บทที่ 3

ระบบไมโครโปรเซสเซอร์แบบกระจายที่ใช้หน่วยความจำร่วม

ตามปกติไมโครโปรเซสเซอร์แต่ละตัวในระบบไมโครโปรเซสเซอร์แบบกระจาย ต่างก็ทำงานเป็นอิสระไม่เกี่ยวข้องกันไปจนกระทั่งจะมีไมโครโปรเซสเซอร์ตัวใดตัวหนึ่งต้องการข้อมูลจากไมโครโปรเซสเซอร์ตัวอื่น จึงจะมีการสื่อสารระหว่างไมโครโปรเซสเซอร์เกิดขึ้น การสื่อสารข้อมูลในกลุ่มไมโครโปรเซสเซอร์โดยผ่านหน่วยความจำร่วม RAM เป็นวิธีการที่เหมาะสมที่สุดวิธีหนึ่ง (2, 5) และสามารถทำได้สามแบบดังนี้

1. การใช้ TWO PORT RAM เป็นหน่วยความจำร่วม (13)

TWO PORT RAM เป็นหน่วยความจำที่มี 2 พอร์ท (port) และสามารถเขียนข้อมูลลงไปในพอร์ทหนึ่งในขณะเดียวกับอ่านข้อมูลจากอีกพอร์ทหนึ่งได้ ทำให้ไม่เกิดการสับสนในกรณีที่ไมโครโปรเซสเซอร์สองตัวต้องการเขียนและอ่านข้อมูลจากหน่วยความจำร่วมพร้อมกัน แต่เนื่องจาก TWO PORT RAM มีขนาดเล็ก (16 X 4 บิท) จึงเหมาะสำหรับระบบที่มีการสื่อสารข้อมูลจำนวนน้อยและมีราคาแพงมากจนไม่คุ้มค่าที่จะนำมาใช้ในระบบไมโครโปรเซสเซอร์แบบกระจายขนาดเล็ก

2. การใช้ Direct Memory Access (DMA)

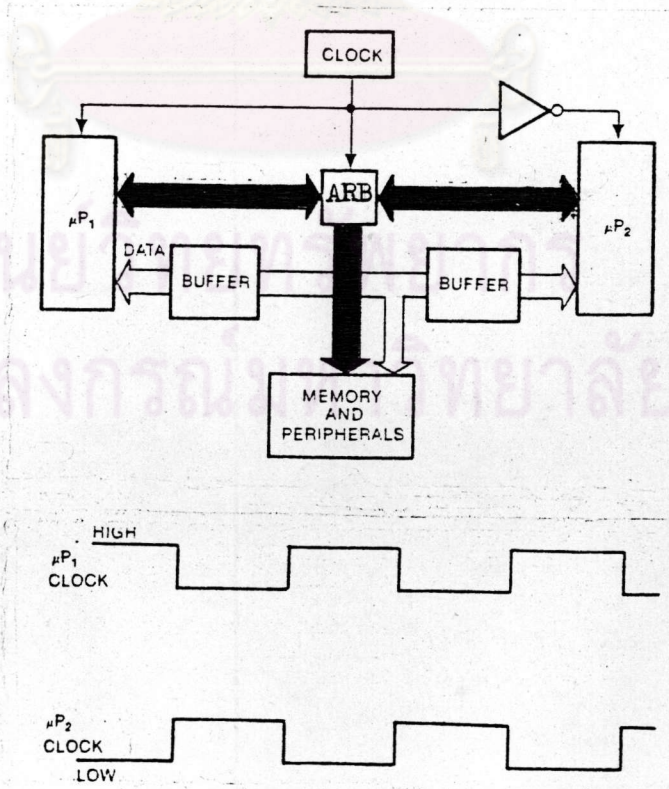
วิธีนี้เป็นการเคลื่อนย้ายข้อมูลเข้าหรือออกจากหน่วยความจำโดยไม่รบกวนการทำงานของ ซีพียู. โดยมีวงจรควบคุม ดีเอ็มเอ.(DMA controller) ควบคุมการเคลื่อนย้ายข้อมูลนี้ ทำให้สามารถรับส่งข้อมูลด้วยความเร็วสูงมาก (มากกว่า 1 megabyte/sec) จึงเหมาะสำหรับระบบที่มีการสื่อสารข้อมูลระหว่างไมโครโปรเซสเซอร์เป็นปริมาณมาก และเนื่องจากเป็นวิธีการที่ยุ่งยากและวงจรควบคุม ดีเอ็มเอ. มีราคาแพง จึงไม่เหมาะที่จะใช้กับระบบไมโครโปรเซสเซอร์แบบกระจายขนาดเล็ก

3. การใช้บัสร่วม (Common Bus) ควบคุมโดยวงจร ARBITER

วิธีนี้ใช้ได้กับหน่วยความจำ RAM ทั่วไป โดยอินเตอร์เฟสบัสของหน่วยความจำร่วมเข้ากับบัสของไมโครโปรเซสเซอร์ทุกตัวในระบบ และมีวงจรภายนอกทำหน้าที่จัดลำดับให้ไมโครโปรเซสเซอร์ในระบบใช้หน่วยความจำร่วมครั้งละหนึ่งตัว พร้อมกับห้ามไม่ให้ไมโครโปรเซสเซอร์ตัวอื่นใช้หน่วยความจำร่วมในช่วงเวลานั้น เพื่อป้องกันการสับสนของข้อมูลในบัส เนื่องจากมีเพียงเส้นทางเดียวเท่านั้นที่ใช้ในการรับส่งข้อมูลระหว่างไมโครโปรเซสเซอร์กับหน่วยความจำร่วมซึ่งเรียกวงจรนี้ว่า วงจร ARBITER วิธีนี้เป็นวิธีที่เหมาะสมที่สุดที่จะใช้กับระบบไมโครโปรเซสเซอร์แบบกระจายขนาดเล็ก เพราะเป็นวิธีการที่ยุ่งยากน้อยกว่าและดัดแปลงให้เหมาะสมกับระบบที่จะนำไปใช้ได้ง่ายกว่าสองวิธีแรก

ได้มีผู้เสนอวิธีการต่าง ๆ เพื่อใช้สร้างวงจร ARBITER สำหรับหน่วยความจำร่วม ในระบบไมโครโปรเซสเซอร์แบบกระจาย แต่ส่วนใหญ่ยังมีปัญหาหรือข้อจำกัด เช่น

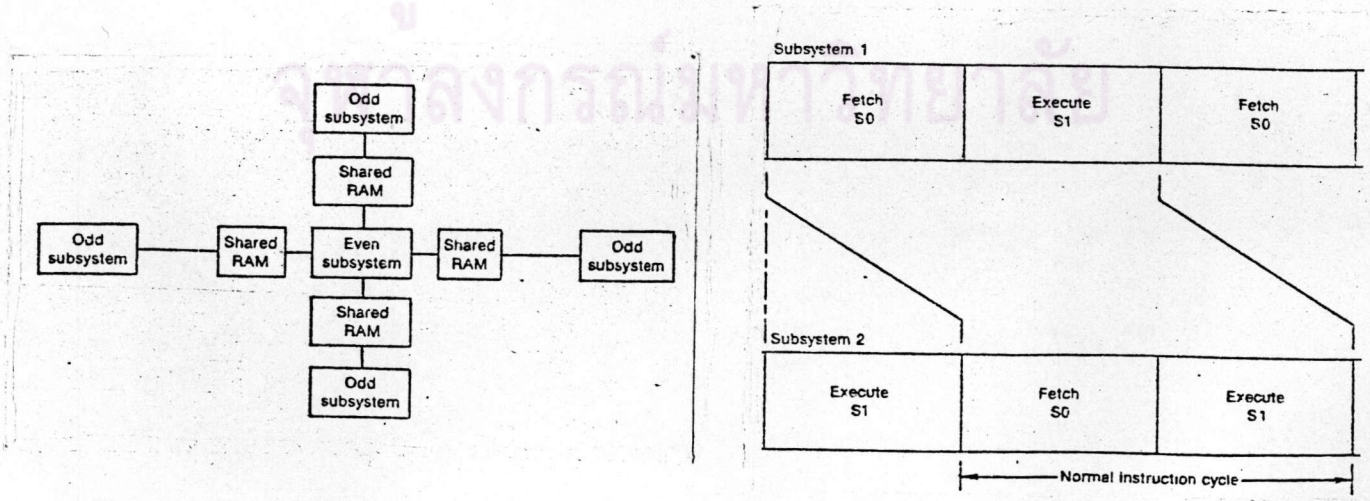
- วิธีของ LOEWER (14) หลักการของวิธีนี้คือ ให้ไมโครโปรเซสเซอร์ใช้สัญญาณ clock ที่เฟส (phase) ตรงข้ามกัน ดังรูป 3.1



รูป 3.1 โครงสร้างและหลักการทำงานของ ARBITER ตามวิธีของ LOEWER (14)

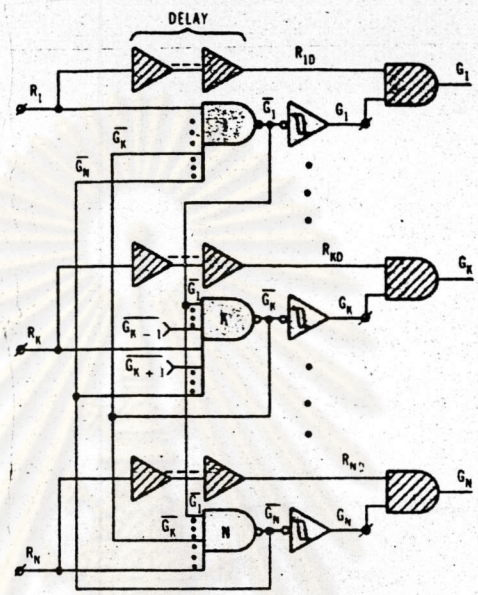
เนื่องจากไมโครโปรเซสเซอร์แต่ละตัวใช้สัญญาณ clock ที่เฟสตรงข้ามกัน ดังนั้น การขอใช้หน่วยความจำร่วมจึงไม่มีโอกาสเกิดขึ้นได้พร้อมกัน ไมโครโปรเซสเซอร์ตัวแรกที่ขอใช้หน่วยความจำร่วมจะเป็นผู้ได้รับสิทธิ์ และหากว่าในระหว่างที่ไมโครโปรเซสเซอร์ตัวนั้นกำลังใช้หน่วยความจำร่วมแล้วไมโครโปรเซสเซอร์อีกตัวหนึ่งขอใช้หน่วยความจำร่วม ไมโครโปรเซสเซอร์ตัวหลังจะได้รับคำสั่งให้หยุดรอจนกว่าตัวแรกจะเสร็จสิ้นการใช้หน่วยความจำร่วม ซึ่งกลไกนี้ควบคุมโดยวงจร ARB ในรูป 3.1 ข้อเสียของวิธีนี้คือใช้ได้กับไมโครโปรเซสเซอร์ไม่เกิน 2 ตัว จะขยายระบบให้ใหญ่ขึ้นอีกไม่ได้

