บทที่ 4

การออกแบบและผลการออกแบบเครื่องจัดการจราจรในระบบเครือข่าย

4.1 การทำงานของเครื่องจัดการจราจรในระบบเครือข่าย

เครื่องจัดการจราจรในระบบเครือข่ายในวิทยานิพนธ์นี้แบ่งการทำงานออกเป็น 4 ส่วนคือ

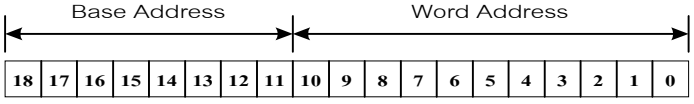
- การติดต่อกับหน่วยความจำหลัก
- การส่งแพ็คเกต
- การรับแพ็คเกต
- การติดต่อกับ MII

เครื่องจัดการจราจรในระบบเครือข่ายนี้รับส่งข้อมูลแบบฟูลดูเพล็กซ์ (Full-Duplex) ผ่านทาง MII ที่อัตราเร็วในการรับส่งข้อมูล 100 เมกกะบิตต่อวินาทีและมีรายละเอียดการทำงานในแต่ละส่วนดังนี้

4.1.1 **การติดต่อกับหน่วยความจำหลัก** ในการติดต่อกับหน่วยความจำหลักมีฮาร์ดแวร์ทำหน้าที่เสมือนเป็น DMA ช่วยในการอ่านและเขียนข้อมูล

4.1.1.1 การทำงานเสมือนเป็น DMA การทำงานในส่วนนี้เสมือนว่ามี DMA อยู่ภายในช่วยไมโครโปรเซสเซอร์ในการอ่านและเขียนข้อมูลลงในหน่วยความจำ ไมโครโปรเซสเซอร์เพียงแต่ระบุตำแหน่งเริ่มต้นในการจัดเก็บข้อมูลจากนั้นฮาร์ดแวร์จะดำเนินการจนสิ้นสุดการรับหรือส่งข้อมูล

4.1.1.2 พอยน์เตอร์ชี้ตำแหน่งหน่วยความจำ กระบวนการรับส่งข้อมูลจำเป็นต้องใช้พอยน์เตอร์ชี้ตำแหน่งของหน่วยความจำเพื่อระบุตำแหน่งของข้อมูลที่ต้องการรับหรือส่ง ในขณะทำการรับหรือส่งข้อมูล พอยน์เตอร์จะถูกปรับค่าโดยอัตโนมัติ



ภาพประกอบ 4-1 พอยน์เตอร์ชี้ตำแหน่งหน่วยความจำ

พอยน์เตอร์ชี้ตำแหน่งหน่วยความจำมีขนาด 19 บิตดังแสดงในภาพประกอบ 4-1 โดย 8 บิตบนคือค่า *Base Address* และ 11 บิตล่างคือค่า *Word Address* ใช้ระบุตำแหน่งไบต์ของ

แพ็คเกตได้สูงสุด 2^{11} หรือ 2 กิโลไบต์ ดังนั้นพอยน์เตอร์ชี้ตำแหน่งหน่วยความจำสามารถระบุตำแหน่งแพ็คเกตได้สูงสุด 2^{8} หรือเท่ากับ 256 แพ็คเกต

4.1.2 การส่งแพ็คเกต ในการส่งแพ็คเกตกระบวนการที่ต้องดำเนินการคือ

4.1.2.1 กำหนดค่า *Base Address* เพื่อระบุแพ็คเกตที่ต้องการส่ง

4.1.2.2 กำหนดค่าเริ่มต้นในการนับเพื่อระบุจำนวนไบต์ของแพ็คเกต

4.1.2.3 ส่งสัญญาณร้องขอการเข้าถึงหน่วยความจำหลัก

4.1.2.4 อ่านข้อมูลจากหน่วยความจำหลัก

4.1.2.5 สร้างสัญญาณ Preamble และ SFD เพื่อบอกการเริ่มต้นการส่งข้อมูล

4.1.2.6 แปลงสัญญาณข้อมูลจากขนานเป็นอนุกรม

4.1.2.7 นับจำนวนไบต์ของข้อมูลที่ส่งออกและปรับค่าพอยน์เตอร์ชี้ตำแหน่งข้อมูล

การส่งข้อมูลเริ่มต้นหลังจากวงจรภาคส่งได้รับอนุญาตให้สามารถเข้าถึงหน่วยความจำได้ สัญญาณ Preamble และ SFD จะถูกสร้างขึ้นเพื่อบอกการเริ่มต้นการส่งข้อมูลก่อนจะส่งข้อมูลจริงออกไป ในขณะที่ทำการส่งสัญญาณ Preamble และ SFD นั้นไมโครโปรเซสเซอร์จะทำการกำหนดค่า *Base Address* และค่าเริ่มต้นในการนับของแพ็คเกตที่ต้องการส่งออกเพื่อบอกตำแหน่งและจำนวนไบต์ของแพ็คเกตตามลำดับ หลังจากนั้นข้อมูลจะถูกอ่านจากหน่วยความจำหลักผ่านทางบัสข้อมูลขนาด 16 บิตไปเก็บไว้ในรีจิสเตอร์ภายใน เมื่อ SFD 4 บิตสุดท้ายถูกส่งออกไป ข้อมูลที่จัดเก็บไว้จะถูกแปลงเป็นสัญญาณอนุกรมและส่งออกทีละ 4 บิตด้วยอัตราเร็ว 25 เมกกะบิตต่อวินาที ขณะเดียวกันก็จะนับจำนวนไบต์ของแพ็คเกตที่ส่งออกไปจนกว่าจะครบจึงจะสิ้นสุดการส่งข้อมูล

4.1.3 การรับแพ็คเกต ในการรับข้อมูลกระบวนการที่ต้องดำเนินการคือ

4.1.3.1 กำหนดค่า *Base Address* เพื่อระบุตำแหน่งที่ต้องการจัดเก็บแพ็คเกต

4.1.3.2 ตรวจหา SFD เพื่อหาจุดเริ่มต้นของแพ็คเกตที่ได้รับ

4.1.3.3 แปลงสัญญาณข้อมูลจากอนุกรมเป็นขนาน

4.1.3.4 ส่งสัญญาณร้องขอการเข้าถึงหน่วยความจำหลัก

4.1.3.5 เขียนข้อมูลลงในหน่วยความจำหลัก

4.1.3.6 เขียนข้อมูลที่ใช้ในการจัดคิวลงในหน่วยความจำของไมโครโปรเซสเซอร์

4.1.3.7 นับจำนวนไบต์ของข้อมูลที่จัดเก็บและปรับค่าพอยน์เตอร์ชี้ตำแหน่งของข้อมูล

การรับข้อมูลนั้นไมโครโปรเซสเซอร์จะกำหนดค่า *Base Address* เพื่อระบุตำแหน่งที่ต้องการจัดเก็บแพ็คเกต เมื่อตรวจพบ SFD วงจรภาครับจะร้องขอการเข้าถึงหน่วยความจำ หลัง

จากได้รับอนุญาตให้เข้าถึงหน่วยความจำได้ ข้อมูลที่ได้รับเข้ามาจะถูกแปลงจากสัญญาณอนุกรมขนาด 4 บิตไปเป็นข้อมูลขนานขนาด 16 บิตเพื่อจัดเก็บในหน่วยความจำหลัก เมื่อแปลงข้อมูลครบ 16 บิตหรือ 2 ไบต์ พอยน์เตอร์ชี้ตำแหน่งหน่วยความจำจะเพิ่มค่าขึ้น 1 ค่า ในขณะที่กำลังรับข้อมูลนั้นจะทำการเก็บข้อมูลที่ใช้สำหรับจัดคิวได้แก่ ชนิดของแพ็คเกต ไอพีแอดเดรสของต้นทางและไอพีแอดเดรสของปลายทางลงในหน่วยความจำของไมโครโปรเซสเซอร์

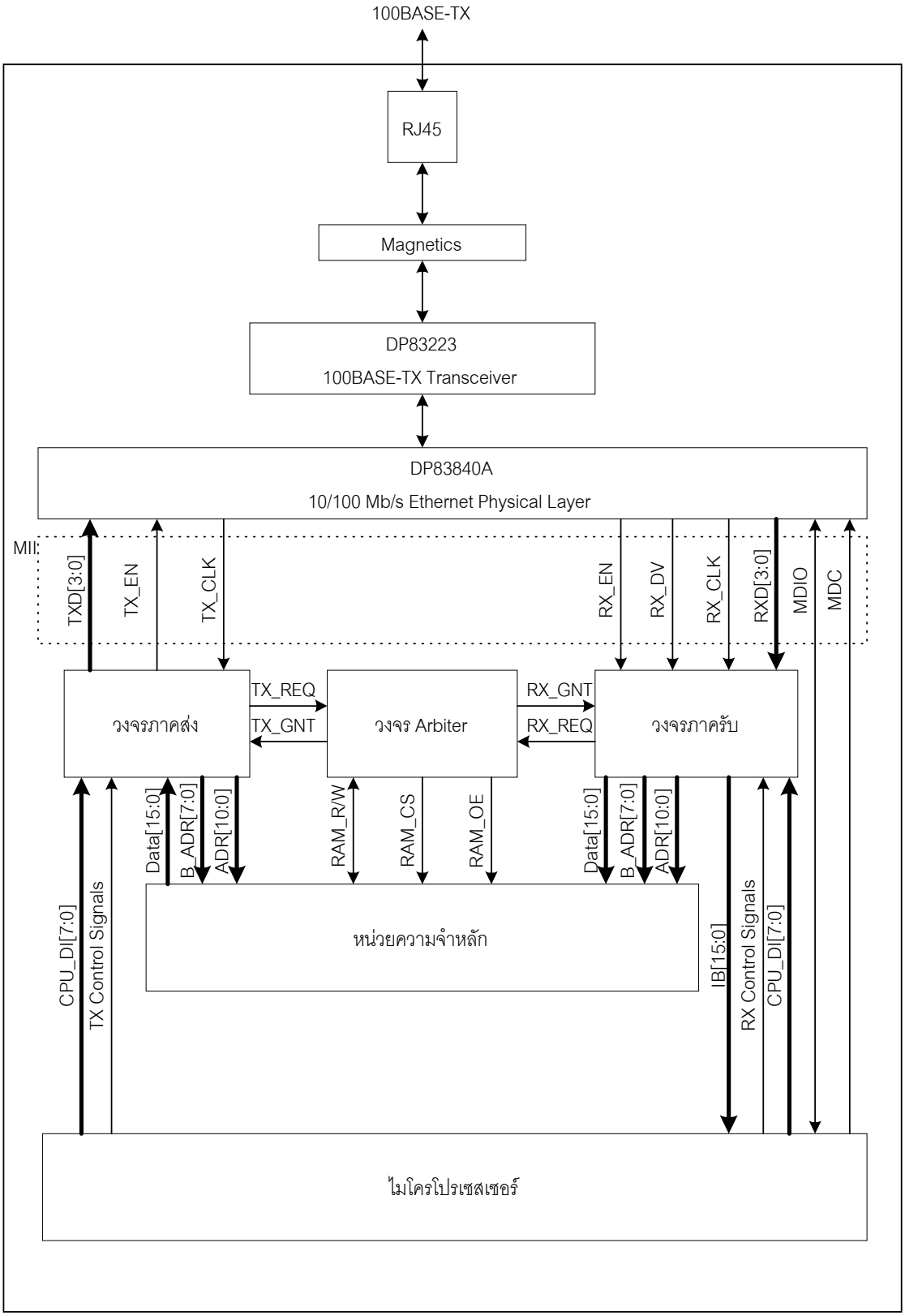
4.1.4 **การติดต่อกับ MII** ชิพ DP83840A ถูกนำมาช่วยในการติดต่อกับ MII เพื่อรับส่งข้อมูล คือ สัญญาณ RXD[3:0] และ TXD[3:0] ผ่านทาง MII และใช้สัญญาณ RX_CLK และ TX_CLK เป็นสัญญาณอ้างอิงในการรับส่งข้อมูลของวงจรภาครับและส่ง ส่วนสัญญาณ RX_DV และ TX_EN ใช้ในการบอกสถานะของการรับส่งข้อมูลผ่านบัสข้อมูล โดยวงจรภาครับและส่งจะได้รับสัญญาณเหล่านี้จากชิพ DP83840A นอกเหนือจากสัญญาณดังกล่าว ไมโครโปรเซสเซอร์ใช้สัญญาณ MDC และ MDIO เพื่อควบคุมการรับส่งข้อมูลผ่านทาง MII

