจำลองการทำงาน เมื่อได้การทำงานที่ถูกต้องแล้ว จึงนำมาทดสอบการทำงานจริงกับบอร์ดทดลองของบริษัท Xilinx โดยใช้ เอฟพีจีเอ เบอร์ 3S500EFG320 ในการอิมพลีเมนท์จะกำหนดพาติชั่นเพื่อป้องกันไม่ให้ส่วนของวงจรเกิดการเชื่อมสายที่ผิดพลาด หลังจากมีการปรับปรุงวงจรในภายหลัง จากการทดสอบปรากฏว่าวงจรทำงานได้อย่างถูกต้อง และใช้เกตจำนวน 141,063 เกต สำหรับวงจรที่ออกแบบรวมกับดีเอ็มเอแบบอสมวาร และ 141,587 เกต สำหรับวงจรที่ออกแบบรวมกับดีเอ็มเอแบบสมวาร

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต สุฟียัน สูเด็ง

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา อาทิตย์ ทองทักษ์

ปีการศึกษา 2549

4670566021 : MAJOR COMPUTER ENGINEERING

KEY WORD: SYSTEM BUS / TRI-STATE / DMA / INTERRUPT / STG/STRUCTURAL ENCODING

SUFIAN SUDENG : A DESIGN OF SYSTEM BUS FOR ASYNCHRONOUS CIRCUITS.

THESIS ADVISOR: ARTHIT THONGTAK, 134 pp.

This thesis proposes a system bus design for asynchronous circuits. Interrupt technique and DMA method are included in the design for bus. Moreover, an interfacing between synchronous and asynchronous circuits with I/O capability is also shown, our proposed system composes of a reference asynchronous processor, which instruction, architecture and control unit have been modified. The asynchronous system bus, DMA controller, both synchronous and asynchronous circuits I/O controller are demonstrated.

The bus structure is divided into six components Bus Interface, Bus Driver, Bus lines, Bus Controller, Bus Receiver, and Synchronous Interface. Bus interface is designed to interface with an asynchronous processor. Bus Driver acts like an on/off switch to enable/disable transition signal on the bus. The weak inverters are added on each bus line to remove high impedance. Bus receiver receives data from bus and transfer to the destination. Bus controller is designed with STG to control the bus synchronization.

Two versions of DMA controller are introduced: synchronous and asynchronous. The state machine model is used for a synchronous version, while structural encoded STG is implemented for an asynchronous version. I/O controller is responsible for arbitrating between two or more I/O modules such as the display, the keyboard and other inputs.

The proposed design has been implemented on Spartan-3E FPGA no. 3S500EFG320 by partitioning each module in order to prevent place and routing conflict. Finally, time simulation and board demonstration are shown. The design consumes 141,063 gates count for system with asynchronous DMA controller and 141,587 gates count for system with synchronous DMA controller.

Department ....Computer Engineering.... Student's signature....Sufian Sudeng....

Field of study..Computer Engineering..... Advisor's signature....Arthit Thongtak....

Academic year .....2006.....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยความกรุณาอย่างดียิ่งของอาจารย์ ดร. อาทิตย์ ทองทักษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งได้ให้ข้อคิดเห็น และคำแนะนำต่าง ๆ แก่ผู้วิจัยมาโดยตลอด เป็นโอกาสที่ดีสำหรับผู้วิจัย ที่ได้ทำงานร่วมกับท่าน และได้ศึกษาเรียนรู้ ความรู้ความสามารถจากท่าน ผู้วิจัยหวังว่า จะสามารถนำความรู้ที่ได้เรียนรู้ทั้งหมดไปศึกษาพัฒนา และเผยแพร่ต่อไปในอนาคต

ขอขอบคุณ คุณตะวัน ภูรัต นิสิตปริญญาเอก สังกัดห้องปฏิบัติการวิศวกรรมระบบดิจิตอล ผู้วิจัยได้มีโอกาส ศึกษาความรู้เบื้องต้นต่าง ๆ เพื่อเป็นนักวิจัย จากท่านได้เป็นอย่างดี

ขอขอบคุณห้องปฏิบัติการวิศกรรมระบบดิจิตอล(Digital System Engineering Laboratory-DSEL) ที่เอื้อเฟื้อสถานที่และอุปกรณ์ในการทำวิจัย

ขอบขอบคุณสมาชิกห้องปฏิบัติการวิศวกรรมระบบดิจิตอล ทุกท่านที่ไม่ได้กล่าวถึง การสัมนารายสัปดาห์ ของห้องปฏิบัติการวิศวกรรมระบบดิจิตอล เป็นโอกาสที่ดี ในการแลกเปลี่ยนความรู้ซึ่งกันและกัน ทำให้เกิดการร่วมกันเสนอแนวคิดดี ๆ ในการทำวิจัยมากขึ้น

ขอบคุณคณะกรรมการสอบวิทยานิพนธ์ รองศาสตราจารย์ ดร ประภาส จงสถิตย์วัฒนา, รองศาสตราจารย์ ดร สาธิต วงศ์ประทีบ, และ คุณชำนาญ ปัญญาใส ในการแนะนำ และขัดเกลาวิทยานิพนธ์ฉบับนี้ให้สำเร็จลุล่วงไปได้ด้วยดี

ท้ายที่สุดนี้ ผู้วิจัยขอขอบพระคุณ บิดา มารดา ที่ให้กำลังใจ และการสนับสนุนผู้วิจัยโดยเสมอมา

สารบัญ

หน้า

หน้า

หน้า

หน้า

สารบัญภาพ

หน้า

หน้า

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

การออกแบบวงจรดิจิตอลสามารถแบ่งการออกแบบใหญ่ ๆ ได้ 2 แบบด้วยกัน คือวงจรสมวารและวงจรอสมวาร [1] วงจรสมวาร เป็นวงจรที่ใช้สัญญาณนาฬิกาจากภายนอกเป็นตัวควบคุมวงจรการทำงานอยู่ในรูปของแผนภาพเชิงเวลา ที่มีสัญญาณนาฬิกาเป็นตัวควบคุมวงจร ทำให้วงจรสามารถออกแบบและทดสอบได้ง่าย เป็นที่นิยมเรื่อยมาถึงปัจจุบัน แต่เนื่องจากเทคโนโลยีไมโครอิเล็กทรอนิกส์ได้ถูกพัฒนาเพิ่มมากขึ้นทำให้วงจรต่าง ๆ ที่ออกแบบมีขนาดเล็กลงและความเร็วสูงทำให้การออกแบบวงจรแบบสมวารนั้นมีข้อจำกัด การเปลี่ยนสัญญาณในวงจรจะทำให้เกิดการแกว่งของสัญญาณนาฬิกา (Clock skew) ทำให้วงจรทำงานผิดพลาดได้ การแกว่งของสัญญาณนาฬิกานับว่าเป็นปัญหาใหญ่ของวงจรสมวารที่ต้องการออกแบบที่ความถี่สูง

วงจรอสมวาร (Asynchronous Circuits) เป็นที่รู้จักและแพร่หลายกันอย่างกว้างขวางมานาน แต่เนื่องจากการออกแบบนั้นซับซ้อนมากกว่าวงจรสมวารทำให้การวิจัยนั้นอยู่ในวงจำกัด วงจรอสมวารจะไม่ใช้สัญญาณนาฬิกาในการควบคุม การควบคุมของวงจรอสมวารจะยึดโดยเหตุการณ์ที่เกิดขึ้นในวงจร (Event Driven) การทำงานของวงจรเกิดขึ้นเมื่อมีการเปลี่ยนแปลงของสายสัญญาณและสามารถให้เอาท์พุทได้ทันทีเมื่อทำงานเสร็จ ความเร็วของวงจรจะไม่ถูกจำกัดด้วยสัญญาณนาฬิกาเหมือนวงจรสมวารที่ถูกกำหนดให้ความเร็วของวงจรขึ้นอยู่กับสัญญาณนาฬิกาส่วนที่ทำงานช้าที่สุดเพื่อให้วงจรเข้าจังหวะกันได้อย่างถูกต้อง วงจรอสมวารยังยังมีข้อดีเหนือวงจรสมวารอีกหลายประการ เช่นประหยัดพลังงานมากกว่าเนื่องจากวงจรอสมวารจะใช้พลังงานเมื่อส่วนของวงจรทำงานจริงเท่านั้น ต่างจากวงจรสมวารที่ต้องมีการใช้พลังงานในการกำเนิดสัญญาณนาฬิกาอยู่ตลอดเวลา ประสิทธิภาพของวงจรอสมวารก็จะอยู่ในระดับเฉลี่ยในขณะที่วงจรสมวารประสิทธิภาพจะอยู่ในระดับช้าที่สุด [2]

จะเห็นว่ามีเหตุผลหลายประการด้วยกันที่ทำให้การออกแบบวงจรอสมวารนั้นเร็วและมีประสิทธิภาพมากกว่าการออกแบบวงจรแบบสมวาร งานวิจัยเกี่ยวกับวงจรอสมวารในปัจจุบันส่วนใหญ่จะเน้นทางด้านการพัฒนาความเร็ว การคำนวณ ซึ่งจะอยู่ในรูปแบบการออกแบบและพัฒนาวงจรทางด้านหน่วยประมวลผลตรรกะ จนถึงออกแบบไมโครโพรเซสเซอร์ ปัจจุบันยังขาดงานวิจัยเกี่ยวกับการออกแบบวิธีการส่งผ่านข้อมูลจากโพรเซสเซอร์ไปยังอุปกรณ์ต่าง ๆ เนื่องจากอาจทำให้วงจรที่ออกแบบไม่เป็นไปตามสมมุติฐานเกี่ยวกับทฤษฎีแบบจำลองความหน่วงของวงจรอสมวาร

ทำให้วงจรที่ได้ทำงานผิดพลาดได้ อีกทั้งปัจจุบันยังไม่มีอุปกรณ์ต่อพ่วง (I/O Devices) ที่เป็นแบบอสมวาร โดยสมบูรณ์แบบ ซึ่งเป็นเหตุผลหนึ่งที่ทำให้นักวิจัยเลือกที่จะพัฒนาส่วนอื่นแทน

งานวิจัยนี้ศึกษาการออกแบบวงจรแบบอสมวาร (Asynchronous Circuits) เพื่อพัฒนาขอบเขตการศึกษาวงจรอสมวาร โดยมีวัตถุประสงค์เพื่อให้สามารถออกแบบระบบบัสแบบอสมวาร (Asynchronous System Bus) ได้ โดยออกแบบโดยใช้รหัสรางคู่เข้ารหัสในการส่งข้อมูล และใช้โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น (4-Cycle Protocol,4-Phase Protocol) ระบบบัสที่ออกแบบ สามารถเชื่อมต่อกับไมโครโพรเซสเซอร์แบบอสมวาร (Asynchronous Processor) หน่วยความจำแบบสมวาร และ อุปกรณ์อินพุต/เอาท์พุท(I/O Device) แบบสมวาร ในส่วนของการเชื่อมต่อกับอุปกรณ์อินพุต/เอาต์พุทออกแบบโดยใช้เทคนิคการเชื่อมต่อระบบสมวารกับระบบอสมวารโดยใช้สัญญาณนาฬิกาแบบยืดหยุ่น (Stretchable Clock) [9] รวมถึงการใช้เทคนิคการเพิ่มความเร็วของบัสโดยการใช้ อินเตอร์รัพท์และ DMA ได้

1.2 วัตถุประสงค์ของการวิจัย

เพื่อออกแบบบัสระบบสำหรับวงจรอสมวาร และสามารถเชื่อมต่อกับไมโครโพรเซสเซอร์แบบอสมวาร หน่วยความจำ และอุปกรณ์ต่อพ่วงในปัจจุบันได้ รวมถึงการออกแบบอินเตอร์รัพท์และดีเอ็มเอ เพื่อเป็นส่วนเพิ่มเติมสำหรับบัสแบบอสมวาร

1.3 ขอบเขตของการวิจัย

1.ออกแบบบัสแบบอสมวารขนาด 8 บิต โดยใช้การเข้าหัสแบบรางคู่ และสามารถเชื่อมต่อกับไมโครโพรเซสเซอร์แบบอสมวารและอุปกรณ์ต่อพ่วงได้

2.ประกอบด้วยอุปกรณ์ต่อพ่วง 2 ตัวเป็นอย่างน้อย

3. ออกแบบในส่วนอินเตอร์รัพท์และดีเอ็มเอเพื่อใช้กับบัสได้

4. ทดสอบระบบโดยการจำลองการทำงาน

1.4 ขั้นตอนการดำเนินงานวิจัย

1. ศึกษาการออกแบบบัสสำหรับคอมพิวเตอร์, ชนิดของบัส, การทำงานของบัส, การแลกเปลี่ยนข้อมูล เพื่อเป็นความรู้พื้นฐานสำหรับงานวิจัย
2. ศึกษาหลักการของดีเอ็มเอและอินเตอร์รัพท์
3. ศึกษาการออกแบบบัสแบบอสมวาร

4. ศึกษาวิธีการเชื่อมต่อระหว่างวงจรสมวารและอสมวาร
5. ศึกษาการออกแบบวงจรอสมวาร
6. ออกแบบระดับบน
7. ออกแบบแต่ละส่วนของวงจร จำลองการทำงาน และทดสอบ
8. รวมทุกส่วนเข้าด้วยกันและจำลองการทำงาน.
9. โปรแกรมวงจรที่ออกแบบลงเอฟพีจีเอ และทดสอบการทำงานจริงกับบอร์ดทดลอง
10. สรุปผลการวิจัย และจัดทำวิทยานิพนธ์

1.5 ประโยชน์ที่ได้รับ

1. สามารถออกแบบบัสแบบอสมวารได้
2. สามารถออกแบบการเชื่อมต่อระหว่างวงจรสมวารและอสมวารได้

1.6 ลำดับขั้นตอนในการเสนอการวิจัย

วิทยานิพนธ์ฉบับนี้แบ่งเนื้อหาออกเป็น 7 บท ดังนี้:

บทที่ 1 เป็นบทนำซึ่งกล่าวถึงที่มาและความสำคัญของปัญหา รวมทั้งวัตถุประสงค์ของงานวิจัย

บทที่ 2 สรุปแนวคิดและทฤษฎีที่เกี่ยวข้อง ดังนี้ การแบ่งหมวดหมู่ของวจรอสมวาร , แบบจำลองการทำงานสิ่งแวดล้อม, ช่องสัญญาณ, โปรโตคอลอาณัติสัญญาณ, การเข้ารหัส และ กราฟบรรยายการเปลี่ยนสัญญาณ การออกแบบบัส ความสำคัญของลอจิกสามสถานะ, ชนิดของบัส, การทำงานของบัส และการรับส่งข้อมูล และนำเสนอแนวคิดการออกแบบดีเอ็มเอและอินเตอร์รัพท์

บทที่ 3 นำเสนอวิธีการออกแบบบัสสำหรับวงจรอสมวาร และความสามารถในการเชื่อมต่อกับอุปกรณ์ต่อพ่วงในปัจจุบันได้ จากนั้นจึงได้นำเสนอวิธีการเข้ารหัสข้อมูลของบัส ตามด้วยนำเสนอการออกแบบแต่ละส่วน ของบัส แสดงการทำงานของบัสในตอนท้ายของบท

บทที่ 4 นำเสนอการออกแบบดีเอ็มเอ, ดีเอ็มเอจะออกแบบทั้งส่วนที่เป็นวงจรสมวาร และอสมวาร ดีเอ็มเอแบบสมวารออกแบบโดยใช้เครื่องจักรสถานะ, ส่วนดีเอ็มเอ

แบบอสมวารออกแบบโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณที่มีการเข้ารหัสแบบโครงสร้าง

บทที่ 5 รวมวงจรทั้งหมดเข้าด้วยกัน ประกอบด้วยไมโครโพรเซสเซอร์ ที่ผ่านการปรับแต่ง 3 ส่วนใหญ่ ๆ คือ โครงสร้าง รหัสดำเนินการ และส่วนควบคุม, การออกแบบแคช การเชื่อมต่ออุปกรณ์ต่อพ่วง

บทที่ 6 เสนอการจำลองการทำงานของวงจร จากนั้นจึงโปรแกรมลงเอฟพีจีเอ และจำลองการทำงานจริงกับบอร์ดทดลอง

บทที่ 7 สรุปผลการวิจัย และเสนอแนะ

1.7 ผลงานที่ตีพิมพ์จากงานวิจัย

ส่วนหนึ่งของวิทยานิพนธ์ได้ตีพิมพ์เป็นบทความทางวิชาการในหัวข้อ “A Design of System Bus for Asynchronous Circuits” โดย สุฟียัน สูเด็ง และ อาทิตย์ ทองทักษ์ ในงานประชุมวิชาการ “International Technical Conference in Circuits/Systems Computer and Communication (ITC-CSCC2006)” ณ โรงแรมโลตัสปางสวนแก้ว จ.เชียงใหม่, ประเทศไทย ระหว่างวันที่ 10-13 กรกฎาคม 2549

ส่วนหนึ่งของวิทยานิพนธ์ได้ตีพิมพ์เป็นบทความทางวิชาการในหัวข้อ “Asynchronous System Bus Enhancement by Interrupt and DMA technique” โดย สุฟียัน สูเด็ง และอาทิตย์ ทองทักษ์ ในงานประชุมวิชาการ “The fourth Annual International Conference (ECTI-CON 2007)” ซึ่งจัดโดย Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI) Association ณ มหาวิทยาลัยแม่ฟ้าหลวง จ.เชียงราย ประเทศไทย, ระหว่างวันที่ 9-12 พฤษภาคม 2550

บทที่ 2

แนวคิดและทฤษฎีที่เกี่ยวข้อง

บทนี้นำเสนอความรู้เบื้องต้นและงานวิจัยที่เกี่ยวข้อง แบ่งออกเป็น 6 ส่วนด้วยกัน โดยนำเสนองานวิจัยที่เกี่ยวข้องในหัวข้อที่ 2.1 จากนั้นนำเสนอวิธีการออกแบบวงจรอสมวารในหัวข้อที่ 2.2, หัวข้อที่ 2.3 นำเสนอวิธีการออกแบบบัสโดยทั่วไป หัวข้อที่ 2.4 นำเสนอแนวคิดการออกแบบดีเอ็มเอและการทำงานของดีเอ็มเอ หัวข้อที่ 2.5 นำเสนอแนวคิดการออกแบบอินเตอร์รัพท์ และสรุปในหัวที่ 2.6

2.1 งานวิจัยที่เกี่ยวข้อง

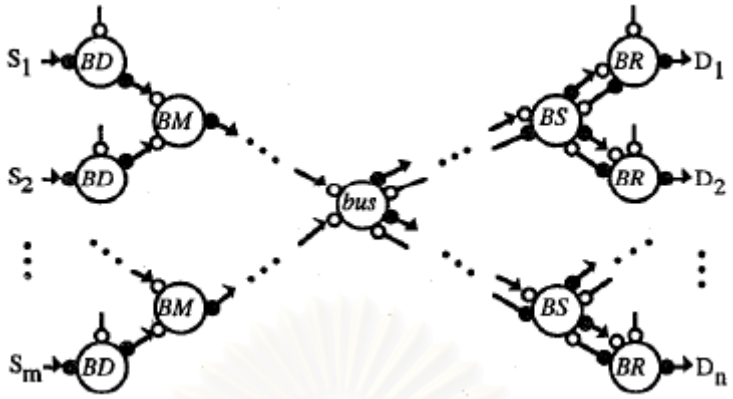
งานวิจัยที่เกี่ยวข้อง ประกอบด้วยงานวิจัยการออกแบบบัสแบบอสมวาร และงานวิจัยการซื่อมต่อวงจรสมวารกับวงจรอสมวาร

2.1.1 การออกแบบบัสแบบอสมวาร

2.1.1.1 การออกแบบบัสแบบอสมวารโดยใช้แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วงชนิดเสมือน

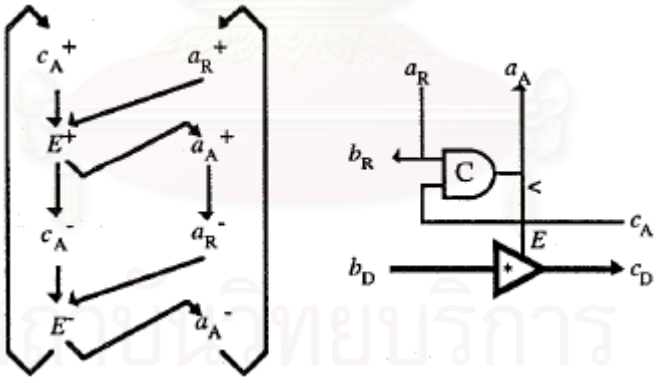
ปัจจุบันงานวิจัยเกี่ยวกับการออกแบบบัสแบบอสมวารนั้นยังไม่เป็นที่แพร่หลาย การวิจัยในระบบอสมวารจะเน้นทางด้าน การคำนวณ ความเร็ว การลดพลังงาน เป็นต้น อีกทั้งขณะนี้ยังไม่มีอุปกรณ์ประเภทอินพุต/เอาท์พุต ที่เป็นอสมวารสมบูรณ์แบบ ประกอบด้วยการออกแบบระบบบัสที่เป็นอสมวารนั้นทำได้ยาก เนื่องจากว่าจะทำให้สมมุติฐานเกี่ยวกับแบบจำลองความหน่วง(Delay Model) ถูกทำลายไป[5] ทำให้วงจรทำงานผิดพลาดได้สังเกตได้ว่างานวิจัยที่ต้องใช้บัสในการทดลองนั้นส่วนใหญ่จะไม่มีการใช้บัสร่วม(Common bus) แต่จะใช้การเชื่อมต่อส่วนประกอบแบบจุดต่อจุดแทนเพื่อให้เป็นไปตามสมมุติฐานตามแบบจำลองความหน่วงแบบต่าง ๆ งานวิจัยที่เกี่ยวกับการออกแบบระบบบัสแบบอสมวารที่นำมาอ้างอิงนี้เป็นการออกแบบระบบบัสแบบอสมวารโดยใช้แบบจำลองความหน่วงชนิดเสมือน (QDI) และใช้รหัสรางคู่ (Dual-rail Code)ในการเข้ารหัสสายสัญญาณของบัส [6]

ระบบบัสแบบอสมวารดังกล่าว ประกอบด้วยส่วนประกอบต่าง ๆ 5 ส่วน คือ ตัวขับบัส (Bus Driver) ตัวรวมบัส (Bus Merger) บัส (Bus) ตัวแยกบัส (Bus Splitter) และ ตัวรับบัส (Bus Receiver) ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 โครงสร้างทั่วไปของการออกแบบบัสแบบอสมวาร

ตัวขับบัส(Bus Driver) – จะอยู่ทางด้านฝั่งส่งข้อมูล ใช้ ลอจิกสามสถานะ (Tri-state Buffer) ในการออกแบบ เพื่อควบคุมการเปลี่ยนสัญญาณในบัส ใช้การเข้ารหัสด้วยรหัสรางคู่และโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น ตัวขับบัสจะไม่รีเซ็ตสายสัญญาณเมื่อบัสยังไม่ได้ถูกเลือกให้ใช้งาน

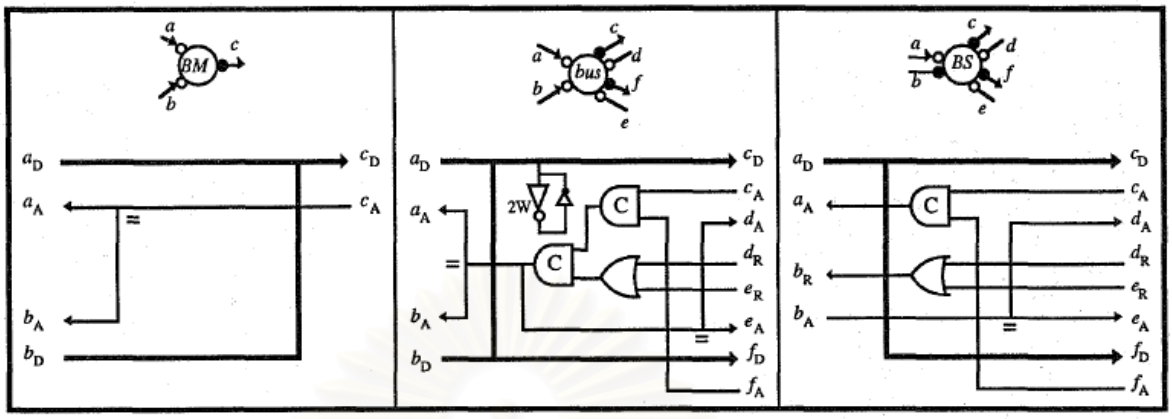


รูปที่ 2.2 กราฟบรรยายการเปลี่ยนสัญญาณและวงจรของตัวขับบัส

ตัวรวมบัส (Bus Merger) – ใช้รวมบัสในกรณีที่มีอินพุตหลาย ๆ ตัว โดยตัวรวมบัสแต่ละตัว มีอินพุทอยู่ 2 อินพุท เพื่อทำการรวมบัสให้ได้เอาท์พุต 1 เอาท์พุต

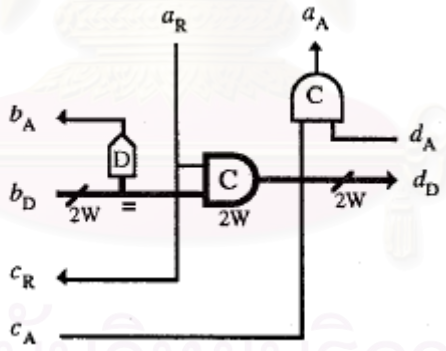
บัส (Bus) – เป็นส่วนเส้นทางผ่านข้อมูลร่วม ที่ข้อมูลสามารถผ่านไปยังฝั่งรับข้อมูลได้

ตัวแยกบัส (Bus Splitter) – ใช้ในการแยกเส้นทางของข้อมูล เพื่อส่งต่อให้ตัวรับบัส



รูปที่ 2.3 ตัวรวมบัส, บัส และ ตัวแยกบัส

ตัวรับบัส (Bus Receiver) – ประกอบด้วยฟังก์ชันการทำงานใหญ่ ๆ อยู่ 2 ฟังก์ชันด้วยกัน คือ ส่งสัญญาณตอบรับ (Acknowledge) กลับไปยังบัสเมื่อได้รับข้อมูลใหม่ และ ส่งข้อมูลไปยังส่วนของ เอาท์พุต

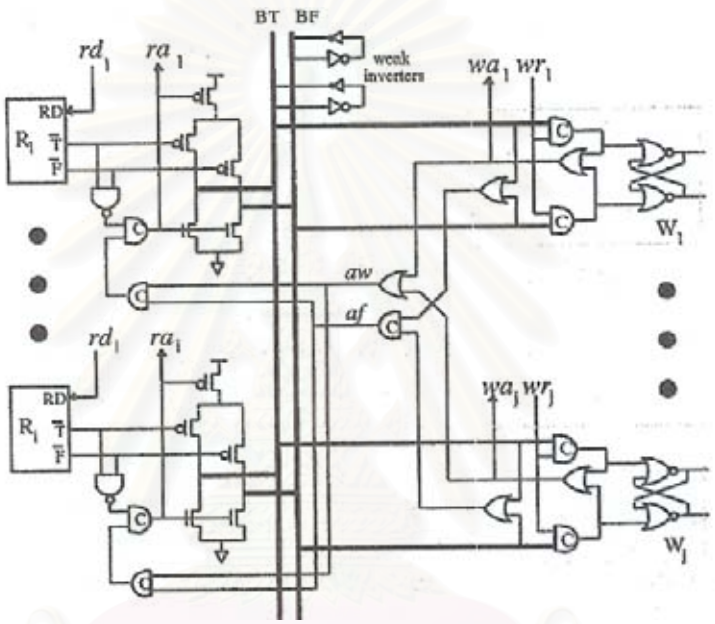


รูปที่ 2.4 วงจรของตัวรับบัส

ส่วนประกอบของบัสที่กล่าวข้างต้นนั้น ออกแบบโดยใช้แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วง (Delay-Insensitive Model) จากนั้นได้เพิ่มเทคนิคที่เรียกว่า Encapsulated Fork โดยการเพิ่มเติมส่วนของวงจรให้มีการตอบรับ (Handshake) อย่างสมบูรณ์แบบ เพื่อให้วงจรทำงานภายใต้แบบจำลองความหน่วงชนิดเสมือน (Quasi-Delay Insensitive Model) ได้อย่างถูกต้อง

2.1.1.2 การออกแบบบัสแบบอสมวารโดยใช้รีจิสเตอร์ [6]

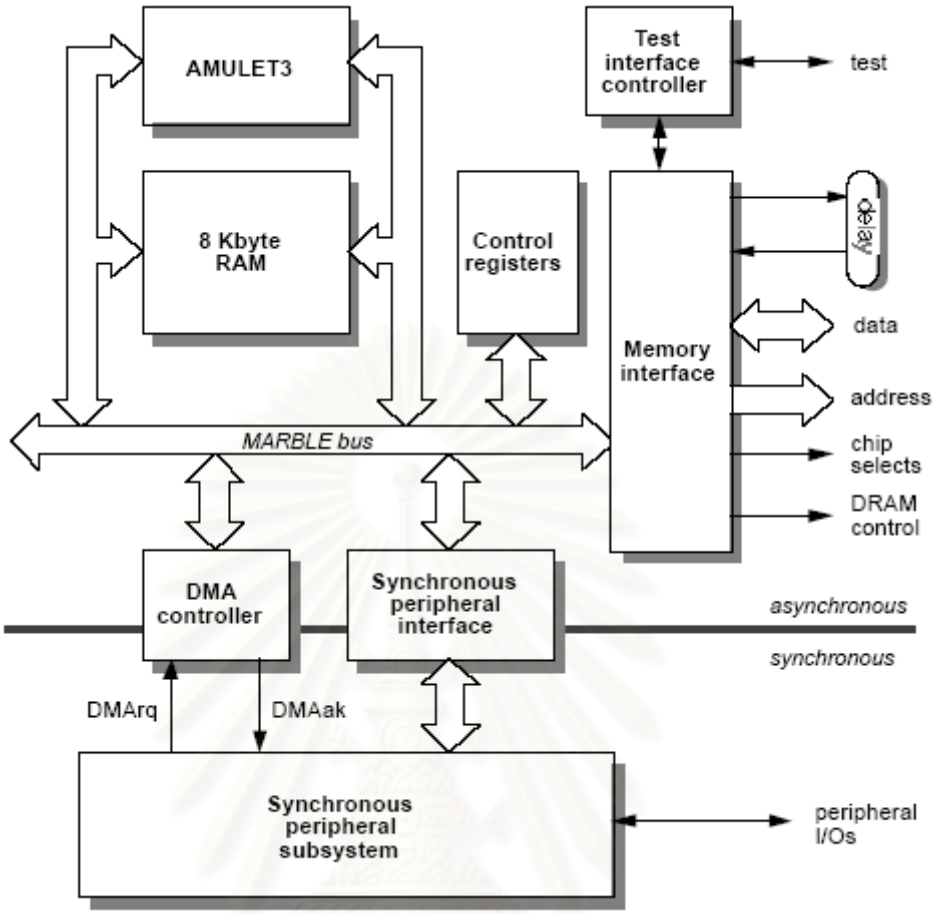
งานวิจัยนี้[5] นำเสนอการออกแบบบัสแบบอสมวาร โดยออกแบบวงจรประกอบด้วย Register ฝั่งอ่านข้อมูล (Read register) และ Register ฝั่งเขียนข้อมูล (Write register) ในส่วนของบัสออกแบบโดยใช้รหัสรางคู่ มีเกตผกผันต่อกับบัสเพื่อรักษาแรงดันในกรณีที่บัสไม่ได้ถูกเรียกใช้



รูปที่ 2.5 บัสแบบอสมวารขนาด 1 บิต

2.1.1.3 ไมโครโพรเซสเซอร์ AMULET 3i

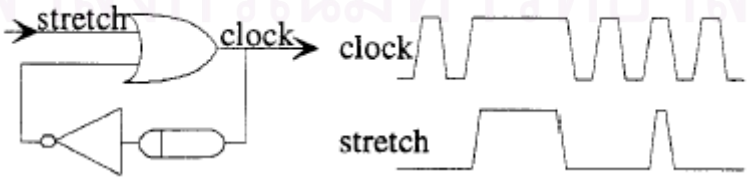
งานวิจัยที่นำมาอ้างอิงอีกงานหนึ่งเป็นงานวิจัยการออกแบบไมโครโพรเซสเซอร์ Amulet 3i [21] ซึ่งเป็นไมโครโพรเซสเซอร์แบบอสมวารที่มีการใช้บัสแบบอสมวาร เรียกบัสที่ใช้ใน Amulet 3i ว่า MARBLE ซึ่งเป็นบัสแบบอสมวารออกแบบโดยใช้พื้นฐานอ้างอิงจาก Micro-pipeline ของ Ivan Sutherland MARBLE [2] เป็นบัสแบบอสมวารที่ออกแบบโดยใช้สายสัญญาณแบบ Single-rail และโปรโตคอลอาณัติสัญญาณแบบ 2 ขั้น (NRZ) เป็นบัสแบบเดดิเคต(Dedicated) คือแยกระหว่างสัญญาณเลขที่อยู่(Address bus) กับสัญญาณข้อมูล (Data bus) ใช้ลอจิกสามสถานะในการขับสัญญาณ ใช้ Bridge ในการสลับการติดต่อระหว่างส่วนที่เป็นอสมวารกับสมวาร และมี ดีเอ็มเอเป็นส่วนเพิ่มเติมของบัส



รูปที่ 2.6 สถาปัตยกรรมของไมโครโพรเซสเซอร์ Amulet 3i

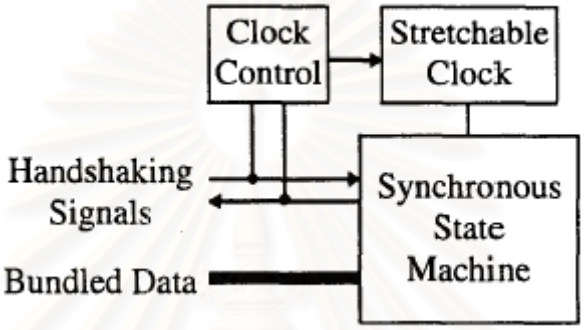
2.1.2 การเชื่อมต่อวงจรสมวารกับอสมวาร

งานวิจัยนี้[9] รวมวงจรที่เป็นอสมวารและสมวารให้สามารถรับส่งข้อมูลด้วยกันได้ โดยใช้สัญญาณนาฬิกาแบบยืดหยุ่น (Stretchable Clocks) ดังแสดงในรูปที่ 2.7



รูปที่ 2.7: แนวคิดการออกแบบสัญญาณนาฬิกาแบบยืดหยุ่น

จากรูปที่ 2.7 สัญญาณ Stretch เป็นสัญญาณควบคุมสัญญาณนาฬิกาให้เป็นไปตามความต้องการ โดยสัญญาณ stretch จะมีผลตอนขาขึ้นของสัญญาณนาฬิกาเท่านั้น สัญญาณนาฬิกาจะเปลี่ยนอีกทีก็ต่อเมื่อสัญญาณ stretch ลง เมื่อสัญญาณ stretch มีค่าน้อยกว่าความถี่ของสัญญาณนาฬิกาแล้ว สัญญาณนาฬิกายังคง sampling ที่ความถี่ปกติ ไม่ยืดหยุ่นไปตามสัญญาณ stretch



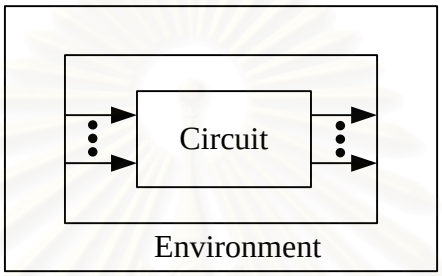
รูปที่ 2.8: การเชื่อมต่อผ่านสัญญาณนาฬิกาแบบยืดหยุ่น

2.2 การออกแบบวงจรอสมวาร

ในการทำความเข้าใจการออกแบบวงจรอสมวารนั้น สิ่งที่เป็นพื้นฐานในการออกแบบคือ ทำความเข้าใจแบบจำลองต่าง ๆ ของการออกแบบวงจรอสมวาร ประกอบด้วยแบบจำลองการทำงานสิ่งแวดล้อม แบบจำลองความหน่วงแบบต่าง ๆ ของเกตและสาย รวมถึงโหมดในการเลือกใช้งานของวงจรที่ออกแบบ แบบจำลองความหน่วงสามารถแบ่งออกเป็น 2 แบบใหญ่ ๆ คือ แบบจำลองความหน่วงที่กำหนดขอบเขตของความหน่วง และแบบจำลองความหน่วงที่ไม่กำหนดขอบเขตของความหน่วง การออกแบบวงจรอสมวารในปัจจุบัน ใช้การออกแบบวงจรโดยใช้แบบจำลองความหน่วงที่ไม่กำหนดขอบเขตของความหน่วงเป็นส่วนใหญ่ และสามารถออกแบบวงจรให้ทำงานได้ถูกต้องภายใต้แบบจำลองที่ตั้งสมมุติฐานไว้ แต่ก็ยังมีให้เห็นการออกแบบวงจรโดยใช้แบบจำลองความหน่วงที่กำหนดขอบเขตความหน่วงอยู่ จากนั้น ทำความเข้าใจโปรโตคอลอาณัติสัญญาณ การเข้ารหัสข้อมูล การส่งข้อมูลแบบต่าง ๆ เป็นต้น

2.2.1 **แบบจำลองการทำงานสิ่งแวดล้อม** (Environment Operation Model)

ระบบบอสมวารประกอบด้วยวงจร และสิ่งแวดล้อมที่ทำหน้าที่รับส่งอินพุต/เอาท์พุตกับวงจร โดยใช้แบบจำลองการทำงานสิ่งแวดล้อมเป็นตัวกำหนดว่าจะให้สิ่งแวดล้อมหรือวงจรทำหน้าที่ตรวจสอบการสิ้นสุดของการเปลี่ยนระดับสัญญาณในวงจร ด้วยการพิจารณาความหน่วงของสิ่งแวดล้อมเทียบกับความหน่วงของวงจร ซึ่งสามารถแบ่งได้เป็น 2 สภาวะคือ สภาวะแวดล้อมมูลฐาน และสภาวะแวดล้อมรับเข้าส่งออก



รูปที่ 2.9: ระบบบอสมวาร

2.2.1.1 **สภาวะแวดล้อมมูลฐาน**

เมื่อความหน่วงของสิ่งแวดล้อมมีค่ามากกว่าค่าความหน่วงของวงจร สามารถใช้ค่าความหน่วงของการทำงานของสิ่งแวดล้อมรับประกันการสิ้นสุดของการเปลี่ยนระดับสัญญาณในวงจรได้ การออกแบบจึงกำหนดให้สิ่งแวดล้อมทำหน้าที่ตรวจสอบการสิ้นสุดของการเปลี่ยนระดับสัญญาณของวงจร โดยสิ่งแวดล้อมจะรับเอาท์พุตมาจากวงจร และส่งอินพุตชุดใหม่ไปให้วงจร ก็ต่อเมื่อทุกการเปลี่ยนระดับสัญญาณภายในวงจรเสร็จสิ้นแล้ว

2.2.1.2 **สภาวะแวดล้อมรับเข้าส่งออก**

เมื่อความหน่วงของสิ่งแวดล้อมมีค่าน้อยกว่าค่าความหน่วงของวงจร การออกแบบจะกำหนดให้วงจรสามารถตรวจสอบการสิ้นสุดการเปลี่ยนระดับสัญญาณในวงจรได้เองและจะให้เอาท์พุตออกมาก็ต่อเมื่อทุกการเปลี่ยนระดับสัญญาณภายในวงจรเสร็จสิ้นแล้ว สภาวะแวดล้อมนี้เป็นที่นิยมใช้ในการออกแบบ เนื่องจากเมื่อมีความแปรปรวนความหน่วงเกิดขึ้นระบบพบว่าวงจรที่ออกแบบในสภาวะแวดล้อมรับเข้าส่งออกสามารถรับประกันความถูกต้องในการทำงานได้ดีกว่าวงจรที่ออกแบบในสภาวะแวดล้อมมูลฐาน

2.2.2 แบบจำลองความหน่วง (Delay Model)

เกตต่าง ๆ ที่นำมาสร้างเป็นวงจรรวม (Integrated Circuits) นั้น ล้วนแต่มีความหน่วงทั้งสิ้น ตามคุณสมบัติทางกายภาพของเกต ในขั้นตอนการสร้างวงจรก็อาจมีความหน่วงที่ไม่ทราบค่าเกิดขึ้นได้เช่นกัน อาจเป็นความหน่วงที่เกิดจากขั้นตอนการเจือสาร (Fabrication) ความหน่วงของอุณหภูมิ และความหน่วงจาก แรงดันที่ป้อนให้กับวงจรในระบบสมวาร ความหน่วงทางกายภาพของเกตจะไม่ถูกนำมาพิจารณาเนื่องจากวงจรทำงานโดยใช้การควบคุมจากสัญญาณนาฬิกา การเปลี่ยนแปลงสัญญาณก็จะเปลี่ยนไปตามตัวควบคุม ส่วนในระบบอสมวารซึ่งเป็นระบบที่ไม่มีสัญญาณนาฬิกาความหน่วงจึงเป็นปัญหาใหญ่ ดังนั้นเพื่อการจัดการกับความหน่วงที่เกิดขึ้นจึงได้มีแบบจำลองความหน่วงขึ้นมาเพื่อจัดการกับความหน่วงที่เกิดขึ้นในวงจรแล้วทำให้เกิดการทำงานของวงจรไม่มีความผิดพลาด

ความหน่วงมักจะถูกกำหนดโดยแบบจำลองเวลา ในแบบจำลองที่กำหนดค่าความหน่วงแน่นอน ความหน่วงก็จะกำหนดให้มีค่าที่แน่นอนอยู่ค่าหนึ่ง ในแบบจำลองที่ความหน่วงเป็นแบบมีขอบเขต ก็จะมีการกำหนดค่าความหน่วงไว้ให้อยู่ในขอบเขตที่กำหนดไว้ ส่วนในแบบจำลองความหน่วงที่ไม่มีขอบเขต ค่าความหน่วงที่กำหนดไม่ได้กำหนดไว้แต่ตั้งสมมุติฐานว่าความหน่วงจะต้องมีค่าอยู่ค่าหนึ่ง

แบบจำลองความหน่วงของวงจรสามารถแบ่งได้ ดังนี้

2.2.2.1 แบบจำลองความหน่วงที่ไม่ขึ้นกับความเร็ว (Speed Independent Model)

แบบจำลองความหน่วงที่ไม่ขึ้นกับความเร็วเป็นแบบจำลองความหน่วงที่ไม่มีการกำหนดความหน่วงในสายสัญญาณ (Zero-wire delays) แต่จะกำหนดความหน่วงให้กับเกตในวงจรเท่านั้น

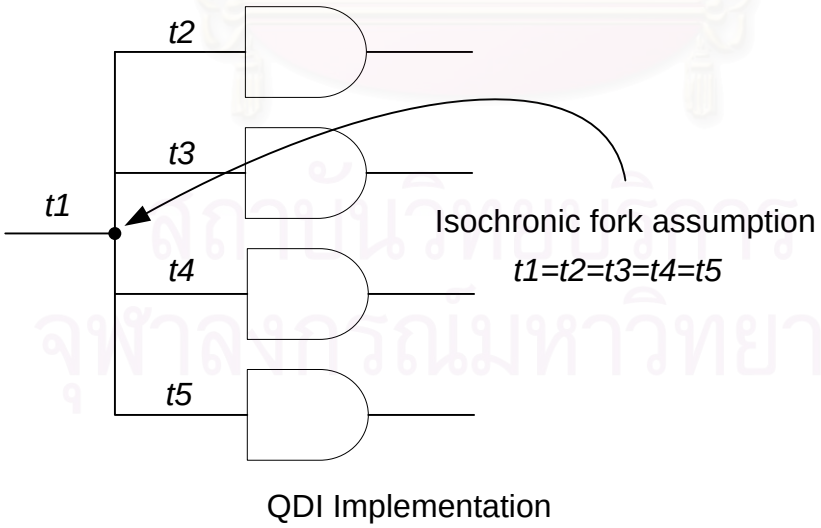
2.2.2.2 แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วง (Delay-Insensitive: DI)

แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วง [10] เป็นแบบจำลองความหน่วงที่ไม่กำหนดค่าความหน่วงเกตและค่าความหน่วงสายในการสร้างวงจรในระดับเลย์เอาต์ (Layout Circuit Implementation) แต่ทราบว่ามีค่าอยู่ในขอบเขตหนึ่งที่ไม่ใช่อนันต์ ดังนั้นเมื่อกำหนดให้วงจรมีการเปลี่ยนระดับสัญญาณ S1 เกิดก่อนการเปลี่ยนระดับสัญญาณ S2 การออกแบบวงจรจะต้องออกแบบให้สัญญาณ S1 เป็นอินพุตของเส้นทางส่งผ่านสัญญาณ (Signal

Propagation Path) ของสัญญาณ S2 เท่านั้น การออกแบบวงจรจึงสามารถใช้ได้เพียงเกตผกผัน และอุปกรณ์ชนิดซี (C-Element) เท่านั้น

2.2.2.3 **แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วงชนิดเสมือน** (Quasi-Delay-Insensitive : QDI)

แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วงชนิดเสมือน [7,8] เป็นแบบจำลองความหน่วงที่ไม่ไวต่อความหน่วงที่กำหนดให้ค่าความหน่วงในกิ่งของสาย (Fork Wire) ทุกกิ่งมีค่าเท่ากันดังรูปที่ 2.10 การออกแบบวงจรโดยใช้แบบจำลองความหน่วงนี้มีความซับซ้อนน้อยกว่าการใช้แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วง เนื่องจากไม่ต้องคำนึงถึงลำดับการเปลี่ยนระดับสัญญาณที่ทุกกิ่งของสาย โดยเมื่อกำหนดให้วงจรมีการเปลี่ยนระดับสัญญาณ t1 เกิดก่อนการเปลี่ยนระดับสัญญาณ t2 t3 t4 t5 การออกแบบวงจรสามารถใช้กิ่งของสายสัญญาณ t1 กิ่งใดกิ่งหนึ่งเป็นอินพุตของเส้นทางส่งผ่านสัญญาณของสัญญาณ t2 t3 t4 t5 อย่างไรก็ตามการสร้างวงจรโดยใช้แบบจำลองดังกล่าวนั้นไม่สามารถทำได้ เนื่องจากในความเป็นจริงแล้วแต่ละกิ่งของสายสัญญาณมีค่าความหน่วงไม่เท่ากัน เมื่อทำการสร้างวงจรตามที่ออกแบบ และกิ่งของสายสัญญาณ t1 ที่เลือกมีความหน่วงน้อยกว่ากิ่งอื่นๆ อาจทำให้สัญญาณ t2 t3 t4 t5 เกิดการเปลี่ยนแปลงก่อนสัญญาณ t1 ได้ ดังนั้นหากต้องการสร้างวงจรให้ใช้งานได้จริง ในการออกแบบจะต้องเลือกกิ่งของสายสัญญาณ t1 ทุกกิ่งเป็นอินพุตของเส้นทางส่งผ่านระดับสัญญาณ t2 t3 t4 t5



รูปที่ 2.10: แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วงชนิดเสมือน

2.2.3 **ช่องทางการรับส่งข้อมูล**

ในการออกแบบวงจรอสมวาร การรับส่งข้อมูลระหว่าง ส่วนต่าง ๆ ของวงจรนั้น ใช้สายสัญญาณจำนวนมาก หรือเรียกว่ากลุ่มของสายสัญญาณ กลุ่มของสายสัญญาณดังกล่าวเมื่อมีการส่งต่อไปยังส่วนใดส่วนหนึ่งในวงจร สามารถรวมกลุ่มสายนั้นเป็นกลุ่มเดียวกัน และเรียกการรับส่งข้อมูลชุดนั้นว่า ช่องทางการรับส่งข้อมูล ช่องทางการรับส่งข้อมูล โดยปกติแล้วจะเป็นการรับส่งข้อมูลแบบทางเดียว และมีการรับส่งแบบจุดต่อจุด โดยข้อมูลจะมีการกำหนดการรับส่งผ่านช่องทางการรับส่งข้อมูลดังนี้

- ฝั่งส่ง คือ ส่วนที่ส่งข้อมมูลเข้าไปในช่องทางการรับส่งข้อมูล
- ฝั่งรับ คือ ส่วนที่รับข้อมูลจากช่องทางการรับส่งข้อมูล

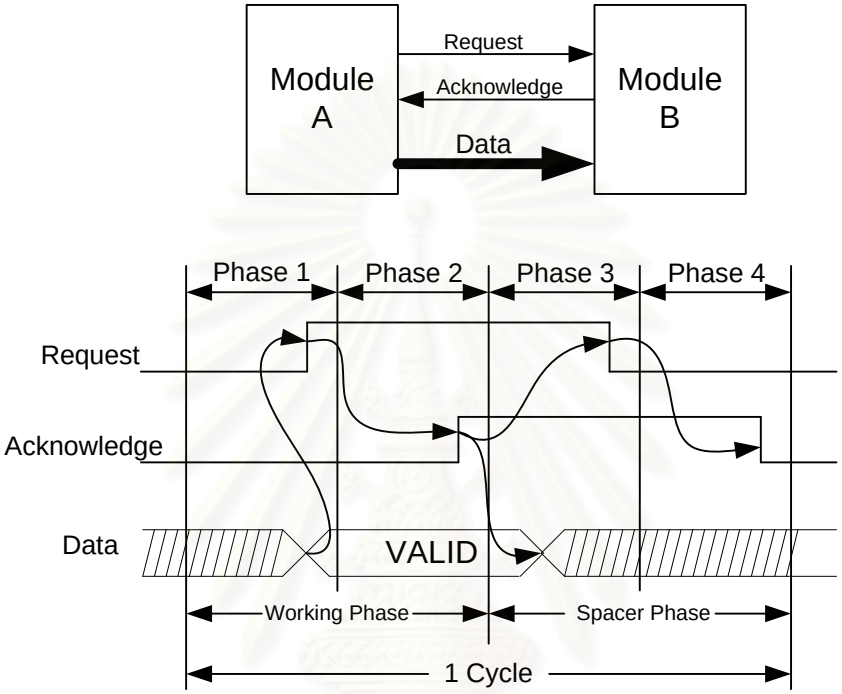
จากแนวคิดการรับส่งข้อมูลผ่านช่องทางการรับส่งข้อมูล สามารถเกิดการรับส่งดังนี้ คือ

- ผู้เริ่มต้น คือส่วนที่ทำให้เกิดการรับส่ง
- ผู้รับ คือส่วนที่ตอบรับจากส่วนเริ่มต้น

2.2.4 **โปรโตคอลอาณัติสัญญาณ** (Signaling Protocol)

วงจรอสมวารการเปลี่ยนสัญญาณจะมีโปรโตคอลอาณัติสัญญาณเป็นข้อกำหนดในการเปลี่ยนสัญญาณแทนสัญญาณนาฬิกา โดยมีสัญญาณ ร้องขอเป็นสัญญาณแสดงการเริ่มต้นการทำงานของวงจรอสมวาร และมีสัญญาณตอบรับเป็นสัญญาณแสดงความบริบูรณ์ของการทำงานหรืออาจเรียกว่าสัญญาณบอกถึงการเสร็จสิ้นการทำงานของวงจร [2] สัญญาณควบคุมจะดำเนินการตามลำดับการทำงานในวงจร หรือเรียก ว่าสัญญาณ แฮนด์เชค (Handshake) โดยสายสัญญาณดังกล่าวมีการทำงานเป็นอิสระไม่มีสัญญาณจากภายนอกควบคุม การควบคุมเป็นลักษณะของการเชื่อมต่อกันของวงจรสามารถควบคุมอย่างอิสระไม่ขึ้นกับส่วนอื่น ๆ ของวงจรที่ไม่ได้ทำงาน โมเดลการทำงานของวงจรจะคล้ายกับการทำงานของโมเดลการไหลข้อมูล สัญญาณที่สั่งให้วงจรทำงานมาจากผลลัพธ์ที่ได้จากวงจรส่วนอื่น ๆ ส่งมาให้ อาจมองในลักษณะการส่งข้อมูลระหว่าง ผู้ส่งกับผู้รับ สมมติให้ A เป็นผู้ส่งข้อมูล B เป็นผู้รับข้อมูล เมื่อ A ต้องการส่งข้อมูล A ก็จะส่งสัญญาณ ร้องขอไปให้ B เพื่อต้องการข้อมูลจาก B เมื่อ B ได้รับสัญญาณร้องขอแล้ว B ก็จะตอบรับสัญญาณร้องขอโดยการส่งสัญญาณ ตอบรับไปให้ A โปรโตคอลอาณัติสัญญาณจะให้การเปลี่ยนสัญญาณร้องขอและสัญญาณ ตอบรับในการตอบสนองต่อเหตุการณ์ต่าง ๆ ที่เกิดขึ้นในวงจร [8]

โปรโตคอลอาณัติสัญญาณที่นิยมใช้ส่วนใหญ่แบ่งออกได้เป็น 2 แบบ ด้วยกัน คือ โปรโตคอลอาณัติสัญญาณแบบ 2 ขั้น และโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น เนื่องจากงานวิจัยนี้ใช้โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น จึงขออธิบายโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้นเท่านั้น