- วิธีของ POLCZYNSKI (15) วิธีนี้ใช้หน่วยความจำร่วมระหว่างไมโครโปรเซสเซอร์เป็นคู่ ๆ ดังรูป 3.2 โดยให้ไมโครโปรเซสเซอร์แต่ละตัวทำงานในจังหวะสลับกัน กล่าวคือ ไมโครโปรเซสเซอร์ใน Odd subsystem ทำงานในจังหวะเฟทช์ ในขณะที่ไมโครโปรเซสเซอร์ใน Even subsystem ทำงานในจังหวะเอ็กซ์คิว จึงไม่มีโอกาสที่ไมโครโปรเซสเซอร์ทั้งสองตัวจะใช้หน่วยความจำร่วมพร้อมกันได้ เนื่องจากการใช้หน่วยความจำร่วมจะเกิดขึ้นในจังหวะเอ็กซ์คิวเท่านั้น ไมโครโปรเซสเซอร์ทุกตัวในระบบจึงต้องใช้สัญญาณ clock ชุดเดียวกันและต้องมีจังหวะเฟทช์ยาวเท่ากับจังหวะเอ็กซ์คิว ข้อจำกัดของวิธีนี้คือ ใช้ได้เฉพาะกับไมโครโปรเซสเซอร์ที่จังหวะเฟทช์และเอ็กซ์คิวยาวเท่ากันเช่น COSMAC 1802 ไม่อาจดัดแปลงใช้กับไมโครโปรเซสเซอร์ชนิดอื่นได้ และคำสั่งบางคำสั่งที่ทำให้จังหวะการทำงานทั้งสองยาวไม่เท่ากันเช่น คำสั่ง jump ข้าม page ก็ไม่สามารถใช้ได้ ทำให้ลดความสามารถทางซอฟท์แวร์ของระบบลงไปมาก



รูป 3.2 โครงสร้างและหลักการทำงานของ ARBITER ตามวิธีของ POLCZYNSKI (15)

- วิธีของ PETRIU (16) ARBITER ตามวิธีนี้ควรสร้างขึ้นในลักษณะ LSI chip มากกว่าที่จะสร้างขึ้นจาก TTL. logic gate เนื่องจากต้องอาศัยคุณสมบัติภายในของ NAND gate แต่ละตัวในการทำงาน โดยมีโครงสร้างดังรูป 3.3

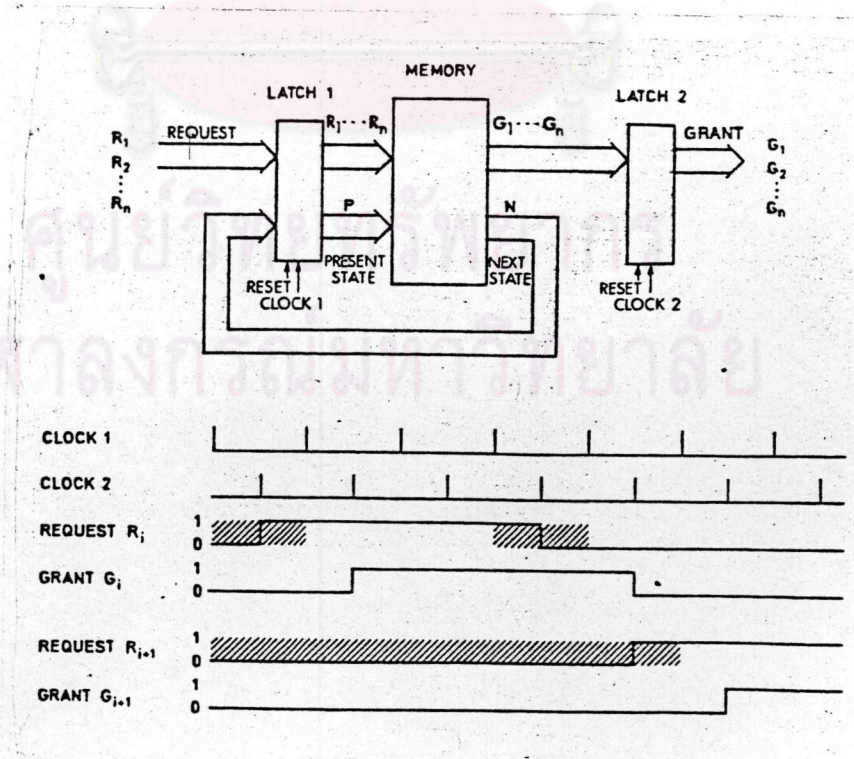


รูป 3.3 โครงสร้างของ ARBITER ตามวิธีของ PETRIU (16)

ในสภาวะเริ่มต้นที่ไมโครโปรเซสเซอร์แต่ละตัวยังไม่ต้องการใช้หน่วยความจำร่วม สัญญาณขอใช้หน่วยความจำร่วม (R1-Rn) จากไมโครโปรเซสเซอร์แต่ละตัวจะเป็น 0 ทำให้เอาท์พุทของ NAND gate ทุกตัวเป็น 1 ซึ่งสัญญาณนี้จะส่งผ่านวงจร Inverter ไปป้อนเป็นอินพุทของ AND gate พร้อมกับสัญญาณขอใช้หน่วยความจำร่วม (R1-Rn) ที่ส่งผ่านวงจร delay มา ทั้งนี้เพื่อกำจัดสัญญาณ Glitch ที่เกิดขึ้นในการทำงาน เอาท์พุทของ AND gate จะเป็นสัญญาณอนุญาตให้ใช้หน่วยความจำร่วมส่งไปยังไมโครโปรเซสเซอร์แต่ละตัว เมื่อใดก็ตามที่ไมโครโปรเซสเซอร์แต่ละตัวส่งสัญญาณขอใช้หน่วยความจำร่วมมาพร้อมกัน (R1-Rn กลายเป็น 1) NAND gate ทุกตัวที่ได้รับสัญญาณนี้จะเริ่มเปลี่ยนสภาวะเอาท์พุทจาก 1 เป็น 0 แต่จะมีเพียง NAND gate ซึ่งเป็นตัวที่ทำงานเร็วที่สุดเพียงตัวเดียวเท่านั้นที่สามารถเปลี่ยนเอาท์พุทจาก 1 เป็น 0 ได้ พร้อมกับป้อนสัญญาณนี้กลับไปเป็นอินพุทของ NAND gate ตัวอื่นในระบบ ทำให้มีสัญญาณอนุญาตให้ใช้หน่วยความจำร่วม (G1-Gn) ที่เป็น 1 เพียงสัญญาณเดียว หมายความว่า

มีไมโครโปรเซสเซอร์เพียงตัวเดียวที่มีโอกาสใช้หน่วยความจำร่วมในช่วงเวลานั้น และไมโครโปรเซสเซอร์ตัวนั้นมี priority สูงกว่าตัวอื่น ๆ ที่ขอใช้หน่วยความจำร่วมพร้อมกัน จะเห็นได้ว่าหากใช้ TTL. logic gate สร้าง ARBITER ตามวิธีนี้ จะต้องเสียเวลาในการทดสอบคุณสมบัติของ gate แต่ละตัวเป็นเวลานาน ขั้นตอนการสร้างจึงยุ่งยากและสิ้นเปลืองเวลามาก โดยเฉพาะอย่างยิ่งเมื่อจำนวนไมโครโปรเซสเซอร์ที่ใช้หน่วยความจำร่วมมีมากขึ้น รวมทั้งดัดแปลงไปใช้กับ priority ลักษณะพิเศษต่าง ๆ ไม่ได้

- วิธีของ HOJBERG (17) วิธีนี้แบ่งแยกการทำงานของ ARBITER ออกเป็นสองส่วนคือ ส่วนที่ทำหน้าที่รับสัญญาณขอใช้หน่วยความจำร่วมและส่งสัญญาณอนุญาตให้ใช้หน่วยความจำร่วมไปยังไมโครโปรเซสเซอร์ เรียกว่า REQUEST/GRANT FLIP-FLOP อีกส่วนหนึ่งทำหน้าที่กำหนดลำดับการใช้หน่วยความจำร่วมสำหรับไมโครโปรเซสเซอร์แต่ละตัว เรียกว่า Priority generator ข้อดีของการแบ่งแยกหน้าที่ของการทำงานของ ARBITER ในลักษณะนี้คือสามารถดัดแปลงใช้กับ priority ลักษณะต่าง ๆ ได้ง่ายกว่าสามวิธีแรก ARBITER ตามวิธีของ HOJBERG มีโครงสร้างและตารางเวลาการทำงานดังรูป 3.4



รูป 3.4 โครงสร้างและการทำงานของ ARBITER ตามวิธีของ HOJBERG (17)

LATCH 1 ทำหน้าที่ REQUEST FLIP-FLOP ทำงานตามจังหวะสัญญาณ clock 1 และ LATCH 2 ทำหน้าที่ GRANT FLIP-FLOP ทำงานตามจังหวะสัญญาณ clock 2 ส่วน memory ทำหน้าที่ Priority generator ซึ่งในวิธีนี้ใช้ priority แบบ Round Robin กล่าวคือ ทุกตัวมี priority เท่ากัน ซึ่งมีข้อดีคือสามารถป้องกันไม่ให้เกิดการยึดครองหน่วยความจำร่วมไว้ใช้เพียงตัวเดียวโดยไมโครโปรเซสเซอร์ที่มี priority สูงกว่าตัวอื่น ข้อมูลในแต่ละแอดเดรสของ memory จะแบ่งเป็นสองส่วนคือ ส่วนแรกเป็นข้อมูลสัญญาณอนุญาตให้ใช้หน่วยความจำร่วม (G1-Gn) อีกส่วนหนึ่งจะเป็นข้อมูลที่ป้อนกลับมาเป็นแอดเดรสของ memory เพื่อกำหนดว่าไมโครโปรเซสเซอร์ตัวใดจะมีสิทธิ์ใช้หน่วยความจำร่วมในคราวต่อไป ตัวอย่างของข้อมูลในแอดเดรสต่าง ๆ ของ memory ที่ทำหน้าที่ priority generator แสดงไว้ในรูป 3.5

	Address		Data	
	$R_1R_2R_3R_4$	$P_1P_2P_3P_4$	$G_1G_2G_3G_4$	$N_1N_2N_3N_4$
1	1011	1000	1000	1000
2	0011	1000	0010	0010
3	0011	0010	0010	0010
4	0001	0010	0001	0001
5	0001	0001	0001	0001
6	0000	0001	0000	1000

รูป 3.5 ตัวอย่างของข้อมูลในแอดเดรสต่าง ๆ ของ priority generator memory

เมื่อไมโครโปรเซสเซอร์ส่งสัญญาณขอใช้หน่วยความจำร่วม (R1-Rn) มายัง LATCH 1 สัญญาณนี้พร้อมกับข้อมูลที่ป้อนกลับจากเอาท์พุทของ memory จะผ่าน LATCH 1 ไปเป็นแอดเดรสของ memory ตามจังหวะสัญญาณ clock 1 ถ้าเป็นกรณีที่ไมโครโปรเซสเซอร์ 1, 3 และ 4 ส่งสัญญาณขอใช้หน่วยความจำร่วมมาพร้อมกันในจังหวะที่ไมโครโปรเซสเซอร์ 1 มีสิทธิ์ใช้หน่วยความจำร่วม แอดเดรสที่ LATCH 1 ส่งให้ memory จะเป็น 1011 1000 ซึ่งข้อมูลในแอดเดรสนี้จะเป็น 1000 1000 (ลำดับที่ 1 ในรูป 3.5) หมายความว่า ไมโครโปรเซสเซอร์ 1 จะได้รับสัญญาณ G1 อนุญาตให้ใช้หน่วยความจำร่วมได้ โดยสัญญาณ G1 จะส่งผ่าน LATCH 2 ในสัญญาณ clock 2 จังหวะถัดมา (รูป 3.3) เอาท์พุทของ memory จะคงสภาพนี้ไปจนกระทั่งไมโครโปร-