4.2 ส่วนประกอบของเครื่องจัดการจราจรในระบบเครือข่าย

สถาปัตยกรรมของเครื่องจัดการจราจรในระบบเครือข่ายที่ออกแบบในวิทยานิพนธ์นี้แสดงดังภาพประกอบ 4-2 ซึ่งมีรายละเอียดดังนี้คือ

4.2.1 **ไมโครโปรเซสเซอร์** เป็นหัวใจหลักของเครื่องจัดการจราจรในระบบเครือข่าย ทำหน้าที่ควบคุมการทำงานทั้งหมดของระบบและประมวลผลข้อมูลต่าง ๆ ดังนี้คือ

- ควบคุมการทำงานของวงจร *Arbiter* โดยการส่งสัญญาณรีเซ็ตวงจร *Arbiter* หลังสิ้นสุดการรับส่งข้อมูลแต่ละครั้ง
- ควบคุมการทำงานของวงจรปรับค่าพอยน์เตอร์ชี้ตำแหน่งหน่วยความจำ
- กำหนดค่า *Base Address* ของข้อมูล
- กำหนดค่าเริ่มต้นในการนับของวงจรนับภายในวงจรรีจิสเตอร์
- ควบคุมการทำงานของวงจรรีจิสเตอร์ซึ่งทำหน้าที่เก็บค่า *Base Address* และค่าเริ่มต้นในการนับ
- ส่งสัญญาณควบคุมการรับส่งข้อมูลผ่าน MII
- ส่งสัญญาณควบคุมการอ่านค่าจากวงจรบัฟเฟอร์ซึ่งจัดเก็บข้อมูลที่ใช้ในการจัดคิว
- ประมวลผลแพ็คเกตเพื่อใช้ในการจัดคิวแพ็คเกตตามอัลกอริทึมที่กำหนด
- ประมวลผลแพ็คเกตเพื่อเลือกเส้นทางสำหรับส่งต่อแพ็คเกต

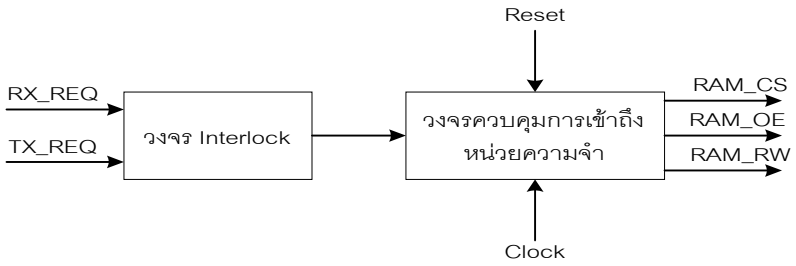


ภาพประกอบ 4-2 สถาปัตยกรรมโดยภาพรวมของเครื่องจัดการจราจรในระบบเครือข่าย

4.2.2 **วงจร** ***Arbiter*** ประกอบด้วยวงจรย่อย ๆ ดังนี้คือ

- วงจรซึ่งทำหน้าที่ควบคุมการเข้าถึงหน่วยความจำของวงจรภาครับและวงจรภาคส่ง โดยทำหน้าที่สร้างสัญญาณ Chip Select สัญญาณควบคุมการอ่านและเขียนหน่วยความจำ และสัญญาณควบคุมการอีเนเบิลเอาท์พุตของหน่วยความจำ
- วงจร Interlock ทำหน้าที่ป้องกันการเข้าถึงหน่วยความจำพร้อมกันของวงจรภาครับและส่ง โดยการตรวจสอบว่าในขณะนั้นมีการเข้าถึงหน่วยความจำเกิดขึ้นหรือไม่ก่อนที่จะอนุญาตให้วงจรที่ร้องขอการเข้าถึงหน่วยความจำสามารถเข้าใช้งานหน่วยความจำได้

การทำงานของวงจร *Arbiter* สามารถอธิบายได้ดังนี้คือ ไมโครโปรเซสเซอร์จะส่งสัญญาณรีเซ็ตวงจรภายใน จากนั้นเมื่อมีการส่งสัญญาณร้องขอการเข้าถึงหน่วยความจำจากวงจรภาครับหรือวงจรภาคส่ง วงจร *Arbiter* จะทำการตรวจสอบว่ามีการเข้าถึงหน่วยความจำหลักอยู่หรือไม่ก่อนจะส่งสัญญาณ *RAM_CS* ไปควบคุมให้หน่วยความจำหลักแอคทีพและส่งสัญญาณอ่านหรือเขียนหน่วยความจำไปควบคุมหน่วยความจำหลักขึ้นอยู่กับว่าวงจรใดส่งสัญญาณร้องขอการเข้าถึงหน่วยความจำ การทำงานของวงจรภายในจะอ้างอิงสัญญาณนาฬิกาที่อัตราเร็วเดียวกับอัตราเร็วในการรับส่งข้อมูลผ่าน MII คือ 25 เมกกะเฮิร์ท บล็อกไดอะแกรมอย่างง่ายของวงจร *Arbiter* แสดงดังภาพประกอบ 4-3



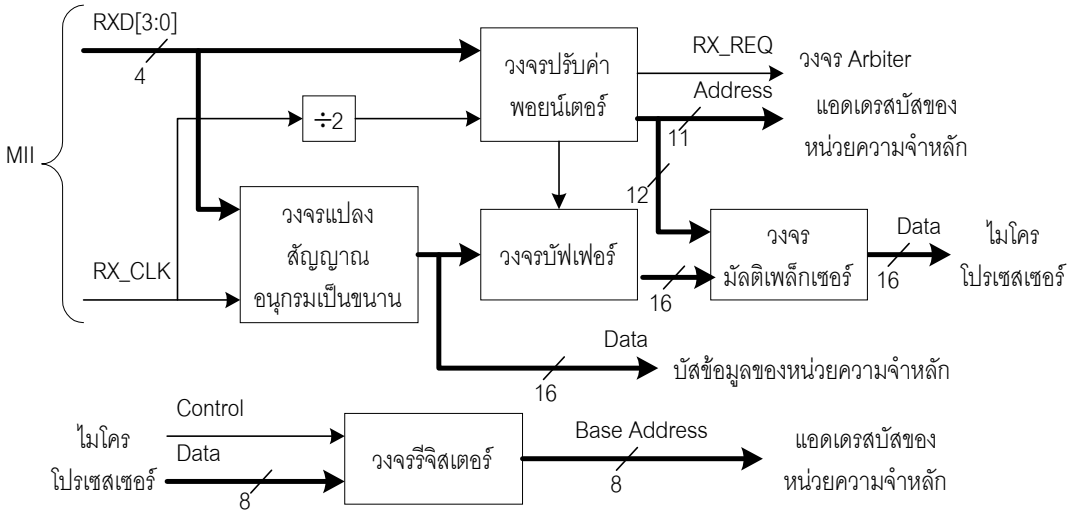
ภาพประกอบ 4-3 บล็อกไดอะแกรมอย่างง่ายของวงจร Arbiter

4.2.3 **วงจรภาครับ** ทำหน้าที่จัดเก็บแพ็คเกตไว้ในหน่วยความจำหลักและจัดเก็บข้อมูลที่ไมโครโปรเซสเซอร์ใช้ในการประมวลผลไว้ภายในหน่วยความจำของไมโครโปรเซสเซอร์ซึ่งประกอบด้วยวงจรย่อย ๆ ดังนี้คือ

- วงจรแปลงสัญญาณอนุกรมเป็นขนาน
- วงจรปรับค่าพอยน์เตอร์ชี้ตำแหน่งหน่วยความจำ

- วงจรบัฟเฟอร์ ทำหน้าที่จัดเก็บข้อมูลที่จำเป็นในการประมวลผล
- วงจรรีจิสเตอร์ ทำหน้าที่เก็บค่า *Base Address*
- วงจรมัลติเพล็กเซอร์

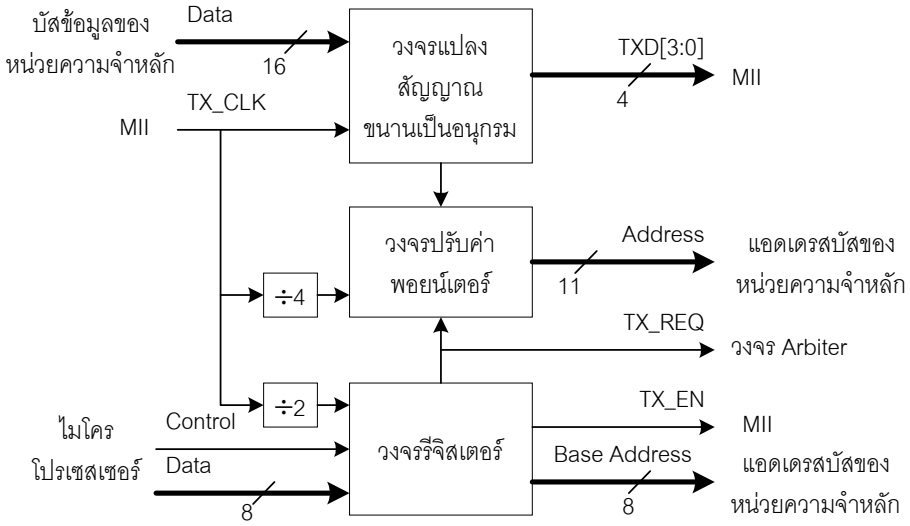
การทำงานของวงจรภาครับสามารถอธิบายได้ดังนี้คือ ไมโครโปรเซสเซอร์จะส่งค่า *Base Address* มาให้วงจรรีจิสเตอร์เพื่อระบุตำแหน่งเริ่มต้นในการจัดเก็บแพ็คเกต โดยวงจรตรวจหา SFD ซึ่งอยู่ภายในวงจรปรับค่าพอยน์เตอร์จะทำการตรวจหา SFD จากข้อมูลที่ได้รับเข้ามาเพื่อหาจุดเริ่มต้นของแพ็คเกตและทำการแปลงข้อมูลจากอนุกรมเป็นขนาน หลังจากพบ SFD วงจรปรับค่าพอยน์เตอร์จะทำการปรับค่าพอยน์เตอร์ชี้ตำแหน่งหน่วยความจำ วงจรนับภายในวงจรปรับค่าพอยน์เตอร์จะทำการเพิ่มค่าทุก ๆ สัญญาณนาฬิกา RX_CLK 2 ลูกหรือทุก ๆ การรับข้อมูล 1 ไบต์ ซึ่งวงจรนับมีขนาด 12 บิตคือ A[11:0] โดย A[11:1] คือค่าแอดเดรสระบุตำแหน่งจัดเก็บแพ็คเกตถูกส่งไปยังแอดเดรสบัสของหน่วยความจำหลัก ส่วน A0 ใช้ระบุจำนวนไบต์ที่จัดเก็บในตำแหน่งสุดท้าย การทำงานของวงจรแปลงสัญญาณอนุกรมเป็นขนานจะอ้างอิงสัญญาณนาฬิกา RX_CLK โดยพอยน์เตอร์ชี้ตำแหน่งของหน่วยความจำสำหรับจัดเก็บข้อมูลจะถูกปรับค่าไปเรื่อย ๆ จนกว่าจะจัดเก็บข้อมูลจนครบ ในขณะจัดเก็บข้อมูลอยู่นั้นวงจรปรับค่าพอยน์เตอร์จะส่งสัญญาณพัลส์ไปควบคุมให้วงจรบัฟเฟอร์ทำการเก็บข้อมูลที่ไมโครโปรเซสเซอร์ใช้ในการประมวลผล โดยมีวงจรมัลติเพล็กเซอร์ทำหน้าที่เลือกเอาท์พุตจากวงจรบัฟเฟอร์หรือค่าจำนวนไบต์ของแพ็คเกตจากวงจรปรับค่าพอยน์เตอร์ส่งไปยังไมโครโปรเซสเซอร์ บล็อกไดอะแกรมอย่างง่ายของวงจรภาครับแสดงดังภาพประกอบ 4-4