รูปที่ 2.11 โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น

โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น หรืออาจเรียก รีเทิร์นทูซีโร (RZ) 4-เฟส และเลเวลซิกแนลลิ่ง (Level Signaling) จากรูปที่ 2.11 จะเห็นการไหลของสัญญาณในรูปของคาบเวลา ซึ่งในการทำงานจริงของวงจรแล้วอาจจะไม่เป็นไปตามรูปก็ได้ ลูกศร โค้งสีดำ แสดงลำดับการเปลี่ยน ก่อน / หลัง ของสัญญาณ ในที่นี้ไม่ได้กำหนดความหน่วงของการแลกเปลี่ยนข้อมูลกันระหว่างผู้ส่งกับผู้รับ โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น ประกอบด้วย การเปลี่ยนสัญญาณ 4 ครั้งด้วยกัน (2 ครั้งสำหรับสัญญาณ Request และ 2 ครั้งสำหรับสัญญาณ Acknowledge) ในการทำงาน 1 ไซเคิล

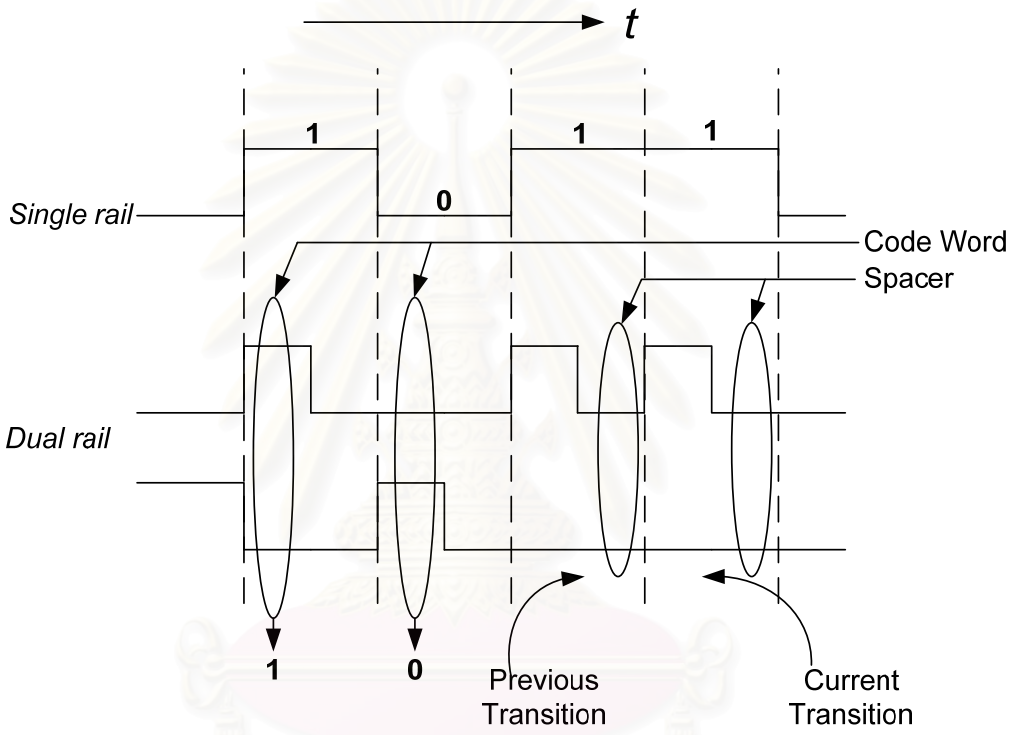
2.2.5 **รหัสรางคู่** (Dual-rail code)

การใช้สัญญาณ 1 หรือ 0 จ่ายให้วงจรจะไม่เกิดปัญหาในกรณีที่ การเปลี่ยนสัญญาณเปลี่ยนจาก 1 ไป 0 หรือเกิดการเปลี่ยนสัญญาณ จาก 0 ไป 1 แต่จะมีปัญหาในขณะที่สัญญาณเปลี่ยนจาก 0 ไป 0 หรือ 1 ไป 1 วงจรควบคุมไม่สามารถตรวจสอบการมาถึงของสัญญาณ 1 ก่อนหน้า กับสัญญาณ 1 ที่เพิ่งจะมาถึง (ในกรณีที่เกิดการเปลี่ยนสัญญาณจาก 1 ไป 1) ดังนั้น ในการออกแบบวงจรอสมวารจึงใช้วิธีการเข้ารหัสข้อมูลเพื่อให้การเปลี่ยนสัญญาณเป็นไปอย่างถูกต้อง สามารถตรวจสอบได้ วิธีการเข้ารหัสวงจรอสมวารที่นิยมใช้ ได้แก่ การเข้ารหัสโดยใช้ข้อมูลรวมชุด กับ การเข้ารหัสโดยใช้รหัสรางคู่ โดยการเข้ารหัสทั้ง 2 วิธีใช้โปรโตคอลอาณัติสัญญาณแบบ 2 หรือ 4 ขั้น ตามที่ผู้ออกแบบได้ออกแบบไว้ ในกรณีที่ใช้การเข้ารหัสโดยใช้ข้อมูลรวมชุด (Bundle Data) นั้น สำหรับข้อมูลที่มีขนาด n บิต ข้อมูลที่ถูกส่งจากผู้ส่งไปยังผู้รับนั้นมีค่าเท่ากับ n+2 (ข้อมูลเท่ากับ n บิต, สัญญาณ Request 1 บิต, สัญญาณ Acknowledge 1 บิต) ในการส่งข้อมูลแบบรวมชุด สายสัญญาณที่ส่งข้อมูลมีความเร็วมากกว่าสายสัญญาณควบคุม ในขั้นตอนการทำงานฝั่งส่งข้อมูลจะส่งสัญญาณที่เป็นข้อมูลไปก่อน เมื่อฝั่งรับได้รับข้อมูลแล้วฝั่งส่งจึงส่งสัญญาณ Request ไปทีหลัง จากนั้นฝั่งรับจะส่งสัญญาณ Acknowledge กลับมาเพื่อบอกว่าตัวเองได้รับข้อมูลเรียบร้อยแล้ว วงจรอสมวารจำนวนมากที่ออกแบบโดยใช้ข้อมูลรวมชุด เนื่องจากว่าการออกแบบโดยการใช้ข้อมูลรวมชุดนั้นใช้สายสัญญาณน้อยกว่าวิธีอื่น ๆ ส่วนการเข้ารหัสอีกแบบคือ การเข้ารหัสโดยใช้รหัสรางคู่ (Dual-rail code) ในการเข้ารหัส โดยรหัสรางคู่จะใช้สายสัญญาณ 2 เส้น ในการรับส่งสัญญาณ 1 บิต โดยการส่งจะส่งรวมกันระหว่างสายควบคุมกับสายข้อมูล การเข้ารหัสโดยใช้รหัสรางคู่ ประกอบด้วยสถานะทั้งสิ้น 4 สถานะ ดังต่อไปนี้

1. 00 - ตัวแบ่งรอบการทำงาน (spacer)
2. 10 - เท่ากับ 1
3. 01 - เท่ากับ 0
4. 11 - ไม่ใช้

ในกรณีนี้ สำหรับการส่งข้อมูลจำนวน n บิต สายที่เชื่อมต่อระหว่าง ผู้ส่งและผู้รับ เท่ากับ 3n ประกอบด้วย 2 เส้นสำหรับแต่ละบิตซึ่งจะรวมสัญญาณ Request ไปในตัว และสัญญาณ Acknowledge 1 เส้นสำหรับแต่ละบิต แต่การรับส่งข้อมูลส่วนใหญ่อยู่ในลักษณะ ไบต์ หรือ

เวิร์ด ดังนั้นจึงสามารถลดสาย Acknowledge ให้เหลือเส้นเดียวได้ ทำให้วงจรมีขนาดเล็กลง เท่ากับ 2n+1 การใช้รหัสรางคู่กับโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้นนั้น แบ่ง ออกเป็น 2 ขั้น คือ ขั้นทำงาน และขั้นว่าง ในขั้นทำงาน ซึ่งเป็นวิธีที่ใช้ในงานวิจัยนี้ด้วย ผู้ส่งจะส่งสัญญาณ 0 หรือ 1 โดยการส่ง 01 หรือ 10 ตามลำดับ จากนั้นเมื่อผู้รับได้รับข้อมูลเรียบร้อยแล้ว ผู้ส่งจะต้องส่ง สัญญาณ 00 เพื่อแบ่งรอบการทำงานของวงจร จึงถือว่าเป็นการเสร็จสิ้นการรับส่งข้อมูล ส่วนการใช้รหัสรางคู่กับ โปรโตคอลอาณัติสัญญาณแบบ 2 ขั้นนั้น การรับส่งข้อมูลจะเสร็จสมบูรณ์ในขั้นตอนเดียว



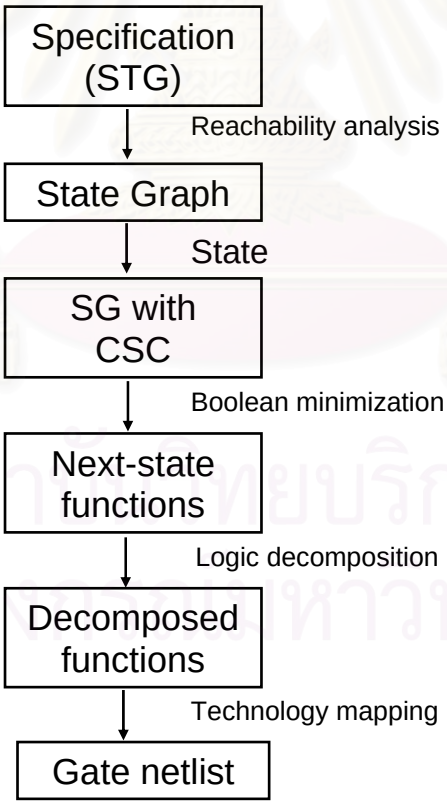
รูปที่ 2.12 การเข้ารหัสโดยใช้รหัสรางคู่

2.2.6 อุปกรณ์ชนิดซี

อุปกรณ์ชนิดซี เป็นอุปกรณ์ที่ถูกใช้อย่างแพร่หลายในการออกแบบวงจรอสมวาร เป็นอุปกรณ์ประเภทจำสถานะ (State Holding) โดยการทำงานของอุปกรณ์ชนิดซีจะให้เอาท์พุตเป็น 1 เมื่อทุกอินพุตเป็น 1 และให้เอาท์พุตเป็น 0 เมื่อทุกอินพุตเป็น 0 นอกจากนั้นอุปกรณ์ชนิดซี จะจำสถานะเดิมเอาไว้ (Hold Previous State)

2.2.7 การออกแบบวงจรอสมวารโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ (STG)

กราฟบรรยายการเปลี่ยนสัญญาณ (Signal transition graph) เป็นวิธีการบรรยายพฤติกรรมการทำงานของวงจรที่ออกแบบโดยอาศัยกราฟ ประกอบด้วยสัญญาณ 3 ชนิดด้วยกัน คือ สัญญาณอินพุต สัญญาณเอาท์พุต และสัญญาณภายใน สัญญาณอินพุตแสดงโดยการขีดเส้นใต้หรือทำเป็นตัวหนาที่ชื่อของสัญญาณ ในการออกแบบวงจรโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ สัญญาณขาขึ้นจะแทนด้วยสัญลักษณ์ '+' และสัญญาณขาลงจะแทนด้วยสัญลักษณ์ '-' สำหรับทุกการออกแบบกราฟบรรยายการเปลี่ยนสัญญาณ จะเรียกว่าซิงเกิลไซเคิล เมื่อเกิดการเปลี่ยนแปลงสัญญาณ ขาขึ้นและขาลงเพียงครั้งเดียว นอกจากนั้นจะเรียกว่า มัลติไซเคิล การออกแบบวงจรโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณต้องออกแบบให้เป็นไปตามกฎ ดังจะอธิบายต่อไป รูปที่ 2.13 แสดงขั้นตอนการออกแบบวงจรโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ



รูปที่ 2.13: ขั้นตอนการออกแบบวงจรโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ

2.2.7.1 กฎการออกแบบโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ

(a) - Input free – choice: only Mutex inputs may control the choice.

(b) - 1- Bounded – สามารถมี token สูงสุด เท่ากับ 1 token ต่อ 1 เฟส

(c) - Liveness – ไม่เกิดปรากฏการ deadlock

กราฟบรรยายการเปลี่ยนสัญญาณ สำหรับการออกแบบวงจรที่ไม่ขึ้นกับความเร็ว:

(a) – มีการกำหนดสัญญาณแบบคงเส้นคงวา – สัญญาณขาขึ้นและสัญญาณขาลง มีการเปลี่ยนอย่างคงเส้นคงวา

(b) – มีความสัมพันธ์เกี่ยวเนื่องกัน

สำหรับการสังเคราะห์วงจร

(a) – การกำหนดสถานะโดยสมบูรณ์ – ไม่เกิดการซ้ำซ้อนกันในการกำหนดสถานะ

2.3 บัส (BUSES)

บัส (Bus) คือเส้นทางการเชื่อมต่อของอุปกรณ์ต่าง ๆ ในคอมพิวเตอร์ ซึ่งจะอนุญาตให้อุปกรณ์ต่าง ๆ ใช้ในลักษณะร่วมกันใช้ (Share) ดังนั้นการออกแบบระบบบัสในคอมพิวเตอร์นั้นต้องมีการจัดลำดับในการเข้าถึง เพื่อให้มีอุปกรณ์เพียงตัวเดียวเท่านั้น ณ เวลาหนึ่งที่สามารถใช้บัสได้[4]

โดยทั่วไปแล้วบัสจะมีการรับส่งข้อมูลสองทิศทาง (Bi-directional) บัสประกอบด้วยสายสัญญาณเพื่อการรับส่ง ค่าไบนารี 0 และ 1 ดังนี้ สายข้อมูล (Data bus) สายเลขที่อยู่ (Address bus) และสายควบคุม (Control line)

2.3.1 โครงสร้างบัส

ระบบบัสของคอมพิวเตอร์ส่วนใหญ่ประกอบด้วยสายสัญญาณประมาณ 50 ถึง 100 เส้น แต่ละเส้นก็จะมีการทำงานตามฟังก์ชันที่ได้ออกแบบไว้ อย่างไรก็ตามระบบบัสที่ถูกออกแบบส่วนใหญ่ แบ่งสายไว้เป็น 3 กลุ่มด้วยกัน คือ สายข้อมูล สายเลขที่อยู่ และสายควบคุม

สายข้อมูล(Data bus) คือสายที่ทำการรับส่งข้อมูลจากจุดหนึ่งไปยังจุดหนึ่งภายในบัส หรือเรียกสายข้อมูลอีกอย่างหนึ่งว่าบัสข้อมูล บัสข้อมูลในคอมพิวเตอร์ประกอบด้วย

สายทั้งสิ้นประมาณ 32 ถึง 100 กว่าเส้น จำนวนเส้นของสายข้อมูลจะบอกถึงความสามารถในการรับส่งของบัสหรือ ความกว้างของบัสเนื่องจากว่าสายแต่ละเส้นนั้นสามารถเป็นเส้นทางรับส่งข้อมูลได้เพียง 1 บิตเท่านั้นต่อเวลา จำนวนสายของสายข้อมูลจะบอกถึงความสามารถของบัสว่าสามารถรับส่งข้อมูลได้ทีละเท่าไร ยกตัวอย่างเช่น บัสมีความกว้างเท่ากับ 8 บิต และคำสั่งข้อมูลมีขนาดเท่ากับ 16 บิต หมายความว่าแต่ละคำสั่งข้อมูลจะต้องทำการเข้าถึงบัส 2 ครั้งด้วยกัน จึงจะได้การทำงานที่ 16 บิตคำสั่ง [10]

สายเลขที่อยู่(Address bus) เป็นสายที่บอกถึงตำแหน่งของต้นทางหรือปลายทางที่ข้อมูลจะถูกนำไปเก็บ ยกตัวอย่างเช่น เมื่อโพรเซสเซอร์ต้องการอ่านข้อมูลจำนวน 1 เวิร์ด (1 เวิร์ดอาจจะมีขนาดเท่ากับ 8 16 หรือ 32 บิต) จากหน่วยความจำ โพรเซสเซอร์จะนำค่าของสายเลขที่อยู่ป้อนลงสู่บัสเพื่อเป็นการบอกกับส่วนควบคุมของหน่วยความจำว่าต้องการข้อมูลที่ตำแหน่งใด ความกว้างของสายเลขที่อยู่นั้นจะถูกกำหนดไว้ด้วยตำแหน่งสูงสุดของหน่วยความจำที่สามารถเข้าถึงได้ ยิ่งไปกว่านั้น สายเลขที่อยู่จะใช้เพื่อกำหนดตำแหน่งของพอร์ท ในส่วนของอุปกรณ์อินพุต/เอาท์พุต ด้วย โดยทั่วไปแล้วสายเลขที่อยู่จะแบ่งเป็น 2 ส่วนด้วยกัน ส่วนบนใช้ในการเลือกว่าจะให้อุปกรณ์ตัวไหนได้ใช้บัส ส่วนล่างจะเป็นตัวบอกการเลือกตำแหน่งของข้อมูลหรือ พอร์ตอินพุต/เอาท์พุตของอุปกรณ์ที่ถูกเลือกโดยส่วนบนของสายเลขที่อยู่ ดังตัวอย่างเช่น สายเลขที่อยู่มีความกว้างขนาด 8 บิต เลขที่อยู่ของ 01111111 ลงมาถึง 00000000 เป็นตัวบอกการเลือกตำแหน่งของข้อมูล ส่วนเลขที่ที่อยู่ 10000000 ขึ้นไปถึง 11111111 เป็นตัวบอกการเลือกอุปกรณ์

สายควบคุม (Control bus) เป็นสายที่ทำการควบคุมการใช้สายข้อมูล กับสายเลขที่อยู่ โดยทั่วไปแล้วสายควบคุมจะมีหน้าที่ควบคุมดังต่อไปนี้

การเขียนข้อมูลลงหน่วยความจำ (Memory Write) : เป็นการเขียนข้อมูลลงในหน่วยความจำจากบัสตามตำแหน่งที่อยู่ที่ได้จากสายที่เลขที่อยู่

การอ่านข้อมูลจากหน่วยความจำ (Memory Read) : เป็นการนำข้อมูลจากหน่วยความจำจากตำแหน่งที่อยู่ที่ส่งมาร้องขอแล้วนำข้อมูลเข้าไปยังบัส

การเขียนอินพุท/เอาท์พุต (I/O Write): เป็นการนำข้อมูลในบัสเขียนลงตำแหน่งของไอโอพอร์ท

การอ่านอินพุท/เอาท์พุต (I/O Read): เป็นการอ่านข้อมูลจากตำแหน่งที่อยู่ของไอโอพอร์ทแล้วนำข้อมูลที่อ่านได้ไปไว้ในบัส

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

สัญญาณร้องขอ(Request): เป็นสัญญาณที่อุปกรณ์แต่ละตัวใช้ในการร้องขอเพื่อใช้บัสในการรับส่งข้อมูล

การอนุญาตให้ใช้บัส(Bus Grant): เป็นผลมาจากการร้องขอ ควบคุมโดยตัวตัดสินใจให้อนุญาตให้ใช้บัส

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

การตอบรับอินเตอร์รัพท์(Interrupt Acknowledge): เป็นการตอบรับจากสัญญาณร้องขออินเตอร์รัพท์

สัญญาณนาฬิกา(Clock): ใช้ในการควบคุมลำดับการทำงานของวงจร (ในกรณีของวงจรแบบอสมวารจะไม่ใช้สัญญาณนาฬิกาในการควบคุม)

รีเซ็ต(Reset): เป็นการเริ่มต้นค่าใหม่

2.3.2 **การออกแบบบัส**

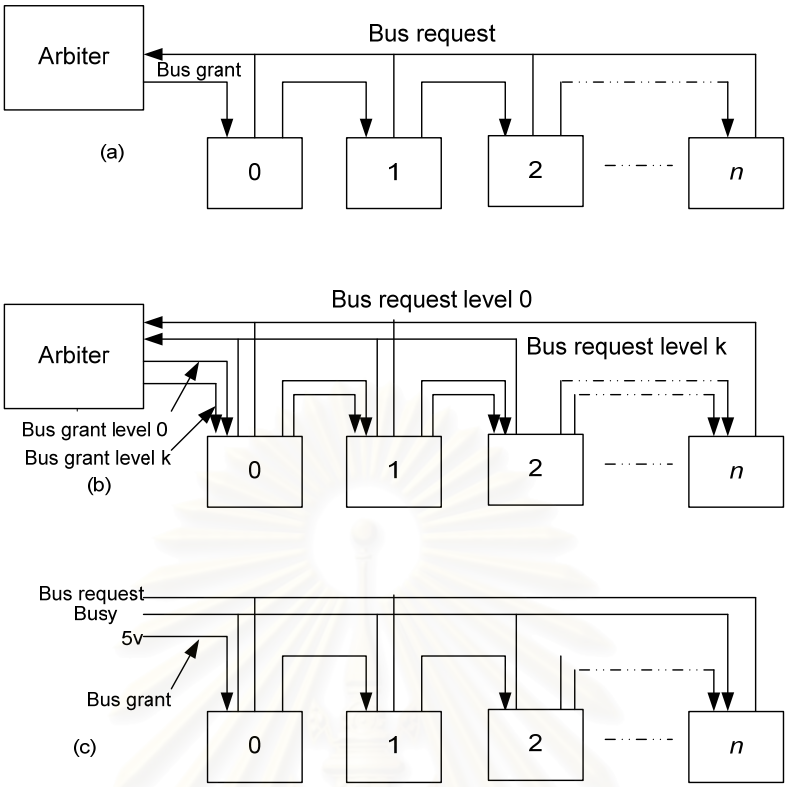
2.3.2.1 **ชนิดของบัส** (Bus Type)

สามารถแบ่งออกเป็น 2 ชนิดด้วยกัน เดดิเคต (Dedicated) และ มัลติเพล็กส์ (Multiplex) เดดิเคตบัสเป็นบัสที่ออกแบบฟังก์ชันการทำงานตายตัวมีสายเลขที่อยู่และสายสัญญาณแยกกันอย่างชัดเจน ส่วนบัสแบบ มัลติเพล็กส์ เป็นการออกแบบบัสแบบใช้สายร่วมกันระหว่างสายเลขที่อยู่และสายข้อมูล

ประโยชน์ของการออกแบบบัสแบบมัลติเพล็กส์นั้นจะใช้สายสัญญาณน้อยกว่าแบบเดดิเคต ส่วนข้อเสียของการออกแบบบัสแบบมัลติเพล็กส์นั้น การออกแบบทำได้ยากกว่า การทำงานจะต้องใช้เวลาในการส่งสายสัญญาณเลขที่อยู่กับสายสัญญาณข้อมูลแยกกัน

2.3.2.2 **ตัวตัดสินใจ** (Arbiter)

เป็นตัวควบคุมให้การใช้บัสของอุปกรณ์ต่าง ๆ สามารถแบ่งออกได้เป็น 2 แบบด้วยกัน คือ แบบศูนย์กลาง (Centralized) และแบบกระจาย (Distributed) ดังแสดงในรูปที่ 2.14

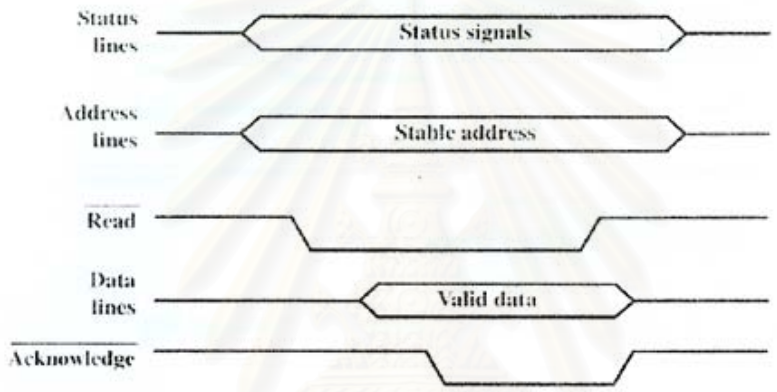


รูปที่ 2.14 การออกแบบตัวตัดสินใจแบบต่าง ๆ

จากรูป (a) แสดงการทำงานของตัวตัดสินใจแบบศูนย์กลาง (Centralized Arbiter) กล่าวคือ เมื่อมีอุปกรณ์หลายตัวต้องการใช้บัส อุปกรณ์แต่ละตัวก็จะส่งสัญญาณร้องขอ (Request)ไปยังตัวตัดสินใจ เมื่อตัวตัดสินใจได้รับสัญญาณร้องขอตัวตัดสินใจก็จะพิจารณาลำดับความสำคัญของอุปกรณ์ต่างๆ ที่ร้องขอเข้ามาแล้วอนุญาตกับอุปกรณ์ที่มีลำดับความสำคัญมากกว่าจากรูปจะเป็นอุปกรณ์ที่ใกล้กับตัวตัดสินใจมากที่สุด ส่วนในรูป (b) เป็นตัวตัดสินใจแบบศูนย์กลางเช่นกัน หากแต่เพิ่มการใช้งานเป็นระดับเข้าไปใช้ในกรณีที่มีอุปกรณ์ต่อพ่วงกับบัสจำนวนมาก ระดับที่มีตัวเลขน้อย ๆ มีลำดับความสำคัญมากกว่าลำดับที่มีตัวเลขมาก ๆ รูป (c) เป็นการออกแบบบัสแบบกระจาย (Distributed) การออกแบบบัสแบบนี้จะไม่มีตัวตัดสินใจร่วม อุปกรณ์ที่ต้องการใช้บัส จะต้องส่งสัญญาณร้องขอไปยังสายร้องขอ เพื่อตรวจสอบว่าบัสว่างอยู่หรือเปล่า หากว่าบัสว่างอุปกรณ์ดังกล่าวก็จะได้ใช้บัส เมื่ออุปกรณ์นั้นได้ใช้บัสแล้ว ก็จะส่งสัญญาณ ไม่ว่างไปยังสายสัญญาณไม่ว่างในบัส เพื่อบอกว่าตอนนี้บัสไม่ว่างและปล่อยสัญญาณร้องขอของตัวเอง อีกกรณีหนึ่งหากอุปกรณ์ทำการร้องขอแล้วปรากฏว่า บัสไม่ว่างก็จะเข้าสู่ขั้นตอนการอินเตอร์รัพท์ เพื่อรอเวลาบัสว่างอีกครั้ง [3]

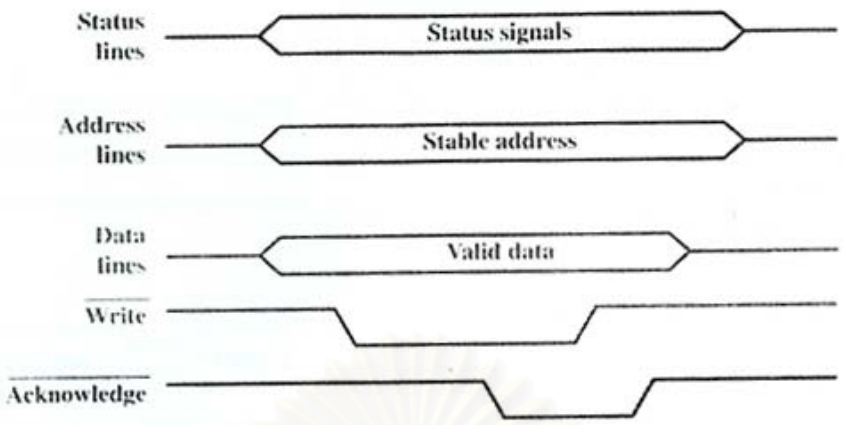
2.3.2.3 การทำงานของบัส

จังหวะเวลาการทำงานของบัสแบ่งได้เป็น 2 แบบด้วยกัน คือ แบบสมวาร และแบบอสมวารในที่นี้ขออธิบายแบบอสมวารเท่านั้น จังหวะเวลาการทำงานของระบบบัสแบบอสมวารสามารถอธิบายขั้นตอนการทำงานในกรณีอ่านข้อมูลและเขียนข้อมูลได้ ดังรูป



รูปที่ 2.15 รอบการอ่านข้อมูลของบัสแบบอสมวาร

จากรูปที่ 2.15 โพรเซสเซอร์จะสั่งให้สัญญาณเลขที่อยู่ และสัญญาณแสดงสถานะ (หรืออาจใช้สัญญาณร้องขอ) ป้อนเข้าบัส จากนั้นก็จะรอให้สัญญาณเลขที่อยู่ที่ป้อนเข้าบัสเสถียรแล้ว จึงป้อนสัญญาณอ่าน เพื่อสั่งให้บัสทำการอ่านข้อมูล ก่อนที่สัญญาณอ่านจะถูกป้อนนั้นวงจรจะต้องแน่ใจก่อนว่าสัญญาณควบคุมกับสัญญาณเลขที่อยู่เสถียรแล้ว จากนั้นเมื่อหน่วยความจำได้รับสัญญาณเลขที่อยู่และสัญญาณอ่านแล้วจะถอดรหัสสัญญาณที่อยู่แล้วจึงนำข้อมูลในตำแหน่งสัญญาณที่อยู่ที่ได้รับป้อนเข้าสู่สัญญาณข้อมูลในบัส เมื่อข้อมูลในบัสเสถียรแล้ว ตัวควบคุมบัสก็จะส่งสัญญาณตอบรับเพื่อบอกว่าบัสได้รับข้อมูลแล้ว จากนั้นเมื่ออุปกรณ์ที่ร้องขอบัสได้อ่านข้อมูลออกไปจากบัสแล้วอุปกรณ์นั้นก็จะปล่อยสัญญาณอ่านที่ร้องขอให้ขั้นตอนแรก เป็นผลให้หน่วยความจำหยุดส่งข้อมูลและสัญญาณตอบรับ สุดท้ายสัญญาณตอบรับก็จะถูกปล่อย อุปกรณ์ที่ร้องขอในขั้นตอนแรกก็จะปล่อยสัญญาณทั้งหมด

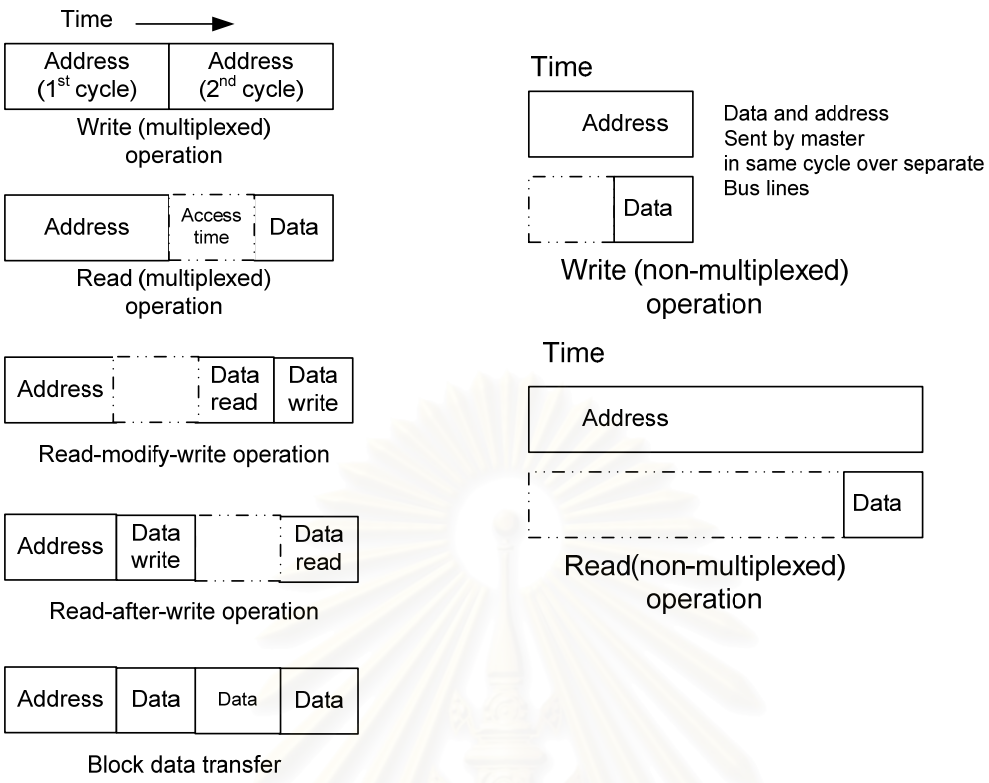


รูปที่ 2.16 รอบการเขียนข้อมูลของบัสแบบอสมวาร

จากรูปที่ 2.16 อุปกรณ์ที่ต้องการเขียนข้อมูลจะป้อนข้อมูลลงในบัส ในขณะเดียวกันก็จะป้อนสัญญาณแสดงสถานะและสัญญาณที่อยู่ไปด้วย จากนั้นบัสจะส่งสัญญาณคำสั่งให้เขียนข้อมูลให้กับหน่วยความจำ เมื่อหน่วยความจำได้รับสัญญาณเขียนข้อมูลแล้ว ก็จะเริ่มทำการเขียนข้อมูลลงในตำแหน่งของสายสัญญาณเลขที่อยู่ได้รับ เมื่อเขียนเสร็จแล้วก็จะส่งสัญญาณตอบรับกลับไปยังบัส จากนั้นอุปกรณ์ที่ร้องขอตอนแรกก็จะปล่อยสัญญาณเขียนหน่วยความจำก็จะปล่อยสัญญาณตอบรับ [10]

2.3.2.4 การแลกเปลี่ยนข้อมูลของบัส

บัสมีวิธีการส่งผ่านข้อมูลได้หลายวิธี ดังรูปที่ 2.17 ในกรณีของการออกแบบบัสแบบมัลติเพล็กส์ ประกอบด้วยสัญญาณเลขที่อยู่และสัญญาณข้อมูล ขั้นแรกบัสจะใช้ในการส่งสัญญาณเลขที่อยู่ จากนั้นก็จะส่งสัญญาณข้อมูล สำหรับการอ่านข้อมูลในบัสปกติแล้วจะต้องรอให้สัญญาณจากตัวส่งป้อนลงก่อนจึงจะอ่านได้ ในกรณีของการออกแบบบัสแบบเดดิเคต สัญญาณเลขที่อยู่และสัญญาณข้อมูลถูกป้อนแบบขนาน สำหรับการเขียนข้อมูล ผู้ส่งจะป้อนข้อมูลเข้าบัสเมื่อสัญญาณเลขที่อยู่นั้นเสถียรแล้ว สำหรับการอ่าน ข้อมูลจากหน่วยความจำจะถูกป้อนสู่บัสเมื่อได้รับสัญญาณเลขที่อยู่แล้ว



รูปที่ 2.17 วิธีการส่งข้อมูลของบัสแบบต่าง ๆ

2.4 **อินเตอร์รัพท์** (Interrupt)

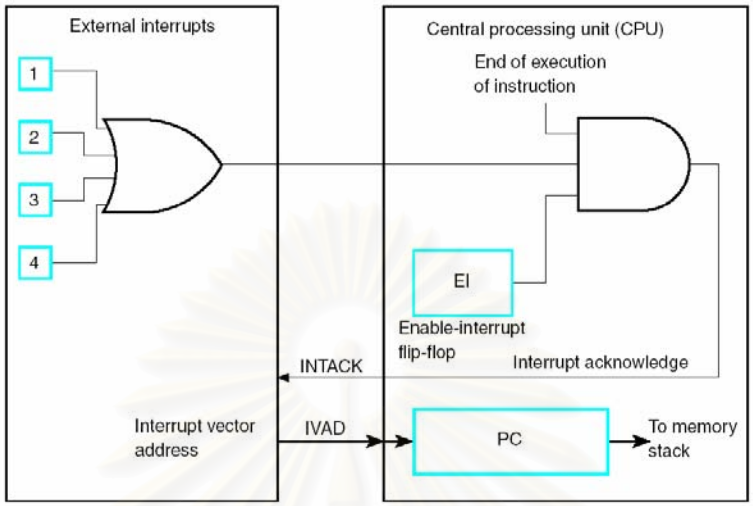
อุปกรณ์อินพุท/เอาต์พุต นั้นทำงานช้ากว่าซีพียูและหน่วยความจำมาก ดังนั้น อินเตอร์รัพท์จึงถูกนำมาใช้ในการติดต่อระหว่างอุปกรณ์อินพุต/เอาท์พุตและซีพียู เพื่อให้ได้ประสิทธิภาพที่ดีที่สุด

อินเตอร์รัพท์ คือการขัดจังหวะการทำงานของซีพียูในขณะกำลังทำงานเพื่อรับส่งข้อมูลกับอุปกรณ์อินพุต/เอาท์พุต ทำให้ซีพียูไม่ต้องเสียเวลามาตรวจสอบว่ามีอุปกรณ์ใด ที่ต้องการรับส่งข้อมูล ณ เวลาใด ๆ

ชนิดของอินเตอร์รัพท์

1. External interrupt ใช้โดยซีพียูเพื่อปฏิสัมพันธ์กับอุปกรณ์อินพุต/เอาท์พุต ซึ่งซีพียูจะเป็นผู้ร้องข้อมูลแก่อุปกรณ์อินพุต/เอาท์พุต เมื่ออุปกรณ์อินพุต/เอาท์พุตพร้อมส่งข้อมูลแล้วจึงส่งสัญญาณอินเตอร์รัพท์ไปให้ซีพียู
2. Internal interrupt เกิดขึ้นภายในซีพียู อาจจะเป็นอินเตอร์รัพท์ของ timer ในซีพียู หรืออินเตอร์รัพท์เพื่อจัดการกระบวนการภายในซีพียู ไม่เกี่ยวข้องกับอุปกรณ์อินพุต/เอาต์พุต

3. Software Interrupts เกิดจากชุดคำสั่งภายในซีพียู เป็นชุดคำสั่งประเภทอินเตอร์รัพท์ ทำงานคล้ายกับสับรูทีน



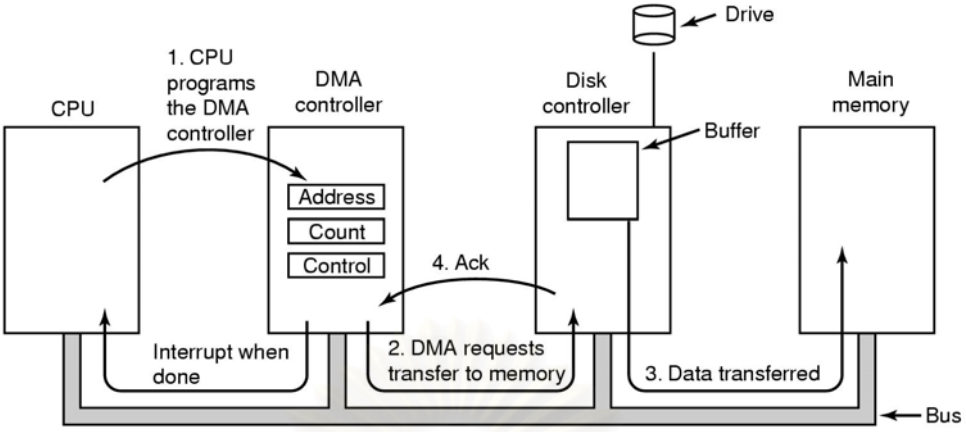
รูปที่ 2.18 การออกแบบอินเตอร์รัพท์ฮาร์ดแวร์

2.5 **ดีเอ็มเอ** (Direct Memory Access [DMA])

เป็นวิธีการอย่างหนึ่งในการเพิ่มประสิทธิภาพในการแลกเปลี่ยนข้อมูลระหว่างหน่วยความจำและอุปกรณ์อินพุต/เอาท์พุต ซึ่งเป็นการทำงานทั่วไปของเครื่องคอมพิวเตอร์ในปัจจุบัน ไม่ว่าจะเป็นการ โหลดโปรแกรมจาก ฮาร์ดดิสก์ไปยังหน่วยความจำ บันทึกข้อมูลลงในฮาร์ดดิสก์ เป็นต้น

เมื่อพิจารณาถึงการทำงานของคอมพิวเตอร์ที่ประกอบด้วย ซีพียู หน่วยความจำ และอุปกรณ์อินพุต/เอาท์พุต เมื่อต้องการโหลดโปรแกรมจาก ฮาร์ดดิสก์ซึ่งเป็นอุปกรณ์อินพุต/เอาท์พุต ไปยังหน่วยความจำ ซีพียูต้องอ่านไบต์แรกของโปรแกรมในฮาร์ดดิสก์ และส่งต่อไปยังหน่วยความจำ จากนั้นไบต์ที่สองก็ต้องทำเช่นเดิม จนโหลดโปรแกรมจากฮาร์ดดิสก์ไปยังหน่วยความจำเสร็จสมบูรณ์ จะเห็นว่าการทำงานดังกล่าวนั้นซีพียูจะมีส่วนในการทำงานทุกขั้นตอนทำให้ไม่ได้ประสิทธิภาพของระบบที่ดี เนื่องจากว่าความเร็วของซีพียูกับ อุปกรณ์อินพุต/เอาท์พุตนั้นต่างกันมาก ดังนั้นทำให้เกิดการใช้งานซีพียูได้อย่างไม่เต็มประสิทธิภาพ

วิธีการแก้ปัญหาดังกล่าวนั้นใช้หลักการ Direct Memory Access (DMA) เป็นการแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์อินพุต/เอาท์พุต โดยตรงไม่ผ่านซีพียู โดยจะมีตัวควบคุม DMA (DMA Controller) เป็นตัวจัดการแทนซีพียู ทำให้ซีพียูไม่ต้องอ่านข้อมูลจากฮาร์ดดิสก์ก่อนแล้วค่อยส่งไปเขียนยังหน่วยความจำเป็นผลให้ระบบมีประสิทธิภาพมากขึ้น



รูปที่ 2.19 การทำงานของดีเอ็มเอ

2.5.1 การทำงานของดีเอ็มเอ

-เริ่มจาก Processor ส่งข้อมูลดังต่อไปนี้ไปยัง DMA Controller

- Address เริ่มต้นของหน่วยความจำ, ความยาวของขนาดข้อมูล(Block length),ทิศทาง(จากหน่วยความจำไปยัง I/O หรือจาก I/O ไปยังหน่วยความจำ,พอร์ตในการติดต่อกับ I/O,สัญญาณสิ้นสุดของข้อมูล)

- จากนั้น Processor ก็จะทำงานต่อไป ในขณะที่ DMA มีการรับส่งข้อมูล

- ในขณะที่ DMA มีการติดต่อกับหน่วยความจำ จะมีสัญญาณบอกกับ Processor เพื่อไม่ให้ Processor ติดต่อกับหน่วยความจำในเวลาเดียวกับ DMA

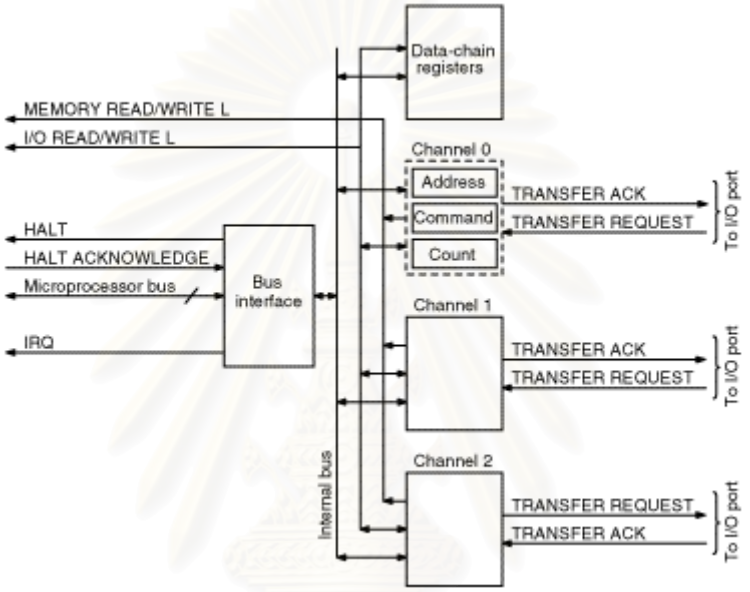
- เมื่อการรับส่งข้อมูลเสร็จสิ้นลงแล้ว DMA Controller จะบันทึกข้อมูลการเสร็จสิ้นการทำงานลงใน Status register หรือส่งสัญญาณ Interrupt ไปยัง Processor

- จากนั้น Processor ก็จะรับถึงการเสร็จสิ้นการรับส่งข้อมูลของ I/O ผ่าน status register หรือดูจากสัญญาณ Interrupt

2.5.2 การออกแบบดีเอ็มเอ

DMA Controller ประกอบด้วย register 3 ส่วนที่เป็นอิสระแก่กันกัน ประกอบด้วย Address register, control register และ byte count register เมื่อต้องการรับส่งข้อมูล

ระหว่าง I/O Port และหน่วยความจำ DMA Controller จะเก็บค่า Address เริ่มต้นที่ส่งมาจาก Processor และเริ่มนับข้อมูลใน Byte count register ก่อนที่จะเกิดการรับส่งข้อมูลนั้น I/O Device จะต้องพร้อม และ Processor อยู่ในสถานะ Idle การเชื่อมต่อระหว่าง DMA Controller กับ I/O Port ประกอบด้วยสัญญาณ Transfer Request และ Transfer Acknowledge ต่อ I/O Port รวมถึงสัญญาณ Memory Read/Write เพื่อบอกถึงทิศทางการส่งข้อมูล



รูปที่ 2.20 การออกแบบดีเอ็มเอ

2.5.3 การเชื่อมต่อ DMA หน่วยความจำ และ อุปกรณ์อินพุต/เอาท์พุต

การเชื่อมต่อกับอุปกรณ์อินพุต/เอาท์พุต (I/O) ประกอบด้วยสัญญาณ Transfer request ,Transfer ack ต่อ 1 port อุปกรณ์อินพุต/เอาท์พุต และสัญญาณ I/O Read/write เพื่อบอกถึงทิศทางการรับส่งข้อมูล DMA Controller รับสัญญาณ Transfer request จาก I/O Port เมื่อ I/O Port มีข้อมูลเตรียมพร้อมที่จะส่งไปเขียนยังหน่วยความจำ หรือบัฟเฟอร์ของ I/O Port ว่างและพร้อมรับข้อมูลจากหน่วยความจำ เมื่อการรับส่งข้อมูลเสร็จสิ้นแล้ว DMA สร้างสัญญาณ Transfer ack เพื่อบอกว่า I/O Port ได้รับข้อมูลแล้ว หรือ ได้เขียนข้อมูลลงหน่วยความจำแล้ว

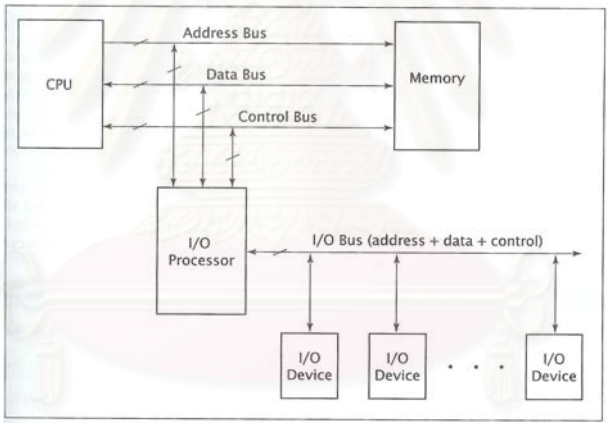
การเชื่อต่อกับหน่วยความจำ การเชื่อมต่อกับหน่วยความจำนั้นจะต่อผ่านบัสร่วมของ Processor กับหน่วยความจำ ประกอบด้วยสัญญาณ Halt request ,Halt acknowledge

ในขณะที่มีการรับส่งข้อมูล DMA Controller จะรอสัญญาณ Transfer request เมื่อสัญญาณ Transfer request มาแล้ว จากนั้น DMA Controller จะส่งสัญญาณ Halt request ไปยัง Processor เพื่อขอให้ Processor หยุดการติดต่อกับหน่วยความจำจากนั้น Processor จะส่งสัญญาณ Halt acknowledge มายัง DMA Controller จากนั้น DMA Cotroller ส่ง Address ลงบัส ต่อด้วยการส่งสัญญาณ Transfer ack ไปยัง I/O port และส่งสัญญาณ I/O read/write เพื่อบอกถึงทิศทางการรับส่งข้อมูล

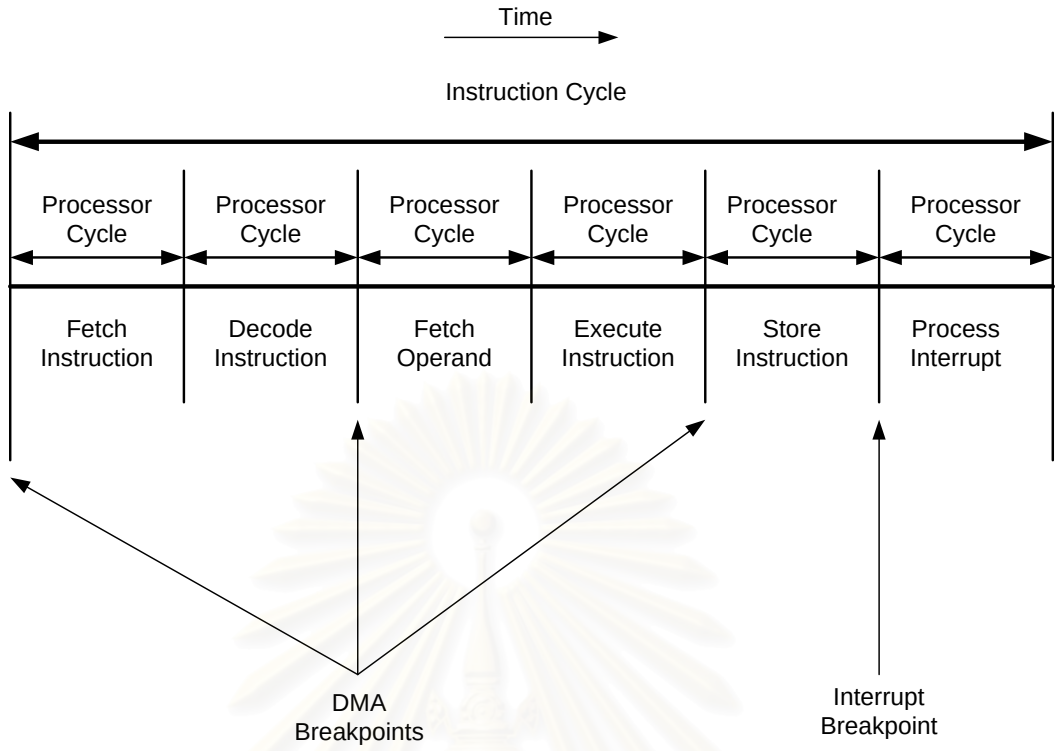
2.5.4 **อุปกรณ์อินพุต/เอาท์พุต**

เมื่อมีอุปกรณ์อินพุต/เอาท์พุตหลายๆ ตัวต่อกับบัสแล้ว การควบคุมทำได้ยากขึ้น อีกทั้งหากเป็นระบบที่ใช้ DMA ด้วยแล้ว อุปกรณ์แต่ละตัวจะต้องมี DMA ของตัวเองต่างหาก

I/O Controller เป็นตัวควบคุมอุปกรณ์อินพุต/เอาท์พุตที่สามารถทำงานเป็น DMA ได้ [2] และเพิ่มตามความสามารถในการจัดการลำดับความสำคัญของอุปกรณ์ต่าง ๆ ด้วย



รูปที่ 2.21 ระบบที่ประกอบด้วยอุปกรณ์ต่อพ่วง



รูปที่ 2.22 จุดที่ดีเอ็มเอและอินเตอร์รัพท์ทำงาน

2.6 **สรุป**

บทนี้นำเสนอความรู้เบื้องต้นในการทำวิจัยและงานวิจัยที่เกี่ยวข้อง ประกอบด้วย งานวิจัยเกี่ยวกับการออกแบบบัสสำหรับวงจรอสมวาร การออกแบบไมโครโพรเซสเซอร์ AMULET 3i และงานวิจัยการเชื่อมต่อวงจรสมวารกับวงจรอสมวาร จากนั้นนำเสนอ ความรู้เบื้องต้นในการออกแบบวงจรอสมวาร ประกอบด้วย แบบจำลองการทำงานสิ่งแวดล้อม แบบจำลองความหน่วง ช่องทางกับรับส่งข้อมูล โปรโตคอลอาณัติสัญญาณ การเข้ารหัส และวิธีการออกแบบวงจรโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ และนำเสนอการออกแบบบัสสำหรับคอมพิวเตอร์ ประกอบด้วย วิธีการออกแบบบัส ความสำคัญของลอจิกสามสถานะ ชนิดของบัส การทำงานของบัส และการรับส่งข้อมูลของบัส สุดท้ายนำเสนอแนวคิดการออกแบบดีเอ็มเอ อินเตอร์รัพท์ เพื่อเป็นความรู้เบื้องต้นในการออกแบบต่อไป

บทที่ 3

การออกแบบบัสสำหรับวงจรอสมวาร

บทนี้นำเสนอวิธีการออกแบบบัสแบบอสมวาร บัสแบบอสมวารที่ออกแบบจะออกแบบเป็นองค์ประกอบแล้วนำทั้งหมดมาประกอบกันเป็นบัส บทนี้แบ่งออกเป็น 5 ส่วนคือ คุณสมบัติของบัสนำเสนอในหัวข้อ 3.1 ส่วนประกอบของบัสแต่ละองค์ประกอบนำเสนอในหัวข้อที่ 3.2 และการทำงานของบัสนำเสนอในหัวข้อที่ 3.3 การจำลองการทำงานของบัส นำเสนอในหัวข้อที่ 3.4 และสุดท้าย สรุปการออกแบบบัสแบบอสมวารนำเสนอในหัวข้อที่ 3.5

3.1 คุณสมบัติของบัส

บัสแบบอสมวารออกแบบเป็นองค์ประกอบ รับส่งข้อมูลแบบมัลติเพล็กส์ ขั้นตอนการการออกแบบบัสเริ่มจาก จากการออกแบบองค์ประกอบแต่ละองค์ประกอบ จำลองการทำงานแต่ละองค์ประกอบให้ทำงานอย่างถูกต้อง แล้วจึงนำทุกองค์ประกอบมารวมกันแล้วจำลองการทำงานรวมกันอีกครั้ง บัสที่ออกแบบมีคุณสมบัติดังนี้