เซสเซอร์ 1 เสร็จสิ้นการใช้หน่วยความจำร่วมและยกเลิกสัญญาณขอใช้หน่วยความจำร่วม R1 จากนั้นในจังหวะสัญญาณ clock 1 ถัดมา แอดเดรสของ memory จะเป็น 0011 1000 ซึ่งเอาท์พุทของ memory จะเป็น 0010 0010 หมายความว่าไมโครโปรเซสเซอร์ 3 จะได้รับสัญญาณ G3 อนุญาตให้ใช้หน่วยความจำร่วม ขอให้สังเกตว่าในจังหวะสัญญาณ clock 1 ถัดมา (ลำดับที่ 3 ในรูป 3.5) ข้อมูลที่ป้อนกลับจากเอาท์พุทของ memory จะเปลี่ยนเป็น 0010 เพื่อกำหนดว่าเป็นลำดับที่ไมโครโปรเซสเซอร์ 3 มีสิทธิ์ใช้หน่วยความจำร่วมและจะคงอยู่ในสภาพนี้จนกระทั่งไมโครโปรเซสเซอร์ 3 เสร็จสิ้นการใช้หน่วยความจำร่วมและยกเลิกสัญญาณขอใช้หน่วยความจำร่วม R3 ต่อจากนั้นไมโครโปรเซสเซอร์ 4 จึงจะมีสิทธิ์ใช้หน่วยความจำร่วมในลักษณะเดียวกับสองตัวแรก หลังจากไมโครโปรเซสเซอร์ 4 เสร็จสิ้นการใช้หน่วยความจำร่วมแล้ว และไม่มีไมโครโปรเซสเซอร์ตัวอื่นต้องการใช้หน่วยความจำร่วมอีก แอดเดรสของ memory จะเปลี่ยนเป็น 0000 0001 แล้วเปลี่ยนเป็น 0000 1000 (ลำดับที่ 5, 6 ในรูป 3.5) ในสัญญาณ clock 1 จังหวะถัดมา เพื่อให้สิทธิ์ไมโครโปรเซสเซอร์ตัวถัดไปมีโอกาสใช้หน่วยความจำร่วมเป็นตัวแรก แล้ววนเวียนลักษณะที่กล่าวมาแล้วไปตลอดการทำงาน อนึ่ง ขอให้สังเกตว่าไม่ว่าสัญญาณขอใช้หน่วยความจำจะเกิดขึ้นเมื่อใดก็ตามในช่วงเวลาที่แลเงาในรูป 3.3 จะมีผลให้เกิดสัญญาณอนุญาตให้ใช้หน่วยความจำร่วมที่เวลาเดียวกัน ทั้งนี้เพราะ REQUEST และ GRANT FLIP-FLOP ทำงานตามสัญญาณ clock คนละชุดและที่ทำหน้าที่ Priority generator สำหรับระบบที่มีไมโครโปรเซสเซอร์ 4 ตัว ต้องมีขนาด 256 แอดเดรส โดยแต่ละแอดเดรสมีข้อมูล 8 บิท ซึ่งจะเห็นว่าข้อเสียของการสร้าง ARBITER ตามวิธีนี้คือ ฮาร์ดแวร์ซับซ้อนและต้องการใช้สัญญาณ clock ความถี่สูงจากภายนอกถึง 2 สัญญาณ ทำให้ขั้นตอนการสร้างยุ่งยากและราคาแพง

วิทยานิพนธ์นี้มีวัตถุประสงค์เพื่อปรับปรุงการใช้ GRANT/REQUEST FLIP-FLOP ร่วมกับ Priority generator ทำหน้าที่ ARBITER ของ HOJBERG (17) ให้มีประสิทธิภาพยิ่งขึ้นโดยไม่ต้องการสัญญาณ clock จากภายนอกระบบในการทำงานและใช้ฮาร์ดแวร์ง่าย ๆ รวมทั้งดัดแปลงให้ใช้กับ priority ลักษณะใดก็ได้

3.1 หลักการของ ARBITER

ไมโครโปรเซสเซอร์ในระบบไมโครโปรเซสเซอร์แบบกระจายที่ใช้หน่วยความจำร่วม จะทำงานในสภาวะต่าง ๆ 3 สภาวะ (state) คือ

1. Free state คือสภาวะที่ไมโครโปรเซสเซอร์ต่างทำงานเป็นอิสระไม่เกี่ยวข้องกับไมโครโปรเซสเซอร์ตัวอื่น ซึ่งเป็นสภาวะปกติของไมโครโปรเซสเซอร์ในระบบ

2. Wait state คือสภาวะที่ไมโครโปรเซสเซอร์รอคอยการใช้หน่วยความจำร่วม ในระหว่างที่ไมโครโปรเซสเซอร์ตัวอื่นกำลังใช้อยู่ เกิดขึ้นเมื่อมีการสื่อสารข้อมูลผ่านหน่วยความจำร่วม

3. Grant state คือสภาวะที่ไมโครโปรเซสเซอร์กำลังใช้หน่วยความจำร่วม เกิดขึ้นเมื่อไมโครโปรเซสเซอร์ต้องการส่งหรือรับข้อมูลจากไมโครโปรเซสเซอร์ตัวอื่น

ในการสื่อสารผ่านหน่วยความจำร่วม ไมโครโปรเซสเซอร์จะต้องส่งสัญญาณขอใช้หน่วยความจำร่วมไปยังวงจร ARBITER ซึ่ง ARBITER จะส่งสัญญาณตอบรับมาเป็นสองกรณีคือสัญญาณอนุญาตให้ใช้หน่วยความจำร่วมได้ หรือสัญญาณให้หยุดรอ และภายในวงจร ARBITER จะต้องมีกลไกทำหน้าที่จัดลำดับการใช้หน่วยความจำร่วม โดยกำหนดว่าในช่วงเวลานี้ไมโครโปรเซสเซอร์ตัวใดจะได้รับสิทธิใช้หน่วยความจำร่วม ดังนั้นจึงสามารถกำหนดสัญญาณต่าง ๆ ที่เกี่ยวข้องกับวงจร ARBITER ได้เป็น 4 สัญญาณคือ

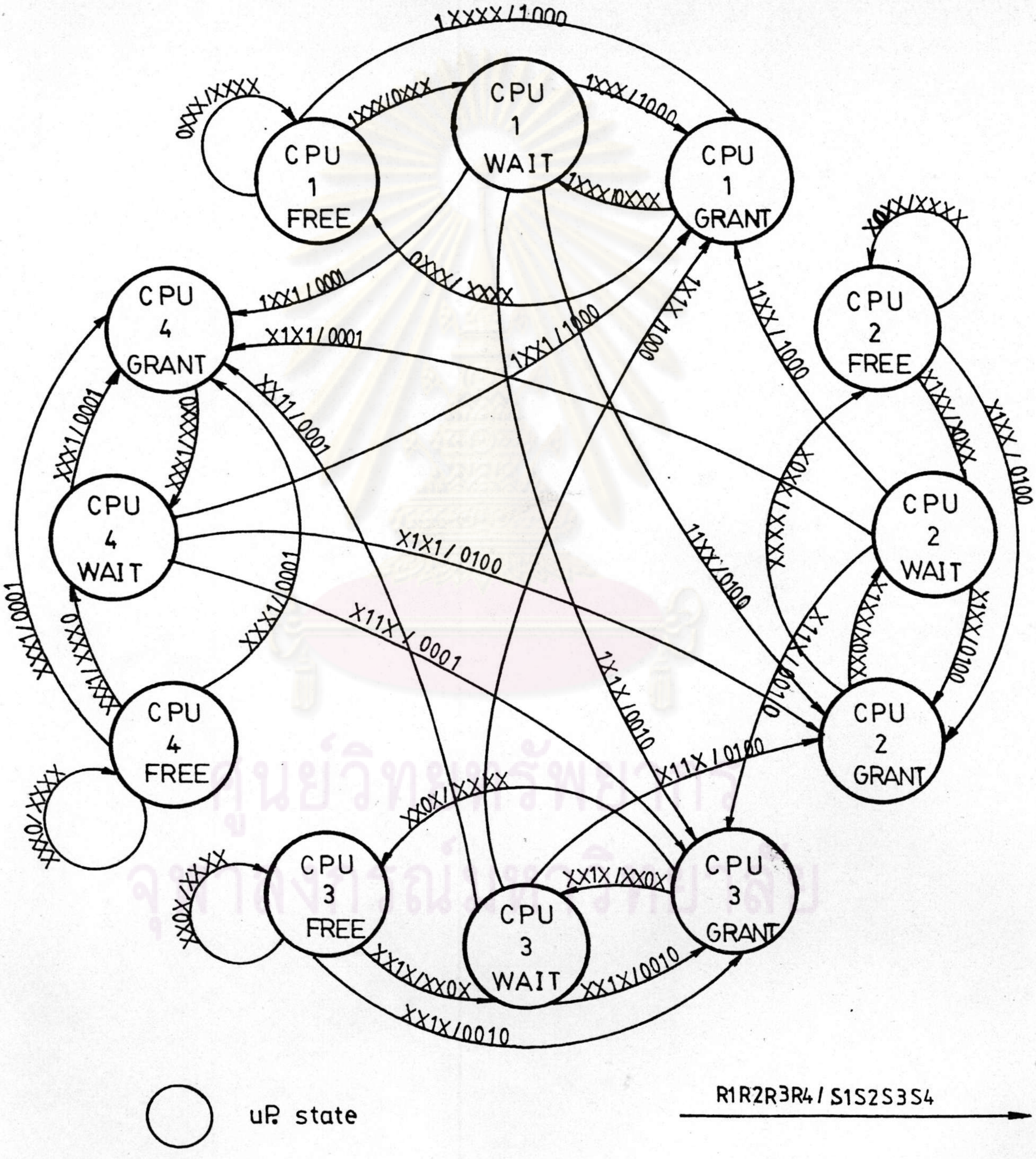
1. สัญญาณ REQUEST (Ri) เป็นสัญญาณขอใช้หน่วยความจำร่วมจากไมโครโปรเซสเซอร์

2. สัญญาณ GRANT (Gi) เป็นสัญญาณที่ ARBITER ส่งไปยังไมโครโปรเซสเซอร์เพื่ออนุญาตให้ใช้หน่วยความจำร่วมได้ หลังจากได้รับสัญญาณ REQUEST จากไมโครโปรเซสเซอร์ตัวนั้น