ภาพประกอบ 4-4 บล็อกไดอะแกรมอย่างง่ายของวงจรภาครับ

4.2.4 วงจรภาคส่ง ทำหน้าที่สร้างสัญญาณ Preamble และ SFD และส่งแพ็คเกต โดยประกอบด้วยวงจรย่อย ๆ ดังนี้คือ

- วงจรสร้างสัญญาณ Preamble และ SFD วงจรในส่วนนี้ถูกรวมอยู่ภายในวงจรแปลงสัญญาณขนานเป็นอนุกรม
- วงจรแปลงสัญญาณขนานเป็นอนุกรม
- วงจรปรับค่าพอยน์เตอร์
- วงจรรีจิสเตอร์ ทำหน้าที่เก็บค่า *Base Address* และค่าเริ่มต้นในการนับซึ่งส่งมาจากไมโครโปรเซสเซอร์และนับจำนวนไบต์ของข้อมูลที่ส่งออก

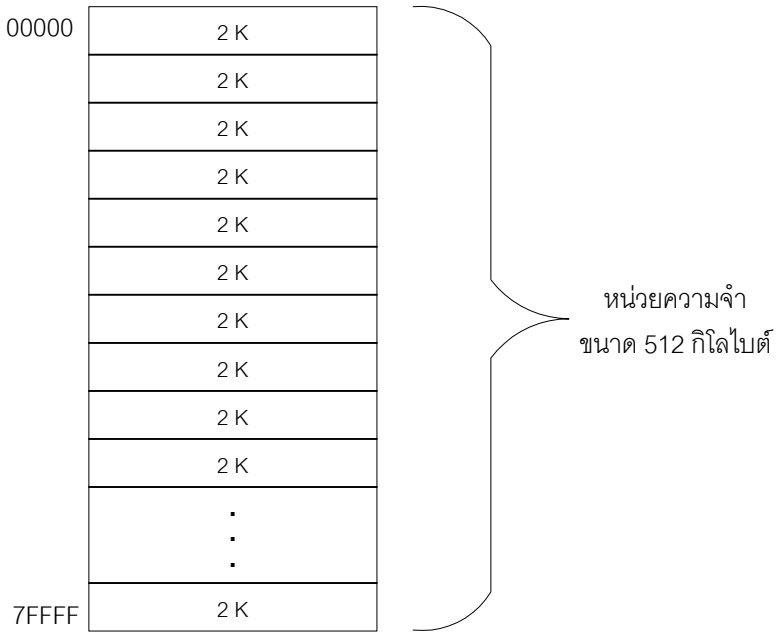


ภาพประกอบ 4-5 บล็อกไดอะแกรมอย่างง่ายของวงจรภาคส่ง

การทำงานของวงจรภาคส่งสามารถอธิบายได้ดังนี้คือ ไมโครโปรเซสเซอร์จะส่งค่า *Base Address* มาให้วงจรรีจิสเตอร์เพื่อระบุตำแหน่งเริ่มต้นของแพ็คเกตที่ต้องการส่ง จากนั้นวงจรภาคส่งจะสร้างสัญญาณ Preamble และ SFD ตามลำดับแล้วจึงส่งแพ็คเกตที่ต้องการส่งออก โดยแพ็คเกตดังกล่าวจะถูกแปลงสัญญาณจากสัญญาณขนาน 16 บิตซึ่งอ่านจากหน่วยความจำหลักไปเป็นสัญญาณอนุกรม 4 บิตก่อนจะส่งผ่าน MII และเมื่อส่งสัญญาณ Preamble และ SFD ครบแล้ว วงจรแปลงสัญญาณขนานเป็นอนุกรมจะส่งสัญญาณมารีเซ็ตวงจรปรับค่าพอยน์เตอร์ ซึ่งวงจรปรับค่าพอยน์เตอร์จะปรับค่าขึ้นเมื่อสัญญาณนาฬิกา TX_CLK เกิดขึ้น 4 ลูกหรือทุก ๆ การส่งข้อมูล 2 ไบต์ และวงจรรีจิสเตอร์จะส่งสัญญาณมาควบคุมพอยน์เตอร์ให้เริ่มหรือหยุดทำงาน โดย

ไมโครโปรเซสเซอร์จะส่งสัญญาณมาควบคุมการโหลดค่าเริ่มต้นในการนับเก็บไว้ในวงจรรีจิสเตอร์ ในระหว่างส่งข้อมูลวงจรนับภายในวงจรรีจิสเตอร์จะทำการนับจำนวนไบต์ที่ส่งออก โดยจะเพิ่มค่าขึ้นทุก ๆ การส่งข้อมูลแต่ละไบต์ เมื่อทำการส่งข้อมูลจนครบแล้ววงจรรีจิสเตอร์จะส่งสัญญาณไปบอกวงจร *Arbiter* และวงจรปรับค่าพอยน์เตอร์เพื่อบอกการสิ้นสุดการส่งข้อมูล การทำงานของวงจรแปลงสัญญาณขนานเป็นอนุกรมจะทำงานโดยอ้างอิงสัญญาณนาฬิกา TX_CLK บล็อกไดอะแกรมอย่างง่ายของวงจรภาคส่งแสดงดังภาพประกอบ 4-5

4.2.5 **หน่วยความจำหลัก** เป็นที่เก็บแพ็คเกตหลังจากได้รับแพ็คเกตเข้ามาและในระหว่างที่ไมโครโปรเซสเซอร์ประมวลผลเพื่อจัดคิวแพ็คเกต ทั้งวงจรภาครับและวงจรภาคส่งจะใช้งานหน่วยความจำหลักนี้ร่วมกันโดยมีวงจร *Arbiter* ควบคุมการเข้าถึงหน่วยความจำ ซึ่งหน่วยความจำที่ใช้มีขนาดเท่ากับ 512 กิโลไบต์ และจากการประมาณขนาดสูงสุดของแพ็คเกตเท่ากับ 2 กิโลไบต์ ดังนั้นหน่วยความจำที่เลือกใช้จะสามารถเก็บแพ็คเกตได้สูงสุด 256 แพ็คเกต การจัดแบ่งเนื้อที่สำหรับจัดเก็บแพ็คเกตในลักษณะนี้เพื่อลดความซับซ้อนของวงจรที่ออกแบบและง่ายต่อการออกแบบวงจร Memory Map ของหน่วยความจำหลักแสดงดังภาพประกอบ 4-6 เนื่องจากขนาดของบัสข้อมูลของหน่วยความจำหลักเท่ากับ 16 บิต ดังนั้นการปรับค่าพอยน์เตอร์ชี้ตำแหน่งหน่วยความจำแต่ละค่าเท่ากับเป็นการเลื่อนตำแหน่งการชี้ตำแหน่งข้อมูลทีละ 2 ไบต์



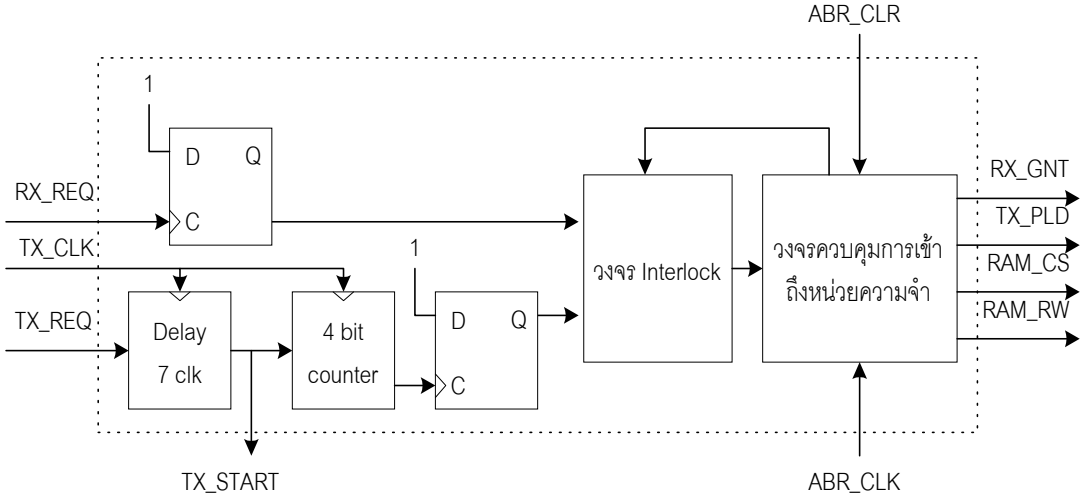
ภาพประกอบ 4-6 Memory Map ของหน่วยความจำหลัก

4.2.6 ชิพ DP83840A และ DP83223 เป็นอุปกรณ์ใน Physical Layer โดยชิพ DP83840A ทำหน้าที่เชื่อมต่อ Physical Signaling Layer เข้ากับ Medium Access Control Layer ผ่านทาง MII การเชื่อมต่อกับ Physical Signaling Layer นั้นทำโดยใช้ชิพ DP83223 ซึ่งเป็นอุปกรณ์ที่เรียกว่า 100 Mb/s Physical Medium Dependent (PMD) Transceiver รายละเอียดการทำงานของชิพทั้งสองกล่าวไว้ในบทที่ 3

4.3 การออกแบบวงจร

การออกแบบวงจรที่จะกล่าวถึงในหัวข้อนี้จะเน้นการออกแบบวงจรที่อยู่ภายในชิพเอฟพีจีเอ หลังจากเลือกใช้ชิฟเอฟพีจีเอของบริษัท Xilinx เข้ามาช่วยในการจัดคิวแพ็คเกต โปรแกรมที่ใช้ในการออกแบบวงจรคือ Xilinx Foundation Series 2.1i ซึ่งวิธีการออกแบบมีด้วยกัน 3 วิธีคือ HDL Editor, FSM Editor และ Schematic Editor สำหรับในวิทยานิพนธ์นี้เลือกใช้วิธีการออกแบบด้วย Schematic Editor โดยวงจรที่ออกแบบได้แก่ วงจร *Arbiter* วงจรภาครับและวงจรภาคส่ง ซึ่งสามารถแยกอธิบายแต่ละวงจรได้ดังนี้คือ

4.3.1 วงจร *Arbiter* บล็อกไดอะแกรมของวงจร *Arbiter* แสดงดังภาพประกอบ 4-7 ส่วนรายละเอียดของวงจรที่ได้ทำการออกแบบจะอยู่ในภาคผนวก วงจร *Arbiter* มีสัญญาณอินพุต 5 สัญญาณ คือ *TX_REQ*, *TX_CLK*, *ABR_CLR*, *ABR_CLK* และ *RX_REQ* และสัญญาณเอาท์พุต 5 สัญญาณคือ *TX_START*, *TX_PLD*, *RAM_CS*, *RAM_RW* และ *RX_GNT* ซึ่งรายละเอียดของสัญญาณเหล่านี้สามารถอธิบายได้ดังนี้คือ



ภาพประกอบ 4-7 บล็อกไดอะแกรมของวงจร *Arbiter*

- *TX_CLK* และ *ABR_CLK* เป็นสัญญาณนาฬิกาที่ใช้อ้างอิงการทำงานของฟลิปฟลอปภายในวงจร *Arbiter*
- *ABR_CLR* เป็นสัญญาณซึ่งไมโครโปรเซสเซอร์ส่งมาเพื่อรีเซ็ตฟลิปฟลอปภายใน โดยจะรีเซ็ตเมื่อสัญญาณ *ABR_CLR* มีสถานะเป็นลอจิก High หรือเป็นลอจิก 1
- *RX_REQ* เป็นสัญญาณที่วงจรภาครับส่งมาเพื่อร้องขอการเข้าถึงหน่วยความจำหลัก
- *TX_REQ* เป็นสัญญาณที่วงจรภาคส่งส่งมาเพื่อร้องขอการเข้าถึงหน่วยความจำหลัก
- *RAM_RW* เป็นสัญญาณเอาท์พุตที่ใช้ในการควบคุมการอ่านและเขียนหน่วยความจำหลัก และควบคุมขาอีเนเบิล (Enable) ของ Tri-State Buffer ซึ่งเป็นขาที่ใช้ควบคุมสถานะอินพุตหรือเอาท์พุตของบัสข้อมูลของหน่วยความจำ
- *RAM_CS* เป็นสัญญาณเอาท์พุตที่ส่งมาเพื่อทำให้หน่วยความจำหลักแอคทีพ
- *TX_START* คือสัญญาณ *TX_GNT* ที่วงจร *Arbiter* ส่งไปควบคุมวงจรภาคส่งเพื่อบอกการเริ่มต้นการส่งข้อมูล
- *TX_PLD* เป็นสัญญาณเอาท์พุตที่ส่งออกไปควบคุมการอ่านข้อมูลจากบัสข้อมูลมาเก็บไว้ภายในฟลิปฟลอปของวงจรภาคส่งและควบคุมขาเอาท์พุตอีเนเบิลของหน่วยความจำ
- *RX_GNT* เป็นสัญญาณเอาท์พุตที่ส่งออกไปควบคุมการเลือกเอาท์พุตของวงจรมัลติเพล็กเซอร์ที่อยู่ภายในวงจร *ADR_CNT* และวงจร *CPU_REGS*