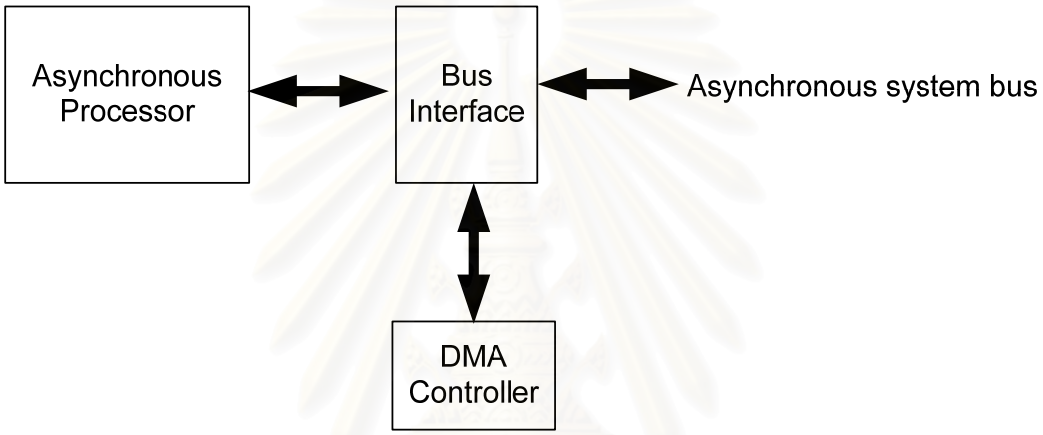
- สายสัญญาณและสายเลขที่อยู่ขนาด 8 บิต
- เข้ารหัสโดยใช้รหัสรางคู่
- ออกแบบโดยใช้แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วงชนิดเสมือน
- ใช้โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น
- มัลติเพล็กส์
- สามารถเชื่อมต่อกับวงจรสมวารและอสมวาร

3.2 องค์ประกอบบัส

องค์ประกอบของบัสประกอบด้วย 6 องค์ประกอบด้วยกัน คือ ส่วนเชื่อมต่อบัส ตัวขับบัส ตัวรับบัส สายสัญญาณบัส ตัวควบคุมบัส และส่วนเชื่อมต่อวงจรอสมวาร แต่ละองค์ประกอบจะออกแบบแยกจากกัน เมื่อได้การออกแบบทุกองค์ประกอบแล้ว จึงนำทั้งหมดมาต่อกันเพื่อทำงานเป็นบัสแบบอสมวารแล้วจำลองการทำงานเพื่อให้ได้การทำงานที่ถูกต้อง

3.2.1 **ส่วนเชื่อมต่อบัส**

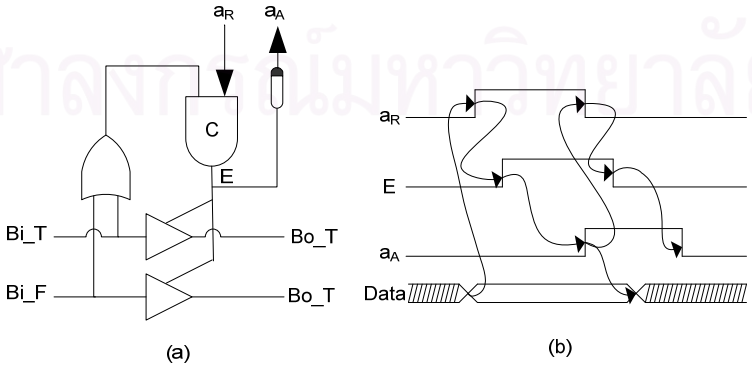
ส่วนเชื่อมต่อบัส ออกแบบเพื่อเชื่อมต่อระหว่าง ไมโครโพรเซสเซอร์แบบอสมวาร บัสแบบอสมวาร และดีเอ็มเอ เนื่องจากการออกแบบบัสแบบมัลติเพล็กส์ ซึ่งการรับส่งสัญญาณเลขที่อยู่และสัญญาณข้อมูลบนสายสัญญาณชุดเดียวกัน ทำให้การออกแบบส่วนเชื่อมต่อบัสต้องรับผิดชอบในการรับส่งข้อมูลด้วย เสมือนตัวเองเป็นส่วนควบคุมการรับส่งระหว่างสัญญาณเลขที่อยู่ สัญญาณขั้นว่าง และสัญญาณข้อมูล ดังแสดงการเชื่อมต่อของส่วนเชื่อมต่อบัสในรูปที่ 3.1



รูปที่ 3.1: การเชื่อมต่อของส่วนเชื่อมต่อบัส

3.2.2 **ตัวขับบัส**

ตัวขับบัส เป็นองค์ประกอบที่ทำงานคล้ายการทำงานของสวิทช์ปิด เปิด เพื่อยินยอมหรือ ปิดสัญญาณเข้าสู่บัส ออกแบบโดยใช้ลอจิกสามสถานะ และอุปกรณ์ชนิดซี

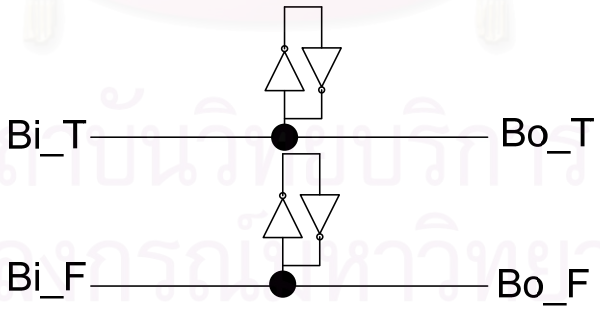


รูปที่ 3.2 การออกแบบตัวขับบัส

จากรูปที่ 3.2 แสดงการออกแบบตัวขับบัส การทำงานเริ่มจากเมื่อ มีสัญญาณข้อมูลมาทางขา Bi_T และ Bi_F ในลักษณะข้อมูลรางคู่ สัญญาณ a_R ซึ่งเป็นสัญญาณร้องขอจากส่วนควบคุมบัส เมื่อสัญญาณ a_R เปลี่ยนสัญญาณจาก 0->1 แล้วทำให้ อุปกรณ์ชนิดซี เปิดการทำงานของลอจิกสามสถานะ ผ่านสัญญาณ E ลอจิกสามสถานะก็จะปล่อยสัญญาณจาก ขา Bi_T และ Bi_F ไปยัง Bo_T และ Bo_F , หลังจากสัญญาณ E เปลี่ยนจาก 0->1 สัญญาณ a_A ซึ่งเป็นสัญญาณตอบรับเปลี่ยนจาก 0->1 เพื่อตอบรับจากสัญญาณร้องขอ ไปยังส่วนควบคุมบัส เป็นอันสิ้นสุดขั้นตอนการทำงาน จากนั้น สัญญาณทุกเส้นจะกลับสู่ศูนย์ ก่อนที่จะรับชุดข้อมูลชุดใหม่ ตามการทำงานของโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น

3.2.3 **สายสัญญาณบัส**

ลอจิกสามสถานะ ที่ใช้ในส่วนของตัวขับบัส จะสร้างสัญญาณที่ไม่ต้องการออกมา เป็นสถานะที่สาม เรียกว่า ไฮ-อิมพีแดนซ์ เป็นสถานะ ที่เกิดการสร้างกระแสสูงในสายสัญญาณ ทำให้ ไม่เกิดทั้งลอจิก 0 และลอจิก 1 อาจทำให้เกิดปัญหากับวงจรอสมวารได้ โดยวงจรอสมวารที่ออกแบบโดยใช้รหัสรางคู่ และใช้โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น นั้นจะยินยอมให้เกิดลอจิก 0 และลอจิก 1 เท่านั้น ดังนั้นจึงต้องหาวิธีการทำให้สายสัญญาณบัสเป็นลอจิก 0 เมื่อลอจิกสามสถานะอยู่ที่สถานะ ไฮ-อิมพีแดนซ์ โดยการเพิ่มเกตผกผันจำนวนสองตัวกลับหัวกันบนสายสัญญาณบัสแต่ละเส้น ดังแสดงในรูปที่ 3.3

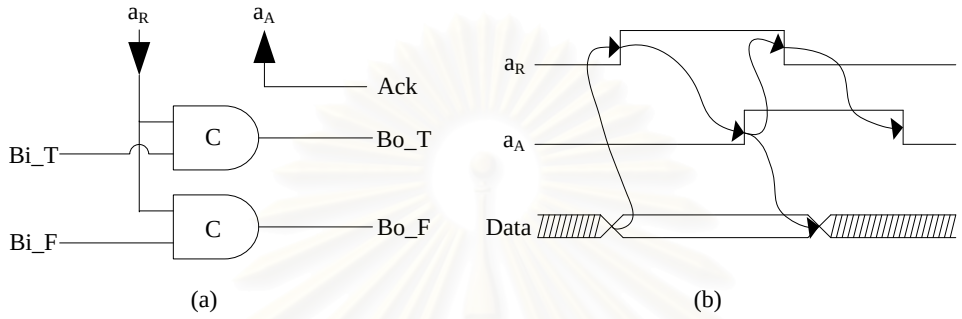


รูปที่ 3.3 การต่อเกตผกผันกับสายสัญญาณบัส

จากรูปที่ 3.3 เมื่อเกิดสถานะไฮ-อิมพีแดนซ์ (Z) ที่มาจากเส้น Bi_T และ Bi_F , จะทำให้สัญญาณเป็น 0 เมื่อผ่านเกตุผกผันที่ต่อกลับหัวกัน ไปยังสัญญาณ Bo_T และ Bo_F

3.2.4 **ตัวรับบัส**

ตัวรับบัส ออกแบบเพื่อเป็นอุปกรณ์รับสัญญาณจากบัส แล้วทำการส่งต่อไปยังปลายทางที่เชื่อมต่อ ออกแบบโดยใช้อุปกรณ์ซนิดซี วัตถุประสงค์หลังของตัวรับบัสคือ จับสัญญาณที่รับมาจากบัส แล้วส่งต่อตามสัญญาณควบคุมจากตัวควบคุมบัส

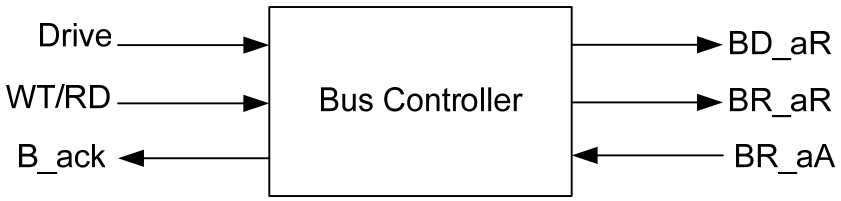


รูปที่ 3.4 : การออกแบบตัวรับบัส

จากรูปที่ 3.4 สัญญาณร้องขอ a_R เป็นสัญญาณที่มาจากตัวควบคุมบัส สัญญาณ Bi_T และ Bi_F เป็นสัญญาณข้อมูลแบบรางคู่ที่รับมาจากบัส เมื่อสัญญาณข้อมูลรางคู่มาถึงตามด้วยสัญญาณ a_R เปลี่ยนจาก 0 ->1 จากนั้นอุปกรณ์ซนิดซี ซึ่งเป็นอุปกรณ์เก็บสถานะ จะส่งสัญญาณรางคู่ที่รับมาจากบัส ส่งผ่านไปยังสัญญาณ Bo_T และ Bo_F จากนั้นสัญญาณ a_A เป็นสัญญาณตอบรับที่มาจากส่วนเชื่อมต่อวงจรสมวารและหน่วยความจำเปลี่ยนจาก 0->1 เพื่อตอบรับการมาถึงของสัญญาณรางคู่ จากนั้นสัญญาณทุกเส้นเริ่มเปลี่ยนจาก 1->0 เพื่อทำการรีเซ็ตสายสัญญาณตามลักษณะการทำงานของโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น

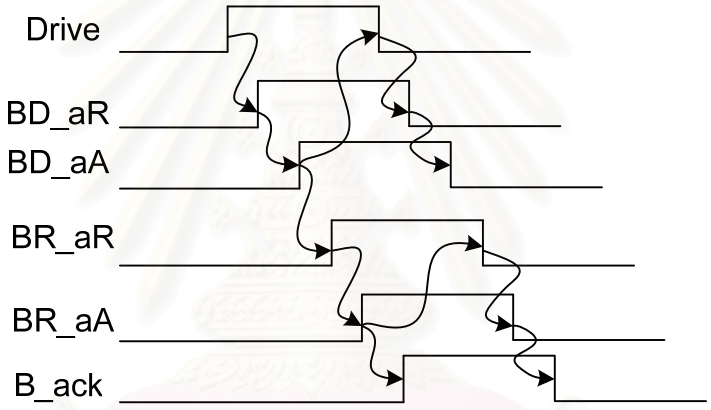
3.2.5 **ตัวควบคุมบัส**

ตัวควบคุมบัส ออกแบบโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ (Signal Transition Graph) ซึ่งเป็นวีธีการออกแบบวงจรโดยใช้กราฟบรรยายการเปลี่ยนสถานะของสัญญาณ กราฟบรรยายการเปลี่ยนสัญญาณถูกใช้เป็นอย่างมากในการออกแบบวงจรควบคุมสำหรับการออกแบบวงจรอสมวาร



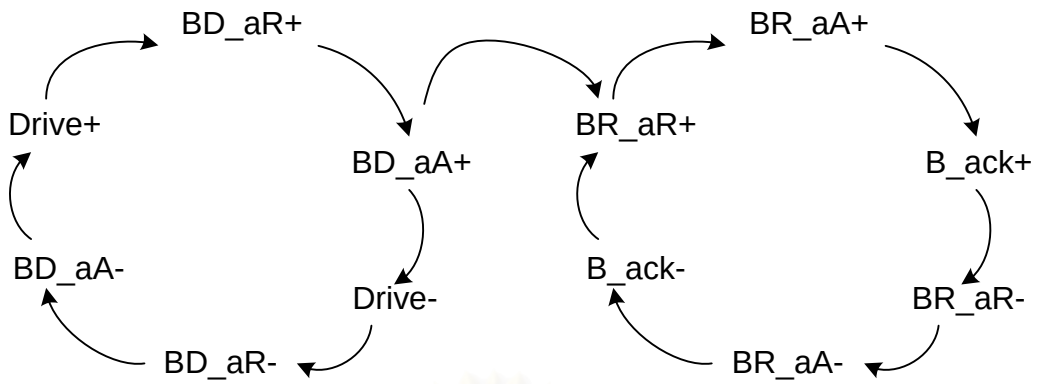
รูปที่ 3.5 การออกแบบตัวควบคุมบัส

ขั้นตอนในการออกตัวควบคุมบัส โดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ เริ่มจากกำหนดคุณสมบัติ การเปลี่ยนสัญญาณ ซึ่งประกอบด้วยสัญญาณอินพุต และเอาท์พุต ดังแสดงในรูปที่ 3.5 และวาดลำดับการเปลี่ยนสัญญาณตามคุณสมบัติของตัวควบคุมบัสดังแสดงในรูปที่ 3.6



รูปที่ 3.6 การกำหนดคุณสมบัติของตัวควบคุมบัส

สัญญาณ Drive เป็นสัญญาณเริ่มต้นการอ่านเขียนข้อมูลของบัส รับมาจากไมโครโพรเซสเซอร์แบบอสมวาร เมื่อสัญญาณ Drive เปลี่ยนจาก 0 ->1 แล้ว สัญญาณ BD_a_R , BD_a_A , BR_a_A , BR_a_A , B_{ack} เปลี่ยนจาก 0 ->1 ตามลำดับ เมื่อสัญญาณทุกเส้นเป็น 1 แล้วจะเสร็จสิ้นขั้นทำงาน ตามการทำงานของโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น จากนั้นสัญญาณทุกเส้นจะเริ่มเปลี่ยนจาก 1->0 ตามลำดับ เป็นขั้นว่าง เพื่อกลับสัญญาณทุกเส้นให้เป็น 0 ก่อนรับสัญญาณชุดใหม่ ตามการทำงานของโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น



รูปที่ 3.7: กราฟบรรยายการเปลี่ยนสัญญาณของตัวควบคุมบัส

เพื่อให้ได้วงจรที่มีความปลอดภัยภายใต้แบบจำลองความหน่วงที่เลือกใช้ ตามการออกแบบวงจรควบคุมโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ ต้องออกแบบภายใต้ข้อกำหนดดังนี้

แบบจำลองความหน่วง

- ความหน่วงเกตเป็นแบบไม่มีขอบเขต ความหน่วงสิ่งแวดล้อมเป็นแบบไม่มีขอบเขต
- ความหน่วงสายสัญญาณมีค่าน้อยมาก จนสามารถประมาณค่าเป็นศูนย์ได้

ข้อกำหนดของกราฟบรรยายการเปลี่ยนสัญญาณ

- ปลอดภัย
- ไลฟ์
- ความสัมพันธ์เกี่ยวเนื่องกัน
- คงเส้นคงวา
- การกำหนดสถานะโดยสมบูรณ์

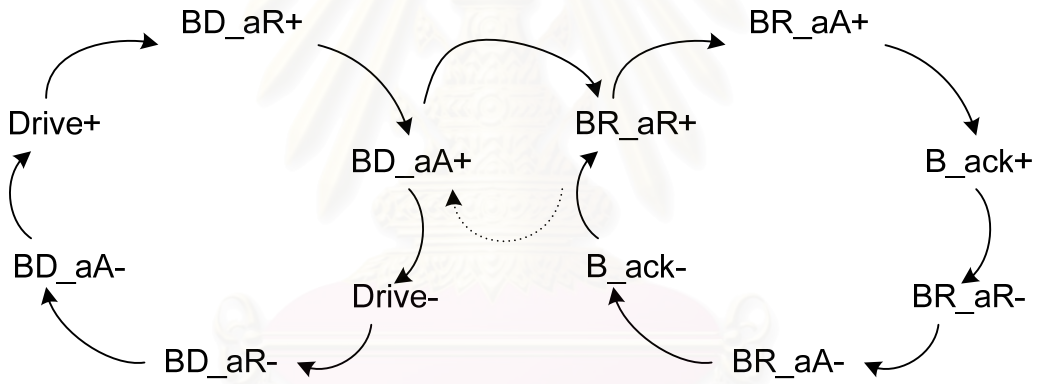
เงื่อนไขสำหรับการอิมพลีเม้นท์:

- ความสัมพันธ์เกี่ยวเนื่องกัน
- คงเส้นคงวา

- การกำหนดสถานะโดยบริบูรณ์

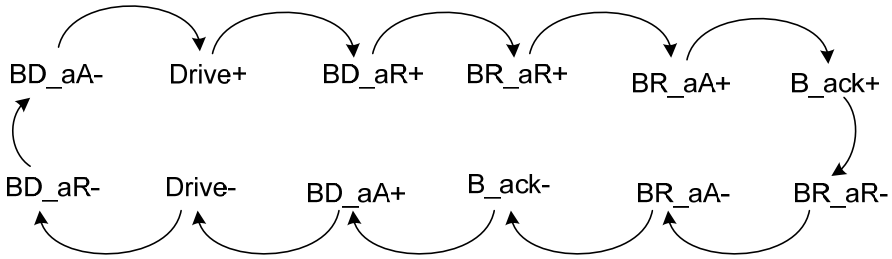
การพิจารณาเงื่อนไขของกราฟบรรยายการเปลี่ยนสัญญาณสำหรับตัวควบคุมบัส

กราฟบรรยายการเปลี่ยนสัญญาณดังแสดงในรูปที่ 3.7 หากพิจารณาดูแล้วไม่มีความปลอดภัย เนื่องจากมีการออกแบบเกือบจะแยกกันเป็นสองส่วน สังเกตหากวงทางซ้ายมีความเร็วมากกว่าวงทางขวามากๆ แล้วอาจทำให้วงทางขวาทำงานผิดพลาด เนื่องจากสัญญาณจากวงทางซ้ายอาจทำให้สัญญาณวงทางขวาเปลี่ยนได้ก่อนเวลาที่ต้องการ ดังนั้นจึงต้องเพิ่มสัญญาณเพื่อหยุดการทำงานของวงทางซ้าย โดยการเพิ่มสัญญาณเพิ่มเติมที่มาจากวงทางขวาดังแสดงในรูปที่ 3.8



รูปที่ 3.8 การเพิ่มสัญญาณเพื่อความปลอดภัย

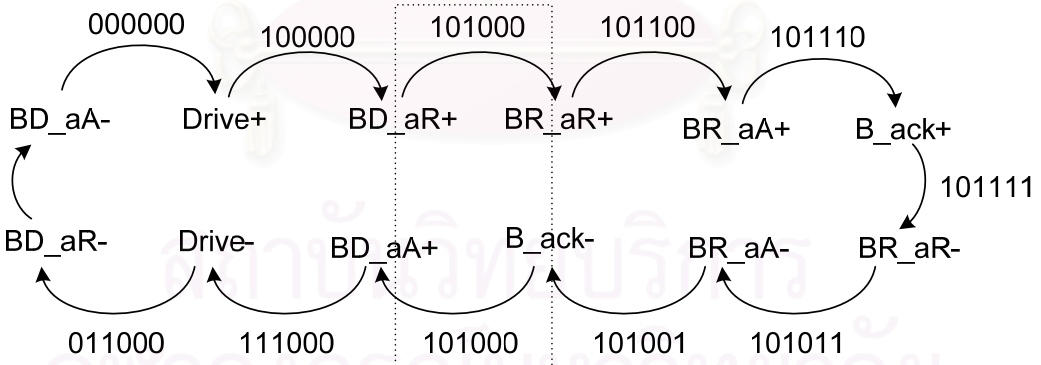
สำหรับคุณสมบัติ การไลฟ์ของกราฟบรรยายการเปลี่ยนสัญญาณของวงจรควบคุมบัสนั้น สังเกตกราฟบรรยายการเปลี่ยนสัญญาณดังแสดงในรูปที่ 3.9 เป็นวงจรที่ได้ใหม่ หลังจากพิจารณาความปลอดภัย จะเห็นได้ว่า สัญญาณทุกเส้นทางสามารถเคลื่อนที่ได้หมด ซึ่งรับประกันคุณสมบัติการไลฟ์ของกราฟบรรยายการเปลี่ยนสัญญาณได้เป็นอย่างดี



รูปที่ 3.9 กราฟบรรยายการเปลี่ยนสัญญาณหลังจากพิจารณาความปลอดภัย

เมื่อพิจารณาคุณสมบัติความสัมพันธ์เกี่ยวเนื่องกัน กราฟบรรยายการเปลี่ยนสถานะดังแสดงในรูปที่ 3.9 สามารถบรรยายได้ถึงความสัมพันธ์ระหว่างสัญญาณ + และสัญญาณ – ได้เป็นอย่างดี

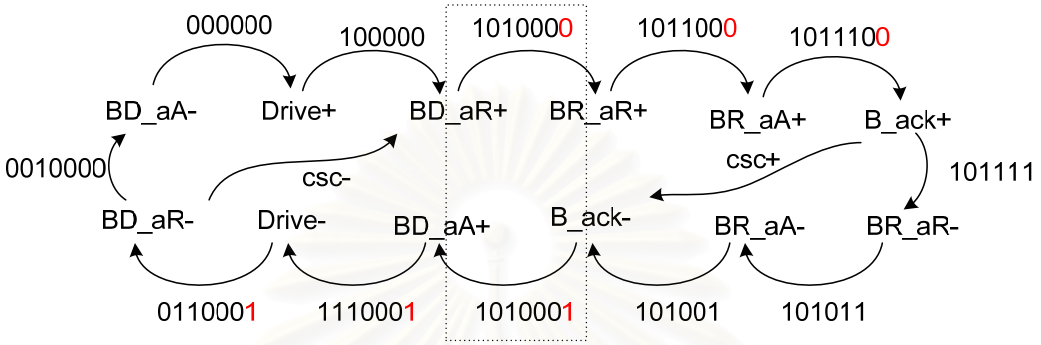
ความคงเส้นคงวา พิจารณาคุณสมบัติความเกี่ยวเนื่องกันแล้ว จะเห็นว่า สัญญาณ + และสัญญาณ – เปลี่ยนในลักษณะที่มีความคงเส้นคงวาได้ดี คือ หากเริ่มด้วยสัญญาณ + แล้วก็จะเป็น + ตลอดไป กราฟบรรยายการเปลี่ยนสถานะของวงจรควบคุมบัสดังแสดงในรูปที่ 3.9 พิสูจน์ถึงการมีความคงเส้นคงวาได้เป็นอย่างดี



รูปที่ 3.10 การกำหนดสถานะสำหรับตัวควบคุมบัส

พิจารณาการกำหนดสัญญาณโดยบริบูรณ์ (Complete State Coding) กราฟบรรยายการเปลี่ยนสัญญาณมีการกำหนดสถานะโดยบริบูรณ์ถ้าไม่มีสถานะที่ซ้ำซ้อนกันในการกำหนดสถานะ จากการกำหนดสถานะดังแสดงในรูปที่ 3.10 จะเห็นว่าเมื่อมีการกำหนดสถานะให้กับกราฟบรรยายการเปลี่ยนสัญญาณของวงจรควบคุมบัสแล้ว จะเกิดการซ้ำซ้อนกันของสถานะ ที่

101000 ดังนั้นกราฟบรรยายการเปลี่ยนสัญญาณดังแสดงในรูปที่ 3.10 ไม่ผ่านข้อกำหนดการกำหนดสัญญาณโดยบริบูรณ์ ต้องหาวิธีแก้ปัญหาโดยการเพิ่มสัญญาณภายใน เพื่อกำจัดสถานะที่ซ้ำซ้อนออกไปดังแสดงในรูปที่ 3.11



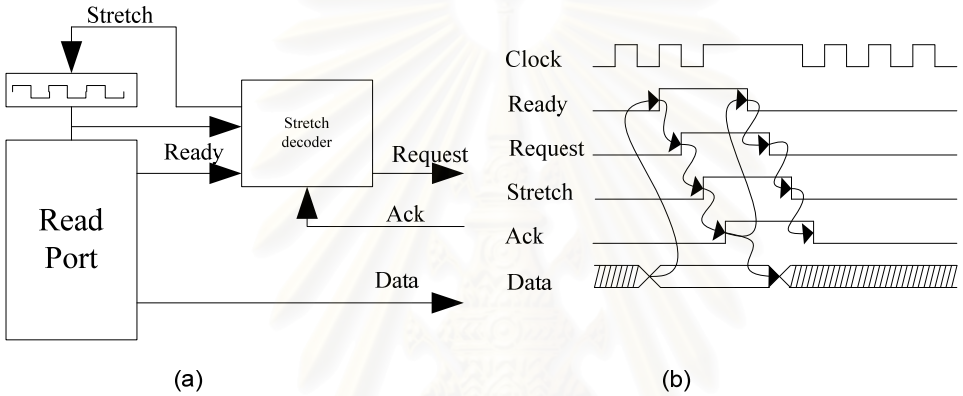
รูปที่ 3.11 การเพิ่มสัญญาณภายใน

รูปที่ 3.11 แสดงการเพิ่มสัญญาณภายใน csc- และ csc+ เพื่อกำจัดความซ้ำซ้อนกันของการกำหนดสถานะสำหรับวงจรควบคุมบัส กราฟบรรยายการเปลี่ยนสัญญาณในรูปที่ 3.11 มีการกำหนดสัญญาณที่ไม่ซ้ำซ้อนกันแล้ว เป็นการรับประกันคุณสมบัติของการกำหนดสัญญาณโดยบริบูรณ์ของกราฟบรรยายการเปลี่ยนสัญญาณของวงจรควบคุมบัสนั่นเอง จากรูปที่ 3.11 จะเห็นว่า กราฟบรรยายการเปลี่ยนสัญญาณ มีคุณสมบัติความสัมพันธ์เกี่ยวเนื่องกัน มีคุณสมบัติความคงเส้นคงวา และมีคุณสมบัติการกำหนดสถานะโดยบริบูรณ์ ดังนั้นกราฟบรรยายการเปลี่ยนสัญญาณดังแสดงในรูปที่ 3.11 สามารถนำมาสังเคราะห์เป็นวงจรได้ ตามข้อกำหนดของกราพบรรยายการเปลี่ยนสัญญาณ โดยการนำไปหา Next-state function และลดรูปวงจรโดยใช้ผังคาโนห์ แล้วจะได้วงจรควบคุมบัสในตอนท้ายที่สุด

3.2.6 **ส่วนเชื่อมต่อวงจรอสมวารและหน่วยความจำ**

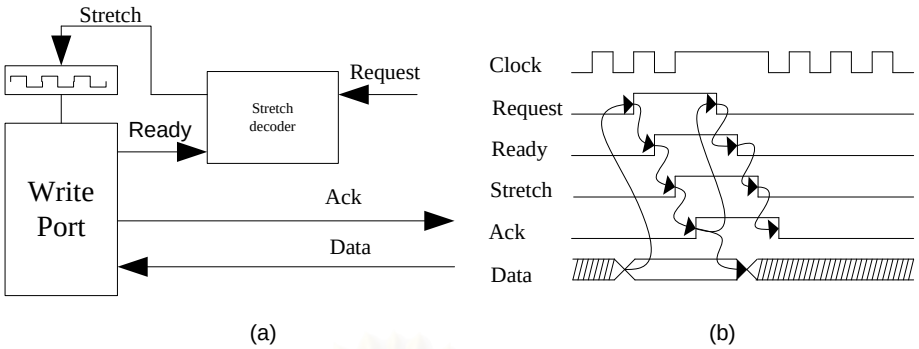
ส่วนนี้เป็นการออกแบบเพื่อให้บัสแบบอสมวารที่ออกแบบนั้นสามารถเชื่อมต่อกับวงจรสมวารได้ ไม่ว่าจะเป็นอุปกรณ์ต่อพ่วงต่าง ๆ หรือหน่วยความจำแบบสมวาร การออกแบบส่วนเชื่อมต่อวงจรสมวารนั้นจะเป็นส่วนที่ใช้สัญญาณนาฬิกา ในการติดต่อกับวงจรที่เป็นสมวาร ซึ่งจะไม่ใช้สัญญาณนาฬิกา การออกแบบส่วนเชื่อมต่อวงจรสมวาร ออกแบบโดยใช้สัญญาณนาฬิกาแบบยืดหยุ่น ซึ่งมีสัญญาณควบคุมสัญญาณนาฬิกาเพื่อให้ทำงานไม่เป็นไปตาม clock

period สัญญาณควบคุมความยืดหยุ่นจะมีผลเฉพาะตอนขาขึ้นของสัญญาณนาฬิกา การทำงานของสัญญาณนาฬิกาแบบยืดหยุ่น เป็นดังนี้ คือ สัญญาณนาฬิกาจะคงค้างไว้เมื่อสัญญาณ ควบคุมความยืดหยุ่นเปลี่ยนจาก 0->1 หลังจากนั้นเมื่อวงจรสมวารกับวงจรอสมวารแลกเปลี่ยนข้อมูลเสร็จสิ้นแล้ว สัญญาณควบคุมความยืดหยุ่นก็จะเปลี่ยนจาก 1->0 เพื่อปล่อยสัญญาณนาฬิกาทำงานตาม clock period ที่ความถี่ปกติ เมื่อความถี่ของสัญญาณควบคุมความยืดหยุ่นมีค่าน้อยกว่าความถี่ของสัญญาณนาฬิกา สัญญาณนาฬิกาจะยังคงทำงานที่ clock period ที่ความถี่ปกติ การเปลี่ยนสัญญาณจาก 0->1 ของส่วนควบคุมความยืดหยุ่นจะไม่มีผลในการทำให้สัญญาณนาฬิกาเปลี่ยนไป



รูปที่ 3.12 การออกแบบวงจรอ่านข้อมูลสำหรับสัญญาณนาฬิกาแบบยืดหยุ่น

เพื่อให้ได้ค่าใกล้เคียงกับความจริงของแหล่งกำเนิดความถี่ การเชื่อมต่อวงจรสมวารกับวงจรอสมวารควรเป็นไปตามข้อกำหนดที่ไม่ทำให้เกิดสภาวะการแย่งชิงกัน (Meta stability) สัญญาณนาฬิกาต้องมีสัญญาณอินพุตเป็นสัญญาณควบคุมความยืดหยุ่นด้วยเพื่อป้องกันการเคลื่อนที่ของ clock period ลูกปัจจุบัน การออกแบบสัญญาณควบคุมความยืดหยุ่น ต้องออกแบบให้ทำงานแบบสมวารในกรณีการเปลี่ยนสัญญาณจาก 0->1 และออกแบบให้เปลี่ยน 1->0 แบบอสมวาร ซึ่งจังหวะเป็นการปล่อยสัญญาณนาฬิกาเพื่อให้ทำงานตาม clock period ตามความถี่ปกติ

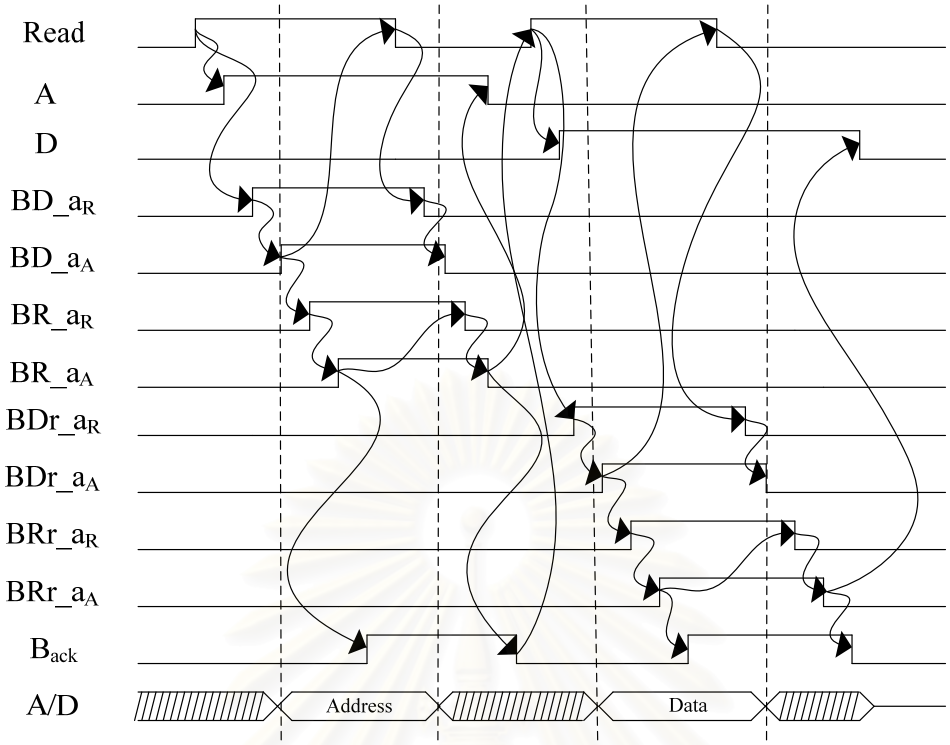


รูปที่ 3.13 การออกแบบวงจรเขียนข้อมูลสำหรับสัญญาณนาฬิกาแบบยืดหยุ่น

การควบคุมสัญญาณนาฬิกาสร้างขึ้นโดยสัญญาณแฮนด์เชคและสัญญาณภายในของสถานะของเครื่องจักรสถานะ ส่วนเชื่อมต่อประกอบด้วยพอร์ทการอ่าน และพอร์ทการเขียน พอร์ทการอ่าน จะทำการอ่านข้อมูลเพื่อออกไปยังอุปกรณ์ภายนอก ในขณะที่พอร์ทการเขียนจะรับข้อมูลเข้ามาเพื่อส่งต่อไปยังส่วนที่เป็นวงจรอสมวารต่อไป การการทำงานของโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น ทุกส่วนเชื่อมต่อ จะรับการรับส่งที่เป็นส่วนอสมวาร 2 ครั้ง เพื่อเสร็จสิ้นการทำงาน 1 ไซเคิล ส่วนที่เป็นส่วนเชื่อมต่อวงจรสมวารจะใช้สัญญาณนาฬิกา 2 ลูกในการทำงานแต่ละครั้งของสัญญาณควบคุมความยืดหยุ่น

3.3 การทำงานของบัส

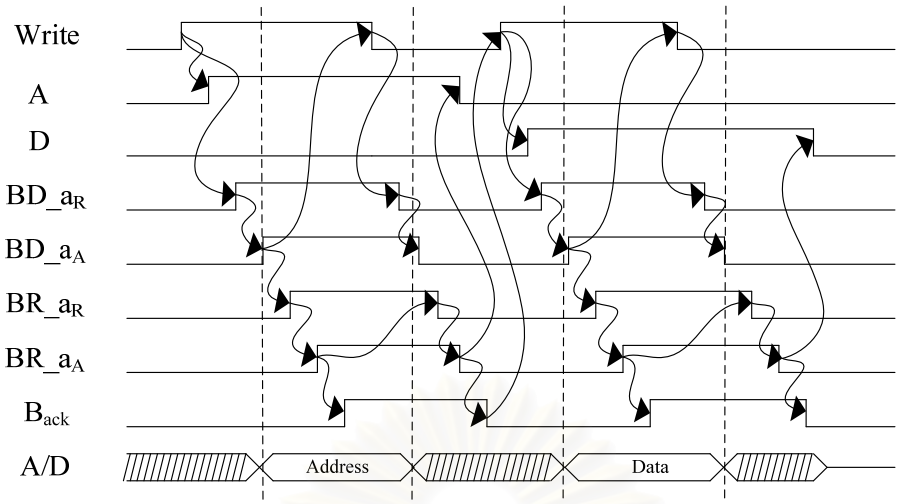
มัลติเพล็กส์บัสแบบอสมวารที่ออกแบบสามรถแบ่งการทำงานได้ ดังนี้คือ จังหวะการอ่านข้อมูลของบัส และจังหวะการเขียนข้อมูลของบัส



รูปที่ 3.14 จังหวะการอ่านข้อมูลของบัสแบบอสมวาร

สำหรับการอ่านข้อมูล สัญญาณร้องขอการอ่านข้อมูลจะเปลี่ยนจาก 0->1 จากนั้นสัญญาณ A ซึ่งเป็นสัญญาณเลขที่อยู่เปลี่ยนจาก 0->1 เพื่อบ่งบอกการส่งสัญญาณเลขที่อยู่เข้าบัส เพื่อไปอ่านข้อมูลออกมา

สัญญาณ BD_a_R, BD_a_A, BR_a_R, BR_a_A เป็นสัญญาณควบคุมที่มาจากส่วนควบคุมบัสเปลี่ยนจาก 0->1 ตามลำดับ บ่งบอกถึงการไหลข้อมูลจากตัวขับบัสไปยังตัวรับบัส จากนั้นสัญญาณ B_{ack} เปลี่ยนจาก 0->1 เพื่อบ่งบอกความบริบูรณ์ของขั้นทำงาน ตามลักษณะการทำงานของโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น เมื่อสิ้นสุดขั้นทำงานแล้วสัญญาณทุกเส้นเริ่มเข้าสู่ขั้นว่าง โดยเปลี่ยน 1->0 สำหรับการรีเซ็ตสายสัญญาณ เสร็จสิ้นการรับส่งแบบ 4 ขั้น จากนั้นสัญญาณ D ซึ่งเป็นสัญญาณข้อมูลเปลี่ยนจาก 0->1 สำหรับการร้องขอเพื่อรับส่งส่วนของข้อมูล ซึ่งการเปลี่ยนสัญญาณก็จะคล้ายกับการเปลี่ยนสัญญาณในขั้นตอนการส่งสัญญาณเลขที่อยู่ แต่จะเปลี่ยนสัญญาณควบคุมจาก BD_a_R, BD_a_A, BR_a_R, BR_a_A เป็น BDr_a_R, BDr_a_A, BRr_a_R, BRr_a_A ตามลำดับ ดังแสดงในรูปที่ 3.14



รูปที่ 3.15 : จังหวะการเขียนข้อมูลของบัสแบบอสมวาร

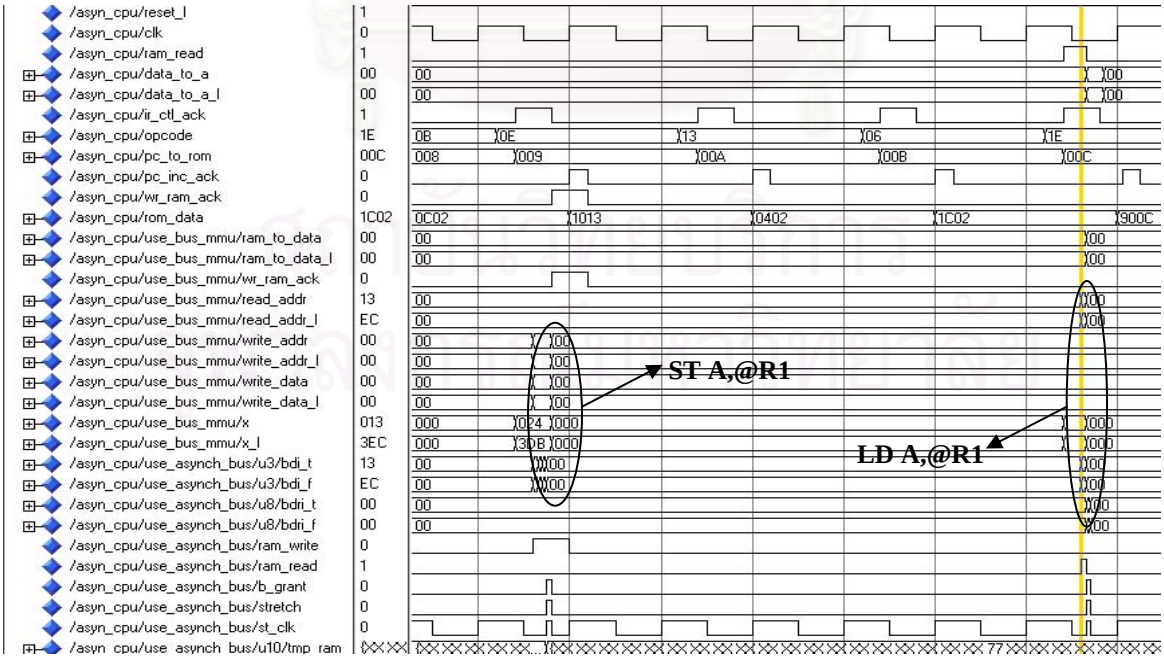
สำหรับการเขียนข้อมูล สัญญาณร้องขอการเขียนข้อมูลจะเปลี่ยนจาก 0->1 จากนั้นสัญญาณ A ซึ่งเป็นสัญญาณเลขที่อยู่เปลี่ยนจาก 0->1 เพื่อบ่งบอกการส่งสัญญาณเลขที่อยู่เข้าบัส

สัญญาณ BD_a_R, BD_a_A, BR_a_R, BR_a_A เป็นสัญญาณควบคุมที่มาจากส่วนควบคุมบัสเปลี่ยนจาก 0->1 ตามลำดับ บ่งบอกถึงการไหลข้อมูลจากตัวขับบัสไปยังตัวรับบัส จากนั้นสัญญาณ B_{ack} เปลี่ยนจาก 0->1 เพื่อบ่งบอกความบริบูรณ์ของขั้นทำงาน ตามลักษณะการทำงานของโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น เมื่อสิ้นสุดขั้นทำงานแล้วสัญญาณทุกเส้นเริ่มเข้าสู่ขั้นว่าง โดยเปลี่ยน 1->0 สำหรับการรีเซ็ตสายสัญญาณ เป็นการเสร็จสิ้นการับส่งแบบ 4 ขั้น จากนั้นสัญญาณ D ซึ่งเป็นสัญญาณข้อมูลเปลี่ยนจาก 0->1 สำหรับการร้องขอเพื่อรับส่งส่วนของข้อมูล ซึ่งการเปลี่ยนสัญญาณก็จะคล้ายกับการเปลี่ยนสัญญาณในขั้นตอนการส่งสัญญาณเลขที่อยู่ ยังคงใช้สัญญาณควบคุมจากส่วนควบคุมบัสชุดเดิมคือ BD_a_R, BD_a_A, BR_a_R, BR_a_A ดังแสดงในรูปที่ 3.15

3.4 การจำลองการทำงาน

โปรแกรมทดสอบที่ใช้จำลองการทำงานการอ่านเขียนข้อมูลของบัสแบบอสมวาร

```
ORG    0000H
LD     A,#77   ; A=77H
ST     A,R0    ; R0=77H
LD     A,#24H  ; A=24H
ST     A,R1    ; R1=24H
ST     A,@R0   ; M(77H) = 24H
LD     A,R0    ; A=77H
ST     A,@13H ; M(13H) =77H
LD     A,@77H ; A= 24H
ST     A,@R1          ; M(24H) = 24H
LD     A,#13H   ; A=13H
ST     A,R1     ; R1=13H
LD     A,@R1  ; A= 77H
HALT :
JMP    HALT
END
```



รูปที่ 3.16 ผลจำลองการทำงานของบัสแบบอสมวาร

3.5 สรุป

บทนี้นำเสนอวิธีการออกแบบบัสสำหรับวงจรอสมวาร บัสที่ออกแบบเป็นแบบมัลติเพล็กส์ กล่าวคือ สายสัญญาณข้อมูลและสายสัญญาณเลขที่อยู่ จะใช้สายสัญญาณร่วมกัน บัสที่ออกแบบสามารถเชื่อมต่อได้ทั้งวงจรสมวารและวงจรอสมวาร วงจรเชิมผสมที่ออกแบบ จะออกแบบโดยใช้รหัสรางคู่ และใช้โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น ชนิดกลับสู่ศูนย์ วงจรที่ออกแบบแบ่งออกเป็น 6 องค์ประกอบด้วยกัน คือ ส่วนเชื่อมต่อบัส ใช้สำหรับเชื่อมต่อระหว่างไมโครโพรเซสเซอร์แบบอสมวาร บัสแบบอสมวาร และดีเอ็มเอ ตัวขับบัสออกแบบโดยใช้ลอจิกสามสถานะ ใช้ในการปิดเปิดสัญญาณเข้าสู่บัส สายสัญญาณบัสออกแบบโดยเพิ่มส่วนเกตผกผัน กลับหัวกันสองตัว เพื่อกำจัดสถานะไฮ-อิมพีแดนซ์ ให้เป็นสัญญาณ 0 ตัวรับบัส ออกแบบโดยใช้อุปกรณ์ชนิดซี ใช้เพื่อรับสัญญาณจากบัส แล้วส่งต่อไปยังจุดหมาย ตัวควบคุมบัส ออกแบบโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ และ ส่วนเชื่อมต่อวงจรสมวารและอสมวารออกแบบโดยใช้สัญญาณนาฬิกาแบบยืดหยุ่น ตามด้วยการแสดงการทำงานของบัส และจำลองการทำงานในตอนท้าย

บทที่ 4

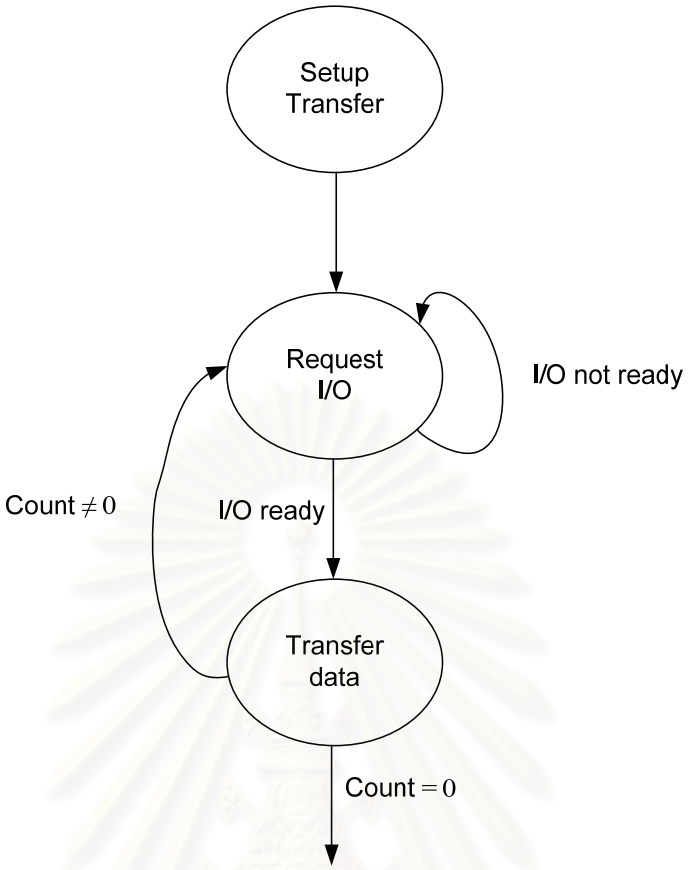
การออกแบบดีเอ็มเอ

บทนี้นำเสนอวิธีการออกแบบดีเอ็มเอ ซึ่งดีเอ็มเอจะออกแบบเป็นทั้งวงจรสมวาร และวงจรอสมวาร โดยการนำเสนอคุณสมบัติของดีเอ็มเอที่ออกแบบ และหลักการพื้นฐานในการออกแบบดีเอ็มเอในหัวข้อที่ 4.1 และนำเสนอการออกแบบดีเอ็มเอ ฟังก์ชั่นการทำงานของดีเอ็มเอ สถาปัตยกรรมดีเอ็มเอในหัวข้อที่ 4.2 จากนั้นนำเสนอการออกแบบส่วนควบคุมดีเอ็มเอแบบสมวารในหัวข้อที่ 4.3 การออกแบบดีเอ็มเอแบบอสมวารนำเสนอในหัวข้อที่ 4.4 และสรุปในหัวข้อที่ 4.5

4.1 คุณสมบัติของดีเอ็มเอ

ดีเอ็มเอที่ออกแบบประกอบด้วย รีจิสเตอร์ ส่วนควบคุม ส่วนคำนวณ ซึ่งรีจิสเตอร์ ที่เก็บข้อมูลเลขที่อยู่ จะเก็บข้อมูลเลขที่อยู่สำหรับการอ่านข้อมูล หรือเขียนข้อมูลของดีเอ็มเอ ส่วนคำนวณก็จะรับผิดชอบในการเพิ่มค่าสำหรับ รีจิสเตอร์เก็บเลขที่อยู่ และลดค่าสำหรับตัวนับ คุณสมบัติของดีเอ็มเอที่ออกแบบสามารถสรุปได้ ดังนี้

- การรับส่งข้อมูลขนาด 8 – ตามการออกแบบบัสแบบอสมวาร
- การรับส่งข้อมูลสามารถทำการรับส่งได้ระหว่าง
  - หน่วยความจำและอุปกรณ์ต่อพ่วง
  - อุปกรณ์ต่อพ่วงกับอุปกรณ์ต่อพ่วง
  - หน่วยความจำกับหน่วยความจำ
- ใช้การส่งข้อมูลแบบเก็บแล้วส่งต่อ – ตามการทำงานของบัสแบบอสมวาร
- มีการเพิ่มสัญญาณพิเศษ เพื่อการซิงโครไนซ์อย่างสมบูรณ์



รูปที่ 4.1 แนวคิดการออกแบบดีเอ็มเอ

4.2 การออกแบบดีเอ็มเอ

ไมโครโพรเซสเซอร์แบบอสมวาร เป็นอุปกรณ์มาสเตอร์ในระบบ หมายความว่า ไมโครโพรเซสเซอร์แบบอสมวารสามารถใช้งานบัสได้ตลอดเวลา ไม่ว่าจะเริ่มการเขียนหรืออ่านข้อมูลผ่านบัส ส่วนอุปกรณ์ตัวอื่นจะเป็นสลาฟ หมายความว่า เมื่อมีอุปกรณ์ตัวใดที่เป็นสลาฟต้องการใช้งานบัสแล้ว จะต้องร้องขอการอนุญาตจากไมโครโพรเซสเซอร์แบบอสมวารในการอ่าน หรือเขียนข้อมูลของตัวเองผ่านบัส ดีเอ็มเอก็เช่นกัน หากต้องการใช้งานแล้ว ดีเอ็มเอต้องทำการร้องของการอนุญาตการใช้บัสจากไมโครโพรเซสเซอร์แบบอสมวาร เมื่อดีเอ็มเอได้รับอนุญาตการใช้บัสแล้ว ดีเอ็มเอจะเป็นอุปกรณ์มาสเตอร์ในระบบแทนไมโครโพรเซสเซอร์แบบอสมวาร จนกว่าดีเอ็มเอจะคืนการทำงานให้กับไมโครโพรเซสเซอร์แบบอสมวาร

เนื่องจากมีการแย่งชิงกันเป็นมาสเตอร์เพื่อควบคุมบัสแบบอสมวารในระบบ จึงต้องมีตัวตัดสินใจ (Arbiter) เพื่อจัดสรรการเป็นมาสเตอร์ให้กับระบบ ตัวตัดสินใจเป็นอุปกรณ์ที่จะเลือกว่าจะให้ไมโครโพรเซสเซอร์แบบอสมวาร หรือดีเอ็มเอเป็นมาสเตอร์ในเวลานั้น ๆ ตามลักษณะการออกแบบ ซึ่งตัวตัดสินใจที่ออกแบบจะออกแบบให้อยู่ภายในไมโครโพรเซสเซอร์แบบอสมวาร