3. สัญญาณ WAIT (Wi) เป็นสัญญาณที่ ARBITER ส่งไปยังไมโครโปรเซสเซอร์เพื่อสั่งให้หยุดรอ หลังจากได้รับสัญญาณ REQUEST และตัดสินว่ายังไม่ถึงลำดับของไมโครโปรเซสเซอร์ตัวนั้น

4. สัญญาณ SCANNING (Si) เป็นสัญญาณภายในของ ARBITER ทำหน้าที่กำหนดลำดับการใช้หน่วยความจำร่วม กล่าวคือในช่วงเวลาหนึ่ง ๆ จะมีสัญญาณ SCANNING สำหรับไมโครโปรเซสเซอร์เพียงตัวเดียว

ความสัมพันธ์ระหว่างสัญญาณทั้งสี่กับสภาวะการทำงานของไมโครโปรเซสเซอร์ในระบบที่มีไมโครโปรเซสเซอร์ 4 ตัว ใช้หน่วยความจำร่วมกัน แสดงไว้ใน state diagram รูป 3.6



รูป 3.6 State diagram ของระบบไมโครโปรเซสเซอร์แบบกระจายที่ใช้หน่วยความจำร่วม

3.2 โครงสร้างของวงจร ARBITER

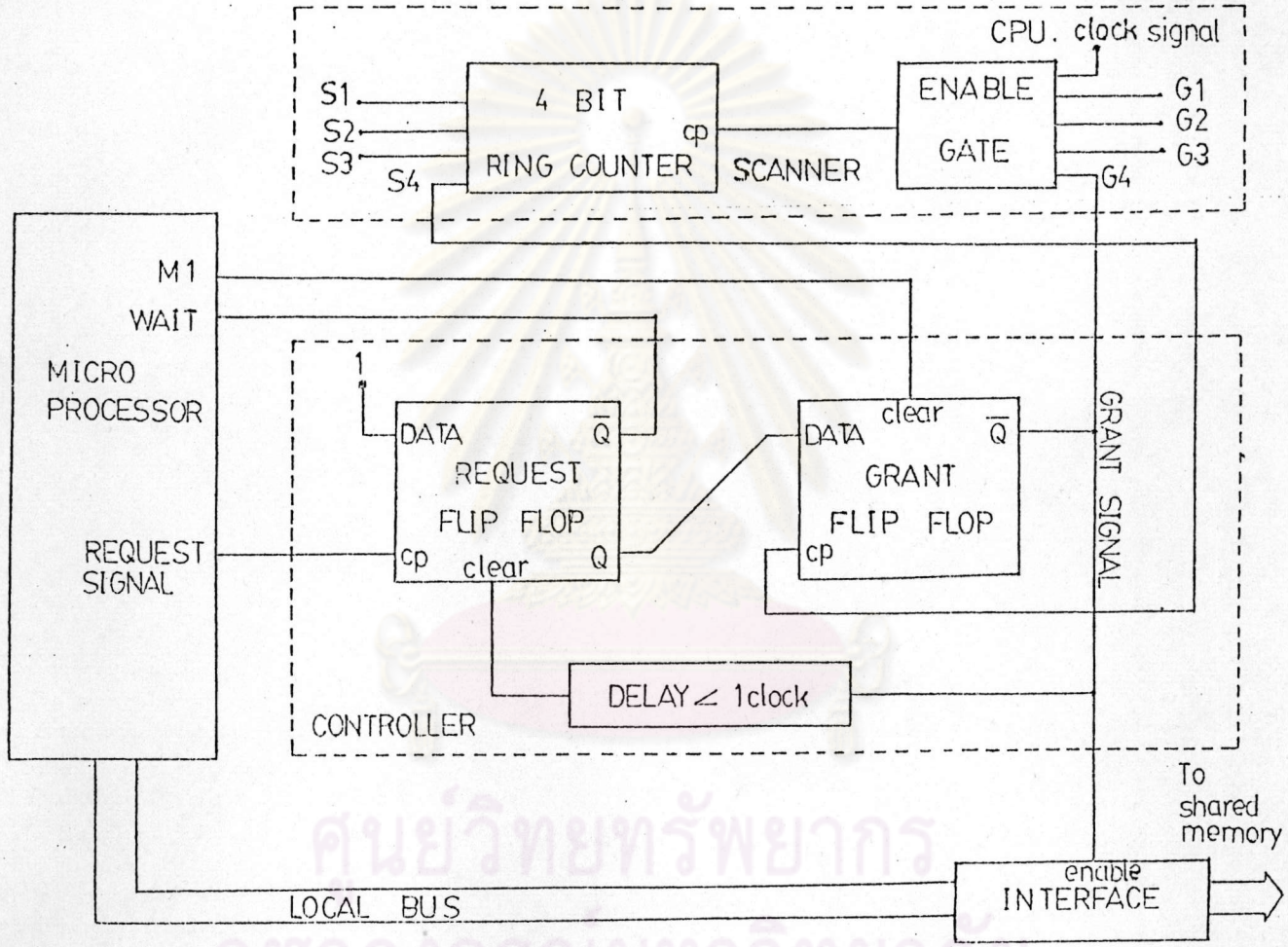
วงจร ARBITER ที่ออกแบบ ใช้ได้กับไมโครโปรเซสเซอร์ Z-80 หรือไมโครโปรเซสเซอร์อื่นที่มีสัญญาณเอาท์พุทบอกสภาวะเฟทช์และสัญญาณอินพุทสั่งให้ไมโครโปรเซสเซอร์อยู่ในสภาวะการรอ (Wait state) จำนวน 4 ตัว โดยอนุญาตให้ไมโครโปรเซสเซอร์แต่ละตัวใช้หน่วยความจำร่วมได้คราวละ 1 คำสั่ง และมีส่วนประกอบสำคัญแบ่งเป็น 2 ส่วนคือ

3.2.1 SCANNER เป็นตัวกำหนดลำดับการใช้หน่วยความจำร่วม โดยสร้างสัญญาณ SCANNING สำหรับไมโครโปรเซสเซอร์เพียงตัวเดียวในช่วงเวลาหนึ่ง และไมโครโปรเซสเซอร์ตัวนั้นจะมีสิทธิใช้หน่วยความจำร่วมเพียงตัวเดียว

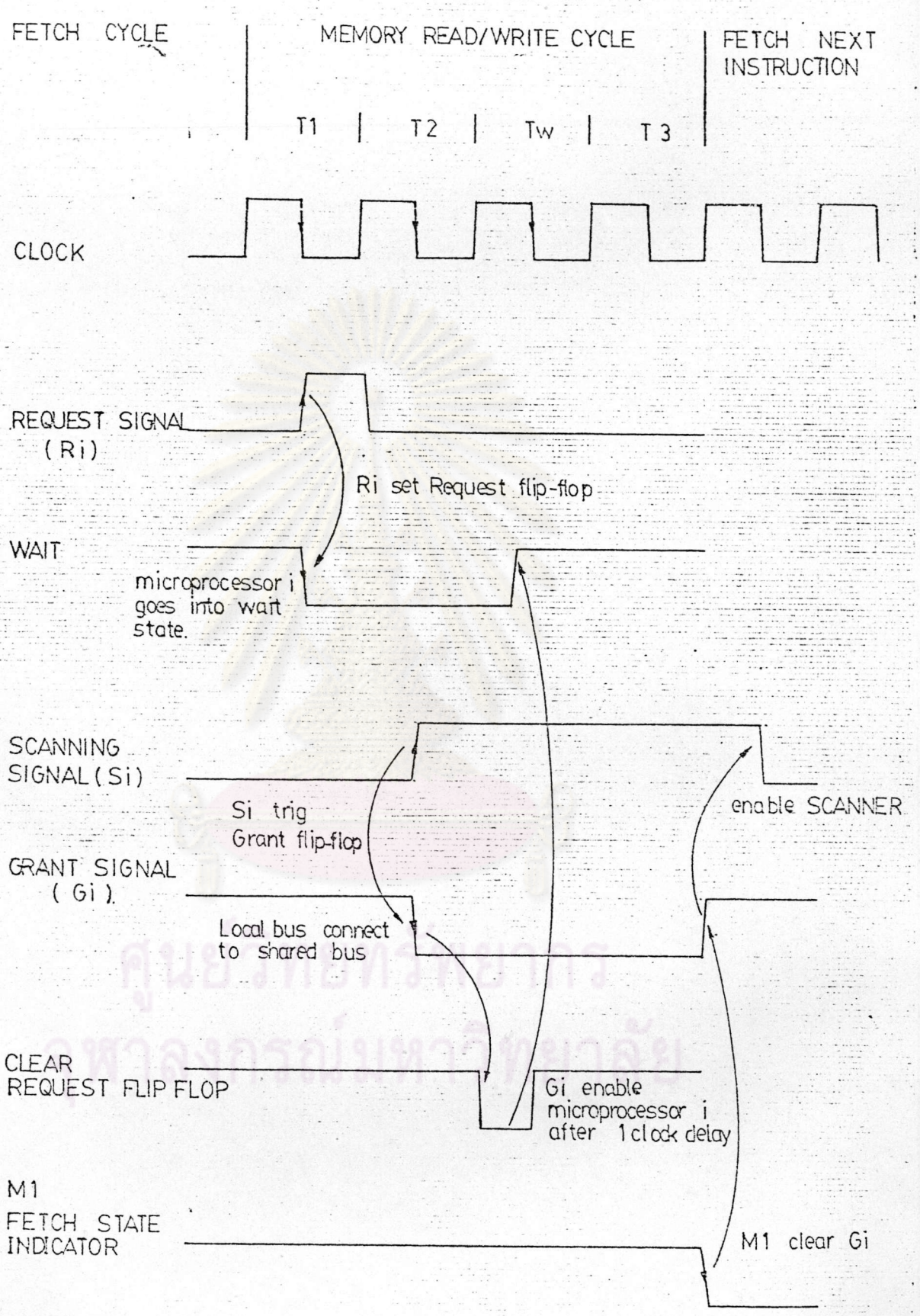
ในการวิจัยนี้ กำหนดให้ไมโครโปรเซสเซอร์ทุกตัวมี priority เท่ากันและมีสิทธิใช้หน่วยความจำร่วมหมุนเวียนกันไป (Ring Priority) สัญญาณ SCANNING สำหรับไมโครโปรเซสเซอร์แต่ละตัวจึงได้จากเอาท์พุทของ RING COUNTER ขนาด 4 บิท แต่หากต้องการใช้ priority ลักษณะอื่น ก็สามารถทำได้โดยใช้ Pattern Generator ชนิดอื่นแทน RING COUNTER RING COUNTER จะถูกขับโดยสัญญาณ clock จากไมโครโปรเซสเซอร์ตัวใดตัวหนึ่ง โดยผ่าน ENABLE GATE ซึ่ง ENABLE GATE จะทำหน้าที่หยุดสัญญาณ clock ไว้ไม่ให้ส่งไปยัง RING COUNTER ในทันทีที่มีไมโครโปรเซสเซอร์ใช้หน่วยความจำร่วม จนกระทั่งไมโครโปรเซสเซอร์เสร็จสิ้นการใช้หน่วยความจำร่วม ENABLE GATE จึงยอมให้ RING COUNTER ทำงานต่อไป ทั้งนี้เพื่อให้มีไมโครโปรเซสเซอร์เพียงตัวเดียวที่ใช้หน่วยความจำร่วมในเวลาหนึ่ง ๆ SCANNER จะมีเพียงชุดเดียวไม่ว่าระบบจะมีไมโครโปรเซสเซอร์กี่ตัวก็ตาม