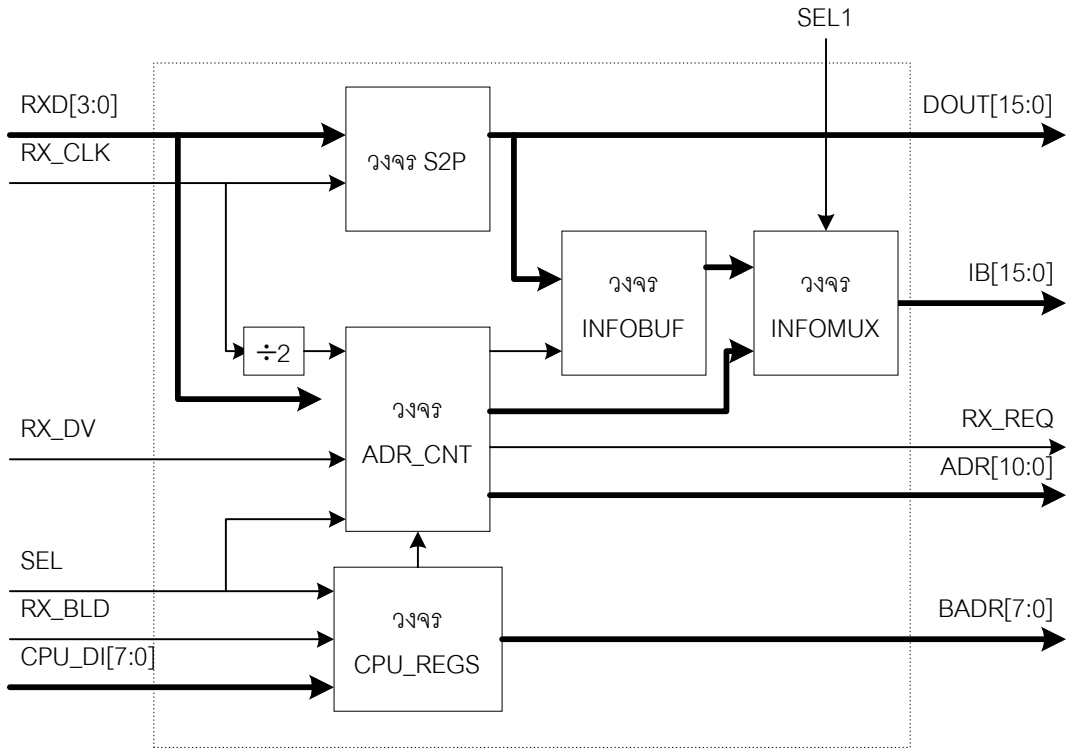
เมื่อวงจรภาครับส่งสัญญาณ *RX_REQ* ซึ่งมีการเปลี่ยนสถานะจากลอจิก Low หรือ 0 เป็นลอจิก High หรือ 1 และตรวจสอบพบว่าในขณะนั้นวงจรภาคส่งยังไม่ได้เข้าถึงหน่วยความจำหลัก หลังจากสัญญาณนาฬิกาขาขึ้นของ *ABR_CLK* ผ่านไป 1 ลูก สัญญาณ *RAM_CS* จะเปลี่ยนสถานะเป็นลอจิก 0 ทำให้หน่วยความจำหลักแอคทีพ สัญญาณ *RAM_RW* จะเปลี่ยนเป็นลอจิก 0 ซึ่งหมายถึง การเขียนข้อมูลลงในหน่วยความจำ และสัญญาณ *RX_GNT* จะเปลี่ยนสถานะจากลอจิก 0 เป็นลอจิก 1 วงจรภาครับจะสามารถเขียนข้อมูลลงในหน่วยความจำได้ ในขณะที่วงจรภาครับเขียนข้อมูลในหน่วยความจำ แม้จะมีสัญญาณร้องขอการใช้หน่วยความจำจากวงจรภาคส่งเข้ามา สัญญาณดังกล่าวจะถูกบล็อกโดยแอนด์เกต (AND Gate) จนกว่าวงจรภาครับจะเขียนข้อมูลเสร็จสิ้น จากนั้นวงจรภาคส่งจึงจะสามารถอ่านข้อมูลจากหน่วยความจำได้ และเมื่อวงจรภาคส่งต้องการอ่านข้อมูลจากหน่วยความจำนั้นจะมีการนับสัญญาณนาฬิกา 7 ลูกหรือ 280 นาโนวินาทีเป็นการหน่วงเวลาไว้สำหรับ Interframe ระหว่างแพ็คเกต โดยจะตรวจสอบว่าในขณะนั้นวงจรภาครับเข้าถึงหน่วยความจำอยู่หรือไม่ ก่อนจะสร้างสัญญาณ *TX_START* ส่งไปยังวงจร

ภาคส่งเพื่อบอกให้เริ่มต้นการส่งข้อมูล หลังจากนั้นจึงส่งสัญญาณ *TX_PLD* ให้วงจรภาคส่งทำการอ่านข้อมูลจากบัสข้อมูลและแปลงสัญญาณขนานเป็นอนุกรมต่อไป ในการออกแบบนั้นน็อตเกต (NOT Gate) ถูกนำมาใช้ด้วยเหตุผลในเรื่องของ Interlock และเป็นการเผื่อเวลาสำหรับการเข้าถึงหน่วยความจำด้วย

4.3.2 **วงจรภาครับ** บล็อกไดอะแกรมของวงจรภาครับแสดงดังภาพประกอบ 4-8 ส่วนรายละเอียดของวงจรที่ได้ทำการออกแบบจะอยู่ในภาคผนวก วงจรภาครับมีสัญญาณอินพุต 17 สัญญาณ คือ *RXD[3:0]*, *RX_CLK*, *RX_DV*, *RX_BLD*, *CPU_DI[7:0]*, *SEL* และ *SEL1* และมีสัญญาณเอาท์พุต 52 สัญญาณคือ *DOUT[15:0]*, *RX_REQ*, *ADR[10:0]*, *BADR[7:0]* และ *IB [15:0]* ซึ่งรายละเอียดของสัญญาณเหล่านี้สามารถอธิบายได้ดังนี้คือ

- *RXD[3:0]* และ *RX_CLK* เป็นสัญญาณข้อมูลและสัญญาณนาฬิกาที่ได้รับผ่านทาง MII
- *RX_DV* เป็นสัญญาณที่ได้รับผ่าน MII เพื่อบอกให้ทราบว่ามีการส่งข้อมูลอยู่ในสัญญาณ *RXD[3:0]* ถูกใช้เป็นสัญญาณ Chip Select สำหรับเลือกให้วงจร *ADR_CNT* ทำงาน
- *SEL* เป็นสัญญาณควบคุมมัลติเพล็กเซอร์ของวงจร *ADR_CNT* และวงจร *CPU_REGS* ให้เลือกเอาท์พุตสำหรับวงจรภาครับซึ่งควบคุมโดยสัญญาณ *RX_GNT* จากวงจร *Arbiter*
- *SEL1* เป็นสัญญาณควบคุมมัลติเพล็กเซอร์ภายในวงจร *INFOMUX* เพื่อเลือกสัญญาณเอาท์พุตจากวงจร *INFOBUF* หรือวงจร *ADR_CNT*
- *CPU_DI[7:0]* เป็นสัญญาณที่ไมโครโปรเซสเซอร์ส่งมาเพื่อระบุค่า *Base Address*
- *RX_BLD* เป็นสัญญาณที่ไมโครโปรเซสเซอร์ส่งมาเพื่อควบคุมการค้างค่า *Base Address* ของฟลิปฟลอปภายใน
- *ADR[10:0]* และ *BADR[7:0]* เป็นสัญญาณแอดเดรสที่วงจรภาครับส่งไปให้หน่วยความจำหลักเพื่อระบุตำแหน่งที่จะจัดเก็บแพ็คเกต
- *DOUT[15:0]* เป็นสัญญาณข้อมูลที่วงจรภาครับส่งให้หน่วยความจำหลักเพื่อจัดเก็บแพ็คเกต
- *RX_REQ* เป็นสัญญาณที่วงจรภาครับส่งไปให้วงจร *Arbiter* เพื่อร้องขอการเข้าถึงหน่วยความจำหลัก

- *IB[15:0]* เป็นสัญญาณข้อมูลที่ใช้ในการประมวลผลเพื่อจัดคิวแพ็คเกตซึ่งส่งมาจากวงจร *INFOBUF* หรือจำนวนไบต์ของแพ็คเกตซึ่งส่งมาจากวงจร *ADR_CNT* ขึ้นอยู่กับสถานะของสัญญาณ *SEL1* ที่ใช้ควบคุมมัลติเพล็กเซอร์



ภาพประกอบ 4-8 บล็อกไดอะแกรมของวงจรภาครับ

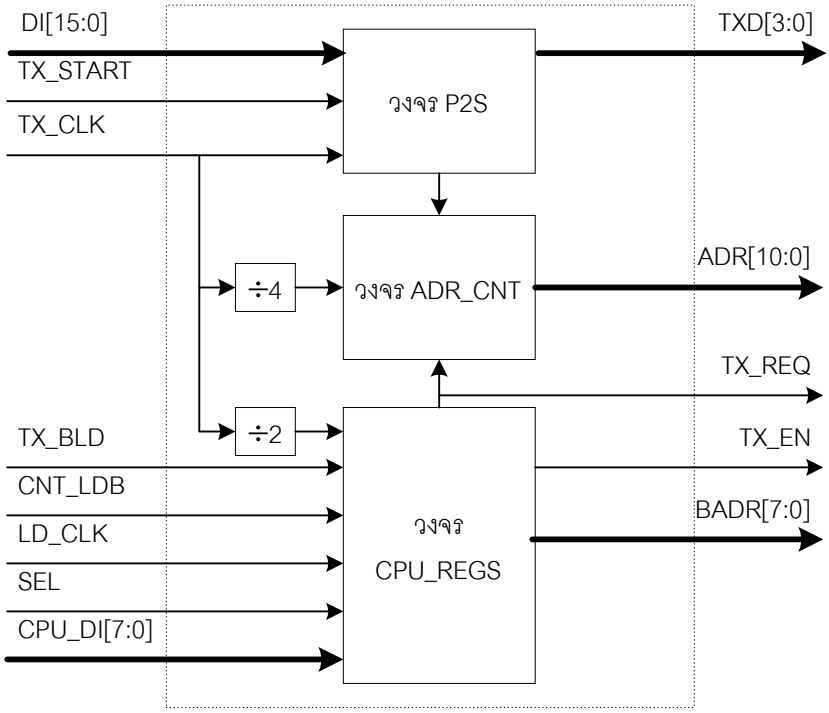
ในการรับข้อมูลไมโครโปรเซสเซอร์ทำการส่งค่า *Base Address* จำนวน 8 บิตของแพ็คเกตไปให้วงจร *CPU_REGS* เพื่อระบุตำแหน่งเริ่มต้นในการจัดเก็บแพ็คเกตและใช้สัญญาณ *RX_BLD* ในการค้างค่า *Base Address* ดังกล่าวไว้ เมื่อมีข้อมูลส่งเข้ามา วงจร *S2P* จะทำการแปลงสัญญาณข้อมูลจากอนุกรมเป็นขนาน หลังจากวงจร *SFD_Det* ซึ่งอยู่ภายในวงจร *ADR_CNT* ตรวจพบ SFD แล้ววงจร *ADR_CNT* ทำการปรับค่าพอยน์เตอร์ที่ชี้ไปยังหน่วยความจำหลักเพื่อจัดเก็บแพ็คเกตไว้ในหน่วยความจำผ่านทางบัสข้อมูลขนาด 16 บิต โดยพอยน์เตอร์ชี้ตำแหน่งของหน่วยความจำจะถูกปรับค่าไปเรื่อย ๆ จนกว่าจะจัดเก็บข้อมูลจนครบ และในขณะที่กำลังจัดเก็บข้อมูลลงในหน่วยความจำอยู่นั้นวงจร *ADR_CNT* จะส่งสัญญาณพัลส์ไปให้วงจร *INFOBUF* ทำการจัดเก็บข้อมูลที่ไมโครโปรเซสเซอร์ใช้ในการประมวลผลไว้ภายในหน่วยความจำของไมโครโปรเซสเซอร์และส่งสัญญาณไปควบคุมการอ่านข้อมูลจากบัสข้อมูลของวงจรแปลง