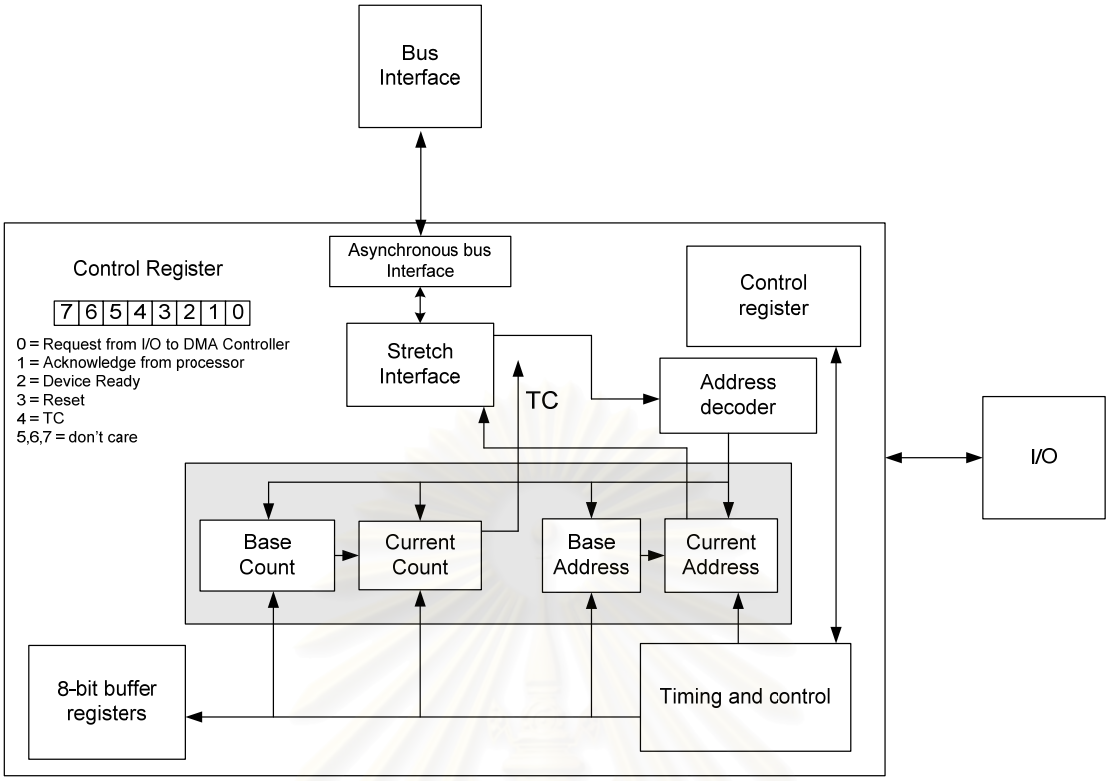
เนื่องจากไมโครโพรเซสเซอร์แบบอสมวารจะเป็นอุปกรณ์ที่ได้สิทธิ์การใช้บัสตลอดเวลา เว้นเสียแต่ว่ามีสัญญาณร้องขอจากดีเอ็มเอ จึงจะยินยอมให้ดีเอ็มเอเป็นอุปกรณ์มาสเตอร์ต่อไป เมื่อดีเอ็มเอต้องการใช้บัสแล้ว ดีเอ็มเอจะส่งสัญญาณร้องขอไปยังไมโครโพรเซสเซอร์แบบอสมวาร จากนั้นไมโครโพรเซสเซอร์แบบอสมวารจะยกสิทธิ์ ในการเป็นอุปกรณ์มาสเตอร์ให้กับดีเอ็มเอ ตามด้วยการโปรแกรมดีเอ็มเอ เพื่อให้ทำการรับส่งข้อมูลตามลักษณะของสัญญาณอินเตอร์รัพท์ที่เข้ามา เมื่อดีเอ็มเอถูกโปรแกรมโดยไมโครโพรเซสเซอร์แบบอสมวารแล้ว ภายในตัวดีเอ็มเอจะประกอบด้วยข้อมูลดังต่อไปนี้ คือ ฐานของเลขที่อยู่ (base address) และค่าของขนาดข้อมูลที่จะทำดีเอ็มเอ (Count Register)

4.2.1 **ฟังก์ชันการทำงานของดีเอ็มเอ**

เมื่อดีเอ็มเอรับสัญญาณร้องของจากอุปกรณ์ต่อพ่วง (*dma_req*) และได้สิทธิ์ในการใช้บัสแล้ว ดีเอ็มเอจะทำการตรวจสอบรีจิสเตอร์ภายใน เพื่อจัดการรับส่งข้อมูลผ่านบัส ยกตัวอย่างเช่น ในการทำงานขั้นตอนการอ่านของดีเอ็มเอ ดีเอ็มเอจะทำการอ่านข้อมูลจากต้นทาง ซึ่งเป็นอุปกรณ์ต่อพ่วง และส่งต่อมายังบัฟเฟอร์ภายใน ตามด้วยเขียนข้อมูลลงปลายทาง ซึ่งคือหน่วยความจำ เมื่อเสร็จสิ้นการรับส่งข้อมูล ดีเอ็มเอจะปล่อยการควบคุมบัส และส่งสัญญาณอินเตอร์รัพท์ เพื่อบอกการเสร็จสิ้นการทำงานกับไมโครโพรเซสเซอร์แบบอสมวาร

4.2.2 **สถาปัตยกรรมของดีเอ็มเอ**

สถาปัตยกรรมของดีเอ็มเอที่ออกแบบแสดงไว้ในรูปที่ 4.2 ประกอบด้วย รีจิสเตอร์ต่าง ๆ ส่วนเชื่อมต่อวงจรอสมวาร สำหรับการออกแบบดีเอ็มเอแบบสมวาร ซึ่งส่วนที่ซับซ้อนที่สุดคือส่วนควบคุมดีเอ็มเอ ซึ่งประกอบด้วยเครื่องจักรสถานะสำหรับดีเอ็มเอแบบสมวาร และกราฟบรรยายการเปลี่ยนสัญญาณสำหรับดีเอ็มเอแบบอสมวาร

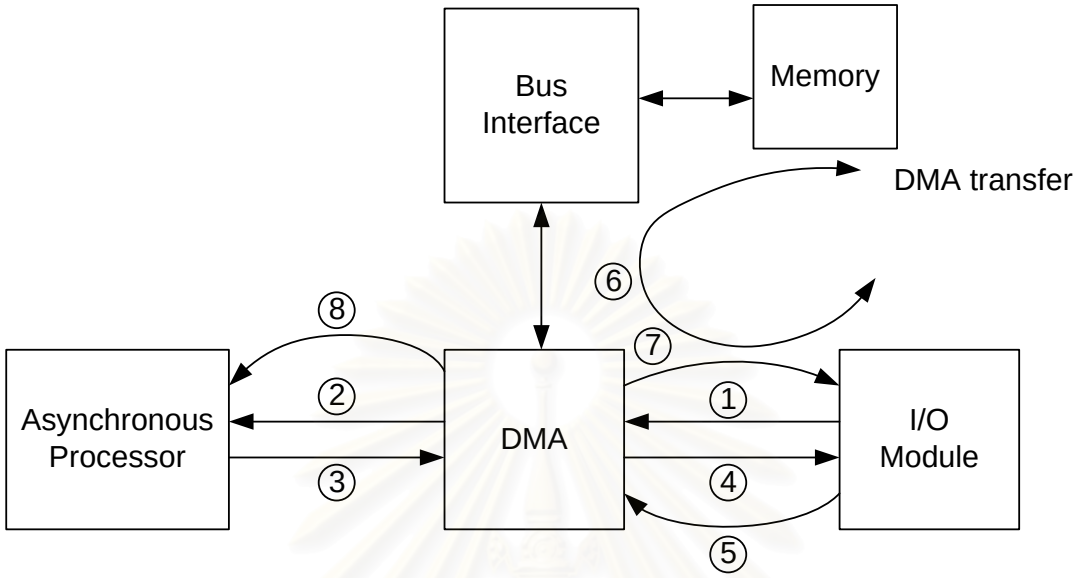


รูปที่ 4.2 สถาปัตยกรรมของดีเอ็มเอ

<u>รีจิสเตอร์ภายใน</u>

- Base Address Register
- Base Word Count Register
- Current Address Register
- Current Word Count Register
- Temporary Address Register
- Temporary Word Count Register
- Control Register

4.2.3 การทำงานของดีเอ็มเอ



รูปที่ 4.3 การทำงานของดีเอ็มเอ

ดีเอ็มเอที่ออกแบบทำงานตามขั้นตอนดังนี้

1. สัญญาณ DMA_req จากอุปกรณ์ต่อพ่วงเปลี่ยนจาก 0->1
2. สัญญาณ Hold จาก DMA เปลี่ยนจาก 0-> 1 ไปยังไมโครโพรเซสเซอร์
3. สัญญาณ Hold_ack จากไมโครโพรเซสเซอร์ เปลี่ยนจาก 0->1 ไปยังดีเอ็มเอ
   - ก่อนที่สัญญาณ Hold_ack จะเปลี่ยนจาก 0->1
     - ไมโครโพรเซสเซอร์จะโปรแกรมตำแหน่งเริ่มต้นไปยังดีเอ็มเอ (ST command)
     - ไมโครโพรเซสเซอร์จะโปรแกรมจำนวนข้อมูลไปยังดีเอ็มเอ (ST command)
4. สัญญาณ DMA_ack จากดีเอ็มเอ เปลี่ยนจาก 0->1 ไปยังอุปกรณ์ต่อพ่วง
5. สัญญาณ Device_ready จากอุปกรณ์ต่อพ่วงเปลี่ยนจาก 0->1 ไปยังดีเอ็มเอ
   - ก่อนที่สัญญาณ Device_ready จะเปลี่ยนจาก 0->1
     - สัญญาณการการอ่าน หรือ เขียน จะเปลี่ยนจาก 0->1 ตามชนิดการทำงาน (Read or Write)
6. เกิดการทำดีเอ็มเอ (read or write transaction occur)
7. สัญญาณ TC จากดีเอ็มเอเปลี่ยนจาก 0->1 เพื่อบอกถึงข้อมูลสุดท้าย
   - ก่อนที่สัญญาณ TC จะเปลี่ยนจาก 0->1

- สัญญาณก่อนหน้าทั้งหมด จะเปลี่ยนจาก 1->0

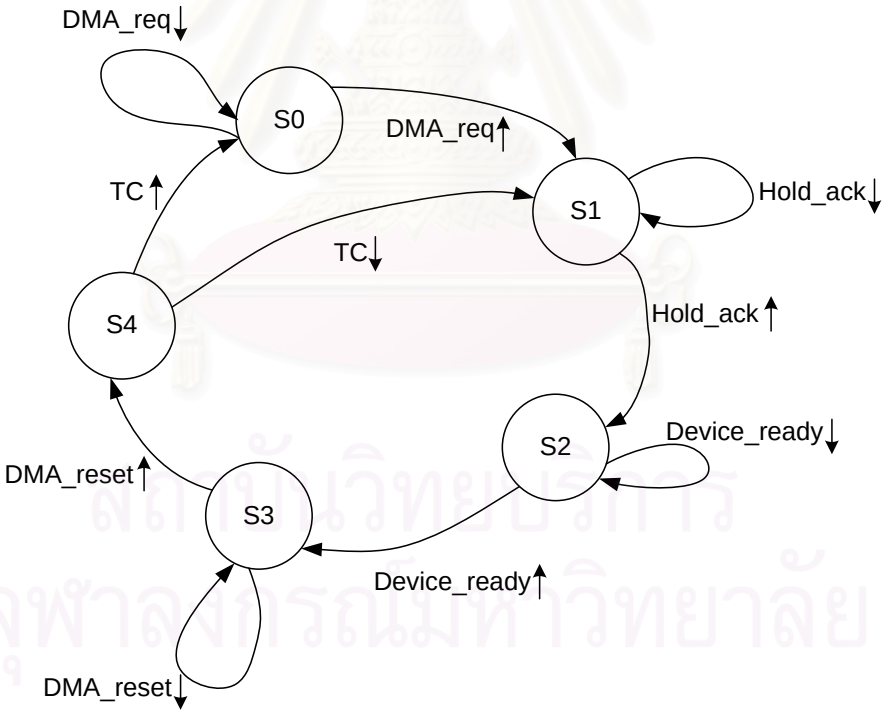
8. ดีเอ็มเอส่งสัญญาณอินเตอร์รัพท์ เพื่อบอกไมโครโพรเซสเซอร์ถึงการเสร็จสิ้นการทำงาน

4.3 การออกแบบดีเอ็มเอแบบสมวาร

ส่วนนี้แสดงการออกแบบส่วนควบคุมดีเอ็มเอแบบสมวาร โดยใช้เครื่องจักรสถานะ (State Machine) ดังแสดงในรูปที่ 4.3

คำอธิบายสัญญาณ

DMA_req	สัญญาณร้องขอจากอุปกรณ์ต่อพ่วง
Hold_ack	สัญญาณตอบรับจากไมโครโพรเซสเซอร์แบบอสมวาร
Device_ready	สัญญาณบอกสถานะของอุปกรณ์ต่อพ่วง
DMA_reset	รีเซ็ตสัญญาณการอ่านเขียนข้อมูล (ภายในดีเอ็มเอ)
TC	ตรวจสอบการนับค่าสุดท้าย



รูปที่ 4.4 เครื่องจักรสถานะของดีเอ็มเอแบบสมวาร

S0 – สถานะเริ่มต้น

S1 – รีจิสเตอร์ในดีเอ็มเอถูกโปรแกรมและส่งสัญญาณตอบรับไปยังอุปกรณ์ต่อพ่วง

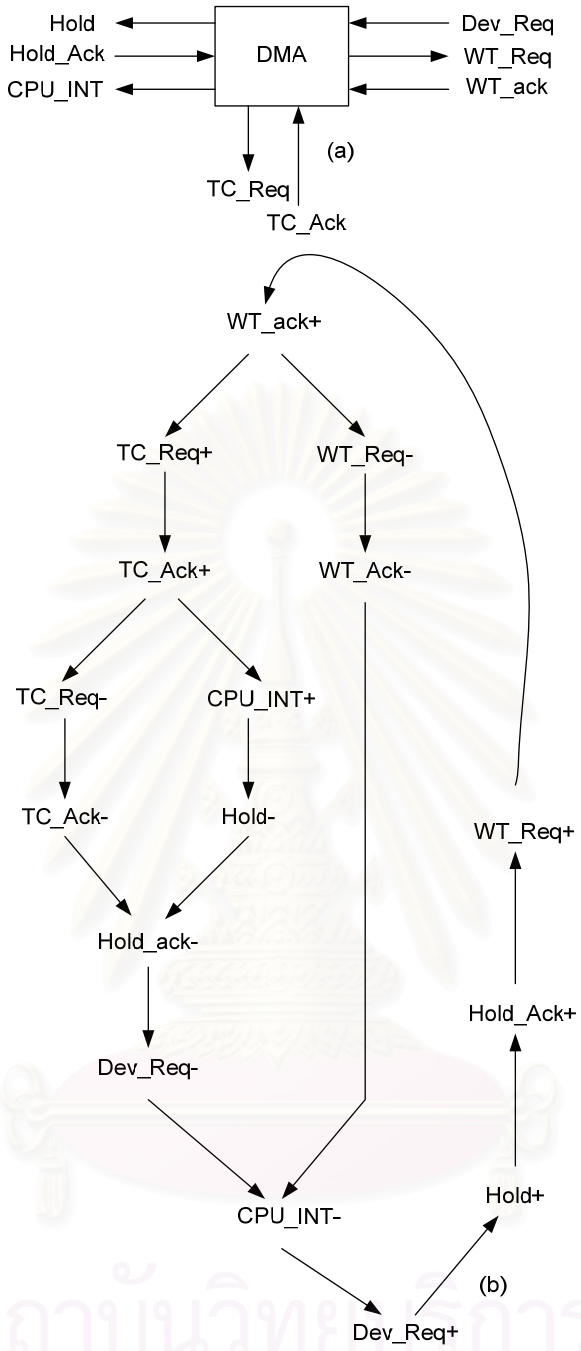
S2 – ตรวจสอบสถานะของอุปกรณ์ต่อพ่วง

S3 – รีเซ็ตสัญญาณอ่านเขียน

S4 – ตรวจสอบค่าสุดท้าย

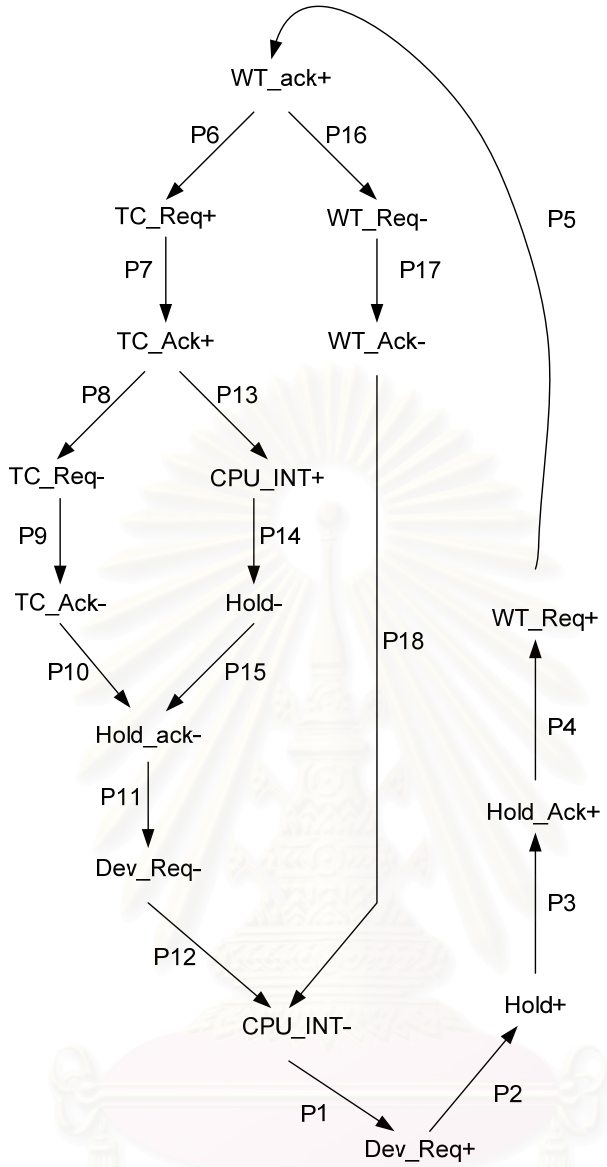
4.4 การออกแบบดีเอ็มเอแบบอสมวาร

การกำหนดสถานะสำหรับการออกแบบวงจรควบคุมโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ จะเกิดปัญหาการชนกันของวงจร เมื่อวงจรมีขนาดใหญ่ ทำให้เกิดความซับซ้อนในการออกแบบ จากการออกแบบดีเอ็มเอโดยใช้การกำหนดสถานะ ให้กับสัญญาณในวงจรแล้ว ปรากฏว่า State graph มีการชนกันมากกว่า 4 สถานะด้วยกัน ดังนั้นจึงต้องการวิธีการออกแบบวงจรโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณที่สามารถลดการชนกันของสถานะได้ดีระดับหนึ่ง ในการออกแบบดีเอ็มเอแบบอสมวาร ผู้วิจัยได้เลือกใช้ วีธีการเข้ารหัสแบบโครงสร้าง ซึ่งสามารถออกแบบวงจรควบคุมที่มีขนาดใหญ่ได้เป็นอย่างดี และยิ่งไปกว่านั้น การออกแบบวงจรควบคุมโดยใช้การเข้ารหัสแบบโครงสร้างสามารถจัดการการชนกันของสถานะ ที่กำหนดไว้ได้เป็นอย่างดี ซึ่งวิธีการออกแบบโดยการเข้ารหัสแบบโครงสร้าง จะทำการเปลี่ยนกราฟบรรยายการเปลี่ยนสัญญาณเป็น Petri Net ก่อน แล้วจะใช้เทคนิคการเข้ารหัสโดยใช้การเข้ารหัสแบบโครงสร้าง จากนั้นจะได้กราฟบรรยายการเปลี่ยนสัญญาณที่ถูกเข้ารหัส ขั้นตอนต่อไปก็จะทำการลดสัญญาณเพื่อให้ได้กราฟบรรยายการเปลี่ยนสัญลักษณ์ที่เล็กที่สุดที่ไม่ทำให้เกิดการชนกันของสถานะ ดังที่จะอธิบายต่อไป



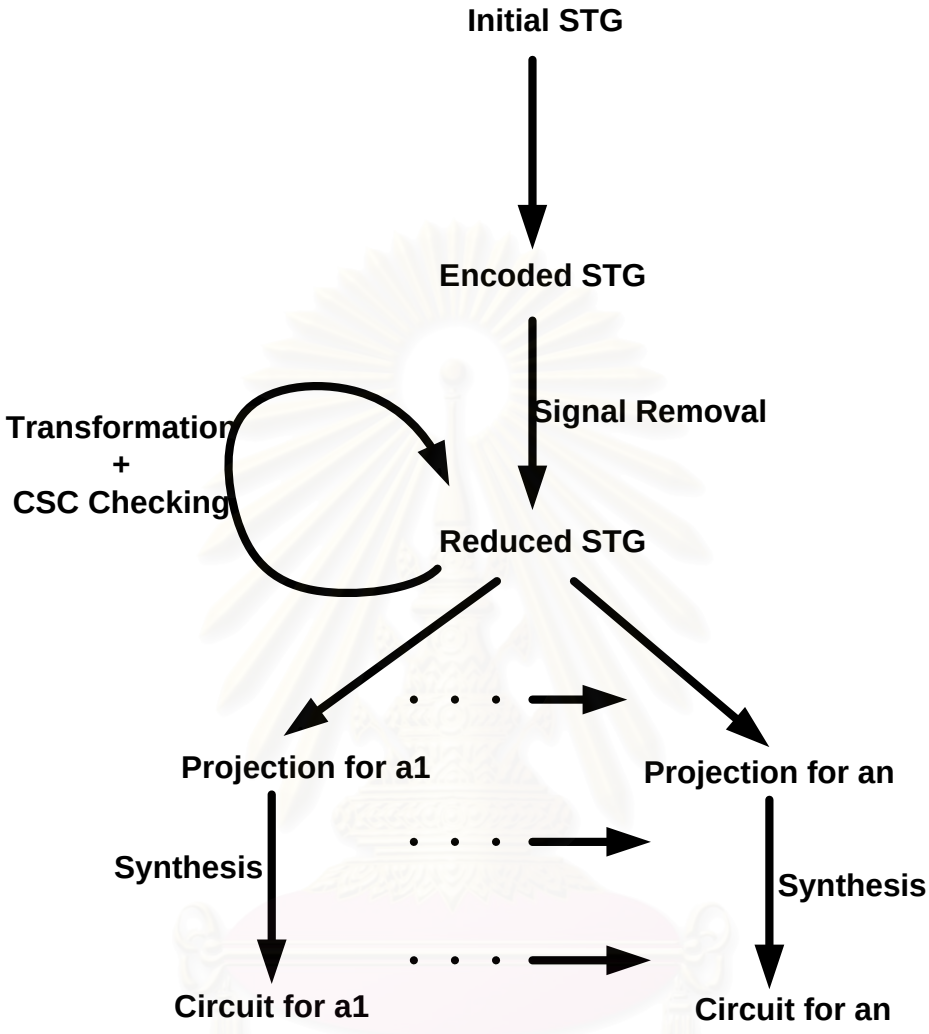
รูปที่ 4.5 ดีเอ็มเอแบบอสมวาร (a) การกำหนดคุณสมบัติ (b) กราฟบรรยายการเปลี่ยนสัญญาณ

ในการสังเคราะห์วงจรอสมวารจากการออกแบบโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ สามารถแบ่งออกเป็น 2 แบบด้วยกัน คือ ตรวจสอบการออกแบบ ตามด้วยหา Next State Function เพื่อให้ได้การกำหนดสถานะที่ไม่เกิดการชนกัน ซึ่งวิธีการออกแบบโดยใช้การเข้ารหัสแบบโครงสร้างจะเป็นวิธีที่ดี วิธีหนึ่ง ในการออกแบบเพื่อไม่ให้เกิดการชนกัน หลักการของการเข้ารหัสแบบโครงสร้างคือ เพิ่มสัญญาณภายในเพื่อรับประกันการชนกันของสถานะอย่างสมบูรณ์



รูปที่ 4.6 การเพิ่มเพลส

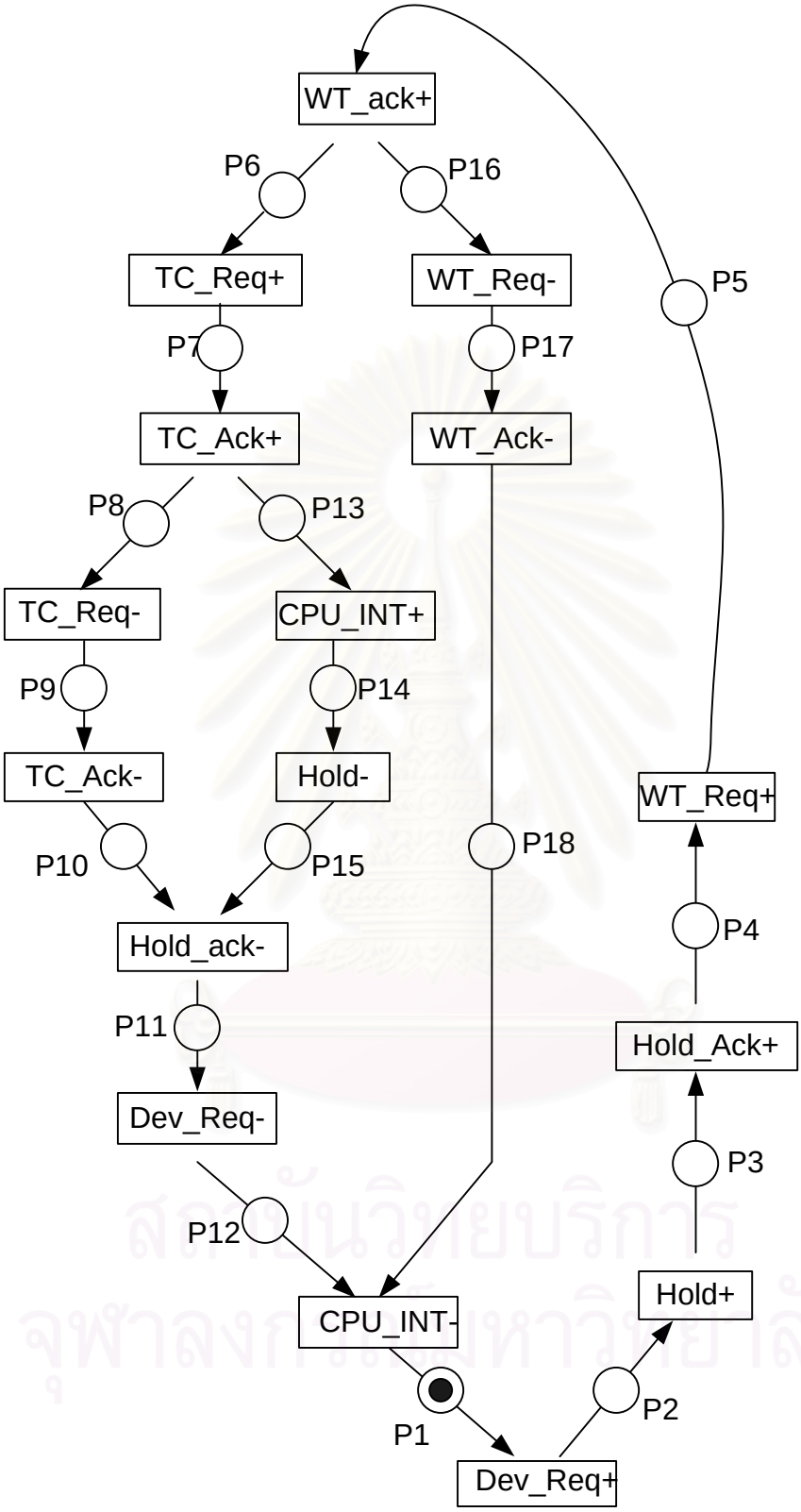
รูปที่ 4.6 แสดงการเตรียมกราฟบรรยายการเปลี่ยนสัญญาณ ที่มีการเพิ่ม เพลสเข้าไป เพื่อแปลงให้เป็น Petri Net กราฟบรรยายการเปลี่ยนสัญญาณจะมีการเข้ารหัสสถานะอย่างสมบูรณ์ เมื่อมีการกำหนดค่าไบนารี โค๊ด ที่ไม่ซ้ำซ้อนกันระหว่างสถานะที่ต่างกัน จากรูปจะเห็นการกำหนดคุณสมบัติของดีเอ็มเอแบบอสมวาร ที่ค่อนข้างจะซับซ้อน สังเกตการแยกกิ่งของกราฟบรรยายการเปลี่ยนสัญญาณออกไปหลายทาง เมื่อนำกราฟบรรยายการเปลี่ยนสัญญาณดังแสดงในรูปที่ 4.6 มาทำการกำหนดสถานะ จะได้สถานะที่ซ้ำซ้อนกัน ประมาณ 4 สถานะขึ้นไป ทำให้แก้ปัญหาการชนกันได้ยาก วิธีการออกแบบการเข้ารหัสแบบโครงสร้าง จะใช้ขั้นตอนการกำหนดสถานะในขั้นตอนสุดท้ายของการออกแบบ และรับประกันการชนกันของสัญญาณอย่างสมบูรณ์



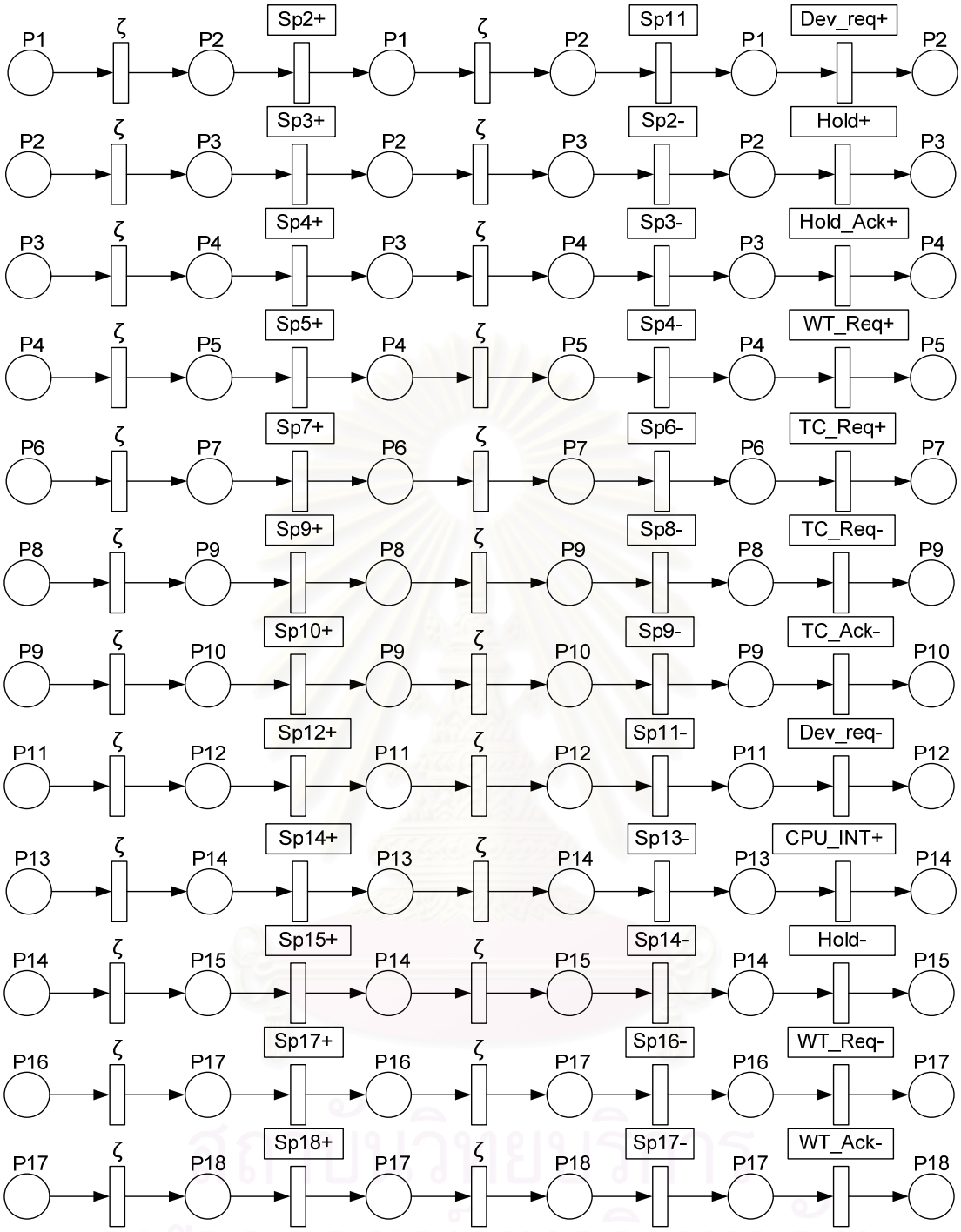
รูปที่ 4.7 ขั้นตอนการสังเคราะห์วงจร

ขั้นตอนการสังเคราะห์วงจรควบคุมแบบอสมวาร เริ่มต้นด้วยการเข้ารหัสกราฟบรรยายการเปลี่ยนสัญญาณตามขั้นตอนวิธีการเข้ารหัสที่แสดงไว้ในรูปที่ 4.8, 4.9 และ 4.10 จากนั้นจะได้กราฟบรรยายการเปลี่ยนสัญญาณที่ถูกเข้ารหัสแล้วในรูปที่ 4.10 รูปที่ 4.10 จะรับประกันการชนกันของสถานะโดยสมบูรณ์เมื่อมีการหา Next-State Function จะเห็นว่าสัญญาณที่เข้ารหัสมีมากเกินจำเป็น หากทำการหา Next-State Function ในขั้นตอนนี้แล้ว จะได้วงจรขนาดใหญ่สิ้นเปลืองโดยใช่เหตุ ดังนั้นขั้นตอนต่อไปคือ ลดสัญญาณทีละสัญญาณจนกว่า จะได้สัญญาณที่เล็กที่สุดและไม่ทำให้เกิดการชนกันในการกำหนดสถานะ รูปที่ 4.12 แสดงกราฟบรรยายการเปลี่ยน

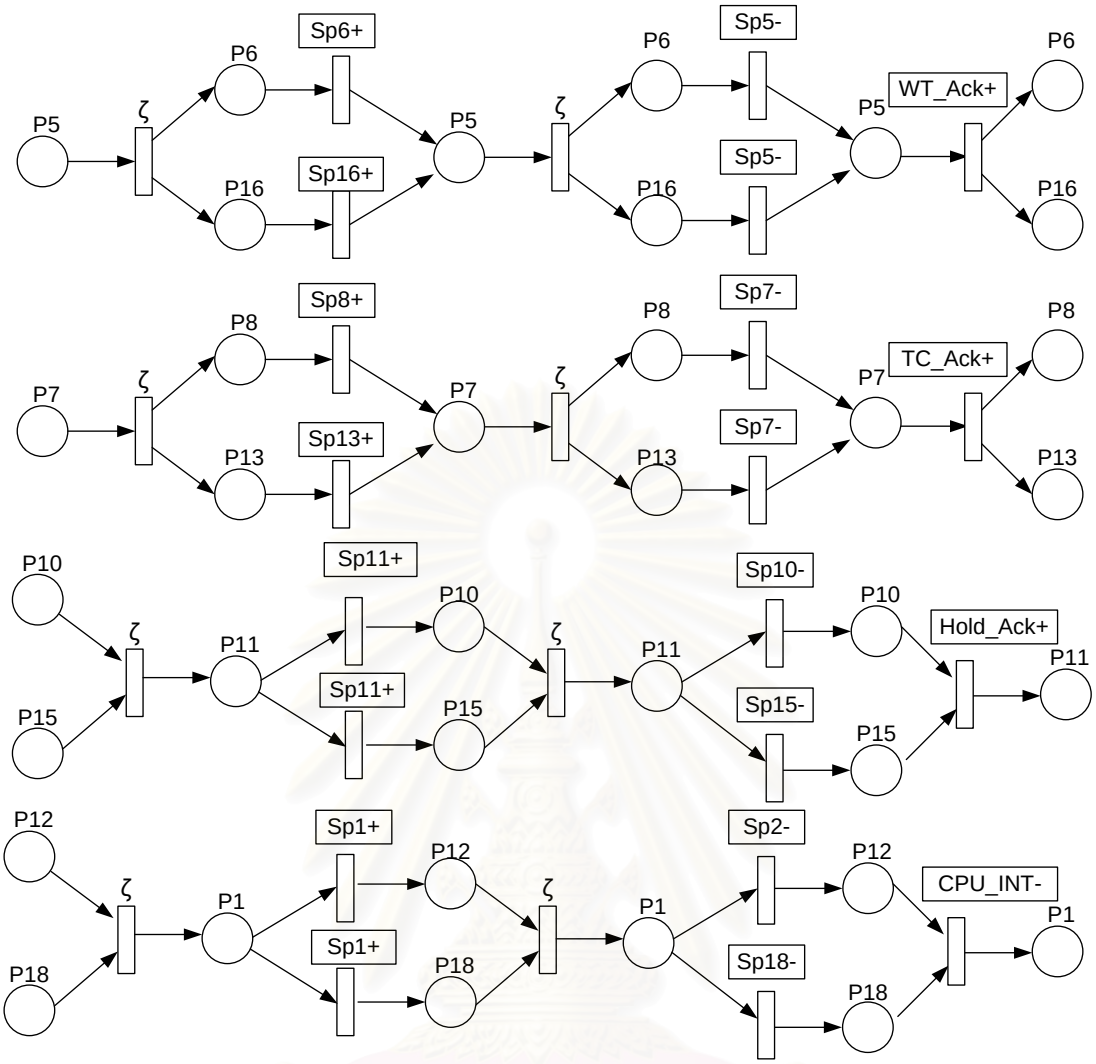
สัญญาณที่ถูกลดสัญญาณแล้ว จากรูปที่ 4.12 เมื่อได้กราฟบรรยายการเปลี่ยนสัญญาณที่ถูกลดสัญญาณแล้ว ขั้นตอนต่อไป ก็จะทำการ คำนวณเพื่อหา CSC Support ซึ่งสัญญาณที่จะถูกชี้คือทุกสัญญาณที่ไม่ใช่สัญญาณอินพุต สุดท้ายจะได้ CSC Support ที่ผ่านการคำนวณไว้สำหรับทุกสัญญาณที่ไม่ใช่สัญญาณอินพุต ดังแสดงไว้ในรูปที่ 4.13 จาก CSC Support ที่ได้ จึงนำไปหา Next-State Function ของทุกสัญญาณที่คำนวณออกไปอีกทีหนึ่ง แล้วจะได้วงจรของแต่ละสัญญาณในที่สุด



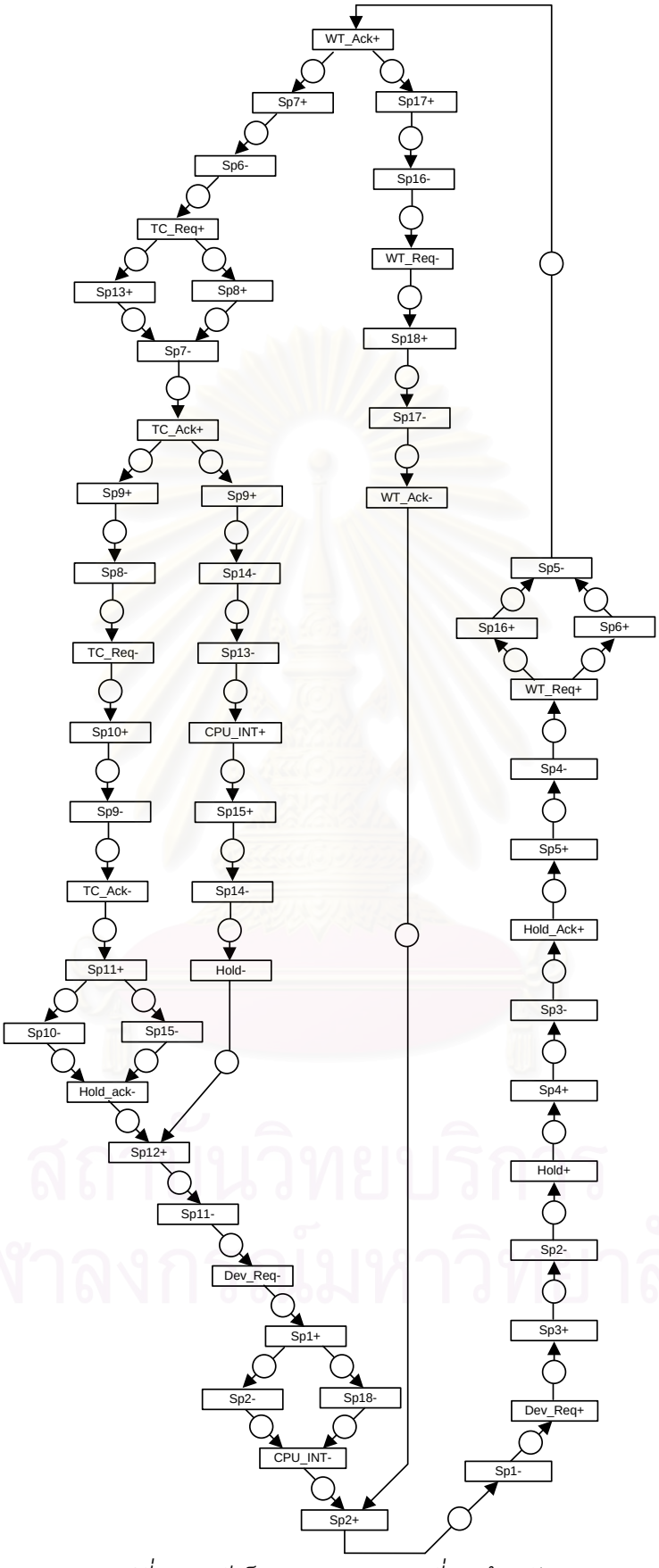
รูปที่ 4.8 การแปลงกราฟบรรยายการเปลี่ยนสัญญาณเป็น Petri-Net



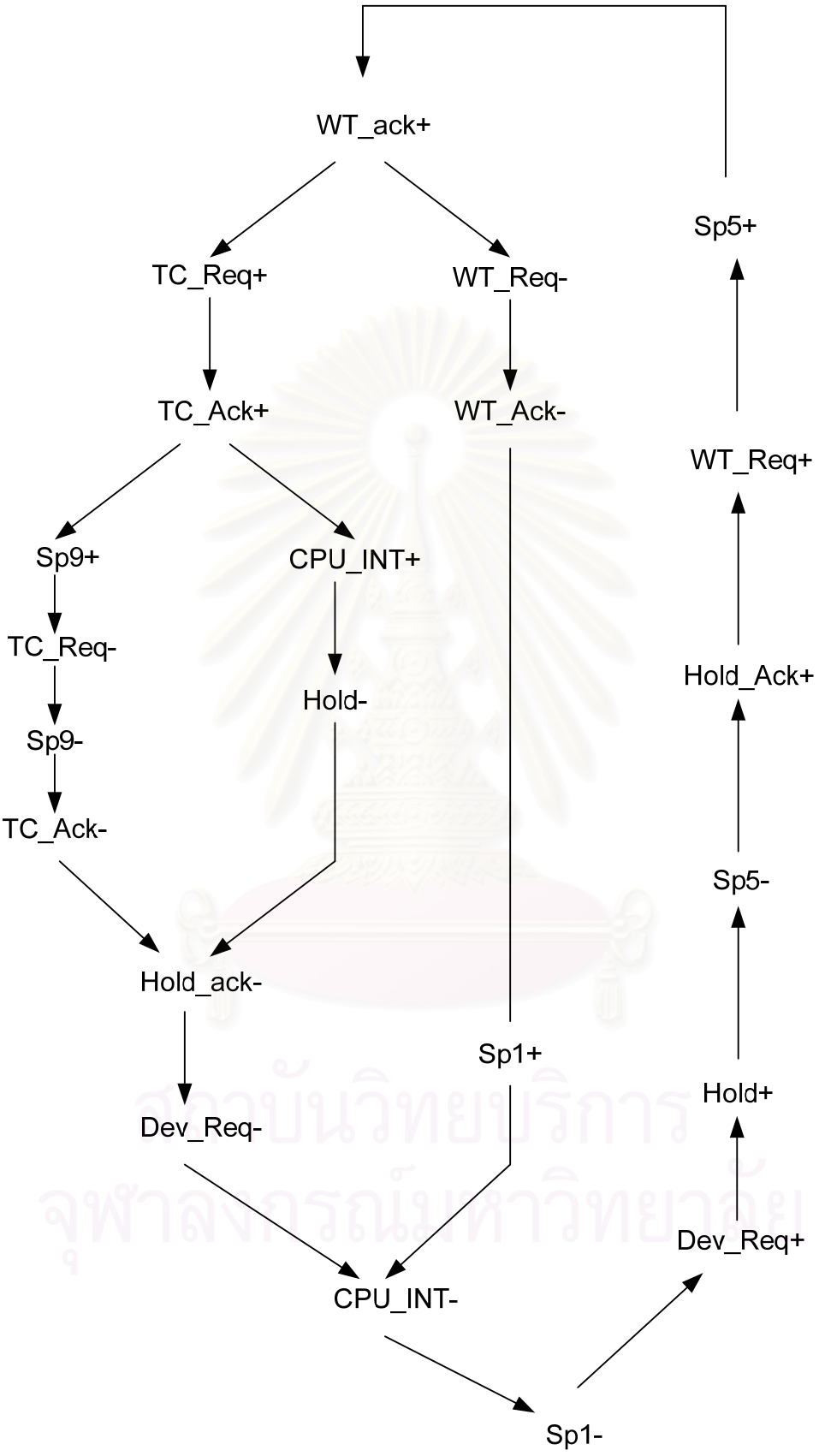
รูปที่ 4.9 วิธีการเข้ารหัส (ตรง)



รูปที่ 4.10 วิธีการเข้ารหัส (กิ่ง)



รูปที่ 4.11 ดีเอ็มเอแบบอสมวารที่ถูกเข้ารหัส



รูปที่ 4.12 กราฟบรรยายการเปลี่ยนสัญญาณที่ถูกลดสัญญาณ

ขั้นตอนวิธีการแก้ปัญหาการชนกันของสัญญาณ :

1. Find conflict (m1,m2) and traces m1→ m2 →m1

2. Find implicit places, s0 and s1 that break the conflict and guarantee consistency of a new signal s

For every transition t with arcs s0→ t → s1 do

Relabel the transition as (t:s+) or (s+;t)

For every transition t with arcs s1→ t → s0 do

Relabel the transition as (t;s-) or (s-,t)

Expand (series or parallel) the re-label transition

ขั้นตอนวิธีในการคำนวณ CSC support :

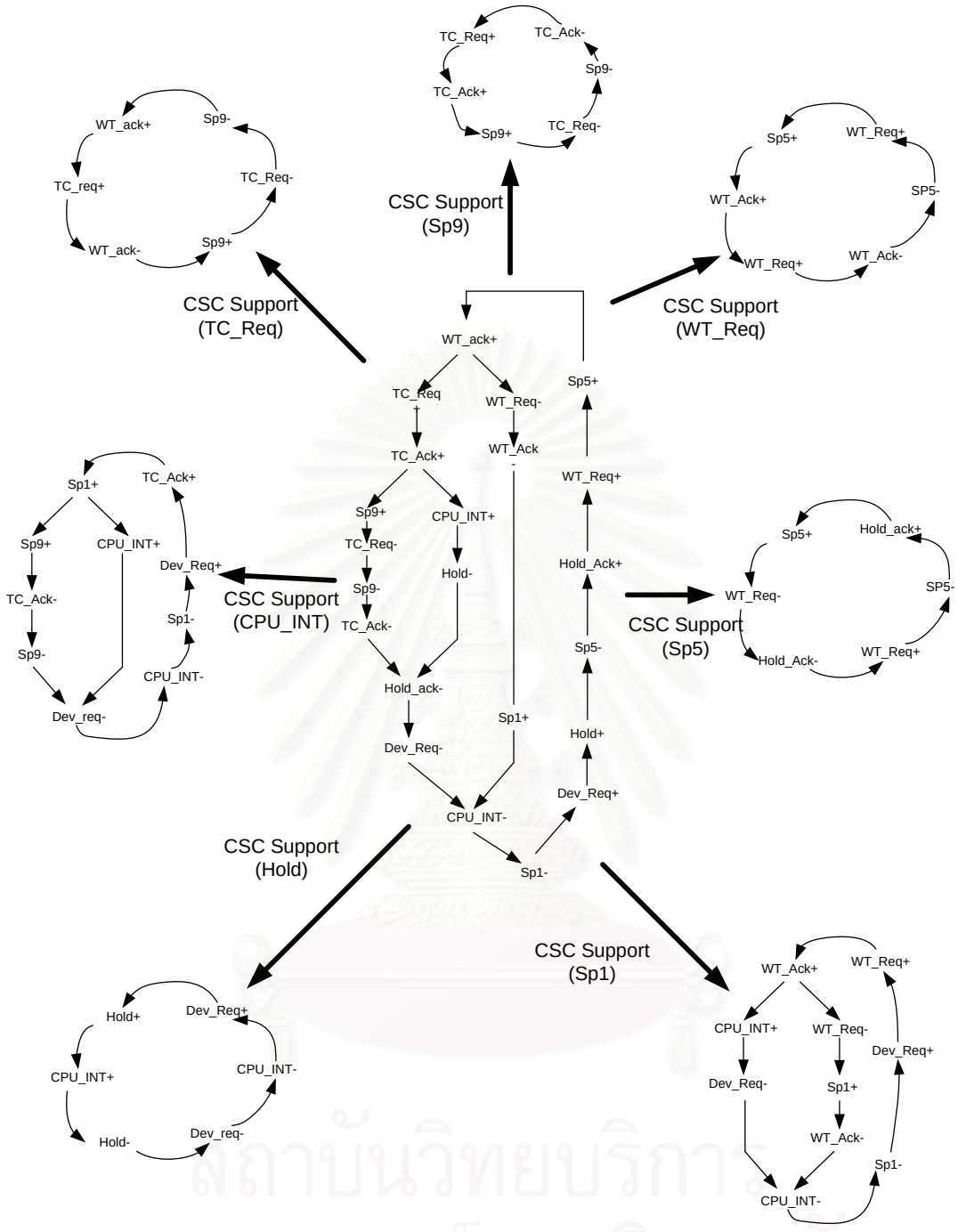
CSC Support (STG S, Signal a) return CSC support of a

CSC support a = Trigger (a) U {a}

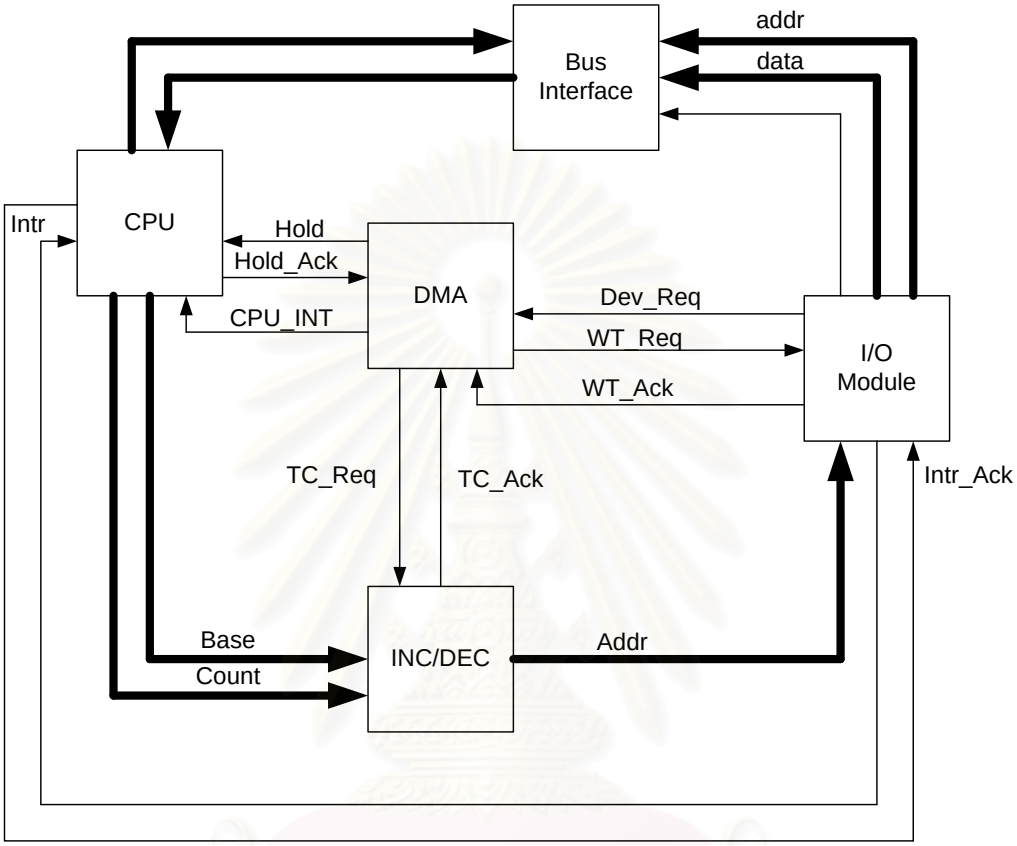
While it's still conflict do

Let b an unbalance signal in z

CSC Support for a = CSC Support for a U {b}



รูปที่ 4.13 การคำนาณ CSC Support สำหรับดีเอ็มเอแบบอสมวาร



รูปที่ 4.14 โครงสร้างการออกแบบดีเอ็มเอแบบอสมวาร

4.5 สรุป

บทนี้นำเสนอการออกแบบดีเอ็มเอ ซึ่งเป็นส่วนเสริมเพื่อเพิ่มประสิทธิภาพสำหรับการทำงานของบัสแบบอสมวาร ดีเอ็มเอที่ออกแบบจะออกแบบเป็นทั้งวงจรสมวารและอสมวาร ดีเอ็มเอแบบสมวารจะออกแบบโดยใช้เครื่องจักรสถานะ ส่วนดีเอ็มเอแบบอสมวารออกแบบโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ โดยใช้การเข้ารหัสแบบโครงสร้าง จากการออกแบบดีเอ็มแบบสมวารที่ออกแบบสามารถทำงานที่ความเร็วสัญญาณนาฬิกาสูงสุดที่ 116 เมกะเฮิร์ท ซึ่งจะมีความเร็วเทียบเคียงกับดีเอ็มแบบอสมวาร ที่ ประมาณ 40-45 ns ในการรับส่งข้อมูลจำนวน 1 ไบต์ อย่างไรก็ตามดีเอ็มแบบสมวารยังสามารถทำงานถูกต้องได้ที่ความเร็วสัญญาณนาฬิกา 149 เมกะเฮิร์ท