3.2.2 CONTROLLER ไมโครโปรเซสเซอร์แต่ละตัวจะมี CONTROLLER เป็นของตนเอง โดยที่ CONTROLLER รับสัญญาณ REQUEST และส่งสัญญาณ WAIT และ GRANT กลับไปให้ไมโครโปรเซสเซอร์ในเวลาที่เหมาะสม

เมื่อไมโครโปรเซสเซอร์ต้องการใช้หน่วยความจำร่วม (รูป 3.8) จะส่งสัญญาณ REQUEST ไป clock REQUEST FLIP-FLOP ซึ่ง REQUEST FLIP-FLOP จะส่งสัญญาณ WAIT กลับมายังไมโครโปรเซสเซอร์ ทำให้ไมโครโปรเซสเซอร์ต้องอยู่ในสภาวะการรอ (Wait) ไปจนกว่าจะได้รับสัญญาณ GRANT และสัญญาณ GRANT จะเกิดขึ้นเมื่อ GRANT FLIP-FLOP ถูก trigger ด้วยสัญญาณ SCANNING จาก SCANNER



รูป 3.7 โครงสร้างของวงจร ARBITER



รูป 3.8 แผนผังเวลาของวงจร **ARBITER**

สัญญาณ GRANT จะ enable วงจรอินเตอร์เฟส ทำให้บัสของไมโครโปรเซสเซอร์ต่อเข้ากับบัสของหน่วยความจำร่วม และหลังจากหน่วงเวลาไว้ 1 clock เพื่อให้หน่วยความจำ RAM มีเวลาพอเพียงในการเตรียมตัวรับหรือส่งข้อมูลในแอดเดรสที่กำหนด สัญญาณ GRANT จะ clear REQUEST FLIP-FLOP ทำให้ไมโครโปรเซสเซอร์หลุดจากสภาวะการรอ และทำการเขียนหรืออ่านข้อมูลจากหน่วยความจำร่วมทันที ในเวลาเดียวกับที่สัญญาณ GRANT enable วงจรอินเตอร์เฟส มันจะถูกส่งไปยัง ENABLE GATE เพื่อทำให้ ENABLE GATE หยุดสัญญาณ clock ที่ส่งไปยัง RING COUNTER ไว้ระยะหนึ่ง จนกระทั่งไมโครโปรเซสเซอร์เสร็จสิ้นการใช้หน่วยความจำร่วมและกำลังจะเฟทช์คำสั่งต่อไป มันจะส่งสัญญาณ M1 บอกสภาวะเฟทช์ออกมา ซึ่งสัญญาณ M1 นี้จะไป clear GRANT FLIP-FLOP ทำให้สัญญาณ GRANT สิ้นสุดลง และทุกอย่างกลับสู่สภาพเริ่มต้นเพื่ออนุญาตให้ไมโครโปรเซสเซอร์ตัวอื่นใช้หน่วยความจำร่วมต่อไป

3.3 วงจร ARBITER และอินเตอร์เฟส หน่วยความจำร่วม RAM สำหรับไมโครโปรเซสเซอร์ Z-80 จำนวน 4 ตัว

วงจร ARBITER ที่ออกแบบและสร้างขึ้นนี้ สามารถนำไปใช้กับไมโครโปรเซสเซอร์ชนิดอื่นนอกจาก Z-80 ก็ได้ แต่ต้องปรับปรุงวงจร CONTROLLER และวงจรอินเตอร์เฟส ระหว่างบัสของไมโครโปรเซสเซอร์และหน่วยความจำร่วมให้เหมาะสมกับไมโครโปรเซสเซอร์ชนิดนั้น ๆ เนื่องจากในการวิจัยนี้ใช้ไมโครโปรเซสเซอร์ Z-80 ดังนั้น จะกล่าวถึงเฉพาะวงจรสำหรับใช้กับไมโครโปรเซสเซอร์ Z-80 เท่านั้น และเพื่อให้เข้าใจการทำงานได้ชัดเจนยิ่งขึ้น จะทบทวนการเขียนและอ่านข้อมูลจากหน่วยความจำของไมโครโปรเซสเซอร์ Z-80 พอสังเขป แล้วจึงจะกล่าวถึงรายละเอียดของวงจร SCANNER CONTROLLER และ อินเตอร์เฟส ตามลำดับ

3.3.1 การอ่านและเขียนข้อมูลในหน่วยความจำของไมโครโปรเซสเซอร์ Z-80 (18)

ไมโครโปรเซสเซอร์ Z-80 มีบัสแอดเดรส บัสข้อมูล และสัญญาณควบคุมที่เกี่ยวข้องกับหน่วยความจำดังนี้

1. บัสแอดเดรส A0 - A15

2. บัสข้อมูล D0 - D7

3. $\overline{M1}$ (Machine Cycle One) เป็นเอาท์พุทบอกให้ทราบว่าไมโครโปรเซสเซอร์อยู่ในสภาวะเฟทช์คำสั่ง

4. $\overline{MREQ}$ (Memory Request) เป็นเอาท์พุทบอกให้ทราบว่า ไมโครโปรเซสเซอร์ต้องการเขียนหรืออ่านข้อมูลจากหน่วยความจำ ตามแอดเดรสที่ปรากฏอยู่ในบัสแอดเดรส

5. $\overline{RD}$ (Memory Read) เป็นเอาท์พุทบอกให้ทราบว่าไมโครโปรเซสเซอร์ต้องการอ่านข้อมูลจากหน่วยความจำ

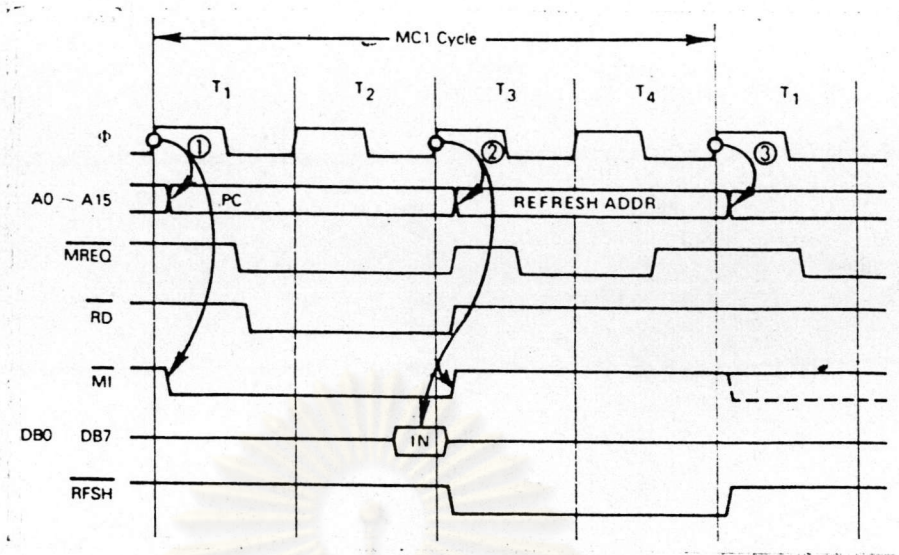
6. $\overline{WR}$ (Memory Write) เป็นเอาท์พุทบอกให้ทราบว่าไมโครโปรเซสเซอร์ต้องการเขียนข้อมูลในหน่วยความจำ

7. $\overline{WAIT}$ (Wait) เป็นอินพุท สั่งให้ไมโครโปรเซสเซอร์หยุดรอ

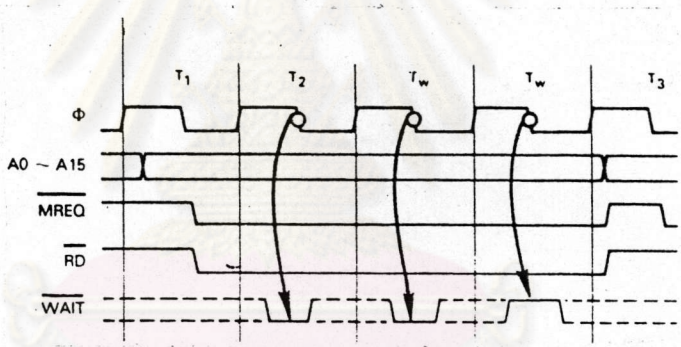
การทำงานของไมโครโปรเซสเซอร์ Z-80 ในแต่ละไซเคิลของคำสั่งประกอบด้วยการทำงานสองจังหวะคือ การเฟทช์คำสั่ง ได้แก่ การอ่านคำสั่งจากหน่วยความจำที่เก็บโปรแกรมแล้วมาถอดรหัส และการเอ็กซ์คิวได้แก่ การทำงานตามคำสั่งที่ได้จากการเฟทช์ ซึ่งประกอบด้วยแมชีนไซเคิล (Machine cycle) หลายแบบ แต่ที่เกี่ยวข้องในที่นี้คือ ไซเคิลการอ่านและเขียนหน่วยความจำ อนึ่งจะเห็นว่าขณะที่ไมโครโปรเซสเซอร์ทำการเฟทช์คำสั่งนั้น จะไม่เกี่ยวข้องกับหน่วยความจำร่วม และหน่วยความจำร่วมถูกใช้งานเฉพาะในขณะที่ไมโครโปรเซสเซอร์อยู่ในสภาวะเอ็กซ์คิวเท่านั้น

3.3.1.1 ไซเคิลการเฟทช์คำสั่ง

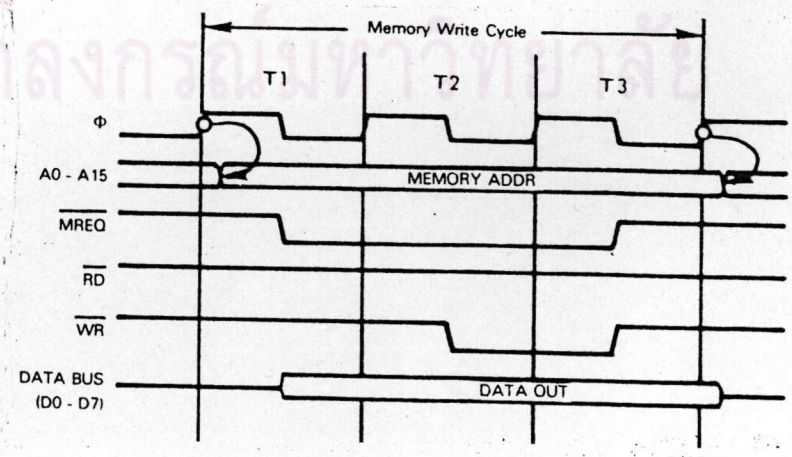
เมื่อเริ่มเข้าสู่สภาวะเฟทช์ สัญญาณ M1 จะเปลี่ยนเป็น 0 พร้อมกับข้อมูลจากโปรแกรมเคาน์เตอร์ (program counter) จะถ่ายลงไปยังบัสแอดเดรสและหลังจากนั้น ไมโครโปรเซสเซอร์จะส่งสัญญาณ $\overline{MREQ}$ ออกมาพร้อมกับสัญญาณ $\overline{RD}$ และไมโครโปรเซสเซอร์จะอ่านข้อมูลจากบัสข้อมูลในช่วง clock ถัดมา แล้วนำไปเก็บไว้ใน Instruction Register เพื่อถอดรหัสต่อไป เมื่อสิ้นสุดสภาวะเฟทช์ สัญญาณ M1 จะกลับเป็น 1 ตามเดิม ดังรูป 3.9