สัญญาณอนุกรมเป็นขนาน วงจร *ADR_CNT* สามารถชี้ตำแหน่งหน่วยความจำได้สูงสุด 2^{11} หรือ 2 กิโลไบต์ สำหรับวงจร *INFOMUX* เป็นวงจรมัลติเพล็กเซอร์ซึ่งใช้สัญญาณ *SEL1* เพื่อเลือกสัญญาณข้อมูลขนาด 16 บิตที่ไมโครโปรเซสเซอร์ใช้สำหรับจัดคิวจากวงจร *INFOBUF* หรือจำนวนไบต์ของแพ็คเกตขนาด 12 บิตจากวงจร *ADR_CNT*

4.3.3 **วงจรภาคส่ง** บล็อกไดอะแกรมของวงจรภาคส่งแสดงดังภาพประกอบ 4-9 ส่วนรายละเอียดของวงจรที่ได้ทำการออกแบบจะอยู่ในภาคผนวก วงจรภาคส่งมีสัญญาณอินพุต 29 สัญญาณ คือ *DI[15:0]*, *TX_CLK*, *TX_BLD*, *SEL*, *CNT_LDB*, *LD_CLK* และ *CPU_DI[7:0]* และสัญญาณเอาท์พุต 25 สัญญาณคือ *TXD[3:0]*, *TX_REQ*, *TX_EN*, *ADR[10:0]* และ *BADR[7:0]* รายละเอียดของสัญญาณเหล่านี้สามารถอธิบายได้ดังนี้คือ

- *TXD[3:0]* เป็นสัญญาณข้อมูลที่ส่งผ่านทาง MII
- *TX_CLK* เป็นสัญญาณนาฬิกาที่ได้ผ่านทาง MII
- *SEL* เป็นสัญญาณควบคุมมัลติเพล็กเซอร์ของวงจร *ADR_CNT* และวงจร *CPU_REGS* ให้เลือกเอาท์พุตสำหรับวงจรภาคส่งซึ่งควบคุมโดยสัญญาณ *RX_GNT* จากวงจร *Arbiter*
- *CPU_DI[7:0]* เป็นสัญญาณที่ไมโครโปรเซสเซอร์ส่งมาเพื่อระบุค่า *Base Address* และค่าเริ่มต้นในการนับ
- *TX_BLD* เป็นสัญญาณที่ไมโครโปรเซสเซอร์ส่งมาเพื่อควบคุมการค้างค่า *Base Address* ของฟลิปฟลอปภายใน
- *ADR[10:0]* และ *BADR[7:0]* เป็นสัญญาณแอดเดรส 11 บิตล่างและ 8 บิตบนที่วงจรภาคส่งส่งไปให้หน่วยความจำหลักเพื่อระบุตำแหน่งที่จัดเก็บแพ็คเกต
- *DI[15:0]* เป็นสัญญาณข้อมูลที่วงจรภาคส่งอ่านจากหน่วยความจำหลักเพื่อส่งออกต่อไป
- *TX_EN* เป็นสัญญาณที่วงจรภาคส่งส่งไปผ่านทาง MII เพื่อบอกให้ทราบว่ามีการส่งข้อมูลเกิดขึ้นในขณะนั้นหรือไม่
- *TX_REQ* เป็นสัญญาณเดียวกับสัญญาณ *TX_ENDB* ซึ่งเป็นสัญญาณที่ส่งไปยังวงจร *Arbiter* เพื่อร้องขอการเข้าถึงหน่วยความจำและส่งไปยังวงจร *ADR_CNT* เพื่อบอกการสิ้นสุดการส่งข้อมูล
- *TX_START* เป็นสัญญาณบอกการเริ่มต้นการส่งข้อมูลซึ่งส่งมาจากวงจร *Arbiter*

- *CNT_LDB* เป็นสัญญาณควบคุมการโหลดค่าเริ่มต้นในการนับของวงจร *CPU_REGS*
- *LD_CLK* เป็นสัญญาณพัลส์เพื่อทำการโหลดค่าเริ่มต้นในการนับ



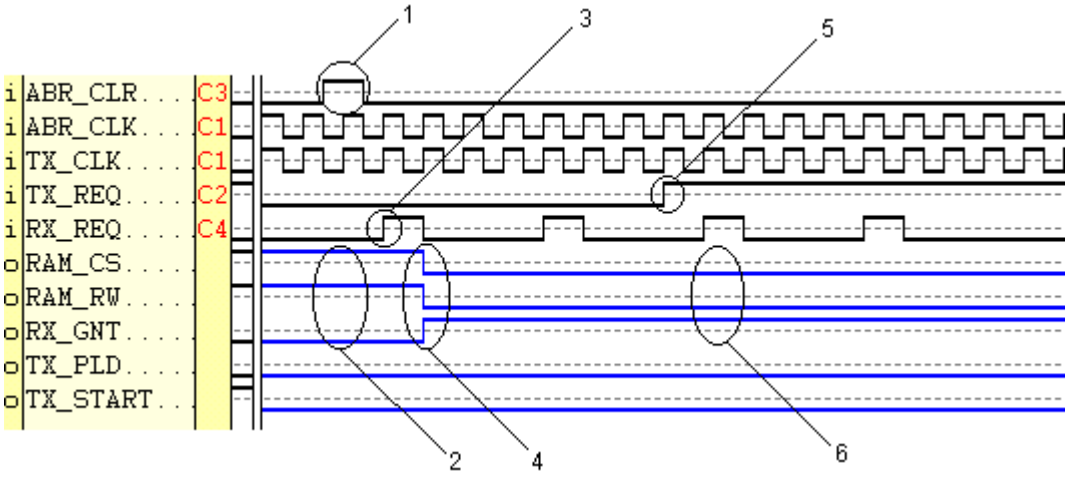
ภาพประกอบ 4-9 บล็อกไดอะแกรมของวงจรภาคส่ง

เมื่อได้รับสัญญาณ *TX_START* จากวงจร *Arbiter* บอกการเริ่มต้นการส่งข้อมูล ไมโครโปรเซสเซอร์จะส่งค่า *Base Address* มาให้วงจร *CPU_REGS* จากนั้นไมโครโปรเซสเซอร์จะส่งสัญญาณ *TX_BLD* เพื่อโหลดค่า *Base Address* แล้วจึงส่งสัญญาณ *CNT_LDB* เพื่อควบคุมการโหลดค่าเริ่มต้นในการนับของวงจร *CPU_REGS* และส่งสัญญาณพัลส์ *LD_CLK* เพื่อทำการโหลดค่าเริ่มต้นในการนับ วงจรภาคส่งจะทำการสร้างสัญญาณ Preamble และ SFD ตามลำดับ เมื่อส่งสัญญาณ Preamble และ SFD ครบแล้วข้อมูลจะถูกอ่านจากหน่วยความจำและแปลงสัญญาณจากขนานไปเป็นอนุกรมแล้วส่งผ่านทาง MII โดยวงจร *ADR_CNT* จะทำการปรับค่าชี้ตำแหน่งหน่วยความจำ ขณะเดียวกันวงจร *CPU_REGS* จะทำการนับจำนวนไบต์ที่ส่งออกไป เมื่อข้อมูลถูกส่งออกไปจนครบ วงจร *CPU_REGS* จะส่งสัญญาณ *TX_EN* ไปยัง MII และเปลี่ยนสถานะของสัญญาณ *TX_REQ* ซึ่งส่งไปยังวงจร *Arbiter* เพื่อบอกการสิ้นสุดการส่งข้อมูล

4.4 ผลการจำลองการทำงานของวงจรที่ได้ออกแบบ

ผลการจำลองการทำงานของวงจรที่นำเสนอในบทนี้จะเป็นผลการจำลองการทำงานของวงจรหลัก ๆ ได้แก่ วงจร *Arbiter* วงจรภาครับและวงจรภาคส่ง ส่วนผลการจำลองของวงจรย่อย ๆ ซึ่งอยู่ภายในวงจรภาครับและวงจรภาคส่งนั้นจะนำเสนอในภาคผนวก ข

4.4.1 ผลการจำลองการทำงานของวงจร *Arbiter* ผลการจำลองการทำงานของวงจร *Arbiter* แสดงดังภาพประกอบ 4-10 และ 4-11 โดยสามารถอธิบายลำดับเหตุการณ์ในการจำลองผลการทำงานของวงจรที่ออกแบบได้ดังนี้คือ

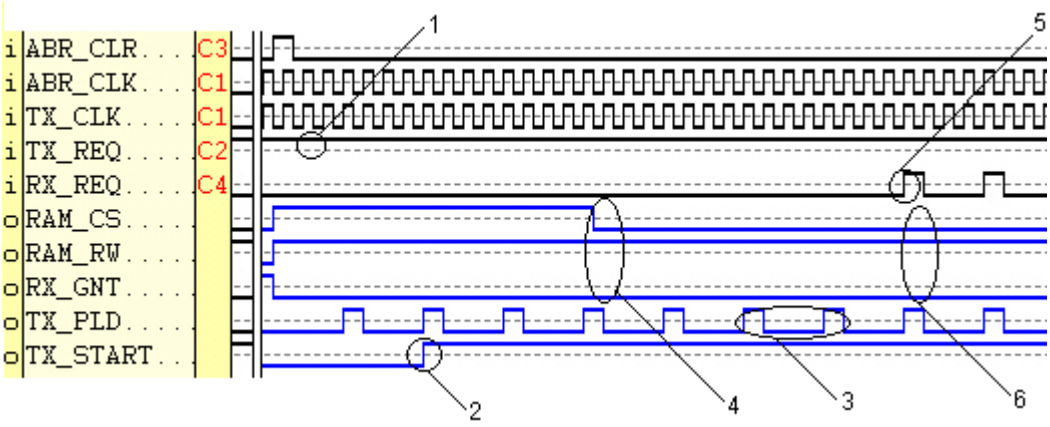


ภาพประกอบ 4-10 ผลการจำลองการทำงานของวงจร *Arbiter*

1. การรีเซ็ตวงจร Arbiter โดยการเปลี่ยนสถานะของสัญญาณ *ABR_CLR* เป็นลอจิก 1
2. หลังการรีเซ็ตวงจร สัญญาณ *RAM_CS* และ *RAM_RW* มีสถานะเป็นลอจิก 1 และสัญญาณ RX_GNT มีสถานะเป็นลอจิก 0 ซึ่งเมื่อสัญญาณ *RAM_CS* มีสถานะเป็นลอจิก 1 นั้น หน่วยความจำจะไม่แอคทีพทำให้ไม่สามารถเข้าถึงหน่วยความจำได้ในขณะนั้น
3. วงจรภาครับส่งสัญญาณร้องขอการใช้หน่วยความจำโดยการเปลี่ยนสถานะของสัญญาณ *RX_REQ* จากลอจิก 0 เป็นลอจิก 1
4. หลังจาก *RX_REQ* เปลี่ยนสถานะจาก 0 เป็น 1 เมื่อสัญญาณนาฬิกาผ่านไป 1 ลูก สัญญาณ *RAM_CS* และ *RAM_RW* มีสถานะเป็นลอจิก 0 และสัญญาณ *RX_GNT*

มีสถานะเป็นลอจิก 1 ซึ่งเมื่อสัญญาณ *RAM_CS* มีสถานะเป็นลอจิก 0 นั้น จะสามารถเข้าถึงหน่วยความจำได้ โดย *RAM_RW* มีสถานะเป็นลอจิก 0 หมายถึงการเขียนข้อมูลและควบคุมขาอีเนเบิลของ Tri-State Buffer ซึ่งควบคุมสถานะอินพุตหรือเอาท์พุตของบัสข้อมูลของหน่วยความจำหลักให้สามารถส่งข้อมูลออกไปได้ ส่วนสัญญาณ *RX_GNT* มีสถานะเป็นลอจิก 1 เป็นการส่งสัญญาณ *SEL* ไปควบคุม *CPU_REGS* ให้เลือกข้อมูล *Base Address* ของวงจรภาครับ