ในระดับการจำลองการทำงาน ทำให้การทำงานของดีเอ็มแบบสมวารเร็วกว่าดีเอ็มเอ็มเอแบบอสมวาร ที่ความเร็วสัญญาณนาฬิกามากกว่า 116 เมกะเฮิร์ท

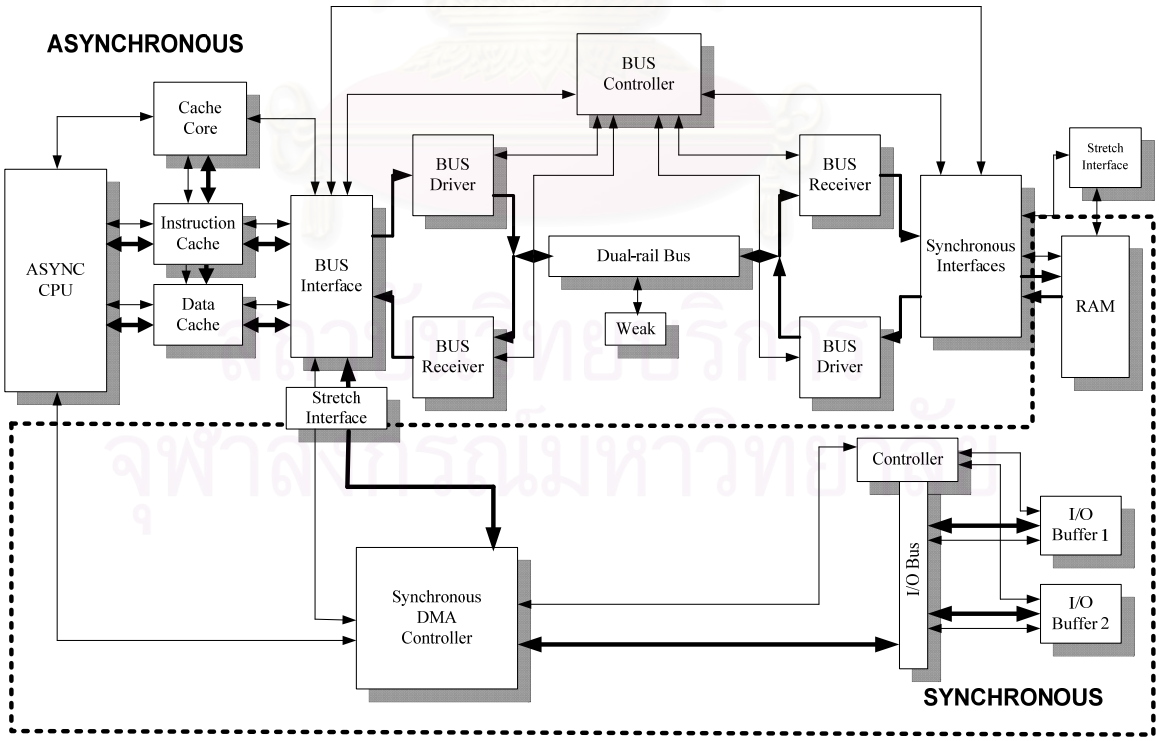
บทที่ 5

การรวมองค์ประกอบวงจรเข้ากับบัสระบบ

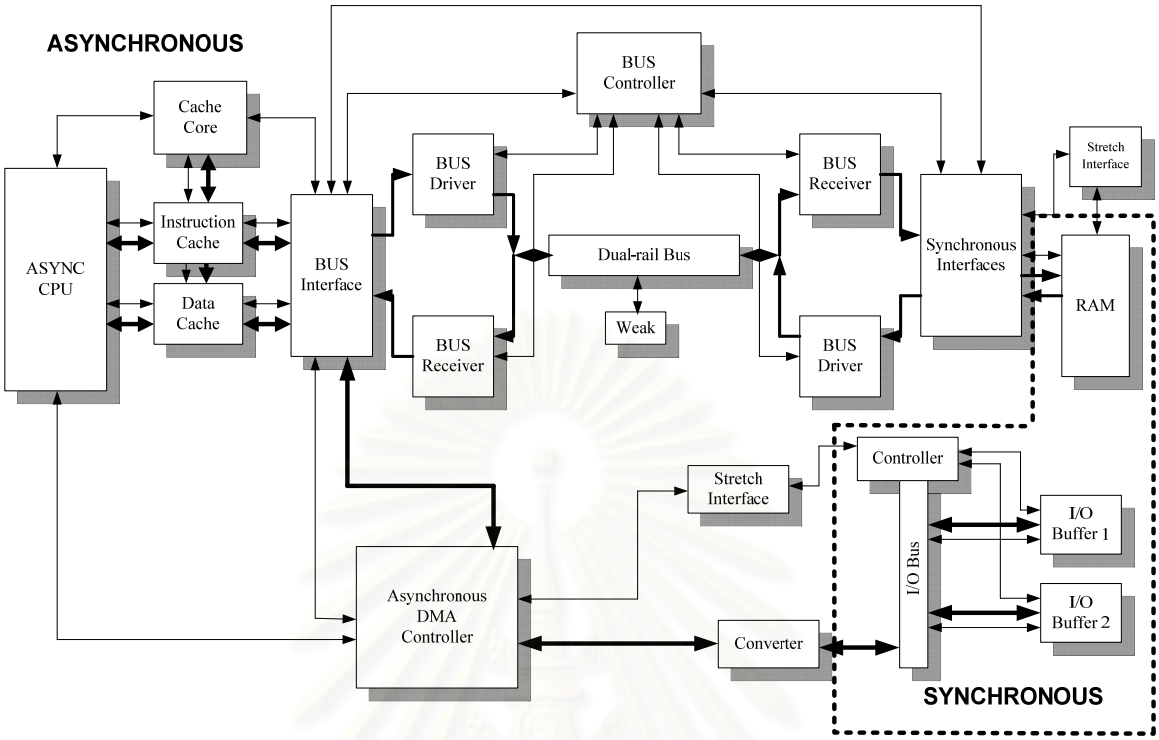
บทนี้นำเสนอการรวมวงจรทุกวงจรที่ออกแบบมารวมกันเป็นบัสระบบ รวมถึงการออกแบบวงจรเพิ่มเติมเพื่อให้ระบบทำงานอย่างสมบูรณ์ โดยนำเสนอระบบโดยรวมในหัวข้อที่ 5.1 ไมโครโพรเซสเซอร์แบบอสมวารที่นำมาใช้ในงานวิจัยนำเสนอในหัวข้อที่ 5.2 จากนั้นนำเสนอวิธีการปรับแต่งไมโครโพเซสเซอร์แบบอสมวารในหัวข้อที่ 5.3 นำเสนอการออกแบบแคชแบบอสมวารและส่วนเชื่อมต่ออุปกรณ์ต่อพ่วง ในหัวข้อที่ 5.4 และ หัวข้อที่ 5.5 ตามลำดับ จากนั้นแสดงการรวมวงจรทั้งหมดเป็นวงจรสมบูรณ์และแสดงการจำลองการทำงานในหัวข้อที่ 5.6 ตามด้วยสรุปในหัวข้อที่ 5.7

5.1 วงจรสมบูรณ์

เมื่อนำวงจรที่ออกแบบทั้งหมดมาประกอบเป็นวงจรเข้ากับบัสระบบแล้วจะได้วงจรสมบูรณ์ ดังแสดงไว้ในรูปที่ 5.1 และ 5.2 ซึ่งเป็นวงจรที่ประกอบด้วยดีเอ็มเอแบบสมวาร และวงจรสมบูรณ์ที่ประกอบด้วยดีเอ็มเอแบบอสมวารตามลำดับ



รูปที่ 5.1 วงจรสมบูรณ์ที่ประกอบด้วยดีเอ็มเอแบบสมวาร



รูปที่ 5.2 วงจรสมบูรณ์ที่ประกอบด้วยดีเอ็เอแบบอสมวาร

จากรูปที่ 5.1 และ 5.2 สายสัญญาณที่เชื่อมต่อกันระหว่างส่วนประกอบต่าง ๆ ประกอบด้วยสายสัญญาณข้อมูลและสายสัญญาณควบคุม สายสัญญาณข้อมูลจะแสดงเส้นทึบ ส่วนสายสัญญาณควบคุมจะแสดงด้วยเส้นบาง

5.2 ไมโครโพรเซสเซอร์แบบอสมวาร

ส่วนนี้นำเสนอภาพรวมของไมโครโพรเซสเซอร์แบบอสมวารที่นำมาใช้ในงานวิจัย โดยแสดงการออกแบบไมโครโพรเซสเซอร์ สถาปัตยกรรมไมโครโพรเซสเซอร์ การออกแบบรหัสดำเนินการ และการออกแบบส่วนควบคุม จากนั้นนำเสนอการปรับแต่งไมโครโพรเซสเซอร์แบบอสมวารเพื่อนำมาใช้กับวงจรที่ออกแบบในหัวข้อถัดไป

ไมโครโพรเซสเซอร์แบบอสมวารที่นำมาใช้ในงานวิจัยประกอบด้วยชุดคำสั่ง (Instruction Set) ที่ใช้งานแบ่งออกได้เป็นสามกลุ่มคือ กลุ่มคำสั่ง Data Transfer Operation, กลุ่มคำสั่ง Program and Machine Control Operation และกลุ่มคำสั่ง Arithmetic and Logic Operation

5.2.1 สถาปัตยกรรมไมโครโพรเซสเซอร์

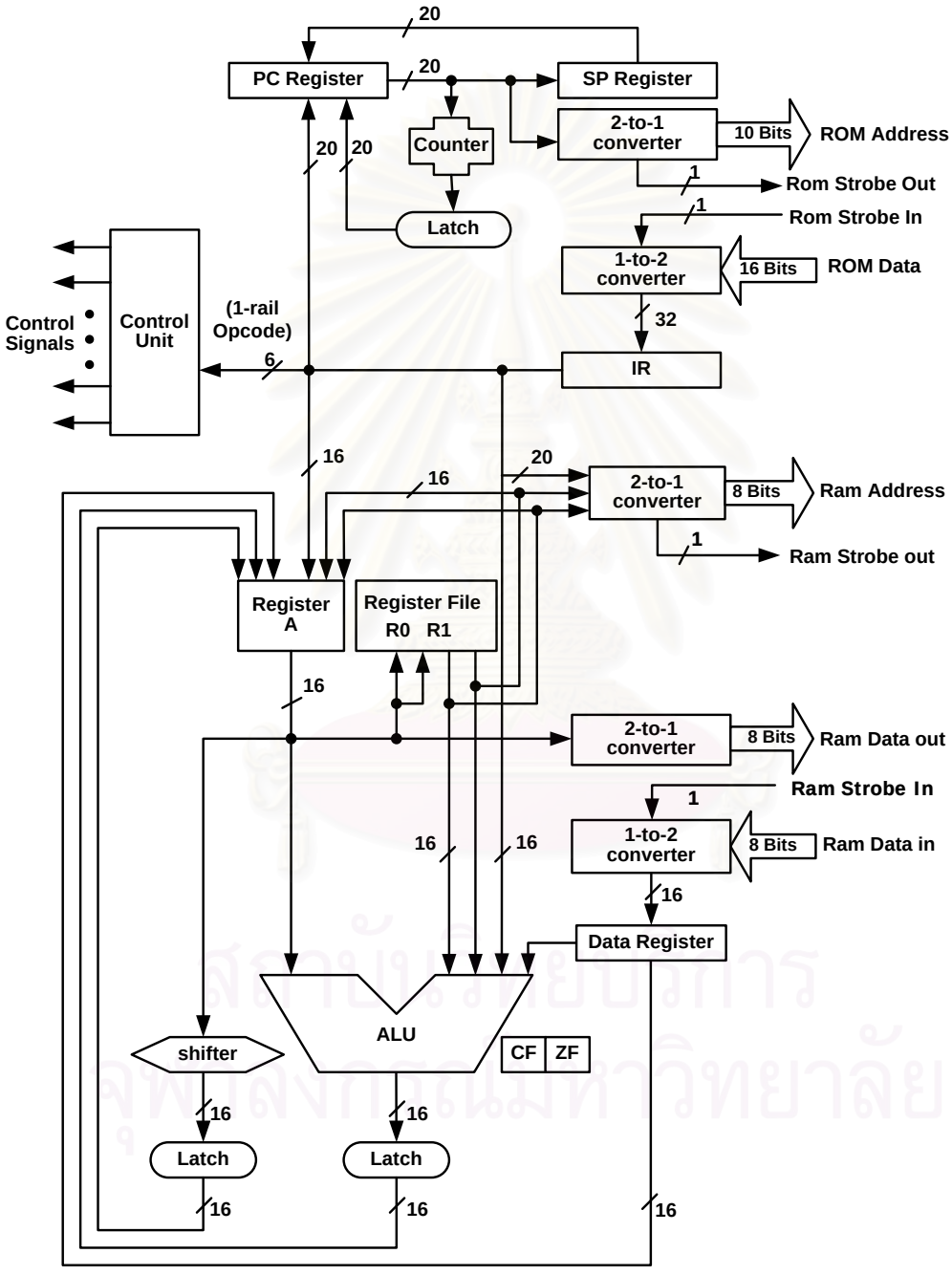
สถาปัตยกรรมของไมโครโปรเซสเซอร์ออกแบบบนเอฟพีจีเอแสดงดังรูปที่ 5.1 โดยมีรีจิสเตอร์ที่ใช้ในการประมวลผลประกอบด้วยรีจิสเตอร์ตัวสะสม (Accumulator) ขนาด 8 บิต เรียกว่ารีจิสเตอร์ A และรีจิสเตอร์ทั่วไป (General Register) ขนาด 8 บิตจำนวนสองตัวเรียกว่า รีจิสเตอร์ R0 และ R1 สำหรับรีจิสเตอร์เฉพาะซึ่งอยู่ภายในไมโครโปรเซสเซอร์ประกอบด้วยตัวนับระบุตำแหน่งคำสั่ง (Program Counter) เรียกว่า รีจิสเตอร์ PC, รีจิสเตอร์ที่ใช้เก็บค่า PC เมื่อมีการทำคำสั่ง CALL เรียกว่า รีจิสเตอร์ SP และรีจิสเตอร์ที่ใช้เก็บรหัสดำเนินการของคำสั่งที่อ่านเข้ามาเรียกว่า รีจิสเตอร์ IR โดยมีสัญญาณแสดงสถานะ (Status Flag) สองสัญญาณคือ สัญญาณทดค่า (Carry Flag) และสัญญาณค่าศูนย์ (Zero Flag)

ในส่วนของวงจรที่ทำหน้าที่ประมวลผลจะประกอบด้วยวงจรเลื่อนข้อมูล (Shfiter) ซึ่งจะทำการเลื่อนค่าของรีจิสเตอร์ A ไปทางซ้ายหรือขวา 1 บิต กับหน่วยคำนวณและประมวลผลตรรกะที่มีแบบจำลองความหน่วงที่ไม่ไวต่อความหน่วงชนิดปรับมาตราส่วนได้ ซึ่งทำงานแบบ Accumulator Based คือ ทำการคำนวณระหว่างรีจิสเตอร์ A กับค่าของตัวถูกดำเนินการ (Operand) แล้วเก็บค่าผลลัพธ์ที่ได้ลงรีจิสเตอร์ A โดยสามารถอ้างอิงค่าของตัวถูกดำเนินการได้สี่แบบคือ แบบค่าคงที่ (Immediate Mode), แบบรีจิสเตอร์ (Register Mode), แบบอ้างอิงหน่วยความจำโดยตรง (Direct Memory Addressing Mode) และแบบอ้างอิงหน่วยความจำโดยอ้อมด้วยรีจิสเตอร์ (Indirect Memory Addressing with Register Indexing Mode)

เนื่องจากการคำนวณและประมวลผลเป็นการทำงานแบบ Accumulator Based ซึ่งจะเขียนผลลัพธ์ที่ได้ไปยังรีจิสเตอร์ A หากค่าในรีจิสเตอร์ A มีการเปลี่ยนแปลง ก็จะทำให้เกิดการประมวลผลใหม่อีกครั้งทันที และทำให้เกิดการทำงานวนซ้ำไปเรื่อยๆ จึงต้องมีรีจิสเตอร์ซึ่งเรียกว่า Latch ทำหน้าที่เก็บผลลัพธ์ของการประมวลผลไว้ชั่วคราวก่อน โดยรีจิสเตอร์ Latch นี้จะถูกนำไปใช้ในการเก็บตัวดำเนินการที่อ่านมาจากหน่วยความจำไว้ชั่วคราว เพื่อใช้ในการประมวลผลต่อไปด้วยเช่นกัน

ไมโครโปรเซสเซอร์ที่สามารถอ้างอิงหน่วยความจำสำหรับโปรแกรม (Programmed Memory) ซึ่งใช้เก็บรหัสดำเนินการขนาด 16 บิต ได้ 1 กิโลตำแหน่ง (1K * 16 bits) และหน่วยความจำสำหรับข้อมูล (Data Memory) ที่มีขนาด 8 บิต ได้ 1 กิโลตำแหน่ง (1K * 8 bits) เนื่องจากการออกแบบวงจรแบบอสมวาร จะใช้สายสัญญาณสองเส้นแทนสัญญาณข้อมูลขนาด 1 บิต การอ้างอิงหน่วยความจำที่มีอยู่ในปัจจุบัน จึงต้องมีวงจรที่แปลงสายสัญญาณ 2 เส้นให้เป็น 1

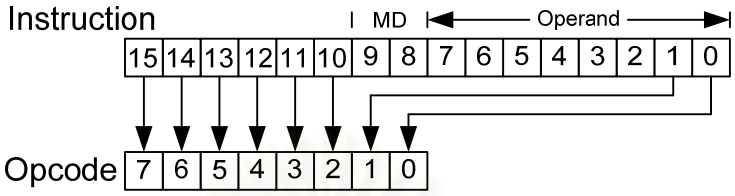
เส้นเพื่อส่งออกไปยังหน่วยความจำ และวงจรที่แปลงสายสัญญาณที่รับมาจากหน่วยความจำจาก 1 เส้นให้เป็น 2 เส้น



รูปที่ 5.3 สถาปัตยกรรมของไมโครโพรเซสเซอร์แบบอสมวาร

5.2.2 **รหัสดำเนินการ**

ไมโครโพรเซสเซอร์แบบอสมวารประกอบด้วยรหัสดำเนินการยาว 16 บิต โดยแบ่งเป็น opcode และ operand ดังแสดงในรูปที่ 5.4



รูปที่ 5.4 การออกแบบรหัสดำเนินการ

MD คือ การอ้างหน่วยความจำ

โดยสามารถอ้างอิงค่าของตัวถูกดำเนินการได้สี่แบบคือ แบบค่าคงที่ (Immediate Mode), แบบรีจิสเตอร์ (Register Mode), แบบอ้างอิงหน่วยความจำโดยตรง (Direct Memory Addressing Mode) และแบบอ้างอิงหน่วยความจำโดยอ้อมด้วยรีจิสเตอร์ (Indirect Memory Addressing with Register Indexing Mode)

00 -> แบบค่าคงที่

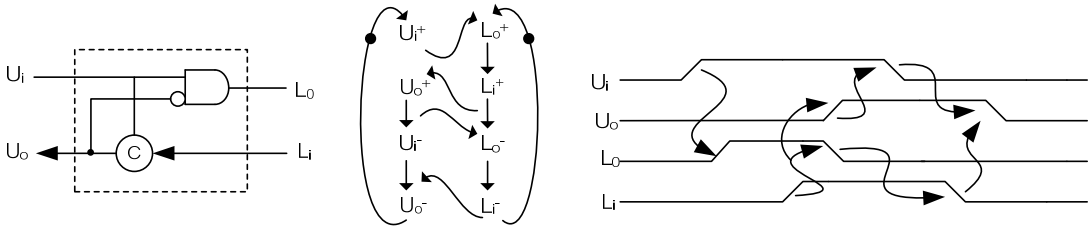
01 -> แบบรีจิสเตอร์

10 -> แบบอ้างอิงหน่วยความจำโดยตรง

11 -> แบบอ้างอิงหน่วยความจำแบบอ้อมโดยรีจิสเตอร์

5.2.3 **ส่วนควบคุม**

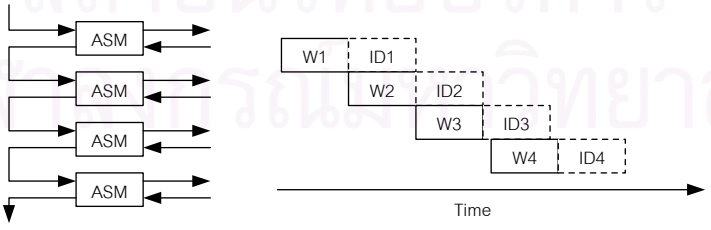
การทำงานของวงจรภายในไมโครโปรเซสเซอร์เป็นแบบ 2-rail 2-phase Return to Zero คือมีการทำงานในขั้นทำงาน และขั้นว่างสลับกันไปเรื่อยๆ จึงใช้อุปกรณ์ออโต้สวีป (Autosweeping Module) ซึ่งมีโครงสร้างของวงจร รวมทั้งรูปแบบการเปลี่ยนระดับสัญญาณร้องขอและสัญญาณตอบรับดังรูปที่ 5.3 ในการควบคุม และสร้างสัญญาณร้องขอกับสัญญาณตอบรับ



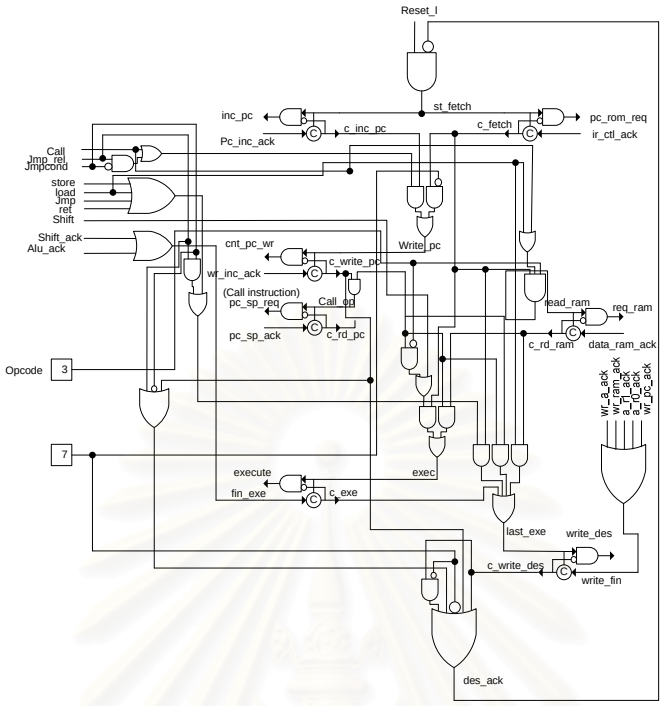
รูปที่ 5.5 อุปกรณ์ออโต้สวีป

จากรูปเมื่อสัญญาณร้องขอของการทำงานก่อนหน้า (U_i) มีการเปลี่ยนระดับสัญญาณจาก 0 เป็น 1 ASM จะเปลี่ยนระดับสัญญาณร้องขอของการทำงานถัดไป (L_o) จาก 0 เป็น 1 เพื่อเข้าสู่ขั้นทำงาน หลังจากที่การทำงานถัดไปเสร็จสิ้นแล้ว จึงส่งสัญญาณตอบรับ (L_i) กลับมายัง ASM ซึ่งจะส่งผลให้สัญญาณตอบรับของการทำงานก่อนหน้า (U_o) เกิดการเปลี่ยนระดับสัญญาณจาก 0 เป็น 1 พร้อมกับที่สัญญาณร้องขอของการทำงานถัดไป เกิดการเปลี่ยนระดับสัญญาณจาก 1 เป็น 0 เพื่อเข้าสู่ขั้นว่างได้เลย นั่นคือ เมื่อเสร็จสิ้นการทำงานในขั้นทำงานแล้ว สัญญาณร้องขอของการทำงานถัดไปจะเกิดการเปลี่ยนระดับสัญญาณจาก 1 เป็น 0 ได้เองทันที ดังนั้นเมื่อนำวงจร ASM มาใช้ในการออกแบบวงจรควบคุม (Controller) จะทำให้สามารถทำงานในขั้นว่างของการทำงานก่อนหน้า ขนานไปพร้อมๆ กับทำงานในขั้นทำงานของการทำงานถัดไปดังรูปที่ 5.5

ในการออกแบบส่วนวงจรควบคุม จะเริ่มจากการออกแบบโครงสร้างการทำงานของแต่ละคำสั่งในไมโครโพรเซสเซอร์ แล้วจึงลดรูปเพื่อให้เห็นลำดับขั้นตอนการถอดรหัสคำสั่ง ซึ่งทำให้ง่ายต่อการออกแบบส่วนถอดรหัสคำสั่ง (Instruction Decoder)



รูปที่ 5.6 การทำงานของอุปกรณ์ออโต้สวีป



External signal (Input)

- reset_l
- opcode
- carry_out
- carry_out_l
- zero
- zero_l
- ir_ctl_ack
- wr_a_ack
- wr_pc_ack
- wr_inc_ack
- pc_inc_ack
- a_r0_ack
- a_r1_ack
- data_ram_ack
- pc_sp_ack
- wr_ram_ack
- alu_ack
- shift_ack

External signal (output)

- Opcode (7 downto 4) and execute → add, sub, orr, andd, xorr
- write_des and Cal → out_to_a
- st_fetch and not c_fetch → pc_rom_req
- inc_pc → pc_inc_req
- call_op and not c_rd_pc → pc_sp_req
- cxecute and Cal → a_alu_req
- **write_des and req_a_ram** → a_ram_req
- write_des and req_a_r0 → a_r0_req
- write_des and req_a_r1 → a_r1_req
- execute and shift → a_shf_req
- write_des and shift → shf_a_req
- execute and shift and not opcode (3) → do_shl
- execute and shift and opcode (3) → do_shr
- cnt_pc_wr → send_to_pc
- execute and reg_mode and opcode(0) → r0_alu_req
- write_des and req_r0_a → r0_a_req
- **req_r0_ram and (req_ram or (write_des and store))** → r0_ram_req
- **req_r1_ram and (req_ram or (write_des and store))** → r1_ram_req
- execute and opcode(3) → data_alu_req
- write_des and load and opcode(3) → data_a_req
- write_des and load and imm_mode → ir_a_req
- **write_des and (intr or call or jump or (jmp_rel and jmpcond)** → ir_pc_req
- **drct_mode and (req_ram or (write_des and store))** → ir_ram_req
- **req_ram** → ram_read
- execute and reg_mode and opcode(1) → r1_alu_req
- write_des and req_r1_a → r1_a_req
- write_des and ret → sp_pc_req
- execute and imm_mode → ir_alu_req

รูปที่ 5.7 ส่วนควบคุม

รูปที่ 5.7 แสดงสถาปัตยกรรมส่วนควบคุมซึ่งออกแบบโดยอุปกรณ์ออโต้สวีป ส่วนควบคุมแบ่งออกเป็น 5 กระบวนการด้วยกัน คือ fetch instruction, decode instruction, fetch operand, execution, and write back

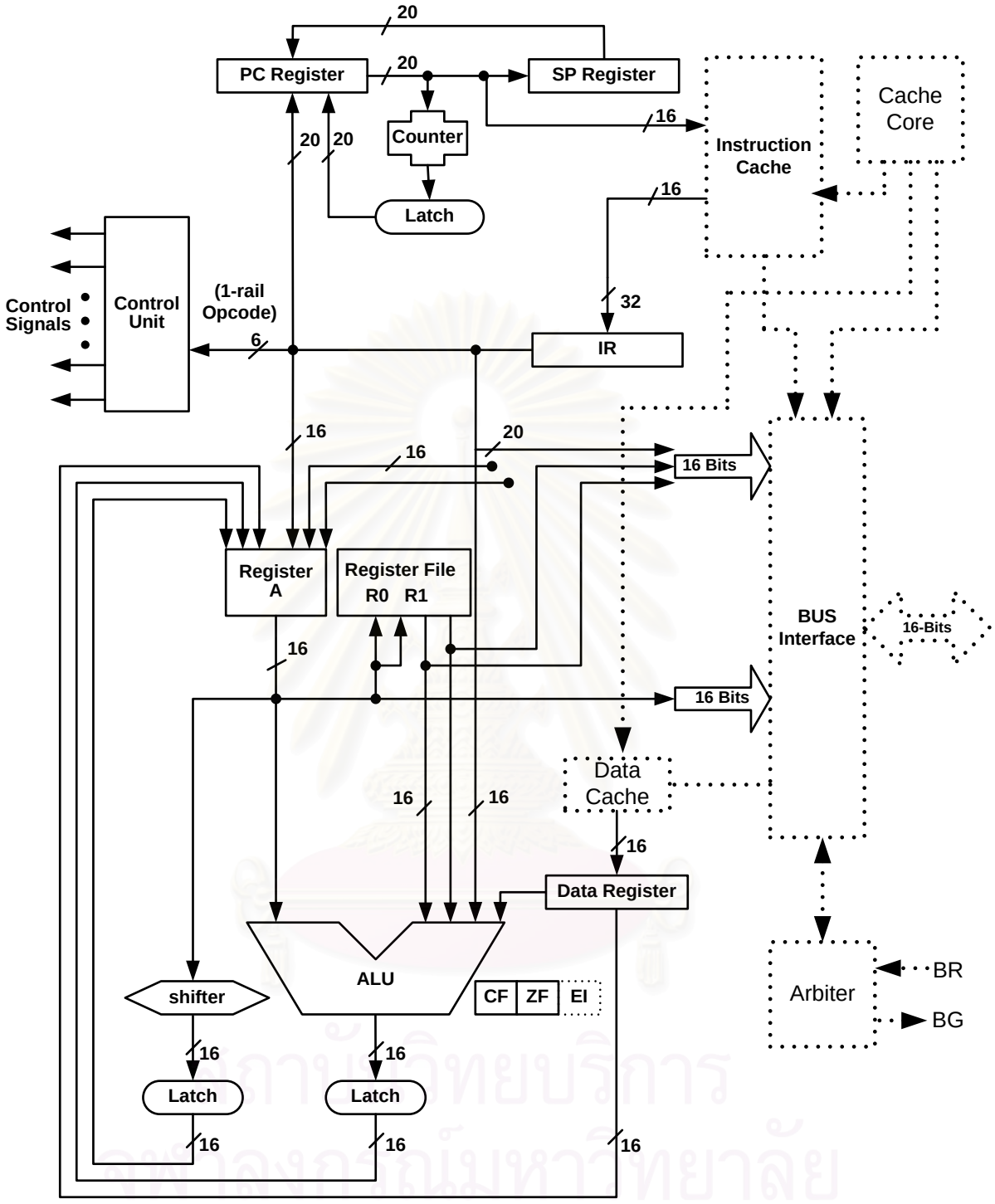
5.3 การปรับแต่งไมโครโพรเซสเซอร์

ส่วนนี้นำเสนอการปรับแต่งไมโครโพรเซสเซอร์แบบอสมวารที่ได้นำเสนอในหัวข้อที่ 5.1 จุดประสงค์ในการปรับแต่งคือ สามารถเชื่อมต่อกับบัสแบบอสมวาร และดีเอ็มเอได้ ไมโครโพรเซสเซอร์แบบอสมวารถูกปรับแต่ง 3 จุดใหญ่ ๆ ดังนี้

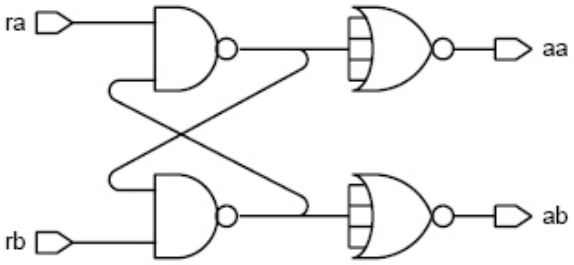
- การปรับแต่งสถาปัตยกรรม
- การปรับแต่งรหัสดำเนินการ
- การปรับปแต่งส่วนควบคุม

5.3.1 การปรับแต่งสถาปัตยกรรม

การปรับแต่งสถาปัตยกรรม ไมโครโพรเซสเซอร์แบบอสมวารถูกปรับแต่งสถาปัตยกรรมเพื่อต้องการเชื่อมต่อกับบัสแบบอสมวาร และดีเอ็มเอ โดยการตัดส่วนเชื่อมต่อหน่วยความจำและสร้างส่วนเชื่อมต่อใหม่ โดยให้ไปเชื่อมต่อกับแคชแบบอสมวารแทน ตามด้วยส่วนเชื่อมต่อบัส จากนั้นได้เพิ่ม Flag บิต ชื่อ EI Flag เพื่อจัดการ การทำงานของอินเตอร์รัพท์ EI flag เปลี่ยนจาก 0->1 เมื่อส่วนควบคุมเสร็จสิ้นการทำงานต่อ 1 รหัสดำเนินการ บ่งบอกถึงการเปิดทางให้กับสัญญาณอินเตอร์รัพท์ นอกจากนั้น EI flag จะเป็น 0 ส่วนที่ถูกเพิ่มเติมอีกคือ ตัวตัดสินใจ (Arbiter) เพิ่มเพื่อรับสัญญาณร้องขอบัสจากดีเอ็มเอ เพื่อใช้งานบัส โดยออกแบบเพื่อให้ทำงานกับโปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น ดังแสดงในรูปที่ 5.8 และ 5.9



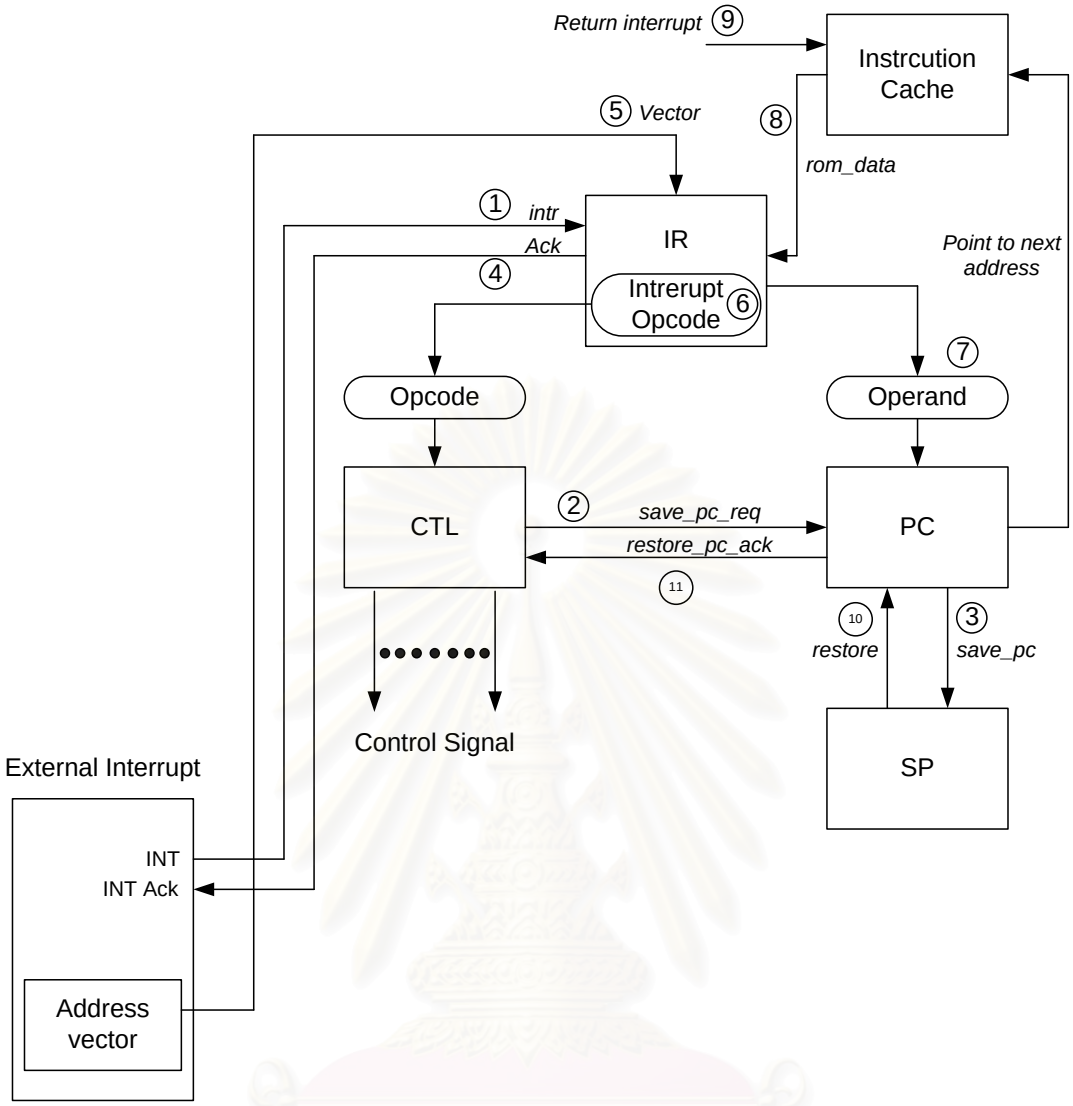
รูปที่ 5.8 การปรับแต่งสถาปัตยกรรมไมโครโพรเซสเซอร์



รูปที่ 5.9 การออกแบบตัวตัดสินใจ

5.3.2 การปรับแต่งรหัสดำเนินการ

รหัสดำเนินการสำหรับไมโครโพรเซสเซอร์แบบอสมวาร มีขนาดความกว้าง 16 บิต โดยการโหลดจาก ROM มายัง รีจิสเตอร์ IR การปรับแต่งส่วนของรหัสดำเนินการนั้น จะปรับแต่งเพื่อให้รีจิสเตอร์ IR สามารถรับสัญญาณอินเตอร์รัพท์ แล้วเกิดการสร้าง รหัสดำเนินการเสมือนภายในตัวรีจิสเตอร์ IR เอง รหัสดำเนินการเสมือนที่ถูกสร้างขึ้นมานั้นคือ รหัสดำเนินการที่ใช้สำหรับสัญญาณอินเตอร์รัพท์ คือ เมื่อมีสัญญาณอินเตอร์รัพท์แล้ว รหัสดำเนินการสร้าง รหัสเพื่อบอกส่วนควบคุมให้สร้างสัญญาณร้องขออินเตอร์รัพท์ และทำตามคำสั่งของการอินเตอร์รัพท์ ดังแสดงการออกแบบวงจรอินเตอร์รัพท์ในรูปที่ 5.10



รูปที่ 5.10 การออกแบบวงจรอินเตอร์รัพท์

1. สัญญาณอินเตอร์รัพท์เปลี่ยนจาก 0->1
2. ส่วนควบคุมบอก PC ให้เก็บข้อมูลปัจจุบันลงรีจิสเตอร์ SP
3. คัดลอกข้อมูลในรีจิสเตอร์ PC ลง รีจิสเตอร์ SP
4. สัญญาณตอบรับอินเตอร์รัพท์เปลี่ยนจาก 0->1
5. Address vector ถูกโหลดเข้าสู่รีจิสเตอร์ IR
6. Interrupt opcode ถูกสร้างขึ้นมา
7. โหลดค่า PC ตัวใหม่
8. PC ชี้ไปยังโปรแกรมบริการอินเตอร์รัพท์
9. เมื่อเจอ RETI
10. คัดลอกค่าจาก SP เข้า PC

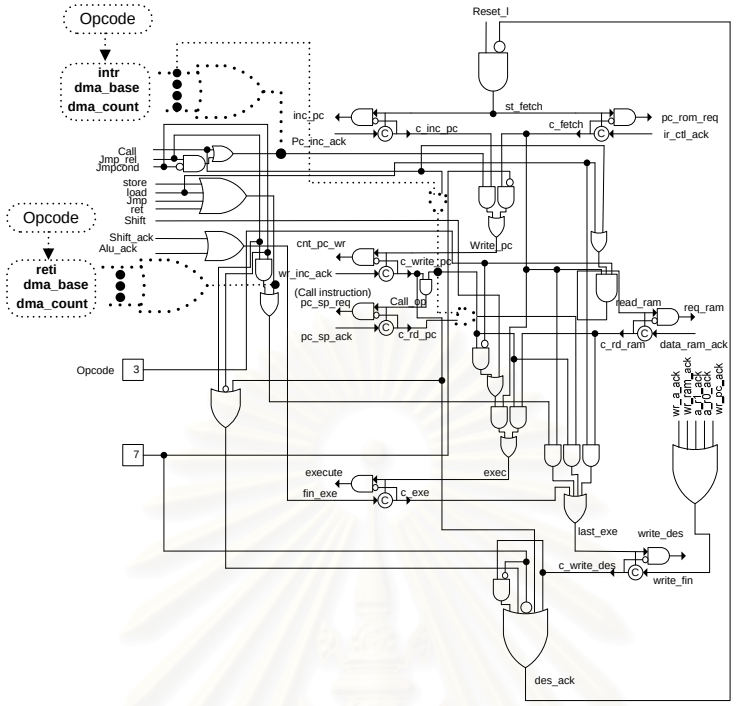
11. PC ส่งสัญญาณตอบรับไปยังส่วนควบคุม

12. ทำงานตามรหัสดำเนินการปกติ

การทำงานของไมโครโพรเซสเซอร์แบบอสมวาร เป็นดังนี้ โหลดรหัสดำเนินการ ที่ชี้โดยรีจิสเตอร์ PC เข้ามายัง รีจิสเตอร์ IR และทำงานตามชนิดของรหัสดำเนินการ เมื่อสัญญาณ EI เป็น 0 ไม่มีการตอบสนองการอินเตอร์รัพท์ใด ๆ ทั้งสิ้น จนกว่า การทำงานของรหัสดำเนินการสิ้นสุดขั้นตอนสุดท้าย สัญญาณ EI จึงเปลี่ยนจาก 0->1 เมื่อสัญญาณอินเตอร์รัพท์เปลี่ยนจาก 0->1 แล้ว รีจิสเตอร์ IR จะตัดการเชื่อมต่อของตัวเอง ที่เชื่อมกับแคชและสร้างรหัสดำเนินการเสมือนภายในตัวเอง จากนั้นรหัสดำเนินการจะถูกแบ่งออกเป็น 2 ส่วนคือ opcode และ operand opcode เป็นคำสั่งที่ส่งต่อไปยังส่วนควบคุมเพื่อให้ทำการ save ค่า PC ลงในรีจิสเตอร์ SP ส่วน operand จะทำการชี้ไปยังโปรแกรมบริการอินเตอร์รัพท์

5.3.3 การปรับแต่งส่วนควบคุม

ส่วนควบคุมจะเป็นส่วนที่ถูกปรับแต่งน้อยที่สุด เนื่องจากความซับซ้อนในการออกแบบโดยการปรับแต่ง จะทำการเพิ่มส่วนของวงจรเชิงผสม เพื่อให้สามารถรู้จัก opcode อันเนื่องจากสัญญาณอินเตอร์รัพท์ได้ จากนั้นเพิ่มส่วนวงจรเชิงผสม เพื่อสร้างสัญญาณ เอาท์พุต ดังนี้ dma_base_req dma_count_req และสัญญาณอินพุต dma_steal ดังแสดงการปรับแต่งในรูปที่ 5.11



External signal (Input)

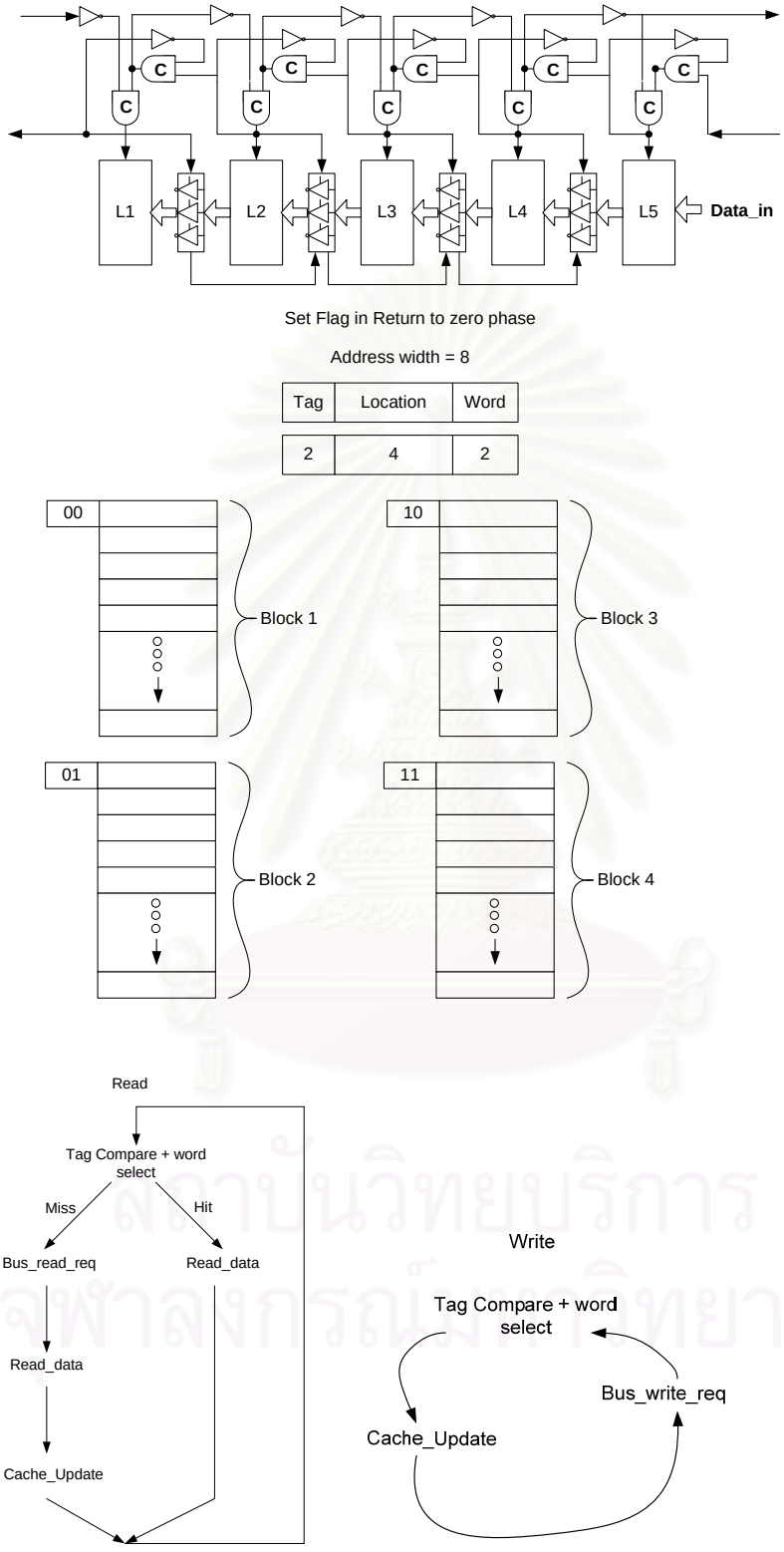
- reset_l
- opcode
- carry_out
- carry_out_l
- zero
- zero_l
- ir_ctl_ack
- wr_a_ack
- wr_pc_ack
- wr_inc_ack
- pc_inc_ack
- a_r0_ack
- a_r1_ack
- data_ram_ack
- pc_sp_ack
- wr_ram_ack
- alu_ack
- shift_ack
- **Dma_steal**

External signal (output)

- Opcode (7 downto 4) and execute → add, sub, orr, andd, xorr
- write_des and Cal → out_to_a
- st_fetch and not c_fetch → pc_rom_req
- inc_pc → pc_inc_req
- call_op and not c_rd_pc → pc_sp_req
- cxecute and Cal → a_alu_req
- **dma_steal and write_des and req_a_ram** → a_ram_req
- write_des and req_a_r0 → a_r0_req
- write_des and req_a_r1 → a_r1_req
- execute and shift → a_shf_req
- write_des and shift → shf_a_req
- execute and shift and not opcode (3) → do_shl
- execute and shift and opcode (3) → do_shr
- cnt_pc_wr → send_to_pc
- execute and reg_mode and opcode(0) → r0_alu_req
- write_des and req_r0_a → r0_a_req
- **dma_steal and req_r0_ram and (req_ram or (write_des and store))** → r0_ram_req
- **dma_steal and req_r1_ram and (req_ram or (write_des and store))** → r1_ram_req
- execute and opcode(3) → data_alu_req
- write_des and load and opcode(3) → data_a_req
- write_des and load and imm_mode → ir_a_req
- **write_des and (intr or call or jump or (jmp_rel and jmpcond)** → ir_pc_req
- **dma_steal and drct_mode and (req_ram or (write_des and store))** → ir_ram_req
- **dma_steal and req_ram** → ram_read
- **Write_des and dma_base** → **Dma_base_req (to IR)**
- **Write_des and dma_count** → **Dma_count_req (to IR)**
- execute and reg_mode and opcode(1) → r1_alu_req
- write_des and req_r1_a → r1_a_req
- write_des and ret → sp_pc_req
- execute and imm_mode → ir_alu_req

รูปที่ 5.11 การปรับแต่งส่วนควบคุม

5.4 การออกแบบแคชแบบอสมวาร



รูปที่ 5.12 การป้อนข้อมูลลงแคช, การออกแบบแคช, การเขียนและอ่านข้อมูลของแคช

ความเร็วของหน่วยความจำ มีค่าน้อยมากเมื่อเทียบกับความเร็วของไมโครโพรเซสเซอร์ ดังนั้นไมโครโครเซสเซอร์ จะต้องเสียเวลารอเมื่อมีการ อ่านและเขียนหน่วยความจำ และเมื่อมีการใช้งานดีเอ็มเอแล้ว คำสั่งสำหรับรหัสดำเนินการที่ต้องเกี่ยวกับหน่วยความจำจะไม่สามารถทำงานต่อไปได้ ทำให้เสียเวลาการรอ ผู้วิจัยจึงได้ออกแบบแคชแบบอสมวาร เพื่อรองรับการทำงานต่าง ๆ ที่ยืดหยุ่น มากขึ้น และเสียเวลาน้อยขึ้น ดังแสดงการออกแบบ วิธีการป้อนข้อมูลลงแคช การอ่านเขียนข้อมูลแคช และสถาปัตยกรรมแคช ไว้ในรูปที่ 5.12

5.5 การออกแบบส่วนเชื่อมต่ออุปกรณ์ต่อพ่วง

ในการเชื่อมต่อกับอุปกรณ์ต่อพ่วง สำหรับการใช้งานอุปกรณ์ต่พ่วงนั้น จะเป็นงานทำงานของไมโครโพรเซสเซอร์กับอุปกรณ์ภายนอก ดังนั้นจึงต้องมีการออกแบบส่วนเชื่อมต่อภายนอกด้วย ซึ่งในการเชื่อมต่อกับอุปกรณ์ต่อพ่วงนั้น สามารถแบ่งวิธีการอกแบบการเชื่อมต่อได้ ดังต่อไปนี้

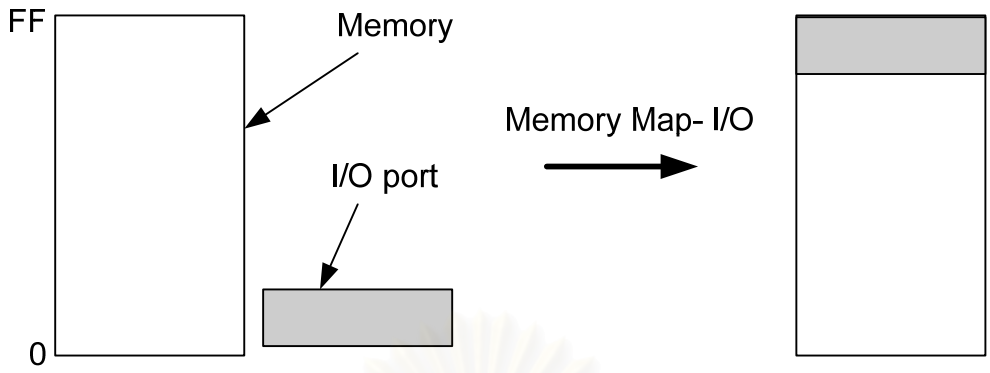
พิจารณาอุปกรณ์ต่อพ่วงให้เป็นตำแหน่งของหน่วยความจำ:

- ในการแลกเปลี่ยนข้อมูลกับอุปกรณ์ต่อพ่วง จะทำการแลกเปลี่ยนข้อมูลคล้ายกับการแลกเปลี่ยนข้อมูลของหน่วยความจำ ซึ่งหากออกแบบโดยใช้วิธีนี้ คำสั่งที่ใช้แลกเปลี่ยนข้อมูลของหน่วยความจำสำหรับไมโครโพรเซสเซอร์ จะสามารถนำมาใช้เพื่อทำการแลกเปลี่ยนข้อมูลกับ อุปกรณ์ต่อพ่วงได้ด้วย จึงไม่ต้องมีการออกแบบคำสั่งใหม่ เรียกวิธีการออกแบบนี้ว่า memory-mapped I/O.