รูป 3.9 การเฟทช์คำสั่งของ Z-80



รูป 3.10 การอ่านข้อมูลจากหน่วยความจำของ Z-80



รูป 3.11 การเขียนข้อมูลลงในหน่วยความจำของ Z-80

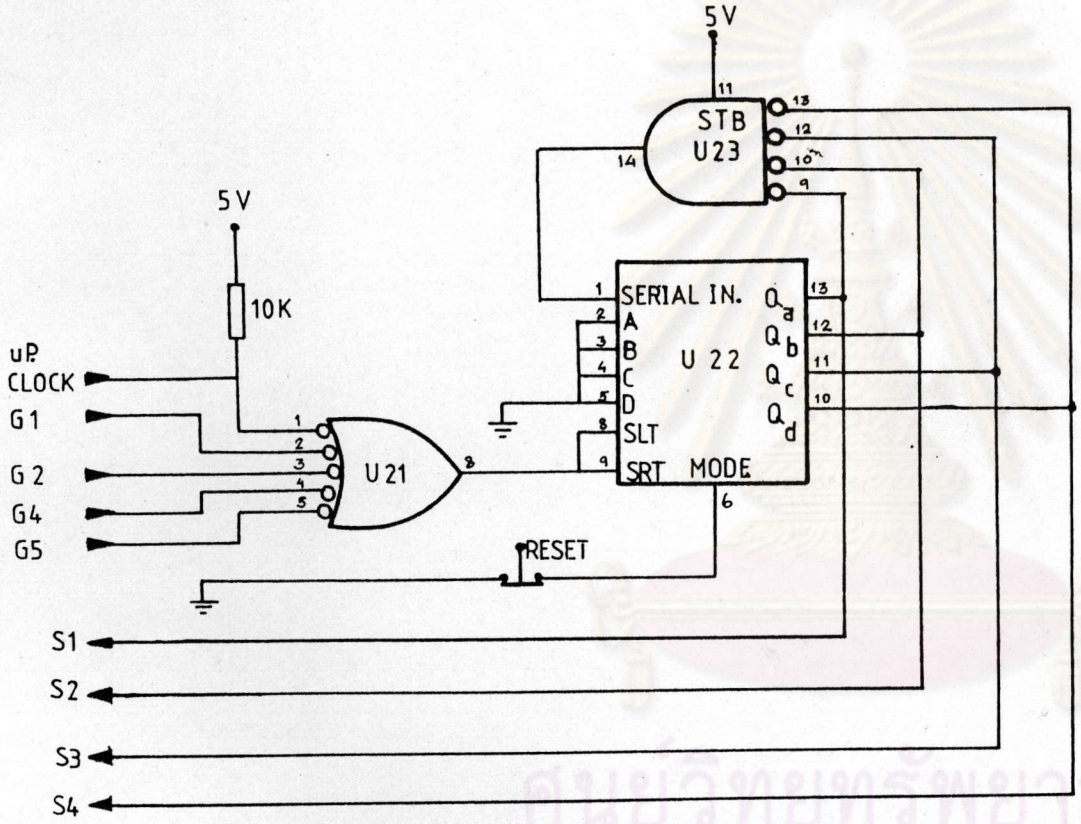
3.3.1.2 ไซเคิลการอ่านและเขียนหน่วยความจำ

หลังจากที่ไมโครโปรเซสเซอร์เฟทช์คำสั่ง และตีความว่าเป็นคำสั่งให้อ่านหรือเขียนข้อมูลลงในหน่วยความจำแล้ว ถ้าเป็นการอ่านข้อมูล (รูป 3.10) ไมโครโปรเซสเซอร์จะส่งแอดเดรสที่ต้องการอ่านเข้าสู่บัสแอดเดรสในตอนแรก ต่อจากนั้นจะส่งสัญญาณ $\overline{MREQ}$ และ $\overline{RD}$ ออกมา และทุกครั้งของขาลงในช่วงจังหวะ T2 ไมโครโปรเซสเซอร์จะตรวจสอบสัญญาณ $\overline{WAIT}$ ว่าเป็น 0 หรือไม่ ถ้าไม่ใช่ไมโครโปรเซสเซอร์ก็จะทำงานของตนต่อไป แต่ถ้าเป็น 0 ก็จะสร้าง Tw (Wait state) เพิ่มขึ้น การอ่านข้อมูลของไมโครโปรเซสเซอร์จะเกิดขึ้นในขาลงของ T3 และเป็นการสิ้นสุดการอ่านข้อมูลจากหน่วยความจำ

ในทำนองเดียวกัน ในการเขียนข้อมูลลงในหน่วยความจำ (รูป 3.11) จะประกอบด้วย T1, T2 และ T3 รวมทั้ง Tw ในกรณีที่ขา $\overline{WAIT}$ เป็น 0 ที่ขาลงของ T2 แต่ต่างกันที่ไมโครโปรเซสเซอร์จะสร้างสัญญาณ $\overline{MREQ}$ และ $\overline{WR}$ แทนสัญญาณ $\overline{RD}$ โดยข้อมูลจะส่งไปไว้ที่บัสข้อมูลล่วงหน้าและยาวนานกว่าสัญญาณ $\overline{WR}$ ขอให้สังเกตด้วยว่า ตราบเท่าที่ขา $\overline{WAIT}$ ยังเป็น 0 ที่ขาลงของ T2 ไมโครโปรเซสเซอร์จะสร้าง Tw (Wait state) ขึ้นเสมอไป และอีกประการหนึ่งคือไมโครโปรเซสเซอร์จะสร้างสัญญาณ $\overline{M1}$ ออกมาที่ขาขึ้นของ T1 ในตอนเริ่มต้นสภาวะเฟทช์ทุกครั้ง

3.3.2 วงจร SCANNER

รูป 3.12 เป็นวงจร SCANNER สำหรับระบบที่มีไมโครโปรเซสเซอร์ใช้หน่วยความจำร่วมไม่เกิน 4 ตัว U22 และ U23 ร่วมกันทำหน้าที่เป็น 4-BIT RING COUNTER สร้างสัญญาณ SCANNING (S1, S2, S3, S4) ทีละบิทเรียงกันตามลำดับและหมุนเวียนกันไป โดยมีสัญญาณ clock จากไมโครโปรเซสเซอร์เป็นตัวขับให้มีการเลื่อนบิทจาก S1 ไป S2 และต่อไปตามลำดับ U21 ทำหน้าที่ ENABLE GATE คอยส่งสัญญาณ clock ให้กับ RING COUNTER ไปจนกว่าจะมีอินพุท $\overline{G1}$ ถึง $\overline{G4}$ ซึ่งเป็นสัญญาณ GRANT จาก CONTROLLER ของไมโครโปรเซสเซอร์ตัวใดตัวหนึ่งกลายเป็น 0 จะทำให้เอาท์พุทของ U21 กลายเป็น 0 อยู่ตลอดช่วงระยะหนึ่ง เอาท์พุทของ RING COUNTER (U22) จึงไม่มีการเปลี่ยนแปลง และมีสภาพเดิมไปจนกระทั่งสัญญาณ GRANT สิ้นสุดลง สัญญาณ clock จึงจะสามารถผ่าน ENABLE GATE (U21) ไปยัง RING COUNTER (U22) ได้ตามเดิม ดังนั้น จะเห็นว่ามีไมโครโปรเซสเซอร์เพียงตัวเดียวเท่านั้นที่ได้รับ



		5V	GND
U 21	74LS 30	14	7
U 22	7495	14	7
U23	7425	14	7

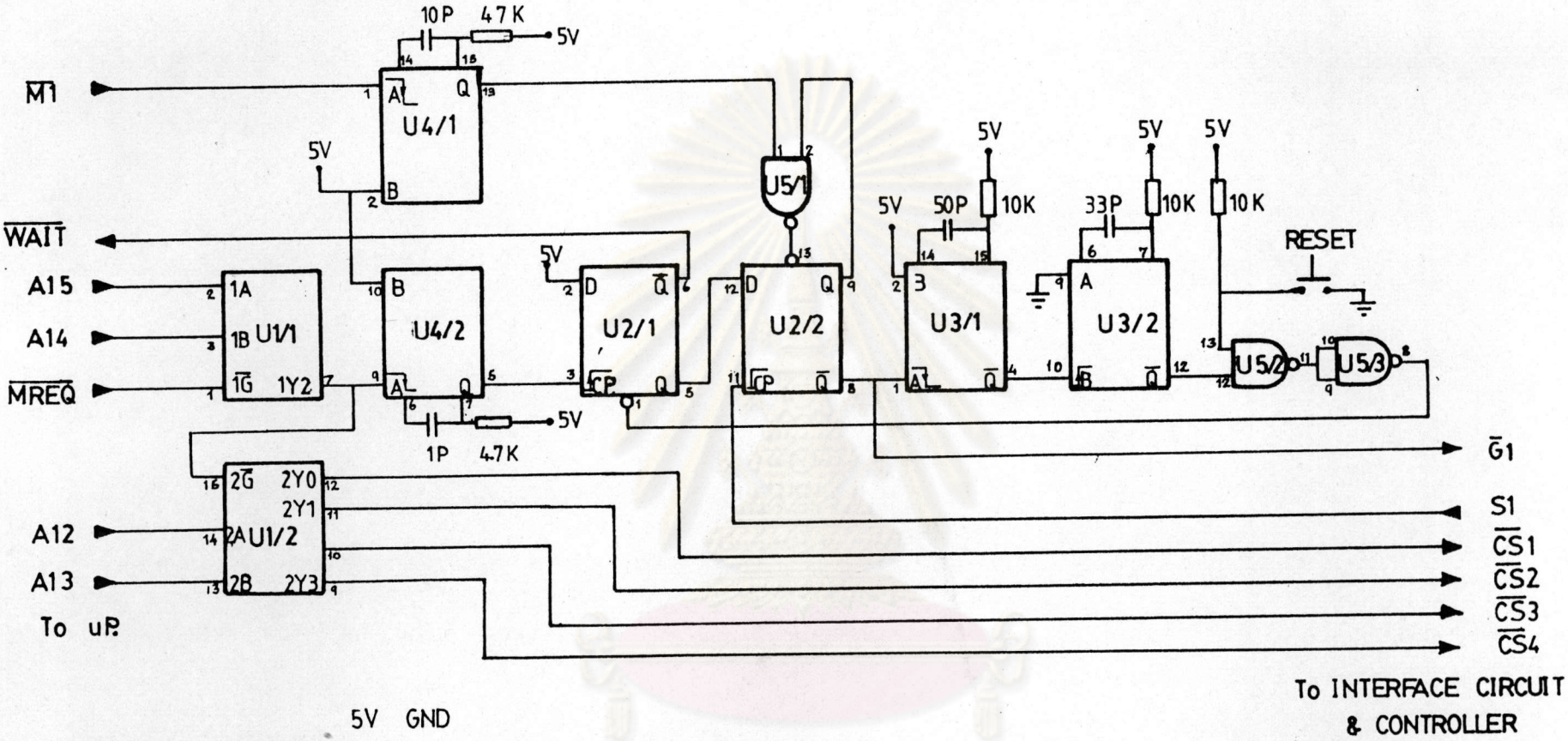
รูป 3.12 วงจร SCANNER

สัญญาณ SCANNING ที่เป็น 1 จากวงจร SCANNER ไม่ว่าจะมีสัญญาณ REQUEST กี่ตัวก็ตาม

3.3.3 วงจร CONTROLLER

วงจร CONTROLLER ในรูป 3.13 ใช้ได้กับไมโครโปรเซสเซอร์ Z-80 เท่านั้น หากต้องการให้ไมโครโปรเซสเซอร์ชนิดอื่นใช้หน่วยความจำร่วมกันด้วย ต้องดัดแปลงวงจร CONTROLLER ให้เหมาะสมกับคุณสมบัติของไมโครโปรเซสเซอร์ชนิดนั้น และไมโครโปรเซสเซอร์ทุกตัวที่ใช้หน่วยความจำร่วมต้องมีวงจร CONTROLLER ชุดนี้เพิ่มเติมขึ้นจึงจะทำงานได้