5. เมื่อวงจรภาคส่งร้องขอการเข้าถึงหน่วยความจำขณะที่วงจรภาครับกำลังเขียนข้อมูลลงในหน่วยความจำ
6. สัญญาณร้องขอการเข้าถึงหน่วยความจำของวงจรภาคส่งจะถูกบล็อกไว้ทำให้สถานะของสัญญาณ *RAM_RW* ยังคงมีสถานะเป็นลอจิก 0

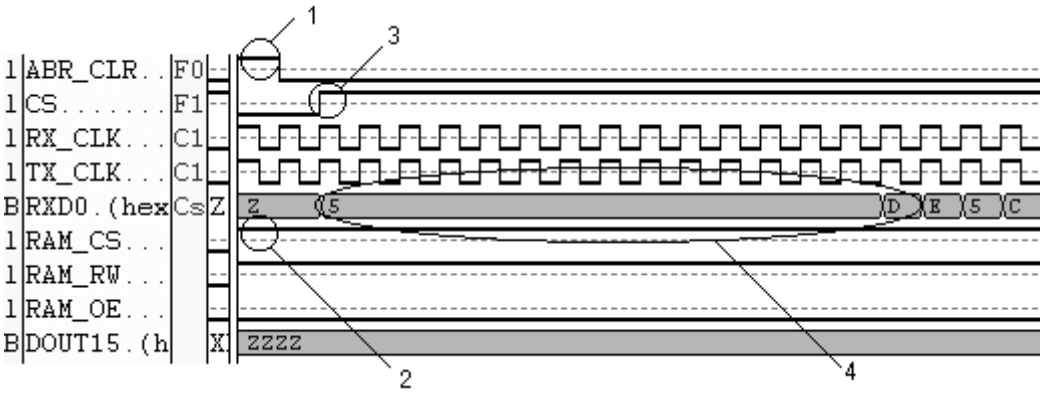


ภาพประกอบ 4-11 ผลการจำลองการทำงานของวงจร *Arbiter* (ต่อ)

1. เมื่อวงจรภาคส่งร้องขอการเข้าถึงหน่วยความจำโดยการเปลี่ยนสถานะของสัญญาณเป็นลอจิก 1
2. หลังจากสัญญาณนาฬิกาผ่านไป 7 ลูก สัญญาณ *TX_START* เปลี่ยนสถานะจากลอจิก 0 เป็นลอจิก 1 ส่งไปยังวงจรภาคส่งเพื่อบอกให้เริ่มต้นการส่งข้อมูล
3. สัญญาณ *TX_PLD* เกิดขึ้นทุก ๆ สัญญาณนาฬิกา 4 ลูกเนื่องจากข้อมูลอนุกรมจะถูกส่งออกทีละ 4 บิต ดังนั้นเมื่อครบสัญญาณนาฬิกา 4 ลูก ข้อมูล 16 บิตจึงถูกแปลงเป็นสัญญาณอนุกรมครบแล้วจึงต้องทำการอ่านข้อมูลจากบัสข้อมูลมาเก็บไว้ในรีจิสเตอร์ใหม่

4. หลังจาก *TX_REQ* มีสถานะเป็นลอจิก 1 และสัญญาณนาฬิกาผ่านไป 15 ลูก เมื่อสัญญาณนาฬิกาเปลี่ยนสถานะจาก 1 เป็น 0 สัญญาณ *RAM_CS* มีสถานะเป็นลอจิก 0 และสัญญาณ *RAM_RW* มีสถานะเป็นลอจิก 1 ส่วนสัญญาณ *RX_GNT* มีสถานะเป็นลอจิก 0 ซึ่งเมื่อสัญญาณ *RAM_CS* มีสถานะเป็นลอจิก 0 นั้น จะสามารถเข้าถึงหน่วยความจำได้ โดย *RAM_RW* มีสถานะเป็นลอจิก 1 หมายถึงการอ่านข้อมูล และควบคุมขาอีเนเบิลของ Tri-State Buffer ซึ่งควบคุมสถานะอินพุตหรือเอาท์พุตของบัสข้อมูลของหน่วยความจำหลักให้สามารถอ่านข้อมูลเข้ามาได้ ส่วนสัญญาณ *RX_GNT* มีสถานะเป็นลอจิก 0 เป็นการส่งสัญญาณ *SEL* ไปควบคุมวงจร *CPU_REGS* ให้เลือกข้อมูล *Base Address* ของวงจรภาคส่ง
5. เมื่อวงจรภาครับร้องขอการเข้าถึงหน่วยความจำขณะที่วงจรภาคส่งกำลังอ่านข้อมูลจากหน่วยความจำ
6. สัญญาณร้องขอการเข้าถึงหน่วยความจำของวงจรภาครับจะถูกบล็อกไว้ ทำให้สถานะของสัญญาณ *RAM_RW* ยังคงมีสถานะเป็นลอจิก 1

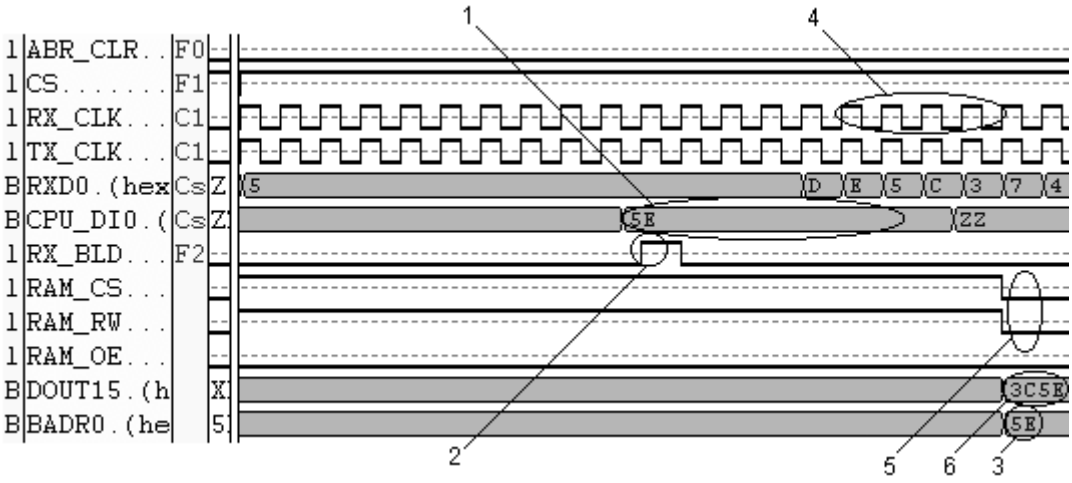
4.4.2 ผลการจำลองการทำงานของวงจรภาครับ ผลการจำลองการทำงานของวงจรภาครับแสดงดังภาพประกอบ 4-12, 4-13 และ 4-14



ภาพประกอบ 4-12 ผลการจำลองการทำงานของวงจรภาครับ

1. สัญญาณ *ABR_CLR* เปลี่ยนสถานะเป็นลอจิก 1 เป็นการรีเซ็ตวงจร *Arbiter*

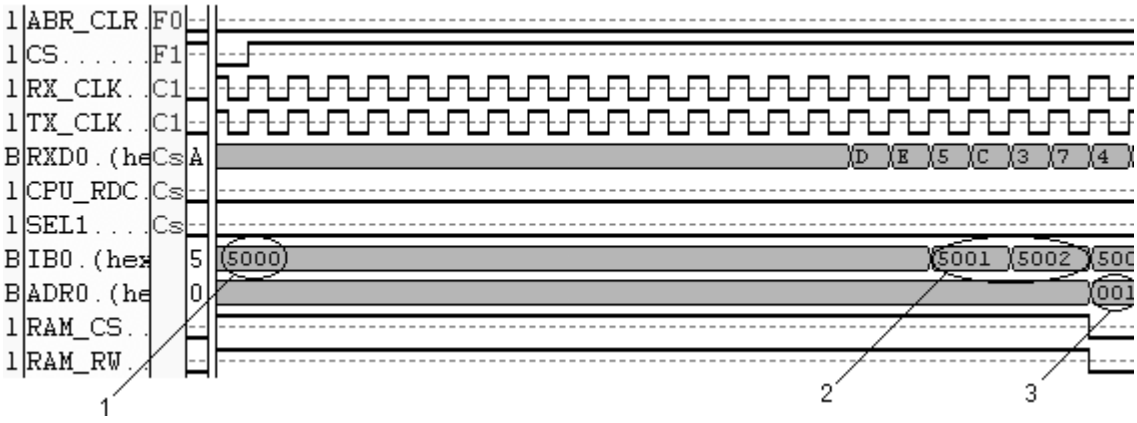
2. หลังการรีเซ็ตวงจร สัญญาณ *RAM_CS* มีสถานะเป็นลอจิก 1 ซึ่งเมื่อสัญญาณ *RAM_CS* มีสถานะเป็นลอจิก 1 หน่วยความจำจะไม่แอคทีพทำให้ไม่สามารถเข้าถึงหน่วยความจำได้ในขณะนั้น
3. สัญญาณ *CS* ซึ่งก็คือสัญญาณ *RX_DV* เปลี่ยนสถานะจากลอจิก 0 เป็น 1
4. สัญญาณ Preamble และ SFD ถูกส่งเข้ามาในสัญญาณ *RXD[3:0]*



ภาพประกอบ 4-13 ผลการจำลองการทำงานของวงจรภาครับ (ต่อ)

1. ไมโครโปรเซสเซอร์ส่งค่า *Base Address* ซึ่งในที่นี้คือค่า 0x5E มายังอินพุตของวงจร *CPU_REGS* คือ *CPU_DI[7:0]*
2. ไมโครโปรเซสเซอร์ส่งสัญญาณ *RX_BLD* มาควบคุมการค้างค่า *Base Address* ไว้ภายในฟลิปฟลอปภายใน โดยจะทำการค้างค่าเมื่อสัญญาณ*RX_BLD* เปลี่ยนสถานะจากลอจิก 0 เป็น 1
3. ค่า *Base Address* ของวงจรภาครับจะไปปรากฏที่เอาท์พุตของวงจร *CPU_REGS*
4. หลังจากพบ SFD วงจร *S2P* จะทำการแปลงข้อมูลจากอนุกรมเป็นขนาน เมื่อสัญญาณนาฬิกา *RX_CLK* ผ่านไป 4 ลูก จะเกิดสัญญาณ *RX_CEO* ลูกแรก ซึ่งจะใช้เป็นสัญญาณ *RX_REQ* ส่งไปยังวงจร Arbiter
5. หลังจากสัญญาณ *RX_REQ* เปลี่ยนสถานะจากลอจิก 0 เป็น 1 เมื่อสัญญาณนาฬิกาผ่านไป 1 ลูก สัญญาณ *RAM_CS* และ *RAM_RW* มีสถานะเป็นลอจิก 0 หมายถึงสามารถเขียนข้อมูลลงในหน่วยความจำได้