เมื่อพิจารณาแยกออกจากหน่วยความจำ:

- อุปกรณ์ต่อพ่วงจะมี หมายเลขพอร์ตของตัวเอง ขนาด 8 บิต ตั้งแต่ 00H ถึง FFH โดยการเพิ่มคำสั่งขึ้นมาในการอ่านและเขียนข้อมูล เช่นคำสั่ง IN เพื่ออ่านข้อมูล และคำสั่ง OUT สำหรับการเขียนข้อมูล เรียกวิธีการออกแบบนี้ว่า I/O-mapped I/O หรือ Peripheral-mapped

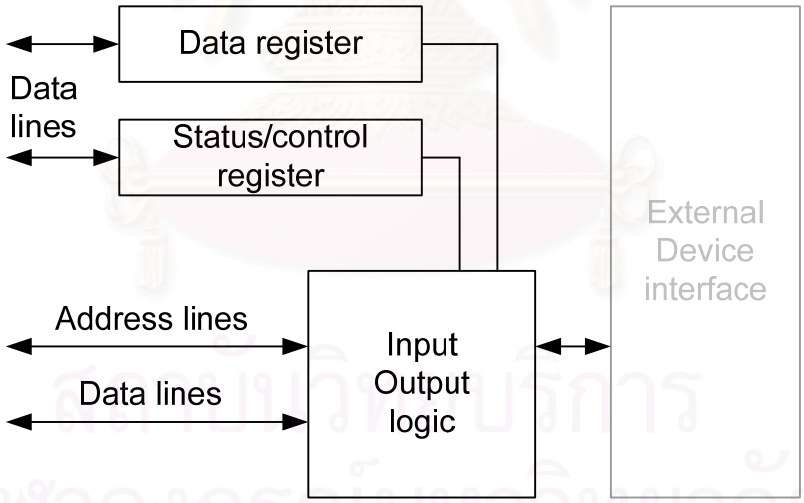
ขั้นตอนแรกในการเชื่อมต่ออุปกรณ์ต่อพ่วง ต้องพิจารณาคำสั่งในการสั่งงานอุปกรณ์ต่อพ่วงว่าจะเพิ่มคำสั่งหรือไม่เพิ่มคำสั่ง หากการเพิ่มคำสั่งสามารถทำได้โดยง่ายก็อาจเพิ่มคำสั่ง IN และคำสั่ง OUT เพื่อทำงานแบบ I/O Mapped I/O แต่หากการเพิ่มคำสั่งแล้วทำให้เกิดภาระความซ้อนมากขึ้น ก็อาจใช้วิธีการใช้คำสั่งที่มีอยู่แล้วของหน่วยความจำ หรือออกแบบโดยใช้วิธี Memory mapped I/O ซึ่งผู้วิจัยได้ออกแบบโดยใช้วิธี Memory mapped I/O หรือแบบไม่เพิ่มคำสั่งนั่นเอง ดังแสดงการออกแบบในรูปที่ 5.13



รูปที่ 5.13 การผนวกหน่วยความจำ

รหัสดำเนินการที่มีผลต่อการแลกเปลี่ยนข้อมูลระหว่างไมโครโพรเซสเซอร์และอุปกรณ์ต่อพ่วง ประกอบด้วย :

- LD ย้ายข้อมูล (8-bits) จากอุปกรณ์ต่อพ่วง ไปยัง accumulator
- ST ย้ายข้อมูบ (8-bits) จาก accumulator มายังอุปกรณ์ต่อพ่วง



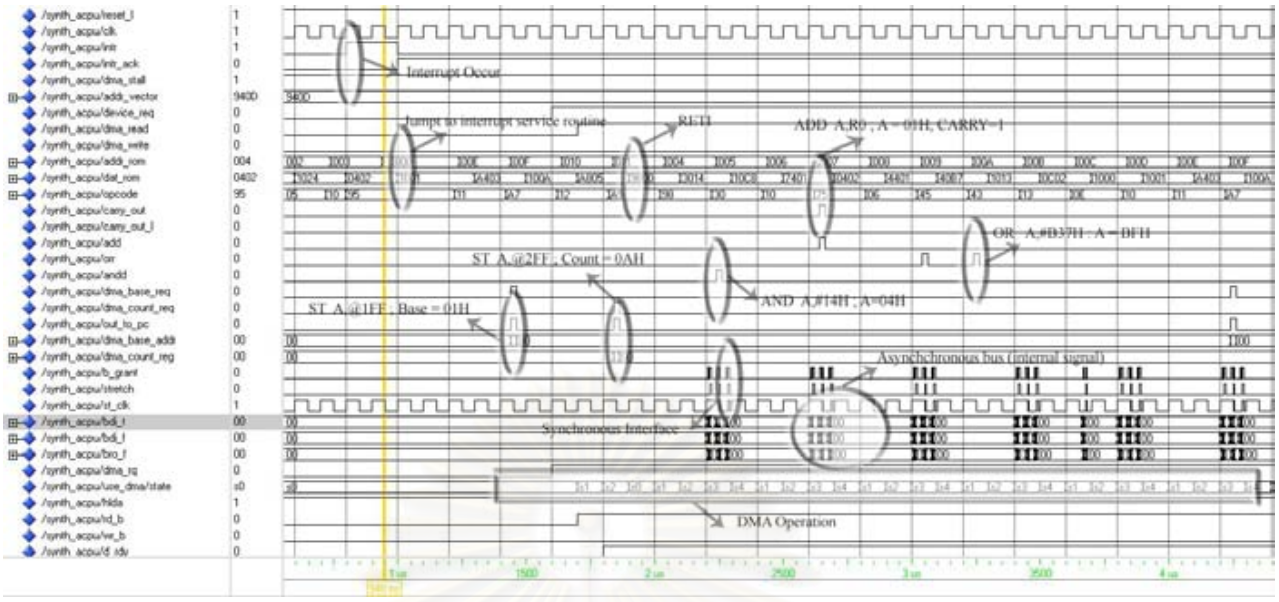
รูปที่ 5.14 การออกแบบส่วนเชื่อมต่ออุปกรณ์ต่อพ่วง

5.6 การจำลองการทำงาน

ส่วนนี้เป็นการรวมทุกวงจรเข้าด้วยกัน ประกอบด้วย ไมโครโพรเซสเซอร์แบบอสมวาร บัสแบบอสมวาร ดีเอ็มเอ ส่วนเชื่อมต่อวงจรสมวาร และหน่วยความจำ ดังแสดงในรูปที่ 5.1 และ 5.2 จากนั้นนำเสนอการจำลองการทำงานของวงจรสมบูรณ์ดังแสดงในรูปที่ 5.15

โปรแกรมสำหรับการจำลองการทำงานวงจรสมบูรณ์

```
        ORG   0000H
        LD    A,#39H; A = 39H
        ST    A,R0    ; R0 = 39H
        LD    A,#24H; A = 24H
        ST    A,R1    ; R1 = 24H
        AND   A,#14H; A= 04H
        LD    A,#C8H          ; A = C8H
        ADD   A,R0    ; A = 01H, CARRY = 1
        ST    A,R1    ; R1 = 01H
        OR    A,R0    ; A = 39H
        OR    A,#B7H          ; A = BFH
        LD    A,#13H; A = 13H
        ST    A,@R1 ; M(01) = 13H
        LD    A,#00H; A=00H
INTR :
        LD    A,#01H ; A=01H
        ST    A,@1FFH ; DMA_Base = 01H
        LD    A,#0AH          ; A=0AH
        ST    A,@2FFH ; DMA_Count = 0AH
        RETI
        END
```



รูปที่ 5.15 การจำลองการทำงานวงจรสมบูรณ์

5.7 **สรุป**

บทนี้นำเสนอการนำวงจรที่ออกแบบทั้งหมดมารวมกันเป็นระบบแล้วจำลองการทำงาน วงจรสมบูรณ์ประกอบด้วย ไมโครโพรเซสเซอร์แบบอสมวาร บัสแบบอสมวาร ดีเอ็มเอ หน่วยความจำ และส่วนเชื่อมต่ออุปกรณ์ต่อพ่วง ในบทนี้ได้นำเสนอวิธีการปรับแต่งไมโครโพรเซสเซอร์แบบอสมวารที่นำมาใช้ในการเชื่อมต่อวงจร โดยจะปรับแต่ง 3 จุดใหญ่ๆ คือ ปรับแต่งสถาปัตยกรรม ปรับแต่งรหัสดำเนินการ และปรับแต่งส่วนควบคุม จากนั้นจึงนำเสนอการออกแบบ แคชแบบอสมวาร ตามด้วยการออกแบบวงจรเชื่อมต่ออุปกรณ์ต่อพ่วง เป็นวงจรเพิ่มเติมเพื่อใช้กับระบบที่ออกแบบ และนำเสนอการจำลองการทำงานในตอนท้าย

บทที่ 6

การจำลองการทำงานแบบอิงเวลา และการโปรแกรมวงจรลงเอฟพีจีเอ

บทนี้นำเสนอการจำลองการทำงานแบบอิงเวลาเนื่องจากความหน่วงเอฟพีจีเอ และการโปรแกรมลงบอร์ดทดลอง เพื่อแสดงการทำงานจริงของวงจรที่ออกแบบ โดยในการทดสอบ จะทำการทดสอบ การอ่านเขียนข้อมูลของบัส การอ่านเขียนข้อมูลของดีเอ็มเอแบบสมวารและอสมวาร จากนั้นทำการเปรียบเทียบระหว่างดีเอ็มเอแบบสมวาร และอสมวาร ในตอนท้ายของบท นำเสนอการโปรแกรมวงจรลงเอฟพีจีเอเพื่อทดสอบการทำงานจริงกับบอร์ดทดลอง

6.1 โปรแกรมที่ใช้จำลองการทำงาน

โปรแกรมที่ใช้ในการจำลองการทำงานประกอบด้วย:

I. Leonardo Spectrum (ใช้ในการสังเคราะห์วงจร)

II. ISE Webpack (ใช้ในการอิมพลีเม้นท์วงจร)

III. ModelSim XE (ใช้ในการจำลองการทำงานแบบอิงเวลา)

6.2 ขั้นตอนการจำลองการทำงาน

เริ่มต้นโดยการนำวงจรที่ออกแบบมาจำลองการทำงานในระดับฟังก์ชัน ให้มีความถูกต้องของการทำงานทุกคำสั่ง หลังจากนั้นจึงเริ่มเข้าสู่กระบวนการจำลองการทำงานแบบอิงเวลา ในการจำลองการทำงานแบบอิงเวลา สามารถแบ่งกระบวนการได้ 3 กระบวนการใหญ่ ๆ ด้วยกัน คือ การสังเคราะห์วงจร การอิมพลีเม้นท์วงจร และการจำลองการทำงานแบบอิงเวลา ในส่วนของการสังเคราะห์วงจร ใช้โปรแกรม Leonardo Spectrum ทำการสังเคราะห์วงจรที่ออกแบบไว้ทั้งหมด แล้วจะได้ไฟล์ *.EDF หลังจากทำการสังเคราะห์ ส่วนการอิมพลีเม้นเป็นการนำ ไฟล์ EDF ที่ได้จากการสังเคราะห์วงจรจากโปรแกรม Leonardo Spectrum แล้วนำมาจัดวางและเชื่อมสายลงเอฟพีจีเอด้วยโปรแกรม ISE Webpack ตามเบอร์ของเอฟพีจีเอที่เลือกไว้ จากนั้น ทำการกำหนดเวลาแบบสแตติก ฟลอแพลน และการวางขาอุปกรณ์ตามข้อกำหนดของวงจรที่ออกแบบ แล้วสร้างไฟล์ที่ใช้ในการจำลองการทำงานแบบอิงเวลา โดยจะได้ไฟล์ .VHD กับ ไฟล์ .SDF ซึ่งไฟล์ .VHD เป็นไฟล์วงจรที่ออกแบบผ่านการอิมพลีเม้นท์เพื่อปรับให้สามารถลงเอฟพีจีเอแล้ว ส่วนไฟล์ .SDF เป็นไฟล์ความหน่วงของเอฟพีจีเอ ที่โปรแกรม ISE Webpack สร้างขึ้นมา เป็นอันสิ้นสุดขั้นตอนการอิมพลีเม้นท์วงจร จากนั้นนำ ไฟล์ .VHD และ SDF ไปทำการจำลองการทำงานแบบอิงเวลา

ผ่านโปรแกรม ModelSimXE เมื่อได้การจำลองการทำงานที่ถูกต้องแล้ว จึงสร้างไฟล์ .bit จากโปรแกรม ISE Webpack อีกครั้งหนึ่ง เพื่อโปรแกรมลงเอฟพีจีเอต่อไปดังแสดงขั้นตอนต่าง ๆ ในรูปที่ 6.1

6.2.1 **การสังเคราะห์วงจร**:

I. เปิดโปรแกรม Leonardo Spectrum

II. เลือก License Level 1

III. กำหนดอุปกรณ์ -> Spartan-3E ->3S500EFG320 ->speed gate = -4

IV. เพิ่มอินพุตไฟล์ (*.vhd , กำหนด top level to -> synth_acpu.vhd)

V. กำหนด Optimized to -> Optimized delay and preserve signal

VI. กด Run flow. (ในขั้นนี้จะได้ไฟล์ Synth_acpu.EDF)

6.2.2 **การอิมพลีเม้นท์**:

VII. เปิดโปรแกรม ISE Web pack

VIII. สร้างโปรเจคใหม่ (File -> New Project)

IX. ทำการตั้งชื่อโปรเจคใหม่และเลือกพาธในการติดตั้ง, เลือก top level เป็นไฟล์ EDIF

X. ที่หน้าต่าง process,ทำการอิมพลีเม้นท์วงจร โดยการเลือก translate -> ->map-> place and route

XI. กด Generate post place and route simulation model เพื่อสร้างไฟล์ .VHD และ .SDF เพื่อใช้ในการจำลองการทำงานแบบอิงเวลา

6.2.3 **การจำลองการทำงานแบบอิงเวลา**

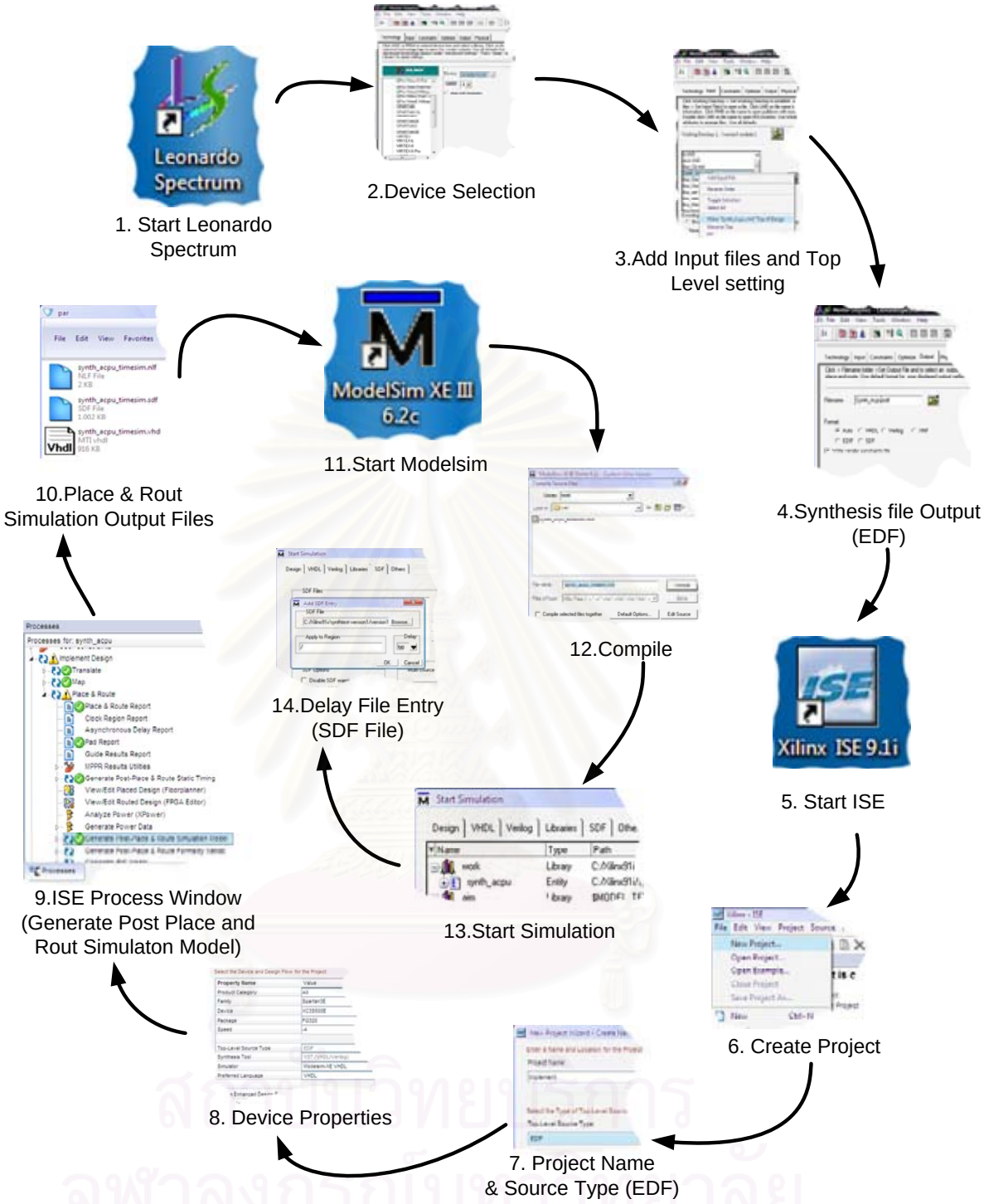
XII. เปิดโปรแกรม Modelsim XE

XIII. เปลี่ยนซอร์สไดเรกทอรี่เป็น *Synth_acpu_time_sim.vhd* และ *Synth_acpu_time_sim.sdf* ที่สร้างโดยโปรแกรม ISE Web pack.

XIV. สร้าง work directory

XV. คอมไพล์

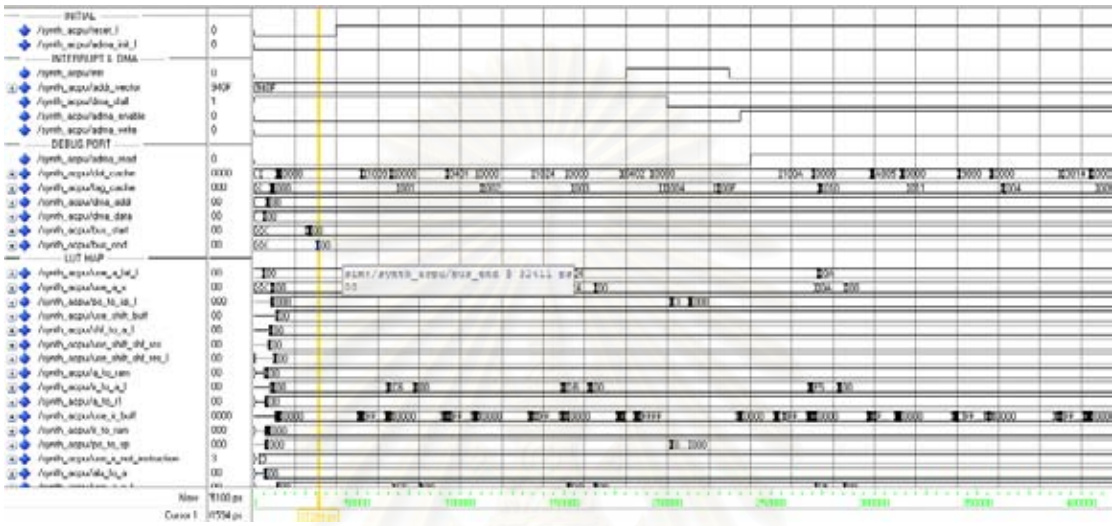
XVI. จำลองการทำงานโดยเลือกไฟล์ .SDF พร้อมกับการจำลองการทำงาน



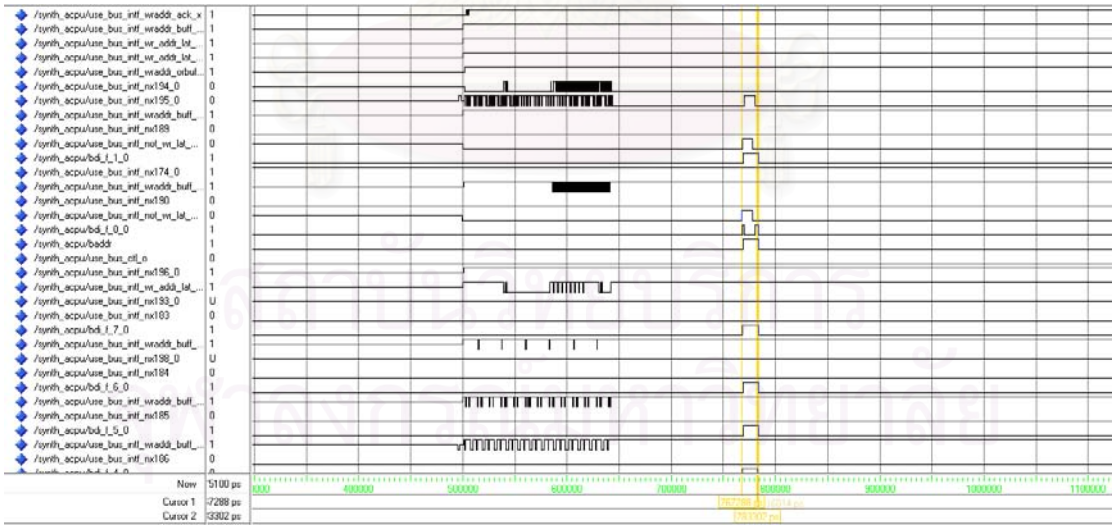
รูปที่ 6.1 ขั้นตอนการจำลองการทำงานแบบอิงเวลา

6.3 ผลการจำลองการทำงาน

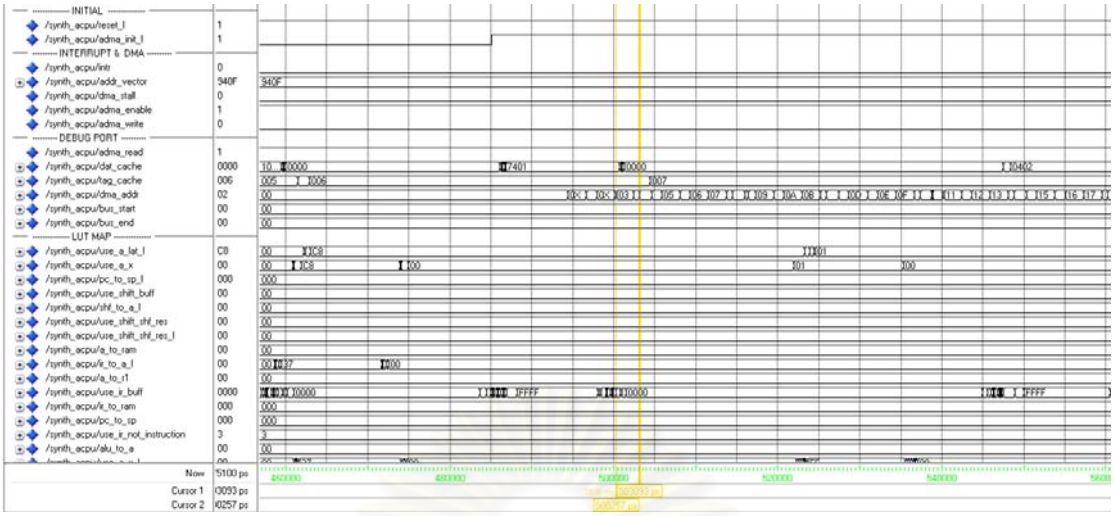
ส่วนนี้แสดงผลการจำลองการทำงานบางสัญญาณเท่านั้น ซึ่งประกอบด้วยเวลาเริ่มต้นการทำงาน เวลาการอ่านข้อมูลของบัส เวลาการอ่านข้อมูลของดีเอ็มเอแบบอสมวาร และเวลาการตอบสนองของสัญญาณอินเตอร์รัพท์ ส่วนสัญญาณอื่น ๆ จะสรุปรวมไว้ในตางรางที่ 6.1



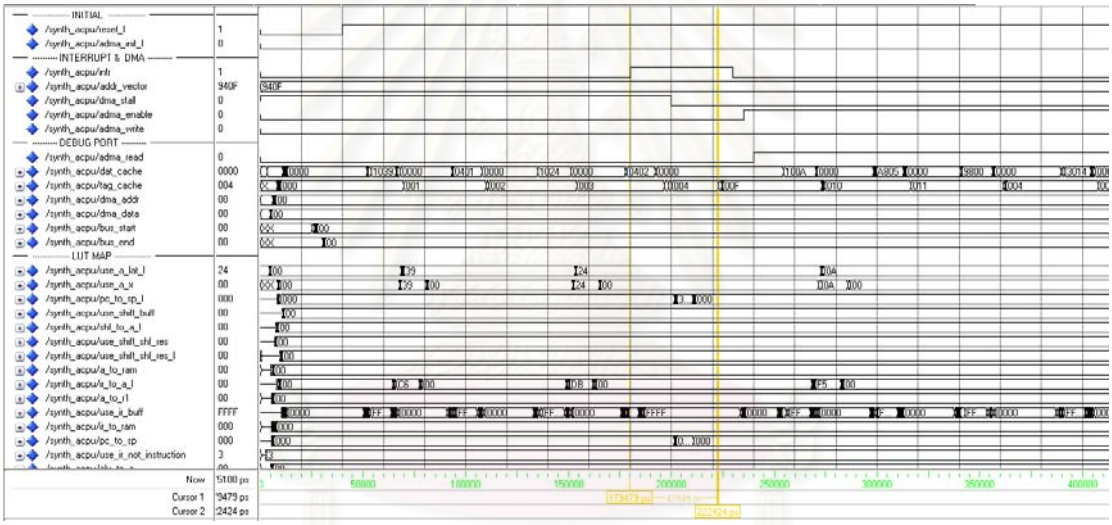
รูปที่ 6.2 เวลาในการเริ่มต้นการทำงาน (32 ns)



รูปที่ 6.3 เวลาการอ่านข้อมูลของบัส (16 ns)



รูปที่ 6.4 เวลาในการอ่านข้อมูลของดีเอ็มเอแบบอสมวาร (28 ns)



รูปที่ 6.5 เวลาในการตอบสนองอินเตอร์รัพท์ (42 ns)

TIMING SUMMARY *(ns) @- 100 Mhz Memory*		
TYPE	This work	Ref. Processor
Setup Time	32	-
Interrupt Response Time	42	-
Bus Read Cycle	16	-
Bus Write Cycle	15	-
Asynchronous DMA Read Cycle	28	-
Asynchronous DMA Write Cycle	30	-
Synchronous DMA Read Cycle	45	-
Synchronous DMA Write Cycle	45	-
LD (Immediate)	42	57
LD (Register)	40	57
LD (Direct)	65	93
LD (Indirect)	58	93
ST (Register)	39	57
ST (Direct)	60	71
ST (Indirect)	57	71
AND (immediate)	55	80
AND (Register)	60	81
AND (Direct)	80	126
AND (Indirect)	80	127
OR (Immediate)	80	86
OR (Register)	80	85
OR (Direct)	120	123
OR (Indirect)	120	123
SUB (Immediate)	90	96
SUB (Register)	100	105
SUB (Direct)	144	144
SUB (Indirect)	130	143
ADD (Immediate)	90	93
ADD (Register)	110	113
ADD (Direct)	130	138
ADD (Indirect)	130	139
JMP	45	59
CALL	50	70
RET	45	59
RETI	43	-

ตารางที่ 6.1 ตารางแสดงเวลาการทำงาน

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	2	9,312	1%	
Number of 4 input LUTs	1,662	9,312	17%	
Logic Distribution				
Number of occupied Slices	880	4,656	18%	
Number of Slices containing only related logic	880	880	100%	
Number of Slices containing unrelated logic	0	880	0%	
Total Number of 4 input LUTs	1,662	9,312	17%	
Number of bonded IOBs	82	232	35%	
Number of Block RAMs	2	20	10%	
Number of GCLKs	1	24	4%	
Total equivalent gate count for design	141,063			
Additional JTAG gate count for IOBs	3,936			

ตารางที่ 6.2 ตารางแสดงการใช้งานเอฟพีจีเอ (วงจรที่ประกอบด้วยดีเอ็มเอแบบอสมวาร)

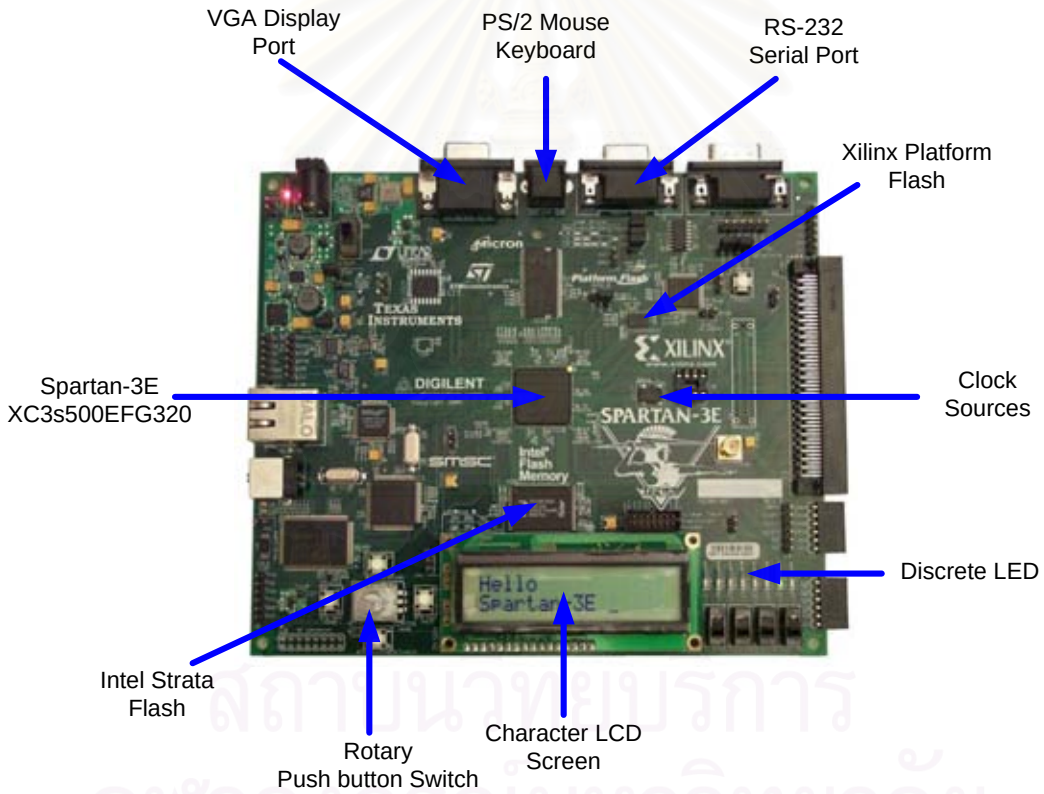
Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	26	9,312	1%	
Number of 4 input LUTs	1,694	9,312	18%	
Logic Distribution				
Number of occupied Slices	899	4,656	19%	
Number of Slices containing only related logic	899	899	100%	
Number of Slices containing unrelated logic	0	899	0%	
Total Number of 4 input LUTs	1,694	9,312	18%	
Number of bonded IOBs	66	232	28%	
IOB Flip Flops	3			
Number of Block RAMs	2	20	10%	
Number of GCLKs	1	24	4%	
Total equivalent gate count for design	141,587			
Additional JTAG gate count for IOBs	3,168			

ตารางที่ 6.3 ตารางแสดงการใช้งานเอฟพีจีเอ (วงจรที่ประกอบด้วยดีเอ็มเอแบบสมวาร)

6.4 การโปรแกรมวงจรลงเอฟพีจีเอ

เพื่อให้ได้การทำงานจริงของวงจรที่ออกแบบ ผู้วิจัยจึงนำวงจรที่ออกแบบมาโปรแกรมลงเอฟพีจีเอ เบอร์ XC3S500EFG320 ซึ่งบรรจุอยู่ในบอร์ดทดลอง Spartan-3E ของบริษัท Xilinx Corporation ในการทดสอบการทำงานของวงจร ผู้วิจัยได้สร้างชุดโปรแกรมทดสอบไว้สามชุดด้วยกัน คือ ชุดทดสอบที่ประกอบด้วยสวิทช์แบบกดหมุนกับหลอดแอลอีดี ชุดทดสอบที่ประกอบด้วยสวิทช์แบบกดติดปล่อยดับกับจอแอลซีดี และชุดทดสอบที่ประกอบด้วยคีย์บอร์ดแบบ PS/2 กับจอมอนิเตอร์

6.4.1 บอร์ดทดลอง Spartan-3E



รูปที่ 6.6 บอร์ดทดลอง Spartan-3E

บอร์ดทดลอง Spartan-3E เป็นบอร์ดทดลองที่ผลิตโดย บริษัท Xilinx เป็นบอร์ดที่รวมอุปกรณ์ต่อพ่วง อาทิ เช่น สวิทช์ คีย์บอร์ด เม้าส์ จอแอลซีดี ช่องเชื่อมต่ออุปกรณ์ภายนอกต่าง ๆ หน่วยความจำแบบแฟลช เป็นต้น สำหรับความสามารถทั้งหมดของบอร์ดทดลอง Spartan-3E นั้น

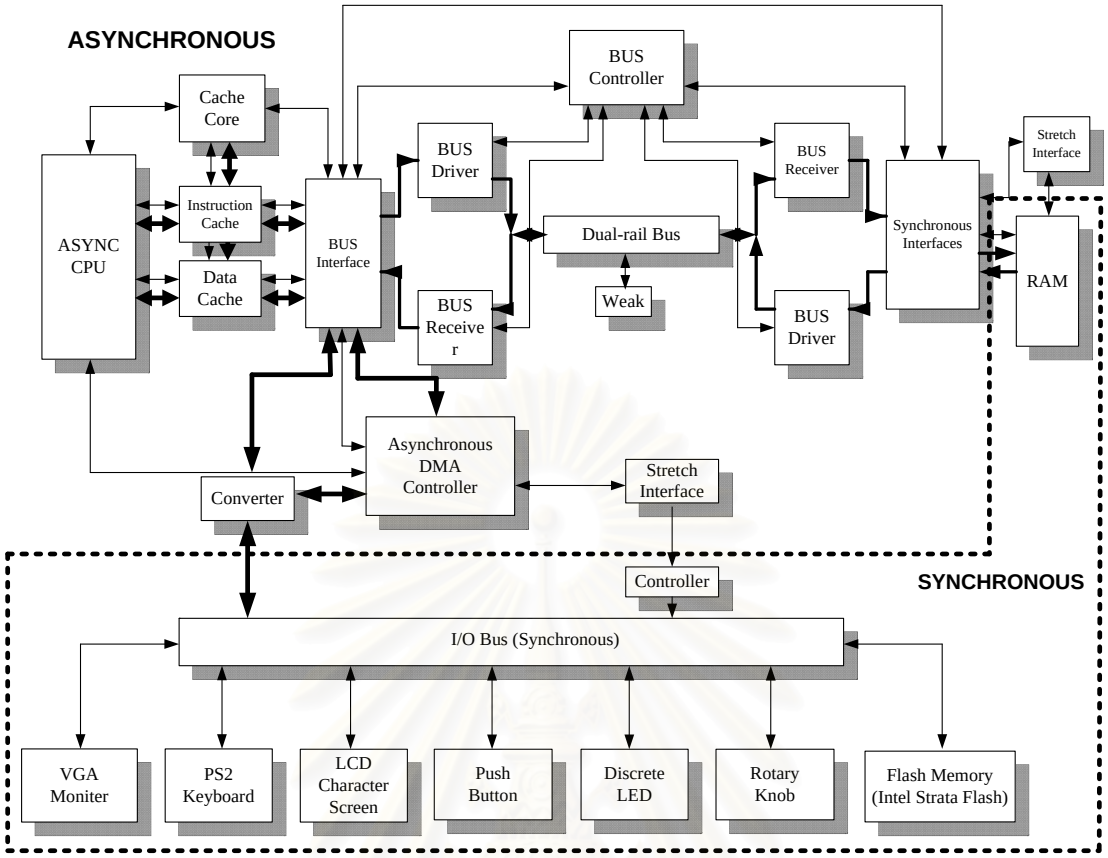
สามารถ ดูได้ที่ www.xilinx.com/s3estarter รูปที่ 6.6 แสดงตำแหน่งการวางอุปกรณ์ต่อพ่วงต่าง ๆ ที่นำมาใช้ในการจำลองการทำงานจริงกับบอร์ดทดลองสำหรับงานวิจัยชิ้นนี้ ซึ่งประกอบด้วย:

- เอฟพีจีเอเบอร์ XC3S500EFG320
- คอนฟิกูเรชั่น PROM
- 16 เมกะไบต์ (128 เมกะบิต) นอร์แฟลช (Intel Strata Flash)
- จอแอลซีดี ขนาด 16 ตัวอักษร/ 2- บรรทัด
- พอร์ท PS/2
- พอร์ท VGA
- สวิทซ์แบบกดติดปล่อยดับ
- หลอดแอลอีดี
- สวิทซ์แบบกดหมุน

6.4.2 **การทดสอบ**

วงจรทดสอบ จะต่อวงจรที่ออกแบบกับอุปกรณ์ต่อพ่วงจำนวน 7 ตัวด้วยกัน ดังแสดงในรูปที่ 6.7 และแบ่งการทดสอบออกเป็น 3 ชุดด้วยกัน:

ชุดที่ 1 : สวิทซ์แบบกดติดปล่อยดับกับหลอดแอลอีดี
ชุดที่ 2 : สวิทซ์แบบกดติดปล่อยดับกับจอแอลซีดี
ชุดที่ 3 : คีย์บอร์ดแบบ PS/2 กับจอมอนิเตอร์

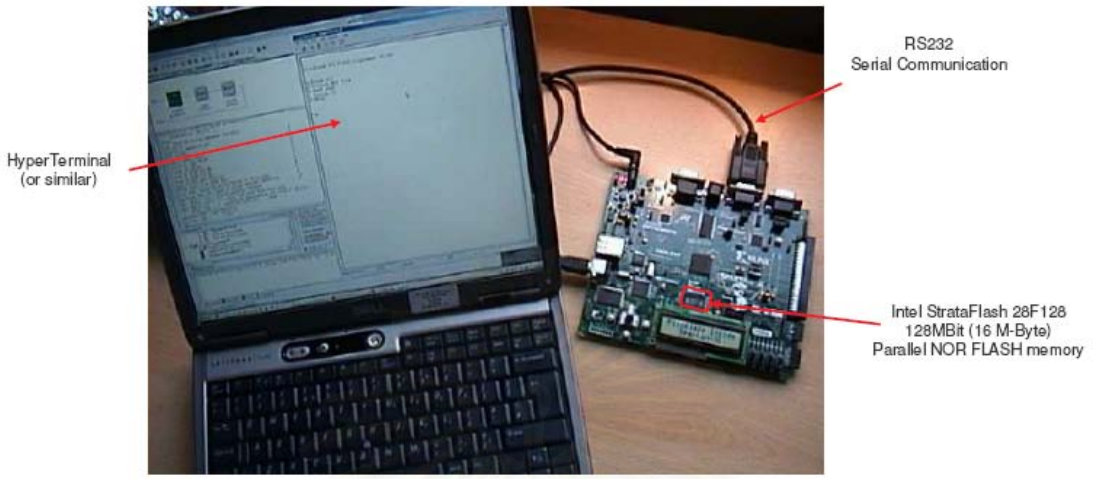


\

รูปที่ 6.7 วงจรทดสอบ

ไฟล์ที่ได้จากการคอมไพล์จาก C32 แอสเซมเบลอร์ ถูกเก็บไว้ที่ Intel strata flash จอมอนิเตอร์เป็นส่วนแสดงผลที่จะแสดงค่า ตามการควบคุมของคีย์บอร์ดแบบ PS/2 จอแอลซีดีจะแสดงตัวอักษรตามการกดปุ่มแบบกดติดปล่อยดับ และหลอดแอลอีดีจะกระพริบตามการหมุนและกดของสวิทช์แบบกดหมุน

6.4.3 การโปรแกรมข้อมูลลง Intel Strata Flash

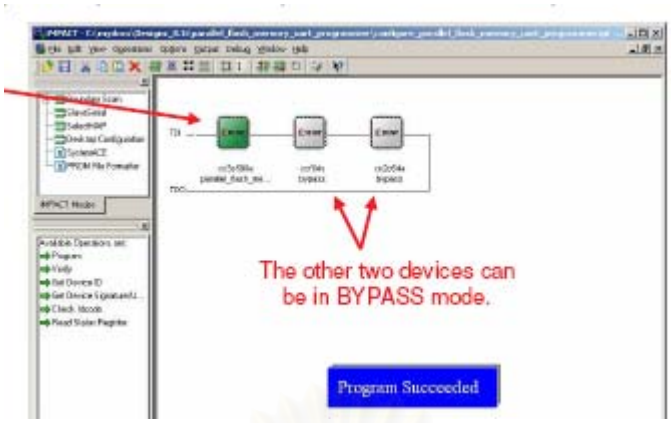


รูปที่ 6.8 การติดตั้งฮาร์ดแวร์สำหรับโปรแกรม Intel Strata Flash

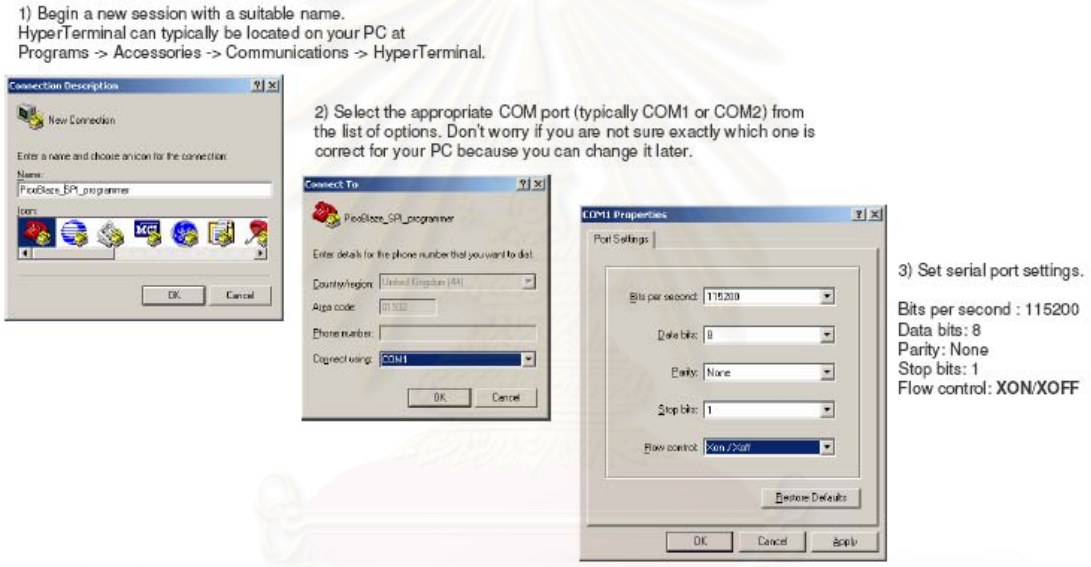
วงจรนี้เป็นการเปลี่ยนเอฟพีจีเอให้ ทำงานเป็น NOR FLASH programmer สำหรับหน่วยความจำแบบแฟลช Intel Strata Flash (IC22) โดยใช้เทอร์มินอลโปรแกรมแบบง่าย ๆ ไฮเปอร์เทอร์มินอลที่มีมาพร้อมกับไมโครซอฟท์วินโดวส์เกือบทุกเวอร์ชั่น

รูปที่ 6.9 การโปรแกรม Nor Flash Programmer ลงเอฟพีจีเอ

NOR Flash programmer ถูกสร้างให้เป็นคอนฟิกูเรชั่นบิตไฟล์ สำหรับโปรแกรมลงเอฟพีจีเอ เบอร์ XC3S500E ซึ่งติดตั้งมาพร้อมกับบอร์ดทดลอง Spartan-3E

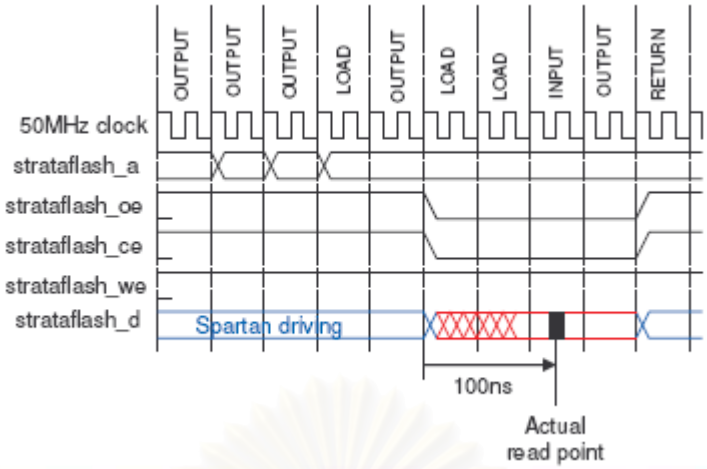


รูปที่ 6.10 การโปรแกรมคอนฟิกูเรชั่นไฟล์ลงเอฟจีจีเอ

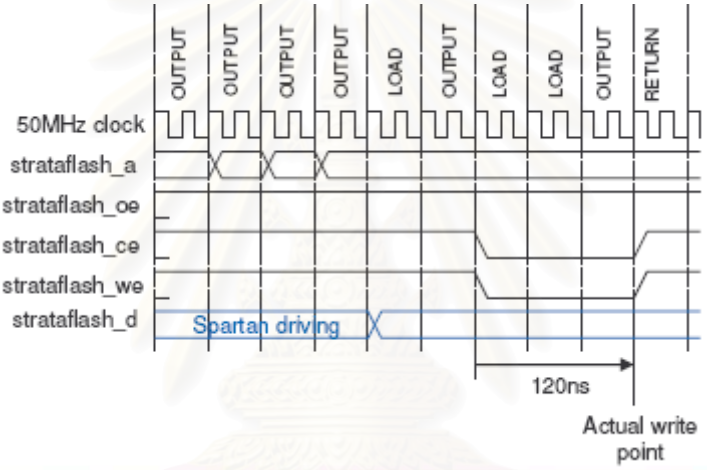


รูปที่ 6.11 การตั้งค่าสำหรับการรับส่งข้อมูลแบบอนุกรม

เมื่อวงจรโหลดเข้าสู่บอร์ดทดลอง Spartan-3E จากนั้นจึงทำการเตรียมสภาพแวดล้อมในการติดต่อแบบอนุกรมผ่านพอร์ท RS-232 ผ่านโปรแกรมไฮเปอร์ เทอร์มินอล ดังแสดงในรูปที่ 6.11 เพื่อเป็นตัวโปรแกรมข้อมูลลง Intel Strata Flash เมื่อตั้งค่าต่าง ๆ เสร็จเรียบร้อยแล้วเริ่มทำงานส่งข้อมูลบิตไฟล์เพื่อโปรแกรมลง Intel Strata Flash ดังแสดงเวลาการอ่านเขียนข้อมูลดังแสดงในรูปที่ 6.12 และ 6.13 ตามลำดับ



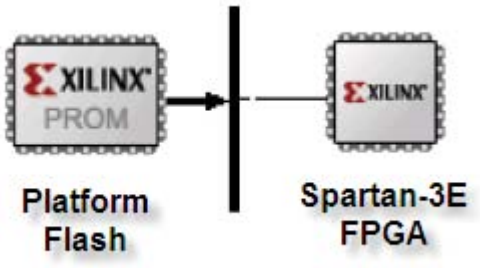
รูปที่ 6.12 เวลาการอ่านข้อมูลจาก Intel Strata Flash



รูปที่ 6.13 เวลาการเขียนข้อมูลลง Intel Strata Flash

6.4.4 **การโปรแกรมข้อมูลลง** Plat form Flash

การเก็บคอนฟิกูเรชั่นไฟล์ ซึ่งเป็นไฟล์วงจรที่ออกแบบทั้งหมด จะเก็บไว้ที่แพลตฟอร์มแฟลช โดยการตั้งค่าบอร์ดทดลองให้ เป็นมาสเตอร์ ซึ่งเมื่อกดปุ่มเปิดบอร์ดทดลองแล้วจะทำการโหลดไฟล์จากแพลตฟอร์มแฟลชเข้าสู่เอฟพีจีเอโดยอัตโนมัติ ดังแสดงในรูปที่ 6.14 ซึ่งการโปรแกรมแพลตฟอร์มแฟลตทำได้โดยง่ายโดยโปรแกรมผ่านสายยูเอสบีผ่านโปรแกรมอิมแพคที่มีอยู่ในโปรแกรม ISE Webpack



รูปที่ 6.14 การโปรแกรมเอฟพีจีเอผ่านแพลตฟอร์มแฟลช

ในการโปรแกรมข้อมูลลง แพลตฟอร์มแฟลช จะทำการสร้างคอนฟิกูเรชั่นไฟล์บิตสตรีมก่อนที่จะสร้าง PROM File ในการสร้างบิตสตรีมไฟล์ เอฟพีจีเอประกอบด้วย สัญญาณนาฬิกาแบบส่งออก CCLK เมื่อทำการโหลดจาก PROM ภายนอก แพลตฟอร์มแฟลช เบอร์ XCF04S สนับสนุนสัญญาณนาฬิกาที่ 25 เมกะเฮิร์ท

คลิ๊กขวาแล้วเลือก Generate Programming File ในหน้าต่าง Processes ของโปรแกรม ISE Webpack ตามด้วยคลิ๊กซ้าย เลือก Properties คลิ๊กที่ Configuration Options เลือก 25 เพื่อเพิ่ม CCLK oscillator ไปที่ 25 MHz ความถี่สูงที่สุดเมื่อเลือกแพลตฟอร์มแฟลชเบอร์ XCF04S คือ 25 เมกะเฮิร์ท คลิ๊ก OK เมื่อเสร็จ เพื่อทำงานสร้างไฟล์สำหรับโปรแกรม โดยการดับเบิ้ลคลิ๊กที่ Generate Programming File ในโปรแกรม ISE Webpack

หลังจากทำงานสร้างไฟล์สำหรับโปรแกรม แล้วดับเบิ้ลคลิ๊กที่ Generate PROM, ACE, or JTAG File เพื่อเปิดโปรแกรม อิมแพค เมื่อโปรแกรมอิมแพค เริ่มเปิดขึ้นมาแล้ว ให้ดับเบิ้ลคลิ๊กที่ PROM File Formatter, แล้วเลือก Xilinx PROM เลือกอีกครั้งเป็น Intel Hex format (MCS) กำหนดตำแหน่งพาธที่จะทำการสร้างไฟล์ PROM แล้วคลิ๊ก Next

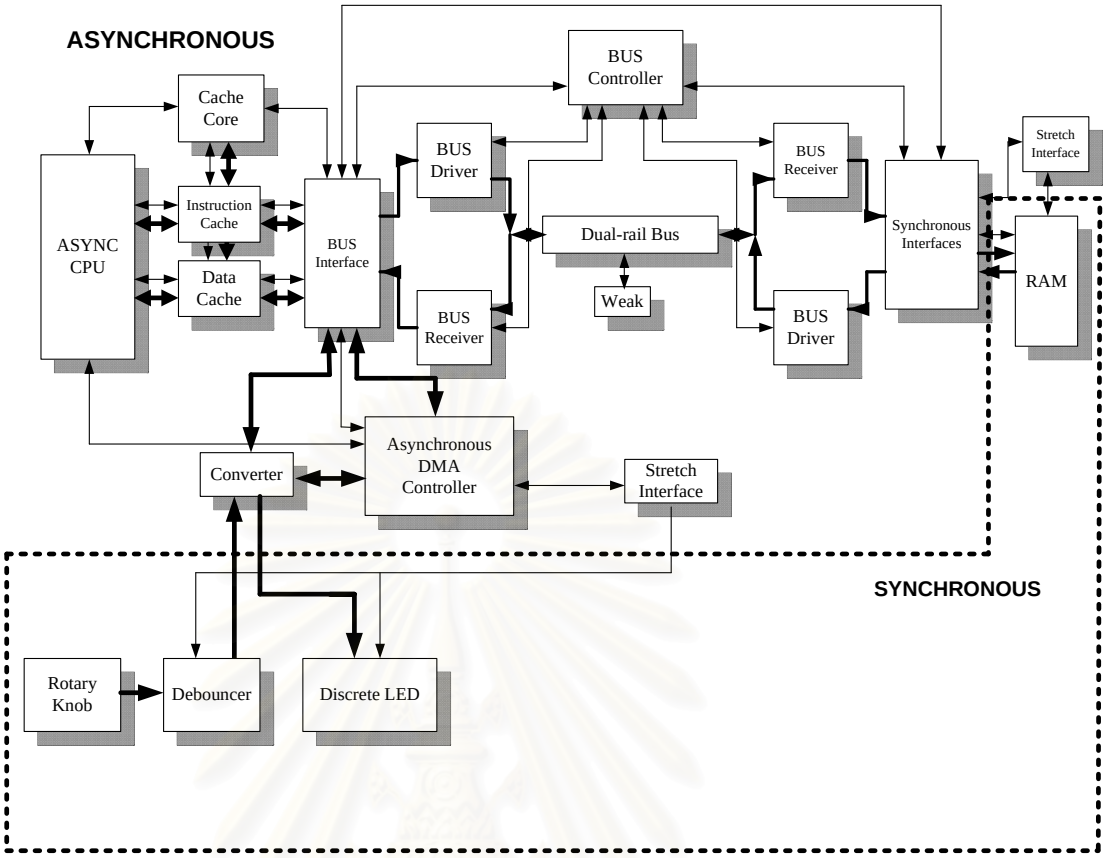
บอร์ดทดลอง Spartan-3E Starter ประกอบด้วยแพลตฟอร์มแฟลชเบอร์ XCF04S ให้เลือก xcf04s จากดรอปดาวน์ลิสท์ คลิ๊ก Add, จากนั้นคลิ๊ก next >. The PROM Formatter คลิ๊ก Finish