U1/1 เป็น Address Decoder แปลงสัญญาณที่ส่งมาจากไมโครโปรเซสเซอร์ให้เป็นสัญญาณ REQUEST (Ri) ที่จะไป clock REQUEST FLIP-FLOP (U2/1) โดยผ่าน Monostable U4/2 เพื่อจำกัดความกว้างของสัญญาณ Ri ที่ไป clock REQUEST FLIP-FLOP ไว้เพียง 50 ns และเปลี่ยน TRIGGER จากขาลงเป็นขาขึ้น ตามความต้องการของ REQUEST FLIP-FLOP (U2/1) เอาท์พุท $\overline{Q}$ ของ REQUEST FLIP-FLOP จะเป็นสัญญาณ WAIT ส่งกลับไปยังไมโครโปรเซสเซอร์ ในขณะที่เอาท์พุท Q จะป้อนเป็น data ให้กับ GRANT FLIP-FLOP (U2/2) เมื่อถึงลำดับที่ไมโครโปรเซสเซอร์ตัวนี้มีสิทธิ์ใช้หน่วยความจำร่วม SCANNER จะส่งสัญญาณ SCANNING (Si) มา clock GRANT FLIP-FLOP (U2/2) เอาท์พุท $\overline{Q}$ ของ U2/2 จะเป็นสัญญาณ GRANT ($\overline{Gi}$) ส่งไป enable วงจรอินเตอร์เฟส ระหว่างบัสของไมโครโปรเซสเซอร์และหน่วยความจำร่วม พร้อมกับส่งไปยัง ENABLE GATE (U21) เพื่อทำให้ RING COUNTER หยุดทำงานชั่วระยะหนึ่ง และสัญญาณ GRANT จะถูก delay โดย Monostable U3/1 ไว้ไม่เกิน 1 clock period (500 ns ที่ 2 MHz) แล้วจึงไป clear REQUEST FLIP-FLOP (U2/1) ทำให้ขา WAIT ($\overline{Q}$) กลายเป็น 1 ความยาวของสัญญาณที่ไป clear U2/1 กำหนดโดย Monostable U3/2 ซึ่งจะต้องนานพอที่จะทำให้ไมโครโปรเซสเซอร์หลุดจากสภาวะการรอได้ โดยปกติมีค่าตั้งแต่ 100 – 500 ns เมื่อไมโครโปรเซสเซอร์สิ้นสุดการใช้หน่วยความจำร่วมและเริ่มเฟทช์คำสั่งถัดไป มันจะส่งสัญญาณ $\overline{M1}$ ออกมา clear GRANT FLIP-FLOP (U2/2) โดย Monostable U4/1 เปลี่ยนสัญญาณ $\overline{M1}$ ให้เป็นสัญญาณ pulse ที่มีขนาดพอเพียงที่จะ clear U2/2 ทำให้ $\overline{Q}$ กลายเป็น 1 และสัญญาณ



		5V	GND
U1	74LS139	16	8
U2	74LS74	14	7
U3	74LS123	16	8
U4	74LS123	16	3
U5	74LS00	14	7

All resistors are in ohm.
All capacitors are in farad.

รูป 3.13 วงจร CONTROLLER

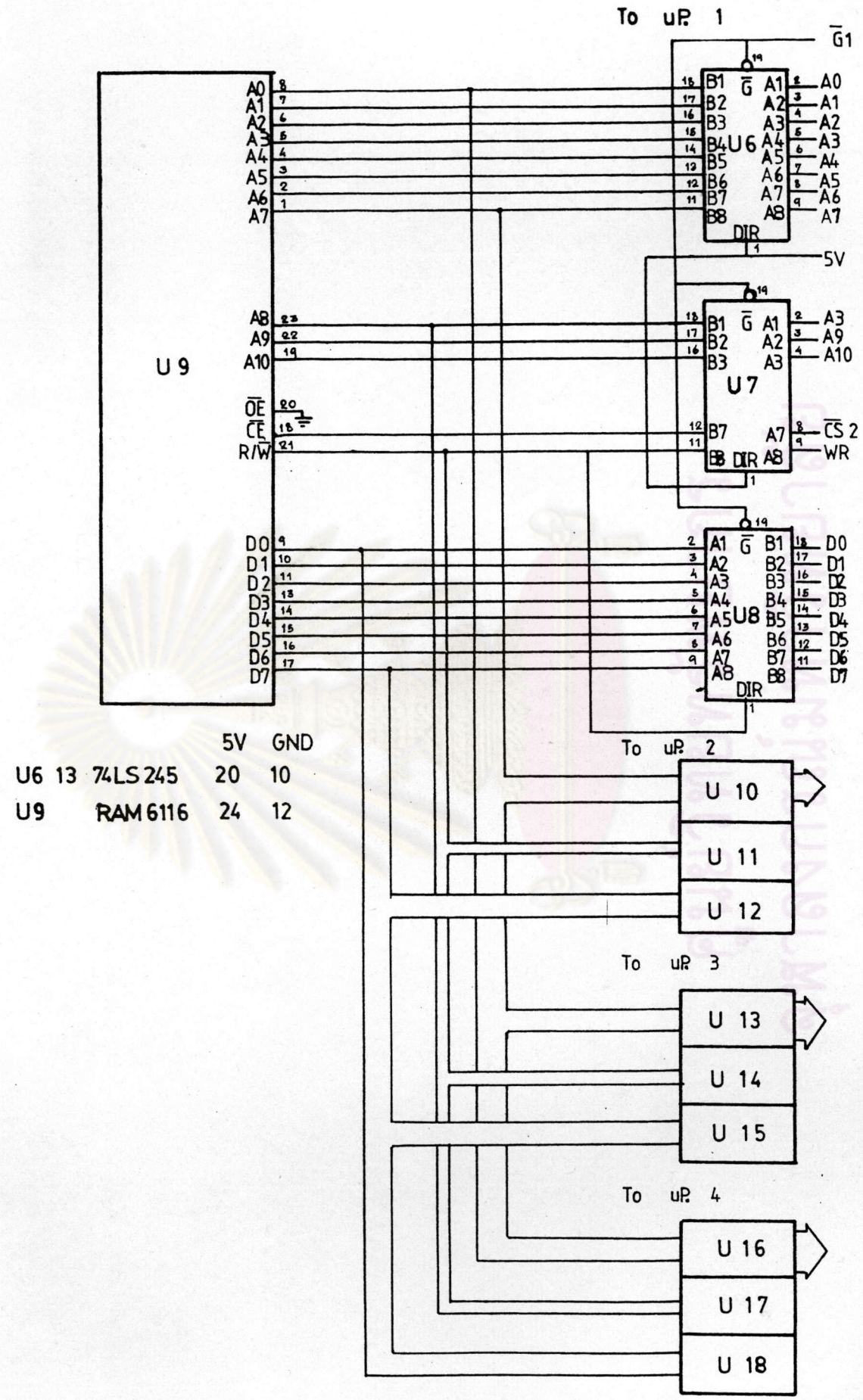
GRANT จึงสิ้นสุดลง NAND gate U5/1 ทำหน้าที่กำจัดการรบกวนของสัญญาณ $\overline{M1}$ ที่มีต่อ GRANT FLIP-FLOP ให้มีอิทธิพลเฉพาะในช่วงที่ไมโครโปรเซสเซอร์กำลังใช้หน่วยความจำร่วมเท่านั้น ($\overline{M1}$ จะมีผลต่อ U2/2 ก็ต่อเมื่อเอาท์พุท Q ของ U2/2 เป็น 1 เท่านั้น) NAND gate U5/2 และ U5/3 เป็นส่วนพิเศษที่เพิ่มขึ้น จะมีหรือไม่ก็ได้ ทำหน้าที่สร้างสัญญาณ RESET ให้ไมโครโปรเซสเซอร์ออกจากสภาวะการรอได้ในกรณีที่เกิดความผิดพลาดขึ้นกับวงจรอื่น

จากวงจร CONTROLLER จะเห็นว่า ในสภาวะปกติที่ไมโครโปรเซสเซอร์ทำงานอยู่กับหน่วยความจำส่วนตัว วงจร CONTROLLER จะไม่มีบทบาทใด ๆ และวงจร CONTROLLER จะมีผลต่อการทำงานของไมโครโปรเซสเซอร์ต่อเมื่อมีการใช้หน่วยความจำร่วมเท่านั้น

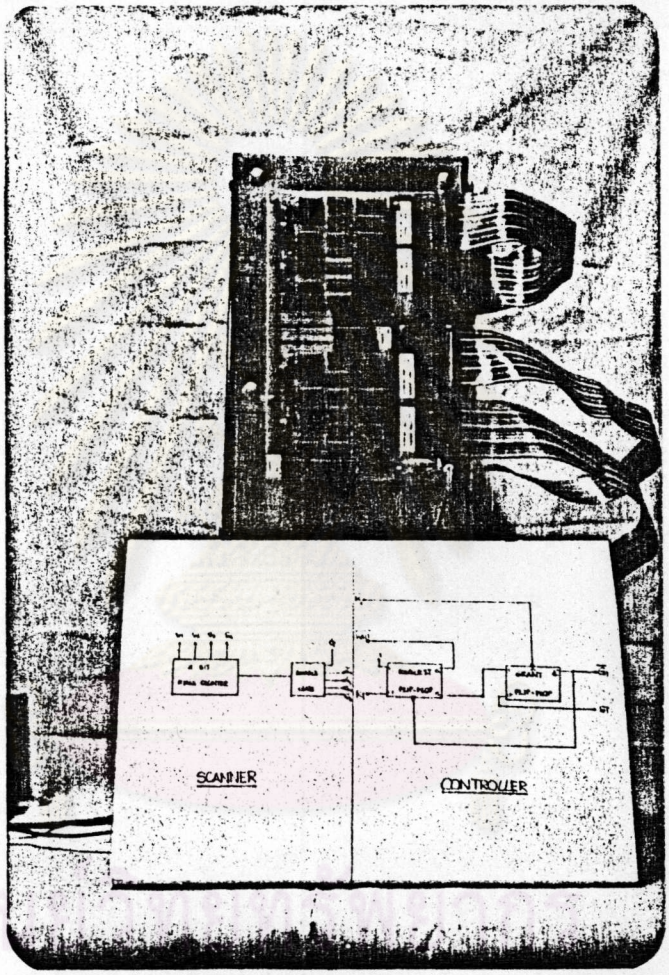
3.3.4 วงจรอินเตอร์เฟสระหว่างบัสของไมโครโปรเซสเซอร์และหน่วยความจำร่วม