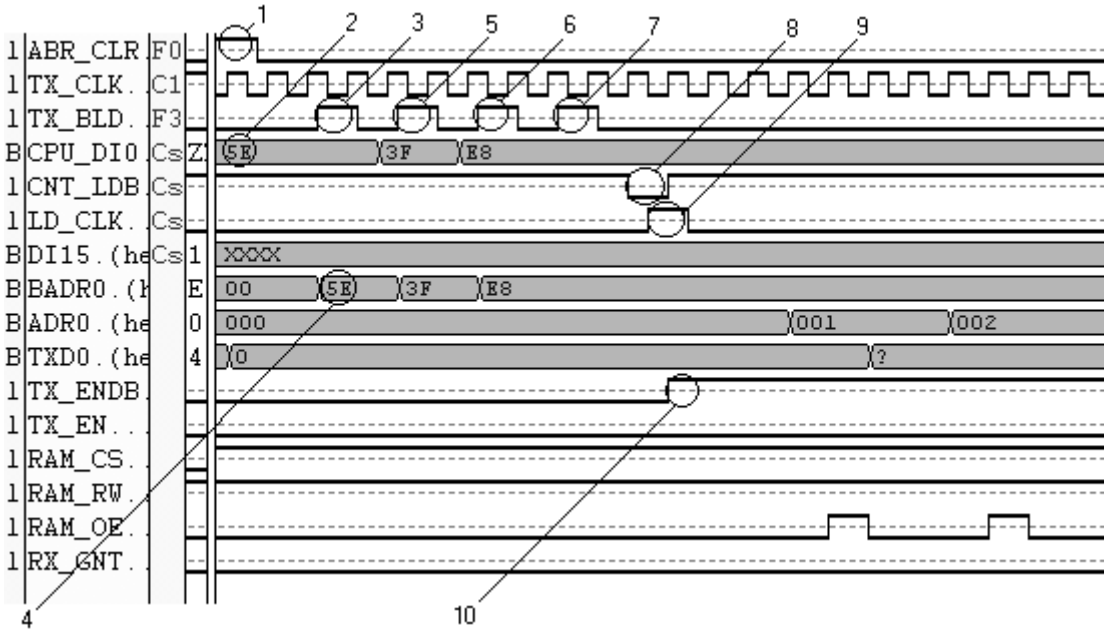
6. ข้อมูลซึ่งผ่านการแปลงจากอนุกรมเป็นขนานซึ่งในที่นี้คือ 0x3C5E ถูกโหลดไปไว้ในบัสข้อมูลของหน่วยความจำ



ภาพประกอบ 4-14 ผลการจำลองการทำงานของวงจรภาครับ (ต่อ)

1. เมื่อสัญญาณ *CS* เปลี่ยนสถานะจากลอจิก 0 เป็น 1 สัญญาณ 12 บิตล่างของสัญญาณ *IB[15:0]* ซึ่งบอกจำนวนไบต์ของแพ็คเกตที่ได้รับมีค่าเท่ากับ 0x0000 และสัญญาณ *ADR[10:0]* ซึ่งก็คือสัญญาณแอดเดรส 11 บิตล่างในการจัดเก็บข้อมูล 16 บิตลงในหน่วยความจำมีค่าเท่ากับ 0x000
2. หลังจากพบ SFD เมื่อสัญญาณนาฬิกาผ่านไป 4 ลูกจะนับจำนวนไบต์ของแพ็คเกตได้เท่ากับ 2 ไบต์
3. แอดเดรสชี้ตำแหน่งหน่วยความจำจะเลื่อนไปยังตำแหน่งถัดไป

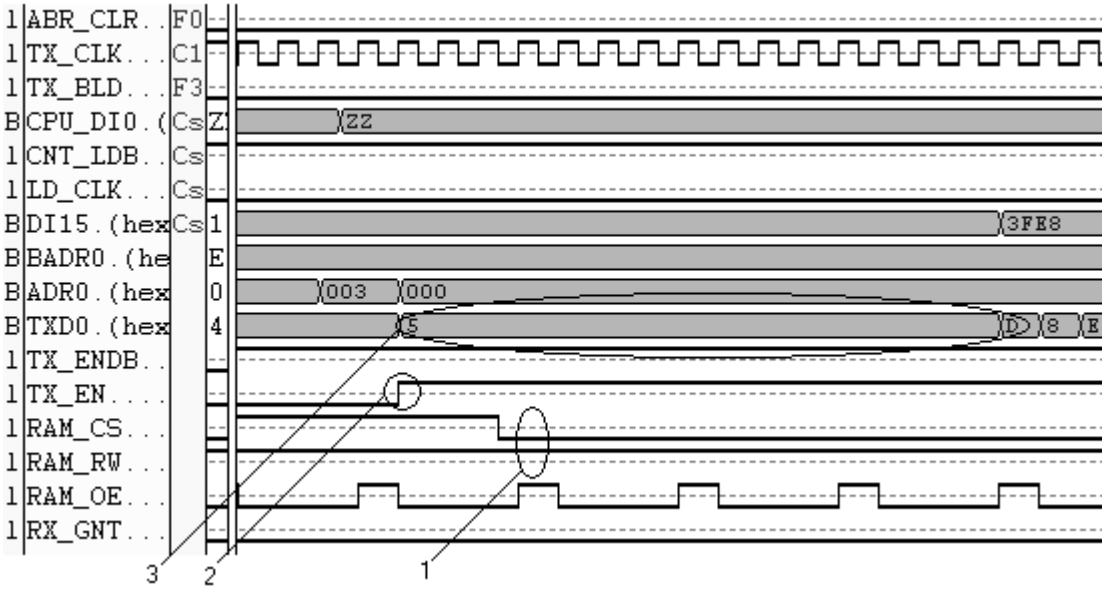
4.4.3 ผลการจำลองการทำงานของวงจรภาคส่ง ผลการจำลองการทำงานของวงจรภาคส่งแสดงดังภาพประกอบ 4-15, 4-16, 4-17 และ 4-18



ภาพประกอบ 4-15 ผลการจำลองการทำงานของวงจรภาคส่ง

1. สัญญาณ *ABR_CLR* เปลี่ยนสถานะเป็นลอจิก 1 เป็นการรีเซ็ตวงจร *Arbiter* และทำให้สัญญาณ *RX_GNT* มีสถานะเป็นลอจิก 0 เป็นการเลือกค่า *Base Address* ของวงจรภาคส่ง
2. ไมโครโปรเซสเซอร์ส่งค่า *Base Address* ซึ่งในที่นี้คือค่า 0x5E มายังอินพุตของวงจร *CPU_REGS* คือ *CPU_DI[7:0]*
3. ไมโครโปรเซสเซอร์ส่งสัญญาณ *TX_BLD* มาควบคุมการค้างค่า *Base Address* ไว้ภายในฟลิปฟลอปภายใน โดยจะทำการค้างค่าเมื่อสัญญาณ*TX_BLD* เปลี่ยนสถานะจากลอจิก 0 เป็น 1
4. ค่า *Base Address* ของวงจรภาครับจะไปปรากฏที่เอาท์พุตของวงจร *CPU_REGS*
5. สัญญาณพัลส์ *TX_BLD* เป็นสัญญาณที่ไมโครโปรเซสเซอร์ส่งมาเพื่อใช้ควบคุมการค้างค่าเริ่มต้นในการนับไว้ในฟลิปฟลอปภายใน โดยพัลส์แรกจะทำการค้างค่า 1 ไบต์แรกไว้ที่อินพุตของวงจร *BYTECNT* ซึ่งก็คือ 0x3F
6. สัญญาณพัลส์ *TX_BLD* ลูกที่ 2 ทำการโหลดค่า 4 บิตบนของค่าเริ่มต้นในการนับมาไว้ที่อินพุตของเคาน์เตอร์ขนาด 12 บิต และโหลดข้อมูลไบต์ที่ 2 ซึ่งก็คือ 0xE8 มาไว้ที่อินพุตของวงจร *BYTECNT*

7. สัญญาณพัลส์ *TX_BLD* ลูกที่ 3 ทำการโหลดค่า 8 บิตล่างของค่าเริ่มต้นในการนับมาไว้ที่อินพุตของเคาน์เตอร์ขนาด 12 บิต
8. ไมโครโปรเซสเซอร์ส่งสัญญาณ *CNT_LDB* ที่มีสถานะเป็นลอจิก 0 เพื่อควบคุมการโหลดค่าเริ่มต้นในการนับไปไว้ในเคาน์เตอร์ขนาด 12 บิต
9. เมื่อไมโครโปรเซสเซอร์ส่งสัญญาณพัลส์ *LD_CLK* เคาน์เตอร์จะทำการโหลดค่าเริ่มต้นในการนับ ซึ่งในที่นี้ค่าเริ่มต้นในการนับคือ 0xFE8
10. สัญญาณ *TX_ENDB* เปลี่ยนสถานะเป็นลอจิก 1 เมื่อพบสัญญาณขาขึ้นแรกของสัญญาณนาฬิกา *TX_CLK* หลังจากทำการโหลดค่าเริ่มต้นในการนับ

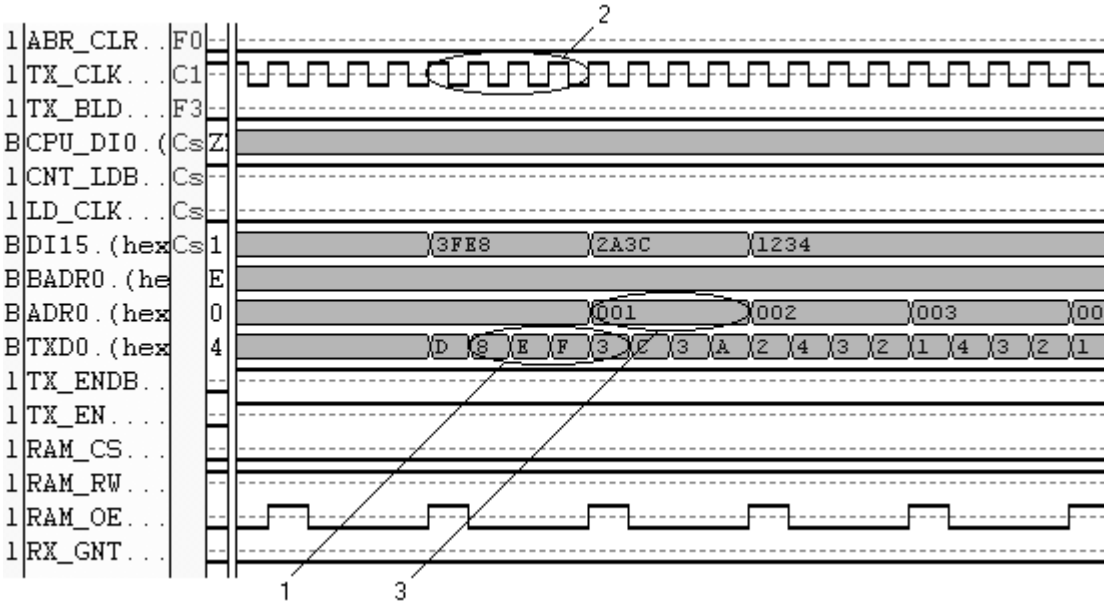


ภาพประกอบ 4-16 ผลการจำลองการทำงานของวงจรภาคส่ง (ต่อ)

1. หลังจากสัญญาณ *TX_ENDB* ซึ่งก็คือ *TX_REQ* มีสถานะเป็นลอจิก 1 และสัญญาณนาฬิกาผ่านไป 15 ลูก เมื่อสัญญาณนาฬิกาเปลี่ยนสถานะจาก 1 เป็น 0 สัญญาณ *RAM_CS* มีสถานะเป็นลอจิก 0 และสัญญาณ *RAM_RW* มีสถานะเป็นลอจิก 1 ส่วนสัญญาณ *RX_GNT* มีสถานะเป็นลอจิก 0 ซึ่งเมื่อสัญญาณ *RAM_CS* มีสถานะเป็นลอจิก 0 นั้น จะสามารถเข้าถึงหน่วยความจำได้ โดย *RAM_RW* มีสถานะเป็นลอจิก 1 หมายถึงการอ่านข้อมูลและควบคุมขาอีเนเบิลของ Tri-State Buffer ซึ่งควบคุมสถานะอินพุตหรือเอาท์พุตของบัสข้อมูลของหน่วยความจำหลักให้สามารถอ่าน