PROM Formatter จะถามชื่อบิตสตรีมไฟล์ของเอฟพีจีเอ ให้คลิ๊ก OK เพื่อเลือกไฟล์แล้วทำการเลือกไฟล์บิตสตรีมคินฟิกูเรชั่นไฟล์ของเอฟพีจีเอที่ต้องการเก็บลงใน แพลตฟอร์มแฟลช จากนั้น คลิ๊ก OK เพื่อไปขั้นต่อไป

6.4.4 การทดสอบวงจรกับสวิทซ์แบบกดหมุนและหลอดแอลอีดี

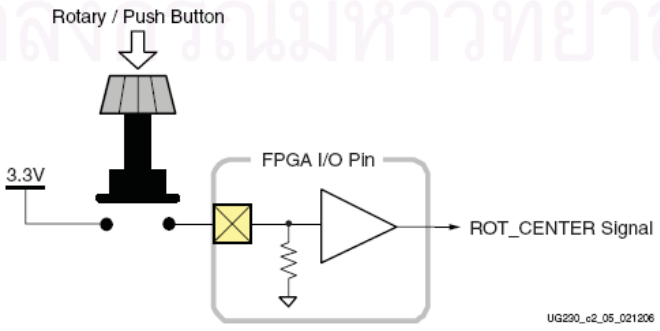
การทำงานของชุดทดสอบชุดนี้ คือ หลอดแอลอีดีจะหมุนซ้ายขวา ติดและดับ ตามการหมุนและการกดของสวิทซ์แบบกดหมุน โดยจะทำงานตามข้อกำหนดดังนี้

- เมื่อเริ่มต้นโปรแกรม หลอดแอลอีดีจะติด 1 หลอดจากทั้งหมด 8 หลอด
- เมื่อหมุนสวิทซ์แบบกดหมุนไปทางซ้าย หลอดแอลอีดีก็จะติดทางซ้ายตามการหมุนและดับหลอดปัจจุบันที่ติดอยู่
- เมื่อหมุนสวิทซ์แบบกดหมุนไปทางขวา หลอดแอลอีดีก็จะติดทางขวาตามการหมุนและดับหลอดปัจจุบันที่ติดอยู่
- เมื่อมีการกดสวิทซ์แบบกดหมุน จะเกิดสัญญาณอินเตอร์รัพท์ไปยังไมโครโพรเซสเซอร์แบบอสมวารแล้วกระโดดไปยังโปรแกรมบริการอินเตอร์รัพท์ ซึ่งในโปรแกรมบริการอินเตอร์รัพท์ประกอบด้วย การอ่านข้อมูลจากหน่วยความจำผ่านบัสแบบอสมวาร ซึ่งข้อมูลที่อ่านได้มีค่าเท่ากับ FF
- จากโปรแกรมบริการอินเตอร์รัพท์ จะส่งข้อมูลที่อ่านได้จากหน่วยความจำซึ่งก็คือ ค่า FF ไปยังส่วนควบคุมหลอดแอลอีดี
- ส่วนควบคุมหลอดแอลอีดี จะรับค่า FF ที่ส่งมาจากไมโครโพรเซสเซอร์แบบอสมวารแล้วนำมาทำการ XOR กับหลอดแอลอีดีที่สถานะปัจจุบัน ทำให้ หลอดติด 7 ดวงจาก 8 ดวง
- เมื่อหมุนสวิทซ์แบบกดหมุนไปทางซ้าย หลอดแอลอีดีก็จะดับทางซ้ายตามการหมุนและติดหลอดปัจจุบันที่ดับอยู่
- เมื่อหมุนสวิทซ์แบบกดหมุนไปทางขวา หลอดแอลอีดีก็จะดับทางขวาตามการหมุนและติดหลอดปัจจุบันที่ดับอยู่
- เมื่อปล่อยสวิทซ์แอลอีดี จะเหลือหลอดที่ติดอยู่เพียงหลอดเดียว
- สิ้นสุดการทดสอบ



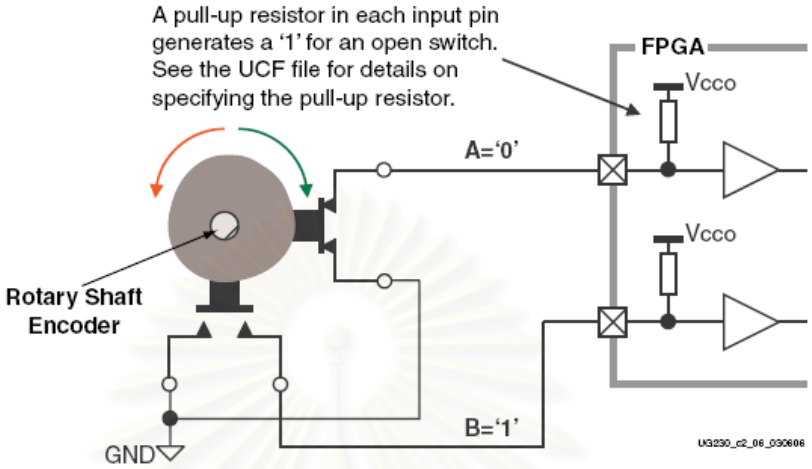
รูปที่ 6.15 วงจรทดสอบที่ประกอบด้วยสวิทซ์แบบกดหมุนกับหลอดแอลอีดี

สวิทซ์แบบหมุนกดประกอบด้วยสองฟังก์ชัน คือ ฟังก์ชันการหมุนซ้ายหมุนขวาและการกด ฟังก์ชันการกดจะทำงานคล้ายกับสวิทซ์แบบกดติดปล่อยดับ การกดสวิทซ์จะทำให้ขาเอฟพีจีเอ ต่อกับแรงดัน 3.3 โวลต์ ดังแสดงในรูปที่ 6.16 โดยการใช้ ตัวต้านทานแบบพูลดาวน์ต่อภายในระหว่างขาเอฟพีจีเอกับสัญญาณกราวนด์



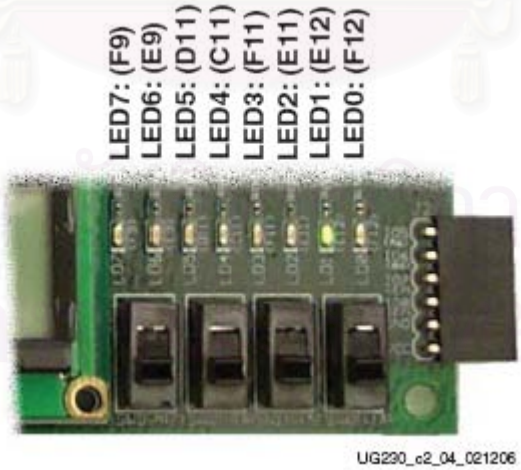
รูปที่ 6.16 วงจรสวิทซ์แบบกดหมุน

ในส่วนของการหมุนซ้ายหมุนขวา จะมีวงจรการถอดรหัส โดยการจับสัญญาณการหมุนซ้ายขวา



รูปที่ 6.17 วงจรถอดรหัสสวิทซ์แบบกดหมุน

เมื่อปิดสวิทซ์จะเป็นการต่อกับสัญญาณกราวนด์ ซึ่งจะสร้างสัญญาณศูนย์ เมื่อสวิทช์เกิดการเปิดสัญญาณ ดังแสดในรูปที่ 6.17 ตัวต้านทานแบบพูลอัพ ที่ต่อกับขาของเอฟพีจีเอ จะทำให้เกิดลอจิก 1 เกิดสัญญาณ 1 ที่ขาเอฟพีจีเอ เพื่อไปควบคุมส่วนที่ต่าง ๆ ตามต้องการ จากรูปจะเห็นว่า A เท่ากับ 0 เมื่อสวิทซ์ปิดวงจร และเท่ากับ 1 เมื่อสวิทช์เปิดวงจร เนื่องจากตัวพูลอัพที่ต่อกับ VCCO นั่นเอง



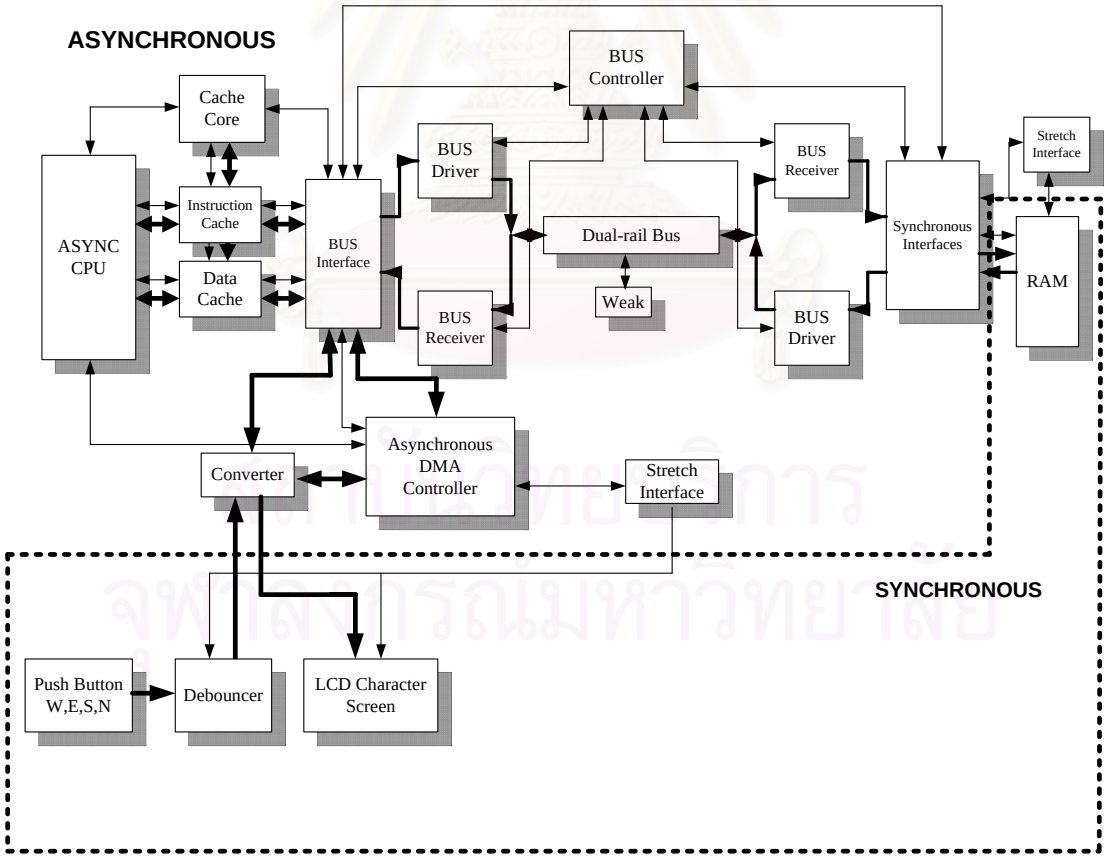
รูปที่ 6.18 หลอดแอลอีดีและตำแหน่งการวางขา

หลอดแอลอีดี ขาข้างหนึ่งจะต่อกับสัญญาณกราวด์ ส่วนอีกข้างจะต่อกับขาของเอฟพีจีเอ ผ่านตัวต้านทาน 390 โอห์มเพื่อลดกระแส การทำให้หลอดแอลอีดีติด ทำได้โดยการป้อนแรงดันทางขาของเอฟพีจีเอ

6.4.5 การทดสอบวงจรกับจอแอลซีดีและสวิทช์แบบกดติดปล่อยดับ

การทำงานของชุดทดสอบชุดนี้ คือ จอแอลซีดีจะแสดงตัวอักษรเมื่อมีการกดสวิทช์แบบกดติดปล่อยดับ ตามลักษณะการทำงานดังนี้ :

- เขียนโปรแกรมคำนวนค่าง่าย ๆ สัก 1 ค่า
- ทำการเก็บค่าของการถอดหรัสตัวอักษรที่จะแสดงบนจอแอลซีดีไว้ในหน่วยความจำ
- เขียนโปรแกรมบริการอินเตอร์รัพท์ไว้ให้ทำดีเอ็มเอโดยเริ่มจากตำแหน่งของหน่วยความจำที่เก็บค่าที่จะถอดรหัสเพื่อแสดงผลบนจอแอลซีดี
- เมื่อทำการกดปุ่มแบบกดติดปล่อยดับ จะเกิดสัญญาณอินเตอร์รัพท์ เพื่อทำดีเอ็มเอโดยนำค่าที่ต้องการแสดงผลมาแสดงบนจอแอลซีดีตามลำดับ
- สิ้นสุดการทดสอบ

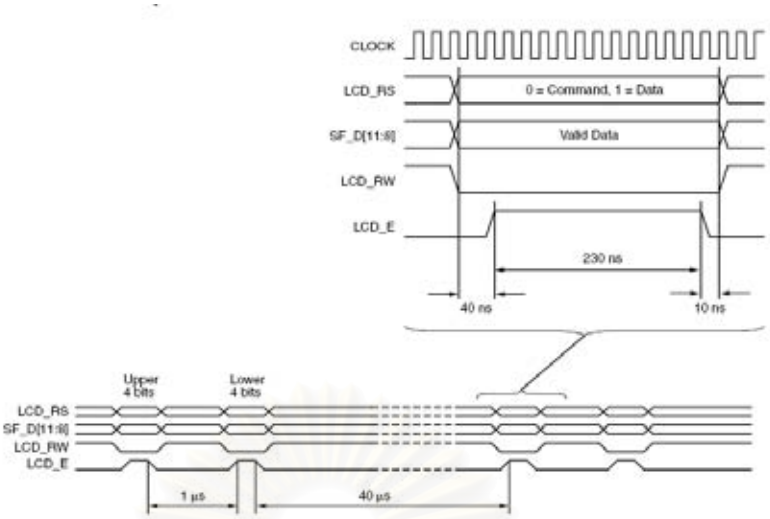


รูปที่ 6.19 วงจรทดสอบที่ประกอบด้วยสวิทช์แบบกดติดปล่อยดับกับจอแอลซีดี

บอร์ดทดลอง Spartan-3E ประกอบด้วยจอแอลซีดีแบบตัวอักษรชนิด 2-บรรทัด ความยาวต่อบรรทัดเท่ากับ 16 ตัวอักษร เอฟพีจีเอจะควบคุมจอแอลซีดีผ่านสายสัญญาณ 4 เส้นเท่านั้น เพื่อลดการใช้ขาเอฟพีจีเอ เนื่องจากบอร์ดทดลองประกอบด้วยอุปกรณ์ต่อพ่วงจำนวนมาก การเชื่อมต่อแบบ 4 บิตทำให้ประหยัดการเชื่อมต่อขาเอฟพีจีเอได้

รูปที่ 6.20 รอมเก็บชุดตัวอักษรสำหรับจอแอลซีดี

การเชื่อมต่อแบบ 4 บิต ในบอร์ดทดลอง Spartan – 3E จะทำงานตามรูปที่ 6.21 แสดงเวลาขั้นต่ำในการเชื่อมต่อกับจอแอลซีดี การส่งข้อมูลจะส่งทีละ 4 บิต สำหรับการเชื่อมต่อแบบ 8 บิต โดย 4 บิตบนจะถูกส่งไปก่อน ตามด้วย 4 บิตล่าง และจะเว้นช่วงเวลา 1us สำหรับการส่งแต่ละครั้ง



รูปที่ 6.21 เวลาการเขียนข้อมูลของจอแอลซีดี

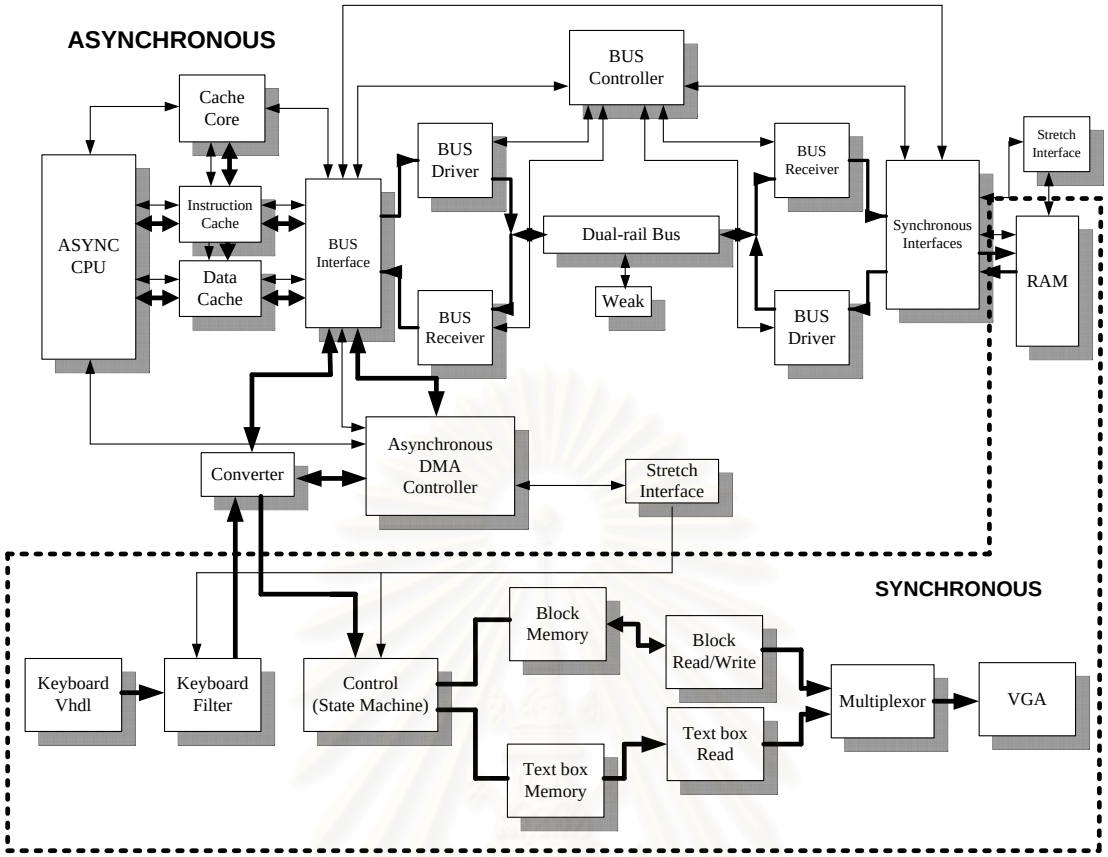
ข้อมูลจากขา SF_D<11:8> สัญญาณ Register Select (LCD_RS) และสัญญาณอ่านเขียนข้อมูล (LCD_RW) จะต้องทำการตั้งค่าภายใน 40 ns ก่อนที่จะให้สัญญาณ LCD_E ซึ่งเป็นสัญญาณเปิดใช้แอลซีดี เป็น 1 สัญญาณ LCD_E จะยังอยู่ที่ค่า 1 จนกว่าเวลาจะล่วงเลยไปอย่างน้อย 230 ns ซึ่งเท่ากับสัญญาณนาฬิกาจำนวน 12 ลูก ที่ความถี่ 50 MHz ของบอร์ดทดลอง ส่วนสัญญาณ LCD_RW จะถูกเซ็ตไว้ที่ 0 ตลอดเวลา

หลังจากทำการตั้งค่าเริ่มต้นสำหรับการเชื่อมต่อแบบ 4 บิตแล้ว จากนั้นจอแอลซีดีพร้อมรับข้อมมูล 4 บิตเพื่อแสดงผลตามที่ต้องการได้ โดยการส่งข้อมูล 2 ครั้งต่อการแสดงผล 1 ตัวอักษร

6.4.6 **การทดสอบวงจรกับจอมอนิเตอร์และคีย์บอร์ดแบบ** PS/2

การทำงานของชุดทดสอบชุดนี้ คือ จอมอนิเตอร์จะแสดงตัวอักษรภาษาไทยทางฝั่งขวา และแสดงการทำงานเนื่องจากการรับคำสั่งจากคีย์บอร์ดดังนี้ :

- เมื่อเปิดเครื่องแล้วจอมอนิเตอร์จะแสดงตัวอักษรภาษาไทยเพื่อแนะนำการเล่นเกมส์ทางด้านขวามือ ส่วนทางด้านซายมือแสดงช่องสำหรับเล่นเกมตัวต่อ
- เมื่อทำการกดปุ่ม F2 เป็นการส่งสัญญาณอินเตอร์รัพท์ไปยังไมโครโพรเซสเซอร์แบบอสมวารเพื่อเริ่มเล่นเกม
- เมื่อทำการกดปุ่ม F1 เป็นการส่งสัญญาณอินเตอร์รัพท์ไปยังไมโครโพรเซสเซอร์แบบอสมวารเพื่อหยุดการเคลื่อนไหว
- เมื่อทำการกดปุ่ม <- เป็นการส่งสัญญาณอินเตอร์รัพท์ไปยังไมโครโพรเซสเซอร์แบบอสมวารเพื่อขยับไปทางซ้าย
- เมื่อทำการกดปุ่ม -> เป็นการส่งสัญญาณอินเตอร์รัพท์ไปยังไมโครโพรเซสเซอร์แบบอสมวารเพื่อขยับไปทางขวา
- เมื่อทำการกดปุ่ม <ลูกศรลง> เป็นการส่งสัญญาณอินเตอร์รัพท์ไปยังไมโครโพรเซสเซอร์แบบอสมวารเพิ่มความเร็ว
- เมื่อทำการกดปุ่ม Space bar เป็นการส่งสัญญาณอินเตอร์รัพท์ไปยังไมโครโพรเซสเซอร์แบบอสมวารเพื่อหมุนวัตถุ
- สิ้นสุดการทดสอบ

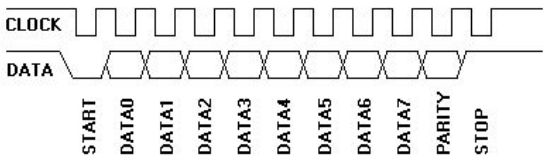


รูปที่ 6.22 วงจรทดสอบที่ประกอบด้วย คีย์บอร์ดแบบ PS/2 กับจอมอนิเตอร์

การออกแบบวงจรทดสอบชุดนี้ใช้พอร์ท PS/2 เพื่อต่อคีย์บอร์ดเป็นอุปกรณ์อินพุท พอร์ท PS/2 ประกอบด้วยสายสัญญาณสองเส้นด้วยกัน คือ สัญญาณนาฬิกาและสัญญาณข้อมูล ใช้ในการติดต่อกับคีย์บอร์ดและพอร์ทมอนิเตอร์ ส่วนพอร์ทมอนิเตอร์ ประกอบด้วยสายสัญญาณ 5 สัญญาณด้วยกัน คือ แดง เขียว น้ำเงิน สัญญาณซิงโครไนซ์แนวตั้ง และสัญญาณซิงโครไนซ์แนวนอน

หน้าจอที่ออกแบบ ประกอบด้วย ด้วยการแสดงผลสองส่วนด้วยกัน คือ ส่วนแสดงผลเกมตัวต่อ และส่วนแสดงผลตัวอักษรภาษาไทย โดยใช้การผนวกกับหน่วยความจำเพื่อใช้ในการแสดงผล ในส่วนของตัวอักษรประกอบด้วย 32x16=512 ตัวอักษร แต่ละตัวจะเก็บเป็น 8 บิตรอม ส่วนหน้าจอเกมประกอบด้วย 10x20 บล๊อก และจัดเก็บในหน่วยความจำขนาด 4 บิต บิตแรกจะเป็นตัวบอกการมีข้อมูล ส่วนบิตที่เหลือ เป็นตัวบอกรหัสสัญญาณสี

ในการเขียนพิกเซลบนจอมอนิเตอร์ ต้องการสัญญาณนาฬิกาที่ความถี่ 25 เมกะเฮิร์ท (1สัญญาณนาฬิกา ต่อ 1พิกเซล) และสัญญาณรหัสสีสามบิต ซึ่งหมายความว่า สามรถแสดงสีบนจอมอนิเตอร์ที่แตกต่างกันได้ ถึง 8 สีด้วยกัน คือ แดง เขียว น้ำเงิน เหลือง ดำ ขาว aqua, fuchsia

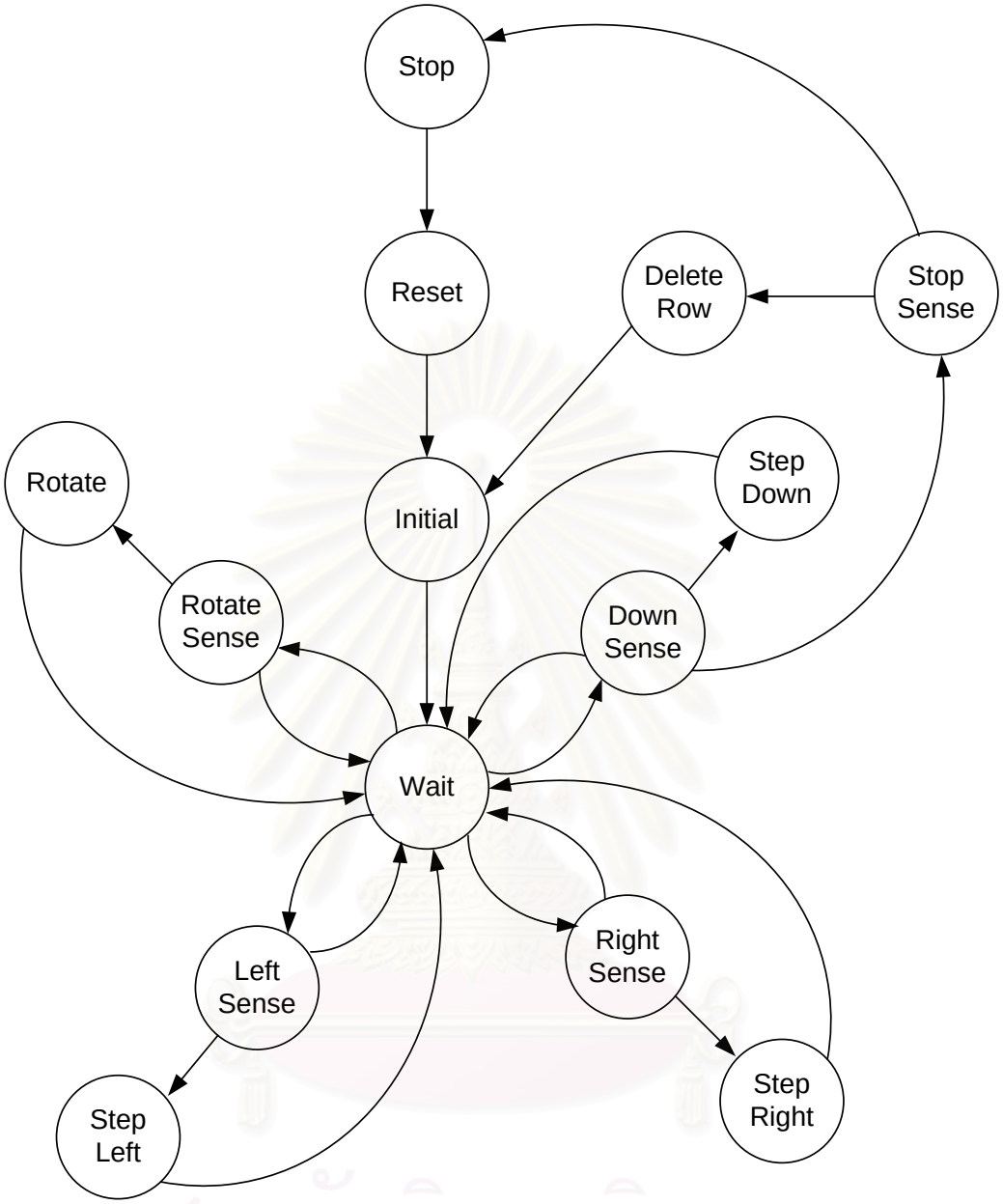


รูปที่ 6.23 การซิงโครไนซ์กับคีย์บอร์ดแบบ PS/2

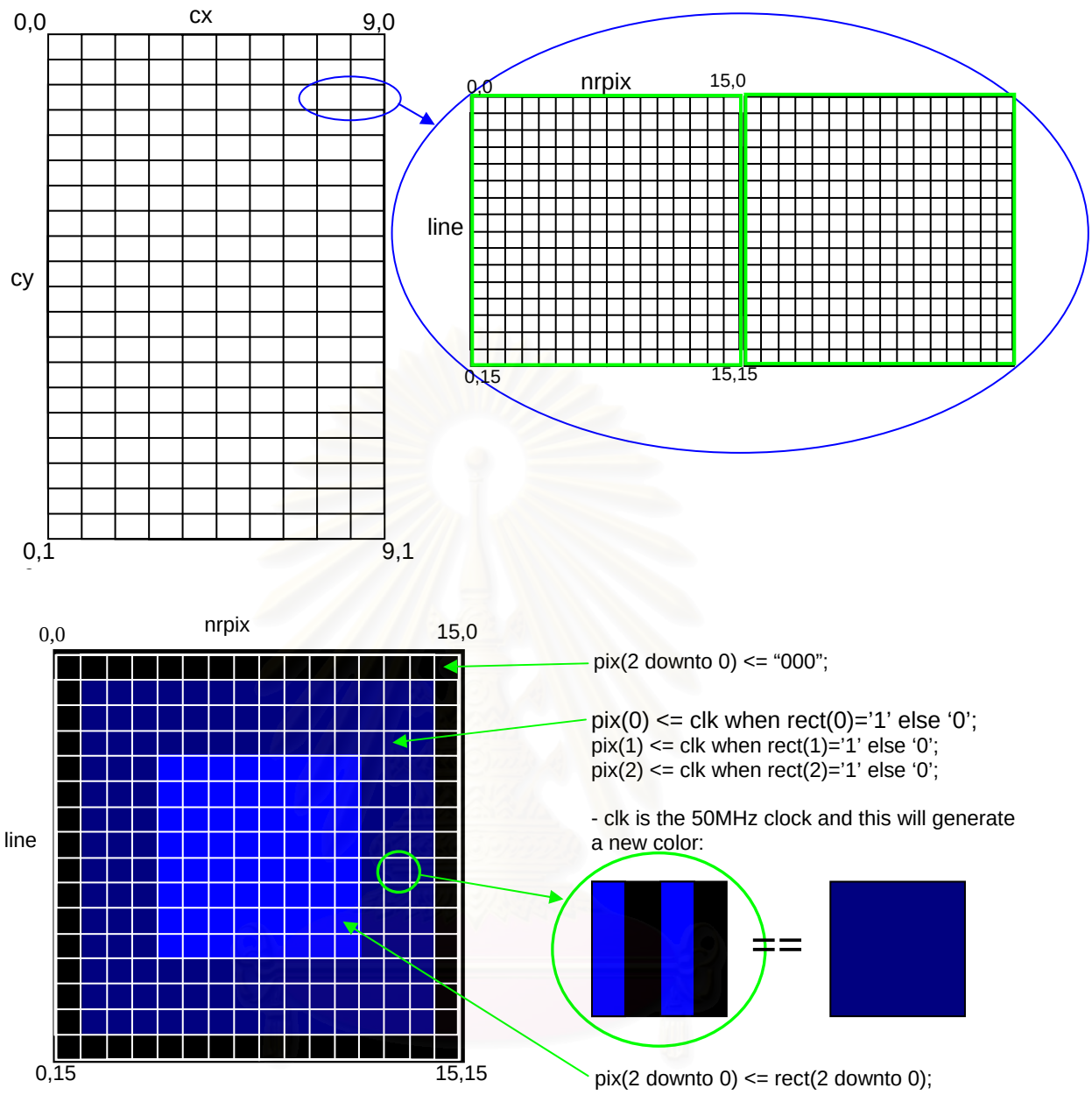
ในการซิงโครไนซ์กับคีย์บอร์ดแบบ PS/2 จะใช้ตัวนับ นับจำนวนสัญญาณนาฬิกา เพื่อรับข้อมูลจากคีย์บอร์ด เมื่อนับสัญญาณนาฬิกาได้ 11 ลูก (สำหรับสัญญาณนาฬิกาความที่ 25 เมกะเฮิร์ท) ข้อมูลที่ส่งมาจากคีย์บอร์ดก็จะถูกอ่านออกมา สัญญาณ “an” จะเปลี่ยนจาก ‘0’ เป็น ‘1’ หรือจาก ‘1’ เป็น ‘0’ เพื่อแสดงถึงการอ่านข้อมูลจากคีย์บอร์ดเสร็จเรียบร้อยแล้ว

ไฟล์ที่ใช้ในการแสดงผลบนจอมอนิเตอร์ ได้ทำการปรับปรุงไฟล์ vga_main.vhd ซึ่งเขียนโดย Barron Barnett เพื่อทำให้เป็นตัวควบคุมจอมอนิเตอร์ตัวใหม่ ประกอบด้วยตัวนับ 2 ตัว ซึ่งนับจาก 0 ถึง 800(hc) และนักจาก 0 ถึง 521(vc) เพื่อทำการสร้างสัญญาณการซิงโครไนซ์ในแนวตั้งและแนวนอน

ตัวนับสองตัว (cx 0 -> 9 and cy 0 -> 19) ซึ่งต่อกับเอาท์พุต “crx” และ “cry” ใช้เพื่ออ่านข้อมูลจาก บล๊อกแรม ของ เกมตัวต่อ แต่ละเลขที่อยู่จะทำการสร้างสายสัญญาณ 4 บิต data(rect(3:0)) โดยบิตแรกจะเป็นตัวบอกการมีอยู่ของข้อมุล และสามบิตที่เหลือจะบอกค่ารหัสสัญญาณสี แต่ละบล๊อกของ tetrix ประกอบด้วย 16x16 พิกเซล ที่จะใช้แสดงบนจอมอนิเตอร์ โดยสัญญาณ “nrpix” (0-> 15) คือเลขคอลัมน์ และ “line” (0-> 15) คือเลขบรรทัด ดังแสดงในรูปที่ 6.25



รูปที่ 6.24: เครื่องจักรสถานะสำหรับวงจรควบคุม



รูปที่ 6.25 การกำหนดพิกเซลสำหรับเขียนข้อมูลบนจอมอนิเตอร์

รูปที่ 26 แสดงการสร้างพิกเซลในการเขียนตัวอักษรลงในช่อง textbox ของจอมอนิเตอร์ ส่วนนี้จะถูกกระตุ้นให้เกิดการทำงานโดยสัญญาณ “text” และสัญญาณ “clkd”(25MHz)

```
type ROM_Array is array (0 to 7) of std_logic_vector(7 downto 0);
  constant A: ROM_Array:=(
          "00111000",
          "01000100",
          "01000100",
          "01000100",
          "01000100",
          "01000100",
          "01000100",
          "00000000");
```

"00111000" "01000100" "01000100" "01000100" "01000100" "01111100" "01000100" "01000100" "00000000"	"01000100" "01000100" "01000100" "01000100" "01000100" "01000100" "00111000" "00000000"	"00111100" "01000010" "01000010" "01110010" "01000010" "01000010" "01000010" "00000000"	"00000110" "00000010" "01000010" "00100010" "00010010" "00001010" "00000110" "00000000"	
"00111000" "00000100" "00000100" "00001100" "00000100" "00000100" "00000100" "00000000"	"00000010" "01100010" "00101100" "00100010" "00100010" "00100010" "00011100" "00000000"	"00000001" "01010001" "00101010" "00010001" "00010001" "00010001" "00001110" "00000000"	"00111100" "01000010" "01000010" "01110010" "01110010" "01100010" "01000010" "00000000"	"00111100" "01000110" "01000110" "01010110" "01110110" "01100110" "01000010" "00000000"	"00111110" "00100001" "00100001" "00100001" "00110001" "00100001" "00111001" "01000010"

รูปที่ 6.26 การเขียนรอมเพื่อเก็บตัวอักษรที่จะแสดงบนจอมอนิเตอร์

6.5 **สรุป**

บทนี้นำเสนอการจำลองการทำงานของวงจรแบบอิงเวลาสำหรับวงจรสมบูรณ์ โดยทำการทดสอบ การอ่านเขียนข้อมูลของบัส การอ่านเขียนข้อมูลของดีเอ็มเอแบบสมวารและอสมวาร จากนั้นทำการเปรียบเทียบระหว่างดีเอ็มเอแบบสมวาร และอสมวาร จากนั้นนำเสนอการโปรแกรมวงจรลงเอฟพีจีเอ เพื่อทดสอบการทำงานจริงบนบอร์ดทดลอง ซึ่งวงจรทดสอบออกแบบเพื่อทดสอบการทำงานไว้ 3 ชุดด้วยกัน คือ วงจรทดสอบที่ประกอบด้วยสวิทซ์แบบกดหมุนกับหลอดแอลอีดี วงจรทดสอบที่ประกอบด้วยสวิทซ์แบบกดติดปล่อยดับกับจอแอลซีดี วงจรทดสอบที่ประกอบด้วยคีย์บอร์ดแบบ PS/2 กับจอมอนิเตอร์

บทที่ 7

สรุปผลการวิจัยและข้อเสนอแนะ

การวิจัยทางด้านไมโครอิเล็กทรอนิกส์ในปัจจุบัน ประกอบด้วยวงจรมากมายหลายชนิด ซึ่งประกอบด้วยทั้งวงจรสมวารและอสมวาร การแลกเปลี่ยนข้อมูลระหว่างวงจรต่าง ๆ มีความซับซ้อนมากยิ่งขึ้นหากใช้การเชื่อมต่อแบบจุดต่อจุด อีกทั้งยังต้องออกแบบส่วนเชื่อมต่อที่ต่างกันสำหรับวงจรที่ต่างกันด้วย ยิ่งไปกว่านั้น หากในระบบที่มีทั้งวงจรสมวารและอสมวารรวมกัน ก็ยิ่งทำให้ความซับซ้อนทวีคูณมากขึ้น ทำให้เกิดการใช้สายสัญญาณอย่างสิ้นเปลือง และเกิดปัญหาเมื่อต้องการปรับปรุงเพื่อพัฒนาต่อไป ดังนั้น งานวิจัยนี้จึงเสนอโปรโตคอลการเชื่อมต่อวงจรเข้าด้วยกันโดยใช้บัส โดยเฉพาะกับระบบที่ประกอบด้วยวงจรสมวารและอสมวาร ที่มีการเชื่อมต่อกับอุปกรณ์ต่อพ่วงจำนวนมาก

เนื่องจากวงจรอสมวารจะไม่ใช้สัญญาณนาฬิกาในการควบคุม และบัสเป็นสายสัญญาณที่ร่วมกันใช้งานของหลาย ๆ อุปกรณ์ อุปกรณ์หลาย ๆ อย่าง ที่ต่อผ่านบัส มีคุณลักษณะทางไฟฟ้าที่ต่างกัน เป็นผลให้เกิดขีดจำกัดในการรับส่งสัญญาณ ผ่านบัสแบบอสมวารอาจส่งผลให้วงจะเกิดการทำงานที่ผิดพลาดได้. ลอจิกสามสถานะ เป็นอุปกรณ์ตัวหนึ่งที่มีส่วนสำคัญในการออกแบบบัสแบบอสมวาร เพื่อใช้ในการกระจายสัญญาณเข้าสู่บัส. ลอจิกสามสถานะประกอบด้วย สถานะ 0, 1 และ ไฮ-อิมพีแดนซ์ สถานะ ไฮ-อิมพีแดนซ์ เป็นสถานะที่ไม่มีการใช้บัส เป็นการจ่ายสัญญาณกระแสสูง จะไม่เป็นทั้งสัญญาณ 0 และ 1 ในการออกแบบวงจรอสมวาร โดยใช้โปรโตคอลอาณัติสัญญาณ แบบ 4 ขั้น ชนิดกลับสู่ศูนย์ ซึ่งในสองขั้นแรก เป็นขั้นการทำงาน ส่วน 2 ขั้นสุดท้ายเป็นขั้นการกลับสัญญาณทุกสัญญาณสู่ศูนย์ ซึ่งการออกแบบวงจรอสมวาร โดยใช้การเข้ารหัสแบบรหัสรางคู่ โดยใช้โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น ชนิดกลับสู่ศูนย์ จะไม่ยินยอมให้สัญญาณเป็น ไฮ-อิมพีแดนซ์ ดังนั้นจึงต้องออกแบบให้วงจรมีสัญญาณ 0 และ 1 เท่านั้นตามข้อกำหนดของการออกแบบวงจรอสมวารดังที่ได้กล่าวไว้ การออกแบบวงจรอสมวาร ผู้ออกแบบยังต้องออกแบบโดยใช้แบบจำลองความหน่วงต่าง ๆ ที่ได้กล่าวไว้ในบทที่ 2 ความซับซ้อนอีกอย่างหนึ่งของการออกแบบบัสแบบอสมวาร คือออกแบบให้วงจรทำงานตามแบบจำลองความหน่วงต่าง ๆ ได้ยาก สำหรับการออกแบบบัสแบบมัลติเพล็กส์ สายสัญญาณเลขที่อยู่ กับสายสัญญาณข้อมูล จะใช้ร่วมกัน ยิ่งทำให้เกิดการจัดการที่ยากยิ่งขึ้น สำหรับกการทำงานแบบอสมวาร

วิทยานิพนธ์ฉบับนี้ นำเสนอการออกแบบบัสแบบอสมวาร เพื่อบริการการเชื่อมต่ออุปกรณ์ต่าง ๆ ที่เป็นทั้งสมวารและอสมวารเข้าด้วยกันได้ แทนการใช้การเชื่อมต่อแบบจุดต่อจุด บัสที่ออกแบบเป็นแบบมัลติเพล็กส์ และสามารถเชื่อมต่อกับวงจรทั้งวงจรสมวารและอสมวาร วงจร

ออกแบบโดยใช้การเข้ารหัสแบบรหัสรางคู่ โดยใช้โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น แบ่งออกเป็น 6 องค์ประกอบด้วยกัน คือ ส่วนเชื่อมต่อบัส ออกแบบเพื่อใช้ในการเชื่อมต่อระหว่างไมโครโพรเซสเซอร์ บัส และ ดีเอ็มเอ, ตัวขับบัส ทำงานคล้ายสวิทช์ เปิด-ปิด ออกแบบโดยใช้ ลอจิกสามสถานะ, สายสัญญาณบัส ออกแบบโดยเพิ่มเกตผกผัน 2 ตัว กลับหัวกัน เพื่อป้องกันไม่ให้เกิดสถานะ ไฮ-อิมพีแดนซ์,ตัวรับบัส ออกแบบโดยใช้อุปกรณ์ชนิดซี, ตัวควบคุมบัส ออกแบบโดยใช้กราฟบรรยายการเปลี่ยนสัญญาณ, ส่วนเชื่อมต่อวงจรสมวารและหน่วยความจำ ออกแบบโดยใช้สัญญาณนาฬิกาแบบยืดหยุ่น.

ดีเอ็มเอ และอินเตอร์รัพท์ ออกแบบเพื่อเป็นส่วนเพิ่มประสิทธิภาพสำหรับบัสแบบอสมวาร ดีเอ็มเอออกแบบเป็นทั้งส่วนที่เป็นวงจรสมวารและวงจรอสมวาร, อินเตอร์รัพท์ ออกแบบโดยการปรับแต่งไมโครโพรเซสเซอร์เพื่อให้รับสัญญาณอินเตอร์รัพท์ได้, ไมโครโพรเซสเซอร์ที่นำมาใช้ในงานวิจัยถูกปรับแต่ง 3 ส่วนใหญ่ ๆ ด้วยกัน คือ การปรับแต่งสถาปัตยกรรม, การปรับแต่งรหัสดำเนินการ และการปรับแต่งส่วนควบคุม

การปรับแต่งสถาปัตยกรรม ปรับแต่งโดยการตัดการเชื่อมต่อไมโครโพรเซสเซอร์ กับ RAM และ ROM โดยเปลี่ยนการเชื่อมต่อไปยัง แคช และ ส่วนเชื่อมต่อบัส, เพิ่ม สถานะ EI Flag เพื่อป้องกันวงจรอินเตอร์รัพท์ และเพิ่ม ตัวตัดสินใจ (Arbiter) เพื่อรับสัญญาณร้องขอจากดีเอ็มเอ เมื่อต้องการใช้บัส

การปรับแต่งรหัสดำเนินการ โดยการออกแบบรหัสดำเนินการเสมือน เพื่อสร้างหรัสดำเนินการใหม่ ในกรณีเกิดสัญญาณอินเตอร์รัพท์จากอุปกรณ์ต่อพ่วง ในการสร้างรหัสดำเนินการนั้น ไมโครโพรเซสเซอร์จะรับสัญญาณอินเตอร์รัพท์กับแอดเรสเวกเตอร์จากอุปกรณ์ต่อพ่วงแล้ว ส่วนรหัสดำเนินการจะทำงานแยกรหัสดำเนินการที่ได้สร้างขึ้นมาเป็นสองส่วน คือส่วนแรก ส่งไปยังส่วนควบคุม ส่วนที่สองส่งไปยังรีจิสเตอร์ต่าง ๆ ตามลักษณะของรหัสดำเนินการ และได้เพิ่มคำสั่ง RETI เข้าไปเพื่อเป็นการรีเทิร์นโปรแกรมบริการอินเตอร์รัพท์

การปรับแต่งส่วนควบคุม เนื่องจากส่วนควบคุมนั้น มีความซับซ้อนมากกว่าส่วนอื่นๆ ผู้วิจัยจึงพยายามหาวิธีการปรับแต่งที่ง่ายที่สุด เพื่อไม่ให้วงจรทำงานผิดพลาด โดยการเพิ่มคอมบิเนชั่นลอจิก เพื่อสร้างสัญญาณร้องขอ *Interrupt_req* สัญญาณร้องขอการร้องขอ *DMA_base_req* และสัญญาณร้องขอ *DMA_Count_req*

การออกแบบแคชแบบอสมวาร เนื่องจากระบบที่ออกแบบจะไม่ใช้รอมในระบบ ผู้วิจัยจึงออกแบบแคช เพื่อเก็บ รหัสดำเนินการ และข้อมูล ที่เป็นแบบอสมวาร ออกแบบโดยใช้พื้นฐานของรีจิสเตอร์แบบอสมวาร และปรับแต่งส่วนควบคุมของวงจรไปป์ไลน์เพื่อใช้ในการป้อนข้อมูลเข้าในตำแหน่งต่าง ๆ ของแคช

ดีเอ็มเอแบบอสมวาร ออกแบบโดยใช้เครื่องจักรสถานะ แบ่งออกเป็น 5 สถานะด้วยกัน หมายความว่า ดีเอ็มแบบอสมวารจะรับส่งข้อมูลขนาด 1 ไบต์ได้ในเวลา 5 สัญญาณนาฬิกา สำหรับไบต์แรก และ 4 สัญญาณนาฬิกาสำหรับไบต์ต่อไป

ดีเอ็มเอแบบอสมวาร ออกแบบโดยใช้การเข้ารหัสกราฟบรรยายการเปลี่ยนสัญญาณ โดยใช้การเข้ารหัสแบบโครงสร้าง การเข้ารหัสแบบโครงสร้างถูกออกแบบเพื่อใช้ออกแบบวงจรควบคุมแบบอสมวารที่มีสัญญาณควบคุมจำนวนมาก, การเข้ารหัสแบบโครงสร้างสามารถป้องกันการเกิดการชนกันของการเปลี่ยนสถานะได้

การจำลองการทำงานโดยอิงเวลา นำเสนอในบทที่ 6 ได้นำเสนอตารางเวลาสำหรับคำสั่งทั้งหมดของไมโครโพรเซสเซอร์ และ และความเร็วโดยรวมของระบบ รวมถึงจำนวนเกตที่ใช้ในการออกแบบวงจรทั้งหมด ในส่วนของการออกแบบดีเอ็มเอ สามารถสรุปได้ดังนี้ ดีเอ็มแบบสมวารที่ออกแบบสามารถทำงานที่ความเร็วสัญญาณนาฬิกาสูงสุดที่ 116 เมกะเฮิร์ท ซึ่งจะมีความเร็วเทียบเคียงกับดีเอ็มแบบอสมวาร ในการรับส่งข้อมูลขนาด 1 ไบต์ เมื่อทำการป้อนสัญญาณนาฬิกาให้กับดีเอ็มเอแบบสมวารขึ้นไปอีก ทำให้ดีเอ็มแบบสมวารยังสามารถทำงานถูกต้องที่ความเร็วสัญญาณนาฬิกา 149 เมกะเฮิร์ท ในระดับการจำลองการทำงาน ทำให้การทำงานของดีเอ็มแบบสมวารเร็วกว่าดีเอ็มเอ็มเอแบบอสมวาร ที่ความเร็วสัญญาณนาฬิกามากกว่า 116 เมกะเฮิร์ท

การโปรแกรมลงเอฟพีจีเอ นำเสนอเพื่อทดสอบการทำงานจริงกับบอร์ดทดลอง ผู้วิจัยได้เลือกบอร์ดทดลอง Spartan-3E ของบริษัท Xilinx เป็นบอร์ดทดลองที่ประกอบด้วยอุปกรณ์ต่อพ่วงอย่างหลากหลาย เหมาะสำหรับนำมาทดสอบกับงานวิจัย ขั้นแรก จัดเก็บโปรแกรมที่คอมไพล์โดย C32 แอสเซมเบลอร์ เข้า Intel Strata Flash จากนั้นนำไฟล์คอนฟิกูเรชั่นของวงจรที่ออกแบบจัดเก็บลง Xilinx Platform Flash การทดสอบ แบ่งออกเป็น 3 ชุดด้วยกัน ชุดแรก ทดสอบกับสวิทช์แบบกดหมุนและหลอดแอลอีดี ชุดที่สองทดสอบกับสวิทช์กดติดปล่อยดับกับจอแอลซีดี ชุดที่สามทดสอบโดยใช้ คีย์บอร์ดแบบ PS/2 และจอมอนิเตอร์

วงจรที่ออกแบบสามารถทำงานได้อย่างถูกต้อง เมื่อเปรียบเทียบกับไมโครโพรเซสเซอร์ที่นำมาอ้างอิงแล้ว สรุปได้ว่า วงจรที่ออกแบบสามารถทำงานได้เร็วกว่าเดิม ประมาณ 18 เปอร์เซ็นต์ และประหยัดจำนวนเกตมากกว่า เนื่องจากอุปกรณ์และซอฟท์แวร์ที่ใช้ในการทดลองใหม่กว่า ส่วนดีเอ็มเอแบบสมวารสามารถทำงานได้ที่ความเร็วสูงสุด 149 Mhz ในระดับจำลองการทำงาน

ข้อเสนอแนะ

สำหรับไมโครโพรเซสเซอร์แบบอสมวาร จะเห็นว่าจำนวนคำสั่งยังน้อยอยู่ ดังนั้นแนวทางการพัฒนาต่อ คือให้เพิ่มคำสั่งของไมโครโพรเซสเซอร์ เช่น คำสั่งการเพิ่มค่า คำสั่งการลดค่า คำสั่งการหมุนบิต เป็นต้น เพื่อให้สามารถโปรแกรมอุปกรณ์ต่อพ่วงได้มากขึ้น เพิ่มหน่วยความจำแบบสแตก เพื่อรองรับการอินเตอร์รัพท์แบบซ้ำซ้อนได้

สำหรับบัสแบบอสมวาร พัฒนาการรับส่งข้อมูลให้ได้หลายๆ บล๊อกและปรับปรุงการจัดการรับส่งข้อมูลให้มีความเร็วมากขึ้น ลดองค์ประกอบและความซับซ้อนของบัสให้น้อยลง โดยเฉพาะ ตัวขับบัส และตัวรับบัส

สำหรับดีเอ็มเอแบบอสมวาร ปรับปรุงให้สามารถทำงานแบบขนานได้ และเพิ่มฟังก์ชัน ให้ให้สามารถทำงานใกล้เคียงกับดีเอ็มเอในปัจจุบันที่เป็นแบบสมวารมากที่สุด

สำหรับแคชแบบอสมวาร หาวิธีการทดแทนข้อมูลที่มีประสิทธิภาพ และ วีธีการป้อนข้อมูลที่รวดเร็วกว่า สำหรับการทำงานแบบอสมวารอย่างสมบูรณ์

รายการอ้างอิง

[1] A.Davis and S.M. Nowick. An Introduction to Asynchronous Circuit Design. University of Utah technical report, Department of Computer Science, UUCS-97-013 (Sept. 19, 1997).

[2] S.Hauck. Asynchronous Design Methhodologies : An Overview. Proceedings of the IEEE. 83(1)(January 1995) :69-93 .

[3] J.Bhasker. A VHDL Primer. Prentice-Hall, 1992.

[4] R.E. Miller. Combinational Circuit. Volume1 of Switching Theory, 1965.

[5] P.Molina, P.Cheung and D.Bormann. Quasi Delay-insensitive Bus for Fully Asynchronous Systems. in Proceeding of the 1996 IEEE International Symposium on Circuits and System. May 1996.

[6] P.A. Molina, P.Y.K Cheung. A Quasi Delay-Insensitive Bus Proposal for Asynchronous Systems. Proceedings Third International Symposium on Advanced Research in Asynchronous Circuits and Systems. 1997 : 129-139.

[7] A.Takamura, M.Kuwao, M.Imai, T.Fuji, M.Ozawa, Y.Ueno, T.Nanya. TITAC-2: An asynchronous 32 bit microprocessor based on Scalable-Delay-Insensitive Model. IEEE Proceedings International Conference on Computer Design : VLSI in Computer and Processor. 1977.

[8] T.Nanya, Y.Ueno, H.Kagotani, M.Kuwako, A.Takamura. TITAC : Design of a Quasi-Delay-Insensitive Microprocessor. IEEE Design & Test of Computers. 11(2)(Summer 1994) : 50-63.