เนื่องจากหน่วยความจำ RAM มีบัสเพียงชุดเดียวที่ใช้รับส่งข้อมูล จึงไม่สามารถอินเตอร์เฟสเข้ากับบัสของไมโครโปรเซสเซอร์ 4 ตัว โดยตรงพร้อมกันได้ จำเป็นต้องมี Bus Isolator เป็นตัวกลางระหว่างบัสของหน่วยความจำและไมโครโปรเซสเซอร์แต่ละตัว ในภาวะปกติที่ไม่มีการใช้หน่วยความจำร่วม Bus Isolator จะมีสภาพเป็น high impedance เสมือนว่า หน่วยความจำร่วมมิได้ต่ออยู่กับไมโครโปรเซสเซอร์ แต่ถ้าไมโครโปรเซสเซอร์ตัวใดได้รับสัญญาณ GRANT อนุญาตให้ใช้หน่วยความจำร่วมแล้ว Bus Isolator จะต่อบัสของไมโครโปรเซสเซอร์ตัวนั้นเข้ากับหน่วยความจำร่วมทันที โดยที่ Bus Isolator ของไมโครโปรเซสเซอร์ตัวอื่นจะคงมีสภาพเป็น high impedance ตามเดิม ดังนั้นในช่วงเวลาหนึ่ง ๆ จึงเสมือนว่ามีไมโครโปรเซสเซอร์เพียงตัวเดียวที่อินเตอร์เฟสกับหน่วยความจำร่วม

รูป 3.14 แสดงวงจรอินเตอร์เฟสบัสของหน่วยความจำร่วม RAM (U9) เข้ากับไมโครโปรเซสเซอร์ 4 ตัว U6 - U18 ทำหน้าที่ Bus Isolator โดยที่ U6, U10, U13 และ U16 ทำหน้าที่ isolate บัสแอดเดรส A0 - A7. U7, U11, U14 และ U17 ทำหน้าที่ isolate บัสแอดเดรส A8-A10 และบัสควบคุม ส่วน U8, U12, U15 และ U18 ทำหน้าที่ isolate บัสข้อมูล U6, U7 และ U8 จะ enable โดย

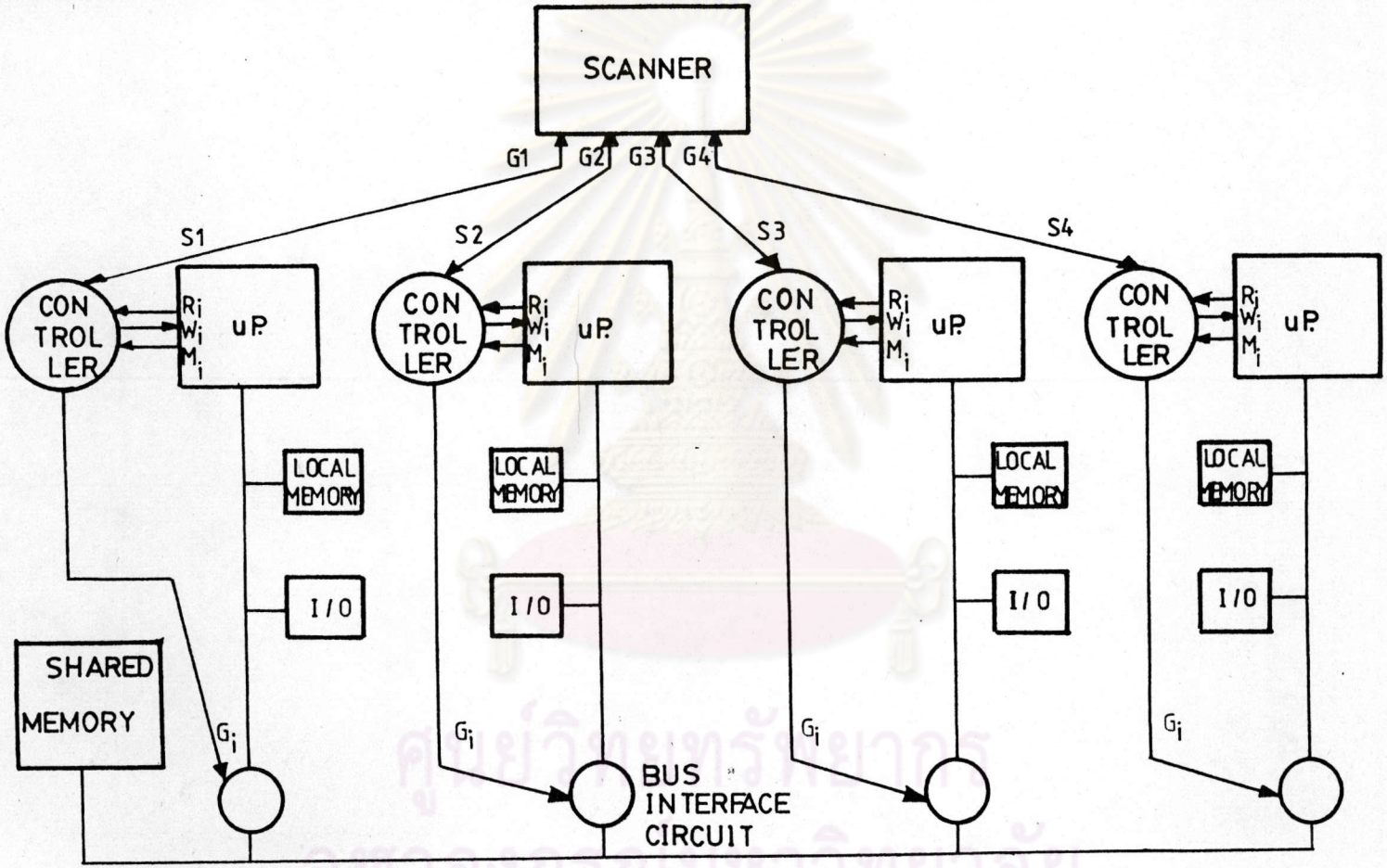


รูป 3.14 วงจรอินเตอร์เฟสระหว่างบัสของไมโครโปรเซสเซอร์และหน่วยความจำรวม

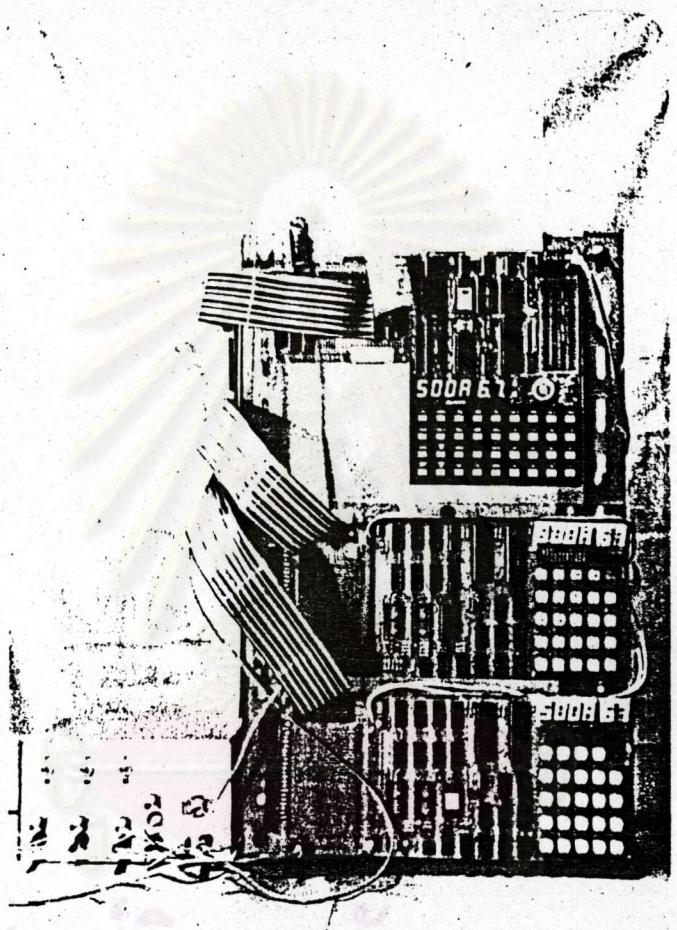


รูป 3.15 แผงวงจร ARBITER และหน่วยความจำร่วม

* Only 3 microprocessors are used in the experiment.



รูป 3.16 โครงสร้างของระบบไมโครโปรเซสเซอร์แบบกระจายที่สร้างขึ้น



รูป 3.17 ระบบไมโครโปรเซสเซอร์แบบกระจายที่สร้างขึ้น

สัญญาณ GRANT $\overline{G1}$ จาก CONTROLLER 1 และในทำนองเดียวกัน U10, U11 และ U12 จะ enable โดยสัญญาณ GRANT $\overline{G2}$ U13, U14 และ U15 enable ได้โดย สัญญาณ GRANT $\overline{G3}$ และสัญญาณ GRANT $\overline{G4}$ จะ enable U16, U17 และ U18 ขอให้สังเกตว่า ถ้าต้องการขยายของหน่วยความจำร่วมให้ใหญ่ขึ้น สามารถทำได้โดย เพิ่มหน่วยความจำ RAM เพียงอย่างเดียว ไม่ต้องเพิ่ม Bus Isolator ขึ้นอีก

วงจร ARBITER ที่ออกแบบและสร้างขึ้นทดลอง (รูป 3.15) สามารถใช้ รวมกลุ่มไมโครโปรเซสเซอร์ Z-80 สามตัวเข้าเป็นระบบไมโครโปรเซสเซอร์แบบ กระจายที่ใช้หน่วยความจำร่วม (รูป 3.16) โดยไมโครโปรเซสเซอร์แต่ละตัวมีคุณสมบัติ ต่างกันออกไปกล่าวคือ ตัวหนึ่งใช้สัญญาณ clock 1.78 MHz ในขณะที่อีกสองตัวใช้ สัญญาณ clock 2.5 MHz และมีโครงสร้างที่ต่างกันอย่างชัดเจน (รูป 3.17) จะเห็น ว่าวงจร ARBITER ที่ออกแบบสร้างได้โดยใช้ฮาร์ดแวร์ที่ไม่ยุ่งยาก และใช้รวมกลุ่ม ไมโครโปรเซสเซอร์ที่มีลักษณะต่างกันได้ จึงเหมาะสมที่จะใช้กับระบบไมโครโปรเซสเซอร์ แบบกระจายขนาดเล็กมากกว่าวิธีอื่นที่เคยมีผู้เสนอมาแล้ว