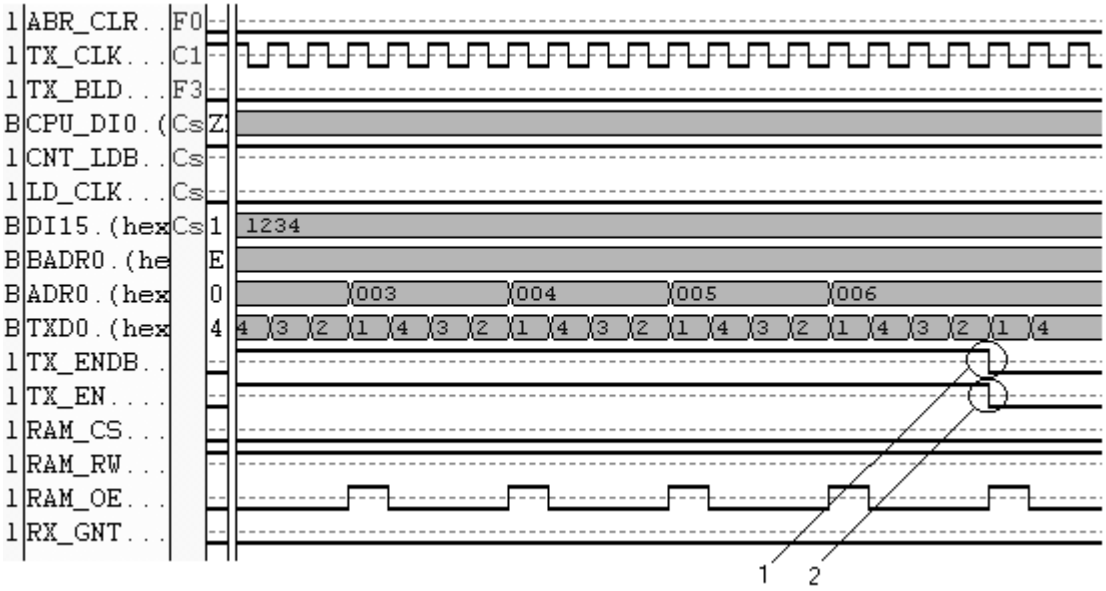
ข้อมูลเข้ามาได้ ส่วนสัญญาณ *RX_GNT* มีสถานะเป็นลอจิก 0 เป็นการส่งสัญญาณ *SEL* ไปควบคุมวงจร *CPU_REGS* ให้เลือกข้อมูล *Base Address* ของวงจรภาคส่ง

2. สัญญาณ *TX_EN* เปลี่ยนสถานะเป็นลอจิก 1 เพื่อบอกให้ทราบว่ามีการส่งข้อมูลใน *TXD[3:0]*
3. สัญญาณ Preamble และ SFD ถูกส่งออกทางสัญญาณข้อมูลของ MII



ภาพประกอบ 4-17 ผลการจำลองการทำงานของวงจรภาคส่ง (ต่อ)

1. ข้อมูลขนาด 16 บิตถูกแปลงเป็นสัญญาณอนุกรม *TXD[3:0]*
2. เมื่อสัญญาณนาฬิกา *TX_CLK* ผ่านไป 4 ลูก
3. พอยน์เตอร์จะปรับค่าเพิ่มขึ้น 1 ค่า



ภาพประกอบ 4-18 ผลการจำลองการทำงานของวงจรภาคส่ง (ต่อ)

1. เมื่อเคาน์เตอร์นับจำนวนข้อมูลที่ส่งออกจนครบแล้วสัญญาณ *TX_ENDB* จะเปลี่ยนสถานะจากลอจิก 1 เป็น 0
2. เมื่อสิ้นสุดการส่งข้อมูลสัญญาณ *TX_EN* เปลี่ยนสถานะจากลอจิก 1 เป็น 0

4.5 **รีจิสเตอร์ที่จำเป็นต้องเซ็ต**

การควบคุมให้เครื่องจัดการจราจรในระบบเครือข่ายทำงานอย่างถูกต้องนั้นรีจิสเตอร์ที่จำเป็นต้องเซ็ตมีดังนี้คือ

4.5.1 **รีจิสเตอร์** ***Base Address*** **ของวงจรภาครับ** ซึ่งจะเก็บค่าแอดเดรส 8 บิตบนของพอยน์เตอร์ชี้ตำแหน่งหน่วยความจำที่ใช้ระบุตำแหน่งที่จะจัดเก็บแพ็คเกตลงในหน่วยความจำหลัก การเซ็ตค่ารีจิสเตอร์ *Base Address* ของวงจรภาครับทำโดยการกำหนดค่าที่ขา *CPU_DI[7:0]* ของชิพเอฟพีจีเอ ค่าดังกล่าวจะถูกจัดเก็บไว้ในรีจิสเตอร์เมื่อเจอสัญญาณขาขึ้นที่ขา *RX_BLD* ของชิพเอฟพีจีเอ

4.5.2 **รีจิสเตอร์** ***Base Address*** **ของวงจรภาคส่ง** ซึ่งจะเก็บค่าแอดเดรส 8 บิตบนของพอยน์เตอร์ชี้ตำแหน่งหน่วยความจำที่ใช้ระบุตำแหน่งของแพ็คเกตที่ต้องการส่ง การเซ็ตค่ารีจิสเตอร์ *Base Address* ของวงจรภาคส่งทำโดยการกำหนดค่าที่ขา *CPU_DI[7:0]* ของชิพ

เอฟพีจีเอ ค่าดังกล่าวจะถูกจัดเก็บไว้ในรีจิสเตอร์เมื่อเจอสัญญาณขาขึ้นที่ขา *TX_BLD* ของชิพเอฟพีจีเอ

4.5.3 **รีจิสเตอร์ค่าเริ่มต้นในการนับ** ทำหน้าที่เก็บค่าเริ่มต้นในการนับ การเซ็ตค่ารีจิสเตอร์นี้จำเป็นต้องระบุเพื่อให้วงจรภาคส่งจัดส่งข้อมูลครบตามขนาดของแพ็คเกต ทำโดยการกำหนดค่าที่ขา *CPU_DI[7:0]* ของชิพเอฟพีจีเอ เมื่อป้อนสัญญาณนาฬิกาขาขึ้นที่ขา *TX_BLD* 3 ลูก ค่าเริ่มต้นในการนับจะถูกโหลดไปรอที่อินพุตของรีจิสเตอร์ การโหลดค่าไปเก็บในรีจิสเตอร์จะเกิดขึ้นเมื่อสัญญาณที่ขา *CNT_LDB* เป็นลอจิก 0 และมีสัญญาณพัลส์มาป้อนที่ *LD_CLK*

4.5.4 **รีจิสเตอร์ชนิดของแพ็คเกต** ทำหน้าที่เก็บค่าชนิดของแพ็คเกต ซึ่งเป็นข้อมูลที่ไมโครโปรเซสเซอร์ใช้ในการประมวลผลเพื่อจัดคิว การเซ็ตค่ารีจิสเตอร์นี้ทำโดยการส่งสัญญาณพัลส์มาที่ขา *CPU_RDC* จะทำให้ข้อมูลชนิดของแพ็คเกตถูกส่งไปที่บัสข้อมูลของไมโครโปรเซสเซอร์เพื่อรอให้ไมโครโปรเซสเซอร์อ่านค่าไปเก็บไว้ในรีจิสเตอร์ภายใน

4.5.5 **รีจิสเตอร์ไอพีแอดเดรสต้นทาง** ทำหน้าที่เก็บค่าไอพีแอดเดรสต้นทาง ซึ่งเป็นข้อมูลที่ไมโครโปรเซสเซอร์ใช้ในการประมวลผลเพื่อจัดคิว การเซ็ตค่ารีจิสเตอร์นี้ทำโดยการส่งสัญญาณพัลส์มาที่ขา *CPU_RDC* จะทำให้ข้อมูลไอพีแอดเดรสต้นทางถูกส่งไปที่บัสข้อมูลของไมโครโปรเซสเซอร์เพื่อรอให้ไมโครโปรเซสเซอร์อ่านค่าไปเก็บไว้ในรีจิสเตอร์ภายใน

4.5.6 **รีจิสเตอร์ไอพีแอดเดรสปลายทาง** ทำหน้าที่เก็บค่าไอพีแอดเดรสปลายทางซึ่งเป็นข้อมูลที่ไมโครโปรเซสเซอร์ใช้ในการประมวลผลเพื่อจัดคิว การเซ็ตค่ารีจิสเตอร์นี้ทำโดยการส่งสัญญาณพัลส์มาที่ขา *CPU_RDC* จะทำให้ข้อมูลไอพีแอดเดรสปลายทางถูกส่งไปที่บัสข้อมูลของไมโคร โปรเซสเซอร์เพื่อรอให้ไมโครโปรเซสเซอร์อ่านค่าไปเก็บไว้ในรีจิสเตอร์ภายใน

4.6 ขั้นตอนการทำงานของซอฟท์แวร์

หัวข้อนี้เป็นการแนะนำขั้นตอนการทำงานของซอฟท์แวร์สำหรับผู้ที่จะนำสถาปัตยกรรมนี้ไปใช้งานจริง ขั้นตอนการทำงานของซอฟท์แวร์มีดังต่อไปนี้คือ

4.6.1 รีเซ็ตวงจร *Arbiter* โดยการกำหนดให้สัญญาณ *ABR_CLR* มีสถานะเป็นลอจิก 1

4.6.2 กำหนดค่า *Base Address* ให้กับวงจรภาครับเพื่อระบุตำแหน่งเริ่มต้นในการจัดเก็บแพ็คเกต

4.6.3 ตรวจสอบว่ามีแพ็คเกตส่งเข้ามาหรือไม่ ถ้ามี ทำการค้างค่า *Base Address* โดยการกำหนดให้สัญญาณ *RX_BLD* เปลี่ยนสถานะจากลอจิก 0 เป็น 1

4.6.4 เมื่อจัดเก็บแพ็คเกตเสร็จสิ้น ทำการส่งสัญญาณ *CPU_RDC* เพื่ออ่านข้อมูลสำหรับประมวลผลเพื่อจัดคิวมาเก็บไว้ในหน่วยความจำของไมโครโปรเซสเซอร์

4.6.5 ตรวจสอบว่ามีแพ็คเกตที่ 2 ส่งเข้ามาหรือไม่ ถ้ามีจะทำการตามขั้นตอนในหัวข้อ 4.6.2-4.6.4 แต่ถ้าไม่มีก็จะทำการประมวลผลเพื่อจัดคิวแพ็คเกตที่จัดเก็บไว้ในหน่วยความจำหลัก

4.6.6 เมื่อว่างจากการรับแพ็คเกต รีเซ็ตวงจร *Arbiter* โดยการกำหนดให้สัญญาณ *ABR_CLR* มีสถานะเป็นลอจิก 1

4.6.7 กำหนดค่า *Base Address* ให้กับวงจรภาคส่งเพื่อระบุตำแหน่งเริ่มต้นของแพ็คเกตที่ต้องการส่ง

4.6.8 ค้างค่า *Base Address* โดยการกำหนดให้สัญญาณ *TX_BLD* เปลี่ยนสถานะจากลอจิก 0 เป็น 1

4.6.9 กำหนดค่าเริ่มต้นในการนับของแพ็คเกตที่ต้องการส่ง

4.6.10 ส่งสัญญาณ *TX_BLD* เพื่อทำการโหลดค่าเริ่มต้นในการนับมาไว้ที่อินพุตของเคาน์เตอร์ขนาด 12 บิต

4.6.11 ส่งสัญญาณ *CNT_LDB* เพื่อควบคุมการโหลดค่าเริ่มต้นในการนับไปไว้ในเคาน์เตอร์ขนาด 12 บิต

4.6.12 ส่งสัญญาณ *LD_CLK* เพื่อโหลดค่าเริ่มต้นในการนับไปไว้ในเคาน์เตอร์ขนาด 12 บิต

4.6.13 กลับไปตรวจสอบว่ามีแพ็คเกตส่งเข้ามาหรือไม่ ถ้ามีแล้วดำเนินการตามขั้นตอนที่ 4.6.1 แต่ถ้าไม่มีแพ็คเกตส่งเข้าก็จะไปดำเนินการตามขั้นตอนที่ 4.6.6

สำหรับบทนี้เป็นการนำเสนอการทำงานและส่วนประกอบของเครื่องจัดการจราจรในระบบเครือข่าย รวมถึงการออกแบบวงจรและผลการจำลองการทำงานของวงจรที่ได้ออกแบบไว้ และในหัวข้อสุดท้ายเป็นรีจิสเตอร์ที่จำเป็นต้องเซ็ตและขั้นตอนในการเขียนซอฟท์แวร์เพื่อใช้เป็นข้อมูลสำหรับผู้ที่จะนำสถาปัตยกรรมนี้ไปใช้งานต่อไป ส่วนบทถัดไปจะเป็นเปรียบเทียบผลการจำลองผลการอิมพลีเมนต์ชิพเอฟพีจีเอ และการสรุปผลการจำลองการทำงานของเครื่องจัดการจราจรในระบบเครือข่าย และท้ายสุดก็คือปัญหาที่พบและข้อเสนอแนะในวิทยานิพนธ์นี้