[9] D.S Bormann, P.A. Molina, P.Y.K Cheung. Combining asynchronous and synchronous circuits using stretchable clocks. IEEE Colloquium on Design and Test of Asynchronous Systems, (28 Feb 1996) : 4/1-4/8.

[10] P.Ruangsilsap. Design of 8-bit scalable-delay-insensitive microprocessor using FPGA. Master thesis, Chulalongkorn University, Bangkok, Thailand, 2001.

[11] W. J. Bainbridge, Stephen B. Furber. Asynchronous Macrocell Interconnect using MARBLE. Proceedings 4th International symposium on Asynchronous Circuits and Systems (ASYNC'98). 1998 :122-132.

[12] J. Carmona, J. Cortadella. State encoding of large asynchronous controllers. In Proc. ACM/IEEE Design Automation Conference. (July 2006) : 939-944.

[13] J. Cortadella, K. Gorgônio, F. Xia, and A. Yakovlev. Automatic synthesis of asynchronous coomunication mechanisms. In Int. Conf. on Application of Concurrency to System Design. (June 2005) : 166-175.

[14] J. Cortadella, A. Kondratyev, L. Lavagno, K. Lwin, and C. Sotiriou. From synchronous to asynchronous: An automatic approach. In Proc. Design, Automation and Test in Europe (DATE). 2 (Febuary 2004) : 1368-1369.

[15] J. Carmona, J. Cortadella. ILP models for the synthesis of asynchronous control circuits. In Proc. International Conf. Computer-Aided Design (ICCAD). (November 2003) : 818-825.

[16] J.L.Hennessy and D. A. Patterson. Computer architecture a quantitative Approach. Morgan Kaufman publishers, Sanfrancisco, California,1996.

[17] W.Stalling. Computer Organization & Architecture ,Designing for Performance. Sixth Edition, New Jersey, Prentice Hall, 2003.

[18] D.A. Patterson and J.L. Hennessy. Computer Organization & Design , Hardware/Software Interface. Second Edition, San Francisco, Morgan Kaufman Publishers, 1997.

[19] Xilinx Incorporation. Xilinx : ISE Foundation[Online]. Available from : http://www.xilinx.com /ise/logic_design_prod/foundation.htm (2007, March 20)

[20] Mentor Graphic Corporation. ModelSim[Online]. Available from : http://www.model.com (2007, March 20)

ภาคผนวก

ภาคผนวก ก

โปรแกรมทดสอบ

โปรแกรมทดสอบการอ่านเขียนข้อมูลของบัสแบบอสมวาร

```
ORG   0000H
LD    A,#77   ; A=77H
ST    A,R0    ; R0=77H
LD    A,#24H  ; A=24H
ST    A,R1    ; R1=24H
ST    A,@R0   ; M(77H) = 24H
LD    A,R0    ; A=77H
ST    A,@13H ; M(13H) =77H
LD    A,@77H ; A= 24H
ST    A,@R1          ; M(24H) = 24H
LD    A,#13H  ; A=13H
ST    A,R1    ; R1=13H
LD    A,@R1  ; A= 77H
HALT :
JMP   HALT
END
```

โปรแกรมทดสอบอินเตอร์รัพท์และดีเอ็มเอ

```
ORG   0000H
LD    A,#39H; A = 39H
ST    A,R0    ; R0 = 39H
LD    A,#24H; A = 24H
ST    A,R1    ; R1 = 24H
AND   A,#14H; A= 04H
LD    A,#C8H          ; A = C8H
ADD   A,R0    ; A = 01H, CARRY = 1
ST    A,R1    ; R1 = 01H
OR    A,R0    ; A = 39H
OR    A,#B7H          ; A = BFH
LD    A,#13H; A = 13H
ST    A,@R1  ; M(01) = 13H
LD    A,#00H; A=00H
INTR :
LD    A,#01H ; A=01H
ST    A,@1FFH ; DMA_Base = 01H
LD    A,#0AH          ; A=0AH
ST    A,@2FFH ; DMA_Count = 0AH
RETI
END
```

โปรแกรมทดสอบการทำงานจริงกับบอร์ดทดลอง

```
ORG   0000H
;INITIAL
-------------------------------------------------
LD    A,#54H; A = 54H
ST    A,@01H; M(01H) = 54H
LD    A,#44H; A = 44H
ST    A,@02H; M(02H) = 44H
LD    A,#45H; A = 45H
ST    A,@03H; M(03H) = 45H
LD    A,#20H; A = 20H
ST    A,@04H; M(04H) = 20H
LD    A,#53H; A = 53H
ST    A,@05H; M(05H) = 53H
LD    A,#4FH; A = 4FH
ST    A,@06H; M(06H) = 4FH
LD    A,#4CH; A = 4CH
ST    A,@07H; M(07H) = 4CH
LD    A,#55H; A = 55H
ST    A,@08H; M(08H) = 55H
LD    A,#54H; A = 54H
ST    A,@09H; M(09H) = 54H
LD    A,#49H; A = 49H
ST    A,@0AH; M(0AH) = 49H
LD    A,#4FH; A = 4FH
ST    A,@0BH; M(0BH) = 4FH
LD    A,#4EH; A = 4EH
ST    A,@0CH; M(0CH) = 4EH
LD    A,#20H; A = 20H
ST    A,@0DH; M(0DH) = 20H
LD    A,#4FH; A = 4FH
ST    A,@0EH; M(0EH) = 4FH
LD    A,#46H; A = 46H
ST    A,@0FH; M(0FH) = 46H
LD    A,#4CH; A = 4CH; <BR>
ST    A,@10H; M(10H) = 4CH
LD    A,#33H; A = 33H
ST    A,@11H; M(11H) = 33H
LD    A,#2BH; A = 2BH
ST    A,@12H; M(12H) = 2BH
LD    A,#35H; A = 35H
ST    A,@13H; M(13H) = 35H
LD    A,#20H; A = 20H
ST    A,@14H; M(14H) = 20H
LD    A,#69H; A = 69H
ST    A,@15H; M(15H) = 69H
LD    A,#73H; A = 73H
ST    A,@16H; M(16H) = 73H
LD    A,#20H; A = 20H
ST    A,@17H; M(17H) = 20H
LD    A,#38H; A = 38H
ST    A,@18H; M(18H) = 38H

WAIT:
JMP   WAIT

INTR_ROTARY:
LD    A, #FFH
ST    A, @3FFH
RET I
INTR_LCD:
```

```
LD      A, #03H
ST      A, R1
LD      A,#05H
ST      A, R2
LD      A, #00H
ADD     A, R1
ADD     A, R2
ST      A, @12H
LD      A, #03H
ST      A, @13H
LD      A, #05H
ST      A, @14H
LD      A,#10H ; A=10H
ST      A, @1FFH; DMA_Base = 10H
LD      A,#0FH ; A=0FH
ST      A, @2FFH; DMA_Count = 0FH
RETI
END
INTR_F1:
LD      A, #05H;SCANCODE F1
ST      A,@5FFH
RETI
INTR_F2:
LD      A,#06H;SCANCODE F2
ST      A,@5FFH
RETI
INTR_LEFT:
LD      A, #6BH;SCANCODE LEFT
ST      A, @5FFH
RETI
INTR_RIGHT:
LD      A, #74H;SCANCODE RIGHT
ST      A, @5FFH
RETI
INTR_DOWN:
LD      A, #72;SCANCODE DOWN
ST      A, @5FFH
RETI
INTR_ROT:
LD      A, #29;SCANCODE_ROTATE
ST      A,@5FFH
RETI
```

ภาคผนวก ข

คำแปลศัพท์ที่ใช้ในวิทยานิพนธ์

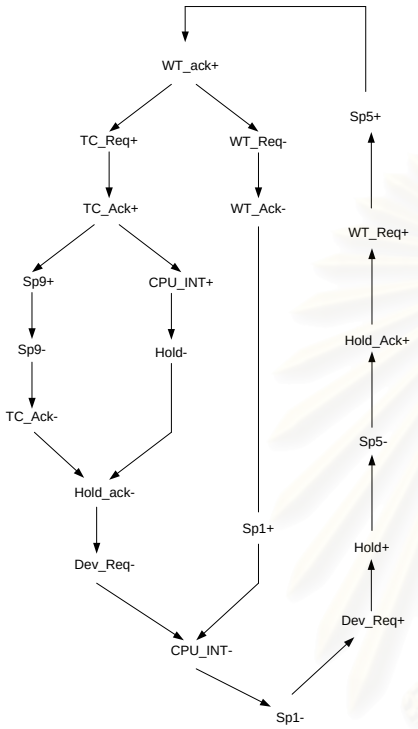
ภาษาไทย	ภาษาอังกฤษ
การแกว่งของสัญญาณนาฬิกา	Clock Skew
การทำงานที่ช้าที่สุด	Worse Case
โปรโตคอลอาณัติสัญญาณแบบ 4 ขั้น	4-Phase Signaling Protocol
การเปลี่ยนระดับสัญญาณ	Transition Signal
การสังเคราะห์	Synthesis
การอิมพลีเม้นท์	Implement
เครื่องจักรสถานะ	State Machine
กราฟบรรยายการเปลี่ยนสัญญาณ	Signal Transition Graph
เกต	Gate
เกตออร์	OR Gate
เกตแอนด์	AND Gate
เกตผกผัน	INVERTER Gate
ขั้นตอนวิธี	Algorithm
ขั้นทำงาน	Working Phase
ขั้นว่าง	Spacer Phase
แบบจำลองการทำงานสิ่งแวดล้อม	Environment Operation Model
สภาวะแวดล้อมมูลฐาน	Fundamental Mode
สภาวะแวดล้อมรับเข้า-ส่งออก	Input-Output Mode
แบบจำลองความหน่วง	Delay Model
แบบจำลองความหน่วงที่ไม่ขึ้นกับความเร็ว	Speed Independent Delay Model
แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วง	Delay Insensitive Delay Model
แบบจำลองความหน่วงที่ไม่ไวต่อความหน่วงชนิดเสมือน	Quasi-Delay Insensitive Delay Model
ช่องทางการรับส่งข้อมูล	Channel
โปรโตคอลอาณัติสัญญาณ	Signaling Protocol
วงจรเชิงผสมแบบอสมวาร	Asynchronous Combinational Circuits

วงจรเชิงลำดับ	Sequential Circuits
สัญญาณตอบรับ	Acknowledge Signal
สัญญาณร้องขอ	Request Signal
บัส	Bus
บัสแบบอสมวาร	Asynchronous Bus
ตัวขับบัส	Bus Driver
ตัวรับบัส	Bus Receiver
ตัวควบคุมบัส	Bus Controller
ตัวตัดสินใจ	Arbiter
สายสัญญาณบัส	Bus Lines
ส่วนเชื่อมต่อวงจรสมวาร	Synchronous Interfacing
สัญญาณนาฬิกาแบบยืดหยุ่น	Stretchable Clock
วงจรรางคู่	Dual-rail Code
วงจรอสมวาร	Asynchronous Circuits
วงจรสมวาร	Synchronous Circuits
ดีเอ็มเอ	DMA Controller
ดีเอ็มเอแบบสมวาร	Synchronous DMA Controller
ดีเอ็มเอแบบอสมวาร	Asynchronous DMA Controller
อินเตอร์รัพท์	Interrupt
การจำลองการทำงานแบบอิงเวลา	Time Simulation
สัญญาณแสดงความบริบูรณ์	Completion Signal
วงจรสมบูรณ์	Full System Integration
ส่วนควบคุม	Control Unit
รหัสดำเนินการ	Instruction
อุปกรณ์ซนิดซี	C-element
อินพุต	Input
เอาท์พุต	Output
อุปกรณ์ต่อพ่วง	Input /Output
เอฟพีจีเอ	FPGA
ไมโครโพรเซสเซอร์	Microprocessor
การเข้ารหัสแบบโครงสร้าง	Structural Encoding

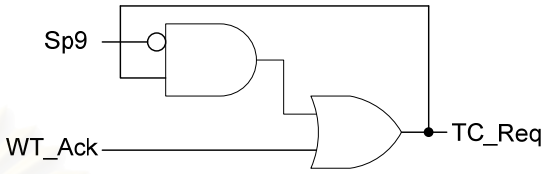
สวิทซ์แบบกดหมุน	Rotary Knob Switch
สวิทซ์แบบกดติดปล่อยดับ	Push Button
คีย์บอร์ด	Keyboard
จอแอลซีดี	LCD
หลอดแอลอีดี	LED
ผังคาร์โนห์	Kanaugh Map
เพลส	Place
การปรับแต่ง	Modification

ภาคผนวก ค

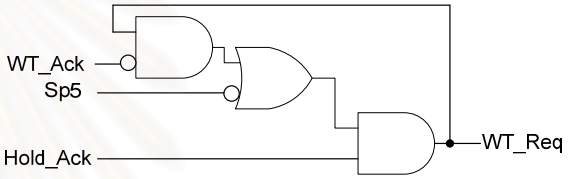
กราฟบรรยายการเปลี่ยนสัญญาณและวงจรของดีเอ็มเอแบบอสมวาร



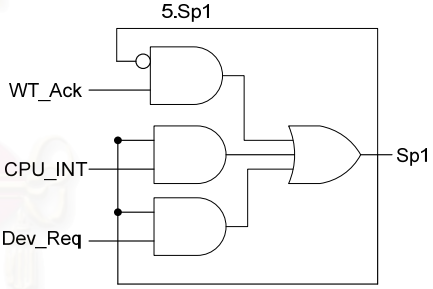
3.TC_Req



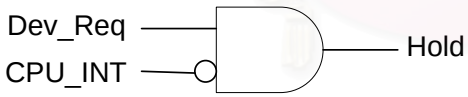
4.WT_Req



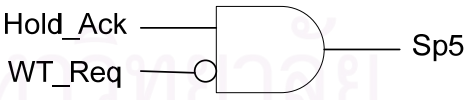
5.Sp1



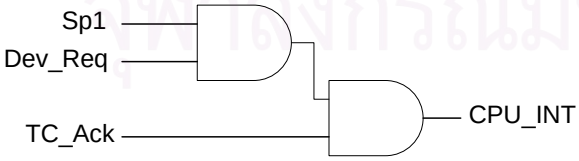
1.Hold



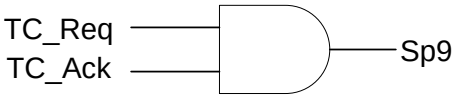
6.Sp5



2.CPU_INT



7.Sp9



ภาคผนวก ง

ผลงานที่ตีพิมพ์จากงานวิจัย

ส่วนหนึ่งของวิทยานิพนธ์ได้ตีพิมพ์เป็นบทความทางวิชาการในหัวข้อ “A Design of System Bus for Asynchronous Circuits” โดย สุฟียัน สูเด็ง และ อาทิตย์ ทองทักษ์ ในงานประชุมวิชาการ “International Technical Conference in Circuits/Systems Computer and Communication (ITC-CSCC2006)” ณ โรงแรมโลตัสปางสวนแก้ว จ.เชียงใหม่, ประเทศไทย ระหว่างวันที่ 10-13 กรกฎาคม 2549

ส่วนหนึ่งของวิทยานิพนธ์ได้ตีพิมพ์เป็นบทความทางวิชาการในหัวข้อ “Asynchronous System Bus Enhancement by Interrupt and DMA technique” โดย สุฟียัน สูเด็ง และอาทิตย์ ทองทักษ์ ในงานประชุมวิชาการ “The fourth Annual International Conference (ECTI-CON 2007)” ซึ่งจัดโดย Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI) Association ณ มหาวิทยาลัยแม่ฟ้าหลวง จ.เชียงราย ประเทศไทย, ระหว่างวันที่ 9-12 พฤษภาคม 2550

ง.1 ส่วนหนึ่งของวิทยานิพนธ์ที่ได้ตีพิมพ์ในงานประชุมวิชาการ

International Technical Conference in Circuits/Systems Computer and Communication (ITC-CSCC2006)

A Design of System Bus for Asynchronous Circuits

Sufian Sudeng and Arthit Thongtak
Digital System Engineering Laboratory (DSEL), Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
254 Phyathai Road, Pathumwan, Bangkok, Thailand 10330
Tel: +66-2-218-6956, Fax: +66-2-218-6955
E-mail: Sufian.S@Student.chula.ac.th, Arthit@cp.eng.chula.ac.th

ABSTRACT

This paper proposes a method to design an asynchronous system bus to connect individual device or circuits with common bus instead of point-to-point connection. The propose scheme are multiplex and be able to interface with both synchronous and asynchronous circuits. The combinational circuits are design with dual-rail code 4-phase signaling protocol. The design is separated to 6 components .There are Bus Interface use to connect with processor, Bus driver act like on/off switch design with tri-state buffer, Bus component a bi-directional line weak inverter attached on each line to remove high impedance state, Bus receiver design with C-element, Bus controller design with STG, synchronous and memory interface design with stretchable clock for communicate with synchronous system. The experimental result read/write simulations of asynchronous system bus are explained.
Keywords: Asynchronous System Bus, Dual-rail code, Tri-state buffer, Stretchable clock

1. INTRODUCTION

Due to rapid development of micro electronics technology cause on circuits design are so small and high-speed communications. The Old circuits design methods are synchronous design cannot serve the ability to correct on the high frequency clock. So it's proposed a clock skew which cannot distribute all high frequency signals on the system[1]. On synchronous circuits generate a worse case delay propagation due to use of global clock for control each part of the circuits. Its make much of waiting time for processing all parts and much more energy consumption on clock distributed[2]. Causes of this problem, asynchronous design method are considered. The asynchronous methods do not need a global clock for controlled the part of circuits to behave on each other. There are depending on Event-Driven behaviour[1]. Speeds of circuits do not reduce by clock. More outstanding shot of asynchronous circuits over synchronous circuits are average case delay; consume less energy, robust and etc.

The forward research on micro electronics now a day there are a lot of circuit parts on one system. The transfer of data between parts of system are more complex if use point-to-point connection method. Its waste more signals and future upgrade problem on individual protocol. So Bus is the Bright way to solve it. We need to use BUS instead of point-to-pint connection. On synchronous system tri-state buffer are choose. It's act like a switch to enable or disable to drive the BUS, on asynchronous need some circuits to remove high impedance state of tri-state buffer. MARBLE is one of famous asynchronous version of bus, a two channel micropipeline bus with bundle data interface, bus bridging support and test interface demonstration, used in AMULET 3i processor[11].

This paper proposed a method to design an asynchronous multiplex system bus which Operate asynchrony based on handshaking method instead of clock method. The proposed bus also can communicate with synchronous world by adding a stretchable clock technique. There is one technique to interface between synchronous and asynchronous circuits. All codes are design with VHDL[3][4] and ModelSim Tools for result simulation.

2. LIMITATION OF ASYNCHRONOUS BUS DESIGN

Due to the absence of global clock on asynchronous system and the design of bus are share communication link. Many components that attach to the bus are strange and different on load capacitance. This cause lack of signal distribution on bus make a circuit worse or wrong behaviour, Tri-state buffer are one of element that require on Bus design for enable/disable signal to distribute on bus. It has 3 states. 0, 1 and high impedance, High impedance state are state disconnect or do not use bus. For asynchronous design there is one of most famous protocol so call 4-phase protocol. There are first 2 phases are working phase and last 2 phases are idle phase or return to zero phase. But an asynchronous dual-rail design not allowed being high impedance. It can be only one or zero. It's one problem cause to hard design asynchronous bus. We need to solve for reject high impedance state to serve robust asynchronous design. On asynchronous circuit, a designer need to design based on delay model assumption for well formed operation. Asynchronous bus designs are difficult to design under various delay assumptions[5][6]. For multiplex bus design it uses the same lines on data and address phase, it's also hard to design the controller for asynchrony operation.

3. ASYNCHRONOUS COMBINATIONAL CIRCUITS DESIGN

3.1 Signalling Protocol

Signalling protocol that use for our research are 4- phase protocol

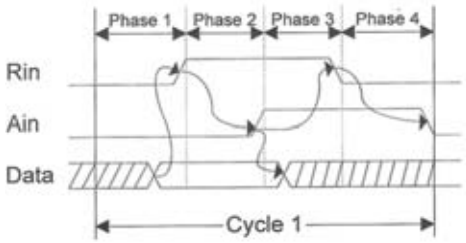


Fig.1: *4-phase protocol[1]*

Fig1 show timing diagram of 4-phase protocol. The free style black arrows are the sequence of signals changing. 4- Phase protocol contain 4-transition behaviour (2 phase for working phase and 2-phase for idle phase)

3.2 Dual-rail Code

There are known problem on asynchronous system design when it's use only one line to transfer to each other part of circuits. There are no conflicts when transition follows on 1=>0 and 0=>1. But there exist a problem for 0=>0 and 1=>1 transition. It's ambiguous for the previous and current signal transition. Dual-rail Codes are considered for encoded. One signal contains 2 line signals. The added signal call check bit. Thus operate on good and health condition[2].

(0, 1) ➔ are for 0 signal (working phase)
(1,0) ➔ are for 1 signal (working phase)
(0, 0) ➔ are spacer (Idle phase)

3.3 Delay Model

Each gate that make combinational digital circuits, all of it contain delay variation according to physical gate property. On circuits making process, it might generate delay too. From fabrication process, temperature and voltage, on synchronous system physical delays are rejects. Because it has a global clock, but an asynchronous system are use handshake communication, Delay model are the assumption of delay by various assumption for design robust asynchronous circuits[7][8].

QDI or Quasi-Delay insensitive is earned for our research. QDI assumption is unbounded gate and wire delay but finite, with Isochronick fork assumption added. There assume all delay on neighbours wire are equal.

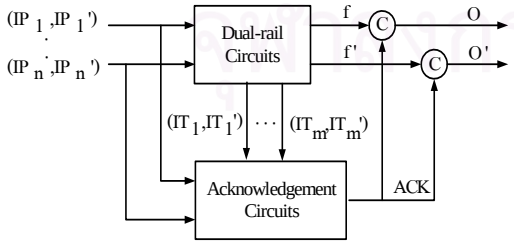


Fig.2: *Asynchronous Combinational Circuits design*[10]

4. A DESIGN OF ASYNCHRONOUS SYSTEM BUS FOR ASYNCHRONOUS CIRCUITS

Our bus designs are Multiplex. There are data and address transition on the same lines. Our System bus are 8-bit encoded with dual-rail code transition by 4-phase signalling transition protocol. The processes are beginning with design each component and group them together.

4.1 Bus Interface

Bus interface is design for interfacing with asynchronous processor. Due to the multiplex bus design, bus interface also handshaking with bus controller for controlled the sequence of data, address or spacer transition.

4.2 Bus Driver

Bus Driver act like an on/off switch to enable/disable transition signal on the bus. Design with tri-state buffer and C-element (the element that produce output 1 when all of input are 1, produce out 0 when all of input are 0, Otherwise its hold previous state)

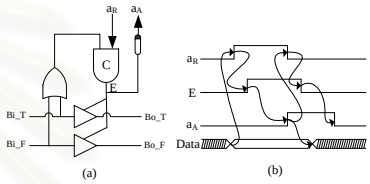


Fig.3: *Bus Driver[5]*

a_R is request line from bus controller when data arrive at Bi_T and Bi_F , C-element is enable tri-state buffer to transfer signal to Bo_T and Bo_F , a_A is the acknowledgement signal to bus controller.

4.3 Bus Component

Due to High impedance state of Tri-state buffer, there were making some problem for asynchronous circuit. Need to improve to be zero value when it's high impedance by added weak inverter with every line on bus lines.

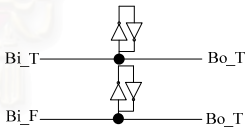


Fig.4: *Bus lines with weak-inverter*

4.4 Bus Receiver

This component for receive and catch all signal before transfer to the destination, Design with C-element

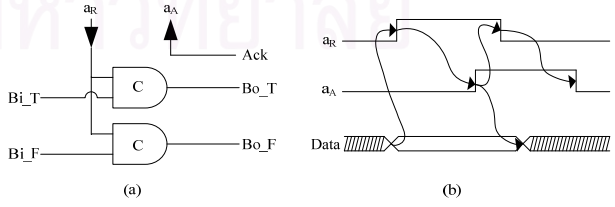


Fig.5: *Bus Receiver and its transition signal*

a_R is request line from bus controller, when data arrive at Bi_T and Bi_F , a_R is transition from 0 ➔1 C-element enable signal transition to Bo_T and Bo_F, a_A is the acknowledgement signal form synchronous interface to bus controller.

4.5 Bus Controller

Design with STG (Signal Transition Graph)

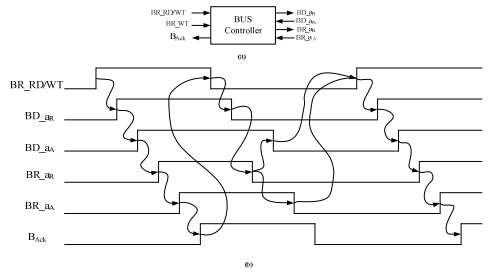


Fig.6: *Bus controller specification and its transition signal*

BR_RD/WT is read or write signal from processor, when BR_RD/WT transition from 0 ➔1, BD_a_R , BD_a_A , BR_a_A , BR_a_A , B_{ack} are transition from 0 ➔1 by ordered, then all signal transition from 1 ➔0 for return to zero indicate the completion of address phase, as same as the data phase.

4.6 Synchronous and Memory Interface

This part is design for communicates with Synchronous Memory or devices design with stretchable clock[9].

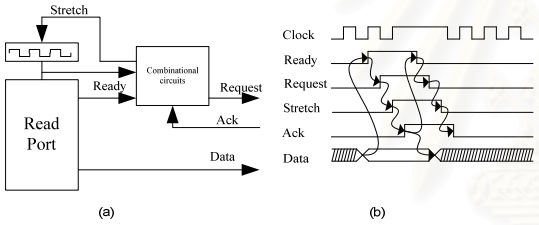


Fig.7: *Stretchable clock for read transaction*

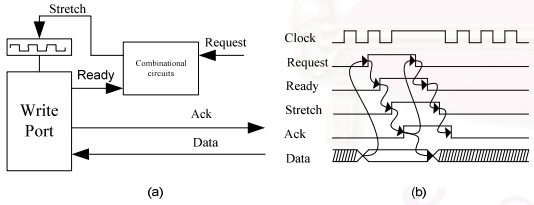


Fig.8: *Stretchable clock for write transaction*

Bus Transaction

Our Multiplex bus transaction can categorized in to two types, there are read and write transaction, write transaction is shown

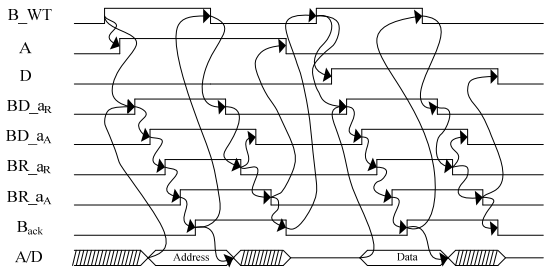


Fig.9: *Bus writes transaction*

For write transaction B_WT signal is write signal transition from 0→1 then signal A is the address signal transition from 0→1 indicate on the address phase

BD_a_R, BD_a_A, BR_a_R, BR_a_A are the control signals from bus controller transition from 0→1 by ordered. Indicate the flowing through of data form bus driver to bus receiver components. B_{ack} signal transition from 0→1 indicate the completion of data transfer of address phase. Then all signals transition from 1→0 for resetting. After that signal D transition from 0→1 for data phase.

BD_a_R, BD_a_A, BR_a_R, BR_a_A signals transition from 0→1 in ordered. B_{ack} transition from 0→1 indicates the completion of data transfer. After that all signal transition from 1→0 for signals resetting

5. SIMULATION RESULT

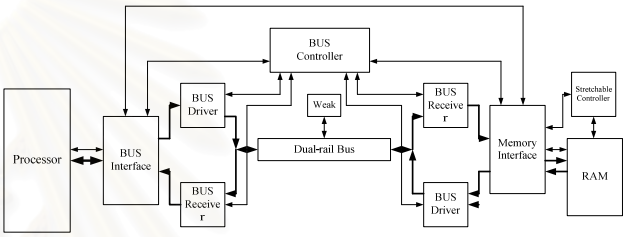


Fig.10 : *Simulation Environment*

Our system is design with VHDL description language separately on each component and simulates each component for correct signal behaviour. Then combined those together to form an asynchronous system bus and connect with asynchronous processor and 8-bit wide synchronous RAM .after that read/write experimental simulation are shown.

The load/store ROM for read/write operation of processor is

```
        ORG     0000H
        LD      A,#77    ; A=77H
        ST      A,R0     ; R0=77H
        LD      A,#24H   ; A=24H
        ST      A,R1     ; R1=24H
        ST      A,@R0    ; M(77H) = 24H
        LD      A,R0     ; A=77H
        ST      A,@13H ; M(13H) =77H
        LD      A,@77H ; A= 24H
        ST      A,@R1  ; M(24H) = 24H
        LD      A,#13H   ; A=13H
        ST      A,R1     ; R1=13H
        LD      A,@R1  ; A= 77H
HALT :
        JMP     HALT
        END
```

6. CONCLUSION

This paper proposes a method to design system bus for asynchronous circuits. The system bus design is multiplex and be able to interface with both synchronous and asynchronous circuits. The combinational circuits are design with dual-rail code 4-phase signalling protocol. the

design are separate to 6 components there are Bus Interface to interface with processor, Bus driver design with tri-state buffer, Bus component a bi-directional line weak inverter attached on each line, Bus receiver design with C-element, Bus controller design with STG and synchronous and memory interface design with stretchable clock. For future works:

- Improve for ability to connect with Input/Output devices
- Add some bus speed increasing technique such as Interrupt and DMA

7. ACKNOWLEDGEMENT

The research is supported by TJTTP - OECF (Thailand - Japan Technology Transfer Project - Japanese Overseas Economic Cooperation Fund)

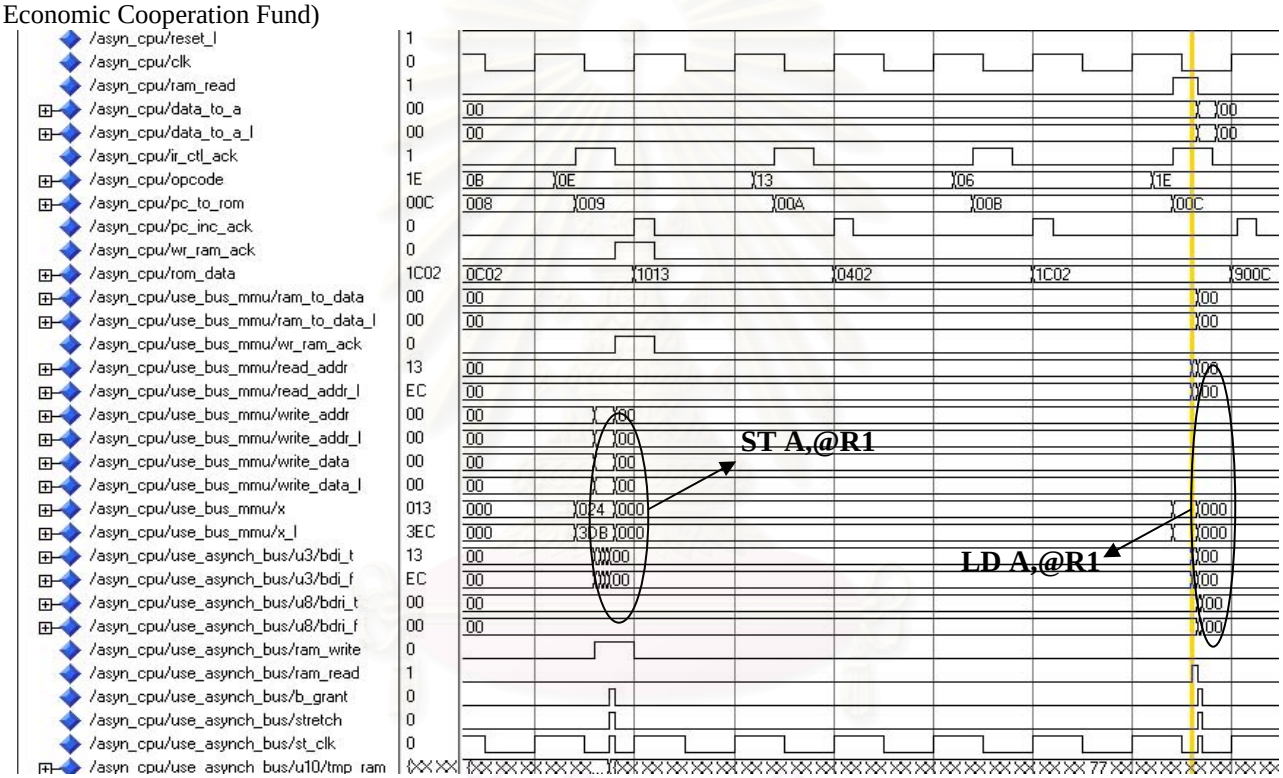


Fig.11: *Experimental simulation result*

8. REFERENCES

[1] Davis. A, Nowick.S.M "An Introduction to Asynchronous Circuit Design" , 1997

[2] S.Hauck. "Asynchronous Design Methhodologies : An Overview". Proceedings of the IEEE. vol.83. No.1. pp.69-93, January 1995.

[3] J.Bhasker. "*A VHDL Primer*". Prentice-Hall, Inc., 1992. ISBN 0-13-952987-X.

[4] R.E. Miller. "*Combinational Circuit"*, Volume1 of *Switching Theory*. John Wiley & Sons,1965.

[5] P.Molina, P.Cheung and D.Bormann, "Quasi Delay-insensitive Bus for Fully Asynchronous Systems", in Proceeding of the 1996 IEEE International Symposium on Circuits and System,May 1996

[6] Molina, P.A.; Cheung, P.Y.K ,"A Quasi Delay-Insensitive Bus Proposal for Asynchronous Systems" Proceedings Third International Symposium on Advanced Research in Asynchronous Circuits and Systms, 1997 Pages:129-139

[7] A.Takamura., M.Kuwao., M.Imai., T.Fuji., M.Ozawa., Y.Ueno., T.Nanya. "TITAC-2: An asynchronous 32 bit microprocessor based on Scalable-Delay-Insensitive model". Presented at IEEE Proceedings International Conference on Computer Design : VLSI in Computer and Processor, 1977.

[8] T.Nanya., Y.Ueno., H.Kagotani., M.Kuwako., A.Takamura. TITAC : Design of a Quasi-Delay-Insensitive Microprocessor. IEEE Design & Test of Computers., Vol.11., No.2, pp.50-63, Summer 1994.

[9] Bormann, D.S.; Molina, P.A.; Cheung, P.Y.K, "Combining asynchronous and synchronous circuits using stretchable clocks" IEEE Colloquium on Design and Test of Asynchronous Systems, 28 Feb 1996,pp.4/1-4/8

[10] P.Ruangsilsap. "Design of 8-bit scalable-delay-insensitive microprocessor using FPGA". Master thesis, Chlalongkorn University, Bangkok, Thailand, 2001

[11] W. J. Bainbridge, Stephen B. Furber: Asynchronous Macrocell Interconnect using MARBLE. ASYNC 1998: 122-132

ง.2 ส่วนหนึ่งของวิทยานิพนธ์ที่ได้ตีพิมพ์ในงานประชุมวิชาการ

The fourth ECTI Annual International Conference (ECTI-CON 2007)

Asynchronous System Bus Enhancement by Interrupt and DMA Technique

Sufian Sudeng and Arthit Thongtak
Digital System Engineering Laboratory (DSEL), Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
254 Phyathai Road, Pathumwan, Bangkok, Thailand 10330
Tel: +66-2-218-6956, Fax: +66-2-218-6955
E-mail: Sufian.S@Student.chula.ac.th, Arthit@cp.eng.chula.ac.th

***Abstract*- This paper proposes a procedure to enhance an asynchronous system bus. The designed bus is incorporated with synchronizing operation between synchronous components and asynchronous ones. Interrupt and DMA technique are chosen for bus capability enhancement.. The prominent point of this work is to take an interrupt and DMA technique assembled with asynchronous system bus. The designed system bus is described in HDL language. The experiments can show the concurrent processing between asynchronous processor execution and DMA transfer operation.**

I. INTRODUCTION

Asynchronous circuits are well known for last decade because of the clock skew avoided. The clock skew occurs when the global clock used to synchronize their operation which can not be distributed to the entire system at the same frequency [1]. It is a major problem cause the system may malfunction. It is very hard to avoid this problem when high speed circuits' transfer needed. As a result, it limits the speed of synchronous circuits. Asynchronous circuits do not use global clock, circuits respond to signal transitions at any time, and outputs can be sent instantly after the operation completion. More outstanding shot of the asynchronous circuit over synchronous circuits is high speed operation because it is not limited by the slowest part of circuits. Low power consumption due to signal transitions are made only when necessary [2].

Causes of complex circuits' design of asynchronous circuit produce some limitation to the design to be large system. Which operate with asynchrony operation, some asynchronous circuits design like processor, there emerge limitation to enhance the design to be large system. It also serious when propagation needed, like connecting an asynchronous processor with multiple peripherals, because memory and I/O module on a present time operate with clock [6], It's a choice if be able to compromise some element that operate with clock to work together with asynchronous circuits, bus enhancement and communicating between synchronous and asynchronous circuits are main purpose of this paper.

DMA design is splash to five working state, compose of registers to store intermediate data, adder/subtracted to count up and down of write and read addresses, the DMA design is capable to work with our asynchronous system bus as well by add stretchable interfacing [6]between bus and DMA module.

Asynchronous processor architecture, control unit, and instruction has modified for interrupt and DMA capability.

Part of paper start with introduction to background of asynchronous system bus design and explain the design of interrupt and DMA, and follow with full system simulation and provide some future work on the final section.

II. ASYNCHRONOUS SYSTEM BUS OVERVIEW

The bus designs are Multiplex. There are data and address transition on the same lines. 8-bit with dual-rail encoded, transition by 4-phase signaling transition protocol[4]. The processes are start with design each component and group them together.

A. Bus components [8]

a. Bus Interface

Bus interface is design to interface with asynchronous processor. Due to the multiplex bus design, bus interface also handshake with bus controller to control the sequence of data, address or spacer transition.

b. Bus Driver

Bus Driver act like an on/off switch to enable/disable transition signal on the bus. Design with tri-state buffer and C-element (the element that produce output 1 when all of input are 1, produce out 0 when all of input are 0, Otherwise its hold previous state)

c. Bus lines

On account of high impedance state of Tri-state buffer, there were making some problem for asynchronous circuits. Need to improve to be zero value when it's high impedance by added weak inverter to each line on the bus.

d. Bus Receiver

This component use to receive data from bus signal and catch all signals before transfer to the destination, Design with C-element

e. Bus Controller

Design with STG (Signal Transition Graph), use to control the synchronization on the bus, due to multiplexing, its responsible to control the sequence of data, address, and spacer transition.

III. INTERRUPT DESIGN

The processor instruction that use in this system has 16-bit length instruction set, by load instruction from ROM to IR register. When 16-bit instruction load in to IR register, IR register response to separate instruction in to two types, *opcode* and *operand*, opcode is load to control unit section, the operand is load to general purpose register or accumulator follow on instruction type [5, 7]. The modification, benefit to generate ***virtual instruction*** that occur during interrupt even, When interrupt instruction has occurred, a virtual instruction has generate by external signal and load to IR register.

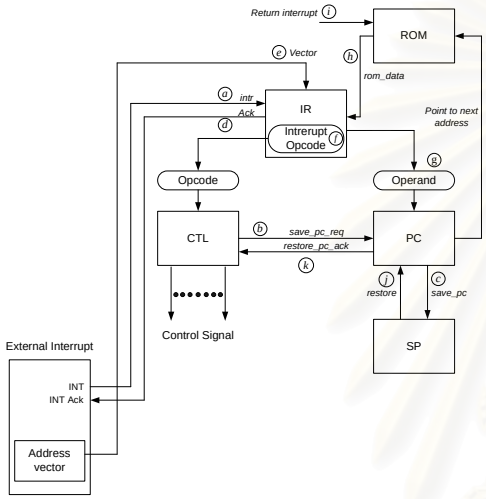


Figure 1. Interrupt Instruction in Asynchronous Processor design

From figure 1 illustrate transaction sequence of an interrupt operation.

a. Interrupt signal asserted
b. Control unit tell pc register to save current pointer
c. Save PC to SP register
d. Interrupt acknowledge asserted
e. Address vector load to IR register
f. Interrupt opcode is generated
g. Load new PC value
h. PC point to interrupt service routine, ROM data load to IR
i. Instruction meets RETI command (return interrupt).
j. RETI command produces SP register to restore value to PC.
k. PC restore acknowledge to control unit
l. Process normal instruction

The processor work on following sequences, First step load instruction that point by PC to IR register, and process base on type of instruction. When EI flags is disable, interrupt is not respond until finish instruction execution of previous instruction EI is enable, if interrupt signal occur by assert *intr* signal by external device, the IR register release ROM connection and produce it self instruction call *virtual instruction* , this instruction is valid for interrupt instruction and separate instruction to two piece, opcode and operand. Opcode is responsible to save / restore signal to pc register , operand is first address that keep an interrupt service routine, when PC register receive save signal from control unit it save current pointer to SP register and jump to next address that produce by virtual opcode.

IV. DMA DESIGN

The Asynchronous processor is a master device on the asynchronous system bus, that it is able to initiate read and write transfers to all devices on the bus. The DMA controller is slave device, which is only able to response to read or write requests. The DMA controller is required to initiate read and write transfers to memory and also other I/O on the bus, DMA controller also required to be a master device

As there will be more than on asynchronous bus master in the system, the asynchronous system bus specification required the presence of a bus arbiter. The bus arbiter selects which master is to have right to the bus at any one instant in time. An arbiter also exits for the asynchronous processor. Initially the DMA controller is programmed by asynchronous processor. This information is stored in internal DMA registers. Include following information: base address for transfer source, base address in memory to where the data is transferred and size of data transfer. The DMA controller also be slave on the asynchronous system bus in order the asynchronous processor access these registers

A. DMA Function

The DMA controller waits for *dma_req* line to assert and take over the asynchronous system bus from asynchronous processor. It checks its internal registers to obtained details of the transfer. The DMA controller read data from the source, store it in internal buffer, and then write it out to the memory, until the transfer is complete. The DMA controller then release usage of asynchronous system bus and activate the IRQ line in order to indicate the asynchronous processor that transfer was complete. If any time the asynchronous processor requires usage of asynchronous system bus it is able to take priority over the DMA controller. The DMA controller check it has control of the bus at each stage, if it does not the controller wait for the bus become available, the asynchronous processor has finished using the bus

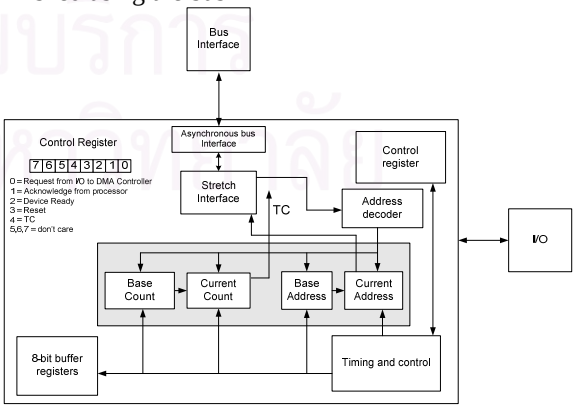


Figure 2. DMA controller architecture

B. DMA Architecture

The internal architecture is very closely based on the classic DMA controller that shown of figure 2, the architecture proposed of the DMA controller split into above functional units. The most complex of these units is timing and control unit, which consist of a large state machine which described in the diagram on the figure 3. The address decoder derives clock enable signal and present 8-bit registers which store details of the transfers.

C. DMA state machine

The most complex part of DMA controller is state machine as shown figure 3

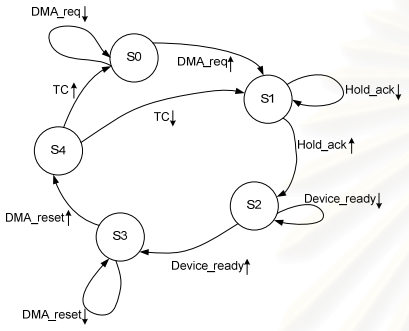


Figure 3. DMA state machine

S0 – Initial State
S1 –DMA Programmed & Ack_to_I/O
S2 – Check status of I/O device
S3 – Reset R/W signals
S4 – Terminal Check

D. Transaction flow

The DMA transaction following on this sequence

a. I/O Module assert *DMA_req* signal
b. DMA assert *Hold* signal to processor
c. Processor assert *Hold_ack* signal to DMA (processor program base address to DMA base register and count register [ST command]).
d. DMA assert *DMA_ack* signal to I/O Module
e. I/O Module assert Device_ready to DMA (transfer type is set by I/O Module (Read or Write)
f. DMA transfer (read or write transaction occur)
g. DMA assert *TC* signal indicate zero value of count register
h. DMA interrupts to processor for complete transaction

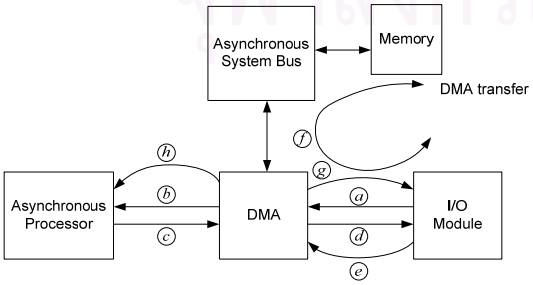


Figure 4. DMA transaction flow

V. Synchronous And Asynchronous Interfacing

This part is design to communicate between synchronous and asynchronous section, design with stretchable clock [6].

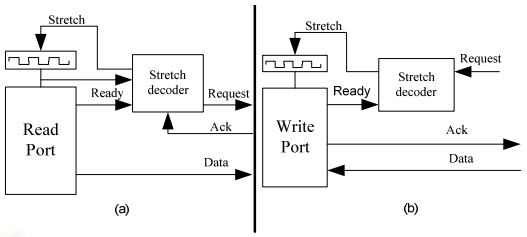


Figure 5. Stretchable clock for read (a) and write (b) interfacing

The clock control is generated using the handshaking signals and the internal state of the synchronous region, a block has either a read port or a write port. Read port provides data for an external element while a write port receive data into the module. In a 4-phase protocol, each module must receive two asynchronous transitions to complete a cycle. Each synchronous module must go through to two clock cycles to allow for two possible stretch signals.

VI. Simulation Result

The system is design with VHDL description language [3] separately on each component and simulates each component for correct signal behavior and functional. Then combined those together to form a full system as shown on figure 6 .after that experimental simulation are shown.

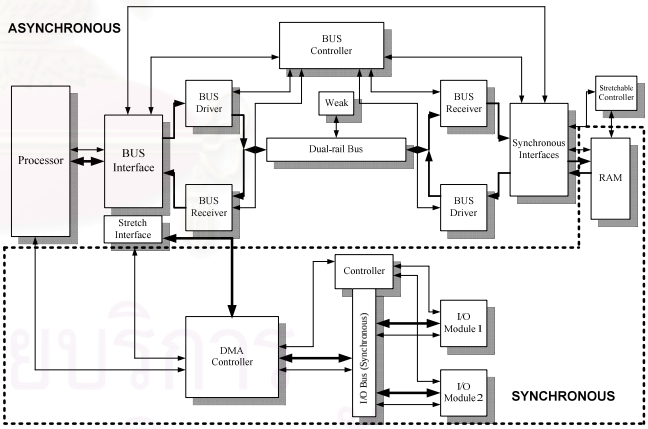


Figure 6. Simulation Environment

Simulation Program

```
ORG     0000H
LD      A,#39H  ; A = 39H
ST      A,R0    ; R0 = 39H
LD      A,#24H  ; A = 24H
ST      A,R1    ; R1 = 24H
AND     A,#14H  ; A= 04H
LD      A,#C8H  ; A = C8H
ADD     A,R0    ; A = 01H, CARRY = 1
ST      A,R1    ; R1 = 01H
```

```
OR      A,R0     ; A = 39H
OR      A,#B7H ; A = BFH
LD      A,#13H  ; A = 13H
ST      A,@R1   ; M(01) = 13H
LD      A,#00H  ; A=00H
INTR :
LD      A,#01H   ; A=01H
ST      A,@1FFH ; DMA_Base = 01H
LD      A,#0AH  ; A=0AH
ST      A,@2FFH ; DMA_Count = 0AH
RETI
END
```

The bold lists on simulation program are shown on waveform diagram.

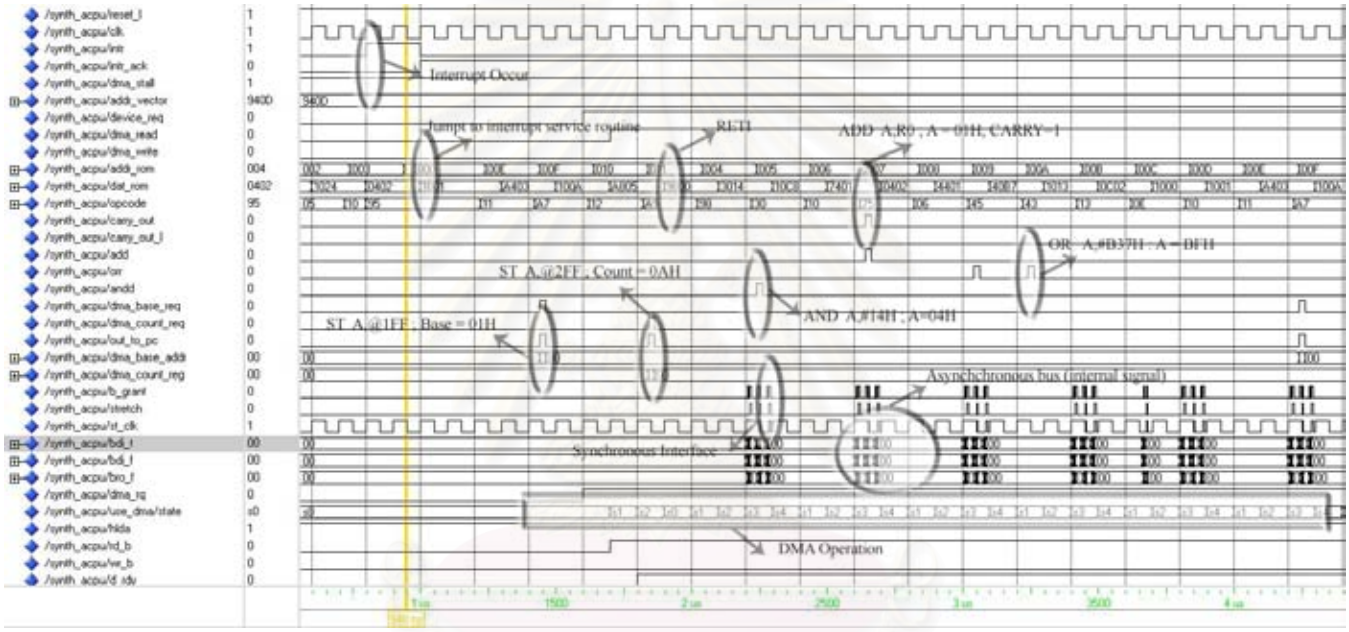


Figure 7. Experimental Simulation Result

VII. Conclusion And Future Work

This paper proposes technique to enhance an asynchronous system bus [8]. Introduce synchronization procedure between synchronous and asynchronous circuits. Interrupt and DMA technique are choose for bus enhanced, DMA is design with state diagram and connect with two input/output buffers, I/O arbitration controlled, The system work as follows, interrupt signal from I/O module assert to tell processor release the bus, when processor release it, operation start by DMA controller concurrently with asynchronous processor processing. For future work, improve the system to interface with present I/O devices, such as USB, LCD screen or serial port interfacing, together with develop the DMA controller to be asynchrony operation.

Acknowledgment

The research is supported by TJTTP-OECF (Thailand - Japan Technology Transfer Project – Japanese Overseas Economic Cooperation Fund)

References

[1] Davis. A, Nowick.S.M “An Introduction to Asynchronous Circuit Design” , 1997

[2] S.Hauck. “Asynchronous Design Methhodologies : An Overview”. Proceedings of the IEEE. vol.83. No.1. pp.69-93, January 1995.

[3] J.Bhasker. “*A VHDL Primer*”. Prentice-Hall, Inc., 1992. ISBN 0-13-952987-X.

[4] Molina, P.A.; Cheung, P.Y.K ,“A Quasi Delay-Insensitive Bus Proposal for Asynchronous Systems” Proceedings Third International Symposium on Advanced Research in Asynchronous Circuits and Systms, 1997 Pages:129-139

[5] T.Nanya., Y.Ueno., H.Kagotani., M.Kuwako., A.Takamura. TITAC : Design of a Quasi-Delay-Insensitive Microprocessor. IEEE Design & Test of Computers., Vol.11., No.2, pp.50-63, Summer 1994.

[6] Bormann, D.S.; Molina, P.A.; Cheung, P.Y.K, “Combining asynchronous and synchronous circuits using stretchable clocks” IEEE Colloquium on Design and Test of Asynchronous Systems, 28 Feb 1996,pp.4/1-4/8

[7] P.Ruangsilsap. “Design of 8-bit scalable-delay-insensitive microprocessor using FPGA”. Master thesis, Chlalongkorn University, Bangkok, Thailand, 2001

[8] S.Sudeng, A.Thongtak:” A Design of System Bus for Asynchronous Circuits” International Technical Conference in Circuits/Systems Computer and Communication (ITC-CSCC2006), Pang Suan Kaew hotel, Chiangmai, Thailand on July 10-13, 2006

ประวัติผู้เขียนวิทยานิพนธ์

สุฟียัน สูเด็ง เกิดที่จังหวัดปัตตานี เมื่อปี พศ 2523 สำเร็จการศึกษาระดับปริญญาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี เมื่อปี พศ 2545 และเข้าศึกษาต่อสาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยเมื่อปี พศ 2546